
Amazon DynamoDB

Guia do desenvolvedor

Versão da API 2012-08-10



Amazon DynamoDB: Guia do desenvolvedor

Copyright © Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigue a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, conectados ou patrocinados pela Amazon.

Table of Contents

O que é o Amazon DynamoDB?	1
Alta disponibilidade e durabilidade	1
Conceitos básicos do DynamoDB	1
Como ele funciona	2
Componentes principais	2
API do DynamoDB	10
Regras de nomeação e tipos de dados	13
Consistência de leituras	17
Modo de capacidade de leitura/gravação	18
Partições e distribuição de dados	23
Do SQL para o NoSQL	25
Relacional ou NoSQL?	26
Características dos bancos de dados	28
Criação de uma tabela	30
Obter informações sobre uma tabela	31
Gravar dados em uma tabela	32
Ler dados de uma tabela	35
Gerenciamento de índices	41
Modificação de dados em uma tabela	44
Exclusão de dados de uma tabela	46
Remoção de uma tabela	47
Recursos adicionais do Amazon DynamoDB	48
Guias, repositórios e publicações de blog	48
Modelagem de dados e padrões de design	48
Padrões de design avançados com Rick Houlihan	49
Cursos de treinamento	49
Ferramentas para codificação e visualização	49
Configuração do DynamoDB	50
Configuração do DynamoDB Local (versão disponível para download)	50
Implantar o	50
Observações sobre o uso	54
Configuração do DynamoDB (serviço Web)	57
Cadastrar-se no AWS	58
Obter umAWSChave de acesso	58
Configurar suas credenciais	59
Acessar o DynamoDB	60
Usar o console	60
Trabalhar com preferências do usuário	61
Como usar AWS CLI	62
Download e configuração da AWS CLI	62
Usar oAWS CLICom DynamoDB	62
Usar oAWS CLICom DynamoDB para download	64
Uso da API	64
Como usar o NoSQL Workbench	64
Intervalos de endereços IP	64
Conceitos básicos do DynamoDB	65
Conceitos básicos	65
Pré-requisitos	65
Etapa 1: Criar uma tabela	65
Etapa 2: Gravar dados	68
Etapa 3: Ler dados	70
Etapa 4: Atualizar dados	71
Etapa 5: Consultar dados	73
Etapa 6: Criar um índice secundário global	75

Etapa 7: Consultar o índice secundário global	78
Etapa 8 Limpar (opcional)	79
Próximas etapas	80
Conceitos básicos do AWSSDKs da	81
Java e DynamoDB	81
Pré-requisitos do tutorial	81
Etapa 1: Criar uma tabela	82
Etapa 2: Carregar dados de exemplo	83
Etapa 3: Criar, ler, atualizar e excluir um item	86
Etapa 4: Consultar e verificar os dados	95
Etapa 5: (Opcional) Excluir a tabela	99
Summary	100
JavaScript e DynamoDB	101
Pré-requisitos do tutorial	101
Etapa 1: Criar uma tabela	102
Etapa 2: Carregar dados de exemplo	103
Etapa 3: Criar, ler, ler, atualizar e excluir um item	106
Etapa 4: Consultar e verificar os dados	116
Etapa 5: (Optional): Excluir a tabela	121
Resumo	122
Node.js e DynamoDB	123
Pré-requisitos do tutorial	123
Etapa 1: Criar uma tabela	124
Etapa 2: Carregar dados de exemplo	125
Etapa 3: Criar, ler, atualizar e excluir um item	127
Etapa 4: Consulta e verificação de dados	135
Etapa 5: (Optional): Excluir a a tabela	139
Summary	140
.NET e DynamoDB	140
Pré-requisitos do tutorial	141
Etapa 1: Criar um cliente	142
Etapa 2: Criar uma tabela	144
Etapa 3: Carregar dados	146
Etapa 4: Adicionar um filme	148
Etapa 5: Ler um filme	149
Etapa 6: Atualizar o filme	150
Etapa 7: Exclusão de um item com condições	152
Etapa 8: Consultar uma tabela	154
Etapa 9: Verificação de uma tabela	156
Etapa 10: Excluir a tabela	157
PHP e DynamoDB	157
Pré-requisitos do tutorial	158
Etapa 1: Criar uma tabela	158
Etapa 2: Carregar dados de exemplo	160
Etapa 3: Criar, ler, atualizar e excluir um item	162
Etapa 4: Consultar e verificar os dados	171
Etapa 5: (Opcional) Excluir a tabela	176
Summary	177
Python e DynamoDB	178
Pré-requisitos do tutorial	178
Etapa 1: Criar uma tabela	179
Etapa 2: Carregar dados de exemplo	180
Etapa 3: Operações de CRUD	182
Etapa 4: Consultar e verificar os dados	188
Etapa 5: (Optional) Excluir a tabela	191
Summary	192
Ruby e DynamoDB	192

Pré-requisitos do tutorial	192
Etapa 1: Criar uma tabela	193
Etapa 2: Carregar dados de exemplo	194
Etapa 3: Criar, ler, atualizar e excluir um item	196
Etapa 4: Consultar e verificar os dados	204
Etapa 5: (Opcional) Excluir a tabela	209
Summary	210
Programação com o DynamoDB	211
Visão geral do AWSSupport do SDK da para o DynamoDB	211
Interfaces programáticas	213
Interfaces de baixo nível	214
Interfaces de documento	215
Interface de persistência de objetos	215
API de baixo nível	217
Formato de solicitação	219
Formato de resposta	219
Descritores de tipo de dados	220
Dados numéricos	221
Dados binários	221
Como tratar erros	221
Componentes de erros	222
Mensagens e códigos de erro	222
Tratamento de erros no seu aplicativo	225
Repetições de erro e recuo exponencial	226
Operações em lote e tratamento de erros	227
Interfaces de programação de nível superior para o DynamoDB	227
Java: DynamoDBMapper	228
.NET: Modelo de documento	276
.NET: Modelo de persistência de objeto	299
Executar exemplos de código	328
Carregar dados de exemplo	329
Exemplos de código Java	334
Exemplos de código .NET	336
Trabalhando com o DynamoDB	339
Trabalho com tabelas	339
Operações básicas nas tabelas do	340
Considerações ao mudar o modo de capacidade de leitura/gravação	344
Tabelas de capacidade provisionada	345
Tamanhos e formatos de item	349
Como gerenciar a capacidade de throughput com o Auto Scaling	350
Trabalhar com tabelas globais	365
Marcar recursos do	388
Trabalho com tabelas: Java	392
Trabalho com tabelas: .NET	397
Trabalho com itens	404
Leitura de um item	405
Gravação de um item	406
Valores de retorno	407
Operações em lote	408
Contadores atômicos	410
Gravações condicionais	410
Usar expressões	414
Expiração de itens com a vida útil	439
Trabalho com itens: Java	446
Trabalho com itens: .NET	466
Como trabalhar com consultas	490
Expressão de condição principal	490

Expressões de filtro para consulta	492
Limitação do número de itens no conjunto de resultados	493
Paginação do resultados da consulta	493
Contagem dos itens nos resultados	495
Unidades de capacidade consumidas por Query	495
Consistência de leitura de query	496
Consulta:: Java	496
Consulta:: .NET	502
Como trabalhar com verificações	508
Expressões de filtro de Scan	509
Limitação do número de itens no conjunto de resultados	509
Paginação de resultados	509
Contagem dos itens nos resultados	511
Unidades de capacidade consumidas por Scan	511
Consistência de leitura de Scan	512
Scan em paralelo	512
Verificação: Java	514
Verificação: .NET	521
Trabalhando com a linguagem de consulta PartiQL	529
O que é o PartiQL?	530
PartiQL no Amazon DynamoDB	530
Conceitos básicos	530
Tipos de dados	536
Declarações	538
Funções	545
Operadores	549
Transações	550
Operações em lote	553
Políticas do IAM	556
Trabalho com índices	559
Índices secundários globais	562
Índices secundários locais	604
Trabalho com streams	641
Opções	642
Como trabalhar com Kinesis Data Streams	643
Como trabalhar com DynamoDB Streams	651
Trabalhar com Transações	680
Como ele funciona	681
Uso de IAM com Transações	686
Código de exemplo	688
Como trabalhar com backups	691
Backup e restauração sob demanda	691
Recuperação point-in-time	706
Aceleração na memória com o DAX	713
Casos de uso para o DAX	714
Observações de uso do DAX	714
Como ele funciona	715
Como o DAX processa solicitações	716
Cache de itens	718
Cache de consultas	718
Componentes de cluster DAX	719
Nós	719
Clusters	719
Regiões e zonas de disponibilidade	720
Grupos de parâmetros	721
Security Groups (Grupos de segurança)	721
ARN do cluster	721

Endpoint de cluster	721
Endpoints de nó	722
Grupos de sub-redes	722
Eventos	722
Janela de manutenção	722
Criar um cluster DAX	723
Criar uma função de serviço do IAM para que o DAX acesse o DynamoDB	723
Usar a AWS CLI	725
Usar o console	729
Modelos de consistência	732
Consistência entre nós de cluster do DAX	732
Comportamento do cache de itens do	732
Comportamento do cache de consultas do	735
Leituras fortemente consistentes e transacionais	735
Armazenamento em cache negativo	736
Estratégias para gravações	736
Desenvolver com o cliente do DAX	738
Tutorial: Executar um aplicativo de exemplo	739
Como modificar um aplicativo existente para usar o DAX	776
Consultar índices secundários globais	779
Gerenciando clusters DAX	782
Permissões do IAM para gerenciar um cluster DAX	782
Escalabilidade de um cluster DAX	784
Personalização das configurações do cluster DAX	785
Configuração de definições de TTL	786
Support à marcação para DAX	787
Integração do AWS CloudTrail	787
Excluindo um cluster DAX	788
Monitoramento DAX	788
Ferramentas de monitoramento	789
Monitorar com o CloudWatch	790
Registrar operações do DAXAWS CloudTrail	803
Instâncias intermitentes DAX T3/T2	803
Família de instâncias DAX T2	804
Família de instâncias do DAX T3	804
Controle de acesso DAX	805
Função de serviço do IAM para DAX	805
Política do IAM para permitir o acesso ao cluster DAX	806
Estudo de caso: Acesso ao DynamoDB e ao DAX	807
Acesso ao DynamoDB, mas sem acesso com DAX	808
Acesso ao DynamoDB e ao DAX	809
Acesso ao DynamoDB via DAX, mas sem acesso direto ao DynamoDB	813
Criptografia DAX em repouso	815
Habilitar a criptografia em repouso usando o AWS Management Console	816
Criptografia do DAX em trânsito	817
Uso de funções vinculadas ao serviço do DAX	817
Permissões de função vinculada ao serviço para o DAX	818
Criar uma função vinculada ao serviço para o DAX	819
Editar uma função vinculada ao serviço do DAX	819
Exclusão de uma função vinculada ao serviço do DAX	819
Acessando DAX entreAWSContas	820
Configurar o IAM	821
Configuração de um VPC	823
Modificar o cliente do DAX para permitir acesso entre contas	824
Guia de dimensionamento de cluster do DAX	827
Overview	828
Estimativa do tráfego	828

Testes de carregamento	829
Referência de API	829
NoSQL Workbench	830
Baixar	830
Modelador de dados	831
Criar um novo modelo	831
Importar um modelo existente	838
Exportar um modelo	841
Editar um modelo existente	843
Visualizador de dados	845
Adicionar dados de exemplo	846
Facetas	847
Exibição agregada	848
Confirmar um modelo de dados	849
Criador de operações	853
Explorar conjuntos de dados	853
Criar operações	857
Exemplos de modelos de dados	868
Modelo de dados do funcionário	869
Modelo de dados do fórum de discussão	869
Modelo de dados da biblioteca de músicas	869
Modelo de dados da estância de esqui	870
Modelo de dados de ofertas de cartão de crédito	870
Modelo de dados de favoritos	870
Histórico de versões	871
Segurança	873
Proteção de dados	873
Criptografia em repouso	874
Proteção de dados no DAX	882
Privacidade do tráfego entre redes	883
Identity and Access Management	883
Identity and Access Management	883
Identity and Access Management no DAX	915
Validação de conformidade	915
Resiliência	916
Segurança da infraestrutura	916
Uso de VPC Endpoints	917
Análise de configuração e vulnerabilidade	923
Melhores práticas de segurança	923
Melhores práticas de segurança preventiva	923
Melhores práticas de segurança preventiva	925
Monitoramento	928
Registro em log e monitoramento no DynamoDB	928
Ferramentas de monitoramento	929
Monitorar com o Amazon CloudWatch	930
Registro de operações do DynamoDB usando oAWS CloudTrail	954
Registro em log e monitoramento no DAX	970
Contributor Insights	970
Como ele funciona	971
Conceitos básicos	975
Uso de IAM	979
Melhores práticas	983
Design do NoSQL	984
NoSQL vs. RDBMS	984
Dois conceitos principais	985
Abordagem geral	985
Design de chave de partição	986

Capacidade de intermitência	986
Capacidade adaptável	987
Distribuição das cargas de trabalho	988
Estilhaçamento de gravação	989
Upload de dados com eficiência	990
Design de chaves de classificação	991
Controle de versões	992
Índices secundários	993
Diretrizes gerais	993
Índices esparsos	995
Agregação	996
Sobrecarga de GSI	997
Estilhaçamento de GSI	998
Criar uma réplica	999
Itens grandes	1000
Compactação	1000
Uso do Amazon S3	1001
Dados de séries temporais	1001
Padrão de design para dados de séries temporais	1001
Exemplos de tabelas de séries temporais	1002
Relações muitos para muitos	1002
Listas de adjacências	1002
Gráficos materializados	1003
DynamoDB híbrido — RDBMs	1006
Sem migração	1006
Implementação do sistema híbrido	1007
Modelagem relacional	1008
Primeiras etapas	1011
Exemplo	1012
Consulta e verificação	1014
Desempenho de Scan	1014
Evitar picos	1014
Verificações paralelas	1016
Integração do com outrosAWSServiços	1018
Integração com o Amazon Cognito	1018
Integração com o Amazon Redshift	1020
Integração com o Amazon EMR	1021
Overview	1021
Tutorial: Trabalho com o Amazon DynamoDB e o Apache Hive	1022
Criação de uma tabela externa no Hive	1028
Processamento de instruções HiveQL	1030
Consulta de dados no DynamoDB	1031
Cópia de dados para e do Amazon DynamoDB	1033
Ajuste de desempenho	1043
Exportar para o Amazon S3	1046
Como funcionam	1047
Solicitar uma exportação	1048
Exportar saída	1051
Utilização de exportações com outros serviços	1054
Cotas	1056
Modo de capacidade de leitura/gravação e taxa de transferência	1056
Tamanhos de unidade de capacidade (para tabelas provisionadas)	1056
Tamanhos de unidade de solicitação (para tabelas sob demanda)	1057
Cotas padrão de taxa de transferência	1057
Aumentar ou diminuir a taxa de transferência (para tabelas provisionadas)	1058
Tables	1058
Tamanho da tabela	1058

Tabelas por conta	1059
Tabelas globais	1059
Índices secundários	1059
Índices secundários por tabela	1059
Atributos de índice secundário projetados por tabela	1059
Chaves de partição e chaves de classificação	1060
Tamanho da chave de partição	1060
Valores de chave de partição	1060
Tamanho da chave de classificação	1060
Valores de chave de classificação	1060
Regras de nomenclatura	1060
Nomes de tabela e nomes de índice secundário	1060
Nomes de atributo	1060
Tipos de dados	1061
String	1061
Number	1061
Binary	1061
Items	1061
Tamanho do item	1061
Tamanho do Item para Tabelas com Índices Secundários Locais	1062
Attributes	1062
Pares de nome-valor de atributo por item	1062
Número de valores em lista, mapa ou conjunto	1062
Valores de atributo	1062
Profundidade de atributo aninhado	1062
Parâmetros de expressão	1062
Lengths	1062
Operadores e operandos	1063
Palavras reservadas	1063
Transações do DynamoDB	1063
DynamoDB Streams	1063
Leitores simultâneos de um estilhço em DynamoDB Streams	1063
Capacidade de gravação máxima para uma tabela com um fluxo habilitado	1063
DynamoDB Accelerator (DAX)	1064
Disponibilidade de regiões do AWS	1064
Nodes	1064
Grupos de parâmetros	1064
Grupos de sub-redes	1064
Limites específicos de API	1064
DynamoDB Encryption em repouso	1065
Exportação de tabela para o Amazon S3	1066
Referência de API	1067
Apêndice	1068
Solução de problemas de estabelecimento de conexão SSL/TLS	1068
Como testar seu aplicativo ou serviço	1068
Como testar o navegador cliente	1069
Como atualizar o aplicativo de software cliente	1069
Como atualizar o navegador cliente	1069
Como atualizar manualmente seu pacote de certificado	1069
Tabelas e dados de exemplo	1070
Arquivos de dados de exemplo	1071
Criar tabelas de exemplo e carregar dados	1079
Como criar tabelas de exemplo e carregar dados - Java	1080
Como criar tabelas de exemplo e carregar dados - .NET	1086
Exemplo de aplicativo do usandoAWS SDK for Python (Boto3)	1095
Etapa 1: Implantar e testar localmente	1096
Etapa 2: Analise o modelo de dados e os detalhes da implantação	1100

Etapa 3: Implantar em produção	1107
Etapa 4: Liberação de recursos	1114
Integração com o AWS Data Pipeline	1114
Pré-requisitos para exportar e importar dados	1117
Exportação de dados do DynamoDB para o Amazon S3	1122
Importação de dados do Amazon S3 para o DynamoDB	1123
Troubleshooting	1124
Modelos predefinidos para AWS Data Pipelinee DynamoDB	1125
Back-end de armazenamento do Amazon DynamoDB para Titan	1125
Palavras reservadas no DynamoDB	1125
Parâmetros condicionais herdados	1134
AttributesToGet	1135
AttributeUpdates	1136
ConditionalOperator	1138
Expected	1138
KeyConditions	1141
QueryFilter	1143
ScanFilter	1145
Criação de condições com parâmetros herdados	1146
Versão anterior da API de baixo nível (2011-12-05)	1152
Batch.GetItem	1152
BatchWriteItem	1157
CreateTable	1162
DeleteItem	1167
DeleteTable	1171
DescribeTables	1174
GetItem	1177
ListTables	1180
PutItem	1182
Query	1187
Scan	1196
UpdateItem	1208
UpdateTable	1214
Histórico do documento	1218
Atualizações anteriores	1222
	mccxliv

O que é o Amazon DynamoDB?

Bem-vindo ao Guia do desenvolvedor do Amazon DynamoDB.

O Amazon DynamoDB é um serviço de banco de dados NoSQL totalmente gerenciado que fornece um desempenho rápido e previsível com escalabilidade integrada. O DynamoDB permite que você transfira os encargos administrativos de operação e escalabilidade de um banco de dados distribuído. Assim, você não precisa se preocupar com provisionamento, instalação e configuração de hardware, replicação, correção de software nem escalabilidade de cluster. Além disso, o DynamoDB oferece criptografia em repouso, o que elimina a carga e a complexidade operacionais envolvidas na proteção de dados confidenciais. Para mais informações, consulte [Criptografia do DynamoDB em repouso \(p. 874\)](#).

Com o DynamoDB, você pode criar tabelas de banco de dados que armazenam e recuperam qualquer quantidade de dados e atendem a todos os níveis de tráfego solicitados. Você pode aumentar ou diminuir a capacidade de throughput das tabelas sem tempo de inatividade ou degradação do desempenho. Você pode usar o AWS Management Console para monitorar a utilização de recursos e as métricas de desempenho.

O DynamoDB oferece o recurso de backup sob demanda. Ele permite que você crie backups completos das suas tabelas para retenção e arquivamento de longo prazo de modo a atender às necessidades de conformidade regulamentar. Para mais informações, consulte [Backup e restauração sob demanda para o DynamoDB \(p. 691\)](#).

Você pode criar backups sob demanda e habilitar a recuperação point-in-time para as tabelas do Amazon DynamoDB. A recuperação point-in-time ajuda a proteger as tabelas de operações acidentais de gravação ou exclusão. Com a recuperação point-in-time, você pode recuperar uma tabela para qualquer ponto durante os últimos 35 dias. Para mais informações, consulte [Recuperação point-in-time: Como ele funciona \(p. 706\)](#).

O DynamoDB permite a exclusão automática dos itens expirados das tabelas, para ajudar você a reduzir o uso e o custo do armazenamento que não são mais relevantes. Para mais informações, consulte [Itens expirando usando o Time to Live \(TL — Time to Live — uso do DynamoDB \(p. 439\)](#).

Alta disponibilidade e durabilidade

O DynamoDB distribui automaticamente os dados e o tráfego para as tabelas, entre um número suficiente de servidores, para lidar com seus requisitos de throughput e armazenamento, sem deixar de manter um desempenho consistente e rápido. Todos os dados são armazenados em discos de estado sólido (SSDs) e automaticamente replicados em várias zonas de disponibilidade em um AWS Região, fornecendo alta disponibilidade integrada e durabilidade de dados. Você pode usar tabelas globais para manter as tabelas do DynamoDB sincronizadas em AWS Regiões. Para mais informações, consulte [Tabelas globais Replicação em várias regiões com o DynamoDB \(p. 365\)](#).

Conceitos básicos do DynamoDB

Recomendamos que você comece lendo as seguintes seções:

- [Amazon DynamoDB: Como ele funciona \(p. 2\)](#)—Para aprender conceitos essenciais do DynamoDB.
- [Configuração do DynamoDB \(p. 50\)](#)—Para saber como configurar o DynamoDB (a versão disponível para download ou o serviço web).

- [Acessar o DynamoDB \(p. 60\)](#)—Para saber como acessar o DynamoDB usando o console, AWS CLI ou API.

Para começar a trabalhar rapidamente com o DynamoDB, consulte [Conceitos básicos do DynamoDB e do AWSSDKs da \(p. 81\)](#).

Para saber mais sobre o desenvolvimento de aplicativos, consulte o seguinte:

- [Programação com o DynamoDB e o AWSSDKs da \(p. 211\)](#)
- [Trabalhar com tabelas, itens, consultas, varreduras e índices \(p. 339\)](#)

Para localizar rapidamente as recomendações para maximizar o desempenho e minimizar os custos de throughput, consulte [Práticas recomendadas para projetar e arquitetar com o DynamoDB \(p. 983\)](#). Para saber como marcar recursos do DynamoDB, consulte [Adicionar tags e rótulos a recursos \(p. 388\)](#).

Para obter melhores práticas, guias de instruções para execução e ferramentas, consulte [Recursos do Amazon DynamoDB](#).

Você pode usar o AWS Database Migration Service(AWS DMS) para migrar dados de um banco de dados relacional ou MongoDB para uma tabela do DynamoDB. Para obter mais informações, consulte o [Guia do usuário do AWS Database Migration Service](#).

Para saber como usar o MongoDB como uma origem de migração, consulte [Uso do MongoDB como origem para o AWS Database Migration Service](#). Para saber como usar o DynamoDB como um destino de migração, consulte [Uso de um banco de dados do Amazon DynamoDB como destino do AWS Database Migration Service](#).

Amazon DynamoDB: Como ele funciona

As seções a seguir fornecem uma visão geral dos componentes do serviço Amazon DynamoDB e de como eles interagem.

Depois de ler essa introdução, tente trabalhar com a seção [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#), que o orienta pelo processo de criar tabelas de exemplo, carregar dados e realizar algumas operações básicas de banco de dados.

Para tutoriais específicos de idioma com código de exemplo, consulte [Conceitos básicos do DynamoDB e do AWSSDKs da \(p. 81\)](#).

Tópicos

- [Componentes principais do Amazon DynamoDB \(p. 2\)](#)
- [API do DynamoDB \(p. 10\)](#)
- [Regras de nomeação e tipos de dados \(p. 13\)](#)
- [Consistência de leituras \(p. 17\)](#)
- [Modo de capacidade de leitura/gravação \(p. 18\)](#)
- [Partições e distribuição de dados \(p. 23\)](#)

Componentes principais do Amazon DynamoDB

No DynamoDB, tabelas, itens e atributos são os componentes principais com que você trabalha. A Tabela é uma coleção de itens, sendo cada item uma coleção de atributos. O DynamoDB usa

chaves primárias para identificar exclusivamente cada item em uma tabela e índices secundários para fornecer mais flexibilidade de consulta. Você pode usar o DynamoDB Streams para capturar eventos de modificação de dados em tabelas do DynamoDB.

Há limites no DynamoDB. Para mais informações, consulte [Cotas de serviço, conta e tabela no Amazon DynamoDB \(p. 1056\)](#).

Tópicos

- [Tabelas, itens e atributos \(p. 3\)](#)
- [Chave primária \(p. 6\)](#)
- [Índices secundários \(p. 7\)](#)
- [DynamoDB Streams \(p. 9\)](#)

Tabelas, itens e atributos

Estes são os componentes básicos do DynamoDB:

- **Tabelas**— semelhante a outros sistemas de banco de dados, o DynamoDB armazena dados em tabelas. Uma tabela é uma coleção de dados. Por exemplo, consulte a tabela de exemplo People que você pode usar para armazenar informações pessoais de contato de amigos, familiares ou qualquer outra pessoa de interesse. Você também pode ter uma tabela Cars para armazenar informações sobre os veículos que as pessoas dirigem.
- **Itens**— Cada tabela contém zero ou mais itens. Um item é um grupo de atributos identificável exclusivamente entre todos os outros itens. Na tabela People de exemplo, cada item representa uma pessoa. Na tabela Cars, cada item representa um veículo. Os itens no DynamoDB são semelhantes de muitas formas a linhas, registros ou tuplas em outros sistemas de banco de dados. No DynamoDB, não há limite para o número de itens que você pode armazenar em uma tabela.
- **Atributos**.— Cada item é composto de um ou mais atributos. Um atributo é um elemento de dados fundamental, algo que não precisa ser dividido ainda mais. Por exemplo, um item na tabela People contém os atributos PersonID, LastName, FirstName etc. Em uma tabela Department, um item pode ter atributos como DepartmentID, Name, Manager etc. Atributos no DynamoDB se parecem de várias maneiras a campos ou colunas em outros sistemas de banco de dados.

O diagrama a seguir mostra uma tabela People com alguns itens e atributos de exemplo.

People

```
{  
    "PersonID": 101,  
    "LastName": "Smith",  
    "FirstName": "Fred",  
    "Phone": "555-4321"  
}
```

```
{  
    "PersonID": 102,  
    "LastName": "Jones",  
    "FirstName": "Mary",  
    "Address": {  
        "Street": "123 Main",  
        "City": "Anytown",  
        "State": "OH",  
        "ZIPCode": 12345  
    }  
}
```

```
{  
    "PersonID": 103,  
    "LastName": "Stephens",  
    "FirstName": "Howard",  
    "Address": {  
        "Street": "123 Main",  
        "City": "London",  
        "PostalCode": "ER3 5K8"  
    },  
    "FavoriteColor": "Blue"  
}
```

Observe o seguinte sobre a tabela People:

- Cada item da tabela tem um identificador exclusivo, ou chave primária, que o distingue de todos os outros na tabela. Na tabela People, a chave primária consiste em um único atributo (PersonID).
- Além da chave primária, a tabela People não tem esquema, o que significa que não é necessário definir seus atributos e tipos de dados previamente. Cada item pode ter seus próprios atributos distintos.
- A maioria dos atributos é escalar, o que significa que podem ter apenas um valor. Strings e números são exemplos comuns de escalares.

- Alguns dos itens têm um atributo aninhado (Endereço). O DynamoDB é compatível com atributos aninhados de até 32 níveis de profundidade.

Veja a seguir outra tabela de exemplo chamada Music, que você pode usar para controlar sua coleção de músicas.

Music

{ "Artist": "No One You Know", "SongTitle": "My Dog Spot", "AlbumTitle": "Hey Now", "Price": 1.98, "Genre": "Country", "CriticRating": 8.4 }	{ "Artist": "No One You Know", "SongTitle": "Somewhere Down The Road", "AlbumTitle": "Somewhat Famous", "Genre": "Country", "CriticRating": 8.4, "Year": 1984 }	{ "Artist": "The Acme Band", "SongTitle": "Still in Love", "AlbumTitle": "The Buck Starts Here", "Price": 2.47, "Genre": "Rock", "PromotionInfo": { "RadioStationsPlaying": ["KHCN", "KQBX", "WTNR", "WJJH"], "TourDates": { "Seattle": "20150625", "Cleveland": "20150630" }, "Rotation": "Heavy" } }	{ "Artist": "The Acme Band", "SongTitle": "Look Out, World", "AlbumTitle": "The Buck Starts Here", "Price": 0.99, "Genre": "Rock" }
---	--	---	---

Observe o seguinte sobre a tabela Music:

- A chave primária da tabela Music é composta por dois atributos (Artist e SongTitle). Cada item da tabela deve ter esses dois atributos. A combinação de Artist e SongTitle distingue cada item da tabela de todos os outros.

- Além da chave primária, a tabela Music não tem esquema, o que significa que nem os atributos nem seus tipos de dados precisam ser definidos previamente. Cada item pode ter seus próprios atributos distintos.
- Um dos itens tem um atributo aninhado (PromotionInfo), que contém outros atributos aninhados. O DynamoDB é compatível com atributos aninhados de até 32 níveis de profundidade.

Para mais informações, consulte [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#).

Chave primária

Ao criar uma tabela, além do nome dela, você deve especificar a chave primária da tabela. A chave primária identifica exclusivamente cada item na tabela, de modo que não possa haver dois itens com a mesma chave.

O DynamoDB é compatível com dois tipos diferentes de chaves primárias:

- Chave de partição – uma chave primária simples, formada por um atributo conhecido como a chave de partição.

O DynamoDB usa o valor da chave de partição como entrada para uma função de hash interna. A saída da função de hash determina a partição (armazenamento físico interno do DynamoDB) em que o item será armazenado.

Em uma tabela que possui somente uma chave de partição, dois itens não podem ter o mesmo valor de chave de partição.

A tabela People descrita em [Tabelas, itens e atributos \(p. 3\)](#) é um exemplo de uma tabela com uma chave primária simples (PersonID). Você pode acessar diretamente qualquer item na tabela People fornecendo o valor PersonId desse item.

- Chave de partição e chave de classificação— Referido como uma chave primária composta, esse tipo de chave é composto de dois atributos. O primeiro atributo é a chave de partição, e o segundo atributo é a chave de classificação.

O DynamoDB usa o valor da chave de partição como entrada para uma função de hash interna. A saída da função de hash determina a partição (armazenamento físico interno do DynamoDB) em que o item será armazenado. Todos os itens com o mesmo valor de chave de partição são armazenados juntos, na ordem classificada por valor de chave de classificação.

Em uma tabela que tenha uma chave de partição e uma chave de classificação, dois itens podem ter o mesmo valor de chave de partição. No entanto, esses dois itens devem ter valores de chave de classificação diferentes.

A tabela Music descrita em [Tabelas, itens e atributos \(p. 3\)](#) é um exemplo de uma tabela com chave primária composta (Artist e SongTitle). Você poderá acessar diretamente qualquer item da tabela Music, se fornecer os valores Artist e SongTitle desse item.

Uma chave primária composta oferece flexibilidade adicional ao consultar dados. Por exemplo, se você fornecer somente o valor deArtista, o DynamoDB recuperará todas as músicas desse artista. Para recuperar apenas um subconjunto de músicas de um determinado artista, você pode fornecer um valor para Artist com um intervalo de valores de SongTitle.

Note

A chave de partição de um item também é conhecida como seu atributo de hash. O termo atributo de hash ou hashO deriva do uso de uma função de hash interna no DynamoDB que distribui uniformemente os itens de dados entre partições, com base em seus valores de chave de partição.

A chave de classificação de um item também é conhecida como seu atributo de intervalo. O termoatributo de intervaloO deriva da forma como o DynamoDB armazena itens fisicamente próximos com a mesma chave de partição, por ordem do valor da chave de classificação.

Cada atributo de chave primária deve ser um escalar (o que significa que ele só pode conter um valor). Os únicos tipos de dados permitidos para atributos de chave primária são string, número ou binário. Essas restrições não existem para outros atributos não relacionados a chaves.

Índices secundários

É possível criar um ou mais índices secundários em uma tabela. Aíndice secundárioO permite consultar os dados na tabela usando uma chave alternativa, além de fazer consultas em relação à chave primária. O DynamoDB não exige que você use índices, mas eles conferem aos aplicativos mais flexibilidade durante a consulta dos dados. Depois de criar um índice secundário em uma tabela, você pode ler os dados do índice da mesma maneira que faz na tabela.

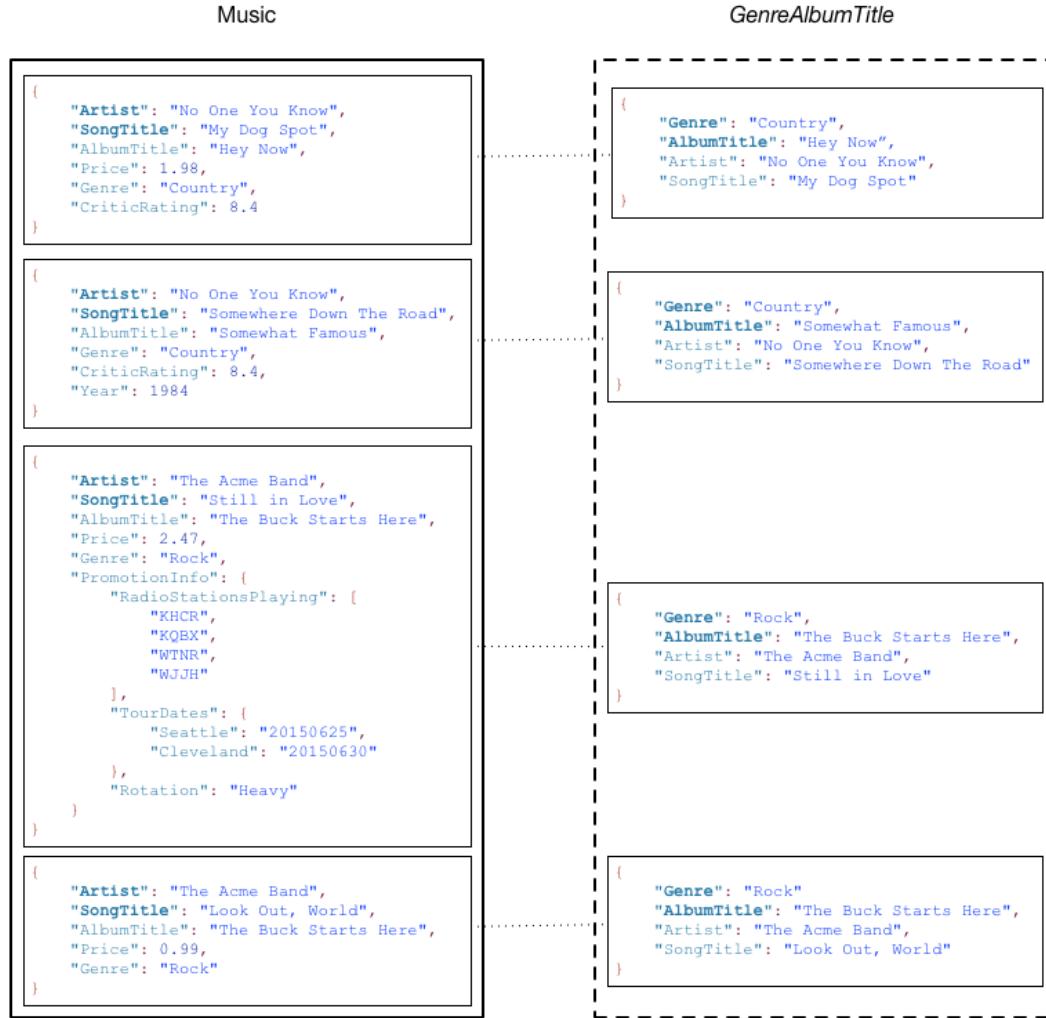
O DynamoDB é compatível com dois tipos de índices:

- Índice secundário global — Um índice com uma chave de partição e uma chave de classificação que podem ser diferentes das contidas na tabela.
- Índice secundário local — Um índice que possui a mesma chave de partição da tabela, mas uma chave de classificação diferente.

Cada tabela no DynamoDB tem uma cota de 20 índices secundários globais (cota padrão) e 5 índices secundários globais por tabela.

Na tabela Music de exemplo mostrada anteriormente, é possível consultar itens de dados por Artist (chave de partição) ou por Artist e SongTitle (chave de partição e chave de classificação). E se você também quiser consultar os dados por Genre e AlbumTitle? Para fazer isso, você pode criar um índice em Genre e AlbumTitle e, em seguida, consultar esse índice da mesma forma como consulta a tabela Music.

O diagrama a seguir mostra a tabela Music de exemplo, com um novo índice chamado GenreAlbumTitle. No índice, Genre é a chave de partição e AlbumTitle é a chave de classificação.



Observe o seguinte sobre o índice **GenreAlbumTitle**:

- Cada índice pertence a uma tabela, que é chamada de tabela base para o índice. No exemplo anterior, **Music** é a tabela base do índice **GenreAlbumTitle**.
- O DynamoDB mantém índices automaticamente. Quando você adiciona, atualiza ou exclui um item da tabela base, o DynamoDB adiciona, atualiza ou exclui o item correspondente em quaisquer índices que pertençam a essa tabela.
- Ao criar um índice, você especifica quais atributos serão copiados, ou projetados, da tabela base para o índice. No mínimo, os projetos do DynamoDB projetarão os atributos de chave da tabela base no índice. Este é o caso com **GenreAlbumTitle**, em que somente os atributos de chave da tabela **Music** são projetados no índice.

Você pode consultar o índice **GenreAlbumTitle** para localizar todos os álbuns de um gênero específico (por exemplo, todos os álbuns de Rock). Você também pode consultar o índice para localizar todos os álbuns

de um gênero específico que tenham determinados títulos de álbum (por exemplo, todos os álbuns Country com títulos que começam com a letra H).

Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

DynamoDB Streams

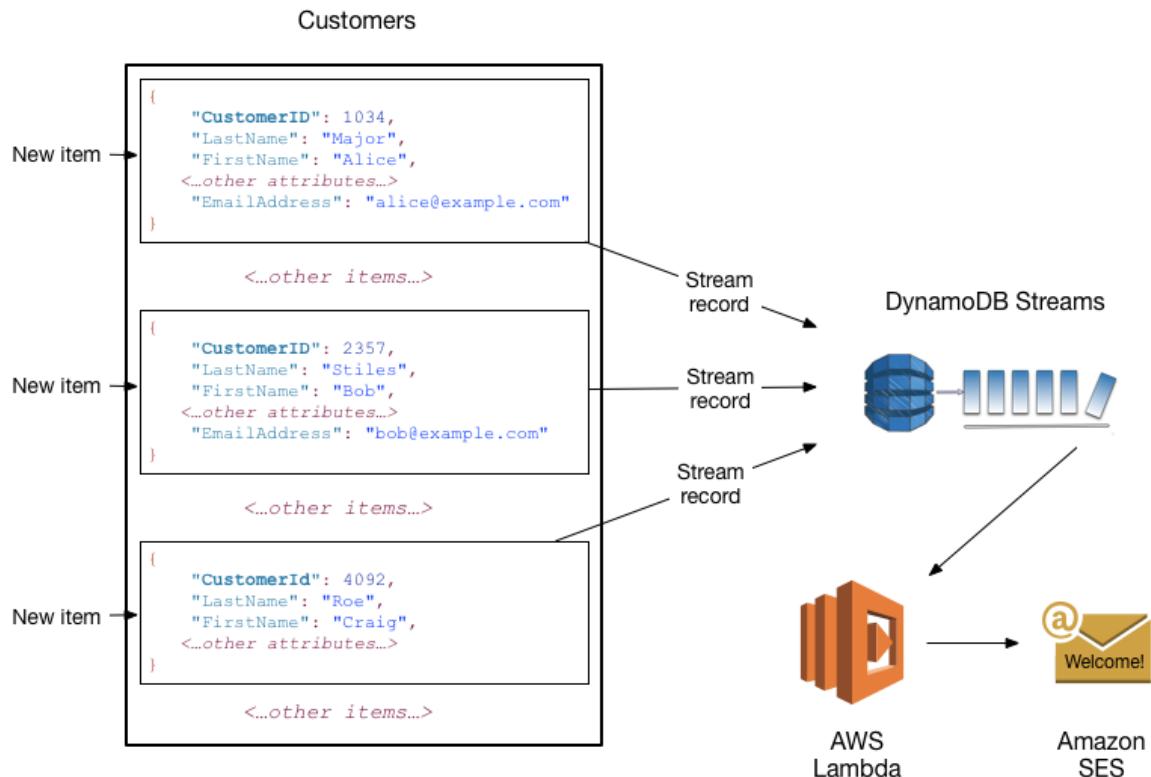
O DynamoDB Streams é um recurso opcional que captura eventos de modificação de dados em tabelas do DynamoDB. Os dados sobre esses eventos são exibidos no fluxo quase em tempo real e na ordem em que os eventos ocorreram.

Cada evento é representado por um registro de streaming. Se você habilitar um fluxo em uma tabela, o DynamoDB Streams gravará um registro de stream sempre que ocorrer um dos seguintes eventos:

- Um novo item é adicionado à tabela: O streaming captura uma imagem de todo o item, incluindo todos os seus atributos.
- Um item é atualizado: O stream captura as imagens “antes” e “depois” de quaisquer atributos que tenham sido modificados no item.
- Um item é excluído da tabela: O streaming captura uma imagem de todo o item antes de ter sido excluído.

Cada registro de stream também contém o nome da tabela, o carimbo de data/hora do evento e outros metadados. Registros de stream têm um tempo de vida de 24 horas; depois disso, eles são automaticamente removidos do fluxo.

Você pode usar o DynamoDB Streams junto com o AWS Lambda para criar um trigger que é executado automaticamente sempre que um evento de interesse aparece em um stream. Por exemplo, considere uma tabela `Customers` que contém informações sobre os clientes de uma empresa. Suponhamos que você queira enviar um e-mail de "boas-vindas" a cada novo cliente. Você pode habilitar um stream nessa tabela e depois associá-lo a uma função do Lambda. A função do Lambda seria executada sempre que um novo registro de stream aparecesse, mas apenas processaria novos itens adicionados ao cliente da tabela `INTO`. Para qualquer item que tenha um `EmailAddress`, a função do Lambda invocaria o Amazon Simple Email Service (Amazon SES) para enviar um e-mail a esse endereço.



Note

Neste exemplo, o último cliente, Craig Roe, não receberá um e-mail, pois não tem umEmailAddress.

Além de gatilhos, o DynamoDB Streams permite soluções poderosas, como a replicação de dados dentro e entre AWS Regiões, visualizações materializadas de dados em tabelas do DynamoDB, análise de dados usando exibições materializadas do Kinesis e muito mais.

Para mais informações, consulte [Captura de dados de alteração para DynamoDB Streams \(p. 651\)](#).

API do DynamoDB

Para trabalhar com o Amazon DynamoDB, o aplicativo deve usar algumas operações simples de API. Veja a seguir um resumo dessas operações, organizadas por categoria.

Tópicos

- [Plano de controle \(p. 10\)](#)
- [Plano de dados \(p. 11\)](#)
- [DynamoDB Streams \(p. 12\)](#)
- [Transactions \(p. 12\)](#)

Plano de controle

As operações de controle permitem que você crie e gerencie as tabelas do DynamoDB. Elas também permitem que você trabalhe com índices, streams e outros objetos que são dependentes de tabelas.

- `CreateTable`— cria uma nova tabela. Opcionalmente, você pode criar um ou mais índices secundários e habilitar o DynamoDB Streams para a tabela.
- `DescribeTable`— retorna informações sobre uma tabela, como seu esquema de chaves primárias, configurações de throughput e informações de índice.
- `ListTables` — retorna os nomes de todas as suas tabelas em uma lista.
- `UpdateTable`— modifica as configurações de uma tabela ou de seus índices, cria ou remove novos índices em uma tabela ou modifica as configurações do DynamoDB Streams para uma tabela.
- `DeleteTable`— remove uma tabela e todos os seus objetos dependentes do DynamoDB.

Plano de dados

As operações do plano de dados permitem criar, ler, atualizar e excluir (também chamadas de CRUD) nos dados de uma tabela. Algumas das operações de plano de dados também permitem que você leia dados de um índice secundário.

Você pode usar o [PartiQL - Uma linguagem de consulta compatível com SQL para o Amazon DynamoDB \(p. 529\)](#), para executar essas operações CRUD ou você pode usar as APIs CRUD clássicas do DynamoDB que separa cada operação em uma chamada de API distinta.

PartiQL - Uma linguagem de consulta compatível com SQL

- `ExecuteStatement`— lê vários itens de uma tabela. Você também pode gravar ou atualizar um único item de uma tabela. Ao gravar ou atualizar um único item, é preciso especificar os atributos da chave primária.
- `BatchExecuteStatement`— Grava, atualiza ou lê vários itens de uma tabela. Isso é mais eficiente do que `ExecuteStatement` porque seu aplicativo precisa apenas de uma única viagem de ida e volta na rede para gravar ou ler os itens.

APIs clássicas

Criação de dados

- `PutItem`— grava um único item em uma tabela. Você deve especificar os atributos de chave primária, mas não precisa especificar outros atributos.
- `BatchWriteItem`— grava até 25 itens em uma tabela. Isso é mais eficiente do que chamar `PutItem` várias vezes, pois seu aplicativo precisa apenas de uma única viagem de ida e volta na rede para gravar os itens. Você também pode usar `BatchWriteItem` para excluir vários itens de uma ou mais tabelas.

Leitura de dados

- `GetItem`— recupera um único item de uma tabela. É necessário especificar a chave primária do item desejado. É possível recuperar o item inteiro ou apenas um subconjunto dos seus atributos.
- `BatchGetItem`— recupera até 100 itens de uma ou mais tabelas. Isso é mais eficiente do que chamar `GetItem` várias vezes, pois seu aplicativo precisa apenas de uma única viagem de ida e volta na rede para ler os itens.
- `Query`— Recupera todos os itens que têm uma chave de partição específica. Você deve especificar o valor da chave de partição. É possível recuperar itens inteiros, ou apenas um subconjunto dos seus atributos. Se desejar, é possível aplicar uma condição aos valores de chaves de classificação, para recuperar somente um subconjunto dos dados que têm a mesma chave de partição. Essa operação pode ser usada em uma tabela, desde que essa tabela tenha uma chave de partição e uma chave de classificação. Ela também pode ser usada em um índice, desde que esse índice tenha uma chave de partição e uma chave de classificação.

- **Scan**— Recupera todos os itens da tabela ou do índice especificado. É possível recuperar itens inteiros, ou apenas um subconjunto dos seus atributos. Opcionalmente, você pode aplicar uma condição de filtragem para retornar apenas os valores de interesse e descartar o restante.

Atualização de dados

- **UpdateItem**— modifica um ou mais atributos em um item. É necessário especificar a chave primária do item que você deseja modificar. É possível adicionar novos atributos e modificar ou remover atributos existentes. Também é possível realizar atualizações condicionais, para que a atualização apenas seja bem-sucedida quando a uma condição definida pelo usuário for atendida. Opcionalmente, você pode implementar um contador atômico, que incrementa ou diminui um atributo numérico sem interferir em outras solicitações de gravação.

Como excluir dados

- **DeleteItem**— exclui um único item de uma tabela. É necessário especificar a chave primária do item que você deseja excluir.
- **BatchWriteItem**— exclui até 25 itens de uma ou mais tabelas. Isso é mais eficiente do que chamar `DeleteItem` várias vezes, pois seu aplicativo precisa apenas de uma única viagem de ida e volta na rede para excluir os itens. Você também pode usar `BatchWriteItem` para adicionar vários itens a uma ou mais tabelas.

DynamoDB Streams

DynamoDB StreamsAs operações do permitem que você habilite ou desabilite um streaming em uma tabela e permitem o acesso a registros de modificação de dados contidos em um streaming.

- `ListStreams` – retorna uma lista de todos os seus streams ou apenas do stream para uma tabela específica.
- `DescribeStream` – retorna informações sobre um stream, como seu Nome de recurso da Amazon (ARN) e onde seu aplicativo pode começar a ler os primeiros registros de stream.
- `GetShardIterator` – retorna um iterador de estilhaços, uma estrutura de dados que seu aplicativo utiliza para recuperar os registros do stream.
- `GetRecords` – recupera um ou mais registros de stream, usando um determinado iterador de estilhaços.

Transactions

Transações fornecem atomicidade, consistência, isolamento e durabilidade (ACID), permitindo que você mantenha a exatidão dos dados em seus aplicativos com mais facilidade.

Você pode usar o[PartiQL - Uma linguagem de consulta compatível com SQL para o Amazon DynamoDB \(p. 529\)](#), para executar operações transacionais ou você pode usar as APIs CRUD clássicas do DynamoDB que separa cada operação em uma chamada de API distinta.

PartiQL - Uma linguagem de consulta compatível com SQL

- `ExecuteTransaction`— uma operação em lote que permite operações CRUD a vários itens dentro e entre tabelas com um resultado garantido de tudo ou nada.

APIs clássicas

- `TransactWriteItems`— Uma operação em lote que permite `Put`, `Update`, e `Delete` Operações para vários itens dentro e entre tabelas com um resultado garantido de tudo ou nada.
- `TransactGetItems`— Uma operação em lote que permite `Get` Operações para recuperar vários itens de uma ou mais tabelas.

Regras de nomeação e tipos de dados

Esta seção descreve as regras de nomeação do Amazon DynamoDB e os vários tipos de dados para os quais o DynamoDB oferece suporte. Há limites que se aplicam a tipos de dados. Para mais informações, consulte [Tipos de dados \(p. 1061\)](#).

Tópicos

- [Regras de nomenclatura \(p. 13\)](#)
- [Tipos de dados \(p. 14\)](#)

Regras de nomenclatura

Tabelas, atributos e outros objetos no DynamoDB devem ter nomes. Os nomes devem ser concisos e significativos; por exemplo, nomes como Products, Books e Authors são autoexplicativos.

Estas são as regras de nomenclatura do DynamoDB:

- Todos os nomes devem ser codificados usando UTF-8 e diferenciam maiúsculas de minúsculas.
- Nomes de tabelas e nomes de índices devem ter entre 3 e 255 caracteres e podem conter apenas os seguintes caracteres:
 - a–z
 - A–Z
 - 0–9
 - _ (sublinhado)
 - – (traço)
 - . (ponto)
- Os nomes de atributos devem ter pelo menos 64 caracteres, mas não ser maiores do que 64 KB.

Veja as exceções a seguir. Estes nomes de atributo não devem ser maiores do que 255 caracteres:

- Nomes de chave de partição de índice secundário.
- Nomes de chave de classificação de índice secundário.
- Os nomes de atributos projetados especificados pelo usuário (aplicável apenas a índices secundários locais).

Palavras reservadas e caracteres especiais

O DynamoDB tem uma lista de palavras reservadas e caracteres especiais. Para obter uma lista completa de palavras reservadas no DynamoDB, consulte [Palavras reservadas no DynamoDB \(p. 1125\)](#). Além disso, os seguintes caracteres têm um significado especial no DynamoDB:`#(hash)` e`:(dois pontos)`.

Embora o DynamoDB permita que você use essas palavras reservadas e caracteres especiais nos nomes, é recomendável evitar, pois será necessário definir variáveis de espaço reservado sempre que usar esses nomes em uma expressão. Para mais informações, consulte [Nomes de atributo de expressão no DynamoDB \(p. 418\)](#).

Tipos de dados

O DynamoDB oferece suporte a vários tipos de dados diferentes para atributos dentro de uma tabela. Eles podem ser categorizados da seguinte maneira:

- Tipos escalares— um tipo escalar pode representar exatamente um valor. Os tipos escalares são número, string, binário, booleano e nulo.
 - Tipos de documentos— Um tipo de documento pode representar uma estrutura complexa com atributos aninhados, como aqueles que você encontraria em um documento JSON. Os tipos de documentos são Lista e Mapa.
 - Tipos de conjunto— um tipo de conjunto pode representar vários valores escalares. Os tipos de conjuntos são conjunto de strings, conjunto de números e conjunto de binários.

Quando você cria uma tabela ou um índice secundário, você deve especificar os nomes e os tipos de dados de cada atributo de chave primária (chave de partição e chave de classificação). Além disso, cada atributo de chave primária deve ser definido como o tipo String, Número ou Binário.

DynamoDB é um banco de dados NoSQL e esquemático. Isso significa que, além dos atributos de chave primária, você não precisa definir nenhum atributo ou tipo de dados ao criar tabelas. Por comparação, bancos de dados relacionais exigem que você defina os nomes e os tipos de dados de cada coluna ao criar uma tabela.

As seguintes descrições representam cada tipo de dados, juntamente com exemplos no formato JSON.

Tipos escalares

Os tipos escalares são número, string, binário, booleano e nulo.

Number

Números podem ser positivos, negativo ou zero. Os números podem ter uma precisão de até 38 dígitos. Se esse limite for excedido, uma exceção será gerada.

No DynamoDB, os números são representados como comprimento variável. Zeros iniciais e finais são cortados.

Todos os números são enviados pela rede ao DynamoDB como strings para maximizar a compatibilidade entre linguagens e bibliotecas. No entanto, o DynamoDB os trata como atributos do tipo Número para operações matemáticas.

Note

Se a precisão dos números for importante, será necessário transmitir números ao DynamoDB usando strings que você converte de um tipo de número.

É possível usar o tipo de dados Número para representar uma data ou um carimbo de data/hora. Uma maneira de fazer isso é usando o tempo epoch — o número de segundos desde 00:00:00 UTC em 1º de janeiro de 1970. Por exemplo, o tempo epoch 1437136300 representa 12:31:40 PM UTC em 17 de julho de 2015.

Para obter mais informações, consulte http://en.wikipedia.org/wiki/Unix_time.

String

Strings são Unicode com codificação binária UTF-8. O tamanho mínimo de uma string poderá ser zero, se o atributo não for usado como uma chave para um índice ou uma tabela, e ser restrito pelo limite máximo de tamanho de item do DynamoDB de 400 KB.

As seguintes restrições adicionais se aplicam aos atributos de chave primária definidos como string de tipo:

- Para uma chave primária simples, o comprimento máximo valor do primeiro atributo (a chave de partição) é 2048 bytes.
- Para uma chave primária composta, o comprimento máximo do valor do segundo atributo (a chave de classificação) é 1024 bytes.

O DynamoDB agrupa e compara strings usando os bytes da codificação de strings UTF-8 subjacente. Por exemplo, "a" (0x61) é maior que "A" (0x41) e "ż" (0xC2BF) é maior que "z" (0x7A).

É possível usar o tipo de dados String para representar uma data ou um carimbo de data/hora. Uma maneira de fazer isso é usando strings ISO 8601, conforme mostrado nestes exemplos:

- 2016-02-15
- 2015-12-21T17:42:34Z
- 20150311T122706Z

Para obter mais informações, consulte http://en.wikipedia.org/wiki/ISO_8601.

Binary

Atributos do tipo Binário podem armazenar quaisquer dados binários, como texto compactado, dados criptografados ou imagens. Sempre que o DynamoDB compara valores binários, ele trata cada byte dos dados binários como sem sinal.

O tamanho de um atributo binário poderá ser zero, se o atributo não for usado como uma chave para um índice ou uma tabela, e ser restrito pelo limite máximo de tamanho de item do DynamoDB de 400 KB.

Se você definir um atributo de chave primária como um atributo do tipo Binário, as seguintes restrições adicionais serão aplicáveis:

- Para uma chave primária simples, o comprimento máximo valor do primeiro atributo (a chave de partição) é 2048 bytes.
- Para uma chave primária composta, o comprimento máximo do valor do segundo atributo (a chave de classificação) é 1024 bytes.

Seus aplicativos devem codificar valores binários no formato codificado em base64 antes de serem enviados ao DynamoDB. Após o recebimento desses valores, o DynamoDB decodifica os dados em uma matriz de bytes sem sinal e a utiliza como o comprimento do atributo binário.

O exemplo a seguir é um atributo binário usando texto codificado em base64.

```
dGhpcyB0ZXh0IGlzIGJhc2U2NC1lbmNvZGVk
```

Boolean

Um atributo do tipo Booleano pode armazenar `true` ou `false`.

Null

Nulo representa um atributo com um estado desconhecido ou indefinido.

Tipos de documentos

Os tipos de documentos são Lista e Mapa. Esses tipos de dados podem ser aninhados entre si para representar estruturas de dados complexas com até 32 níveis de profundidade.

Não há limite para o número de valores em uma lista ou em um mapa, desde que o item que contém os valores fique no limite de tamanho de item do DynamoDB (400 KB).

Um valor de atributo poderá ser um valor binário vazio se o atributo não for usado para uma tabela ou chave de índice. Um valor de atributo não pode ser um conjunto vazio (conjunto de strings, conjunto de números e conjunto binário). No entanto, listas e mapas vazios são permitidos. Valores binários e de string vazios são permitidos dentro de listas e mapas. Para mais informações, consulte [Attributes \(p. 1062\)](#).

List

Um atributo do tipo Lista pode armazenar uma coleção ordenada de valores. Listas são delimitadas por colchetes: [...]

Uma lista é semelhante a uma matriz JSON. Não há restrições quanto aos tipos de dados que podem ser armazenados em um elemento de lista, e os elementos de um elemento de lista não precisam ser do mesmo tipo.

O exemplo a seguir mostra uma lista que contém duas strings e um número.

```
FavoriteThings: ["Cookies", "Coffee", 3.14159]
```

Note

O DynamoDB permite que você trabalhe com elementos individuais em listas, mesmo que esses elementos estejam profundamente aninhados. Para mais informações, consulte [Uso de expressões no DynamoDB \(p. 414\)](#).

Map

Um atributo do tipo Mapa pode armazenar uma coleção não ordenada de pares de nome/valor. Mapas são delimitados por chaves: { ... }

Um mapa é semelhante a um objeto JSON. Não há restrições quanto aos tipos de dados que podem ser armazenados em um elemento de mapa, e os elementos de um mapa não precisam ser do mesmo tipo.

Mapas são ideais para armazenar documentos JSON no DynamoDB. O exemplo a seguir mostra um mapa que contém uma string, um número e uma lista aninhada que contém outro mapa.

```
{
  Day: "Monday",
  UnreadEmails: 42,
  ItemsOnMyDesk: [
    "Coffee Cup",
    "Telephone",
    {
      Pens: { Quantity : 3},
      Pencils: { Quantity : 2},
      Erasers: { Quantity : 1}
    }
  ]
}
```

Note

O DynamoDB permite que você trabalhe com elementos individuais em mapas, mesmo que esses elementos estejam profundamente aninhados. Para mais informações, consulte [Uso de expressões no DynamoDB \(p. 414\)](#).

Sets

O DynamoDB oferece suporte a tipos que representam conjuntos de valores de número, string ou binário. Todos os elementos de um conjunto devem ser do mesmo tipo. Por exemplo, um atributo do tipo Conjunto de números só pode conter números, um Conjunto de strings só pode conter apenas strings e assim por diante.

Não há limite para o número de valores em um conjunto, desde que o item que contém os valores fique no limite de tamanho de item do DynamoDB (400 KB).

Cada valor dentro de um conjunto deve ser exclusivo. A ordem dos valores em um conjunto não é preservada. Portanto, seus aplicativos não devem depender de nenhuma ordem específica de elementos no conjunto. O DynamoDB não oferece suporte a conjuntos vazios. No entanto, valores binários e de string vazios são permitidos dentro de um conjunto.

O exemplo a seguir mostra um conjunto de strings, um conjunto de números e um conjunto de binários:

```
[ "Black", "Green", "Red"]  
[ 42.2, -19, 7.5, 3.14]  
[ "U3Vubnk=", "UmFpbnk=", "U25vd3k="]
```

Consistência de leituras

O Amazon DynamoDB está disponível em vários AWS Regiões ao redor do mundo. Cada região é independente e isolada das demais AWS Regiões. Por exemplo, se você tiver uma tabela chamada People na região us-east-2 e outra tabela chamada People na região us-west-2, elas serão consideradas duas tabelas totalmente separadas. Para obter uma lista de todos os AWS Regiões em que o DynamoDB está disponível, consulte [AWS Regiões e endpoints](#) no Referência geral do Amazon Web Services.

Cada AWS Region consiste em vários locais distintos chamados zonas de disponibilidade. Cada zona de disponibilidade é isolada das falhas de outras zonas de disponibilidade e fornece conectividade de rede de baixa latência e baixo custo para outras zonas de disponibilidade da mesma região. Isso permite a rápida replicação dos seus dados entre várias zonas de disponibilidade em uma região.

Quando seu aplicativo grava dados em uma tabela do DynamoDB e recebe uma resposta HTTP 200 (OK), a gravação ocorreu e é durável. Os dados serão eventualmente consistentes em todos os locais de armazenamento, geralmente em um segundo ou menos.

DynamoDB oferece suporte a eventualmente consistente e fortemente consistente.

Leituras eventualmente coerentes

Quando você lê dados de uma tabela do DynamoDB, a resposta pode não refletir os resultados de uma operação de gravação recentemente concluída. A resposta pode incluir alguns dados obsoletos. Se você repetir sua solicitação de leitura após um curto período, a resposta deverá retornar os dados mais recentes.

Leituras fortemente consistentes

Quando você solicita uma leitura fortemente consistente, o DynamoDB retorna uma resposta com os dados mais atualizados, refletindo as atualizações de todas as operações de gravação anteriores que foram bem-sucedidas. No entanto, essa consistência traz algumas desvantagens.

- Uma leitura fortemente consistente poderá não estar disponível se houver um atraso ou uma interrupção na rede. Nesse caso, o DynamoDB pode retornar um erro de servidor (HTTP 500).

- As leituras altamente consistentes podem ter maior latência do que as leituras eventualmente consistentes.
- Leituras consistentes não são compatíveis em índices secundários globais.
- As leituras altamente consistentes usam maior capacidade de taxa de transferência do que as leituras eventualmente consistentes. Para obter mais detalhes, consulte [Modo de capacidade de leitura/gravação \(p. 18\)](#).

Note

O DynamoDB usa leituras eventualmente consistentes, a menos que você especifique o contrário. As operações de leitura (como `GetItem`, `Query` e `Scan`) fornecem um parâmetro `ConsistentRead`. Se você definir esse parâmetro para `true`, o DynamoDB usará leituras fortemente consistentes durante a operação.

Modo de capacidade de leitura/gravação

O Amazon DynamoDB tem dois modos de capacidade de leitura/gravação para processar leituras e gravações em suas tabelas:

- Sob demanda
- Provisionada (padrão, qualificada para o nível gratuito)

O modo de capacidade de leitura/gravação controla como você é cobrado por taxa de rendimento de leitura e gravação e como você gerencia a capacidade. Você pode definir o modo de capacidade de leitura/gravação ao criar uma tabela ou pode mudá-lo posteriormente.

Índices secundários locais herdam o modo de capacidade leitura/gravação da tabela base. Para mais informações, consulte [Considerações ao mudar o modo de capacidade de leitura/gravação \(p. 344\)](#).

Tópicos

- [Modo sob demanda \(p. 18\)](#)
- [Modo provisionado \(p. 20\)](#)

Modo sob demanda

O Amazon DynamoDB sob demanda é uma opção de faturamento flexível capaz de servir centenas de solicitações por segundo sem planejamento de capacidade. DynamoDB sob demanda oferece definição de preço "pague por solicitação" para solicitações de leitura e gravação, para que você pague apenas pelo o que usar.

Quando você escolhe o modo sob demanda, o DynamoDB acomoda instantaneamente o crescimento e a redução das cargas de trabalho para qualquer nível de tráfego previamente registrado. Se o nível de tráfego de uma carga de trabalho atingir um novo pico, o DynamoDB se adaptará rapidamente para acomodar a carga de trabalho. Tabelas que usam o modo sob demanda entregam a mesma latência milissegundo de digito único, compromisso de acordo de nível de serviço (SLA) e segurança que o DynamoDB já oferece. Você pode escolher sob demanda para tabelas novas e existentes e continuar usando as APIs do DynamoDB sem alterar códigos.

Modo sob demanda é uma boa opção se qualquer uma das declarações a seguir for verdadeira:

- Você cria novas tabelas com cargas de trabalho desconhecidas.
- Você tem tráfego de aplicativos imprevisível.
- Você prefere a facilidade de pagar somente pelo que usar.

A taxa de solicitação só é limitada pelas cotas da tabela padrão de taxa de transferência do DynamoDB, mas pode ser criado por solicitação. Para mais informações, consulte [Cotas padrão de taxa de transferência \(p. 1057\)](#).

Para começar som sob demanda, você pode criar ou atualizar uma tabela para usar modo sob demanda. Para mais informações, consulte [Operações básicas nas tabelas do DynamoDB \(p. 340\)](#).

Você pode alternar entre os modos de capacidade de leitura/gravação uma vez a cada 24 horas. Para problemas que você deva considerar ao trocar os modos de capacidade de leitura/gravação, consulte [Considerações ao mudar o modo de capacidade de leitura/gravação \(p. 344\)](#).

Tópicos

- [Unidades de solicitação de leitura e unidades de solicitação de gravação \(p. 19\)](#)
- [Tráfego de pico e propriedades de escalabilidade \(p. 19\)](#)
- [Taxa de rendimento inicial para modo de capacidade sob demanda \(p. 20\)](#)
- [Comportamento de tabela enquanto troca para o modo de capacidade leitura/gravação \(p. 20\)](#)

Unidades de solicitação de leitura e unidades de solicitação de gravação

Para tabelas de modo sob demanda, você não precisa especificar quanto de taxa de transferência de leitura e gravação você espera que seu aplicativo execute. O DynamoDB cobra você pelas leituras e gravações que seu aplicativo realiza em suas tabelas em termos de unidades de solicitação de leitura e unidades de solicitação de gravação.

- Oneunidade de solicitação de leituraO representa uma solicitação de leitura fortemente consistente ou duas solicitações de leitura eventualmente consistentes, para um item de até 4 KB de tamanho. Duas unidades de solicitação de leitura representam uma leitura transacional para itens de até 4 KB. Se você precisar ler um item maior que 4 KB, o DynamoDB precisará de unidades de solicitação de leitura adicionais. O número total de unidades de solicitação de leitura necessárias varia de acordo com o tamanho do item e se você deseja uma leitura eventualmente consistente ou fortemente consistente. Por exemplo, se o tamanho do seu item for 8 KB, você precisa de 2 unidades de solicitação de leitura para suportar uma leitura fortemente consistente, 1 unidade de solicitação de leitura se você escolher leituras eventualmente consistente, ou 4 unidades de solicitação de leitura para uma solicitação de leitura transacional.

Note

Para saber mais sobre os modelos de consistência de leitura do DynamoDB, consulte [Consistência de leituras \(p. 17\)](#).

- Oneunidade de solicitação de gravaçãorepresenta uma gravação para um item de até 1 KB de tamanho. Se você precisar gravar um item maior que 1 KB, o DynamoDB precisará consumir unidades de solicitação de gravação adicionais. As solicitações de gravação transacional exigem 2 unidades de solicitação de gravação para executar uma gravação para itens de até 1 KB. O número total de unidades de solicitação de gravação necessárias varia de acordo com o tamanho do item. Por exemplo, se o tamanho do seu item for 2 KB, são necessárias duas unidades de solicitação de gravação para dar suporte a uma solicitação de gravação ou quatro unidades de solicitação de gravação para uma solicitação de gravação transacional.

Para uma lista de AWS Regiões onde o DynamoDB sob demanda está disponível, consulte [Definição de preço do Amazon DynamoDB](#).

Tráfego de pico e propriedades de escalabilidade

Tabelas do DynamoDB usando modo de capacidade sob demanda automaticamente adaptam-se ao volume de tráfego do seu aplicativo. O modo de capacidade sob demanda instantaneamente a até o dobro

do pico de tráfego anterior em uma tabela. Por exemplo, se o padrão de tráfego de seu aplicativo varia entre 25.000 e 50.000 leituras fortemente consistentes por segundo, onde 50.000 leituras por segundo é o pico de tráfego anterior, o modo de capacidade sob demanda instantaneamente acomoda tráfego sustentado de até 100.000 leituras por segundo. Se seu aplicativo aguentar tráfego de 100.000 leituras por segundo, esse pico torna-se o novo pico anterior, habilitando tráfego subsequente de até 200.000 leituras por segundo.

Se precisar de mais que o dobro de seu pico anterior na tabela, DynamoDB aloca automaticamente mais capacidade que seu volume de tráfego aumenta para ajudar a garantir sua carga de trabalho não passar por controle de utilização. No entanto, pode ocorrer controle de utilização se você exceder o dobro de seu pico anterior dentro de 30 minutos. Por exemplo, se o padrão de tráfego de seu aplicativo varia entre 25.000 e 50.000 leituras fortemente consistentes por segundo, onde 50.000 leituras por segundo é o pico de tráfego anteriormente alcançado, DynamoDB recomenda espaçar seu crescimento de tráfego em, pelo menos, 30 minutos, antes de colocar mais de 100.000 leituras por segundo.

Taxa de rendimento inicial para modo de capacidade sob demanda

Se recentemente você trocou uma tabela existente para um modo de capacidade sob demanda pela primeira vez ou se você criou uma nova tabela com modo de capacidade sob demanda habilitado, a tabela tem as configurações de pico anterior a seguir, mesmo se a tabela não tiver apresentado tráfego anteriormente usando o modo de capacidade sob demanda:

- Tabela recém-criada com modo de capacidade sob demanda: O pico anterior é 2.000 unidades de solicitação de gravação ou 6.000 unidades de solicitação de leitura. Você pode adicionar até o dobro do pico anterior imediatamente, que possibilita tabelas sob demanda criadas recentemente para servir até 4.000 unidades de solicitação de gravação ou 12.000 unidades de solicitação de leitura, ou qualquer combinação linear dos dois.
- Tabela existente comutada para o modo de capacidade sob demanda: O pico anterior é a metade da quantidade de unidades de capacidade de gravação e de unidades de capacidade de leitura provisionadas desde a criação da tabela ou as configurações para uma tabela recém-criada com o modo de capacidade sob demanda, o que for maior. Em outras palavras, a tabela fornecerá pelo menos a mesma taxa de transferência fornecida de mudar para o modo de capacidade sob demanda.

Comportamento de tabela enquanto troca para o modo de capacidade leitura/gravação

Quando você troca uma tabela de modo de capacidade provisionado para modo de capacidade sob demanda, DynamoDB faz várias alterações na estrutura de sua tabela e partições. Esse processo pode levar alguns minutos. Durante o período de troca, sua tabela entrega taxa de rendimento que é consistente com as unidades de valor de capacidade de gravação provisionada anteriormente e unidade de capacidade de leitura. Ao voltar para o modo de capacidade sob demanda para modo de capacidade provisionado, sua tabela entrega taxa de rendimento consistente com o pico anterior alcançado quando a tabela foi definida como modo de capacidade sob demanda.

Modo provisionado

Se escolher o modo provisionado, você especifica o número de leituras e gravações por segundo exigidas pelo aplicativo. Você pode usar Auto Scaling para ajustar sua capacidade provisionada de tabela automaticamente em resposta às alterações de tráfego. Isso ajuda a governar seu uso do DynamoDB para permanecer no lugar ou abaixo de uma taxa de solicitação definida para obter previsão de custos.

Modo provisionado é uma boa opção se qualquer uma das declarações a seguir for verdadeira:

- Você tem tráfego de aplicativos previsível.
- Você executa aplicativos com tráfego consistente e que aumenta gradualmente.

- Você pode prever os requisitos de capacidade para controlar os custos.

Unidades de capacidade de leitura e unidades de capacidade de gravação

Para tabelas de modo provisionados, você especifica a capacidade de taxa de rendimento em termos de unidades de capacidade de leitura (RCUs) e unidades de capacidade de gravação (WCUs):

- OneUnidade de capacidade de leituraO representa uma leitura fortemente consistente por segundo ou duas leituras eventualmente consistentes por segundo, para um item de até 4 KB de tamanho. As solicitações de leitura transacional exigem duas unidades de capacidade de leitura para executar uma leitura por segundo para itens de até 4 KB. Se você precisar ler um item maior que 4 KB, o DynamoDB deve consumir unidades de capacidade de leitura adicionais. O número total de unidades de capacidade de leitura necessárias varia de acordo com o tamanho do item e se você deseja uma leitura eventualmente consistente ou fortemente consistente. Por exemplo, se o tamanho do seu item for 8 KB, você precisa de 2 unidades de capacidade de leitura para suportar uma leitura fortemente consistente por segundo, 1 unidade de capacidade de leitura se você escolher leituras eventualmente consistentes, ou 4 unidades de capacidade de leitura para uma solicitação de leitura transacional. Para mais informações, consulte [Consumo de unidade de capacidade para leituras \(p. 346\)](#).

Note

Para saber mais sobre os modelos de consistência de leitura do DynamoDB, consulte[Consistência de leituras \(p. 17\)](#).

- Oneunidade de capacidade de gravaçãoO representa uma gravação por segundo para um item de até 1 KB de tamanho. Se você precisar gravar um item maior que 1 KB, o DynamoDB deve consumir unidades de capacidade de gravação adicionais. As solicitações de gravação transacional exigem 2 unidades de capacidade de gravação para executar uma gravação por segundo para itens de até 1 KB. O número total de unidades de capacidade de gravação necessárias varia de acordo com o tamanho do item. Por exemplo, se o tamanho do seu item for 2 KB, você precisa de 2 unidades de capacidade de gravação para suportar uma solicitação de gravação por segundo ou 4 unidades de capacidade de gravação para uma solicitação de gravação transacional. Para mais informações, consulte [Consumo de unidade de capacidade para gravações \(p. 347\)](#).

Important

Ao chamar `DescribeTable` em uma tabela sob demanda, as unidades de capacidade de leitura e unidades de capacidade de gravação são definidas como 0.

Se o seu aplicativo ler ou gravar itens maiores (até o tamanho de item máximo do DynamoDB de 400 KB), ele consumirá mais unidades de capacidade.

Por exemplo, suponha que você crie uma tabela provisionada com 6 unidades de capacidade de leitura e 6 unidades de capacidade de gravação. Com essas configurações, seu aplicativo pode fazer o seguinte:

- Faça leituras fortemente consistentes de até 24 KB por segundo (4 KB × 6 unidades de capacidade de leitura).
- Realizar leituras eventualmente consistentes de até 48 KB por segundo (o dobro do throughput de leitura).
- Realize solicitações de leitura transacional de até 12 KB por segundo.
- Gravar até 6 KB por segundo (1 KB × 6 unidades de capacidade de gravação).
- Realize solicitações de gravação transacional de até 3 KB por segundo.

Para mais informações, consulte [Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB \(p. 345\)](#).

O throughput provisionado é o máximo da capacidade que um aplicativo pode consumir em uma tabela ou índice. Se o aplicativo exceder sua capacidade de configurações de throughput provisionado em uma tabela ou índice, ele estará sujeito à controle de utilização de solicitações.

A limitação impede que o seu aplicativo consuma muitas unidades de capacidade. Quando uma solicitação é limitada, ela falha com um código HTTP 400 (Bad Request) e uma ProvisionedThroughputExceededException. O AWSSDKs têm suporte integrado para repetir solicitações limitadas (consulte [Repetições de erro e recuo exponencial \(p. 226\)](#)), para que você precise gravar essa lógica manualmente.

Você pode usar o AWS Management Console para monitorar seu throughput provisionado e real e modificar suas configurações de throughput, se necessário.

Auto Scaling do DynamoDB

O Auto Scaling do DynamoDB gerencia a capacidade de throughput para tabelas e índices secundários globais. Com o Auto Scaling, você define um intervalo (limites superior e inferior) para unidades de capacidade de leitura e gravação. Você também define uma porcentagem de utilização alvo dentro desse intervalo. O Auto Scaling do DynamoDB procura manter sua utilização alvo, mesmo que a carga de trabalho do seu aplicativo aumente ou diminua.

Com o Auto Scaling do DynamoDB, uma tabela ou um índice secundário global pode aumentar sua capacidade provisionada de leitura e gravação para processar aumentos repentinos no tráfego, sem a limitação de solicitações. Quando a carga de trabalho diminuir, o Auto Scaling do DynamoDB diminuirá o throughput, para que você não precise pagar por uma capacidade provisionada não utilizada.

Note

Se você usar o AWS Management Console para criar uma tabela ou um índice secundário global, o DynamoDB Auto Scaling é habilitada por padrão.

Você pode gerenciar as configurações de Auto Scaling a qualquer momento usando o console, o AWS CLI, ou um dos AWSSDKs.

Para mais informações, consulte [Como gerenciar a capacidade de throughput automaticamente com o Auto Scaling do DynamoDB \(p. 350\)](#).

Capacidade reservada

Como cliente do DynamoDB, você pode comprar Capacidade reservada com antecedência, conforme descrito em [Definição de preço do Amazon DynamoDB](#). Com a capacidade reservada, você paga uma taxa única antecipada e se compromete a um nível mínimo de uso provisionado ao longo de um período. Sua capacidade reservada é cobrada de acordo com a taxa de capacidade reservada por hora. Ao reservar suas unidades de capacidade de leitura e gravação antes do tempo, é possível obter uma economia significativa em comparação com as configurações de throughput sob demanda ou provisionado. Qualquer capacidade que você provisionar além da sua capacidade reservada será cobrada de acordo com as taxas de capacidade provisionada padrão.

Note

Capacidade reservada não está disponível em modo sob demanda.

Para gerenciar a capacidade reservada, vá para o [Console do DynamoDB](#) e escolha Capacidade reservada.

Note

Você pode impedir que os usuários visualizem ou adquiram capacidade reservada e, ao mesmo tempo, permitir que eles acessem o restante do console. Para obter mais informações, consulte

"Conceder permissões para evitar a compra de ofertas de capacidade reservada", em [Identity and Access Management no Amazon DynamoDB \(p. 883\)](#).

Partições e distribuição de dados

O Amazon DynamoDB armazena dados em partições. A partição é uma alocação de armazenamento para uma tabela, respaldada por SSDs (unidades de estado sólido) e automaticamente replicada em várias zonas de disponibilidade em um AWS Região : O gerenciamento de partições é feito inteiramente pelo DynamoDB — você nunca precisará gerenciar partições por conta própria.

Quando você cria uma tabela, seu estado inicial é CREATING. Durante essa fase, o DynamoDB aloca partições suficientes para a tabela, para que ela possa lidar com suas necessidades de throughput provisionado. Você pode começar a gravar e ler dados da tabela depois que o status da tabela mudar para ACTIVE.

O DynamoDB alocará partições adicionais em uma tabela nas seguintes situações:

- Se você aumentar as configurações de throughput provisionado da tabela além do que as partições existentes podem suportar.
- Se uma partição existente for ocupada até a capacidade máxima e mais espaço de armazenamento for necessário.

O gerenciamento de partições ocorre automaticamente em segundo plano e é transparente para os aplicativos. Sua tabela permanece disponível durante todo o processo e oferece suporte total para os seus requisitos de throughput provisionado.

Para obter mais detalhes, consulte [Design de chave de partição \(p. 986\)](#).

Os índices secundários globais no DynamoDB também são compostos por partições. Os dados em um índice secundário global são armazenados separadamente dos dados em sua tabela base, mas as partições de índice se comportam da mesma forma que partições de tabela.

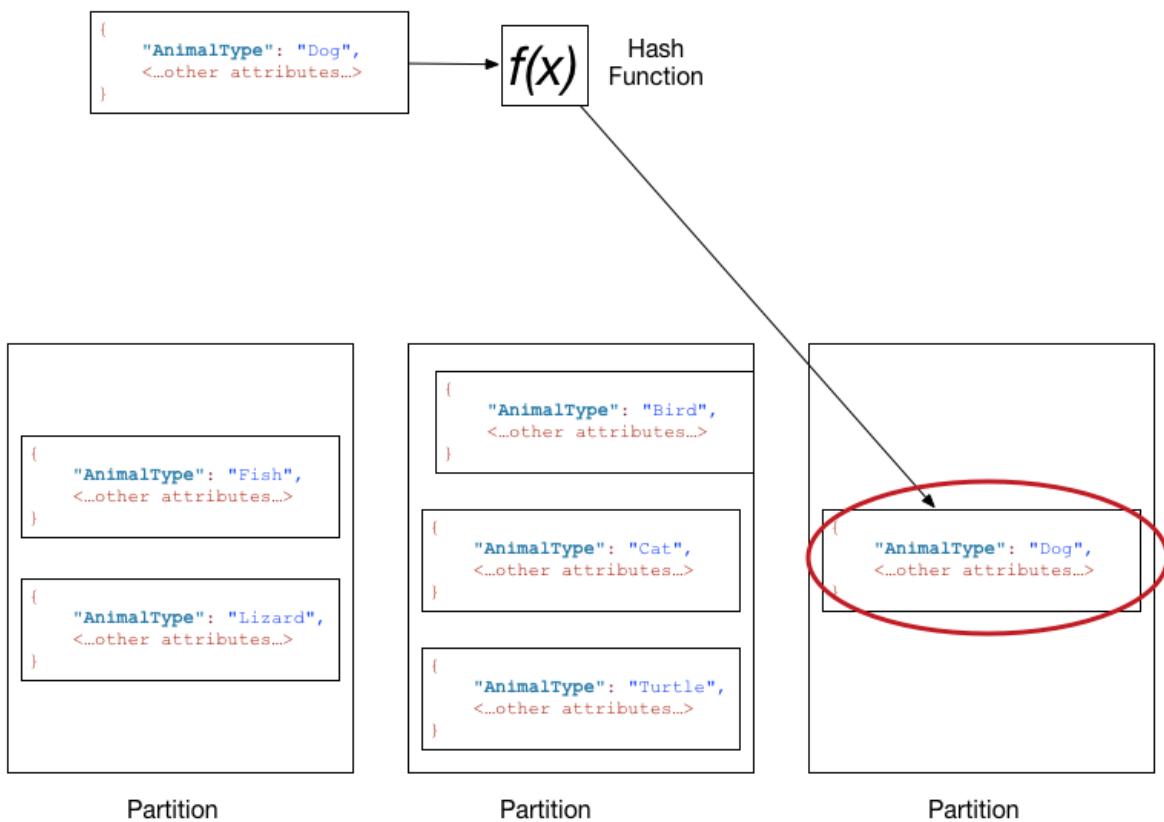
Distribuição de dados: Chave de partição

Se a sua tabela tiver uma chave primária simples (somente uma chave de partição), o DynamoDB armazenará e recuperará cada item com base em seu valor de chave de partição.

Para gravar um item na tabela, o DynamoDB usa o valor da chave de partição como entrada para uma função de hash interna. O valor de saída da função de hash determina a partição na qual o item será armazenado.

Para ler um item da tabela, você deve especificar o valor da chave de partição desse item. O DynamoDB usa esse valor como entrada para sua função de hash, resultando na partição na qual o item pode ser encontrado.

O diagrama a seguir mostra uma tabela chamada Pets, que se estende por várias partições. A chave primária da tabela é AnimalType (somente esse atributo de chave é mostrado). DynamoDB usa sua função de hash para determinar onde armazenar um novo item, neste caso, com base no valor de hash da string Cão. Observe que os itens não são armazenados na ordem classificada. A localização de cada item é determinada pelo valor de hash de sua chave de partição.



Note

DynamoDB é otimizado para distribuição uniforme de itens entre partições de uma tabela, independentemente de quantas partições possam existir. Recomendamos que você escolha uma chave de partição que possa ter um grande número de valores distintos em relação ao número de itens da tabela.

Distribuição de dados: Chave de partição e chave de classificação

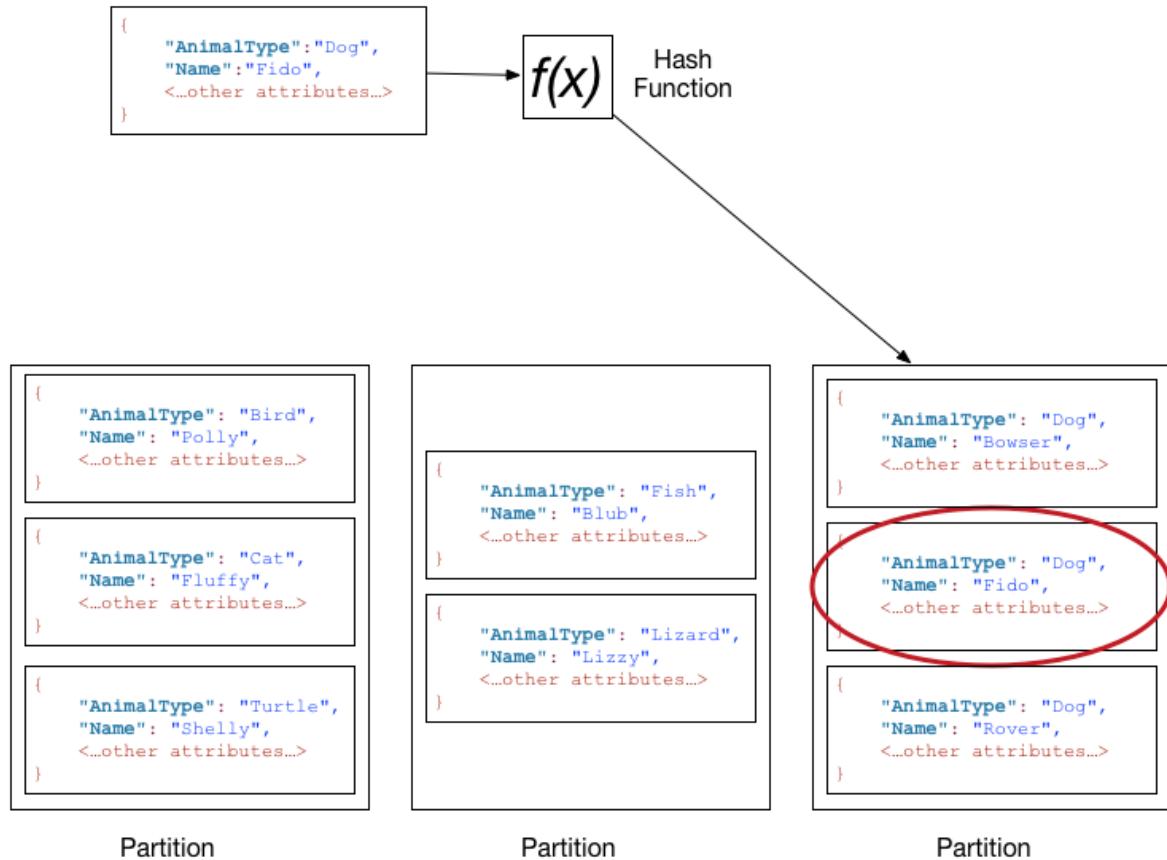
Se a tabela tiver uma chave primária composta (chave de partição e chave de classificação), o DynamoDB calculará o valor de hash da chave de partição da mesma forma que descrito em [Distribuição de dados: Chave de partição \(p. 23\)](#). Entretanto, ele armazena todos os itens com o mesmo valor de chave de partição fisicamente próximos, ordenados por valor de chave de classificação.

Para gravar um item na tabela, o DynamoDB calcula o valor de hash da chave de partição a fim de determinar qual partição deve conter esse item. Nessa partição, vários itens poderiam ter o mesmo valor de chave de partição. Portanto, o DynamoDB armazena o item entre os outros com a mesma chave de partição, em ordem ascendente por chave de classificação.

Para ler um item da tabela, você deve especificar seu valor de chave de partição e seu valor de chave de classificação. O DynamoDB calcula o valor de hash da chave de partição, resultando na partição na qual o item pode ser encontrado.

Você pode ler vários itens da tabela em uma única operação (`Query`) se os itens desejados tiverem o mesmo valor de chave de partição. O DynamoDB retorna todos os itens com esse valor de chave de partição. Opcionalmente, você pode aplicar uma condição à chave de classificação para que ela retorne somente os itens em um determinado intervalo de valores.

Suponhamos que a tabela Pets possua uma chave primária composta que consiste em AnimalType (chave de partição) e Name (chave de classificação). O diagrama a seguir mostra o DynamoDB gravando um item com um valor de chave de partição de Cão e um valor de chave de classificação de Fido.



Para ler o mesmo item doAnimais, o DynamoDB calcula o valor de hash deCão, produzindo a partição na qual esses itens são armazenados. Em seguida, o DynamoDB verifica os valores de atributos de chave de classificação até encontrarFido.

Para ler todos os itens com um AnimalType de Dog, você pode emitir uma operação Query sem especificar uma condição de chave de classificação. Por padrão, os itens são retornados na ordem em que são armazenados (ou seja, em ordem crescente por chave de classificação). Existe a opção de solicitar a ordem decrescente em vez disso.

Para consultar somente alguns itens de Dog, você pode aplicar uma condição à chave de classificação (por exemplo, apenas os itens Dog em que Name comece com uma letra que esteja dentro do intervalo de A a K).

Note

Em uma tabela do DynamoDB, não há limite superior quanto ao número de valores de chave de classificação distintos por valor de chave de partição. Se você precisava armazenar muitos bilhões deCãoITEMS noAnimais, o DynamoDB alocaará armazenamento suficiente para atender a esse requisito automaticamente.

Do SQL para o NoSQL

Se você é um desenvolvedor de aplicativos, talvez tenha alguma experiência no uso do sistema de gerenciamento de banco de dados relacional (RDBMS) e Structured Query Language (SQL). Ao começar

a trabalhar com o Amazon DynamoDB, você encontrará muita similaridade, mas também muitas coisas que são diferentes. Esta seção descreve tarefas comuns de banco de dados, comparando e contrastando instruções SQL com suas operações equivalentes do DynamoDB.

NoSQL é um termo usado para descrever os sistemas de bancos de dados não relacionais altamente disponíveis, dimensionáveis e otimizados para alto desempenho. Em vez de usar o modelo relacional, os bancos de dados NoSQL (como o DynamoDB) usam modelos alternativos para o gerenciamento de dados, como pares de chave-valor ou armazenamento de documentos. Para obter mais informações, consulte <http://aws.amazon.com/nosql>.

Note

Os exemplos de SQL nesta seção são compatíveis com o RDBMS MySQL.

Os exemplos do DynamoDB nesta seção mostram o nome da operação do DynamoDB, junto com os parâmetros dessa operação no formato JSON. Para obter exemplos de código que usam essas operações, consulte [Conceitos básicos do DynamoDB e do AWSSDKs da \(p. 81\)](#).

Tópicos

- [Relacional \(SQL\) ou NoSQL? \(p. 26\)](#)
- [Características dos bancos de dados \(p. 28\)](#)
- [Criação de uma tabela \(p. 30\)](#)
- [Obter informações sobre uma tabela \(p. 31\)](#)
- [Gravar dados em uma tabela \(p. 32\)](#)
- [Principais diferenças ao ler dados de uma tabela \(p. 35\)](#)
- [Gerenciamento de índices \(p. 41\)](#)
- [Modificação de dados em uma tabela \(p. 44\)](#)
- [Exclusão de dados de uma tabela \(p. 46\)](#)
- [Remoção de uma tabela \(p. 47\)](#)

Relacional (SQL) ou NoSQL?

Atualmente, os aplicativos têm exigências maiores do que nunca. Por exemplo, um jogo online pode começar com apenas alguns usuários e uma pequena quantidade de dados. No entanto, se o jogo obtiver sucesso, ele poderá facilmente superar os recursos do sistema de gerenciamento de banco de dados subjacente. É comum que aplicativos baseados na web tenham centenas, milhares ou milhões de usuários simultâneos, com terabytes ou mais de novos dados gerados por dia. Os bancos de dados para tais aplicativos devem lidar com dezenas (ou centenas) de milhares de leituras e gravações por segundo.

O Amazon DynamoDB é perfeitamente adequado para esse tipo de cargas de trabalho. Como um desenvolvedor, você pode começar com uma pequena quantidade e aumentar gradualmente sua utilização, conforme o aplicativo se tornar mais popular. O DynamoDB é dimensionado perfeitamente para lidar com quantidades muito grande de dados e um número muito grande de usuários.

A tabela a seguir mostra algumas diferenças de alto nível entre um RDBMS e o DynamoDB.

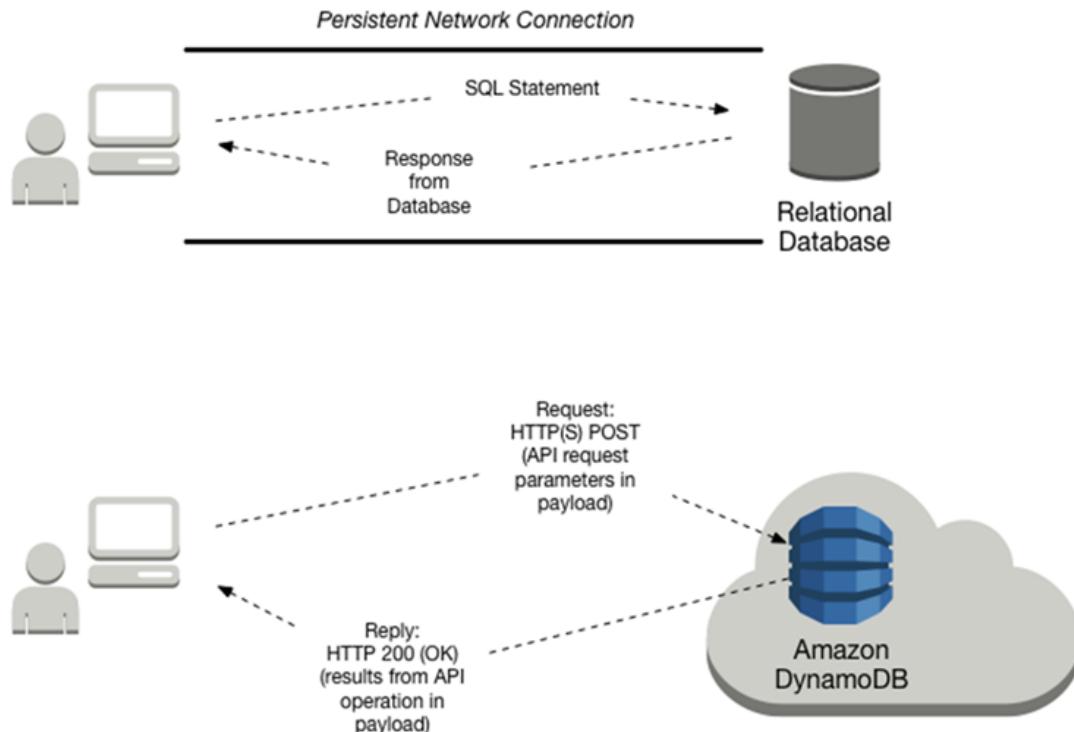
Característica	Sistema de gerenciamento de banco de dados relacional (RDBMS)	Amazon DynamoDB
Cargas de trabalho ideais	Consultas ad hoc; data warehouse; OLAP (processamento analítico online).	Aplicativos em escala da web, incluindo redes sociais, jogos, compartilhamento de mídia e Internet das Coisas (IoT).

Característica	Sistema de gerenciamento de banco de dados relacional (RDBMS)	Amazon DynamoDB
Modelo de dados	O modelo relacional exige um esquema bem definido, onde os dados são normalizados em tabelas, linhas e colunas. Além disso, todos os relacionamentos são definidos entre tabelas, colunas, índices e outros elementos de banco de dados.	O DynamoDB é sem esquema. Cada tabela deve ter uma chave primária para identificar exclusivamente cada item de dados, mas não há restrições semelhantes em outros atributos não chave. O DynamoDB pode gerenciar dados estruturados ou semiestruturados, incluindo documentos JSON.
Acesso aos dados	SQL é o padrão para armazenar e recuperar dados. Os bancos de dados relacionais oferecem um conjunto completo de ferramentas para simplificar o desenvolvimento de aplicativos para banco de dados, mas todas essas ferramentas usam o SQL.	Você pode usar o AWS Management Console ou o AWS CLI Para trabalhar com o DynamoDB e executar tarefas ad hoc. Os aplicativos podem usar o AWS SDKs para trabalhar com o DynamoDB usando interfaces baseadas em objeto, centrada em documento ou de baixo nível.
Desempenho	Os bancos de dados relacionais são otimizados para armazenamento, portanto, o desempenho geralmente depende do subsistema de disco. Os desenvolvedores e os administradores de banco de dados devem otimizar as consultas, os índices e as estruturas de tabela para obter o máximo desempenho.	O DynamoDB é otimizado para computação, portanto, o desempenho é principalmente uma função do hardware subjacente e da latência de rede. Como um serviço gerenciado, o DynamoDB protege você e seus aplicativos a partir desses detalhes de implementação, para que você possa se concentrar em projetar e construir aplicativos robustos e de alto desempenho.
Escalabilidade	É mais fácil aumentar a escala com hardware mais rápido. Também é possível que as tabelas de banco de dados se expandam em vários hosts em um sistema distribuído, mas isso exige investimentos adicionais. Os bancos de dados relacionais têm tamanhos máximos para o número e o tamanho dos arquivos, o que impõe limites superiores na escalabilidade.	O DynamoDB é projetado para escalar usando clusters distribuídos de hardware. Esse design permite aumentar o throughput sem aumentar a latência. Os clientes especificam seus requisitos de throughput, e o DynamoDB aloca recursos suficientes para atender a esses requisitos. Não há limites superiores no número de itens por tabela, nem no tamanho total da tabela.

Características dos bancos de dados

Para que seu aplicativo possa acessar um banco de dados, ele deve ser autenticado para garantir que o aplicativo tenha permissão para usar o banco de dados. Ele deve ser autorizado para que o aplicativo possa realizar somente as ações para as quais tem permissões.

O diagrama a seguir mostra a interação do cliente com um banco de dados relacional e com o Amazon DynamoDB.



A tabela a seguir tem mais detalhes sobre tarefas de interação com o cliente.

Característica	Sistema de gerenciamento de banco de dados relacional (RDBMS)	Amazon DynamoDB
Ferramentas para acessar o banco de dados	A maioria dos bancos de dados relacionais fornecem uma interface de linha de comando (CLI), para que você possa inserir instruções SQL ad hoc e visualizar os resultados imediatamente.	Na maioria dos casos, você grava o código do aplicativo. Você também pode usar o AWS Management Console ou o AWS Command Line Interface (AWS CLI) para enviar solicitações ad hoc ao DynamoDB e visualizar os resultados.
Conexão ao banco de dados	Um programa aplicativo estabelece e mantém uma conexão de rede com o banco de dados. Quando o aplicativo terminar, ele encerra a conexão.	O DynamoDB é um serviço da web, e as interações com ele são stateless. Os aplicativos não precisam manter conexões de rede persistentes. Em vez disso, a interação com o DynamoDB

Característica	Sistema de gerenciamento de banco de dados relacional (RDBMS)	Amazon DynamoDB
		ocorre usando solicitações e respostas HTTP (S).
Autenticação	Um aplicativo não pode se conectar ao banco de dados até ser autenticado. O RDBMS pode realizar a autenticação em si, ou pode passar essa tarefa para o sistema operacional host ou um serviço de diretório.	Cada solicitação para o DynamoDB deve ser acompanhada por uma assinatura de criptografia, que autentica essa determinada solicitação. OAWSOs SDKs fornecem toda a lógica necessária para a criação de assinaturas e a assinatura de solicitações. Para obter mais informações, consulte Assinatura AWS Solicitações da API no AWS Referência geral.
Autorização	Os aplicativos podem executar somente as ações para as quais tenham sido autorizados. Os administradores de banco de dados ou os proprietários de aplicativos podem usar as instruções SQL GRANT e REVOKE para controlar o acesso aos objetos de banco de dados (tais como tabelas), dados (como linhas em uma tabela), ou a habilidade de emitir determinadas instruções SQL.	No DynamoDB, a autorização é controlada peloAWS Identity and Access Management(IAM). Você pode escrever uma política IAM para conceder permissões em um recurso do DynamoDB (tal como uma tabela) e, em seguida, permitir que os usuários e as funções do IAM usem essa política. O IAM também oferece controle de acesso minucioso para itens de dados individuais em tabelas do DynamoDB. Para mais informações, consulte Identity and Access Management no Amazon DynamoDB (p. 883).
Envio de uma solicitação	O aplicativo emite uma instrução SQL para cada operação de banco de dados que ele deseja executar. Após o recebimento da instrução SQL, o RDBMS verifica a sintaxe, cria um plano para realizar a operação e, em seguida, executa o plano.	O aplicativo envia solicitações HTTP (S) para o DynamoDB. As solicitações contêm o nome da operação do DynamoDB a ser executada, juntamente com parâmetros. O DynamoDB executa a solicitação imediatamente.
Recebimento de uma resposta	O RDBMS retorna os resultados da instrução SQL. Se houver um erro, o RDBMS retorna um status e uma mensagem de erro.	O DynamoDB retorna uma resposta HTTP (S) contendo os resultados da operação. Se houver um erro, o DynamoDB retornará um status e mensagens de erro HTTP.

Criação de uma tabela

Tabelas são as estruturas de dados fundamentais em bancos de dados relacionais e no Amazon DynamoDB. Um Relational Database Management System (RDBMS – Sistema de gerenciamento de banco de dados relacional) exige que você defina o esquema da tabela ao criá-la. Por outro lado, as tabelas do DynamoDB não têm esquemas além da chave primária, não é necessário definir nenhum atributo ou tipo de dados extras ao criar uma tabela.

Tópicos

- [SQL \(p. 30\)](#)
- [DynamoDB \(p. 30\)](#)

SQL

Use a instrução `CREATE TABLE` para criar uma tabela, conforme mostrado no exemplo a seguir.

```
CREATE TABLE Music (
    Artist VARCHAR(20) NOT NULL,
    SongTitle VARCHAR(30) NOT NULL,
    AlbumTitle VARCHAR(25),
    Year INT,
    Price FLOAT,
    Genre VARCHAR(10),
    Tags TEXT,
    PRIMARY KEY(Artist, SongTitle)
);
```

A chave primária dessa tabela consiste em `Artist` e `SongTitle`.

Você deve definir todas as colunas e os tipos de dados da tabela, e a chave primária da tabela. (Você pode usar a instrução `ALTER TABLE` para alterar essas definições mais tarde, se necessário.)

Muitas implementações de SQL permitem que você defina especificações de armazenamento para a sua tabela, como parte da instrução `CREATE TABLE`. A não ser que você indique de outra forma, a tabela é criada com configurações de armazenamento padrão. Em um ambiente de produção, um administrador de banco de dados pode ajudar a determinar os parâmetros de armazenamento ideais.

DynamoDB

Use a ação `CreateTable` para criar uma tabela de modo provisionado, especificando parâmetros conforme mostrado a seguir:

```
{
  TableName : "Music",
  KeySchema: [
    {
      AttributeName: "Artist",
      KeyType: "HASH", //Partition key
    },
    {
      AttributeName: "SongTitle",
      KeyType: "RANGE" //Sort key
    }
  ],
  AttributeDefinitions: [
    {
      AttributeName: "Artist",
      AttributeType: "S"
    }
  ],
  ProvisionedThroughput: {
    ReadCapacityUnits: 5,
    WriteCapacityUnits: 5
  }
}
```

```
        AttributeType: "S"
    },
{
    AttributeName: "SongTitle",
    AttributeType: "S"
}
],
ProvisionedThroughput: {           // Only specified if using provisioned mode
    ReadCapacityUnits: 1,
    WriteCapacityUnits: 1
}
}
```

A chave primária dessa tabela consiste em Artist (chave de partição) e SongTitle (chave de classificação).

Você deve fornecer os parâmetros a seguir para `CreateTable`:

- `TableName` – Nome da tabela.
- `KeySchema` - Os atributos que são usados para a chave primária. Para mais informações, consulte e .
- `AttributeDefinitions` - Os tipos de dados dos atributos de esquema de chaves.
- `ProvisionedThroughput` (*for provisioned tables*) - Número de leituras e gravações por segundo que você precisa para esta tabela. O DynamoDB reserva recursos do sistema e de armazenamento suficientes para que seus requisitos de throughput sejam sempre atendidos. Você pode usar a ação `UpdateTable` para alterar essas configurações mais tarde, se necessário. Você não precisa especificar os requisitos de armazenamento de uma tabela porque a alocação de armazenamento é gerenciada inteiramente pelo DynamoDB.

Note

Para obter exemplos de código que usam `CreateTable`, consulte [Conceitos básicos do DynamoDB e do AWS SDKs da \(p. 81\)](#).

Obter informações sobre uma tabela

Você pode verificar se uma tabela foi criada de acordo com suas especificações. Em um banco de dados relacional, todo o esquema da tabela é mostrado. As tabelas do Amazon DynamoDB não têm esquemas, portanto, somente os atributos de chave primária são mostrados.

Tópicos

- [SQL \(p. 31\)](#)
- [DynamoDB \(p. 32\)](#)

SQL

A maioria dos sistemas de gerenciamento de banco de dados relacional (RDBMS) permitem que você descreva a estrutura de uma tabela – colunas, tipos de dados, definição de chave primária, e assim por diante. Não há nenhuma maneira padrão de fazer isso no SQL. No entanto, muitos sistemas de banco de dados fornecem um comando `DESCRIBE`. Veja a seguir um exemplo do MySQL.

```
DESCRIBE Music;
```

Isso retorna a estrutura de sua tabela, com todos os nomes de colunas, tipos de dados e tamanhos.

```
+-----+-----+-----+-----+-----+
```

Field	Type	Null	Key	Default	Extra
Artist	varchar(20)	NO	PRI	NULL	
SongTitle	varchar(30)	NO	PRI	NULL	
AlbumTitle	varchar(25)	YES		NULL	
Year	int(11)	YES		NULL	
Price	float	YES		NULL	
Genre	varchar(10)	YES		NULL	
Tags	text	YES		NULL	

A chave primária dessa tabela consiste em Artist e SongTitle.

DynamoDB

O DynamoDB tem um `DescribeTable`, que é semelhante. O único parâmetro é o nome da tabela.

```
{  
    TableName : "Music"  
}
```

A resposta de `DescribeTable` é semelhante ao seguinte.

```
{  
    "Table": {  
        "AttributeDefinitions": [  
            {  
                "AttributeName": "Artist",  
                "AttributeType": "S"  
            },  
            {  
                "AttributeName": "SongTitle",  
                "AttributeType": "S"  
            }  
        ],  
        "TableName": "Music",  
        "KeySchema": [  
            {  
                "AttributeName": "Artist",  
                "KeyType": "HASH" //Partition key  
            },  
            {  
                "AttributeName": "SongTitle",  
                "KeyType": "RANGE" //Sort key  
            }  
        ],  
        ...  
    }  
}
```

`DescribeTable` também retorna informações sobre índices na tabela, configurações de throughput provisionado, uma contagem aproximada de itens e outros metadados.

Gravar dados em uma tabela

As tabelas de bancos de dados relacionais contêm linhas de dados. As linhas são compostas de colunas. Tabelas do Amazon DynamoDB contêm itens. Os itens são compostos de atributos.

Esta seção descreve como gravar uma linha (ou item) em uma tabela.

Tópicos

- [SQL \(p. 33\)](#)
- [DynamoDB \(p. 33\)](#)

SQL

A tabela em um banco de dados relacional é uma estrutura de dados bidimensional composta por linhas e colunas. Alguns sistemas de gerenciamento de banco de dados também fornecem suporte para dados semiestruturados, geralmente com tipos de dados nativos JSON ou XML. No entanto, os detalhes de implementação variam entre fornecedores.

No SQL, use a instrução `INSERT` para adicionar uma linha a uma tabela.

```
INSERT INTO Music
    (Artist, SongTitle, AlbumTitle,
     Year, Price, Genre,
     Tags)
VALUES(
    'No One You Know', 'Call Me Today', 'Somewhat Famous',
    2015, 2.14, 'Country',
    '{"Composers": ["Smith", "Jones", "Davis"], "LengthInSeconds": 214}'
);
```

A chave primária dessa tabela consiste em `Artist` e `SongTitle`. Você deve especificar valores para essas colunas.

Note

Este exemplo usa a coluna `Tags` para armazenar dados semiestruturados sobre as músicas na tabela `Music`. A coluna `Tags` está definida como tipo `TEXT`, que pode armazenar até 65.535 caracteres em MySQL.

DynamoDB

No Amazon DynamoDB, você usa o PartiQL, uma linguagem de consulta compatível com SQL ou as APIs clássicas do DynamoDB para adicionar um item a uma tabela.

PartiQL

No Amazon DynamoDB, você usa o `ExecuteStatement` para adicionar um item a uma tabela, usando o `Insert` instrução PartiQL.

```
insert into Music value {
    'Artist': 'No One You Know',
    'SongTitle': 'Call Me Today',
    'AlbumTitle': 'someonewhat famous',
    'Year' : true,
    'Genre' : 'Acme'
}
```

A chave primária dessa tabela consiste em `Artist` e `SongTitle`. Você deve especificar valores para esses atributos.

Note

Para obter exemplos de códigos que usam `ExecuteStatement`, consulte [Instruções de inserção PartiQL para o DynamoDB \(p. 544\)](#).

Classic APIs

No Amazon DynamoDB, você usa o `PutItem` para adicionar um item a uma tabela.

```
{  
    TableName: "Music",  
    Item: {  
        "Artist": "No One You Know",  
        "SongTitle": "Call Me Today",  
        "AlbumTitle": "Somewhat Famous",  
        "Year": 2015,  
        "Price": 2.14,  
        "Genre": "Country",  
        "Tags": {  
            "Composers": [  
                "Smith",  
                "Jones",  
                "Davis"  
            ],  
            "LengthInSeconds": 214  
        }  
    }  
}
```

A chave primária dessa tabela consiste em `Artist` e `SongTitle`. Você deve especificar valores para esses atributos.

Aqui estão algumas coisas importantes para saber sobre este exemplo de `PutItem`:

- O DynamoDB oferece suporte nativo para documentos usando JSON. Isso torna o DynamoDB ideal para armazenar dados semiestruturados, como o `Tags`. Você também pode recuperar e manipular dados de dentro de documentos JSON.
- A tabela `Music` não tem atributos predefinidos, além da chave primária (`Artist` e `SongTitle`).
- A maioria dos bancos de dados SQL são orientados a transação. Quando você emite uma instrução `INSERT`, as modificações de dados não são permanentes até que você execute uma instrução `COMMIT`. Com o Amazon DynamoDB, os efeitos de uma `PutItem` ação do DynamoDB responde com um código de status HTTP 200 (OK).

Note

Para obter exemplos de códigos que usam `PutItem`, consulte [Conceitos básicos do DynamoDB e do AWSSDKs da \(p. 81\)](#).

A seguir há alguns outros exemplos de `PutItem`.

```
{  
    TableName: "Music",  
    Item: {  
        "Artist": "No One You Know",  
        "SongTitle": "My Dog Spot",  
        "AlbumTitle": "Hey Now",  
        "Price": 1.98,  
        "Genre": "Country",  
        "CriticRating": 8.4  
    }  
}
```

```
{  
    TableName: "Music",  
}
```

```
Item: {  
    "Artist": "No One You Know",  
    "SongTitle": "Somewhere Down The Road",  
    "AlbumTitle": "Somewhat Famous",  
    "Genre": "Country",  
    "CriticRating": 8.4,  
    "Year": 1984  
}  
}
```

```
{  
    TableName: "Music",  
    Item: {  
        "Artist": "The Acme Band",  
        "SongTitle": "Still In Love",  
        "AlbumTitle": "The Buck Starts Here",  
        "Price": 2.47,  
        "Genre": "Rock",  
        "PromotionInfo": {  
            "RadioStationsPlaying": [  
                "KHCR", "KBOX", "WTNR", "WJJH"  
            ],  
            "TourDates": {  
                "Seattle": "20150625",  
                "Cleveland": "20150630"  
            },  
            "Rotation": "Heavy"  
        }  
    }  
}
```

```
{  
    TableName: "Music",  
    Item: {  
        "Artist": "The Acme Band",  
        "SongTitle": "Look Out, World",  
        "AlbumTitle": "The Buck Starts Here",  
        "Price": 0.99,  
        "Genre": "Rock"  
    }  
}
```

Note

Além do `PutItem`, o DynamoDB oferece suporte a um `BatchWriteItem`ação para gravar vários itens ao mesmo tempo.

Principais diferenças ao ler dados de uma tabela

Com o SQL, você usa a instrução `SELECT` para recuperar uma ou mais linhas de uma tabela. Use a cláusula `WHERE` para determinar os dados que são retornados para você.

O Amazon DynamoDB oferece as seguintes operações para a leitura de dados:

- `ExecuteStatement`— Recupera um ou vários itens de uma tabela usando o PartiQL (uma linguagem de consulta compatível com SQL). O DynamoDB também oferece a `BatchExecuteStatement`ação, permitindo que você recupere vários itens de tabelas diferentes em uma única operação.
- `GetItem`— recupera um único item de uma tabela. Esta é a maneira mais eficiente de ler um único item porque ele fornece acesso direto à localização física do item. (O DynamoDB também fornece

O `BatchGetItem` operação, permitindo que você realize até 100 `GetItem` chama em uma única operação.)

- `Query`— recupera todos os itens que têm uma chave de partição específica. Nesses itens, você pode aplicar uma condição à chave de classificação e recuperar apenas um subconjunto dos dados. `Query` fornece acesso rápido e eficiente às partições em que os dados são armazenados. (Para obter mais informações, consulte [Partições e distribuição de dados \(p. 23\)](#).)
- `Scan`— recupera todos os itens da tabela especificada. (Essa operação não deve ser usada com tabelas grandes, pois ela pode consumir grandes quantidades de recursos do sistema.)

Note

Com um banco de dados relacional, você pode usar a instrução `SELECT` para juntar os dados de várias tabelas e retornar os resultados. Junções são fundamentais para o modelo relacional. Para garantir que as junções sejam executadas de forma eficiente, o banco de dados e seus aplicativos devem ter o desempenho ajustado de forma contínua. O DynamoDB é um banco de dados NoSQL não relacional que não dá suporte a junções de tabela. Em vez disso, os aplicativos leem dados de uma tabela por vez.

As seções a seguir descrevem diferentes casos de uso para a leitura de dados e como executar essas tarefas com um banco de dados relacional e com o DynamoDB.

Tópicos

- [Leitura de um item usando sua chave primária \(p. 36\)](#)
- [Consultar uma tabela \(p. 38\)](#)
- [Verificação de uma tabela \(p. 40\)](#)

Leitura de um item usando sua chave primária

Um padrão de acesso comuns para bancos de dados é ler um único item de uma tabela. Você precisa especificar a chave primária do item que deseja.

Tópicos

- [SQL \(p. 36\)](#)
- [DynamoDB \(p. 37\)](#)

SQL

No SQL, você usa a instrução `SELECT` para recuperar dados de uma tabela. Você pode solicitar uma ou mais colunas no resultado (ou todas elas, se você usar o operador `*`). A cláusula `WHERE` determina quais linhas devem ser retornadas.

A seguinte é uma instrução `SELECT` para recuperar uma única linha da tabela `Music`. A cláusula `WHERE` especifica os valores de chave primária.

```
SELECT *
FROM Music
WHERE Artist='No One You Know' AND SongTitle = 'Call Me Today'
```

É possível modificar a consulta para recuperar somente um subconjunto de colunas.

```
SELECT AlbumTitle, Year, Price
FROM Music
```

```
WHERE Artist='No One You Know' AND SongTitle = 'Call Me Today'
```

Observe que a chave primária dessa tabela consiste em Artist e SongTitle.

DynamoDB

PartiQL

O DynamoDB oferece o `ExecuteStatement` para recuperar um item por sua chave primária usando a instrução `SELECT`.

```
select AlbumTitle, Year, Price
FROM Music
WHERE Artist='No One You Know' AND SongTitle = 'Call Me Today'
```

Observe que a chave primária dessa tabela consiste em Artist e SongTitle.

Note

A instrução `select` PartiQL pode ser usada também para consultar ou digitalizar uma tabela do DynamoDB

Para obter exemplos de códigos que usam `Select` e `ExecuteStatement`, consulte [Instruções do PartiQL Select para o DynamoDB \(p. 538\)](#).

Classic APIs

O DynamoDB oferece o `GetItem` para recuperar um item por sua chave primária. `GetItem` é altamente eficiente, pois fornece acesso direto à localização física do item. (Para obter mais informações, consulte [Partições e distribuição de dados \(p. 23\)](#).)

Por padrão, `GetItem` retorna todo o item com todos os seus atributos.

```
{
  TableName: "Music",
  Key: {
    "Artist": "No One You Know",
    "SongTitle": "Call Me Today"
  }
}
```

É possível adicionar um parâmetro `ProjectionExpression` para retornar apenas alguns dos atributos.

```
{
  TableName: "Music",
  Key: {
    "Artist": "No One You Know",
    "SongTitle": "Call Me Today"
  },
  "ProjectionExpression": "AlbumTitle, Year, Price"
}
```

Observe que a chave primária dessa tabela consiste em Artist e SongTitle.

O DynamoDB `GetItem` é muito eficiente: Ele usa valores de chave primária para determinar o local de armazenamento exato do item em questão e recupera-o diretamente dali. A instrução SQL `SELECT` é similarmente eficiente, no caso da recuperação de itens por valores de chave primária.

O SQLSELECTO é compatível com muitos tipos de consultas e verificações de tabela. O DynamoDB oferece funcionalidade semelhante com seuQueryeScanAs ações, que são descritas em[Consultar uma tabela \(p. 38\)](#)e[Verificação de uma tabela \(p. 40\)](#).

A instrução SQL SELECT pode realizar junções de tabela, permitindo que você recupere dados de várias tabelas ao mesmo tempo. As junções são mais eficazes onde as tabelas de banco de dados são normalizadas e os relacionamentos entre as tabelas são claros. No entanto, se você juntar muitas tabelas em uma instrução SELECT, o desempenho do aplicativo pode ser afetado. Você pode resolver esses problemas usando replicação de banco de dados, visualizações materializadas ou regravações de consulta.

O DynamoDB é um banco de dados não relacional, e não dá suporte a junções de tabela. Se você estiver migrando um aplicativo existente de um banco de dados relacional para o DynamoDB, será necessário desnormalizar o seu modelo de dados para eliminar a necessidade de junções.

Note

Para obter exemplos de código que usam GetItem, consulte [Conceitos básicos do DynamoDB e do AWSSDKs da \(p. 81\)](#).

Consultar uma tabela

Outro padrão de acesso comum é a leitura de vários itens de uma tabela, com base em seus critérios de consulta.

Tópicos

- [SQL \(p. 38\)](#)
- [DynamoDB \(p. 39\)](#)

SQL

A instrução SQL SELECT permite consultar as colunas-chave, colunas que não são chave ou qualquer combinação. A cláusula WHERE determina quais linhas são retornadas, conforme mostrado nos exemplos a seguir.

```
/* Return a single song, by primary key */

SELECT * FROM Music
WHERE Artist='No One You Know' AND SongTitle = 'Call Me Today';
```

```
/* Return all of the songs by an artist */

SELECT * FROM Music
WHERE Artist='No One You Know';
```

```
/* Return all of the songs by an artist, matching first part of title */

SELECT * FROM Music
WHERE Artist='No One You Know' AND SongTitle LIKE 'Call%';
```

```
/* Return all of the songs by an artist, with a particular word in the title...
...but only if the price is less than 1.00 */

SELECT * FROM Music
WHERE Artist='No One You Know' AND SongTitle LIKE '%Today%'
AND Price < 1.00;
```

Observe que a chave primária dessa tabela consiste em Artist e SongTitle.

DynamoDB

PartiQL

O DynamoDB oferece o `ExecuteStatement` para recuperar vários itens usando a instrução `select`.

```
select AlbumTitle, Year, Price
FROM Music
WHERE Artist='No One You Know'
```

Observe que esta declaração retornará todos os itens para o Artista especificado.

Para obter exemplos de códigos que usam `Select` e `ExecuteStatement`, consulte [Instruções do PartiQL Select para o DynamoDB \(p. 538\)](#).

Classic APIs

Amazon DynamoDB `Query` Ação do permite que você recupere dados de uma forma semelhante. A ação `Query` oferece acesso rápido e eficiente aos locais físicos onde os dados são armazenados. Para mais informações, consulte [Partições e distribuição de dados \(p. 23\)](#).

Você pode usar `Query` com qualquer tabela que tenha uma chave primária composta (chave de partição e chave de classificação). Você deve especificar uma condição de igualdade para a chave de partição e, se desejar, poderá fornecer outra condição para a chave de classificação.

O parâmetro `KeyConditionExpression` especifica os valores de chave que você deseja consultar. Você pode usar uma `FilterExpression` opcional para remover determinados itens dos resultados antes que eles sejam retornados para você.

No DynamoDB, você deve usar `ExpressionAttributeValues` como espaços reservados em parâmetros de expressão (tais como `KeyConditionExpression` e `FilterExpression`). Isso é análogo ao uso de variáveis de ligação em bancos de dados relacionais, onde você substitui os valores reais na instrução `SELECT` em tempo de execução.

Observe que a chave primária dessa tabela consiste em Artist e SongTitle.

Veja os seguintes `Query` Exemplos de código.

```
// Return a single song, by primary key
{
    TableName: "Music",
    KeyConditionExpression: "Artist = :a and SongTitle = :t",
    ExpressionAttributeValues: {
        ":a": "No One You Know",
        ":t": "Call Me Today"
    }
}
```

```
// Return all of the songs by an artist
{
    TableName: "Music",
    KeyConditionExpression: "Artist = :a",
    ExpressionAttributeValues: {
        ":a": "No One You Know"
    }
}
```

```
}
```

```
// Return all of the songs by an artist, matching first part of title
{
    TableName: "Music",
    KeyConditionExpression: "Artist = :a and begins_with(SongTitle, :t)",
    ExpressionAttributeValues: {
        ":a": "No One You Know",
        ":t": "Call"
    }
}
```

Note

Para obter exemplos de código que usam `Query`, consulte [Conceitos básicos do DynamoDB e do AWS SDKs](#) da (p. 81).

Verificação de uma tabela

No SQL, uma instrução `SELECT` sem uma cláusula `WHERE` retornará cada linha em uma tabela. No Amazon DynamoDB, o `Scan` operação executa a mesma ação. Em ambos os casos, é possível recuperar todos os itens ou somente alguns deles.

Se você estiver usando um banco de dados SQL ou um NoSQL, as verificações devem ser usadas com moderação, pois elas podem consumir grandes quantidades de recursos do sistema. Às vezes, uma verificação é apropriada (como a verificação de uma pequena tabela) ou inevitável (como a execução de uma exportação de dados em massa). No entanto, como uma regra geral, você deve projetar seus aplicativos para evitar a execução de verificações.

Tópicos

- [SQL \(p. 40\)](#)
- [DynamoDB \(p. 40\)](#)

SQL

No SQL, você pode verificar uma tabela e recuperar todos os seus dados usando uma instrução `SELECT` sem especificar uma cláusula `WHERE`. Você pode solicitar uma ou mais colunas no resultado. Ou é possível solicitar todas elas se usar o caractere curinga (*).

Veja a seguir exemplos do uso de uma instrução `SELECT`.

```
/* Return all of the data in the table */
SELECT * FROM Music;
```

```
/* Return all of the values for Artist and Title */
SELECT Artist, Title FROM Music;
```

DynamoDB

PartiQL

O DynamoDB oferece o `ExecuteStatement` para retornar todo o conteúdo de uma tabela usando a instrução `select`.

```
select AlbumTitle, Year, Price
```

```
FROM Music
```

Observe que esta declaração retornará todos os itens para na tabela Música.

Para obter exemplos de códigos que usam `SelectStatement`, consulte [Instruções do PartiQL Select para o DynamoDB \(p. 538\)](#).

Classic APIs

O DynamoDB oferece uma `Scan`ação que funciona da mesma forma. Veja os seguintes exemplos.

```
// Return all of the data in the table
{
    TableName: "Music"
}
```

```
// Return all of the values for Artist and Title
{
    TableName: "Music",
    ProjectionExpression: "Artist, Title"
}
```

A ação `Scan` também oferece um parâmetro `FilterExpression`, que pode ser usado para descartar itens que você não deseja que sejam exibidos nos resultados. A `FilterExpression` é aplicado depois que a verificação é executada, mas antes que os resultados sejam retornados para você. (Isso não é recomendado com tabelas grandes. Você ainda é cobrado por todo o `Scan`, mesmo se apenas alguns itens correspondentes forem retornados.)

Note

Para obter exemplos de código que usam `Scan`, consulte [Conceitos básicos do DynamoDB e do AWS SDKs da \(p. 81\)](#).

Gerenciamento de índices

Índices dão a você acesso a padrões de consulta alternativos, e você pode agilizar as consultas. Esta seção compara e contrasta a criação e a utilização de índices no SQL e no Amazon DynamoDB.

Caso esteja usando um banco de dados relacional ou o DynamoDB, você deve ser criterioso com a criação do índice. Sempre que uma gravação ocorre em uma tabela, todos os índices da tabela devem ser atualizados. Em um ambiente que exige muita gravação com tabelas grandes, isso pode consumir grandes quantidades de recursos do sistema. Em um ambiente somente leitura ou em sua maioria de leitura, essa não é uma grande preocupação. Entretanto, você deve garantir que os índices estejam realmente sendo usados por seu aplicativo, e não simplesmente ocupando espaço.

Tópicos

- [Criação de um índice \(p. 41\)](#)
- [Consulta e verificação de um índice \(p. 43\)](#)

Criação de um índice

Compare o `CREATE INDEX` instrução no SQL com o `UpdateTable` operação no Amazon DynamoDB.

Tópicos

- [SQL \(p. 42\)](#)
- [DynamoDB \(p. 42\)](#)

SQL

Em um banco de dados relacional, índice é uma estrutura de dados que permite realizar consultas rápidas em diferentes colunas em uma tabela. Você pode usar a instrução SQL `CREATE INDEX` para adicionar um índice a uma tabela existente, especificando as colunas a serem indexadas. Após a criação do índice, você pode consultar os dados na tabela como sempre, mas agora o banco de dados pode usar o índice para localizar rapidamente as linhas especificadas na tabela, em vez de verificar toda a tabela.

Depois que você cria um índice, o banco de dados o mantém para você. Sempre que você modifica os dados na tabela, o índice é modificado automaticamente para refletir as alterações da tabela.

No MySQL, é possível criar um índice da seguinte forma.

```
CREATE INDEX GenreAndPriceIndex
ON Music (genre, price);
```

DynamoDB

No DynamoDB, você pode criar e usar um índice secundário para fins semelhantes.

Índices no DynamoDB são diferentes dos seus parceiros relacionais. Ao criar um índice secundário, você deve especificar os atributos de chave — uma chave de partição e uma chave de classificação. Depois de criar o índice secundário, você pode query ele ou scan assim como faria com uma mesa. O DynamoDB não tem um otimizador de consulta, portanto, um índice secundário é usado apenas quando você usa a operação `query` ou `scan`.

O DynamoDB é compatível com dois tipos diferentes de índices:

- Os índices secundários globais – a chave primária do índice pode ser qualquer dois atributos de sua tabela.
- Os índices secundários locais – a chave de partição do índice deve ser a mesma que a chave de partição de sua tabela. No entanto, a chave de classificação pode ser qualquer outro atributo.

O DynamoDB garante que os dados em um índice secundário sejam eventualmente consistentes com sua tabela. Você pode solicitar fortemente consistente query ou scan ações em uma tabela ou um índice secundário local. No entanto, índices secundários globais somente oferecem suporte à consistência eventual.

Você pode adicionar um índice secundário global a uma tabela existente, usando o `updateTable` e a especificação `GlobalSecondaryIndexUpdates`.

```
{
    TableName: "Music",
    AttributeDefinitions:[
        {AttributeName: "Genre", AttributeType: "S"},
        {AttributeName: "Price", AttributeType: "N"}
    ],
    GlobalSecondaryIndexUpdates: [
        {
            Create: {
                IndexName: "GenreAndPriceIndex",
                KeySchema: [
                    {AttributeName: "Genre", KeyType: "HASH"}, //Partition key
                    {AttributeName: "Price", KeyType: "RANGE"} //Sort key
                ],
                Projection: {
                    "ProjectionType": "ALL"
                },
            }
        }
    ]
}
```

```
        ProvisionedThroughput: { // Only specified
            if using provisioned mode
                "ReadCapacityUnits": 1, "WriteCapacityUnits": 1
            }
        }
    ]
}
```

Você deve fornecer os parâmetros a seguir para `UpdateTable`:

- `TableName` - a tabela à qual o índice será associado.
- `AttributeDefinitions` - os tipos de dados dos atributos de esquema de chaves do índice.
- `GlobalSecondaryIndexUpdates` - detalhes sobre o índice que você deseja criar:
 - `IndexName` - um nome para o índice.
 - `KeySchema` - os atributos que são usados para a chave primária do índice.
 - `Projection` - atributos da tabela que são copiados para o índice. Neste caso, `ALL` significa que todos os atributos são copiados.
 - `ProvisionedThroughput` (`for provisioned tables`) - o número de leituras e gravações por segundo que você precisa para este índice. (Isso é separado das configurações de throughput provisionado da tabela.)

Parte dessa operação envolve backfilling de dados da tabela para o novo índice. Durante o backfilling, a tabela permanece disponível. No entanto, o índice não está pronto até que seu atributo `Backfilling` mude de verdadeiro para falso. Você pode usar a ação `DescribeTable` para visualizar esse atributo.

Note

Para obter exemplos de código que usam `UpdateTable`, consulte [Conceitos básicos do DynamoDB e do AWSSDKs da \(p. 81\)](#).

Consulta e verificação de um índice

Compare a consulta e a verificação de um índice usando a instrução `SELECT` no SQL com o `Query` e `Scan` no Amazon DynamoDB.

Tópicos

- [SQL \(p. 43\)](#)
- [DynamoDB \(p. 44\)](#)

SQL

Em um banco de dados relacional, você não trabalha diretamente com índices. Em vez disso, você consulta tabelas, emitindo instruções `SELECT`, e o otimizador de consultas pode fazer uso de índices.

Um otimizador de consultas é um componente do sistema de gerenciamento de banco de dados relacional (RDBMS - relational database management system) que avalia os índices disponíveis e determina se eles podem ser usados para agilizar uma consulta. Se os índices puderem ser usados para acelerar uma consulta, o RDBMS acessará o índice primeiro e, em seguida, o usará para localizar os dados na tabela.

Estas são algumas instruções SQL que podem usar `GenreAndPriceIndex` para melhorar o desempenho. Presumimos que a tabela `Music` tenha dados suficientes para que o otimizador de consultas decida usar esse índice, em vez de simplesmente verificar a tabela inteira.

```
/* All of the rock songs */
```

```
SELECT * FROM Music
WHERE Genre = 'Rock';
```

```
/* All of the cheap country songs */

SELECT Artist, SongTitle, Price FROM Music
WHERE Genre = 'Country' AND Price < 0.50;
```

DynamoDB

No DynamoDB, você executa `query` operações diretamente no índice, da mesma forma que faria em uma tabela. Você deve especificar `TableName` e `IndexName`.

Veja os seguintes questionamentos sobre `Genre` and `Price` no DynamoDB. (O esquema de chaves desse índice consiste em `Genre` e `Price`.)

```
// All of the rock songs

{
    TableName: "Music",
    IndexName: "GenreAndPriceIndex",
    KeyConditionExpression: "Genre = :genre",
    ExpressionAttributeValues: {
        ":genre": "Rock"
    },
};
```

```
// All of the cheap country songs

{
    TableName: "Music",
    IndexName: "GenreAndPriceIndex",
    KeyConditionExpression: "Genre = :genre and Price < :price",
    ExpressionAttributeValues: {
        ":genre": "Country",
        ":price": 0.50
    },
    ProjectionExpression: "Artist, SongTitle, Price"
};
```

Este exemplo usa uma `ProjectionExpression` para indicar que somente alguns dos atributos, em vez de todos eles, são exibidos nos resultados.

Você também pode executar `Scan` operações em um índice secundário, da mesma forma como faria em uma tabela. Veja a seguir uma verificação em `GenreAndPriceIndex`.

```
// Return all of the data in the index

{
    TableName: "Music",
    IndexName: "GenreAndPriceIndex"
}
```

Modificação de dados em uma tabela

A linguagem SQL fornece o `UPDATE` instrução para modificar dados. Amazon DynamoDB usa o `UpdateItem` operação para realizar tarefas semelhantes.

Tópicos

- [SQL \(p. 45\)](#)
- [DynamoDB \(p. 45\)](#)

SQL

No SQL, você usa a instrução UPDATE para modificar uma ou mais linhas. A cláusula SET especifica novos valores para uma ou mais colunas, e a cláusula WHERE determina quais linhas são modificadas. Veja um exemplo a seguir.

```
UPDATE Music
SET RecordLabel = 'Global Records'
WHERE Artist = 'No One You Know' AND SongTitle = 'Call Me Today';
```

Se nenhuma linha corresponder à cláusula WHERE, a instrução UPDATE não terá efeito.

DynamoDB

No DynamoDB, você usa o UpdateItem para modificar um único item. (Caso deseje modificar vários itens, você deve usar várias operações UpdateItem.)

Veja um exemplo a seguir.

```
{
    TableName: "Music",
    Key: {
        "Artist": "No One You Know",
        "SongTitle": "Call Me Today"
    },
    UpdateExpression: "SET RecordLabel = :label",
    ExpressionAttributeValues: {
        ":label": "Global Records"
    }
}
```

Você deve especificar o Key atributos do item a ser modificado e um UpdateExpression para especificar valores de atributo. UpdateItem se comporta como uma operação “upsert”. O item é atualizado se existir na tabela, caso contrário, um novo item é adicionado (inserido).

UpdateItem oferece suporte a gravações condicionais, nas quais a operação será bem-sucedida apenas se uma ConditionExpression específica for verdadeira. Por exemplo, a ação UpdateItem a seguir não realiza a atualização, a menos que o preço da música seja maior ou igual a 2,00.

```
{
    TableName: "Music",
    Key: {
        "Artist": "No One You Know",
        "SongTitle": "Call Me Today"
    },
    UpdateExpression: "SET RecordLabel = :label",
    ConditionExpression: "Price >= :p",
    ExpressionAttributeValues: {
        ":label": "Global Records",
        ":p": 2.00
    }
}
```

`UpdateItem` também oferece suporte a contadores atômicos ou a atributos do tipo `Number` que podem ser aumentados ou reduzidos. Os contadores atômicos são semelhantes de muitas maneiras a geradores de sequências, colunas de identidade ou campos de incremento automático em bancos de dados SQL.

Veja a seguir um exemplo de uma ação `UpdateItem` para inicializar um novo atributo (`Plays`) para controlar o número de vezes em que uma música foi reproduzida.

```
{  
    TableName: "Music",  
    Key: {  
        "Artist": "No One You Know",  
        "SongTitle": "Call Me Today"  
    },  
    UpdateExpression: "SET Plays = :val",  
    ExpressionAttributeValues: {  
        ":val": 0  
    },  
    ReturnValue: "UPDATED_NEW"  
}
```

O parâmetro `ReturnValue` é definido como `UPDATED_NEW`, o que retorna os novos valores de todos os atributos que foram atualizados. Neste caso, ele retorna 0 (zero).

Sempre que alguém reproduzir essa música, podemos usar a ação `UpdateItem` a seguir para aumentar `Plays` por um.

```
{  
    TableName: "Music",  
    Key: {  
        "Artist": "No One You Know",  
        "SongTitle": "Call Me Today"  
    },  
    UpdateExpression: "SET Plays = Plays + :incr",  
    ExpressionAttributeValues: {  
        ":incr": 1  
    },  
    ReturnValue: "UPDATED_NEW"  
}
```

Note

Para obter exemplos de código que usam `UpdateItem`, consulte [Conceitos básicos do DynamoDB e do AWS SDKs](#) (p. 81).

Exclusão de dados de uma tabela

No SQL, o comando `DELETE` remove uma ou mais linhas de uma tabela. Amazon DynamoDB usa a operação `DeleteItem` para excluir um item por vez.

Tópicos

- [SQL \(p. 46\)](#)
- [DynamoDB \(p. 47\)](#)

SQL

No SQL, você usa a instrução `DELETE` para excluir uma ou mais linhas. A cláusula `WHERE` determina as linhas que você deseja modificar. Veja um exemplo a seguir.

```
DELETE FROM Music
WHERE Artist = 'The Acme Band' AND SongTitle = 'Look Out, World';
```

Você pode modificar a cláusula `WHERE` para excluir várias linhas. Por exemplo, é possível excluir todas as músicas de um determinado artista, conforme mostrado no exemplo a seguir.

```
DELETE FROM Music WHERE Artist = 'The Acme Band'
```

Note

Se você omitir a cláusula `WHERE`, o banco de dados tenta excluir todas as linhas da tabela.

DynamoDB

No DynamoDB, você usa o `DeleteItem` para excluir dados de uma tabela, um de cada vez. Você deve especificar os valores de chave primária do item.

```
{
  TableName: "Music",
  Key: {
    Artist: "The Acme Band",
    SongTitle: "Look Out, World"
  }
}
```

Note

Além do `DeleteItem`, o Amazon DynamoDB oferece suporte a um `BatchWriteItem` para excluir vários itens ao mesmo tempo.

`DeleteItem` oferece suporte a gravações condicionais, nas quais a operação será bem-sucedida apenas se uma `ConditionExpression` específica for verdadeira. Por exemplo, a ação `DeleteItem` a seguir exclui o item somente se ele tiver um atributo `RecordLabel`.

```
{
  TableName: "Music",
  Key: {
    Artist: "The Acme Band",
    SongTitle: "Look Out, World"
  },
  ConditionExpression: "attribute_exists(RecordLabel)"
}
```

Note

Para obter exemplos de código que usam `DeleteItem`, consulte [Conceitos básicos do DynamoDB e do AWS SDKs](#) (p. 81).

Remoção de uma tabela

No SQL, você usa a instrução `DROP TABLE` para remover uma tabela. No Amazon DynamoDB, você usa o `DeleteTable` operação.

Tópicos

- [SQL \(p. 48\)](#)
- [DynamoDB \(p. 48\)](#)

SQL

Quando você não precisar mais de uma tabela e quiser descartá-la permanentemente, use a instrução `DROP TABLE` no SQL.

```
DROP TABLE Music;
```

Depois que uma tabela é eliminada, ela não pode ser recuperada. (Alguns bancos de dados relacionais permitem que você desfaça uma operação `DROP TABLE`, mas essa funcionalidade é específica do fornecedor e não é amplamente implementada.)

DynamoDB

O DynamoDB tem uma ação semelhante: `DeleteTable`. No exemplo a seguir, a tabela é excluída permanentemente.

```
{
    TableName: "Music"
}
```

Note

Para obter exemplos de código que usam `DeleteTable`, consulte [Conceitos básicos do DynamoDB e do AWSSDKs](#) (p. 81).

Recursos adicionais do Amazon DynamoDB

Tópicos

- [Guias, repositórios e publicações de blog](#) (p. 48)
- [Modelagem de dados e padrões de design](#) (p. 48)
- [Padrões de design avançados com Rick Houlihan](#) (p. 49)
- [Cursos de treinamento](#) (p. 49)
- [Ferramentas para codificação e visualização](#) (p. 49)

Guias, repositórios e publicações de blog

- [Como alternar do RDBMS para o DynamoDB em 20 etapas fáceis](#)— Uma lista irreverente de [Jeremy Daly](#) de etapas úteis para aprender a modelagem de dados.
- [Folha de truques do DynamoDB JavaScript DocumentClient](#)— uma página de dicas para ajudá-lo a começar a criar aplicativos com DynamoDB em um ambiente Node.js ou JavaScript.
- [Do banco de dados relacional à tabela única do DynamoDB: Uma exploração passo a passo](#)— Um post aprofundado de [Forrest Brazeal](#) que inclui uma abordagem passo a passo para a mudança para um design de tabela única no estilo DynamoDB.

Modelagem de dados e padrões de design

- [AWSre:Invent 2019: Modelagem de dados com o DynamoDB](#)— Uma palestra de [Alex DeBrie](#) que ajuda a entender os princípios da modelagem de dados do DynamoDB.

Padrões de design avançados com Rick Houlihan

- [AWSre:Invent 2019: Advanced design patterns](#)
 - Jeremy Daly compartilha [12 key takeaways](#) nesta sessão.
- [AWSre:Invent 2018: Advanced design patterns](#)
- [AWSre:Invent 2017: Advanced design patterns](#)

Note

Cada sessão abrange diferentes casos de uso e exemplos.

Cursos de treinamento

- [Curso de mergulho profundo do DynamoDB](#)— um curso da Linux Academy com ajuda da equipe do Amazon DynamoDB.
- [Amazon DynamoDB: Criando aplicativos orientados por banco de dados NoSQL](#)— Um curso da AWS Equipe de treinamento e certificação hospedada no edX.
- [AWS DynamoDB — do iniciante ao profissional](#)— um curso de A Cloud Guru.

Ferramentas para codificação e visualização

- [NoSQL Workbench para Amazon DynamoDB](#)— uma ferramenta visual unificada que fornece recursos de modelagem de dados, visualização de dados e desenvolvimento de consulta para ajudá-lo a projetar, criar, consultar e gerenciar tabelas do DynamoDB.
- [DynamoDB Toolbox](#)— um projeto de Jeremy Daly, que fornece utilitários úteis para trabalhar com a modelagem de dados em JavaScript e Node.js.
- [Dynobase](#)— uma ferramenta de área de trabalho que facilita a visualização e o trabalho com as tabelas do DynamoDB, a criação de código de aplicativos e a edição de registros com validação em tempo real.
- [DynamoDB Streams Processor](#)— uma ferramenta simples que facilita muito o trabalho com o [DynamoDB Streams](#) super fácil.

Configuração do DynamoDB

Além do serviço Web do Amazon DynamoDB, AWS fornece uma versão disponível para download do DynamoDB que pode executar em seu computador e é perfeita para desenvolvimento e teste de seu código. A versão disponível para download permite gravar e testar aplicativos localmente sem acessar o serviço web DynamoDB.

Os tópicos desta seção descrevem como configurar o DynamoDB (versão disponível para download) e o serviço web do DynamoDB.

Tópicos

- [Configuração do DynamoDB Local \(versão disponível para download\) \(p. 50\)](#)
- [Configuração do DynamoDB \(serviço Web\) \(p. 57\)](#)

Configuração do DynamoDB Local (versão disponível para download)

Com a versão disponível para download do Amazon DynamoDB, você pode desenvolver e testar aplicativos sem acessar o serviço web DynamoDB. Em vez disso, o banco de dados é autossuficiente em seu computador. Quando estiver pronto para implantar seu aplicativo em produção, remova o endpoint local no código e ele apontará para o web service do DynamoDB.

Essa versão local ajuda você a economizar em taxa de transferência, armazenamento de dados e taxas de transferência de dados. Além disso, você não precisa ter uma conexão com a Internet ao desenvolver o aplicativo.

O DynamoDB Local está disponível como um download (requer o JRE), como uma dependência do Apache Maven ou como uma imagem do Docker.

Se você preferir usar o serviço web do Amazon DynamoDB, consulte[Configuração do DynamoDB \(serviço Web\) \(p. 57\)](#).

Tópicos

- [Implantar o DynamoDB localmente em seu computador \(p. 50\)](#)
- [Observações sobre o uso local do DynamoDB \(p. 54\)](#)

Implantar o DynamoDB localmente em seu computador

Download Locally

A versão disponível para download do Amazon DynamoDB é fornecida como um executável .jarfile. O aplicativo é executado no Windows, Linux, macOS e outras plataformas compatíveis com Java.

Siga estas etapas para configurar e executar o DynamoDB em seu computador.

Para configurar o DynamoDB no computador

1. Faça download do DynamoDB gratuitamente em um dos seguintes locais.

Região	Links para fazer download	Somas de verificação
Asia Pacific (Mumbai) Region	.tar.gz .zip	.tar.gz.sha256 .zip.sha256
Asia Pacific (Singapore) Region	.tar.gz .zip	.tar.gz.sha256 .zip.sha256
Asia Pacific (Tokyo) Region	.tar.gz .zip	.tar.gz.sha256 .zip.sha256
Europe (Frankfurt) Region	.tar.gz .zip	.tar.gz.sha256 .zip.sha256
South America (São Paulo) Region	.tar.gz .zip	.tar.gz.sha256 .zip.sha256
US West (Oregon) Region	.tar.gz .zip	.tar.gz.sha256 .zip.sha256

O DynamoDB também está disponível como parte do AWS Toolkit for Eclipse. Para mais informações, consulte [AWS Toolkit for Eclipse](#).

Important

Para executar o DynamoDB em seu computador, é necessário ter o Java Runtime Environment (JRE) versão 8.x ou mais recente. O aplicativo não executa em versões mais antigas do JRE.

2. Depois de fazer download do arquivo, extraia o conteúdo e copie o diretório extraído para um local de sua escolha.
3. Para iniciar o DynamoDB em seu computador, abra uma janela de prompt de comando, navegue até o diretório onde você extraiu o `DynamoDBLocal.jar` e insira o comando a seguir.

```
java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb
```

Note

Se estiver usando o Windows PowerShell, inclua o nome do parâmetro ou todo o nome e valor da seguinte forma:

```
java -D"java.library.path=./DynamoDBLocal_lib" -jar  
DynamoDBLocal.jar
```

O DynamoDB processa as solicitações de entrada até que você o interrompa. Para interromper o DynamoDB, pressione Ctrl+C no prompt de comando.

O DynamoDB usa a porta 8000 por padrão. Se a porta 8000 estiver indisponível, este comando lançará uma exceção. Para obter uma lista completa das opções de tempo de execução do DynamoDB, incluindo `-port`, insira este comando.

```
java -Djava.library.path=./DynamoDBLocal_lib -jar  
DynamoDBLocal.jar -help
```

4. Antes que você possa acessar o DynamoDB de forma programática ou por meio do AWS Command Line Interface(AWS CLI), você deve configurar suas credenciais para habilitar a autorização para seus aplicativos. O DynamoDB disponível para download requer que todas as credenciais funcionem, conforme mostrado no exemplo a seguir.

```
AWS Access Key ID: "fakeMyKeyId"  
AWS Secret Access Key: "fakeSecretAccessKey"
```

Versão da API 2012-08-10

Você pode usar o comando `aws configure` da AWS CLI para configurar credenciais. Para mais informações, consulte [Como usar AWS CLI \(p. 62\)](#).

5. Comece a escrever aplicativos. Para acessar o DynamoDB em execução localmente com o AWS CLI, use `--endpoint-url` parâmetro . Por exemplo, use o comando a seguir para listar as tabelas do DynamoDB.

```
aws dynamodb list-tables --endpoint-url http://localhost:8000
```

Docker

A versão disponível para download do Amazon DynamoDB está disponível como uma imagem do Docker. Para obter mais informações, consulte [dynamodb-local](#).

Para obter um exemplo de uso do DynamoDB local como parte de um aplicativo REST criado com o AWS Serverless Application Model(AWS SAM), consulte [Aplicativo SAM DynamoDB para gerenciamento de pedidos](#). Esse aplicativo de exemplo demonstra como usar o DynamoDB local para testes.

Se você quiser executar um aplicativo de vários contêineres que também use o contêiner local do DynamoDB, use o Docker Compose para definir e executar todos os serviços em seu aplicativo, incluindo o DynamoDB local.

Para instalar e executar o DynamoDB local com o Docker Compose:

1. Faça download e instale o [Docker Desktop](#).
2. Copie o código a seguir para um arquivo e salve-o como `docker-compose.yml`.

```
version: '3.8'
services:
  dynamodb-local:
    command: "-jar DynamoDBLocal.jar -sharedDb -optimizeDbBeforeStartup -dbPath ./data"
    image: "amazon/dynamodb-local:latest"
    container_name: dynamodb-local
    ports:
      - "8000:8000"
    volumes:
      - "./docker/dynamodb:/home/dynamodblocal/data"
    working_dir: /home/dynamodblocal
```

Se você quiser que seu aplicativo e o DynamoDB local estejam em contêineres separados, use o seguinte arquivo yaml.

```
version: '3.8'
services:
  dynamodb-local:
    command: "-jar DynamoDBLocal.jar -sharedDb -optimizeDbBeforeStartup -dbPath ./data"
    image: "amazon/dynamodb-local:latest"
    container_name: dynamodb-local
    ports:
      - "8000:8000"
    volumes:
      - "./docker/dynamodb:/home/dynamodblocal/data"
    working_dir: /home/dynamodblocal
  app-node:
    depends_on:
```

```
- dynamodb-local
image: banst/awscli
container_name: app-node
ports:
- "8080:8080"
environment:
AWS_ACCESS_KEY_ID: 'DUMMYIDEXAMPLE'
AWS_SECRET_ACCESS_KEY: 'DUMMYEXAMPLEKEY'
command:
dynamodb describe-limits --endpoint-url http://dynamodb-local:8000 --region
us-west-2
```

Este script docker-compose.yml cria um contêiner app-node e um contêiner dynamodb-local. O script executa um comando no app-node que usa o AWS CLI para conectar o a dynamodb-local e descreve os limites de conta e tabela.

Para usar com sua própria imagem de aplicativo, substitua a image no exemplo abaixo com o de seu aplicativo.

```
version: '3.8'
services:
dynamodb-local:
command: "-jar DynamoDBLocal.jar -sharedDb -optimizeDbBeforeStartup -dbPath ./
data"
image: "amazon/dynamodb-local:latest"
container_name: dynamodb-local
ports:
- "8000:8000"
volumes:
- "./docker/dynamodb:/home/dynamodblocal/data"
working_dir: /home/dynamodblocal
app-node:
image: location-of-your-dynamodb-demo-app:latest
container_name: app-node
ports:
- "8080:8080"
depends_on:
- "dynamodb-local"
links:
- "dynamodb-local"
environment:
AWS_ACCESS_KEY_ID: 'DUMMYIDEXAMPLE'
AWS_SECRET_ACCESS_KEY: 'DUMMYEXAMPLEKEY'
REGION: 'eu-west-1'
```

Note

Os scripts YAML exigem que você especifique uma AWS chave de acesso e uma AWS chave secreta, mas eles não são obrigados a ser válidos AWS para você acessar o DynamoDB Local.

3. Execute o comando da linha de comando a seguir:

```
docker-compose up
```

Apache Maven

Siga estas etapas para usar o Amazon DynamoDB em seu aplicativo como uma dependência.

Para implantar o DynamoDB como um repositório do Apache Maven

1. Faça download do Apache Maven e instale-o. Para obter mais informações, consulte [Download do Apache Maven e Instalação do Apache Maven](#).
2. Adicione o repositório do DynamoDB Maven ao arquivo Project Object Model (POM) do seu aplicativo.

```
<!--Dependency:-->
<dependencies>
    <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>DynamoDBLocal</artifactId>
        <version>[1.12, 2.0)</version>
    </dependency>
</dependencies>
<!--Custom repository:-->
<repositories>
    <repository>
        <id>dynamodb-local-oregon</id>
        <name>DynamoDB Local Release Repository</name>
        <url>https://s3-us-west-2.amazonaws.com/dynamodb-local/release</url>
    </repository>
</repositories>
```

Note

Você também pode usar um dos seguintes URLs do repositório, dependendo da sua AWS Região :

id	URL de repositório
dynamodb-local-mumbai	https://s3.ap-south-1.amazonaws.com/dynamodb-local-mumbai/release
dynamodb-local-Singapura	https://s3.ap-southeast-1.amazonaws.com/dynamodb-local-singapore/release
dynamodb-local-tokyo	https://s3.ap-northeast-1.amazonaws.com/dynamodb-local-tokyo/release
dynamodb-local-frankfurt	https://s3.eu-central-1.amazonaws.com/dynamodb-local-frankfurt/release
dynamodb-local-sao-paulo	https://s3.sa-east-1.amazonaws.com/dynamodb-local-sao-paulo/release

Observações sobre o uso local do DynamoDB

Exceto para o endpoint, os aplicativos que são executados com a versão disponível para download do Amazon DynamoDB também devem funcionar com o serviço web DynamoDB. No entanto, ao usar o DynamoDB localmente, você deve estar ciente do seguinte:

- Se você usar o `sharedDb`, o DynamoDB cria um único arquivo de banco de dados chamado `shared-local-instance.db`. Todos os programas que se conectarem ao DynamoDB acessarão este arquivo. Se você excluir o arquivo, perderá todos os dados armazenados nele.

- Se você omitir `-sharedDb`, o arquivo de banco de dados é chamado `myaccesskeyid_region.db`, com o AWSID de chave de acesso e AWSRegião como eles aparecem na configuração do aplicativo. Se você excluir o arquivo, perderá todos os dados armazenados nele.
- Se você usar `-inMemory`, o DynamoDB não gravará nenhum arquivo de banco de dados. Em vez disso, todos os dados são gravados na memória, e eles não são salvos ao encerrar o DynamoDB.
- Se você usar `-optimizeDbBeforeStartup`, você também deve especificar `-dbPath` para que o DynamoDB possa encontrar seu arquivo de banco de dados.
- OAWSOs SDKs para DynamoDB exigem que a configuração de seu aplicativo especifique um valor de chave de acesso e um AWSValor da região. A menos que você esteja usando `-sharedDb` ou `-inMemory`, o DynamoDB usará esses valores para nomear o arquivo de banco de dados local. Esses não precisam ser válidos AWSpara executar localmente. No entanto, pode ser conveniente usar valores válidos, de modo que você possa executar o código na nuvem mais tarde alterando o endpoint que você está usando.

Tópicos

- [Opções de linha de comando \(p. 55\)](#)
- [Definição do endpoint local \(p. 56\)](#)
- [Diferenças entre o DynamoDB disponível para download e serviço da Web do DynamoDB \(p. 56\)](#)

Opções de linha de comando

Você pode usar uma das seguintes opções de linha de comando com a versão disponível para download do DynamoDB:

- `-cors value-` ativa o suporte ao CORS (compartilhamento de recursos entre origens) para JavaScript. Você deve fornecer uma lista de "permissão" de domínios específicos separada por vírgulas. A configuração padrão de `-cors` é um asterisco (*), que permite acesso público.
- `-dbPath value-` o diretório em que o DynamoDB grava seu arquivo de banco de dados. Se você não especificar essa opção, o arquivo será gravado no diretório atual. Não é possível especificar `-dbPath` e `-inMemory` ao mesmo tempo.
- `-delayTransientStatuses-` Faz com que o DynamoDB introduza atrasos em determinadas operações. O DynamoDB (versão disponível para download) pode executar algumas tarefas quase instantaneamente, como criar/atualizar/excluir operações em tabelas e índices. No entanto, o serviço do DynamoDB requer mais tempo para essas tarefas. Definir esse parâmetro ajuda o DynamoDB em execução no seu computador a simular o comportamento do serviço web DynamoDB de maneira mais precisa. (No momento, esse parâmetro apresenta atrasos apenas para índices secundários globais que estão no status `CREATING` ou `DELETING`.)
- `-help -l` imprime um resumo de uso e opções.
- `-inMemory-` o DynamoDB é executado na memória, em vez de usar um arquivo de banco de dados. Quando você interrompe o DynamoDB, nenhum dos dados é salvo. Não é possível especificar `-dbPath` e `-inMemory` ao mesmo tempo.
- `-optimizeDbBeforeStartup-` otimiza as tabelas de banco de dados subjacentes antes de iniciar o DynamoDB no seu computador. Você também deve especificar `-dbPath` ao usar este parâmetro.
- `-port value-` o número da porta que o DynamoDB usa para se comunicar com seu aplicativo. Se você não especificar essa opção, a porta padrão será 8000.

Note

O DynamoDB usa a porta 8000 por padrão. Se a porta 8000 estiver indisponível, este comando lançará uma exceção. Você pode usar a opção `-port` para especificar um número de porta diferente. Para obter uma lista completa das opções de tempo de execução do DynamoDB, incluindo `-port`, digite este comando:

```
java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -  
help
```

- **-sharedDb**— Se você especificar `-sharedDb`, o DynamoDB usará um único arquivo de banco de dados, em vez de usar arquivos separados para cada credencial e região.

Definição do endpoint local

Por padrão, oAWSOS SDKs e as ferramentas usam endpoints para o serviço web do Amazon DynamoDB. Para usar os SDKs e as ferramentas com as versões disponíveis para download do DynamoDB, especifique o endpoint local:

`http://localhost:8000`

AWS Command Line Interface

Você pode usar oAWS Command Line Interface(AWS CLI) para interagir com o DynamoDB disponível para download. Por exemplo, use-o para executar todas as etapas em [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#).

Para acessar o DynamoDB em execução localmente, use `--endpoint-url`parâmetro . Veja a seguir um exemplo de uso doAWS CLIPara listar as tabelas no DynamoDB no computador.

```
aws dynamodb list-tables --endpoint-url http://localhost:8000
```

Note

OAWS CLIO não pode usar as versões do DynamoDB disponíveis para download como um endpoint padrão. Portanto, você deve especificar `--endpoint-url` com cada comando na AWS CLI.

SDKs da AWS

A maneira como você especifica um endpoint depende da linguagem de programação e daAWSSDK que você está usando. As seções a seguir descrevem como fazer isso:

- [Java: Configurar oAWSRegião e endpoints \(p. 336\)](#)(O DynamoDB Local oferece suporte aoAWSSDK for Java V1 e V2)
- [.NET: Configurar oAWSRegião e endpoints \(p. 338\)](#)

Note

Para obter exemplos de outras linguagens de programação, consulte [Conceitos básicos do DynamoDB e doAWSSDKs da \(p. 81\)](#).

Diferenças entre o DynamoDB disponível para download e serviço da Web do DynamoDB

A versão para download do DynamoDB são destinadas somente para testes e desenvolvimento. Em comparação, o serviço web DynamoDB é gerenciado com recursos de escalabilidade, disponibilidade durabilidade, que o tornam ideal para uso na fase de produção.

A versão disponível para download do DynamoDB difere do serviço web das seguintes formas:

- AWS Regiões e distintas AWS Contas não são aceitas no nível do cliente.
- As configurações de throughput provisionado são ignoradas no DynamoDB disponível para download, embora a `CreateTable` operação requer eles. Para `CreateTable`, você pode especificar quaisquer números desejados para o throughput de leitura e gravação provisionado, mesmo que esses números não sejam usados. Você pode chamar `UpdateTable` quantas vezes quiser por dia. No entanto, todas as alterações nos valores do throughput provisionado são ignoradas.
- As operações `Scan` são executadas sequencialmente. Verificações paralelas não são aceitas. Os parâmetros `Segment` e `TotalSegments` da operação `Scan` são ignorados.
- A velocidade das operações de leitura e gravação nos dados da tabela é limitada apenas pela velocidade do computador. As operações `CreateTable`, `UpdateTable` e `DeleteTable` ocorrem imediatamente, e o estado da tabela é sempre ACTIVE. As operações `UpdateTable` que alteram apenas as configurações de throughput provisionado em tabelas ou índices secundários globais ocorrem imediatamente. Se uma operação `UpdateTable` criar ou excluir qualquer índice secundário global, esses índices farão uma transição pelos estados normais (como CREATING e DELETING, respectivamente) antes de passarem para um estado ACTIVE. A tabela permanece no estado ACTIVE durante esse tempo.
- As operações de leitura são eventualmente consistentes. No entanto, devido à velocidade do DynamoDB que está em execução no seu computador, a maioria das leituras parecerão ser fortemente consistentes.
- As métricas de coleção de itens e os tamanhos de coleção de itens não são controlados. Nas respostas de operação, nulos são retornados em vez de métricas de coleção de itens.
- No DynamoDB, há um limite de 1 MB em dados retornados por conjunto de resultados. O serviço web e a versão disponível para download do DynamoDB impõem esse limite. No entanto, ao consultar um índice, o serviço DynamoDB calcula apenas o tamanho da chave e dos atributos projetados. Por outro lado, a versão disponível para download do DynamoDB calcula o tamanho do item inteiro.
- Se você estiver usando DynamoDB Streams, a taxa na qual estilhaços são criados pode ser diferente. No serviço web DynamoDB, o comportamento de criação de estilhaço é parcialmente influenciado pela atividade de partição da tabela. Quando o DynamoDB é executado localmente, não há particionamento de tabelas. Em qualquer um dos casos, os estilhaços são efêmeros, portanto, seu aplicativo não deve ser dependente do comportamento do estilhaço.
- `TransactionConflictExceptions` não são lançadas pelo DynamoDB disponível para download para APIs transacionais. Recomendamos o uso de uma estrutura de imitação Java para simular `TransactionConflictExceptions` no manipulador do DynamoDB para testar como o aplicativo responde a transações em conflito.
- No serviço web do DynamoDB, os nomes de tabelas diferenciam maiúsculas de minúsculas. Uma tabela chamada `Authors` e outra chamada `authors` podem existir como tabelas separadas. Na versão que pode ser obtida por download, os nomes de tabelas não diferenciam maiúsculas de minúsculas, e a criação dessas duas tabelas resulta em erro.

Configuração do DynamoDB (serviço Web)

Para usar o serviço Web do Amazon DynamoDB:

1. [Cadastre-se no AWS. \(p. 58\)](#)
2. [Obtenha um AWS chave de acesso \(p. 58\)](#)(usado para acessar o DynamoDB programaticamente).

Note

Se você planeja interagir com o DynamoDB somente por meio do AWS Management Console, você não precisa de um AWS chave de acesso da e você pode ir direto para o [Usar o console \(p. 60\)](#).

3. [Configurar suas credenciais da \(p. 59\)](#)(usado para acessar o DynamoDB programaticamente).

Cadastrar-se no AWS

Para usar o serviço do DynamoDB, você deve ter umAWSconta. Se você ainda não tiver uma conta, será solicitado a criar uma ao se cadastrar. Você não é cobrado por qualquerAWSServiços nos quais se cadastrou a menos que você os utilize.

Para cadastrar-se na AWS

1. Abra <https://portal.aws.amazon.com/billing/signup>.
2. Siga as instruções online.

Parte do procedimento de cadastro envolve uma chamada telefônica e a digitação de um código de verificação usando o teclado do telefone.

Obter umAWSChave de acesso

Antes que você possa acessar o DynamoDB de forma programática ou por meio doAWS Command Line Interface(AWS CLI), você deve ter umAWSChave de acesso da . Você não precisa de uma chave de acesso se planejar usar apenas o console do DynamoDB.

As chaves de acesso consistem em um ID de chave de acesso e uma chave de acesso secreta, usados para assinar solicitações programáticas feitas por você aoAWS. Se não tiver chaves de acesso, você poderá criá-las a partir da guiaAWS Management Console. Como prática recomendada, não use aAWSPara qualquer tarefa em que ele não seja necessário. Em vez disso,Criar um novo usuário do IAM do administradorcom as chaves de acesso para si mesmo.

A única vez que você pode visualizar ou fazer download da chave de acesso secreta é quando você cria as chaves. Não será possível recuperá-las posteriormente. No entanto, você pode criar novas chaves de acesso a qualquer momento. Você também deve ter permissões para executar as ações do IAM necessárias. Para obter mais informações, consultePermissões necessárias para acessar os recursos do IAMnoGuia do usuário do IAM.

Para criar chaves de acesso para um usuário do IAM

1. Faça login no AWS Management Console e abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. No painel de navegação, escolha Usuários.
3. Escolha o nome do usuário cujas chaves de acesso você deseja criar e a caixa de seleçãoCredenciais de segurançaGuia.
4. NoChaves de acesso, selecioneCriar chave de acesso.
5. Para ver o novo key pair de acesso, selecioneMostrar. Você não terá mais acesso à chave de acesso secreta depois que essa caixa de diálogo for fechada. Suas credenciais terão a seguinte aparência:
 - ID de chave de acesso: AKIAIOSFODNN7EXAMPLE
 - Chave de acesso secreta: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. Para baixar o par de chaves, escolha Baixar arquivo .csv. Armazene as chaves em um lugar seguro. Você não terá mais acesso à chave de acesso secreta depois que essa caixa de diálogo for fechada.

Mantenha a confidencialidade das chaves para proteger suaAWSconta e nunca as envie por e-mail. Não compartilhe as chaves fora da sua organização, mesmo se uma pesquisa parecer vir da AWS ou da Amazon.com. Alguém que legitimamente represente a Amazon jamais pedirá a você sua chave secreta.

7. Depois de baixar o .csv, escolhaFechar. Quando você cria uma chave de acesso, o par de chaves é ativo por padrão, e você pode usar o par imediatamente.

Tópicos relacionados

- [O que é o IAM?](#) no Guia do usuário do IAM
- [AWS Credenciais de segurança](#) da em AWS Referência geral

Configurar suas credenciais

Antes que você possa acessar o DynamoDB de forma programática ou por meio do AWS CLI, você deve configurar suas credenciais para habilitar a autorização para seus aplicativos.

Há várias maneiras de fazer isso. Por exemplo, é possível criar manualmente o arquivo de credenciais para armazenar o ID da chave de acesso e a chave de acesso secreta. Também é possível usar o comando `aws configure` da AWS CLI para criar o arquivo automaticamente. Como alternativa, você pode usar variáveis de ambiente. Para obter mais informações sobre como configurar suas credenciais, consulte [AWS Guia do desenvolvedor do SDK](#).

Para instalar e configurar a AWS CLI, consulte [Como usar AWS CLI \(p. 62\)](#).

Acessar o DynamoDB

Você pode acessar o Amazon DynamoDB usando oAWS Management Console, oAWS Command Line Interface(AWS CLI) ou a API do DynamoDB.

Tópicos

- [Usar o console \(p. 60\)](#)
- [Como usar AWS CLI \(p. 62\)](#)
- [Uso da API \(p. 64\)](#)
- [Como usar o NoSQL Workbench para o DynamoDB \(p. 64\)](#)
- [Intervalos de endereços IP \(p. 64\)](#)

Usar o console

Você pode acessar oAWS Management Consolepara o Amazon DynamoDB em<https://console.aws.amazon.com/dynamodb/home>.

Você pode usar o console para fazer o seguinte no DynamoDB:

- Monitore alertas recentes, a capacidade total, a integridade do serviço e as mais recentes novidades do DynamoDB no painel do DynamoDB.
- Criar, atualizar e excluir tabelas. A calculadora de capacidade fornece estimativas de quantas unidades de capacidade devem ser solicitadas com base nas informações de uso que você fornece.
- Gerencie os streams.
- Visualize, adicione, atualize e exclua itens que estão armazenados em tabelas. Gerencie o tempo de vida (TTL) para definir quando os itens em uma tabela expiram para que eles possam ser excluídos automaticamente do banco de dados.
- Consulta e verificação de uma tabela.
- Configure e visualize os alarmes para monitorar a utilização de capacidade da tabela. Visualize as principais métricas de monitoramento da sua tabela em gráficos em tempo real do CloudWatch.
- Modifique a capacidade provisionada de uma tabela.
- Crie e exclua índices secundários globais.
- Crie triggers para conectar streams do DynamoDB aoAWS LambdaFunções do .
- Aplique tags aos seus recursos para ajudar a organizá-los e identificá-los.
- Adquira capacidade reservada.

O console exibe uma tela de introdução que solicita a criação da sua primeira tabela. Para visualizar suas tabelas, no painel de navegação no lado esquerdo do console, escolha Tables (Tabelas).

Veja a visão geral de alto nível das ações disponíveis por tabela dentro de cada guia de navegação:

- Visão geralVisualize detalhes de streams e tabelas, e gerencie streams e tempo de vida (TTL).
- Itens – gerencie itens e execute consultas e verificações.
- Métricas— monitore métricas do Amazon CloudWatch.
- Alarmes – gerencie alarmes do CloudWatch.
- Capacidade – modifique a capacidade provisionada de uma tabela.
- Índices— gerencie índices secundários globais.

- Triggers – gerencie triggers para conectar streams do DynamoDB a funções do Lambda.
- Controle de acesso – configure um controle de acesso minucioso com federação de identidades da web.
- Tags – aplique tags aos seus recursos para ajudar a organizá-los e identificá-los.

Trabalhar com preferências do usuário

Você pode configurar algumas das configurações padrão no console do Amazon DynamoDB. Por exemplo, você pode alterar o tipo de consulta padrão usado quando você acessa a guia Items (Itens). Se tiver entrado no console como um usuário do IAM, você poderá armazenar informações sobre como prefere usar o console. Essas informações, também conhecidas como suas preferências de usuário, são armazenadas e aplicadas sempre que você usa o console. Sempre que você acessa o console do DynamoDB, essas preferências são aplicadas a todas as tabelas em todos os AWSRegiões para seu usuário do IAM. Elas não são específicas da tabela ou da região. Elas não têm efeito sobre suas interações com o AWS CLI, a API do DynamoDB ou outros serviços que interagem com o DynamoDB.

Você ainda pode alterar configurações individuais nas páginas do console sem ter preferências de usuário salvas. Essas opções permanecerão até você fechar a janela do console. Quando retornar ao console, todas as preferências de usuário salvas serão aplicadas.

Note

As preferências de usuário só estão disponíveis para usuários do IAM. Não é possível definir preferências se usar o acesso federado, o acesso temporário ou um AWSpara acessar o console.

As preferências de usuário incluem as seguintes opções:

- Modo de visualização de detalhes da tabela: Visualize todas as informações específicas à tabela no modo vertical, horizontal ou de tela cheia (se habilitada, a barra de navegação ainda será exibida).
- Mostrar barra de navegação: Habilite essa opção para visualizar a barra de navegação no lado esquerdo (expandida). Ao desabilitá-la, a barra de navegação é recolhida automaticamente (é possível expandi-la com o chevron certo).
- Default entry page (Dashboard or Tables): Selecione a página que será carregada quando você acessar o DynamoDB. Esta opção carrega automaticamente a página do painel ou das tabelas, respectivamente.
- Modo de editor de itens (Árvore ou Texto): Escolha o modo de editor padrão a ser usado ao criar ou editar um item.
- Tipo de consulta padrão de itens (Verificação ou Consulta): Selecione o tipo de consulta padrão a ser usado ao acessar o ItemsGuia. Selecione Scan (Verificar) para habilitar ou desabilitar a operação de verificação automática que ocorre ao acessar a guia Items (Itens).
- Automatic scan operation when access the items tab (Operação de verificação automática SeScanComo o tipo de consulta padrão de itens, e você habilitar essa configuração, uma operação de verificação automática ocorrerá ao acessar o ItemsGuia. Se você desabilitar essa configuração, será possível executar uma verificação selecionando Start search (Iniciar pesquisa) na guia Items (Itens).

Visualizar e salvar as preferências do usuário

Você pode visualizar e alterar suas preferências de usuário para o console do DynamoDB. Essas configurações aplicam-se apenas a seu usuário do IAM no console do DynamoDB. Elas não têm efeito sobre outros usuários do IAM em seu AWSconta.

Como visualizar e salvar as preferências no console do DynamoDB para o usuário do IAM do

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.

Faça login como um usuário do IAM. Não é possível configurar as preferências de usuário para outros tipos de usuários.

2. Na navegação da página, escolha Preferences (Preferências).
3. Em Preferences (Preferências), configure suas preferências.

Faça uma das coisas a seguir:

- Para salvar e aplicar suas alterações, escolha Save (Salvar).
- Para visualizar as configurações padrão do console do DynamoDB, selecione Restaurar. Esses padrões serão aplicados se você escolher Save (Salvar).

As configurações padrão são:

- Modo de visualização de detalhes da tabela: Vertical
- Mostrar barra de navegação: Sim
- Default entry page: Painel
- Modo de editor de itens: Árvore
- Itens padrão de consulta do tipo de itens Scan
- Automatic scan operation when access the items tab (Operação de verificação automática) Sim
- Para retornar à página do console na qual você estava anteriormente, clique em Back (Voltar). Ou escolha Painel para acessar a página do DynamoDB Dashboard.

Como usar AWS CLI

Você pode usar o AWS Command Line Interface(AWS CLI) para controlar múltiplos AWSA partir da linha de comando e automatize-os usando scripts do. Use o AWS CLI para operações ad hoc, como a criação de uma tabela. Também é possível usá-la para operações do incorporadas no Amazon DynamoDB em scripts.

Antes de poder usar o AWS CLICom o DynamoDB, é necessário obter um ID de chave de acesso e uma chave de acesso secreta. Para mais informações, consulte [Obter um AWS Chave de acesso \(p. 58\)](#).

Para obter uma lista completa de todos os comandos disponíveis para o DynamoDB no AWS CLI, consulte [a AWS CLI Referência de comandos da](#).

Tópicos

- [Download e configuração da AWS CLI \(p. 62\)](#)
- [Usar o AWS CLI com DynamoDB \(p. 62\)](#)
- [Usar o AWS CLI com DynamoDB para download \(p. 64\)](#)

Download e configuração da AWS CLI

O AWS CLIO está disponível em.<http://aws.amazon.com/cli>. Ela é executada no Windows, macOS ou Linux. Depois de fazer download da AWS CLI, siga estas etapas para instalá-la e configurá-la:

1. Acesse o [AWS Command Line Interface Guia do usuário](#).
2. Siga as instruções para o [Instalar a AWS CLI e Configuração do AWS CLI](#).

Usar o AWS CLI com DynamoDB

O formato de linha de comando consiste em um nome de operação do DynamoDB, seguido pelos parâmetros dessa operação. O AWS CLI suporta uma sintaxe abreviada para os valores de parâmetro, assim como JSON.

Por exemplo, o comando a seguir cria uma tabela chamada Music. A chave de partição é Artist, e a chave de classificação é SongTitle. (Para facilitar a leitura, comandos longos nesta seção são divididos em linhas separadas.)

```
aws dynamodb create-table \
--table-name Music \
--attribute-definitions \
    AttributeName=Artist,AttributeType=S \
    AttributeName=SongTitle,AttributeType=S \
--key-schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE \
--provisioned-throughput ReadCapacityUnits=1,WriteCapacityUnits=1
```

Os comandos a seguir adicionam novos itens à tabela. Esses exemplos usam uma combinação de sintaxe abreviada e JSON.

```
aws dynamodb put-item \
--table-name Music \
--item \
    '{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Call Me Today"}, \
"AlbumTitle": {"S": "Somewhat Famous"}}' \
--return-consumed-capacity TOTAL

aws dynamodb put-item \
--table-name Music \
--item '{ \
    "Artist": {"S": "Acme Band"}, \
    "SongTitle": {"S": "Happy Day"}, \
    "AlbumTitle": {"S": "Songs About Life"} }' \
--return-consumed-capacity TOTAL
```

Na linha de comando, pode ser difícil compor um JSON válido. No entanto, a AWS CLI pode ler arquivos JSON. Por exemplo, considere o seguinte trecho de código JSON, que é armazenado em um arquivo chamado key-conditions.json.

```
{
    "Artist": {
        "AttributeValueList": [
            {
                "S": "No One You Know"
            }
        ],
        "ComparisonOperator": "EQ"
    },
    "SongTitle": {
        "AttributeValueList": [
            {
                "S": "Call Me Today"
            }
        ],
        "ComparisonOperator": "EQ"
    }
}
```

Agora você pode emitir uma solicitação Query usando a AWS CLI. Neste exemplo, o conteúdo do arquivo key-conditions.json é usado para o parâmetro --key-conditions.

```
aws dynamodb query --table-name Music --key-conditions file://key-conditions.json
```

Usar oAWS CLICom DynamoDB para download

OAWS CLITambém podem interagir com o DynamoDB (versão disponível para download) que é executado no seu computador. Para ativar isso, adicione o parâmetro a seguir em cada comando:

```
--endpoint-url http://localhost:8000
```

O exemplo a seguir usa a AWS CLI para listar as tabelas em um banco de dados local.

```
aws dynamodb list-tables --endpoint-url http://localhost:8000
```

Se o DynamoDB estiver usando um número de porta diferente do padrão (8000), modifique a propriedade--endpoint-urlValor adequadamente.

Note

OAWS CLIO não pode usar a versão do DynamoDB disponível para download como um endpoint padrão. Portanto, você deve especificar --endpoint-url com cada comando.

Uso da API

Você pode usar oAWS Management ConsoleO e aAWS Command Line Interfacepara trabalhar interativamente com o Amazon DynamoDB. No entanto, para aproveitar ao máximo o DynamoDB, você pode escrever o código do aplicativo usando oAWSSDKs.

OAWSSDKs fornecem um amplo suporte ao DynamoDB em[Java](#),[JavaScript](#) no [navegador](#),[.NET](#),[Node.js](#),[PHP](#),[Python](#),[Ruby](#),[C++](#),[Go](#),[Android](#), e[iOS](#). Para começar a trabalhar rapidamente com essas linguagens, consulte [Conceitos básicos do DynamoDB e do AWSSDKs](#) da (p. 81).

Antes de poder usar oAWSSDKs com o DynamoDB, você deve obter umAWSID de chave de acesso e chave de acesso secreta. Para mais informações, consulte [Configuração do DynamoDB \(serviço Web\)](#) (p. 57).

Para obter uma visão geral de alto nível da programação de aplicativos do DynamoDB com oAWSSDKs, consulte[Programação com o DynamoDB e o AWSSDKs](#) da (p. 211).

Como usar o NoSQL Workbench para o DynamoDB

Você também pode acessar o DynamoDB, usando[download](#) e usar o[NoSQL Workbench para o DynamoDB](#) (p. 830).

Intervalos de endereços IP

Amazon Web Services (AWS) publica seus intervalos de endereços IP atuais em formato JSON. Para visualizar os intervalos atuais, faça download do arquivo [ip-ranges.json](#). Para obter mais informações, consulte[AWS Intervalos de endereços IP](#)na AWS Referência geral.

Para encontrar os intervalos de endereços IP que podem ser usados para [acessar os índices e as tabelas do DynamoDB](#), procure no arquivo ip-ranges.json a seguinte string: "service": "DYNAMODB".

Note

Os intervalos de endereços IP não se aplicam ao DynamoDB Streams ou ao DynamoDB Accelerator (DAX).

Conceitos básicos do DynamoDB

Use os tutoriais práticos nesta seção para começar a usar o e saber mais sobre o Amazon DynamoDB.

Tópicos

- [Conceitos básicos no DynamoDB \(p. 65\)](#)
- [Pré-requisitos - tutorial de conceitos básicos \(p. 65\)](#)
- [Etapa 1: Criar uma tabela \(p. 65\)](#)
- [Etapa 2: Gravar dados em uma tabela usando o console do ou aAWS CLI \(p. 68\)](#)
- [Etapa 3: Ler dados de uma tabela \(p. 70\)](#)
- [Etapa 4: Atualizar dados em uma tabela \(p. 71\)](#)
- [Etapa 5: Consultar dados em uma tabela \(p. 73\)](#)
- [Etapa 6: Criar um índice secundário global \(p. 75\)](#)
- [Etapa 7: Consultar o índice secundário global \(p. 78\)](#)
- [Etapa 8 Limpar recursos \(p. 79\)](#)
- [Conceitos básicos do DynamoDB: Próximas etapas \(p. 80\)](#)

Conceitos básicos no DynamoDB

Antes de começar, é fundamental conhecer os conceitos básicos do Amazon DynamoDB. Para obter mais informações, consulte[Componentes principais do DynamoDB](#).

Depois, vá para[Pré-requisitos](#)para saber mais sobre como configurar o DynamoDB.

Pré-requisitos - tutorial de conceitos básicos

Antes de iniciar o tutorial do Amazon DynamoDB, siga estas etapas em[Configurar o DynamoDB](#). Depois, vá para [Etapa 1: Criar uma tabela \(p. 65\)](#).

Note

- Se você planeja interagir com o DynamoDB somente por meio do AWS Management Console, você não precisa de AWS Chave de acesso da . Siga as etapas em[Como se cadastrar na AWS](#), em seguida, vá para[Etapa 1: Criar uma tabela \(p. 65\)](#).
- Se você não quer se cadastrar em uma conta de nível gratuito, pode fazer a [Configuração do DynamoDB local \(versão disponível para download\)](#). Depois, vá para [Etapa 1: Criar uma tabela \(p. 65\)](#).

Etapa 1: Criar uma tabela

Nesta etapa, você cria um `Musictable` no Amazon DynamoDB. Essa tabela tem os seguintes detalhes:

- Chave de partição —`Artist`
- Chave de classificação —`SongTitle`

Para obter mais informações sobre operações em tabelas, consulte [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#).

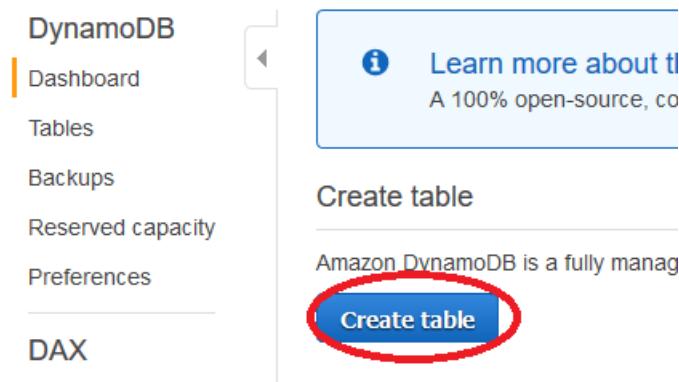
Note

Antes de começar, siga as etapas em [Pré-requisitos - tutorial de conceitos básicos \(p. 65\)](#).

AWS Management Console

Para criar um novo **Music** Tabela usando o console do DynamoDB:

1. Faça login no AWS Management Console e abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Dashboard (Painel).
3. No lado direito do console, escolha Create Table (Criar tabela).



4. Insira os detalhes da tabela assim:
 - a. Para o nome da tabela, insira **Music**.
 - b. Para a chave de partição, insira **Artist**.
 - c. Escolha Add sort key (Adicionar chave de classificação).
 - d. Insira **SongTitle** como a chave de classificação.
5. Escolha Create (Criar) para criar a tarefa.

The screenshot shows the 'Create DynamoDB table' wizard. In the 'Table settings' section, there's a note that says: 'Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.' Below this, there's a checkbox for 'Use default settings' followed by a list of three items: 'No secondary indexes.', 'Auto Scaling capacity set to 70% target utilization, at minimum capacity of 5 reads and 5 writes.', and 'Encryption at Rest with DEFAULT encryption type.' At the bottom, there are 'Cancel' and 'Create' buttons.

AWS CLI

O exemplo da AWS CLI a seguir cria uma nova tabela Music usando `create-table`.

```
aws dynamodb create-table \  
    --table-name Music \  
    --attribute-definitions \  
        AttributeName=Artist,AttributeType=S \  
        AttributeName=SongTitle,AttributeType=S \  
    --key-schema \  
        AttributeName=Artist,KeyType=HASH \  
        AttributeName=SongTitle,KeyType=RANGE \  
    --provisioned-throughput \  
        ReadCapacityUnits=10,WriteCapacityUnits=5
```

O uso de `create-table` retorna o seguinte resultado de exemplo.

```
{  
    "TableDescription": {  
        "TableArn": "arn:aws:dynamodb:us-west-2:522194210714:table/Music",  
        "AttributeDefinitions": [  
            {  
                "AttributeName": "Artist",  
                "AttributeType": "S"  
            },  
            {  
                "AttributeName": "SongTitle",  
                "AttributeType": "S"  
            }  
        ],  
        "ProvisionedThroughput": {  
            "NumberOfDecreasesToday": 0,  
            "WriteCapacityUnits": 5,  
            "ReadCapacityUnits": 10  
        },  
        "TableSizeBytes": 0,  
        "TableName": "Music",  
        "TableStatus": "CREATING",  
        "TableId": "d04c7240-0e46-435d-b231-d54091fe1017",  
        "KeySchema": [  
            {  
                "KeyType": "HASH",  
                "AttributeName": "Artist"  
            },  
            {  
                "KeyType": "RANGE",  
                "AttributeName": "SongTitle"  
            }  
        ],  
        "ItemCount": 0,  
        "CreationDateTime": 1558028402.69  
    }  
}
```

Observe que o valor do campo `TableStatus` está definido como `CREATING`.

Para verificar se o DynamoDB concluiu a criação do `Music`, use a `tabledescribe-table` Comando da.

```
aws dynamodb describe-table --table-name Music | grep TableStatus
```

Esse comando retorna o seguinte resultado. Quando o DynamoDB conclui a criação da tabela, o valor do `TableStatus` campo está definido como `ACTIVE`.

```
"TableStatus": "ACTIVE",
```

Após criar uma nova tabela, vá para [Etapa 2: Gravar dados em uma tabela usando o console do ou aAWS CLI \(p. 68\)](#).

Etapa 2: Gravar dados em uma tabela usando o console do ou aAWS CLI

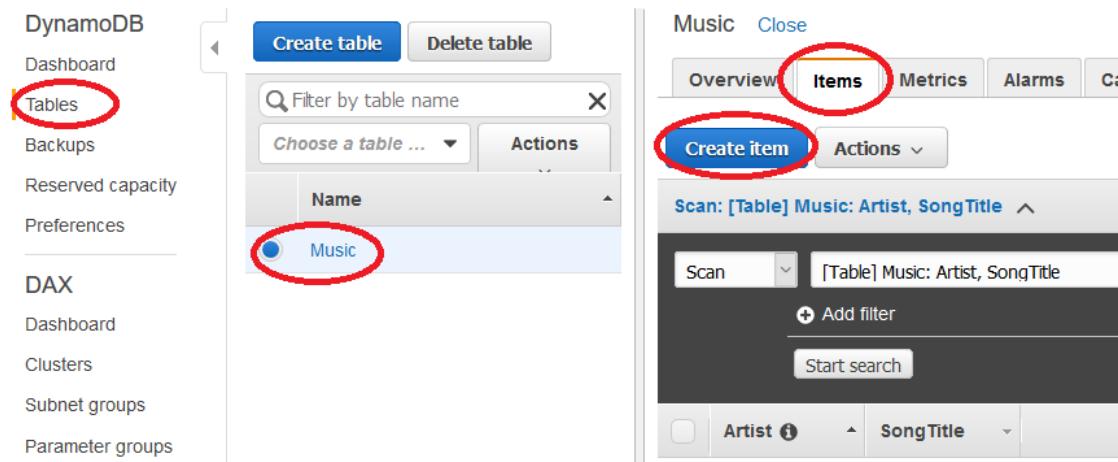
Nesta etapa, você insere dois itens na tabela Music que criou em [Etapa 1: Criar uma tabela \(p. 65\)](#).

Para obter mais informações sobre operações de gravação, consulte [Gravação de um item \(p. 406\)](#).

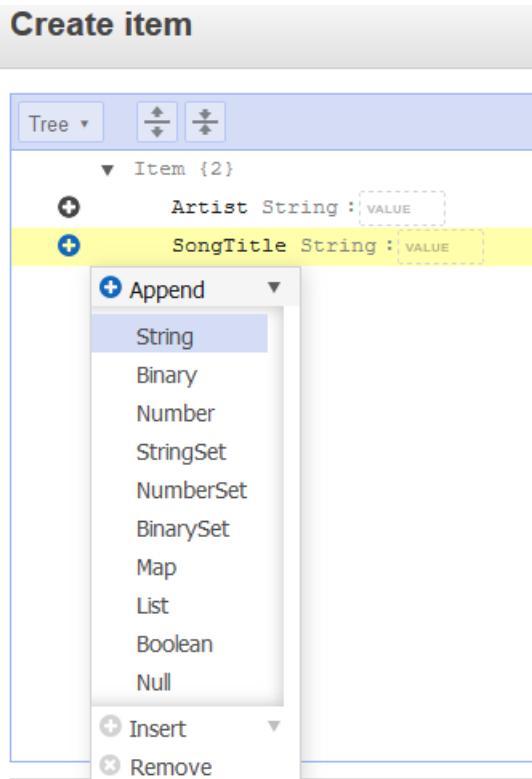
AWS Management Console

Siga estas etapas para gravar dados no Music tabela usando o console do DynamoDB.

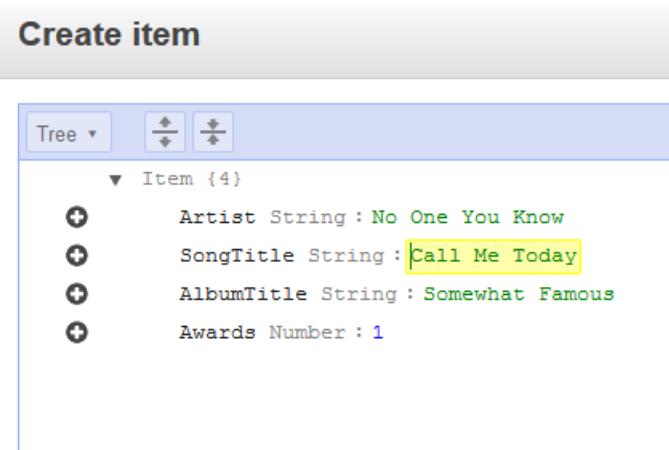
1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Na lista de tabelas, escolha a tabela Music (Música).
4. Escolha a guia Items (Itens) para a tabela Music (Música).
5. Na guia Items (Itens), escolha Create item (Criar item).



6. Escolha o sinal de adição (+) ao lado de SongTitle (Nome da música).
7. Escolha Append (Anexar) e Number (Número). Nomeie o campo como Awards (Prêmios).



8. Repita o processo para criar o AlbumTitle (Nome do álbum) do tipo String.
9. Escolha os seguintes valores para seu item:
 - a. Em Artist (Artista), insira **No One You Know** como o valor.
 - b. Em SongTitle (Nome da música), insira **Call Me Today**.
 - c. Em AlbumTitle (Nome do álbum), insira **Somewhat Famous**.
 - d. Em Awards (Prêmios), insira **1**.
10. Escolha Save (Salvar).



11. Repita o processo para criar outro item com os seguintes valores:
 - a. Em Artist (Artista), insira **Acme Band**.

- b. Em SongTitle (Nome da música) insira **Happy Day**.
- c. Em AlbumTitle (Nome do álbum), insira **Songs About Life**.
- d. Em Awards (Prêmios), insira **10**.

AWS CLI

O exemplo da AWS CLI a seguir cria dois itens novos na tabela **Music** usando `put-item`.

```
aws dynamodb put-item \
  --table-name Music \
  --item \
    '{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Call Me Today"}, \
"AlbumTitle": {"S": "Somewhat Famous"}, "Awards": {"N": "1"}}'

aws dynamodb put-item \
  --table-name Music \
  --item \
    '{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}, "AlbumTitle": \
"S": "Songs About Life"}, "Awards": {"N": "10"} }'
```

Para obter mais informações sobre tipos de dados compatíveis com o DynamoDB, consulte[Tipos de dados](#).

Para obter mais informações sobre como representar os tipos de dados do DynamoDB no JSON, consulte[Valores de atributo](#).

Após gravar os dados na sua tabela, vá para [Etapa 3: Ler dados de uma tabela \(p. 70\)](#).

Etapa 3: Ler dados de uma tabela

Nesta etapa, você lerá um item que foi criado em [Etapa 2: Gravar dados em uma tabela usando o console do ou aAWS CLI \(p. 68\)](#). É possível usar o console do DynamoDB ou oAWS CLIpara ler um item do**Music**, especificando**Artist**e**SongTitle**.

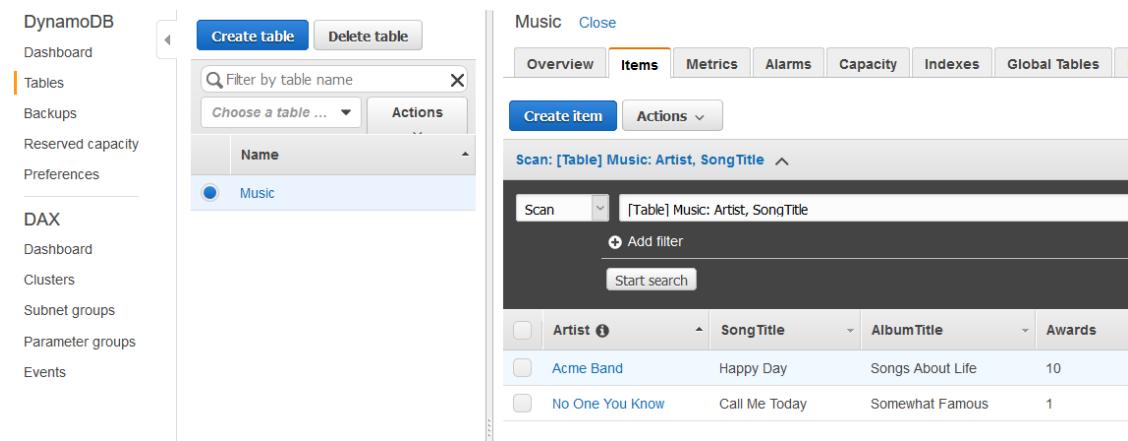
Para obter mais informações sobre as operações de leitura no DynamoDB, consulte[Leitura de um item \(p. 405\)](#).

AWS Management Console

Siga estas etapas para ler os dados do**Music**tabela usando o console do DynamoDB.

1. Abra o console do DynamoDB em<https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Escolha a tabela **Music** (Música) na lista de tabelas.
4. Escolha a guia Items (Itens) para a tabela **Music**.
5. Na guia Items (Itens), veja a lista de itens gravados na tabela, classificados por **Artist** e **SongTitle**.

O primeiro item nas listas é aquele com **Artist****Acme Band** e **SongTitle****Happy Day**.



AWS CLI

O exemplo da AWS CLI a seguir lê um item da tabela `Music` usando `get-item`.

Note

O comportamento padrão do DynamoDB é o de leituras eventualmente consistentes. O parâmetro `consistent-read` é usado abaixo para demonstrar leituras fortemente consistentes.

```
aws dynamodb get-item --consistent-read \
--table-name Music \
--key '{ "Artist": { "S": "Acme Band"}, "SongTitle": { "S": "Happy Day"} }'
```

O uso de `get-item` retorna o seguinte resultado de exemplo.

```
{
  "Item": {
    "AlbumTitle": {
      "S": "Songs About Life"
    },
    "Awards": {
      "N": "10"
    },
    "SongTitle": {
      "S": "Happy Day"
    },
    "Artist": {
      "S": "Acme Band"
    }
  }
}
```

Para atualizar os dados na tabela, vá para [Etapa 4: Atualizar dados em uma tabela \(p. 71\)](#).

Etapa 4: Atualizar dados em uma tabela

Nesta etapa, você atualiza um item que criou em [Etapa 2: Gravar dados em uma tabela usando o console do ou a AWS CLI \(p. 68\)](#). É possível usar o console do DynamoDB ou o AWS CLI para atualizar o `AlbumTitle` de um item no `Music`, especificando `Artist`, `SongTitle`, e a atualização do `AlbumTitle`.

Para obter mais informações sobre operações de gravação, consulte [Gravação de um item \(p. 406\)](#).

AWS Management Console

Use o console do DynamoDB para atualizar os dados no `Music` Tabela do.

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Escolha a tabela Music (Música) na lista de tabelas.
4. Escolha a guia Items (Itens) para a tabela Music.
5. Escolha o item em que o valor Artist é Acme Band e o valor SongTitle é Happy Day.
6. Atualize o valor AlbumTitle (Nome do álbum) para **Updated Album Title** e escolha Save (Salvar).

A imagem a seguir mostra um item atualizado no console.

The screenshot shows the AWS Management Console interface for the 'Music' table. On the left, the table list shows 'Music' selected. On the right, the 'Items' tab is active under the 'Overview' section. A search bar at the top says 'Scan: [Table] Music: Artist, SongTitle'. Below it, a filter dropdown is set to 'Scan' and the search term is '[Table] Music: Artist, SongTitle'. A 'Start search' button is present. The main area displays a table with columns: Artist, SongTitle, AlbumTitle, and Awards. Two items are listed: 'Acme Band' and 'No One You Know'. A modal dialog titled 'Edit item' is open over the table. It contains a tree view with four items:

- AlbumTitle String : **Updated Album Title**
- Artist String : **Acme Band**
- Awards Number : **10**
- SongTitle String : **Happy Day**

AWS CLI

O exemplo da AWS CLI a seguir atualiza um item na tabela `Music` usando `update-item`.

```
aws dynamodb update-item \
--table-name Music \
--key '{ "Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"} }' \
--update-expression "SET AlbumTitle = :newval" \
--expression-attribute-values '{":newval":{"S":"Updated Album Title"}}' \
--return-values ALL_NEW
```

O uso de `update-item` retorna o seguinte resultado de exemplo.

```
{
```

```
"Attributes": {  
    "AlbumTitle": {  
        "S": "Updated Album Title"  
    },  
    "Awards": {  
        "N": "10"  
    },  
    "SongTitle": {  
        "S": "Happy Day"  
    },  
    "Artist": {  
        "S": "Acme Band"  
    }  
}
```

Para consultar os dados na tabela **Music**, vá para [Etapa 5: Consultar dados em uma tabela \(p. 73\)](#).

Etapa 5: Consultar dados em uma tabela

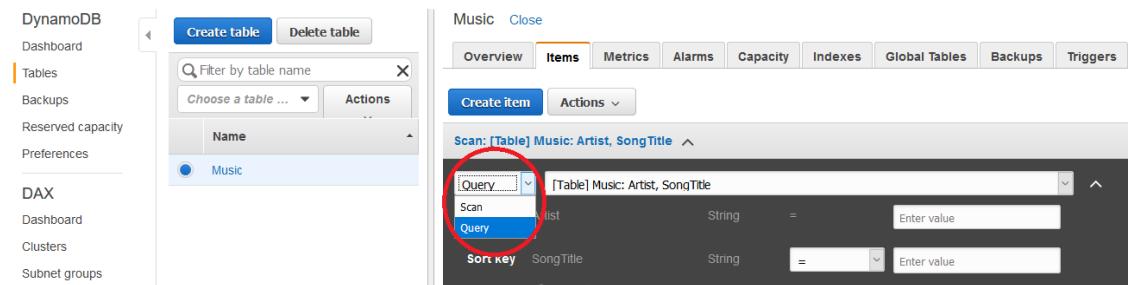
Nesta etapa, você consulta dados que gravou na tabela **Music** em [the section called “Etapa 2: Gravar dados” \(p. 68\)](#) especificando **Artist**.

Para obter mais informações sobre as operações de consulta, veja [Como trabalhar com consultas no DynamoDB \(p. 490\)](#).

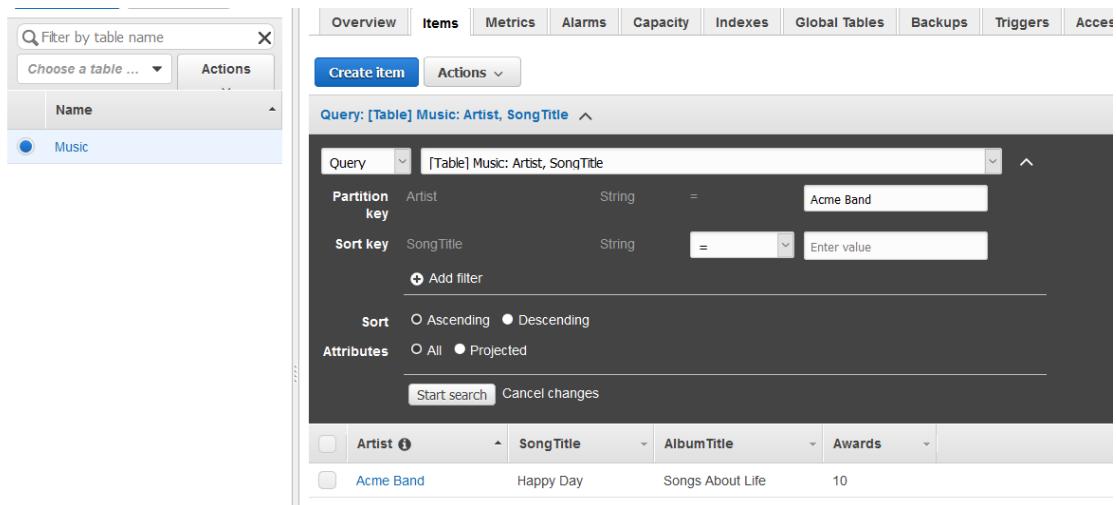
AWS Management Console

Siga estas etapas para usar o console do DynamoDB para consultar dados na **Music** Tabela do.

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Escolha a tabela **Music** (Música) na lista de tabelas.
4. Escolha a guia Items (Itens) para a tabela **Music**.
5. Na lista suspensa, escolha Query (Consultar).



6. Em Partition key (Chave de partição), insira **Acme Band** e escolha Start search (Iniciar pesquisa).



AWS CLI

O exemplo da AWS CLI a seguir consulta um item na tabela `Music` usando `query`.

```
aws dynamodb query \
--table-name Music \
--key-condition-expression "Artist = :name" \
--expression-attribute-values '{":name":{"S":"Acme Band"}}'
```

O uso de `query` retorna o seguinte resultado de exemplo.

```
{
    "Count": 1,
    "Items": [
        {
            "AlbumTitle": {
                "S": "Updated Album Title"
            },
            "Awards": {
                "N": "10"
            },
            "SongTitle": {
                "S": "Happy Day"
            },
            "Artist": {
                "S": "Acme Band"
            }
        }
    ],
    "ScannedCount": 1,
    "ConsumedCapacity": null
}
```

Para criar um índice secundário global para sua tabela, vá para [Etapa 6: Criar um índice secundário global \(p. 75\)](#).

Etapa 6: Criar um índice secundário global

Nesta etapa, você cria um índice secundário global para a tabela Music que você criou em [Etapa 1: Criar uma tabela \(p. 65\)](#).

Para obter mais informações sobre índices secundários globais , consulte [Como usar índices secundários globais no DynamoDB \(p. 562\)](#).

AWS Management Console

Para usar o console do Amazon DynamoDB para criar um índice secundário global para a tabela Music:

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Escolha a tabela Music (Música) na lista de tabelas.
4. Escolha a guia Indexes (Índices) para a tabela Music.
5. Escolha Create index (Criar índice).



6. Em Partition key (Chave de partição), insira **AlbumTitle** e escolha Create index (Criar índice).

Create index

Primary key* Partition key
 String

Add sort key

Index name*

Projected attributes

Read capacity units Write capacity units

Estimated cost \$3.39 / month ([Capacity calculator](#))

Approximate creation time is 5 minutes. Additional write capacity may decrease creation time. A notification will be sent to the SNS topic dynamodb once the index creation is complete. Basic Alarms with 80% upper threshold using SNS topic 'dynamodb' will be automatically created. Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced configuration for alarms can be done in the alarms tab.

[Cancel](#) [Create index](#)

AWS CLI

O exemplo da AWS CLI a seguir cria um índice secundário global `AlbumTitle-index` para a tabela `Music` usando `update-table`.

```
aws dynamodb update-table \
    --table-name Music \
    --attribute-definitions AttributeName=AlbumTitle,AttributeType=S \
    --global-secondary-index-updates \
        "[{\\"Create\\":{\\\"IndexName\\\": \\"AlbumTitle-index\\\", \\\"KeySchema\\\": [{\\\"AttributeName\\\": \\"AlbumTitle\\\", \\\"KeyType\\\": \\\"HASH\\\"}], \
            \\\"ProvisionedThroughput\\\": {\\\"ReadCapacityUnits\\\": 10, \\\"WriteCapacityUnits\\\": 5 \
            },\\\"Projection\\\":{\\\"ProjectionType\\\": \\\"ALL\\\"}}}]"
```

O uso de `update-table` retorna o seguinte resultado de exemplo.

```
{
    "TableDescription": {
        "TableArn": "arn:aws:dynamodb:us-west-2:522194210714:table/Music",
        "AttributeDefinitions": [
            {
                "AttributeName": "AlbumTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            }
        ],
        "IndexDefinitions": [
            {
                "IndexName": "AlbumTitle-index",
                "KeySchema": [
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 5
                }
            }
        ],
        "TableStatus": "ACTIVE",
        "CreationDateTime": "2012-08-10T14:00:00.000Z"
    }
}
```

```
        },
        {
            "AttributeName": "SongTitle",
            "AttributeType": "S"
        }
    ],
    "GlobalSecondaryIndexes": [
        {
            "IndexSizeBytes": 0,
            "IndexName": "AlbumTitle-index",
            "Projection": {
                "ProjectionType": "ALL"
            },
            "ProvisionedThroughput": {
                "NumberOfDecreasesToday": 0,
                "WriteCapacityUnits": 5,
                "ReadCapacityUnits": 10
            },
            "IndexStatus": "CREATING",
            "Backfilling": false,
            "KeySchema": [
                {
                    "KeyType": "HASH",
                    "AttributeName": "AlbumTitle"
                }
            ],
            "IndexArn": "arn:aws:dynamodb:us-west-2:522194210714:table/Music/index/AlbumTitle-index",
            "ItemCount": 0
        }
    ],
    "ProvisionedThroughput": {
        "NumberOfDecreasesToday": 0,
        "WriteCapacityUnits": 5,
        "ReadCapacityUnits": 10
    },
    "TableSizeBytes": 0,
    "TableName": "Music",
    "TableStatus": "UPDATING",
    "TableId": "d04c7240-0e46-435d-b231-d54091fe1017",
    "KeySchema": [
        {
            "KeyType": "HASH",
            "AttributeName": "Artist"
        },
        {
            "KeyType": "RANGE",
            "AttributeName": "SongTitle"
        }
    ],
    "ItemCount": 0,
    "CreationDateTime": 1558028402.69
}
}
```

Observe que o valor do campo `IndexStatus` está definido como `CREATING`.

Para verificar se o DynamoDB concluiu a criação do `AlbumTitle-index`índice secundário global, use o `describe-table`Comando da.

```
aws dynamodb describe-table --table-name Music | grep IndexStatus
```

Esse comando retorna o seguinte resultado. O índice estará pronto para ser utilizado quando o valor retornado do campo `IndexStatus` for `ACTIVE`.

```
"IndexStatus": "ACTIVE",
```

Depois, você poderá consultar o índice secundário global. Para obter mais detalhes, consulte [Etapa 7: Consultar o índice secundário global \(p. 78\)](#).

Etapa 7: Consultar o índice secundário global

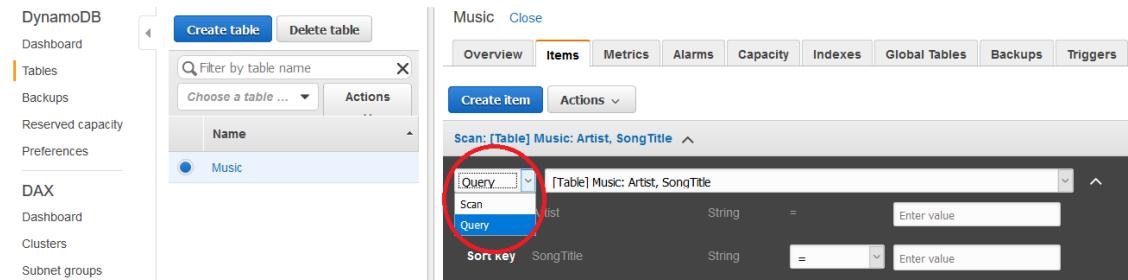
Nesta etapa, você consulta o índice secundário global no Music usando o console do Amazon DynamoDB ou o AWS CLI.

Para obter mais informações sobre índices secundários globais , consulte [Como usar índices secundários globais no DynamoDB \(p. 562\)](#).

AWS Management Console

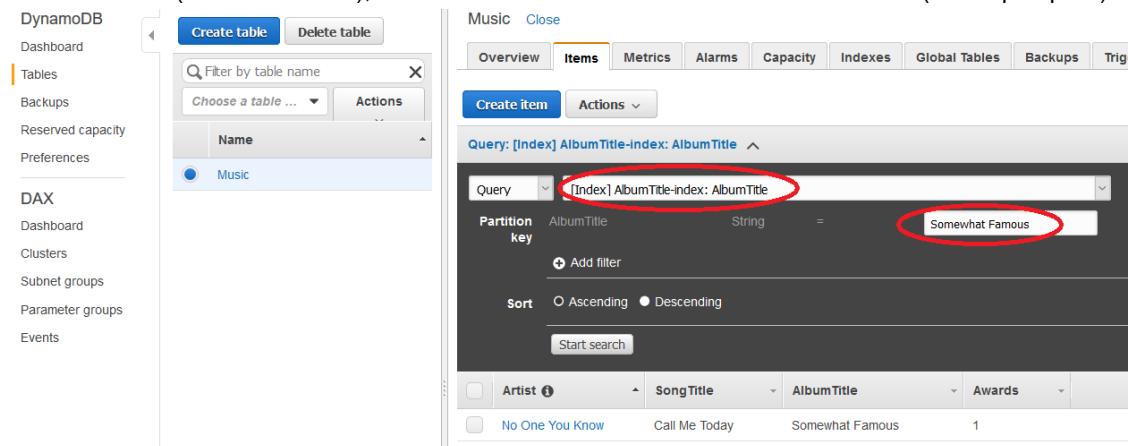
Siga estas etapas para usar o console do DynamoDB para consultar dados por meio do `AlbumTitle-index` índice secundário global.

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Escolha a tabela Music (Música) na lista de tabelas.
4. Escolha a guia Items (Itens) para a tabela Music.
5. Na lista suspensa, escolha Query (Consultar).



6. Na lista suspensa ao lado de `Consulta`, escolha `[Índice]` Índice de título de álbum: `AlbumTitle`.

Em `AlbumTitle` (Nome do álbum), insira `Somewhat Famous` e escolha Start search (Iniciar pesquisa).



AWS CLI

O exemplo da AWS CLI a seguir consulta um índice secundário global `AlbumTitle-index` na tabela `Music`.

```
aws dynamodb query \
--table-name Music \
--index-name AlbumTitle-index \
--key-condition-expression "AlbumTitle = :name" \
--expression-attribute-values '{":name":{"S":"Somewhat Famous"}}'
```

O uso de `query` retorna o seguinte resultado de exemplo.

```
{
    "Count": 1,
    "Items": [
        {
            "AlbumTitle": {
                "S": "Somewhat Famous"
            },
            "Awards": {
                "N": "1"
            },
            "SongTitle": {
                "S": "Call Me Today"
            },
            "Artist": {
                "S": "No One You Know"
            }
        }
    ],
    "ScannedCount": 1,
    "ConsumedCapacity": null
}
```

Etapa 8 Limpar recursos

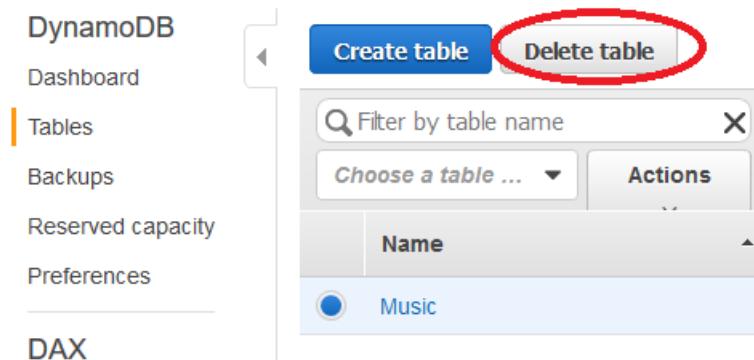
Se você não precisa mais da tabela do Amazon DynamoDB que criou no tutorial, você pode excluí-la. Esta etapa ajuda a garantir que você não será cobrado pelos recursos que não está utilizando. É possível usar o console do DynamoDB ou o AWS CLI para excluir a tabela criada na [Etapa 1: Criar uma tabela \(p. 65\)](#).

Para obter mais informações sobre as operações de tabelas no DynamoDB, consulte [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#).

AWS Management Console

Para excluir uma tabela `Music` usando o console

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Escolha a tabela `Music` (Música) na lista de tabelas.
4. Escolha a guia Indexes (Índices) da tabela `Music`.
5. Selecione Delete table (Excluir tabela).



AWS CLI

O exemplo da AWS CLI a seguir exclui a tabela Music usando delete-table.

```
aws dynamodb delete-table --table-name Music
```

Conceitos básicos do DynamoDB: Próximas etapas

Para obter mais informações sobre como usar o Amazon DynamoDB, consulte os seguintes tópicos:

- Como trabalhar com tabelas e dados no DynamoDB (p. 339)
- Trabalho com itens e atributos (p. 404)
- Como trabalhar com consultas no DynamoDB (p. 490)
- Como usar índices secundários globais no DynamoDB (p. 562)
- Trabalhar com Transações (p. 680)
- Aceleração na memória com o DynamoDB Accelerator (DAX) (p. 713)
- Conceitos básicos do DynamoDB e do AWSSDKs da (p. 81)
- Programação com o DynamoDB e o AWSSDKs da (p. 211)

Conceitos básicos do DynamoDB e do AWSSDKs da

Esta seção contém tutoriais úteis para ajudar você a saber mais sobre o Amazon DynamoDB. Encorajamos você a trabalhar em um dos tutoriais de linguagem específica. Os exemplos de código desses tutoriais podem ser executados na versão para download do DynamoDB ou no serviço web do DynamoDB.

Note

Os SDKs da AWS estão disponíveis para uma ampla variedade de linguagens. Para obter uma lista completa, consulte [Ferramentas para a Amazon Web Services](#).

Tópicos

- [Conceitos básicos de Java e do DynamoDB \(p. 81\)](#)
- [Conceitos básicos do JavaScript e do DynamoDB \(p. 101\)](#)
- [Conceitos básicos sobre Node.js e o DynamoDB \(p. 123\)](#)
- [Conceitos básicos do .NET e do DynamoDB \(p. 140\)](#)
- [PHP e DynamoDB \(p. 157\)](#)
- [Conceitos básicos do desenvolvimento com Python e DynamoDB \(p. 178\)](#)
- [Ruby e DynamoDB \(p. 192\)](#)

Conceitos básicos de Java e do DynamoDB

Neste tutorial, você usa o AWS SDK for JavaPara escrever programas simples a fim de realizar as seguintes operações do Amazon DynamoDB:

- Criar uma tabela chamada `Movies` e carregar dados de amostra no formato JSON.
- Realizar operações `create`, `read`, `update` e `delete` na tabela.
- Executar consultas simples.

O SDK for Java oferece vários modelos de programação para diferentes casos de uso. Neste exercício, o código Java usa o modelo de documento, que fornece um nível de abstração capaz de facilitar o trabalho com documentos JSON.

À medida que você trabalha neste tutorial, consulte o [AWS SDK for Java Documentação do](#).

Pré-requisitos do tutorial

- Faça download e execute o DynamoDB no computador. Para mais informações, consulte [Configuração do DynamoDB Local \(versão disponível para download\) \(p. 50\)](#).

O DynamoDB (versão para download) também está disponível como parte do AWS Toolkit for Eclipse. Para mais informações, consulte [AWS Toolkit for Eclipse](#).

Note

Você pode usar as versões disponíveis para download do DynamoDB neste tutorial. Para obter informações sobre como executar o mesmo código para o web service do DynamoDB, consulte [oSummary \(p. 100\)](#).

- Configure umAWSA chave de acesso da para usar oAWSSDKs. Para mais informações, consulte [Configuração do DynamoDB \(serviço Web\) \(p. 57\)](#).
- Configurar o AWS SDK for Java:
 - Instale um ambiente de desenvolvimento Java. Se estiver usando o IDE Eclipse, instale o AWS Toolkit for Eclipse.
 - Instale o AWS SDK for Java.
 - Configure seuAWSCredenciais de segurança para uso com o SDK for Java.

Para obter instruções, consulte[Conceitos básicos](#)noAWS SDK for JavaGuia do desenvolvedor.

Etapa 1: Criar uma tabela usando o Java e o DynamoDB

Nesta etapa, você cria uma tabela chamada `Movies`. A chave primária da tabela é composta dos seguintes atributos:

- `year`– A chave de partição. O `ScalarAttributeType` é `N` para número.
- `title`– A chave de classificação. O `ScalarAttributeType` é `S` para string.

1. Copie o seguinte programa e cole-o no seu ambiente de desenvolvimento Java.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
package com.amazonaws.codesamples.gsg;  
  
import java.util.Arrays;  
  
import com.amazonaws.client.builder.AwsClientBuilder;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Table;  
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;  
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;  
import com.amazonaws.services.dynamodbv2.model.KeyType;  
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;  
import com.amazonaws.services.dynamodbv2.model.ScalarAttributeType;
```

```
public class MoviesCreateTable {  
  
    public static void main(String[] args) throws Exception {  
  
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()  
            .withEndpointConfiguration(new  
                AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))  
            .build();  
  
        DynamoDB dynamoDB = new DynamoDB(client);  
  
        String tableName = "Movies";  
  
        try {  
            System.out.println("Attempting to create table; please wait...");  
            Table table = dynamoDB.createTable(tableName,  
                Arrays.asList(new KeySchemaElement("year", KeyType.HASH), // Partition  
                    // key  
                    new KeySchemaElement("title", KeyType.RANGE)), // Sort key  
                Arrays.asList(new AttributeDefinition("year", ScalarAttributeType.N),  
                    new AttributeDefinition("title", ScalarAttributeType.S)),  
                new ProvisionedThroughput(10L, 10L));  
            table.waitForActive();  
            System.out.println("Success. Table status: " +  
                table.getDescription().getTableStatus());  
  
        }  
        catch (Exception e) {  
            System.err.println("Unable to create table: ");  
            System.err.println(e.getMessage());  
        }  
  
    }  
}
```

Note

- Você define o endpoint para indicar que está criando a tabela no DynamoDB no computador.
- Na chamada `createTable`, você especifica o nome da tabela, os atributos da chave primária e seus tipos de dados.
- O `ProvisionedThroughput` parâmetro é necessário, mas a versão para download do DynamoDB o ignora. (O throughput provisionado está além do escopo deste exercício.)

2. Compile e execute o programa.

Para saber mais sobre como gerenciar tabelas, consulte [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#).

Etapa 2: Carregar dados de exemplo com o Java e o DynamoDB

Nesta etapa, você preenche a tabela `Movies` com dados de exemplo.

Tópicos

- [Etapa 2.1: Faça download do arquivo de dados de exemplo \(p. 84\)](#)
- [Etapa 2.2: Carregar os dados de exemplo na tabela de filmes \(p. 85\)](#)

Esse cenário usa um arquivo de dados de exemplo que contém informações sobre milhares de filmes do Internet Movie Database (IMDb). Os dados de filme estão no formato JSON, conforme mostrado no exemplo a seguir. Para cada filme, existe um `year`, um `title` e um mapa JSON chamado `info`.

```
[  
  {  
    "year" : ... ,  
    "title" : ... ,  
    "info" : { ... }  
  },  
  {  
    "year" : ... ,  
    "title" : ... ,  
    "info" : { ... }  
  },  
  ...  
]
```

Nos dados JSON, observe o seguinte:

- Você deve usar `year` e `title` como os valores de atributo de chave primária da tabela `Movies`.
- Você deve armazenar o resto dos valores `info` em um único atributo chamado `info`. Este programa ilustra como você pode armazenar o JSON em um atributo do Amazon DynamoDB.

Veja a seguir um exemplo de dados de filme.

```
{  
  "year" : 2013,  
  "title" : "Turn It Down, Or Else!",  
  "info" : {  
    "directors" : [  
      "Alice Smith",  
      "Bob Jones"  
    ],  
    "release_date" : "2013-01-18T00:00:00Z",  
    "rating" : 6.2,  
    "genres" : [  
      "Comedy",  
      "Drama"  
    ],  
    "image_url" : "http://ia.media-imdb.com/images/N/  
09ERWAU7FS797AJ7LU8HN09AMUP908RLlo5JF90EWR7LJKQ7@._V1_SX400_.jpg",  
    "plot" : "A rock band plays their music at high volumes, annoying the neighbors.",  
    "rank" : 11,  
    "running_time_secs" : 5215,  
    "actors" : [  
      "David Matthewman",  
      "Ann Thomas",  
      "Jonathan G. Neff"  
    ]  
  }  
}
```

Etapa 2.1: Faça download do arquivo de dados de exemplo

1. Faça download do arquivo de dados de exemplo: [moviedata.zip](#)
2. Extraia o arquivo de dados (`moviedata.json`) do arquivo.

3. Copie e cole o arquivo `moviedata.json` para o diretório atual.

Etapa 2.2: Carregar os dados de exemplo na tabela de filmes

Depois de fazer download dos dados de exemplo, é possível executar o programa a seguir para preencher a tabela `Movies`.

1. Copie o seguinte programa e cole-o no seu ambiente de desenvolvimento Java.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
package com.amazonaws.codesamples.gsg;  
  
import java.io.File;  
import java.util.Iterator;  
  
import com.amazonaws.client.builder.AwsClientBuilder;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.Table;  
import com.fasterxml.jackson.core.JsonFactory;  
import com.fasterxml.jackson.core.JsonParser;  
import com.fasterxml.jackson.databind.JsonNode;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import com.fasterxml.jackson.databind.node.ObjectNode;  
  
public class MoviesLoadData {  
  
    public static void main(String[] args) throws Exception {  
  
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()  
            .withEndpointConfiguration(new  
AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))  
            .build();  
  
        DynamoDB dynamoDB = new DynamoDB(client);  
  
        Table table = dynamoDB.getTable("Movies");  
  
        JsonParser parser = new JsonFactory().createParser(new File("moviedata.json"));  
  
        JsonNode rootNode = new ObjectMapper().readTree(parser);  
        Iterator<JsonNode> iter = rootNode.iterator();  
  
        ObjectNode currentNode;  
  
        while (iter.hasNext()) {  
            currentNode = (ObjectNode) iter.next();  
        }  
    }  
}
```

```
int year = currentNode.path("year").asInt();
String title = currentNode.path("title").asText();

try {
    table.putItem(new Item().withPrimaryKey("year", year, "title",
title).withJSON("info",
        currentNode.path("info").toString()));
    System.out.println("PutItem succeeded: " + year + " " + title);

}
catch (Exception e) {
    System.err.println("Unable to add movie: " + year + " " + title);
    System.err.println(e.getMessage());
    break;
}
parser.close();
}

}
```

Esse programa usa a biblioteca Jackson de código-fonte aberto para processar o JSON. O Jackson está incluído no AWS SDK for Java. Você não precisa instalá-lo separadamente.

2. Compile e execute o programa.

Etapa 3: Criar, ler, atualizar e excluir um item

Nesta etapa, você realiza operações de leitura e gravação em um item na tabela Movies.

Para saber mais sobre leitura e gravação de dados, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Tópicos

- [Etapa 3.1: Criar um novo item \(p. 86\)](#)
- [Etapa 3.2: Ler um item \(p. 88\)](#)
- [Etapa 3.3: Atualização de um item \(p. 89\)](#)
- [Etapa 3.4: Incrementar um Contador atômico \(p. 91\)](#)
- [Etapa 3.5: Atualização de um item \(condicionalmente\) \(p. 92\)](#)
- [Etapa 3.6: Exclusão de um item \(p. 93\)](#)

Etapa 3.1: Criar um novo item

Nesta etapa, você adiciona um novo item à tabela Movies.

1. Copie o seguinte programa e cole-o no seu ambiente de desenvolvimento Java.

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
```

```
* specific language governing permissions and limitations under the License.  
*/  
  
package com.amazonaws.codesamples.gsg;  
  
import java.util.HashMap;  
import java.util.Map;  
  
import com.amazonaws.client.builder.AwsClientBuilder;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.PutItemOutcome;  
import com.amazonaws.services.dynamodbv2.document.Table;  
  
public class MoviesItemOps01 {  
  
    public static void main(String[] args) throws Exception {  
  
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()  
            .withEndpointConfiguration(new  
                AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))  
            .build();  
  
        DynamoDB dynamoDB = new DynamoDB(client);  
  
        Table table = dynamoDB.getTable("Movies");  
  
        int year = 2015;  
        String title = "The Big New Movie";  
  
        final Map<String, Object> infoMap = new HashMap<String, Object>();  
        infoMap.put("plot", "Nothing happens at all.");  
        infoMap.put("rating", 0);  
  
        try {  
            System.out.println("Adding a new item...");  
            PutItemOutcome outcome = table  
                .putItem(new Item().withPrimaryKey("year", year, "title",  
                    title).withMap("info", infoMap));  
  
            System.out.println("PutItem succeeded:\n" + outcome.getPutItemResult());  
        }  
        catch (Exception e) {  
            System.err.println("Unable to add item: " + year + " " + title);  
            System.err.println(e.getMessage());  
        }  
    }  
}
```

Note

A chave primária é obrigatória. Este código adiciona um item que tem chave primária (`year`, `title`) e atributos `info`. O atributo `info` armazena o código de exemplo JSON que fornece mais informações sobre o filme.

2. Compile e execute o programa.

Etapa 3.2: Ler um item

No programa anterior, você adicionou os seguintes itens à tabela.

```
{  
    year: 2015,  
    title: "The Big New Movie",  
    info: {  
        plot: "Nothing happens at all.",  
        rating: 0  
    }  
}
```

Você pode usar o método `getItem` para ler o item da tabela `Movies`. Você deve especificar os valores de chave primária para que possa ler qualquer item de `Movies`, caso saiba seu `year` e `title`.

1. Copie o seguinte programa e cole-o no seu ambiente de desenvolvimento Java.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
package com.amazonaws.codesamples.gsg;  
  
import com.amazonaws.client.builder.AwsClientBuilder;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.Table;  
import com.amazonaws.services.dynamodbv2.document.spec.GetItemSpec;  
  
public class MoviesItemOps02 {  
  
    public static void main(String[] args) throws Exception {  
  
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()  
            .withEndpointConfiguration(new  
        AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))  
            .build();  
  
        DynamoDB dynamoDB = new DynamoDB(client);  
  
        Table table = dynamoDB.getTable("Movies");  
  
        int year = 2015;  
        String title = "The Big New Movie";  
  
        GetItemSpec spec = new GetItemSpec().withPrimaryKey("year", year, "title",  
            title);
```

```
try {
    System.out.println("Attempting to read the item...");
    Item outcome = table.getItem(spec);
    System.out.println("GetItem succeeded: " + outcome);

}
catch (Exception e) {
    System.err.println("Unable to read item: " + year + " " + title);
    System.err.println(e.getMessage());
}

}
```

2. Compile e execute o programa.

Etapa 3.3: Atualização de um item

Você pode usar o método `updateItem` para modificar um item existente. Você pode atualizar valores de atributos existentes, adicionar novos atributos ou remover atributos.

Neste exemplo, você executa as seguintes atualizações:

- Altere o valor dos atributos existentes (`rating`, `plot`).
- Adicione um novo atributo de lista (`actors`) ao mapa `info` existente.

O item é alterado do seguinte:

```
{
    year: 2015,
    title: "The Big New Movie",
    info: {
        plot: "Nothing happens at all.",
        rating: 0
    }
}
```

Para isto:

```
{
    year: 2015,
    title: "The Big New Movie",
    info: {
        plot: "Everything happens all at once.",
        rating: 5.5,
        actors: ["Larry", "Moe", "Curly"]
    }
}
```

1. Copie o seguinte programa e cole-o no seu ambiente de desenvolvimento Java.

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 */
```

```
/*
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples.gsg;

import java.util.Arrays;

import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.document.UpdateItemOutcome;
import com.amazonaws.services.dynamodbv2.document.spec.UpdateItemSpec;
import com.amazonaws.services.dynamodbv2.document.utils.ValueMap;
import com.amazonaws.services.dynamodbv2.model.ReturnValue;

public class MoviesItemOps03 {

    public static void main(String[] args) throws Exception {

        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
            .withEndpointConfiguration(new
        AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))
            .build();

        DynamoDB dynamoDB = new DynamoDB(client);

        Table table = dynamoDB.getTable("Movies");

        int year = 2015;
        String title = "The Big New Movie";

        UpdateItemSpec updateItemSpec = new UpdateItemSpec().withPrimaryKey("year",
year, "title", title)
            .withUpdateExpression("set info.rating = :r, info.plot=:p, info.actors=:a")
            .WithValueMap(new ValueMap().withNumber(":r", 5.5).withString(":p",
"Everything happens all at once.")
            .withList(":a", Arrays.asList("Larry", "Moe", "Curly")))
            .withReturnValues(ReturnValue.UPDATED_NEW);

        try {
            System.out.println("Updating the item...");
            UpdateItemOutcome outcome = table.updateItem(updateItemSpec);
            System.out.println("UpdateItem succeeded:\n" +
outcome.getItem().toJSONPretty());
        }
        catch (Exception e) {
            System.err.println("Unable to update item: " + year + " " + title);
            System.err.println(e.getMessage());
        }
    }
}
```

Note

Esse programa usa `UpdateExpression` para descrever todas as atualizações que você deseja realizar no item especificado.

O `OReturnValues` instrui o Amazon DynamoDB a retornar somente os atributos atualizados (`UPDATED_NEW`).

2. Compile e execute o programa.

Etapa 3.4: Incrementar um Contador atômico

DynamoDB oferece suporte a contadores atômicos. Use o método `updateItem` para aumentar ou reduzir o valor de um atributo existente sem interferir em outras solicitações de gravação. (Todas as solicitações de gravação são aplicadas na ordem em que são recebidas.)

O programa a seguir mostra como aumentar a `rating` de um filme. Toda vez que você o executa, o programa aumenta esse atributo uma vez.

1. Copie o seguinte programa e cole-o no seu ambiente de desenvolvimento Java.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
package com.amazonaws.codesamples.gsg;  
  
import com.amazonaws.client.builder.AwsClientBuilder;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Table;  
import com.amazonaws.services.dynamodbv2.document.UpdateItemOutcome;  
import com.amazonaws.services.dynamodbv2.document.spec.UpdateItemSpec;  
import com.amazonaws.services.dynamodbv2.document.utils.ValueMap;  
import com.amazonaws.services.dynamodbv2.model.ReturnValue;  
  
public class MoviesItemOps04 {  
  
    public static void main(String[] args) throws Exception {  
  
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()  
            .withEndpointConfiguration(new  
                AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))  
            .build();  
  
        DynamoDB dynamoDB = new DynamoDB(client);  
  
        Table table = dynamoDB.getTable("Movies");  
  
        int year = 2015;  
        String title = "The Big New Movie";  
  
        UpdateItemSpec updateItemSpec = new UpdateItemSpec().withPrimaryKey("year",  
            year, "title", title)  
            .withUpdateExpression("set info.rating = info.rating + :val")
```

```
        .withValueMap(new ValueMap().withNumber(":val",
1)).withReturnValues(ReturnValue.UPDATED_NEW);

    try {
        System.out.println("Incrementing an atomic counter...");
        UpdateItemOutcome outcome = table.updateItem(updateItemSpec);
        System.out.println("UpdateItem succeeded:\n" +
outcome.getItem().toJSONPretty());
    }
}
}
```

2. Compile e execute o programa.

Etapa 3.5: Atualização de um item (condicionalmente)

O programa a seguir mostra como usar `UpdateItem` com uma condição. Se a condição for verdadeira, a atualização será bem-sucedida; caso contrário, a atualização não será realizada.

Neste caso, o item de filme será atualizado somente se houver mais de três atores.

1. Copie o seguinte programa e cole-o no seu ambiente de desenvolvimento Java.

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples.gsg;

import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.PrimaryKey;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.document.UpdateItemOutcome;
import com.amazonaws.services.dynamodbv2.document.spec.UpdateItemSpec;
import com.amazonaws.services.dynamodbv2.document.utils.ValueMap;
import com.amazonaws.services.dynamodbv2.model.ReturnValue;

public class MoviesItemOps05 {

    public static void main(String[] args) throws Exception {
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
```

```
.withEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))
.build();

DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("Movies");

int year = 2015;
String title = "The Big New Movie";

UpdateItemSpec updateItemSpec = new UpdateItemSpec()
    .withPrimaryKey(new PrimaryKey("year", year, "title",
title)).withUpdateExpression("remove info.actors[0]")
    .withConditionExpression("size(info.actors) > :num").withValueMap(new
ValueMap().withNumber(":num", 3))
    .withReturnValues(ReturnValue.UPDATED_NEW);

// Conditional update (we expect this to fail)
try {
    System.out.println("Attempting a conditional update...");
    UpdateItemOutcome outcome = table.updateItem(updateItemSpec);
    System.out.println("UpdateItem succeeded:\n" +
outcome.getItem().toJSONPretty());
}

catch (Exception e) {
    System.err.println("Unable to update item: " + year + " " + title);
    System.err.println(e.getMessage());
}
}
}
```

2. Compile e execute o programa.

O programa deve falhar com a mensagem a seguir:

A solicitação condicional falhou

Isto acontece porque o filme tem três atores, mas a condição está em busca de mais de três atores.

3. Modifique o programa para que ConditionExpression fique semelhante ao seguinte.

```
.withConditionExpression("size(info.actors) >= :num")
```

A condição agora é maior ou igual a 3 em vez de maior que 3.

4. Compile e execute o programa. A operação UpdateItem agora deve ser bem-sucedida.

Etapa 3.6: Exclusão de um item

Você pode usar o método `deleteItem` para excluir um item, especificando sua chave primária. Se desejar, você pode fornecer uma `ConditionExpression` para evitar a exclusão do item, se a condição não for atendida.

No exemplo a seguir, você tenta excluir um item de filme específico se a sua classificação for 5 ou menos.

1. Copie o seguinte programa e cole-o no seu ambiente de desenvolvimento Java.

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
```

```
* This file is licensed under the Apache License, Version 2.0 (the "License").  
* You may not use this file except in compliance with the License. A copy of  
* the License is located at  
*  
* http://aws.amazon.com/apache2.0/  
*  
* This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
* CONDITIONS OF ANY KIND, either express or implied. See the License for the  
* specific language governing permissions and limitations under the License.  
*/  
  
package com.amazonaws.codesamples.gsg;  
  
import com.amazonaws.client.builder.AwsClientBuilder;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.PrimaryKey;  
import com.amazonaws.services.dynamodbv2.document.Table;  
import com.amazonaws.services.dynamodbv2.document.spec.DeleteItemSpec;  
import com.amazonaws.services.dynamodbv2.document.utils.ValueMap;  
  
public class MoviesItemOps06 {  
  
    public static void main(String[] args) throws Exception {  
  
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()  
            .withEndpointConfiguration(new  
                AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))  
            .build();  
  
        DynamoDB dynamoDB = new DynamoDB(client);  
  
        Table table = dynamoDB.getTable("Movies");  
  
        int year = 2015;  
        String title = "The Big New Movie";  
  
        DeleteItemSpec deleteItemSpec = new DeleteItemSpec()  
            .withPrimaryKey(new PrimaryKey("year", year, "title",  
                title)).withConditionExpression("info.rating <= :val")  
            .WithValueMap(new ValueMap().withNumber(":val", 5.0));  
  
        // Conditional delete (we expect this to fail)  
  
        try {  
            System.out.println("Attempting a conditional delete...");  
            table.deleteItem(deleteItemSpec);  
            System.out.println("DeleteItem succeeded");  
        }  
        catch (Exception e) {  
            System.err.println("Unable to delete item: " + year + " " + title);  
            System.err.println(e.getMessage());  
        }  
    }  
}
```

2. Compile e execute o programa.

O programa deve falhar com a mensagem a seguir:

A solicitação condicional falhou

Isso ocorre porque a classificação desse filme em especial é maior que 5.

3. Modifique o programa para remover a condição em `DeleteItemSpec`.

```
DeleteItemSpec deleteItemSpec = new DeleteItemSpec()  
.withPrimaryKey(new PrimaryKey("year", 2015, "title", "The Big New Movie"));
```

4. Compile e execute o programa. Agora, a exclusão é bem-sucedida, pois você removeu a condição.

Etapa 4: Consultar e verificar os dados

Você pode usar o método `query` para recuperar os dados de uma tabela. Você deve especificar o valor de uma chave de partição. A chave de classificação é opcional.

A chave primária da tabela `Movies` é composta pelo seguinte:

- `year`– A chave de partição. O tipo de atributo é número.
- `title`– A chave de classificação. O tipo de atributo é string.

Para encontrar todos os filmes lançados durante um ano, você precisa especificar somente o `year`. Você também pode fornecer o `title` para recuperar um subconjunto de filmes baseados na mesma condição (na chave de classificação). Por exemplo, é possível encontrar filmes lançados em 2014 que têm um título começando com a letra "A".

Além do método `query`, também há um método `scan` que pode recuperar todos os dados da tabela.

Para saber mais sobre como consultar e verificar dados, consulte [Como trabalhar com consultas no DynamoDB \(p. 490\)](#) e [Como trabalhar com verificações no DynamoDB \(p. 508\)](#), respectivamente.

Tópicos

- [Etapa 4.1: Consulta \(p. 95\)](#)
- [Etapa 4.2: Scan \(p. 97\)](#)

Etapa 4.1: Consulta

O código incluído nesta etapa realiza as seguintes consultas:

- Recupere todos os filmes lançados no `year` de 1985.
- Recupere todos os filmes lançados no `year` de 1992, com um `title` que comece com a letra "A" até a letra "L".

1. Copie o seguinte programa e cole-o no seu ambiente de desenvolvimento Java.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */
```

```
*/  
  
package com.amazonaws.codesamples.gsg;  
  
import java.util.HashMap;  
import java.util.Iterator;  
  
import com.amazonaws.client.builder.AwsClientBuilder;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.ItemCollection;  
import com.amazonaws.services.dynamodbv2.document.QueryOutcome;  
import com.amazonaws.services.dynamodbv2.document.Table;  
import com.amazonaws.services.dynamodbv2.document.spec.QuerySpec;  
  
public class MoviesQuery {  
  
    public static void main(String[] args) throws Exception {  
  
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()  
            .withEndpointConfiguration(new  
        AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))  
            .build();  
  
        DynamoDB dynamoDB = new DynamoDB(client);  
  
        Table table = dynamoDB.getTable("Movies");  
  
        HashMap<String, String> nameMap = new HashMap<String, String>();  
        nameMap.put("#yr", "year");  
  
        HashMap<String, Object> valueMap = new HashMap<String, Object>();  
        valueMap.put(":yyyy", 1985);  
  
        QuerySpec querySpec = new QuerySpec().withKeyConditionExpression("#yr  
= :yyyy").withNameMap(nameMap)  
            .withValueMap(valueMap);  
  
        ItemCollection<QueryOutcome> items = null;  
        Iterator<Item> iterator = null;  
        Item item = null;  
  
        try {  
            System.out.println("Movies from 1985");  
            items = table.query(querySpec);  
  
            iterator = items.iterator();  
            while (iterator.hasNext()) {  
                item = iterator.next();  
                System.out.println(item.getNumber("year") + ": " +  
item.getString("title"));  
            }  
        }  
        catch (Exception e) {  
            System.err.println("Unable to query movies from 1985");  
            System.err.println(e.getMessage());  
        }  
  
        valueMap.put(":yyyy", 1992);  
        valueMap.put(":letter1", "A");  
        valueMap.put(":letter2", "L");  
    }  
}
```

```
querySpec.withProjectionExpression("#yr, title, info.genres, info.actors[0]")
    .withKeyConditionExpression("#yr = :yyyy and title between :letter1
and :letter2").withNameMap(nameMap)
    .WithValueMap(valueMap);

try {
    System.out.println("Movies from 1992 - titles A-L, with genres and lead
actor");
    items = table.query(querySpec);

    iterator = items.iterator();
    while (iterator.hasNext()) {
        item = iterator.next();
        System.out.println(item.getNumber("year") + ": " +
item.getString("title") + " " + item.getMap("info"));
    }
}
catch (Exception e) {
    System.err.println("Unable to query movies from 1992:");
    System.err.println(e.getMessage());
}
```

Note

- nameMap fornece substituição de nome. Isso é usado porque yearO é uma palavra reservada no Amazon DynamoDB. Não é possível usá-la diretamente em nenhuma expressão, incluindo KeyConditionExpression. Você deve usar o nome de atributo de expressão #yr para resolver essa questão.
- valueMap fornece substituição de valor. Ele é usado porque não é possível usar literais em nenhuma expressão, incluindo KeyConditionExpression. Você deve usar o valor de atributo de expressão :yyyy para resolver essa questão.

Primeiro, você cria o objeto querySpec, que descreve os parâmetros da consulta, e depois transmite esse objeto ao método query.

2. Compile e execute o programa.

Note

O programa anterior mostra como consultar uma tabela por seus atributos de chave primária. No DynamoDB, você pode opcionalmente criar um ou mais índices secundários em uma tabela e consultá-los da mesma forma que consulta uma tabela. Os índices secundários oferecem aos seus aplicativos mais flexibilidade ao permitir consultas nos atributos que não são chave. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 4.2: Scan

O método scan lê todos os itens da tabela inteira e retorna todos os dados nela contidos. Você pode fornecer uma filter_expression opcional, para que apenas os itens que correspondem aos seus critérios sejam retornados. No entanto, o filtro é aplicado somente depois que a tabela inteira foi verificada.

O programa a seguir verifica a tabela Movies inteira, que contém aproximadamente 5.000 itens. A verificação especifica o filtro opcional para recuperar somente os filmes da década de 1950 (aproximadamente 100 itens) e descartar todos os outros.

1. Copie o seguinte programa e cole-o no seu ambiente de desenvolvimento Java.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
package com.amazonaws.codesamples.gsg;  
  
import java.util.Iterator;  
  
import com.amazonaws.client.builder.AwsClientBuilder;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.ItemCollection;  
import com.amazonaws.services.dynamodbv2.document.ScanOutcome;  
import com.amazonaws.services.dynamodbv2.document.Table;  
import com.amazonaws.services.dynamodbv2.document.spec.ScanSpec;  
import com.amazonaws.services.dynamodbv2.document.utils.NameMap;  
import com.amazonaws.services.dynamodbv2.document.utils.ValueMap;  
  
public class MoviesScan {  
  
    public static void main(String[] args) throws Exception {  
  
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()  
            .withEndpointConfiguration(new  
                AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))  
            .build();  
  
        DynamoDB dynamoDB = new DynamoDB(client);  
  
        Table table = dynamoDB.getTable("Movies");  
  
        ScanSpec scanSpec = new ScanSpec().withProjectionExpression("#yr, title,  
info.rating")  
            .withFilterExpression("#yr between :start_yr and :end_yr").withNameMap(new  
NameMap().with("#yr", "year")  
            .WithValueMap(new ValueMap().withNumber(":start_yr",  
1950).withNumber(":end_yr", 1959));  
  
        try {  
            ItemCollection<ScanOutcome> items = table.scan(scanSpec);  
  
            Iterator<Item> iter = items.iterator();  
            while (iter.hasNext()) {  
                Item item = iter.next();  
                System.out.println(item.toString());  
            }  
        }  
        catch (Exception e) {
```

```
        System.err.println("Unable to scan the table:");
        System.err.println(e.getMessage());
    }
}
```

No código, observe o seguinte:

- `ProjectionExpression` especifica os atributos que você deseja no resultado da verificação.
 - `FilterExpression` especifica uma condição que retorna apenas os itens que satisfazem à condição. Todos os outros itens são descartados.
2. Compile e execute o programa.

Note

Também é possível usar a operação `Scan` com quaisquer índices secundários criados na tabela. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 5: (Opcional) Excluir a tabela

Para excluir a tabela `Movies`:

1. Copie o seguinte programa e cole-o no seu ambiente de desenvolvimento Java.

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples.gsg;

import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Table;

public class MoviesDeleteTable {

    public static void main(String[] args) throws Exception {

        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
            .withEndpointConfiguration(new
        AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))
            .build();

        DynamoDB dynamoDB = new DynamoDB(client);

        Table table = dynamoDB.getTable("Movies");
    }
}
```

```
try {
    System.out.println("Attempting to delete table; please wait...");
    table.delete();
    table.waitForDelete();
    System.out.print("Success.");
}

}
catch (Exception e) {
    System.err.println("Unable to delete table: ");
    System.err.println(e.getMessage());
}
}
```

2. Compile e execute o programa.

Summary

Neste tutorial, você criou oMoviesNo Amazon DynamoDB em seu computador e realizou operações básicas. A versão para download do DynamoDB é útil durante o desenvolvimento e o teste de aplicativos. No entanto, quando você estiver pronto para executar o aplicativo em um ambiente de produção, precisará modificar o código para que ele use o serviço web do DynamoDB.

Modificação do código para usar o DynamoDB Service

Para usar o serviço do DynamoDB, altere o endpoint em seu aplicativo.

1. Remova a seguinte importação.

```
import com.amazonaws.client.builder.AwsClientBuilder;
```

2. Depois disso, acesse AmazonDynamoDB no código.

```
AmazonDynamoDB client =
    AmazonDynamoDBClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))
    .build();
```

3. Agora, modifique o cliente para que ele tenha acesso a umaAWSRegião em vez de um ponto final específico.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
    .withRegion(Regions.REGION)
    .build();
```

Por exemplo, se quiser acessar a região us-west-2 region, você faria o seguinte.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();
```

Em vez de usar o DynamoDB no computador, o programa agora usa o endpoint de serviço web do DynamoDB na região do Oeste dos EUA (Oregon).

O DynamoDB está disponível emAWSRegiões do mundo todo. Para obter a lista completa, consulte[Regiões e endpoints](#) donoAWSReferência geral. Para obter mais informações sobre como

definir regiões e endpoints no seu código, consulte [AWS Seleção de região no AWS SDK for Java](#) no [Guia do desenvolvedor](#).

Conceitos básicos do JavaScript e do DynamoDB

Neste tutorial, você usa o JavaScript para escrever programas simples a fim de realizar as seguintes operações do Amazon DynamoDB:

- Criar uma tabela chamada `Movies` e carregar dados de amostra no formato JSON.
- Realizar operações `create`, `read`, `update` e `delete` na tabela.
- Executar consultas simples.

À medida que você trabalha neste tutorial, consulte o [AWS SDK for JavaScript Referência de API](#) do.

Pré-requisitos do tutorial

- Faça download e execute o DynamoDB no seu computador. Para mais informações, consulte [Configuração do DynamoDB Local \(versão disponível para download\) \(p. 50\)](#).

Note

Você pode usar as versões disponíveis para download do DynamoDB neste tutorial. Para obter informações sobre como executar o mesmo código para o web service do DynamoDB, consulte o [Resumo e análise do tutorial do JavaScript e do DynamoDB \(p. 122\)](#).

- Configurar um AWS A chave de acesso a ser usada AWSSDKs. Para mais informações, consulte [Configuração do DynamoDB \(serviço Web\) \(p. 57\)](#).
- Configure o AWS SDK for JavaScript. Para fazer isso, adicione ou modifique a seguinte tag de script para suas páginas HTML:

```
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>
```

Note

A versão do AWS SDK for JavaScript pode ter sido atualizada. Para obter a versão mais recente, consulte o [AWS SDK for JavaScript Referência de API](#) do.

- Habilite o [Compartilhamento de recursos de origem cruzada \(CORS\)](#) para que o navegador do computador possa se comunicar com a versão para download do DynamoDB.

Para habilitar o CORS

1. Baixe a extensão gratuita do navegador Chrome ModHeader (ou qualquer outra extensão de navegador que permita modificar cabeçalhos de resposta HTTP).
2. Execute a extensão do navegador Chrome ModHeader e adicione um cabeçalho de resposta HTTP com o nome definido como "Access-Control-Allow-Origin" e um valor "null" ou "*".

Important

Essa configuração é necessária apenas durante a execução desse programa tutorial para JavaScript no computador. Após concluir o tutorial, você deve desabilitar ou remover essa configuração.

3. Agora, você pode executar os arquivos do programa de tutorial JavaScript.

Versão da API 2012-08-10

Caso prefira executar uma versão completa do programa de tutorial JavaScript em vez de realizar as instruções passo-a-passo, faça o seguinte:

1. Faça o download do seguinte arquivo: [MoviesJavaScript.zip](#).
2. Extraia o arquivo `MoviesJavaScript.html` do arquivo.
3. Modifique o arquivo `MoviesJavaScript.html` para usar seu endpoint.
4. Execute o arquivo `MoviesJavaScript.html`.

Etapa 1: Criação de uma tabela do DynamoDB com JavaScript

Nesta etapa, você cria uma tabela chamada `Movies`. A chave primária da tabela é composta dos seguintes atributos:

- `year`– A chave de partição. O `AttributeType` é `N` para número.
- `title`– A chave de classificação. O `AttributeType` é `S` para string.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesCreateTable.html`.

```
<!--
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

This file is licensed under the Apache License, Version 2.0 (the "License").
You may not use this file except in compliance with the License. A copy of
the License is located at

http://aws.amazon.com/apache2.0/

This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the
specific language governing permissions and limitations under the License.
-->
<html>
<head>
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>

<script>
AWS.config.update({
    region: "us-west-2",
    endpoint: 'http://localhost:8000',
    // accessKeyId default can be used while using the downloadable version of DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    accessKeyId: "fakeMyKeyId",
    // secretAccessKey default can be used while using the downloadable version of
    DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    secretAccessKey: "fakeSecretAccessKey"
});

var dynamodb = new AWS.DynamoDB();

function createMovies() {
    var params = {
        TableName : "Movies",
        KeySchema: [
            { AttributeName: "year", KeyType: "HASH"},
```

```
        { AttributeName: "title", KeyType: "RANGE" }
    ],
    AttributeDefinitions: [
        { AttributeName: "year", AttributeType: "N" },
        { AttributeName: "title", AttributeType: "S" }
    ],
    ProvisionedThroughput: {
        ReadCapacityUnits: 5,
        WriteCapacityUnits: 5
    }
};

dynamodb.createTable(params, function(err, data) {
    if (err) {
        document.getElementById('textarea').innerHTML = "Unable to create table: " +
+ "\n" + JSON.stringify(err, undefined, 2);
    } else {
        document.getElementById('textarea').innerHTML = "Created table: " + "\n" +
JSON.stringify(data, undefined, 2);
    }
});

</script>
</head>

<body>
<input id="createTableButton" type="button" value="Create Table"
    onclick="createMovies();;" />
<br><br>
<textarea readonly id= "textarea" style="width:400px; height:800px"></textarea>

</body>
</html>
```

Note

- Você define o endpoint para indicar que está criando a tabela no Amazon DynamoDB em seu computador.
 - Na função `createMovies`, você especifica o nome da tabela, os atributos da chave primária e seus tipos de dados.
 - O `ProvisionedThroughput` é necessário, mas a versão para download do DynamoDB o ignora. (O throughput provisionado está além do escopo deste tutorial.)
2. Abra o arquivo `MoviesCreateTable.html` no navegador.
 3. Selecione Create Table (Criar tabela).

Para saber mais sobre como gerenciar tabelas, consulte [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#).

Etapa 2: Carregar dados de amostra com JavaScript no DynamoDB

Nesta etapa, você preenche a tabela `Movies` com dados de exemplo.

Tópicos

- [Etapa 2.1: Faça download do arquivo de dados de amostra \(p. 104\)](#)
- [Etapa 2.2: Carregar os dados de amostra para a tabela de filmes \(p. 105\)](#)

Esse cenário usa um arquivo de dados de exemplo que contém informações sobre milhares de filmes do Internet Movie Database (IMDb). Os dados de filme estão no formato JSON, conforme mostrado no exemplo a seguir. Para cada filme, existe um `year`, um `title` e um mapa JSON chamado `info`.

```
[  
  {  
    "year" : ... ,  
    "title" : ... ,  
    "info" : { ... }  
  },  
  {  
    "year" : ... ,  
    "title" : ... ,  
    "info" : { ... }  
  },  
  ...  
]
```

Nos dados JSON, observe o seguinte:

- Usamos `year` e `title` como os valores de atributo de chave primária da tabela `Movies`.
- O restante dos valores `info` é armazenado em um único atributo chamado `info`. Este programa ilustra como você pode armazenar o JSON em um atributo do Amazon DynamoDB.

Veja a seguir um exemplo de dados de filme.

```
{  
  "year" : 2013,  
  "title" : "Turn It Down, Or Else!",  
  "info" : {  
    "directors" : [  
      "Alice Smith",  
      "Bob Jones"  
    ],  
    "release_date" : "2013-01-18T00:00:00Z",  
    "rating" : 6.2,  
    "genres" : [  
      "Comedy",  
      "Drama"  
    ],  
    "image_url" : "http://ia.media-imdb.com/images/N/  
09ERWAU7FS797AJ7LU8HN09AMUP908RLlo5JF90EWR7LJKQ7@._V1_SX400_.jpg",  
    "plot" : "A rock band plays their music at high volumes, annoying the neighbors.",  
    "rank" : 11,  
    "running_time_secs" : 5215,  
    "actors" : [  
      "David Matthewman",  
      "Ann Thomas",  
      "Jonathan G. Neff"  
    ]  
  }  
}
```

Etapa 2.1: Faça download do arquivo de dados de amostra

1. Faça download do arquivo de dados de exemplo: [moviedata.zip](#)
2. Extraia o arquivo de dados (`moviedata.json`) do arquivo.

3. Copie e cole o arquivo `moviedata.json` para o diretório atual.

Etapa 2.2: Carregar os dados de amostra para a tabela de filmes

Depois de fazer download dos dados de exemplo, é possível executar o programa a seguir para preencher a tabela `Movies`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesLoadData.html`.

```
<!--
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

This file is licensed under the Apache License, Version 2.0 (the "License").
You may not use this file except in compliance with the License. A copy of
the License is located at

http://aws.amazon.com/apache2.0/

This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the
specific language governing permissions and limitations under the License.
-->
<html>
<head>
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>

<script type="text/javascript">
AWS.config.update({
    region: "us-west-2",
    endpoint: 'http://localhost:8000',
    // accessKeyId default can be used while using the downloadable version of DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    accessKeyId: "fakeMyKeyId",
    // secretAccessKey default can be used while using the downloadable version of
    DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    secretAccessKey: "fakeSecretAccessKey"
});

var docClient = new AWS.DynamoDB.DocumentClient();

function processFile(evt) {
    document.getElementById('textarea').innerHTML = "";
    document.getElementById('textarea').innerHTML += "Importing movies into DynamoDB.
Please wait..." + "\n";
    var file = evt.target.files[0];
    if (file) {
        var r = new FileReader();
        r.onload = function(e) {
            var contents = e.target.result;
            var allMovies = JSON.parse(contents);

            allMovies.forEach(function (movie) {
                document.getElementById('textarea').innerHTML += "Processing: " +
movie.title + "\n";
                var params = {
                    TableName: "Movies",
                    Item: {
                        "year": movie.year,
                        "title": movie.title,
                        "info": movie.info
                    }
                };
                docClient.put(params, function(err, data) {
                    if (err) {
                        console.error("Error:", err);
                    } else {
                        console.log("Success:", data);
                    }
                });
            });
        }
    }
}

</script>
</head>
<body>
<input type="file" id="fileInput" />
<button type="button" onclick="processFile(event);">Carregar
<div id="output" style="border: 1px solid black; padding: 10px; height: 100px; overflow-y: scroll;">
```

```
        }
    };
    docClient.put(params, function (err, data) {
        if (err) {
            document.getElementById('textarea').innerHTML += "Unable to add
movie: " + count + movie.title + "\n";
            document.getElementById('textarea').innerHTML += "Error JSON: "
+ JSON.stringify(err) + "\n";
        } else {
            document.getElementById('textarea').innerHTML += "PutItem
succeeded: " + movie.title + "\n";
            textarea.scrollTop = textarea.scrollHeight;
        }
    });
};

r.readAsText(file);
} else {
    alert("Could not read movie data file");
}
}

</script>
</head>

<body>
<input type="file" id="fileinput" accept='application/json' />
<br><br>
<textarea readonly id= "textarea" style="width:400px; height:800px"></textarea>

<script>
    document.getElementById('fileinput').addEventListener('change', processFile,
false);
</script>
</body>
</html>
```

2. Abra o arquivo `MoviesLoadData.html` no navegador.
3. Selecione **Browse** (Navegador) e carregue o arquivo `moviedata.json`.

Etapa 3: Criar, ler, ler, atualizar e excluir um item

Nesta etapa, você realiza operações de leitura e gravação em um item na tabela `Movies`.

Para saber mais sobre leitura e gravação de dados, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Tópicos

- [Etapa 3.1: Criar um novo item \(p. 106\)](#)
- [Etapa 3.2: Ler um item \(p. 108\)](#)
- [Etapa 3.3: Atualização de um item \(p. 109\)](#)
- [Etapa 3.4: Incrementar um Contador atômico \(p. 111\)](#)
- [Etapa 3.5: Atualização de um item \(condicionalmente\) \(p. 113\)](#)
- [Etapa 3.6: Exclusão de um item \(p. 114\)](#)

Etapa 3.1: Criar um novo item

Nesta etapa, você adiciona um novo item à tabela `Movies`.

1. Copie e cole o programa a seguir em um arquivo chamado MoviesItemOps01.html.

```
<!--
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

This file is licensed under the Apache License, Version 2.0 (the "License").
You may not use this file except in compliance with the License. A copy of
the License is located at

http://aws.amazon.com/apache2.0/

This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the
specific language governing permissions and limitations under the License.
-->
<html>
<head>
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>

<script>
AWS.config.update({
    region: "us-west-2",
    endpoint: 'http://localhost:8000',
    // accessKeyId default can be used while using the downloadable version of DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    accessKeyId: "fakeMyKeyId",
    // secretAccessKey default can be used while using the downloadable version of
    DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    secretAccessKey: "fakeSecretAccessKey"
});

var docClient = new AWS.DynamoDB.DocumentClient();

function createItem() {
    var params = {
        TableName :"Movies",
        Item:{ 
            "year": 2015,
            "title": "The Big New Movie",
            "info":{
                "plot": "Nothing happens at all.",
                "rating": 0
            }
        }
    };
    docClient.put(params, function(err, data) {
        if (err) {
            document.getElementById('textarea').innerHTML = "Unable to add item: " +
"\n" + JSON.stringify(err, undefined, 2);
        } else {
            document.getElementById('textarea').innerHTML = "PutItem succeeded: " +
"\n" + JSON.stringify(data, undefined, 2);
        }
    });
}

</script>
</head>

<body>
<input id="createItem" type="button" value="Create Item" onclick="createItem();" />
<br><br>
```

```
<textarea readonly id= "textarea" style="width:400px; height:800px"></textarea>  
</body>  
</html>
```

Note

A chave primária é obrigatória. Este código adiciona um item que tem uma chave primária (`year`, `title`) e atributos `info`. O atributo `info` armazena o JSON de amostra que fornece mais informações sobre o filme.

2. Abra o arquivo `MoviesItemOps01.html` no navegador.
3. Selecione Create Item (Criar item).

Etapa 3.2: Ler um item

No programa anterior, você adicionou os seguintes itens à tabela.

```
{  
    year: 2015,  
    title: "The Big New Movie",  
    info: {  
        plot: "Nothing happens at all.",  
        rating: 0  
    }  
}
```

Você pode usar o método `get` para ler o item da tabela `Movies`. Você deve especificar os valores de chave primária para que possa ler qualquer item de `Movies`, caso saiba seu `year` e `title`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps02.html`.

```
<!--  
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
  
This file is licensed under the Apache License, Version 2.0 (the "License").  
You may not use this file except in compliance with the License. A copy of  
the License is located at  
  
http://aws.amazon.com/apache2.0/  
  
This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
CONDITIONS OF ANY KIND, either express or implied. See the License for the  
specific language governing permissions and limitations under the License.  
-->  
<html>  
<head>  
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>  
  
<script>  
AWS.config.update({  
    region: "us-west-2",  
    endpoint: 'http://localhost:8000',  
    // accessKeyId default can be used while using the downloadable version of DynamoDB.  
    // For security reasons, do not store AWS Credentials in your files. Use Amazon  
    Cognito instead.  
    accessKeyId: "fakeMyKeyId",  
    // secretAccessKey default can be used while using the downloadable version of  
    // DynamoDB.  
    // For security reasons, do not store AWS Credentials in your files. Use Amazon  
    Cognito instead.  
}</script>
```

```
    secretAccessKey: "fakeSecretAccessKey"
});

var docClient = new AWS.DynamoDB.DocumentClient();

function readItem() {
    var table = "Movies";
    var year = 2015;
    var title = "The Big New Movie";

    var params = {
        TableName: table,
        Key:{ 
            "year": year,
            "title": title
        }
    };
    docClient.get(params, function(err, data) {
        if (err) {
            document.getElementById('textarea').innerHTML = "Unable to read item: " +
"\n" + JSON.stringify(err, undefined, 2);
        } else {
            document.getElementById('textarea').innerHTML = "GetItem succeeded: " +
"\n" + JSON.stringify(data, undefined, 2);
        }
    });
}

</script>
</head>

<body>
<input id="readItem" type="button" value="Read Item" onclick="readItem();"/>
<br><br>
<textarea readonly id= "textarea" style="width:400px; height:800px"></textarea>

</body>
</html>
```

2. Abra o arquivo MoviesItemOps02.html no navegador.
3. Selecione Read Item (Ler item).

Etapa 3.3: Atualização de um item

Você pode usar o método `update` para modificar um item. Você pode atualizar valores de atributos existentes, adicionar novos atributos ou remover atributos.

Neste exemplo, você executa as seguintes atualizações:

- Altere o valor dos atributos existentes (`rating`, `plot`).
- Adicione um novo atributo de lista (`actors`) ao mapa `info` existente.

O item é alterado do seguinte:

```
{
    year: 2015,
    title: "The Big New Movie",
    info: {
        plot: "Nothing happens at all.",
        rating: 0
    }
}
```

```
}
```

Para isto:

```
{
    year: 2015,
    title: "The Big New Movie",
    info: {
        plot: "Everything happens all at once.",
        rating: 5.5,
        actors: ["Larry", "Moe", "Curly"]
    }
}
```

1. Copie e cole o programa a seguir em um arquivo chamado MoviesItemOps03.html.

```
<!--
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

This file is licensed under the Apache License, Version 2.0 (the "License").
You may not use this file except in compliance with the License. A copy of
the License is located at

http://aws.amazon.com/apache2.0/

This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the
specific language governing permissions and limitations under the License.
-->
<html>
<head>
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>

<script>
AWS.config.update({
    region: "us-west-2",
    endpoint: 'http://localhost:8000',
    // accessKeyId default can be used while using the downloadable version of DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    accessKeyId: "fakeMyKeyId",
    // secretAccessKey default can be used while using the downloadable version of
    DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    secretAccessKey: "fakeSecretAccessKey"
});

var docClient = new AWS.DynamoDB.DocumentClient();

function updateItem() {
    var table = "Movies";
    var year = 2015;
    var title = "The Big New Movie";

    var params = {
        TableName:table,
        Key:{
            "year": year,
            "title": title
        },
        UpdateExpression: "set info.rating = :r, info.plot=:p, info.actors=:a",
        ExpressionAttributeValues:{
```

```
        ":r":5.5,
        ":p":"Everything happens all at once.",
        ":a":["Larry", "Moe", "Curly"]
    },
    ReturnValues:"UPDATED_NEW"
};

docClient.update(params, function(err, data) {
    if (err) {
        document.getElementById('textarea').innerHTML = "Unable to update item: " +
"\n" + JSON.stringify(err, undefined, 2);
    } else {
        document.getElementById('textarea').innerHTML = "UpdateItem succeeded: " +
"\n" + JSON.stringify(data, undefined, 2);
    }
});

</script>
</head>

<body>
<input id="updateItem" type="button" value="Update Item" onclick="updateItem();"/>
<br><br>
<textarea readonly id="textarea" style="width:400px; height:800px"></textarea>

</body>
</html>
```

Note

Esse programa usa `UpdateExpression` para descrever todas as atualizações que você deseja realizar no item especificado.

O `ReturnValues` instrui o Amazon DynamoDB a retornar apenas os atributos atualizados ("`UPDATED_NEW`").

2. Abra o arquivo `MoviesItemOps03.html` no navegador.
3. Selecione `Update Item` (Atualizar item).

Etapa 3.4: Incrementar um Contador atômico

O DynamoDB oferece suporte a contadores atômicos, onde você usa `updatePara` aumentar ou diminuir o valor de um atributo sem interferir em outras solicitações de gravação. (Todas as solicitações de gravação são aplicadas na ordem em que são recebidas.)

O programa a seguir mostra como aumentar a `rating` de um filme. Toda vez que você o executa, o programa aumenta esse atributo uma vez.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps04.html`.

```
<!--
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

This file is licensed under the Apache License, Version 2.0 (the "License").
You may not use this file except in compliance with the License. A copy of
the License is located at

http://aws.amazon.com/apache2.0/

This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the
specific language governing permissions and limitations under the License.
```

```
-->
<html>
<head>
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>

<script>
AWS.config.update({
    region: "us-west-2",
    endpoint: 'http://localhost:8000',
    // accessKeyId default can be used while using the downloadable version of DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    accessKeyId: "fakeMyKeyId",
    // secretAccessKey default can be used while using the downloadable version of
    DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    secretAccessKey: "fakeSecretAccessKey"
});

var docClient = new AWS.DynamoDB.DocumentClient();

function increaseRating() {
    var table = "Movies";
    var year = 2015;
    var title = "The Big New Movie";

    var params = {
        TableName:table,
        Key:{
            "year": year,
            "title": title
        },
        UpdateExpression: "set info.rating = info.rating + :val",
        ExpressionAttributeValues:{
            ":val":1
        },
        ReturnValues:"UPDATED_NEW"
    };

    docClient.update(params, function(err, data) {
        if (err) {
            document.getElementById('textarea').innerHTML = "Unable to update rating: " +
            + "\n" + JSON.stringify(err, undefined, 2);
        } else {
            document.getElementById('textarea').innerHTML = "Increase Rating succeeded: " +
            + "\n" + JSON.stringify(data, undefined, 2);
        }
    });
}

</script>
</head>

<body>
<input id="increaseRating" type="button" value="Increase Rating"
    onclick="increaseRating();;" />
<br><br>
<textarea readonly id= "textarea" style="width:400px; height:800px"></textarea>
</body>
</html>
```

2. Abra o arquivo MoviesItemOps04.html no navegador.
3. Selecione Increase Rating (Aumentar a avaliação).

Etapa 3.5: Atualização de um item (condicionalmente)

O programa a seguir mostra como usar `UpdateItem` com uma condição. Se a condição for verdadeira, a atualização será bem-sucedida; caso contrário, a atualização não será realizada.

Neste caso, o item será atualizado somente se houver mais de três atores no filme.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps05.html`.

```
<!--
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

This file is licensed under the Apache License, Version 2.0 (the "License").
You may not use this file except in compliance with the License. A copy of
the License is located at

http://aws.amazon.com/apache2.0/

This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the
specific language governing permissions and limitations under the License.
-->
<html>
<head>
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>

<script>
AWS.config.update({
  region: "us-west-2",
  endpoint: 'http://localhost:8000',
  // accessKeyId default can be used while using the downloadable version of DynamoDB.
  // For security reasons, do not store AWS Credentials in your files. Use Amazon
  Cognito instead.
  accessKeyId: "fakeMyKeyId",
  // secretAccessKey default can be used while using the downloadable version of
  DynamoDB.
  // For security reasons, do not store AWS Credentials in your files. Use Amazon
  Cognito instead.
  secretAccessKey: "fakeSecretAccessKey"
});

var docClient = new AWS.DynamoDB.DocumentClient();

function conditionalUpdate() {
  var table = "Movies";
  var year = 2015;
  var title = "The Big New Movie";

  // Conditional update (will fail)
  var params = {
    TableName:table,
    Key:{
      "year": year,
      "title": title
    },
    UpdateExpression: "remove info.actors[0]",
    ConditionExpression: "size(info.actors) > :num",
    ExpressionAttributeValues:{
      ":num":3
    },
    ReturnValues:"UPDATED_NEW"
  };

  docClient.update(params, function(err, data) {
```

```
if (err) {
    document.getElementById('textarea').innerHTML = "The conditional update
failed: " + "\n" + JSON.stringify(err, undefined, 2);
} else {
    document.getElementById('textarea').innerHTML = "The conditional update
succeeded: " + "\n" + JSON.stringify(data, undefined, 2);
}
});

</script>
</head>

<body>
<input id="conditionalUpdate" type="button" value="Conditional Update"
onclick="conditionalUpdate();"/>
<br><br>
<textarea readonly id= "textarea" style="width:400px; height:800px"></textarea>
</body>
</html>
```

2. Abra o arquivo MoviesItemOps05.html no navegador.
3. Selecione Conditional Update (Atualização condicional).

O programa deve falhar com a mensagem a seguir:

A atualização condicional falhou.

Isto acontece porque o filme tem três atores, mas a condição está em busca de mais de três atores.

4. Modifique o programa para que ConditionExpression fique semelhante ao seguinte.

```
ConditionExpression: "size(info.actors) >= :num",
```

A condição agora é maior ou igual a 3 em vez de maior que 3.

5. Execute o programa novamente. A operação updateItem agora deve ser bem-sucedida.

Etapa 3.6: Exclusão de um item

Você pode usar o método delete para excluir um item, especificando sua chave primária. Se desejar, você pode fornecer uma ConditionExpression para evitar a exclusão do item, se a condição não for atendida.

No exemplo a seguir, você tenta excluir um item de filme específico se a sua classificação for 5 ou menos.

1. Copie e cole o programa a seguir em um arquivo chamado MoviesItemOps06.html.

```
<!--
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

This file is licensed under the Apache License, Version 2.0 (the "License").
You may not use this file except in compliance with the License. A copy of
the License is located at

http://aws.amazon.com/apache2.0/

This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the
specific language governing permissions and limitations under the License.
-->
```

```
<html>
<head>
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>

<script>
AWS.config.update({
    region: "us-west-2",
    endpoint: 'http://localhost:8000',
    // accessKeyId default can be used while using the downloadable version of DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    accessKeyId: "fakeMyKeyId",
    // secretAccessKey default can be used while using the downloadable version of
    DynamoDB.
    // For security reasons, do not store AWS Credentials in your files. Use Amazon
    Cognito instead.
    secretAccessKey: "fakeSecretAccessKey"
});

var docClient = new AWS.DynamoDB.DocumentClient();

function conditionalDelete() {
    var table = "Movies";
    var year = 2015;
    var title = "The Big New Movie";

    var params = {
        TableName:table,
        Key:{
            "year":year,
            "title":title
        },
        ConditionExpression:"info.rating <= :val",
        ExpressionAttributeValues: {
            ":val": 5.0
        }
    };

    docClient.delete(params, function(err, data) {
        if (err) {
            document.getElementById('textarea').innerHTML = "The conditional delete
failed: " + "\n" + JSON.stringify(err, undefined, 2);
        } else {
            document.getElementById('textarea').innerHTML = "The conditional delete
succeeded: " + "\n" + JSON.stringify(data, undefined, 2);
        }
    });
}

</script>
</head>

<body>
<input id="conditionalDelete" type="button" value="Conditional Delete"
onclick="conditionalDelete();"/>
<br><br>
<textarea readonly id= "textarea" style="width:400px; height:800px"></textarea>

</body>
</html>
```

2. Abra o arquivo MoviesItemOps06.html no navegador.
3. Selecione Conditional Delete (Exclusão condicional).

O programa deve falhar com a mensagem a seguir:

A exclusão condicional falhou

Isso ocorre porque a classificação desse filme em especial é maior que 5.

4. Modifique o programa para remover a condição de `params`.

```
var params = {  
    TableName:table,  
    Key:{  
        "title":title,  
        "year":year  
    }  
};
```

5. Execute o programa novamente. A exclusão é bem-sucedida, pois você removeu a condição.

Etapa 4: Consultar e verificar os dados com JavaScript e DynamoDB

Você pode usar o método `query` para recuperar os dados de uma tabela. Você deve especificar um valor de chave de partição; a chave de classificação é opcional.

A chave primária da tabela `Movies` é composta pelo seguinte:

- `year`— A chave de partição. O tipo de atributo é número.
- `title`— A chave de classificação. O tipo de atributo é string.

Para encontrar todos os filmes lançados durante um ano, você precisa especificar somente o `year`. Você também pode fornecer o `title` para recuperar um subconjunto de filmes com base em uma certa condição (na chave de classificação); por exemplo, para encontrar filmes lançados em 2014 que tenham um título que comece com a letra "A".

Além do método `query`, é possível usar o método `scan` para recuperar todos os dados da tabela.

Para saber mais sobre como consultar e verificar dados, consulte [Como trabalhar com consultas no DynamoDB \(p. 490\)](#) e [Como trabalhar com verificações no DynamoDB \(p. 508\)](#), respectivamente.

Tópicos

- [Etapa 4.1: Consulta — Todos os filmes lançados em um ano \(p. 116\)](#)
- [Etapa 4.2: Consulta — Todos os filmes lançados em um ano com determinados títulos \(p. 118\)](#)
- [Etapa 4.3: Scan \(p. 119\)](#)

Etapa 4.1: Consulta — Todos os filmes lançados em um ano

O programa incluído nesta etapa recupera todos os filmes lançados no `year` 1985.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesQuery01.html`.

```
<!--  
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
  
This file is licensed under the Apache License, Version 2.0 (the "License").  
You may not use this file except in compliance with the License. A copy of  
the License is located at
```

```
http://aws.amazon.com/apache2.0/  
  
This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
CONDITIONS OF ANY KIND, either express or implied. See the License for the  
specific language governing permissions and limitations under the License.  
-->  
<html>  
<head>  
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>  
  
<script>  
AWS.config.update({  
    region: "us-west-2",  
    endpoint: 'http://localhost:8000',  
    // accessKeyId default can be used while using the downloadable version of DynamoDB.  
    // For security reasons, do not store AWS Credentials in your files. Use Amazon  
    Cognito instead.  
    accessKeyId: "fakeMyKeyId",  
    // secretAccessKey default can be used while using the downloadable version of  
    // DynamoDB.  
    // For security reasons, do not store AWS Credentials in your files. Use Amazon  
    Cognito instead.  
    secretAccessKey: "fakeSecretAccessKey"  
});  
  
var docClient = new AWS.DynamoDB.DocumentClient();  
  
function queryData() {  
    document.getElementById('textarea').innerHTML += "Querying for movies from 1985."  
  
    var params = {  
        TableName : "Movies",  
        KeyConditionExpression: "#yr = :yyyy",  
        ExpressionAttributeNames:{  
            "#yr": "year"  
        },  
        ExpressionAttributeValues: {  
            ":yyyy":1985  
        }  
    };  
  
    docClient.query(params, function(err, data) {  
        if (err) {  
            document.getElementById('textarea').innerHTML += "Unable to query. Error: " + "\n" + JSON.stringify(err, undefined, 2);  
        } else {  
            document.getElementById('textarea').innerHTML += "Querying for movies from 1985: " + "\n" + JSON.stringify(data, undefined, 2);  
        }  
    });  
}  
  
</script>  
</head>  
  
<body>  
<input id="queryData" type="button" value="Query" onclick="queryData();"/>  
<br><br>  
<textarea readonly id= "textarea" style="width:400px; height:800px"></textarea>  
  
</body>  
</html>
```

Note

`ExpressionAttributeNames` fornece substituição de nome. Isso é usado porque `year` é uma palavra reservada no Amazon DynamoDB. Não é possível usá-la diretamente em nenhuma expressão, incluindo `KeyConditionExpression`. Por esse motivo, você usa o nome de atributo de expressão `#yr`.

`ExpressionAttributeValues` fornece substituição de valor. Ele é usado porque não é possível usar literais em nenhuma expressão, incluindo `KeyConditionExpression`. Por esse motivo, você usa o valor de atributo de expressão `:yyyy`.

2. Abra o arquivo `MoviesQuery01.html` no navegador.
3. Selecione Query (Consultar).

Note

O programa anterior mostra como consultar uma tabela por seus atributos de chave primária. No DynamoDB, você pode opcionalmente criar um ou mais índices secundários em uma tabela e consultá-las da mesma forma que consulta uma tabela. Os índices secundários oferecem aos seus aplicativos mais flexibilidade ao permitir consultas nos atributos que não são chave. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 4.2: Consulta — Todos os filmes lançados em um ano com determinados títulos

O programa incluído nesta etapa recupera todos os filmes lançados no `year` 1992, cujo `title` começa com a letra "A" até a letra "L".

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesQuery02.html`.

```
<!--
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

This file is licensed under the Apache License, Version 2.0 (the "License").
You may not use this file except in compliance with the License. A copy of
the License is located at

http://aws.amazon.com/apache2.0/

This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the
specific language governing permissions and limitations under the License.
-->
<html>
<head>
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>

<script>
AWS.config.update({
  region: "us-west-2",
  endpoint: 'http://localhost:8000',
  // accessKeyId default can be used while using the downloadable version of DynamoDB.
  // For security reasons, do not store AWS Credentials in your files. Use Amazon
  Cognito instead.
  accessKeyId: "fakeMyKeyId",
  // secretAccessKey default can be used while using the downloadable version of
  DynamoDB.
  // For security reasons, do not store AWS Credentials in your files. Use Amazon
  Cognito instead.
  secretAccessKey: "fakeSecretAccessKey"
})
```

```
});

var docClient = new AWS.DynamoDB.DocumentClient();

function queryData() {
    document.getElementById('textarea').innerHTML += "Querying for movies from 1985.';

    var params = {
        TableName : "Movies",
        ProjectionExpression:"#yr, title, info.genres, info.actors[0]",
        KeyConditionExpression: "#yr = :yyyy and title between :letter1 and :letter2",
        ExpressionAttributeNames:{
            "#yr": "year"
        },
        ExpressionAttributeValues: {
            ":yyyy":1992,
            ":letter1": "A",
            ":letter2": "L"
        }
    };

    docClient.query(params, function(err, data) {
        if (err) {
            document.getElementById('textarea').innerHTML += "Unable to query. Error: " +
            + "\n" + JSON.stringify(err, undefined, 2);
        } else {
            document.getElementById('textarea').innerHTML += "Querying for movies
from 1992 - titles A-L, with genres and lead actor: " + "\n" + JSON.stringify(data,
undefined, 2);
        }
    });
}

</script>
</head>

<body>
<input id="queryData" type="button" value="Query" onclick="queryData();" />
<br><br>
<textarea readonly id= "textarea" style="width:400px; height:800px"></textarea>

</body>
</html>
```

2. Abra o arquivo MoviesQuery02.html no navegador.
3. Selecione Query (Consultar).

Etapa 4.3: Scan

O método `scan` lê todos os itens da tabela inteira e retorna todos os dados nela contidos. Você pode fornecer uma `filter_expression` opcional, para que apenas os itens que correspondem aos seus critérios sejam retornados. No entanto, o filtro é aplicado somente depois que a tabela inteira foi verificada.

O programa a seguir verifica a tabela `Movies` inteira, que contém aproximadamente 5.000 itens. A verificação especifica o filtro opcional para recuperar somente os filmes da década de 1950 (aproximadamente 100 itens) e descartar todos os outros.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesScan.html`.

```
<!--
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
This file is licensed under the Apache License, Version 2.0 (the "License").  
You may not use this file except in compliance with the License. A copy of  
the License is located at  
  
http://aws.amazon.com/apache2.0/  
  
This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
CONDITIONS OF ANY KIND, either express or implied. See the License for the  
specific language governing permissions and limitations under the License.  
-->  
<html>  
<head>  
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>  
  
<script>  
AWS.config.update({  
    region: "us-west-2",  
    endpoint: 'http://localhost:8000',  
    // accessKeyId default can be used while using the downloadable version of DynamoDB.  
    // For security reasons, do not store AWS Credentials in your files. Use Amazon  
    Cognito instead.  
    accessKeyId: "fakeMyKeyId",  
    // secretAccessKey default can be used while using the downloadable version of  
    DynamoDB.  
    // For security reasons, do not store AWS Credentials in your files. Use Amazon  
    Cognito instead.  
    secretAccessKey: "fakeSecretAccessKey"  
});  
  
var docClient = new AWS.DynamoDB.DocumentClient();  
  
function scanData() {  
    document.getElementById('textarea').innerHTML += "Scanning Movies table." + "\n";  
  
    var params = {  
        TableName: "Movies",  
        ProjectionExpression: "#yr, title, info.rating",  
        FilterExpression: "#yr between :start_yr and :end_yr",  
        ExpressionAttributeNames: {  
            "#yr": "year",  
        },  
        ExpressionAttributeValues: {  
            ":start_yr": 1950,  
            ":end_yr": 1959  
        }  
    };  
  
    docClient.scan(params, onScan);  
  
    function onScan(err, data) {  
        if (err) {  
            document.getElementById('textarea').innerHTML += "Unable to scan the table:  
" + "\n" + JSON.stringify(err, undefined, 2);  
        } else {  
            // Print all the movies  
            document.getElementById('textarea').innerHTML += "Scan succeeded. " + "\n";  
            data.Items.forEach(function(movie) {  
                document.getElementById('textarea').innerHTML += movie.year + ":" +  
                    movie.title + " - rating: " + movie.info.rating + "\n";  
            });  
  
            // Continue scanning if we have more movies (per scan 1MB limitation)  
            document.getElementById('textarea').innerHTML += "Scanning for more..." +  
"\n";  
            params.ExclusiveStartKey = data.LastEvaluatedKey;  
            docClient.scan(params, onScan);  
        }  
    }  
}
```

```
        }
    }

</script>
</head>

<body>
<input id="scanData" type="button" value="Scan" onclick="scanData();"/>
<br><br>
<textarea readonly id="textarea" style="width:400px; height:800px"></textarea>
</body>
</html>
```

No código, observe o seguinte:

- `ProjectionExpression` especifica os atributos que você deseja no resultado da verificação.
 - `FilterExpression` especifica uma condição que retorna apenas os itens que satisfazem à condição. Todos os outros itens são descartados.
2. Abra o arquivo `MoviesScan.html` no navegador.
 3. Selecione Scan (Verificar).

Note

Também é possível usar a operação `Scan` com quaisquer índices secundários criados na tabela. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 5: Excluir a tabela com JavaScript

Para excluir a tabela `Movies`:

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesDeleteTable.html`.

```
<!--
Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

This file is licensed under the Apache License, Version 2.0 (the "License").
You may not use this file except in compliance with the License. A copy of
the License is located at

http://aws.amazon.com/apache2.0/

This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the
specific language governing permissions and limitations under the License.
-->
<html>
<head>
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>

<script>
AWS.config.update({
  region: "us-west-2",
  endpoint: 'http://localhost:8000',
  // accessKeyId default can be used while using the downloadable version of DynamoDB.
  // For security reasons, do not store AWS Credentials in your files. Use Amazon
  Cognito instead.
  accessKeyId: "fakeMyKeyId",
```

```
// secretAccessKey default can be used while using the downloadable version of
DynamoDB.
// For security reasons, do not store AWS Credentials in your files. Use Amazon
Cognito instead.
secretAccessKey: "fakeSecretAccessKey"
});

var dynamodb = new AWS.DynamoDB();

function deleteMovies() {
    var params = {
        TableName : "Movies"
    };

    dynamodb.deleteTable(params, function(err, data) {
        if (err) {
            document.getElementById('textarea').innerHTML = "Unable to delete table: "
+ "\n" + JSON.stringify(err, undefined, 2);
        } else {
            document.getElementById('textarea').innerHTML = "Table deleted.";
        }
    });
}

</script>
</head>

<body>
<input id="deleteTableButton" type="button" value="Delete Table"
onclick="deleteMovies();;" />
<br><br>
<textarea readonly id= "textarea" style="width:400px; height:800px"></textarea>

</body>
</html>
```

2. Abra o arquivo MoviesDeleteTable.html no navegador.
3. Selecione Delete Table (Excluir tabela).

Resumo e análise do tutorial do JavaScript e do DynamoDB

Neste tutorial, você criou o Movies no Amazon DynamoDB no seu computador e realizou operações básicas. A versão para download do DynamoDB é útil durante o desenvolvimento e o teste de aplicativos. No entanto, quando você estiver pronto para executar o aplicativo em um ambiente de produção, precisará modificar o código para que ele use o serviço web do DynamoDB.

Atualização do AWSRegião de configuração para usar o web service do DynamoDB

Para usar o web service do DynamoDB, atualize o AWSRegião em seu aplicativo. Também é necessário certificar-se de que o Amazon Cognito esteja disponível na mesma região, para que os scripts do navegador possam ser autenticados com êxito.

```
AWS.config.update({region: "aws-region"});
```

Por exemplo, se você quiser usar a região us-west-2, defina a região a seguir.

```
AWS.config.update({region: "us-west-2"});
```

O programa agora usa o web service do Amazon DynamoDB na região Oeste dos EUA (Oregon).

O DynamoDB está disponível em AWS Regiões do mundo todo. Para obter a lista completa, consulte [Regiões e endpoints](#) no [AWS Referência geral](#). Para obter mais informações sobre como definir regiões e endpoints no seu código, consulte [Definir a região](#) no [AWS SDK for JavaScript Guia de conceitos básicos](#).

Configurar o AWS Credenciais em seus arquivos usando o Amazon Cognito

A maneira recomendada de obter AWS Para seus aplicativos móveis e da Web é usar o Amazon Cognito. O Amazon Cognito ajuda você a evitar a aplicação de código fixo do seu AWS Credenciais do nos seus arquivos. O Amazon Cognito usa o AWS Identity and Access Management Funções do (IAM) com o objetivo de gerar credenciais temporárias para os usuários autenticados e não autenticados do aplicativo.

Para mais informações, consulte [Configurar o AWS Credenciais em seus arquivos usando o Amazon Cognito \(p. 1018\)](#).

Conceitos básicos sobre Node.js e o DynamoDB

Neste tutorial, você usa o AWS SDK for JavaScript Para escrever aplicativos simples a fim de executar as seguintes operações do Amazon DynamoDB:

- Criar uma tabela chamada `Movies` e carregar dados de amostra no formato JSON.
- Realizar operações `create`, `read`, `update` e `delete` na tabela.
- Executar consultas simples.

À medida que você trabalha neste tutorial, consulte o [AWS SDK for JavaScript Referência de API](#) do.

Pré-requisitos do tutorial

- Faça download e execute o DynamoDB no seu computador. Para mais informações, consulte [Configuração do DynamoDB Local \(versão disponível para download\) \(p. 50\)](#).

Note

Você pode usar as versões disponíveis para download do DynamoDB neste tutorial. Para obter informações sobre como executar o mesmo código para o serviço web do DynamoDB, consulte [o Summary \(p. 140\)](#).

- Configurar um AWS Para usar a chave de acesso do AWS SDKs. Para mais informações, consulte [Configuração do DynamoDB \(serviço Web\) \(p. 57\)](#).
- Configurar o AWS SDK for JavaScript:
 - Instale o [Node.js](#).
 - Instale o [AWS SDK for JavaScript](#).

Para obter mais informações, consulte o [AWS SDK for JavaScript Guia de conceitos básicos](#) do.

Etapa 1: Crie uma tabela com no DynamoDB comAWS SDK for JavaScript

Nesta etapa, você cria uma tabela chamada `Movies`. A chave primária da tabela é composta dos seguintes atributos:

- `year`- A chave de partição. O `AttributeType` é `N` para número.
- `title`- A chave de classificação. O `AttributeType` é `S` para string.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesCreateTable.js`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
var AWS = require("aws-sdk");  
  
AWS.config.update({  
    region: "us-west-2",  
    endpoint: "http://localhost:8000"  
});  
  
var dynamodb = new AWS.DynamoDB();  
  
var params = {  
    TableName : "Movies",  
    KeySchema: [  
        { AttributeName: "year", KeyType: "HASH"}, //Partition key  
        { AttributeName: "title", KeyType: "RANGE" } //Sort key  
    ],  
    AttributeDefinitions: [  
        { AttributeName: "year", AttributeType: "N" },  
        { AttributeName: "title", AttributeType: "S" }  
    ],  
    ProvisionedThroughput: {  
        ReadCapacityUnits: 10,  
        WriteCapacityUnits: 10  
    }  
};  
  
dynamodb.createTable(params, function(err, data) {  
    if (err) {  
        console.error("Unable to create table. Error JSON:", JSON.stringify(err, null, 2));  
    } else {  
        console.log("Created table. Table description JSON:", JSON.stringify(data, null, 2));  
    }  
});
```

Note

- Você define o endpoint para indicar que está criando a tabela no Amazon DynamoDB em seu computador.
 - Na chamada `createTable`, você especifica o nome da tabela, os atributos da chave primária e seus tipos de dados.
 - O `ProvisionedThroughput` parâmetro é necessário, mas a versão para download do DynamoDB o ignora. (O throughput provisionado está além do escopo deste tutorial.)
2. Para executar o programa, digite o comando a seguir.

```
node MoviesCreateTable.js
```

Para saber mais sobre como gerenciar tabelas, consulte [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#).

Etapa 2: Carregar dados de exemplo no DynamoDB com o AWS SDK for JavaScript

Nesta etapa, você preenche a tabela `Movies` com dados de exemplo.

Tópicos

- [Etapa 2.1: Faça download do arquivo de dados de exemplo \(p. 126\)](#)
- [Etapa 2.2: Carregar os dados de exemplo para a tabela de filmes \(p. 126\)](#)

Você usa um arquivo de dados de exemplo que contém informações sobre milhares de filmes do Internet Movie Database (IMDb). Os dados de filme estão no formato JSON, conforme mostrado no exemplo a seguir. Para cada filme, existe um `year`, um `title` e um mapa JSON chamado `info`.

```
[  
  {  
    "year" : ... ,  
    "title" : ... ,  
    "info" : { ... }  
  },  
  {  
    "year" : ... ,  
    "title" : ... ,  
    "info" : { ... }  
  },  
  ...  
]
```

Nos dados JSON, observe o seguinte:

- Usamos `year` e `title` como os valores de atributo de chave primária da tabela `Movies`.
- O restante dos valores `info` é armazenado em um único atributo chamado `info`. Este programa ilustra como você pode armazenar o JSON em um atributo do Amazon DynamoDB.

Veja a seguir um exemplo de dados de filme.

```
{  
    "year" : 2013,  
    "title" : "Turn It Down, Or Else!",  
    "info" : {  
        "directors" : [  
            "Alice Smith",  
            "Bob Jones"  
        ],  
        "release_date" : "2013-01-18T00:00:00Z",  
        "rating" : 6.2,  
        "genres" : [  
            "Comedy",  
            "Drama"  
        ],  
        "image_url" : "http://ia.media-imdb.com/images/N/  
O9ERWAU7FS797AJ7LU8HN09AMUP908RLlo5JF90EWR7LJKQ7@._V1_SX400_.jpg",  
        "plot" : "A rock band plays their music at high volumes, annoying the neighbors.",  
        "rank" : 11,  
        "running_time_secs" : 5215,  
        "actors" : [  
            "David Matthewman",  
            "Ann Thomas",  
            "Jonathan G. Neff"  
        ]  
    }  
}
```

Etapa 2.1: Faça download do arquivo de dados de exemplo

1. Faça download do arquivo de dados de exemplo: [moviedata.zip](#)
2. Extraia o arquivo de dados (`moviedata.json`) do arquivo.
3. Copie e cole o arquivo `moviedata.json` para o diretório atual.

Etapa 2.2: Carregar os dados de exemplo para a tabela de filmes

Depois de fazer download dos dados de exemplo, é possível executar o programa a seguir para preencher a tabela `Movies`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesLoadData.js`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
var AWS = require("aws-sdk");  
var fs = require('fs');  
  
AWS.config.update({  
    region: "us-west-2",  
    endpoint: "http://localhost:8000"  
});
```

```
var docClient = new AWS.DynamoDB.DocumentClient();

console.log("Importing movies into DynamoDB. Please wait.");

var allMovies = JSON.parse(fs.readFileSync('moviedata.json', 'utf8'));
allMovies.forEach(function(movie) {
    var params = {
        TableName: "Movies",
        Item: {
            "year": movie.year,
            "title": movie.title,
            "info": movie.info
        }
    };

    docClient.put(params, function(err, data) {
        if (err) {
            console.error("Unable to add movie", movie.title, ". Error JSON:", JSON.stringify(err, null, 2));
        } else {
            console.log("PutItem succeeded:", movie.title);
        }
    });
});
```

2. Para executar o programa, digite o comando a seguir.

```
node MoviesLoadData.js
```

Etapa 3: Criar, ler, atualizar e excluir um item

Nesta etapa, você realiza operações de leitura e gravação em um item na tabela `Movies`.

Para saber mais sobre leitura e gravação de dados, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Tópicos

- [Etapa 3.1: Criar um novo item \(p. 127\)](#)
- [Etapa 3.2: Ler um item \(p. 128\)](#)
- [Etapa 3.3: Atualização de um item \(p. 129\)](#)
- [Etapa 3.4: Incrementar um Contador atômico \(p. 131\)](#)
- [Etapa 3.5: Atualização de um item \(condicionalmente\) \(p. 132\)](#)
- [Etapa 3.6: Exclusão de um item \(p. 133\)](#)

Etapa 3.1: Criar um novo item

Nesta etapa, você adiciona um novo item à tabela `Movies`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps01.js`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at
```

```
/*
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
var AWS = require("aws-sdk");

AWS.config.update({
    region: "us-west-2",
    endpoint: "http://localhost:8000"
});

var docClient = new AWS.DynamoDB.DocumentClient();

var table = "Movies";

var year = 2015;
var title = "The Big New Movie";

var params = {
    TableName:table,
    Item:{
        "year": year,
        "title": title,
        "info": {
            "plot": "Nothing happens at all.",
            "rating": 0
        }
    }
};

console.log("Adding a new item...");
docClient.put(params, function(err, data) {
    if (err) {
        console.error("Unable to add item. Error JSON:", JSON.stringify(err, null, 2));
    } else {
        console.log("Added item:", JSON.stringify(data, null, 2));
    }
});
```

Note

A chave primária é obrigatória. Este código adiciona um item que tem uma chave primária (`year`, `title`) e atributos `info`. O atributo `info` armazena o JSON de amostra que fornece mais informações sobre o filme.

2. Para executar o programa, digite o comando a seguir.

```
node MoviesItemOps01.js
```

Etapa 3.2: Ler um item

No programa anterior, você adicionou os seguintes itens à tabela.

```
{
    year: 2015,
    title: "The Big New Movie",
    info: {
        plot: "Nothing happens at all.",
        rating: 0
    }
}
```

}

Você pode usar o método `get` para ler o item da tabela `Movies`. Você deve especificar os valores de chave primária para que possa ler qualquer item de `Movies`, caso saiba seu `year` e `title`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps02.js`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
var AWS = require("aws-sdk");  
  
AWS.config.update({  
    region: "us-west-2",  
    endpoint: "http://localhost:8000"  
});  
  
var docClient = new AWS.DynamoDB.DocumentClient();  
  
var table = "Movies";  
  
var year = 2015;  
var title = "The Big New Movie";  
  
var params = {  
    TableName: table,  
    Key:{  
        "year": year,  
        "title": title  
    }  
};  
  
docClient.get(params, function(err, data) {  
    if (err) {  
        console.error("Unable to read item. Error JSON:", JSON.stringify(err, null, 2));  
    } else {  
        console.log("GetItem succeeded:", JSON.stringify(data, null, 2));  
    }  
});
```

2. Para executar o programa, digite o comando a seguir.

```
node MoviesItemOps02.js
```

Etapa 3.3: Atualização de um item

Você pode usar o método `update` para modificar um item existente. Você pode atualizar valores de atributos existentes, adicionar novos atributos ou remover atributos.

Neste exemplo, você executa as seguintes atualizações:

- Altere o valor dos atributos existentes (`rating`, `plot`).
- Adicione um novo atributo de lista (`actors`) ao mapa `info` existente.

Anteriormente, você adicionou o seguinte item à tabela.

```
{  
    year: 2015,  
    title: "The Big New Movie",  
    info: {  
        plot: "Nothing happens at all.",  
        rating: 0  
    }  
}
```

O item é atualizado conforme o seguinte.

```
{  
    year: 2015,  
    title: "The Big New Movie",  
    info: {  
        plot: "Everything happens all at once.",  
        rating: 5.5,  
        actors: ["Larry", "Moe", "Curly"]  
    }  
}
```

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps03.js`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
var AWS = require("aws-sdk");  
  
AWS.config.update({  
    region: "us-west-2",  
    endpoint: "http://localhost:8000"  
});  
  
var docClient = new AWS.DynamoDB.DocumentClient()  
  
var table = "Movies";  
  
var year = 2015;  
var title = "The Big New Movie";  
  
// Update the item, unconditionally,  
  
var params = {  
    TableName:table,  
    Key:{
```

```
        "year": year,
        "title": title
    },
    UpdateExpression: "set info.rating = :r, info.plot=:p, info.actors=:a",
    ExpressionAttributeValues:{
        ":r":5.5,
        ":p":"Everything happens all at once.",
        ":a":["Larry", "Moe", "Curly"]
    },
    ReturnValue:"UPDATED_NEW"
};

console.log("Updating the item...");
docClient.update(params, function(err, data) {
    if (err) {
        console.error("Unable to update item. Error JSON:", JSON.stringify(err, null, 2));
    } else {
        console.log("UpdateItem succeeded:", JSON.stringify(data, null, 2));
    }
});
```

Note

Esse programa usa `UpdateExpression` para descrever todas as atualizações que você deseja realizar no item especificado.
O `ReturnValues` instrui o DynamoDB a retornar somente os atributos atualizados ("`UPDATED_NEW`").

2. Para executar o programa, digite o comando a seguir.

```
node MoviesItemOps03.js
```

Etapa 3.4: Incrementar um Contador atômico

O DynamoDB oferece suporte a contadores atômicos, onde você usa `update` para aumentar ou diminuir o valor de um atributo existente sem interferir em outras solicitações de gravação. (Todas as solicitações de gravação são aplicadas na ordem em que são recebidas.)

O programa a seguir mostra como aumentar a `rating` de um filme. Toda vez que o programa for executado, o atributo é incrementado uma vez.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps04.js`.

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
var AWS = require("aws-sdk");

AWS.config.update({
    region: "us-west-2",
```

```
        endpoint: "http://localhost:8000"
    });

var docClient = new AWS.DynamoDB.DocumentClient()

var table = "Movies";

var year = 2015;
var title = "The Big New Movie";

// Increment an atomic counter

var params = {
    TableName:table,
    Key:{
        "year": year,
        "title": title
    },
    UpdateExpression: "set info.rating = info.rating + :val",
    ExpressionAttributeValues:{
        ":val": 1
    },
    ReturnValues:"UPDATED_NEW"
};

console.log("Updating the item...");
docClient.update(params, function(err, data) {
    if (err) {
        console.error("Unable to update item. Error JSON:", JSON.stringify(err, null, 2));
    } else {
        console.log("UpdateItem succeeded:", JSON.stringify(data, null, 2));
    }
});
```

2. Para executar o programa, digite o comando a seguir.

```
node MoviesItemOps04.js
```

Etapa 3.5: Atualização de um item (condicionalmente)

O programa a seguir mostra como usar `UpdateItem` com uma condição. Se a condição for verdadeira, a atualização será bem-sucedida; caso contrário, a atualização não será realizada.

Neste caso, o item será atualizado somente se houver mais de três atores no filme.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps05.js`.

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
var AWS = require("aws-sdk");
```

```
AWS.config.update({
  region: "us-west-2",
  endpoint: "http://localhost:8000"
});

var docClient = new AWS.DynamoDB.DocumentClient()

var table = "Movies";

var year = 2015;
var title = "The Big New Movie";

// Conditional update (will fail)

var params = {
  TableName:table,
  Key:{
    "year": year,
    "title": title
  },
  UpdateExpression: "remove info.actors[0]",
  ConditionExpression: "size(info.actors) > :num",
  ExpressionAttributeValues:{
    ":num": 3
  },
  ReturnValues:"UPDATED_NEW"
};

console.log("Attempting a conditional update...");
docClient.update(params, function(err, data) {
  if (err) {
    console.error("Unable to update item. Error JSON:", JSON.stringify(err, null, 2));
  } else {
    console.log("UpdateItem succeeded:", JSON.stringify(data, null, 2));
  }
});
```

2. Para executar o programa, digite o comando a seguir.

```
node MoviesItemOps05.js
```

O programa deve falhar com a mensagem a seguir.

A solicitação condicional falhou

Isto acontece porque o filme tem três atores, mas a condição está em busca de mais de três atores.

3. Modifique o programa para que `ConditionExpression` fique semelhante ao seguinte.

```
ConditionExpression: "size(info.actors) >= :num",
```

A condição agora é maior ou igual a 3 em vez de maior que 3.

4. Execute o programa novamente. A operação `updateItem` agora deve ser bem-sucedida.

Etapa 3.6: Exclusão de um item

Você pode usar o método `delete` para excluir um item, especificando sua chave primária. Se desejar, você pode fornecer uma `ConditionExpression` para evitar a exclusão do item, se a condição não for atendida.

No exemplo a seguir, você tenta excluir um item de filme específico se a sua classificação for 5 ou menos.

1. Copie e cole o programa a seguir em um arquivo chamado MoviesItemOps06.js.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
var AWS = require("aws-sdk");  
  
AWS.config.update({  
    region: "us-west-2",  
    endpoint: "http://localhost:8000"  
});  
  
var docClient = new AWS.DynamoDB.DocumentClient();  
  
var table = "Movies";  
  
var year = 2015;  
var title = "The Big New Movie";  
  
var params = {  
    TableName:table,  
    Key:{  
        "year": year,  
        "title": title  
    },  
    ConditionExpression:"info.rating <= :val",  
    ExpressionAttributeValues: {  
        ":val": 5.0  
    }  
};  
  
console.log("Attempting a conditional delete...");  
docClient.delete(params, function(err, data) {  
    if (err) {  
        console.error("Unable to delete item. Error JSON:", JSON.stringify(err, null, 2));  
    } else {  
        console.log("DeleteItem succeeded:", JSON.stringify(data, null, 2));  
    }  
});
```

2. Para executar o programa, digite o comando a seguir.

```
node MoviesItemOps06.js
```

O programa deve falhar com a mensagem a seguir.

A solicitação condicional falhou

Isso ocorre porque a classificação desse filme em especial é maior que 5.

3. Modifique o programa para remover a condição de params.

```
var params = {  
    TableName:table,  
    Key:{  
        "title":title,  
        "year":year  
    }  
};
```

4. Execute o programa novamente. Agora, a exclusão é bem-sucedida, pois você removeu a condição.

Etapa 4: Consulta e verificação de dados com o AWS SDK for JavaScriptNo DynamoDB

Você pode usar o método `query` para recuperar os dados de uma tabela. Você deve especificar um valor de chave de partição; a chave de classificação é opcional.

A chave primária da tabela `Movies` é composta pelo seguinte:

- `year`— A chave de partição. O tipo de atributo é número.
- `title`— A chave de classificação. O tipo de atributo é string.

Para encontrar todos os filmes lançados durante um ano, você precisa especificar somente o `year`. Você também pode fornecer o `title` para recuperar um subconjunto de filmes baseados na mesma condição (na chave de classificação). Por exemplo, é possível encontrar filmes lançados em 2014 que têm um título começando com a letra "A".

Além do método `query`, é possível usar o método `scan`, que pode recuperar todos os dados da tabela.

Para saber mais sobre como consultar e verificar dados, consulte [Como trabalhar com consultas no DynamoDB \(p. 490\)](#) e [Como trabalhar com verificações no DynamoDB \(p. 508\)](#), respectivamente.

Tópicos

- [Etapa 4.1: Consulta — Todos os filmes lançados em um ano \(p. 135\)](#)
- [Etapa 4.2: Consulta — Todos os filmes lançados em um ano com determinados títulos \(p. 136\)](#)
- [Etapa 4.3: Scan \(p. 138\)](#)

Etapa 4.1: Consulta — Todos os filmes lançados em um ano

O programa incluído nesta etapa recupera todos os filmes lançados no `year` 1985.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesQuery01.js`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */
```

```
/*
var AWS = require("aws-sdk");

AWS.config.update({
  region: "us-west-2",
  endpoint: "http://localhost:8000"
});

var docClient = new AWS.DynamoDB.DocumentClient();

console.log("Querying for movies from 1985.");

var params = {
  TableName : "Movies",
  KeyConditionExpression: "#yr = :yyyy",
  ExpressionAttributeNames:{
    "#yr": "year"
  },
  ExpressionAttributeValues: {
    ":yyyy": 1985
  }
};

docClient.query(params, function(err, data) {
  if (err) {
    console.error("Unable to query. Error:", JSON.stringify(err, null, 2));
  } else {
    console.log("Query succeeded.");
    data.Items.forEach(function(item) {
      console.log(" -", item.year + ": " + item.title);
    });
  }
});
});
```

Note

`ExpressionAttributeNames` fornece substituição de nome. Você usa isso porque `year` é uma palavra reservada no Amazon DynamoDB. Não é possível usá-la diretamente em nenhuma expressão, incluindo `KeyConditionExpression`. Você deve usar o nome de atributo de expressão `#yr` para resolver essa questão.

`ExpressionAttributeValues` fornece substituição de valor. Use isso porque não é possível usar literais em nenhuma expressão, incluindo `KeyConditionExpression`. Você deve usar o valor de atributo de expressão `:yyyy` para resolver essa questão.

2. Para executar o programa, digite o comando a seguir.

```
node MoviesQuery01.js
```

Note

O programa anterior mostra como consultar uma tabela por seus atributos de chave primária. No DynamoDB, você pode opcionalmente criar um ou mais índices secundários em uma tabela e consultá-los da mesma forma que consulta uma tabela. Os índices secundários oferecem aos seus aplicativos mais flexibilidade ao permitir consultas nos atributos que não são chave. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 4.2: Consulta — Todos os filmes lançados em um ano com determinados títulos

O programa incluído nesta etapa recupera todos os filmes lançados no `year` 1992, cujo `title` começa com a letra "A" até a letra "L".

1. Copie e cole o programa a seguir em um arquivo chamado MoviesQuery02.js.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
var AWS = require("aws-sdk");  
  
AWS.config.update({  
    region: "us-west-2",  
    endpoint: "http://localhost:8000"  
});  
  
var docClient = new AWS.DynamoDB.DocumentClient();  
  
console.log("Querying for movies from 1992 - titles A-L, with genres and lead actor");  
  
var params = {  
    TableName : "Movies",  
    ProjectionExpression:"#yr, title, info.genres, info.actors[0]",  
    KeyConditionExpression: "#yr = :yyyy and title between :letter1 and :letter2",  
    ExpressionAttributeNames:{  
        "#yr": "year"  
    },  
    ExpressionAttributeValues: {  
        ":yyyy": 1992,  
        ":letter1": "A",  
        ":letter2": "L"  
    }  
};  
  
docClient.query(params, function(err, data) {  
    if (err) {  
        console.log("Unable to query. Error:", JSON.stringify(err, null, 2));  
    } else {  
        console.log("Query succeeded.");  
        data.Items.forEach(function(item) {  
            console.log(" -", item.year + ": " + item.title  
            + " ... " + item.info.genres  
            + " ... " + item.info.actors[0]);  
        });  
    }  
});
```

2. Para executar o programa, digite o comando a seguir.

```
node MoviesQuery02.js
```

Etapa 4.3: Scan

O método `scan` lê todos os itens da tabela e retorna todos os dados nela contidos. Você pode fornecer uma `filter_expression` opcional, para que apenas os itens que correspondem aos seus critérios sejam retornados. No entanto, o filtro é aplicado somente depois que a tabela inteira foi verificada.

O programa a seguir verifica a tabela `Movies` inteira, que contém aproximadamente 5.000 itens. O exame especifica o filtro opcional para recuperar somente os filmes de 1950 (aproximadamente 100 itens) e descartar todos os outros.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesScan.js`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
var AWS = require("aws-sdk");  
  
AWS.config.update({  
    region: "us-west-2",  
    endpoint: "http://localhost:8000"  
});  
  
var docClient = new AWS.DynamoDB.DocumentClient();  
  
var params = {  
    TableName: "Movies",  
    ProjectionExpression: "#yr, title, info.rating",  
    FilterExpression: "#yr between :start_yr and :end_yr",  
    ExpressionAttributeNames: {  
        "#yr": "year",  
    },  
    ExpressionAttributeValues: {  
        ":start_yr": 1950,  
        ":end_yr": 1959  
    }  
};  
  
console.log("Scanning Movies table.");  
docClient.scan(params, onScan);  
  
function onScan(err, data) {  
    if (err) {  
        console.error("Unable to scan the table. Error JSON:", JSON.stringify(err,  
null, 2));  
    } else {  
        // print all the movies  
        console.log("Scan succeeded.");  
        data.Items.forEach(function(movie) {  
            console.log(  
                movie.year + ":",  
                movie.title, "- rating:", movie.info.rating);  
        });  
    }  
}
```

```
// continue scanning if we have more movies, because
// scan can retrieve a maximum of 1MB of data
if (typeof data.LastEvaluatedKey != "undefined") {
    console.log("Scanning for more...");  

    params.ExclusiveStartKey = data.LastEvaluatedKey;
    docClient.scan(params, onScan);
}
}
```

No código, observe o seguinte:

- `ProjectionExpression` especifica os atributos que você deseja no resultado da verificação.
 - `FilterExpression` especifica uma condição que retorna apenas os itens que satisfazem à condição. Todos os outros itens são descartados.
2. Para executar o programa, digite o comando a seguir.

```
node MoviesScan.js
```

Note

Também é possível usar a operação `Scan` com quaisquer índices secundários que tenha criado na tabela. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 5: (Optional): Excluir a tabela do usando oAWS SDK for JavaScript

Para excluir a tabela `Movies`:

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesDeleteTable.js`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
var AWS = require("aws-sdk");  
  
AWS.config.update({  
    region: "us-west-2",  
    endpoint: "http://localhost:8000"  
});  
  
var dynamodb = new AWS.DynamoDB();  
  
var params = {  
    TableName : "Movies"  
};
```

```
dynamodb.deleteTable(params, function(err, data) {
    if (err) {
        console.error("Unable to delete table. Error JSON:", JSON.stringify(err, null,
2));
    } else {
        console.log("Deleted table. Table description JSON:", JSON.stringify(data,
null, 2));
    }
});
```

2. Para executar o programa, digite o comando a seguir.

```
node MoviesDeleteTable.js
```

Summary

Neste tutorial, você criou oMoviesNo Amazon DynamoDB no seu computador e realizou operações básicas. A versão para download do DynamoDB é útil durante o desenvolvimento e o teste de aplicativos. No entanto, quando você estiver pronto para executar o aplicativo em um ambiente de produção, precisará modificar o código para que ele use o serviço web do DynamoDB.

Modificação do código para usar o serviço DynamoDB

Para usar o serviço DynamoDB, você deve alterar o endpoint em seu aplicativo. Para fazer isso, modifique o seguinte código.

```
AWS.config.update({endpoint: "https://dynamodb.aws-region.amazonaws.com"});
```

Por exemplo, se você quiser usar a região us-west-2, defina o endpoint a seguir.

```
AWS.config.update({endpoint: "https://dynamodb.us-west-2.amazonaws.com"});
```

Em vez de usar o DynamoDB no computador, o programa agora usa o endpoint de serviço web do DynamoDB na região oeste dos EUA (Oregon).

O DynamoDB está disponível emAWSRegiões do mundo inteiro. Para obter a lista completa, consulte[Regiões e endpoints](#) do[AWSReferência geral](#). Para obter mais informações sobre como definir regiões e endpoints no seu código, consulte [Definição da região](#) no Guia do desenvolvedor do AWS SDK for JavaScript.

Conceitos básicos do .NET e do DynamoDB

Neste tutorial, use oAWS SDK for .NETPara escrever programas simples a fim de realizar as seguintes operações do Amazon DynamoDB:

- Crie uma tabela chamada `Movies` usando um programa utilitário escrito em C# e carregue dados de exemplo no formato JSON.
- Realizar operações create, read, update e delete na tabela.
- Executar consultas simples.

O módulo DynamoDB do AWS SDK for .NET oferece vários modelos de programação para diferentes casos de uso. Neste exercício, o código C# usa o modelo de documento, que fornece um nível de

abstração que muitas vezes é conveniente. Ele também usa a API de baixo nível, que lida com atributos aninhados de maneira mais efetiva.

Para obter informações sobre a API do modelo de documento, consulte [.NET: Modelo de documento \(p. 276\)](#). Para obter informações sobre a API de baixo nível, consulte [Como trabalhar com tabelas do DynamoDB no .NET \(p. 397\)](#).

Tópicos

- [Pré-requisitos do Tutorial .NET e DynamoDB \(p. 141\)](#)
- [Etapa 1: Criação de um cliente DynamoDB \(p. 142\)](#)
- [Etapa 2: Criar uma tabela do DynamoDB usando a API de baixo nível \(p. 144\)](#)
- [Etapa 3: Carregar dados de amostra na tabela do DynamoDB \(p. 146\)](#)
- [Etapa 4: Adicionar um filme à tabela do DynamoDB \(p. 148\)](#)
- [Etapa 5: Ler e exibir um registro da tabela do DynamoDB \(p. 149\)](#)
- [Etapa 6: Atualizar o novo registro de filme na tabela do DynamoDB \(p. 150\)](#)
- [Etapa 7: Excluir condicionalmente \(p. 152\)](#)
- [Etapa 8: Consulte uma tabela do DynamoDB com .NET \(p. 154\)](#)
- [Etapa 9: Verificar a tabela de filmes com .NET \(p. 156\)](#)
- [Etapa 10: Exclusão da tabela de filmes com .NET \(p. 157\)](#)

Pré-requisitos do Tutorial .NET e DynamoDB

O [Tutorial do Microsoft .NET e DynamoDB \(p. 140\)](#) Descreve como usar o AWS SDK for .NET para criar programas simples para executar operações do Amazon DynamoDB.

Antes de começar, siga estas etapas para garantir que você tenha todos os pré-requisitos necessários para concluir o tutorial:

- Use um computador que estejam executando uma versão recente do Microsoft Windows e uma versão atual do Microsoft Visual Studio. Se você não tiver o Visual Studio instalado, poderá fazer download de uma cópia gratuita da Community Edition no [site do Microsoft Visual Studio](#).
- Faça o download e execute o DynamoDB (versão para download). Para mais informações, consulte [Configuração do DynamoDB Local \(versão disponível para download\) \(p. 50\)](#).

Note

Você pode usar as versões disponíveis para download do DynamoDB neste tutorial. Para obter mais informações sobre como executar o mesmo código para o serviço web do DynamoDB, consulte [Etapa 1: Criação de um cliente DynamoDB \(p. 142\)](#).

- Configure uma AWS Chave de acesso para usar o AWS SDKs. Para mais informações, consulte [Configuração do DynamoDB \(serviço Web\) \(p. 57\)](#).
- Configure um perfil de segurança para o DynamoDB no Visual Studio. Para obter instruções detalhadas, consulte [Exemplos de código .NET \(p. 336\)](#).
- Abra a solução de demonstração de conceitos básicos que é usada neste tutorial no Visual Studio:
 1. Faça o download de um arquivo.zip contendo a solução do [DynamoDB_intro.zip](#).
 2. Salve o arquivo em um local conveniente no seu computador e extraia (descompacte) os arquivos nele contidos.
 3. No Visual Studio, pressione Ctrl+Shift+O, ou escolha **Arquivo > Arquivo e escolha Projeto/Solução**.
 4. Navegue até `DynamoDB_intro.sln` no `DynamoDB_intro` que você descompactou e escolha **Aberto**.

- Build o `DynamoDB_intro.cs`, em seguida, abra a caixa de diálogo `00_Main.cs` no Visual Studio.
 1. No Build, selecione Build uma solução (ou pressione `Ctrl+Shift+B`). A solução deve ser compilada com sucesso.
 2. Verifique se o Explorer de soluções está sendo exibido e fixado no Visual Studio. Se ainda não foi, você pode encontrá-lo no Exibir, ou pressionando `Ctrl+Alt+L`.
 3. Dentro do Explorer de soluções, abra o `00_Main.cs`. Este é o arquivo que controla a execução do programa de demonstração que é usado neste tutorial.

Note

Este tutorial mostra como usar métodos assíncronos em vez de métodos síncronos. Isso ocorre, pois o núcleo do .NET oferece suporte somente a métodos assíncronos, e também ao modelo assíncrono geralmente é preferível quando o desempenho é fundamental. Para obter mais informações, consulte [AWS APIs assíncronas da para .NET](#).

Instalando as dependências externas da solução `DynamoDB_Intro`

O `DynamoDB_intro` já tem o Amazon DynamoDB SDK instalado nele. Ele também tem o código aberto `Newtonsoft.Json` para deserializar dados JSON, licenciados sob a licença MIT (MIT) (MIT) (consulte <https://github.com/JamesNK/Newtonsoft.Json/blob/master/LICENSE.md>).

Para instalar o pacote NuGet para o módulo DynamoDB do AWS SDK for .NET versão 3 em seus próprios programas, abra o Console do Gerenciador de Pacotes NuGet no Ferramentas no Visual Studio. Depois, execute o seguinte comando no `PM> editor.exe?`:

```
Install-Package AWSSDK.DynamoDBv2
```

De forma semelhante, você pode usar o Console do Gerenciador de Pacotes NuGet para carregar o `Newtonsoft.Json` para seus próprios projetos no Visual Studio. No `PM>`, insira o seguinte comando:

```
Install-Package Newtonsoft.Json
```

Próxima etapa

[Etapa 1: Criação de um cliente DynamoDB \(p. 142\)](#)

Etapa 1: Criação de um cliente DynamoDB

O primeiro passo no [Tutorial do Microsoft .NET e DynamoDB \(p. 140\)](#) é criar um cliente que conceda acesso à API do Amazon DynamoDB. A função `Main` no `DynamoDB_intro` faz isso chamando uma função `createClient` implementada no arquivo `01_CreateClient.cs`.

```
using System;
using System.Net;
using System.Net.NetworkInformation;
using Amazon.DynamoDBv2;

namespace DynamoDB_intro
{
    public static partial class DdbIntro
    {
```

```
/*
-----
 * If you are creating a client for the Amazon DynamoDB service, make sure your
credentials
 * are set up first, as explained in:
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
SettingUp.DynamoWebService.html,
 *
 * If you are creating a client for DynamoDBLocal (for testing purposes),
 * DynamoDB-Local should be started first. For most simple testing, you can keep
 * data in memory only, without writing anything to disk. To do this, use the
 * following command line:
 *
 *     java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -
inMemory
 *
 * For information about DynamoDBLocal, see:
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
DynamoDBLocal.html.
-----
// So we know whether local DynamoDB is running
private static readonly string Ip = "localhost";
private static readonly int Port = 8000;
private static readonly string EndpointUrl = "http://" + Ip + ":" + Port;
private static bool IsPortInUse()
{
    bool isAvailable = true;
    // Evaluate current system TCP connections. This is the same information
provided
    // by the netstat command line application, just in .Net strongly-typed object
    // form. We will look through the list, and if our port we would like to use
    // in our TcpClient is occupied, we will set isAvailable to false.
    IPGlobalProperties ipGlobalProperties =
IPGlobalProperties.GetIPGlobalProperties();
    IPEndPoint[] tcpConnInfoArray = ipGlobalProperties.GetActiveTcpListeners();
    foreach (IPEndPoint endpoint in tcpConnInfoArray)
    {
        if (endpoint.Port == Port)
        {
            isAvailable = false;
            break;
        }
    }

    return isAvailable;
}

public static bool createClient(bool useDynamoDbLocal)
{
    if (useDynamoDbLocal)
    {
        // First, check to see whether anyone is listening on the DynamoDB local
port
        // (by default, this is port 8000, so if you are using a different port,
modify this accordingly)
        var portUsed = IsPortInUse();
        if (portUsed)
        {
            Console.WriteLine("The local version of DynamoDB is NOT running.");
            return (false);
        }

        // DynamoDB-Local is running, so create a client
        Console.WriteLine(" -- Setting up a DynamoDB-Local client (DynamoDB Local
seems to be running)");
    }
}
```

```
AmazonDynamoDBConfig ddbConfig = new AmazonDynamoDBConfig();
ddbConfig.ServiceURL = EndpointUrl;
try
{
    Client = new AmazonDynamoDBClient(ddbConfig);
}
catch (Exception ex)
{
    Console.WriteLine("      FAILED to create a DynamoDBLocal client; " +
ex.Message);
    return false;
}
else
{
    Client = new AmazonDynamoDBClient();
}

return true;
}
```

`Main` chama essa função com o parâmetro `useDynamoDBLocal` definido como `true`. Portanto, a versão de teste local do DynamoDB já deve estar sendo executada em seu computador usando a porta padrão (8000), ou a chamada falhará. Se ainda não o tiver instalado, consulte [Execução do DynamoDB no computador \(p. 50\)](#).

Configurar `ouseDynamoDBLocalParâmetro para false` cria um cliente para o próprio serviço do DynamoDB do, em vez do programa de teste local.

Próxima etapa

[Etapa 2: Criar uma tabela do DynamoDB usando a API de baixo nível \(p. 144\)](#)

Etapa 2: Criar uma tabela do DynamoDB usando a API de baixo nível

O modelo de documento no AWS SDK for .NET não favorece a criação de tabelas; portanto, você precisa usar as APIs de nível inferior. Para mais informações, consulte [Como trabalhar com tabelas do DynamoDB no .NET \(p. 397\)](#).

Nesta etapa do [Tutorial do Microsoft .NET e DynamoDB \(p. 140\)](#), você cria uma tabela chamada `Movies` no Amazon DynamoDB. A chave primária da tabela é composta dos seguintes atributos:

- `year`— A chave de partição. O `AttributeType` é `N` para número.

Note

Como “`year`” (ano) é uma palavra reservada no DynamoDB, você precisa criar um alias para ela (`como#yr`) usando `umaExpressionAttributeName` quando se refere a ele em uma expressão de baixo nível.

- `title`— A chave de classificação. O `AttributeType` é `S` para string.

A função `Main` no `DynamoDB_intro` faz isso aguardando uma função `CreatingTable_async` assíncrona implementada no arquivo `02_CreatingTable.cs`:

```
using System;
```

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.DynamoDBv2.Model;

namespace DynamoDB_intro
{
    public static partial class DdbIntro
    {
        public static async Task<bool> CheckingTableExistence_async(string tblNm)
        {
            var response = await DdbIntro.Client.ListTablesAsync();
            return response.TableNames.Contains(tblNm);
        }

        public static async Task<bool> CreateTable_async(string tableName,
            List<AttributeDefinition> tableAttributes,
            List<KeySchemaElement> tableKeySchema,
            ProvisionedThroughput provisionedThroughput)
        {
            bool response = true;

            // Build the 'CreateTableRequest' structure for the new table
            var request = new CreateTableRequest
            {
                TableName = tableName,
                AttributeDefinitions = tableAttributes,
                KeySchema = tableKeySchema,
                // Provisioned-throughput settings are always required,
                // although the local test version of DynamoDB ignores them.
                ProvisionedThroughput = provisionedThroughput
            };

            try
            {
                var makeTbl = await DdbIntro.Client.CreateTableAsync(request);
            }
            catch (Exception)
            {
                response = false;
            }

            return response;
        }

        public static async Task<TableDescription> GetTableDescription(string tableName)
        {
            TableDescription result = null;

            // If the table exists, get its description.
            try
            {
                var response = await DdbIntro.Client.DescribeTableAsync(tableName);
                result = response.Table;
            }
            catch (Exception)
            {}

            return result;
        }
    }
}
```

A função `CreateTable_async` começa aguardando uma função `CheckingTableExistence_async` assíncrona para determinar se uma tabela chamada "Movies" (Filmes) já existe. Se a tabela existir, `CheckingTableExistence_async` retorna o `TableDescription` para a `Table` existente.

Se a tabela ainda não existir, `CreatingTable_async` espera no `CreateNewTable_async` para criar o `Movies` usando a API do cliente do DynamoDB `CreateTableAsync`.

O exemplo do `DynamoDB_intro` usa métodos assíncronos em vez de métodos síncronos sempre que possível. Isso ocorre, pois o núcleo do .NET oferece suporte somente a métodos assíncronos, e o modelo assíncrono geralmente é preferível quando o desempenho é fundamental. Para obter mais informações, consulte [AWS APIs assíncronas da para .NET](#).

Para saber mais sobre como gerenciar tabelas, consulte [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#).

Próxima etapa

[Etapa 3: Carregar dados de amostra na tabela do DynamoDB \(p. 146\)](#)

Etapa 3: Carregar dados de amostra na tabela do DynamoDB

Nesta etapa do [Tutorial do Microsoft .NET e DynamoDB \(p. 140\)](#), você preenche o novo `Movies` no Amazon DynamoDB com dados de amostra do Internet Movie Database (IMDb). Esses dados são armazenados no formato JSON em um arquivo de texto local chamado `moviedata.json`.

Para cada filme, o `moviedata.json` define um par de nome/valor `year`, um par de nome/valor `title` e um objeto `info` complexo, conforme ilustrado pelo exemplo a seguir.

```
{  
    "year" : 2013,  
    "title" : "Turn It Down, Or Else!",  
    "info" : {  
        "directors" : [  
            "Alice Smith",  
            "Bob Jones"  
        ],  
        "release_date" : "2013-01-18T00:00:00Z",  
        "rating" : 6.2,  
        "genres" : [  
            "Comedy",  
            "Drama"  
        ],  
        "image_url" : "http://ia.media-imdb.com/images/N/  
09ERWAU7FS797AJ7LU8HN09AMUP908RLlo5JF90EWR7LJKQ7@._V1_SX400_.jpg",  
        "plot" : "A rock band plays their music at high volumes, annoying the neighbors.",  
        "rank" : 11,  
        "running_time_secs" : 5215,  
        "actors" : [  
            "David Matthewman",  
            "Ann Thomas",  
            "Jonathan G. Neff"  
        ]  
    }  
}
```

Antes de carregar o arquivo `moviedata.json`, a função `Main` no `DynamoDB_intro` faz uma verificação para determinar se a tabela `Movies` existe e se ainda está vazia. Se for o caso, ela aguardará uma função `LoadingData_async` assíncrona que é implementada no arquivo `03_LoadingData.cs`.

```
using System;  
using System.IO;
```

```
using System.Threading.Tasks;
using Amazon.DynamoDBv2.DocumentModel;

using Newtonsoft.Json;
using Newtonsoft.Json.Linq;

namespace DynamoDB_intro
{
    public static partial class DdbIntro
    {
        public static async Task<bool> LoadingData_async(Table table, string filePath)
        {
            var movieArray = await ReadJsonMovieFile_async(filePath);

            if (movieArray != null)
                await LoadJsonMovieData_async(table, movieArray);

            return true;
        }

        public static async Task<JArray> ReadJsonMovieFile_async(string jsonMovieFilePath)
        {
            StreamReader sr = null;
            JsonTextReader jtr = null;
            JArray movieArray = null;

            Console.WriteLine(" -- Reading the movies data from a JSON file...");

            try
            {
                sr = new StreamReader(jsonMovieFilePath);
                jtr = new JsonTextReader(sr);
                movieArray = (JArray)await JToken.ReadFromAsync(jtr);
            }
            catch (Exception ex)
            {
                Console.WriteLine("      ERROR: could not read the file!\n      Reason: " + ex.Message);
            }
            finally
            {
                jtr?.Close();
                sr?.Close();
            }

            return movieArray;
        }

        public static async Task<bool> LoadJsonMovieData_async(Table moviesTable, JArray moviesArray)
        {
            int n = moviesArray.Count;
            Console.Write("      -- Starting to load {0:#,##0} movie records into the Movies table asynchronously...\n" + "      Wrote: ", n);
            for (int i = 0, j = 99; i < n; i++)
            {
                try
                {
                    string itemJson = moviesArray[i].ToString();
                    Document doc = Document.FromJson(itemJson);
                    Task putItem = moviesTable.PutItemAsync(doc);
                    if (i >= j)
                    {
                        j++;
                        Console.Write("{0,5:#,##0}, ", j);
                    }
                }
            }
        }
    }
}
```

```
        if (j % 1000 == 0)
            Console.WriteLine("\n");
        j += 99;
    }
    await putItem;
}
catch (Exception)
{
    return false;
}

return true;
}
}
```

`LoadingData_async` começa aguardando `ReadJsonMovieFile_async`. Essa função lê o arquivo `moviedata.json` usando a [biblioteca do Json.NET Newtonsoft](#) de código-fonte aberto, que é licenciada sob a [Licença MIT](#).

Quando os dados foram lidos com êxito, o `LoadingData_async` espera `noLoadJsonMovieData_async` para carregar os registros de filme no `Movies` tabela usando o modelo de documento do `DynamoDBTable.PutItemAsyncAPI`. Para obter informações sobre a API do modelo de documento, consulte [.NET: Modelo de documento \(p. 276\)](#).

Próxima etapa

[Etapa 4: Adicionar um filme à tabela do DynamoDB \(p. 148\)](#)

Etapa 4: Adicionar um filme à tabela do DynamoDB

Nesta etapa do [Tutorial do Microsoft .NET e DynamoDB \(p. 140\)](#), você adiciona um novo registro de filme ao `Movies` no Amazon DynamoDB. O main função `noDynamoDB_intro` começa criando um modelo de documento do `DynamoDBDocument`, em seguida, aguarda em `WritingNewMovie_async`, que é implementado no `04_WritingNewItem.cs`.

```
using System;
using System.Threading.Tasks;
using Amazon.DynamoDBv2.DocumentModel;

namespace DynamoDB_intro
{
    public static partial class DdbIntro
    {
        public static async Task<bool> CheckingForMovie_async(Document newItem)
        {
            int year = (int)newItem["year"];
            string name = newItem["title"];

            var response = await ReadingMovie_async(year, name);

            return response.Count > 0;
        }

        public static async Task<bool> WritingNewMovie_async(Document newItem)
        {
            var result = false;

            try
            {
```

```
var writeNew = await MoviesTable.PutItemAsync(newItem);
Console.WriteLine(" -- Writing a new movie to the Movies table...");

Console.WriteLine(" -- Wrote the item successfully!");
result = true;
}
catch (Exception ex)
{
    Console.WriteLine(" FAILED to write the new movie, because:\n{0}.".format(ex.Message));
}

return result;
}
}
```

`WritingNewMovie_async` começa verificando se o novo filme já foi adicionado à tabela `Movies`. Se ainda não foi, ele aguarda o `DynamoDBTable.PutItemAsyn` para adicionar o registro do novo filme.

Para obter mais informações

- Para saber mais sobre leitura e gravação de dados nas tabelas do DynamoDB, consulte[Trabalho com itens e atributos \(p. 404\)](#).
- Para obter mais informações sobre a API do modelo de documento do DynamoDB, consulte[.NET: Modelo de documento \(p. 276\)](#).
- Para obter mais informações sobre os métodos assíncronos, consulte[AWSAPIs assíncronas da para .NET](#).

Próxima etapa

[Etapa 5: Ler e exibir um registro da tabela do DynamoDB \(p. 149\)](#)

Etapa 5: Ler e exibir um registro da tabela do DynamoDB

Nesta etapa do[Tutorial do Microsoft .NET e DynamoDB \(p. 140\)](#), você recuperará e exibirá o novo registro de filme que você adicionou no[Etapa 4 \(p. 148\)](#). A função `Main` no `DynamoDB_intro` faz isso aguardando uma função `ReadingMovie_async`, que é implementada no arquivo `05_ReadingItem.cs`.

```
using System;
using System.Threading.Tasks;
using Amazon.DynamoDBv2.DocumentModel;

namespace DynamoDB_intro
{
    public static partial class DdbIntro
    {
        public static async Task<Document> ReadingMovie_async(int year, string title)
        {
            // Create Primitives for the HASH and RANGE portions of the primary key
            Primitive hash = new Primitive(year.ToString(), true);
            Primitive range = new Primitive(title, false);

            try
            {
                var movieItem = await MoviesTable.GetItemAsync(hash, range, Token);
            }
        }
    }
}
```

```
        return movieItem;
    }
    catch (Exception)
    {
        return null;
    }
}
```

ReadingMovie_async por sua vez aguarda no DynamoDBTable.GetItemAsync para recuperar o novo registro de filme como um Documento. ReadingMovie_async, em seguida, exibe o filme como texto JSON usando o DocumentoToJsonPretty Método do.

Para obter mais informações

- Para saber mais sobre leitura e gravação de dados nas tabelas do DynamoDB, consulte [Trabalho com itens e atributos \(p. 404\)](#).
- Para obter mais informações sobre a API do modelo de documento do DynamoDB, consulte [.NET: Modelo de documento \(p. 276\)](#).
- Para obter mais informações sobre os métodos assíncronos, consulte [AWS APIs assíncronas da para .NET](#).

Próxima etapa

[Etapa 6: Atualizar o novo registro de filme na tabela do DynamoDB \(p. 150\)](#)

Etapa 6: Atualizar o novo registro de filme na tabela do DynamoDB

Nesta etapa do [Tutorial do Microsoft .NET e DynamoDB \(p. 140\)](#), atualize o registro do novo filme de várias maneiras diferentes.

Tópicos

- [Alterar o enredo, a classificação e adicionar atores \(p. 150\)](#)
- [Incrementar a avaliação do filme atomicamente \(p. 152\)](#)
- [Tentar atualizar usando uma condição que apresenta falha \(p. 152\)](#)
- [Para obter mais informações \(p. 152\)](#)
- [Próxima etapa \(p. 152\)](#)

Alterar o enredo, a classificação e adicionar atores

A função Main em DynamoDB_intro altera o enredo e a classificação do registro de filme adicionado na [Etapa 4 \(p. 148\)](#), e também adiciona uma lista de atores a ele. O modelo de documento no AWS SDK for .NET não oferece suporte à atualização de atributos aninhados, como os itens no atributo info. Por causa disso, a função Main usa a API do cliente de baixo nível em vez do método Table de um modelo de documento.

Ela começa criando um UpdateItemRequest de baixo nível para fazer essa alteração.

```
UpdateItemRequest updateRequest = new UpdateItemRequest()
```

```
{  
    TableName = movies_table_name,  
    Key = new Dictionary<string, AttributeValue>  
    {  
        { partition_key_name, new AttributeValue { N = "2018" } },  
        { sort_key_name, new AttributeValue { S = "The Big New Movie" } }  
    },  
    ExpressionAttributeValues = new Dictionary<string, AttributeValue>  
    {  
        { ":r", new AttributeValue { N = "5.5" } },  
        { ":p", new AttributeValue { S = "Everything happens all at once!" } },  
        { ":a", new AttributeValue { L = new List<AttributeValue>  
            { new AttributeValue { S = "Larry" },  
              new AttributeValue { S = "Moe" },  
              new AttributeValue { S = "Curly" } } }  
    }  
},  
    UpdateExpression = "SET info.rating = :r, info.plot = :p, info.actors = :a",  
    ReturnValues = "NONE"  
};
```

Definir `ReturnValues` como `NONE` especifica que nenhuma informação de atualização deve ser retornada. No entanto, quando `Main` aguarda `UpdatingMovie_async`, ela define o parâmetro relatório como `true`. Isso faz com que `UpdatingMovie_async` altere `ReturnValues` para `ALL_NEW`, indicando que o item atualizado deve ser retornado na íntegra.

`UpdatingMovie_async` é implementada no arquivo `06_UpdatingItem.cs`.

```
using System;  
using System.Threading.Tasks;  
using Amazon.DynamoDBv2.Model;  
  
namespace DynamoDB_intro  
{  
    public static partial class DdbIntro  
    {  
        public static async Task<bool> UpdatingMovie_async(UpdateItemRequest updateRequest)  
        {  
            var result = false;  
  
            try  
            {  
                await Client.UpdateItemAsync(updateRequest);  
                result = true;  
            }  
            catch (Exception)  
            {  
                result = false;  
            }  
  
            return result;  
        }  
    }  
}
```

`UpdatingMovie_async` guarda o DynamoDB de baixo nível `Client.UpdateItemAsync` para atualizar o registro do filme. Se a atualização for bem-sucedida, e se o parâmetro relatório for `true`, `UpdatingMovie_async` exibirá o registro do filme atualizado.

Quando o modelo de documento tiver um método útil `Document.ToJsonPretty()` para exibir conteúdos do documento, será um pouco mais complicado trabalhar com valores de atributo de baixo

nível. O arquivo `00b_DDB_Attributes.cs` pode fornecer alguns exemplos de como acessar e trabalhar com objetos `AttributeValue`.

Incrementar a avaliação do filme atomicamente

O DynamoDB oferece suporte à atualização atômica de contadores, em que você usa um método de atualização de baixo nível para aumentar ou diminuir o valor de um atributo existente sem interferência de outras solicitações de gravação. (Todas as solicitações de gravação no DynamoDB são aplicadas na ordem em que são recebidas.)

Para aumentar o valor de avaliação no filme que você acabou de criar, a função `Main` faz as seguintes alterações em `UpdateItemRequest` que usou na atualização anterior.

```
updateRequest.ExpressionAttributeValues = new Dictionary<string, AttributeValue>
{
    { ":inc", new AttributeValue { N = "1" } }
};
updateRequest.UpdateExpression = "SET info.rating = info.rating + :inc";
```

Então, novamente, aguarda `UpdatingMovie_async` fazer a alteração.

Tentar atualizar usando uma condição que apresenta falha

Também é possível adicionar uma condição a uma solicitação de atualização. Assim, se a condição não for atendida, a atualização não ocorrerá.

Para demonstrar isso, a função `Main` faz as seguintes alterações em `UpdateItemRequest` que acabou de usar para aumentar a avaliação do filme.

```
updateRequest.ExpressionAttributeValues.Add( ":n", new AttributeValue { N = "3" } );
updateRequest.ConditionExpression = "size(info.actors) > :n";
```

Agora a atualização só poderá ocorrer se houver mais de três atores no registro do filme que estiver sendo atualizado. Como há somente três atores listados, a condição apresentará falha quando `Main` aguardar `UpdatingMovie_async`, e a atualização não ocorrerá.

Para obter mais informações

- Para saber mais sobre leitura e gravação de dados nas tabelas do DynamoDB, consulte [Trabalho com itens e atributos \(p. 404\)](#).
- Para obter mais informações sobre a API do modelo de documento do DynamoDB, consulte [.NET: Modelo de documento \(p. 276\)](#).
- Para obter mais informações sobre os métodos assíncronos, consulte [AWSAPIs assíncronas da para .NET](#).

Próxima etapa

[Etapa 7: Excluir condicionalmente \(p. 152\)](#)

Etapa 7: Excluir condicionalmente

Nesta etapa do [Tutorial do Microsoft .NET e DynamoDB \(p. 140\)](#), você tentará excluir um registro de filme com uma condição que não é atendida e a exclusão falhará. Então, quando a condição for alterada para que seja atendida, a exclusão será bem-sucedida.

A função Main em DynamoDB_intro começa criando uma condição conforme mostrado a seguir.

```
Expression condition = new Expression();
condition.ExpressionAttributeValues[":val"] = 5.0;
condition.ExpressionStatement = "info.rating <= :val";
```

A função Main transmite Expression como um dos parâmetros de DeletingItem_async e aguarda. DeletingItem_async é implementada no arquivo 07_DeletingItem.cs.

```
using System;
using System.Threading.Tasks;
using Amazon.DynamoDBv2.DocumentModel;

namespace DynamoDB_intro
{
    public static partial class DdbIntro
    {
        public static async Task<bool> DeletingItem_async( Table table, int year, string title,
                                                       Expression condition=null )
        {
            Document deletedItem;
            // Create Primitives for the HASH and RANGE portions of the primary key
            Primitive hash = new Primitive(year.ToString(), true);
            Primitive range = new Primitive(title, false);
            DeleteItemOperationConfig deleteConfig = new DeleteItemOperationConfig( );
            deleteConfig.ConditionalExpression = condition;
            deleteConfig.ReturnValues = ReturnValues.AllOldAttributes;

            Console.WriteLine( " -- Trying to delete the {0} movie \"{1}\"...", year, title );
            try
            {
                var delItem = table.DeleteItemAsync( hash, range, deleteConfig );
                deletedItem = await delItem;
            }
            catch( Exception ex )
            {
                Console.WriteLine( "      FAILED to delete the movie item, for this reason:\n{0}\n", ex.Message );
                return false;
            }

            Console.WriteLine( "      -- SUCCEEDED in deleting the movie record that looks like
this:\n" +
                               deletedItem.ToJsonPretty( ) );
            return true;
        }
    }
}
```

DeletingItem_asyncPor sua vez, inclui a condiçãoExpressionem umDeleteItemOperationConfigque ele passa para o DynamoDBTable.DeleteItemAsyncquando ele aguarda nele.

Como a avaliação do filme é 6,5, que é maior que 5,0, a condição não é atendida e a exclusão falha.

Em seguida, quando a função Main altera o limite de avaliação na condição para 7,0 em vez de 5,0, a exclusão é bem-sucedida.

Próxima etapa

[Etapa 8: Consulte uma tabela do DynamoDB com .NET \(p. 154\)](#)

Etapa 8: Consulte uma tabela do DynamoDB com .NET

Nesta etapa do [Tutorial do Microsoft .NET e DynamoDB](#) (p. 140), você consulta o `MovieTable` de três maneiras diferentes.

Tópicos

- [Usar uma pesquisa do modelo de documento simples para consultar filmes de 1985](#) (p. 154)
- [Usar `QueryOperationConfig` para criar uma pesquisa de consulta mais complexa](#) (p. 155)
- [Use uma consulta de baixo nível para encontrar filmes de 1992 com títulos entre "M..." e "Tzz..."](#) (p. 155)
- [Próxima etapa](#) (p. 156)

Usar uma pesquisa do modelo de documento simples para consultar filmes de 1985

Para configurar uma consulta de modelo de documento simples, a função `Main` em `DynamoDB_intro` cria um objeto `Search` chamando a API `Table.Query` com 1985 como a chave de partição (também conhecida como chave de hash) e um filtro vazio `Expression`.

```
try { search = moviesTable.Query( 1985, new Expression( ) ); }
```

Elá aguarda `SearchListing_async`, que é implementada em `08_Querying.cs` para recuperar e exibir os resultados da consulta.

```
using System;
using System.Threading.Tasks;
using Amazon.DynamoDBv2.Model;
using Amazon.DynamoDBv2.DocumentModel;
using System.Collections.Generic;

namespace DynamoDB_intro
{
    public static partial class DdbIntro
    {
        public static async Task<List<List<Document>>> SearchListing_async(Search search)
        {
            List<List<Document>> docsList = new List<List<Document>>();

            do
            {
                try
                {
                    var getNextBatch = search.GetNextSetAsync();
                    var docList = await getNextBatch;
                    docsList.Add(docList);
                }
                catch (Exception)
                {
                    return null;
                }
            } while (!search.IsDone);

            return docsList;
        }
    }
}
```

```
public static async Task<QueryResponse> ClientQuerying_async(QueryRequest qRequest)
{
    var response = await Client.QueryAsync(qRequest);
    return response;
}
```

Usar QueryOperationConfig para criar uma pesquisa de consulta mais complexa

Para consultar filmes de 1992 com títulos de "B..." a "Hzz...", a função Main cria um objeto QueryOperationConfig com um QueryFilter e vários outros campos.

```
QueryOperationConfig config = new QueryOperationConfig();
config.Filter = new QueryFilter();
config.Filter.AddCondition( "year", QueryOperator.Equal, new DynamoDBEntry[ ]
{ 1992 } );
config.Filter.AddCondition( "title", QueryOperator.Between, new DynamoDBEntry[ ]
{ "B", "Hzz" } );
config.AttributesToGet = new List<string> { "year", "title", "info" };
config.Select = SelectValues.SpecificAttributes;
```

Novamente, será criado um objeto Search chamando a API Table.Query, dessa vez com o objeto QueryOperationConfig como o único parâmetro.

E, novamente, ela aguarda SearchListing_async para recuperar e exibir os resultados da consulta.

Use uma consulta de baixo nível para encontrar filmes de 1992 com títulos entre "M..." e "Tzz..."

Para usar uma consulta de baixo nível para recuperar filmes de 1992 com títulos de "M..." a "Tzz...", a função Main cria um objeto QueryRequest.

```
QueryRequest qRequest= new QueryRequest
{
    TableName = "Movies",
    ExpressionAttributeNames = new Dictionary<string, string>
    {
        { "#yr", "year" }
    },
    ExpressionAttributeValues = new Dictionary<string, AttributeValue>
    {
        { ":qYr", new AttributeValue { N = "1992" } },
        { ":tSt", new AttributeValue { S = "M" } },
        { ":tEn", new AttributeValue { S = "Tzz" } }
    },
    KeyConditionExpression = "#yr = :qYr and title between :tSt and :tEn",
    ProjectionExpression = "#yr, title, info.actors[0], info.genres,
    info.running_time_secs"
};
```

Em seguida, ele aguarda noClientQuerying_async implementada na função 08_Querying.csfile.ClientQuerying_async que guarda o método do DynamoDB de baixo nível AmazonDynamoDBClient.QueryAsync para recuperar os resultados da consulta.

Note

Como "year" (ano) é uma palavra reservada no DynamoDB, você precisa criar um alias para ela (aqui #yr) usando ExpressionAttributeNames para usá-lo em uma expressão de baixo nível.

Próxima etapa

[Etapa 9: Verificar a tabela de filmes com .NET \(p. 156\)](#)

Etapa 9: Verificar a tabela de filmes com .NET

Nesta etapa do [Tutorial do Microsoft .NET e DynamoDB \(p. 140\)](#), você varre o `Movies` de duas maneiras diferentes: usando uma verificação do modelo de documento do e uma verificação de baixo nível.

Tópicos

- [Usar uma pesquisa do modelo de documento para procurar filmes de 1950 \(p. 156\)](#)
- [Usar uma verificação de baixo nível para recuperar filmes da década de 1960 \(p. 156\)](#)
- [Próxima etapa \(p. 157\)](#)

Usar uma pesquisa do modelo de documento para procurar filmes de 1950

Para configurar uma verificação do modelo de documento para filmes de 1950, a função `Main` em `DynamoDB_intro` cria um objeto `ScanOperationConfig` com um `ScanFilter`:

```
ScanFilter filter = new ScanFilter();
filter.AddCondition("year", ScanOperator.Between, new DynamoDBEntry[] { 1950,
1959 });
ScanOperationConfig scanConfig = new ScanOperationConfig
{
    AttributesToGet = new List<string> { "year, title, info" },
    Filter = filter
};
```

Para obter um objeto `Search` para a verificação, o objeto `ScanOperationConfig` é transmitido para `Table.Scan`. Ao usar o objeto `Search`, ela, então, aguarda `SearchListing_async` (implementado em `08_Querying.cs`) para recuperar e exibir os resultados da verificação.

Usar uma verificação de baixo nível para recuperar filmes da década de 1960

Para configurar uma verificação de baixo nível para filmes da década de 1960, a função `Main` em `DynamoDB_intro` cria um objeto `ScanRequest` com vários campos.

```
ScanRequest sRequest = new ScanRequest
{
    TableName = "Movies",
    ExpressionAttributeNames = new Dictionary<string, string>
    {
        { "#yr", "year" }
    },
    ExpressionAttributeValues = new Dictionary<string, AttributeValue>
    {
        { ":y_a", new AttributeValue { N = "1960" } },
        { ":y_z", new AttributeValue { N = "1969" } },
    },
    FilterExpression = "#yr between :y_a and :y_z",
    ProjectionExpression = "#yr, title, info.actors[0], info.directors,
    info.running_time_secs"
```

```
};
```

Em seguida, ele aguarda no `clientScanning_async` implementada na função `Scanning.cs`. O `ClientScanning_async` guarda o método do DynamoDB de baixo nível `AmazonDynamoDBClient.ScanAsync` para recuperar os resultados da consulta.

Note

Como “year” (ano) é uma palavra reservada no DynamoDB, é necessário criar um alias para ela (`#yr`) usando `ExpressionAttributeNames` para usá-lo em uma expressão de baixo nível.

Próxima etapa

[Etapa 10: Exclusão da tabela de filmes com .NET \(p. 157\)](#)

Etapa 10: Exclusão da tabela de filmes com .NET

Nesta etapa do [Tutorial do Microsoft .NET e DynamoDB \(p. 140\)](#), você exclui o `Movies` no Amazon DynamoDB.

A função `Main` no `DynamoDB_intro` aguarda uma função `DeletingTable_async`, que é implementada no arquivo `10_DeletingTable.cs`.

```
using System;
using System.Threading.Tasks;

namespace DynamoDB_intro
{
    public static partial class DdbIntro
    {
        public static async Task<bool> DeletingTable_async(string tableName)
        {
            var tblDelete = await Client.DeleteTableAsync(tableName);
            return true;
        }
    }
}
```

`DeletingTable_async` guarda o método do DynamoDB de baixo nível `AmazonDynamoDBClient.DeleteTableAsync` para excluir a tabela.

Para obter mais informações

- Para saber mais sobre leitura e gravação de dados nas tabelas do DynamoDB, consulte [Trabalho com itens e atributos \(p. 404\)](#).
- Para obter mais informações sobre a API do modelo de documento do DynamoDB, consulte [.NET: Modelo de documento \(p. 276\)](#).
- Para obter mais informações sobre os métodos assíncronos, consulte [AWS APIs assíncronas da para .NET](#).

PHP e DynamoDB

Neste tutorial, você usa a AWS SDK for PHP para escrever programas simples a fim de realizar as seguintes operações do Amazon DynamoDB:

- Criar uma tabela chamada `Movies` e carregar dados de amostra no formato JSON.
- Realizar operações `create`, `read`, `update` e `delete` na tabela.
- Executar consultas simples.

À medida que você trabalha neste tutorial, consulte o [AWS SDK for PHP Guia do desenvolvedor](#). O [Seção do Amazon DynamoDB no AWS SDK for PHP Referência de API](#) descreve os parâmetros e os resultados para operações do DynamoDB.

Pré-requisitos do tutorial

- Faça download e execute o DynamoDB no computador. Para mais informações, consulte [Configuração do DynamoDB Local \(versão disponível para download\) \(p. 50\)](#).

Note

Você pode usar a versão para download do DynamoDB neste tutorial. Para obter informações sobre como executar o mesmo código para o serviço do DynamoDB, consulte [o Summary \(p. 177\)](#).

- Configurar um AWS Para usar a AWS SDKs. Para mais informações, consulte [Configuração do DynamoDB \(serviço Web\) \(p. 57\)](#).
- Configurar o AWS SDK for PHP:
 - [Instale o PHP](#).
 - [Instale o SDK for PHP](#).

Para obter mais informações, consulte [Conceitos básicos](#) no [AWS SDK for PHP Guia de conceitos básicos](#).

Etapa 1: Criar uma tabela

Nesta etapa, você cria uma tabela chamada `Movies`. A chave primária da tabela é composta dos dois atributos a seguir:

- `year`– A chave de partição. O `AttributeType` é `N` para número.
- `title`– A chave de classificação. O `AttributeType` é `S` para string.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesCreateTable.php`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
require 'vendor/autoload.php';  
  
date_default_timezone_set('UTC');
```

```
use Aws\DynamoDb\Exception\DynamoDbException;

$sdk = new Aws\Sdk([
    'endpoint' => 'http://localhost:8000',
    'region'   => 'us-west-2',
    'version'  => 'latest'
]);

$dynamodb = $sdk->createDynamoDb();

$params = [
    'TableName' => 'Movies',
    'KeySchema' => [
        [
            'AttributeName' => 'year',
            'KeyType' => 'HASH' //Partition key
        ],
        [
            'AttributeName' => 'title',
            'KeyType' => 'RANGE' //Sort key
        ]
    ],
    'AttributeDefinitions' => [
        [
            'AttributeName' => 'year',
            'AttributeType' => 'N'
        ],
        [
            'AttributeName' => 'title',
            'AttributeType' => 'S'
        ],
    ],
    'ProvisionedThroughput' => [
        'ReadCapacityUnits' => 10,
        'WriteCapacityUnits' => 10
    ]
];

try {
    $result = $dynamodb->createTable($params);
    echo 'Created table. Status: ' .
        $result['TableDescription']['TableStatus'] . "\n";
} catch (DynamoDbException $e) {
    echo "Unable to create table:\n";
    echo $e->getMessage() . "\n";
}
```

Note

- Defina o endpoint para indicar que está criando a tabela no Amazon DynamoDB no computador.
- Na chamada `createTable`, especifique o nome da tabela, os atributos da chave primária e seus tipos de dados.
- O `ProvisionedThroughput` parâmetro é necessário, mas a versão para download do DynamoDB o ignora. (O throughput provisionado está além do escopo deste exercício.)

2. Para executar o programa, digite o comando a seguir.

```
php MoviesCreateTable.php
```

Para saber mais sobre como gerenciar tabelas, consulte [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#).

Etapa 2: Carregar dados de exemplo

Nesta etapa, você preenche a tabela `Movies` com dados de exemplo.

Tópicos

- [Etapa 2.1: Faça download do arquivo de dados de exemplo \(p. 161\)](#)
- [Etapa 2.2: Carregar os dados de exemplo para a tabela de filmes \(p. 161\)](#)

Esse cenário usa um arquivo de dados de exemplo que contém informações sobre milhares de filmes do Internet Movie Database (IMDb). Os dados de filme estão no formato JSON, conforme mostrado no exemplo a seguir. Para cada filme, existe um `year`, um `title` e um mapa JSON chamado `info`.

```
[  
  {  
    "year" : ... ,  
    "title" : ... ,  
    "info" : { ... }  
  },  
  {  
    "year" : ....,  
    "title" : ....,  
    "info" : { ... }  
  },  
  ...  
]
```

Nos dados JSON, observe o seguinte:

- Usamos `year` e `title` como os valores de atributo de chave primária da tabela `Movies`.
- O restante dos valores `info` é armazenado em um único atributo chamado `info`. Este programa ilustra como você pode armazenar o JSON em um atributo do Amazon DynamoDB.

Veja a seguir um exemplo de dados de filme.

```
{  
  "year" : 2013,  
  "title" : "Turn It Down, Or Else!",  
  "info" : {  
    "directors" : [  
      "Alice Smith",  
      "Bob Jones"  
    ],  
    "release_date" : "2013-01-18T00:00:00Z",  
    "rating" : 6.2,  
    "genres" : [  
      "Comedy",  
      "Drama"  
    ],  
    "image_url" : "http://ia.media-imdb.com/images/N/  
09ERWAU7FS797AJ7LU8HN09AMUP908RLlo5JF90EWR7LJKQ7@._V1_SX400_.jpg",  
    "plot" : "A rock band plays their music at high volumes, annoying the neighbors.",  
    "rank" : 11,  
  }  
}
```

```
        "running_time_secs" : 5215,
        "actors" : [
            "David Matthewman",
            "Ann Thomas",
            "Jonathan G. Neff"
        ]
    }
```

Etapa 2.1: Faça download do arquivo de dados de exemplo

1. Faça download do arquivo de dados de exemplo: [moviedata.zip](#)
2. Extraia o arquivo de dados (`moviedata.json`) do arquivo.
3. Copie e cole o arquivo `moviedata.json` para o diretório atual.

Etapa 2.2: Carregar os dados de exemplo para a tabela de filmes

Depois de fazer download dos dados de exemplo, é possível executar o programa a seguir para preencher a tabela `Movies`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesLoadData.php`.

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

require 'vendor/autoload.php';

date_default_timezone_set('UTC');

use Aws\DynamoDb\Exception\DynamoDbException;
use Aws\DynamoDb\Marshaler;

$sdk = new Aws\Sdk([
    'endpoint' => 'http://localhost:8000',
    'region'   => 'us-west-2',
    'version'  => 'latest'
]);

$dynamodb = $sdk->createDynamoDb();
$marshaller = new Marshaler();

$tableName = 'Movies';

$movies = json_decode(file_get_contents('moviedata.json'), true);

foreach ($movies as $movie) {

    $year = $movie['year'];
    $title = $movie['title'];
```

```
$info = $movie['info'];

$json = json_encode([
    'year' => $year,
    'title' => $title,
    'info' => $info
]);

$params = [
    'TableName' => $tableName,
    'Item' => $marshaler->marshalJson($json)
];

try {
    $result = $dynamodb->putItem($params);
    echo "Added movie: " . $movie['year'] . " " . $movie['title'] . "\n";
} catch (DynamoDbException $e) {
    echo "Unable to add movie:\n";
    echo $e->getMessage() . "\n";
    break;
}

}
```

Note

O [Classe Marshaler do DynamoDB](#) tem dois métodos para converter documentos JSON e matrizes PHP no formato do DynamoDB. Neste programa, `$marshaler->marshalJson($json)` usa um documento JSON e o converte em um item do DynamoDB.

2. Para executar o programa, digite o comando a seguir.

```
php MoviesLoadData.php
```

Etapa 3: Criar, ler, atualizar e excluir um item

Nesta etapa, você realiza operações de leitura e gravação em um item na tabela `Movies`.

Para saber mais sobre leitura e gravação de dados, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Tópicos

- [Etapa 3.1: Criar um novo item \(p. 162\)](#)
- [Etapa 3.2: Ler um item \(p. 164\)](#)
- [Etapa 3.3: Atualização de um item \(p. 165\)](#)
- [Etapa 3.4: Incrementar um Contador atômico \(p. 167\)](#)
- [Etapa 3.5: Atualização de um item \(condicionalmente\) \(p. 168\)](#)
- [Etapa 3.6: Exclusão de um item \(p. 170\)](#)

Etapa 3.1: Criar um novo item

Nesta etapa, você adiciona um novo item à tabela `Movies`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps01.php`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
require 'vendor/autoload.php';  
  
date_default_timezone_set('UTC');  
  
use Aws\DynamoDb\Exception\DynamoDbException;  
use Aws\DynamoDb\Marshaler;  
  
$sdk = new Aws\Sdk([  
    'endpoint' => 'http://localhost:8000',  
    'region' => 'us-west-2',  
    'version' => 'latest'  
]);  
  
$dynamodb = $sdk->createDynamoDb();  
$marshaler = new Marshaler();  
  
$tableName = 'Movies';  
  
$year = 2015;  
$title = 'The Big New Movie';  
  
$item = $marshaler->marshalJson(''  
{  
    "year": ' . $year . ',  
    "title": "' . $title . '",  
    "info": {  
        "plot": "Nothing happens at all.",  
        "rating": 0  
    }  
}  
' );  
  
$params = [  
    'TableName' => 'Movies',  
    'Item' => $item  
];  
  
try {  
    $result = $dynamodb->putItem($params);  
    echo "Added item: $year - $title\n";  
}  
catch (DynamoDbException $e) {  
    echo "Unable to add item:\n";  
    echo $e->getMessage() . "\n";  
}
```

Note

A chave primária é obrigatória. Este código adiciona um item que tem chave primária (`year`, `title`) e atributos `info`. O atributo `info` armazena um mapa que fornece mais informações sobre o filme.

2. Para executar o programa, digite o comando a seguir.

```
php MoviesItemOps01.php
```

Etapa 3.2: Ler um item

No programa anterior, você adicionou os seguintes itens à tabela.

```
{  
    year: 2015,  
    title: "The Big New Movie",  
    info: {  
        plot: "Nothing happens at all.",  
        rating: 0  
    }  
}
```

Você pode usar o método `getItem` para ler o item da tabela `Movies`. Você deve especificar os valores de chave primária para que possa ler qualquer item de `Movies`, caso saiba seu `year` e `title`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps02.php`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
require 'vendor/autoload.php';  
  
date_default_timezone_set('UTC');  
  
use Aws\DynamoDb\Exception\DynamoDbException;  
use Aws\DynamoDb\Marshaler;  
  
$sdk = new Aws\Sdk([  
    'endpoint' => 'http://localhost:8000',  
    'region' => 'us-west-2',  
    'version' => 'latest'  
]);  
  
$dynamodb = $sdk->createDynamoDb();  
$marshaller = new Marshaler();
```

```
$tableName = 'Movies';

$year = 2015;
$title = 'The Big New Movie';

$key = $marshaler->marshalJson(
{
    "year": ' . $year . ',
    "title": "' . $title . '"
}
');

$params = [
    'TableName' => $tableName,
    'Key' => $key
];

try {
    $result = $dynamodb->getItem($params);
    print_r($result["Item"]);
} catch (DynamoDbException $e) {
    echo "Unable to get item:\n";
    echo $e->getMessage() . "\n";
}
```

2. Para executar o programa, digite o comando a seguir.

```
php MoviesItemOps02.php
```

Etapa 3.3: Atualização de um item

Você pode usar o método `updateItem` para modificar um item existente. Você pode atualizar valores de atributos existentes, adicionar novos atributos ou remover atributos.

Neste exemplo, você executa as seguintes atualizações:

- Altere o valor dos atributos existentes (`rating`, `plot`).
- Adicione um novo atributo de lista (`actors`) ao mapa `info` existente.

Veja a seguir o item existente.

```
{
    year: 2015,
    title: "The Big New Movie",
    info: {
        plot: "Nothing happens at all.",
        rating: 0
    }
}
```

O item é atualizado conforme o seguinte.

```
{
    year: 2015,
    title: "The Big New Movie",
    info: {
        plot: "Everything happens all at once.",
        rating: 5
    }
}
```

```
        rating: 5.5,
        actors: ["Larry", "Moe", "Curly"]
    }
}
```

1. Copie e cole o programa a seguir em um arquivo chamado MoviesItemOps03.php.

```
/** 
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

require 'vendor/autoload.php';

date_default_timezone_set('UTC');

use Aws\DynamoDb\Exception\DynamoDbException;
use Aws\DynamoDb\Marshaler;

$sdk = new Aws\Sdk([
    'endpoint' => 'http://localhost:8000',
    'region'   => 'us-west-2',
    'version'  => 'latest'
]);

$dynamodb = $sdk->createDynamoDb();
$marshaler = new Marshaler();

$tableName = 'Movies';

$year = 2015;
$title = 'The Big New Movie';

$key = $marshaler->marshalJson(
{
    "year": ' . $year . ',
    "title": "' . $title . '"
}
');

$eav = $marshaler->marshalJson(
{
    ":r": 5.5 ,
    ":p": "Everything happens all at once.",
    ":a": [ "Larry", "Moe", "Curly" ]
}
');

$params = [
    'TableName' => $tableName,
    'Key' => $key,
    'UpdateExpression' =>
        'set info.rating = :r, info.plot=:p, info.actors=:a',
]
```

```
'ExpressionAttributeValues'=> $eav,
'ReturnValues' => 'UPDATED_NEW'
];
try {
    $result = $dynamodb->updateItem($params);
    echo "Updated item.\n";
    print_r($result['Attributes']);
} catch (DynamoDbException $e) {
    echo "Unable to update item:\n";
    echo $e->getMessage() . "\n";
}
```

Note

Esse programa usa `UpdateExpression` para descrever todas as atualizações que você deseja realizar no item especificado.

O `ReturnValues` parâmetro instrui o Amazon DynamoDB a retornar somente os atributos atualizados (`UPDATED_NEW`).

2. Para executar o programa, digite o comando a seguir.

```
php MoviesItemOps03.php
```

Etapa 3.4: Incrementar um Contador atômico

O DynamoDB oferece suporte a contadores atômicos, que usam `updateItem` para aumentar ou diminuir o valor de um atributo existente sem interferir em outras solicitações de gravação. (Todas as solicitações de gravação são aplicadas na ordem em que são recebidas.)

O programa a seguir mostra como aumentar a `rating` de um filme. Toda vez que o programa for executado, o atributo é incrementado uma vez.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps04.php`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
require 'vendor/autoload.php';  
  
date_default_timezone_set('UTC');  
  
use Aws\DynamoDb\Exception\DynamoDbException;  
use Aws\DynamoDb\Marshaler;  
  
$sdk = new Aws\Sdk([
```

```
'endpoint' => 'http://localhost:8000',
'region' => 'us-west-2',
'version' => 'latest'
]);
$dynamodb = $sdk->createDynamoDb();
$marshaler = new Marshaler();

$tableName = 'Movies';

$year = 2015;
$title = 'The Big New Movie';

$key = $marshaler->marshalJson(
{
    "year": ' . $year . ',
    "title": "' . $title . ''"
}
');

$eav = $marshaler->marshalJson(
{
    ":val": 1
}
');

$params = [
    'TableName' => $tableName,
    'Key' => $key,
    'UpdateExpression' => 'set info.rating = info.rating + :val',
    'ExpressionAttributeValues'=> $eav,
    'ReturnValues' => 'UPDATED_NEW'
];
try {
    $result = $dynamodb->updateItem($params);
    echo "Updated item. ReturnValues are:\n";
    print_r($result['Attributes']);
} catch (DynamoDbException $e) {
    echo "Unable to update item:\n";
    echo $e->getMessage() . "\n";
}
```

2. Para executar o programa, digite o comando a seguir.

```
php MoviesItemOps04.php
```

Etapa 3.5: Atualização de um item (condicionalmente)

O programa a seguir mostra como usar `UpdateItem` com uma condição. Se a condição for verdadeira, a atualização será bem-sucedida; caso contrário, a atualização não será realizada.

Neste caso, o item será atualizado somente se houver mais de três atores no filme.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps05.php`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 */
```

```
/*
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

require 'vendor/autoload.php';

date_default_timezone_set('UTC');

use Aws\DynamoDb\Exception\DynamoDbException;
use Aws\DynamoDb\Marshaler;

$Sdk = new Aws\Sdk([
    'endpoint' => 'http://localhost:8000',
    'region'   => 'us-west-2',
    'version'  => 'latest'
]);

$dynamodb = $Sdk->createDynamoDb();
$marshaller = new Marshaler();

$tableName = 'Movies';

$year = 2015;
$title = 'The Big New Movie';

$key = $marshaller->marshalJson(
{
    "year": ' . $year . ',
    "title": "' . $title . '"
}
');

$eav = $marshaller->marshalJson(
{
    ":num": 3
}
');

$params = [
    'TableName' => $tableName,
    'Key' => $key,
    'UpdateExpression' => 'remove info.actors[0]',
    'ConditionExpression' => 'size(info.actors) > :num',
    'ExpressionAttributeValues'=> $eav,
    'ReturnValues' => 'UPDATED_NEW'
];

try {
    $result = $dynamodb->updateItem($params);
    echo "Updated item. ReturnValues are:\n";
    print_r($result['Attributes']);
}

} catch (DynamoDbException $e) {
    echo "Unable to update item:\n";
    echo $e->getMessage() . "\n";
}
```

2. Para executar o programa, digite o comando a seguir.

```
php MoviesItemOps05.php
```

O programa deve falhar com a mensagem a seguir.

A solicitação condicional falhou

O programa falha porque o filme tem três atores, mas a condição está em busca de mais de três atores.

3. Modifique o programa para que ConditionExpression fique semelhante ao seguinte.

```
ConditionExpression="size(info.actors) >= :num",
```

A condição agora é maior ou igual a 3 em vez de maior que 3.

4. Execute o programa novamente. A operação UpdateItem agora deve ser bem-sucedida.

Etapa 3.6: Exclusão de um item

Você pode usar o método `deleteItem` para excluir um item, especificando sua chave primária. Se desejar, você pode fornecer uma `ConditionExpression` para evitar a exclusão do item, se a condição não for atendida.

No exemplo a seguir, você tenta excluir um item de filme específico se a sua classificação for 5 ou menos.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps06.php`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
require 'vendor/autoload.php';  
  
date_default_timezone_set('UTC');  
  
use Aws\DynamoDb\Exception\DynamoDbException;  
use Aws\DynamoDb\Marshaler;  
  
$sdk = new Aws\Sdk([  
    'endpoint' => 'http://localhost:8000',  
    'region' => 'us-west-2',  
    'version' => 'latest'  
]);  
  
$dynamodb = $sdk->createDynamoDb();  
$marshaller = new Marshaler();
```

```
$tableName = 'Movies';

$year = 2015;
$title = 'The Big New Movie';

$key = $marshaler->marshalJson(
{
    "year": ' . $year . ',
    "title": "' . $title . '"
}
');

$eav = $marshaler->marshalJson(
{
    ":val": 5
}
');

$params = [
    'TableName' => $tableName,
    'Key' => $key,
    'ConditionExpression' => 'info.rating <= :val',
    'ExpressionAttributeValues'=> $eav
];

try {
    $result = $dynamodb->deleteItem($params);
    echo "Deleted item.\n";

} catch (DynamoDbException $e) {
    echo "Unable to delete item:\n";
    echo $e->getMessage() . "\n";
}
```

2. Para executar o programa, digite o comando a seguir.

```
php MoviesItemOps06.php
```

O programa deve falhar com a mensagem a seguir.

A solicitação condicional falhou

O programa falha porque a classificação desse filme específico é maior que 5.

3. Modifique o programa para remover a condição.

```
$params = [
    'TableName' => $tableName,
    'Key' => $key
];
```

4. Execute o programa. Agora, a exclusão é bem-sucedida, pois você removeu a condição.

Etapa 4: Consultar e verificar os dados

Você pode usar o método `query` para recuperar os dados de uma tabela. Você deve especificar o valor de uma chave de partição. A chave de classificação é opcional.

A chave primária da tabela `Movies` é composta pelo seguinte:

- **year**— A chave de partição. O tipo de atributo é `number`.
- **title**— A chave de classificação. O tipo de atributo é `string`.

Para encontrar todos os filmes lançados durante um ano, você precisa especificar somente o `year`. Você também pode fornecer o `title` para recuperar um subconjunto de filmes baseados na mesma condição (na chave de classificação). Por exemplo, encontrar filmes lançados em 2014 que têm um título começando com a letra "A".

Além do método `query`, é possível usar o método `scan` para recuperar todos os dados da tabela.

Para saber mais sobre como consultar e verificar dados, consulte [Como trabalhar com consultas no DynamoDB \(p. 490\)](#) e [Como trabalhar com verificações no DynamoDB \(p. 508\)](#), respectivamente.

Tópicos

- [Etapa 4.1: Consulta — Todos os filmes lançados em um ano \(p. 172\)](#)
- [Etapa 4.2: Consulta — Todos os filmes lançados em um ano com determinados títulos \(p. 173\)](#)
- [Etapa 4.3: Scan \(p. 175\)](#)

Etapa 4.1: Consulta — Todos os filmes lançados em um ano

O programa incluído nesta etapa recupera todos os filmes lançados no `year` 1985.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesQuery01.php`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
require 'vendor/autoload.php';  
  
date_default_timezone_set('UTC');  
  
use Aws\DynamoDb\Exception\DynamoDbException;  
use Aws\DynamoDb\Marshaler;  
  
$sdk = new Aws\Sdk([  
    'endpoint' => 'http://localhost:8000',  
    'region'  => 'us-west-2',  
    'version'  => 'latest'  
]);  
  
$dynamodb = $sdk->createDynamoDb();  
$marshaller = new Marshaler();  
  
$tableName = 'Movies';  
  
$eav = $marshaller->marshalJson(''  
    ':yyyy": 1985
```

```
    }
');

$params = [
    'TableName' => $tableName,
    'KeyConditionExpression' => '#yr = :yyyy',
    'ExpressionAttributeNames'=> [ '#yr' => 'year' ],
    'ExpressionAttributeValues'=> $eav
];
echo "Querying for movies from 1985.\n";

try {
    $result = $dynamodb->query($params);

    echo "Query succeeded.\n";

    foreach ($result['Items'] as $movie) {
        echo $marshaler->unmarshalValue($movie['year']) . ': ' .
            $marshaler->unmarshalValue($movie['title']) . "\n";
    }
} catch (DynamoDbException $e) {
    echo "Unable to query:\n";
    echo $e->getMessage() . "\n";
}
```

Note

- `ExpressionAttributeNames` fornece substituição de nome. Usamos isso porque `year` é uma palavra reservada no DynamoDB. Você não pode usá-la diretamente em nenhuma expressão, incluindo `KeyConditionExpression`. Use o nome de atributo de expressão `#yr` para resolver essa questão.
 - `ExpressionAttributeValues` fornece substituição de valor. Ele é usado porque você não pode usar literais em nenhuma expressão, incluindo `KeyConditionExpression`. Você pode usar o valor de atributo de expressão `:yyyy` para resolver essa questão.
2. Para executar o programa, digite o comando a seguir:

```
php MoviesItemQuery01.php
```

Note

O programa anterior mostra como consultar uma tabela por seus atributos de chave primária. No Amazon DynamoDB, você pode opcionalmente criar um ou mais índices secundários em uma tabela e consultá-los da mesma forma que consulta uma tabela. Os índices secundários oferecem aos seus aplicativos mais flexibilidade ao permitir consultas nos atributos que não são chave. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 4.2: Consulta — Todos os filmes lançados em um ano com determinados títulos

O programa incluído nesta etapa recupera todos os filmes lançados no `year` 1992, cujo `title` começa com a letra "A" até a letra "L".

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesQuery02.php`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
require 'vendor/autoload.php';  
  
date_default_timezone_set('UTC');  
  
use Aws\DynamoDb\Exception\DynamoDbException;  
use Aws\DynamoDb\Marshaler;  
  
$sdk = new Aws\Sdk([  
    'endpoint' => 'http://localhost:8000',  
    'region' => 'us-west-2',  
    'version' => 'latest'  
]);  
  
$dynamodb = $sdk->createDynamoDb();  
$marshaler = new Marshaler();  
  
$tableName = 'Movies';  
  
$eav = $marshaler->marshalJson(''  
{  
    ":yyyy":1992,  
    ":letter1": "A",  
    ":letter2": "L"  
}  
'');  
  
$params = [  
    'TableName' => $tableName,  
    'ProjectionExpression' => '#yr, title, info.genres, info.actors[0]',  
    'KeyConditionExpression' =>  
        '#yr = :yyyy and title between :letter1 and :letter2',  
    'ExpressionAttributeNames'=> [ '#yr' => 'year' ],  
    'ExpressionAttributeValues'=> $eav  
];  
  
echo "Querying for movies from 1992 - titles A-L, with genres and lead actor\n";  
  
try {  
    $result = $dynamodb->query($params);  
  
    echo "Query succeeded.\n";  
  
    foreach ($result['Items'] as $i) {  
        $movie = $marshaler->unmarshalItem($i);  
        print $movie['year'] . ':' . $movie['title'] . ' ... ';  
  
        foreach ($movie['info']['genres'] as $gen) {  
            print $gen . ' ';  
        }  
    }  
}
```

```
        echo ' ... ' . $movie['info']['actors'][0] . "\n";
    }

} catch (DynamoDbException $e) {
    echo "Unable to query:\n";
    echo $e->getMessage() . "\n";
}
```

2. Para executar o programa, digite o comando a seguir.

```
php MoviesQuery02.php
```

Etapa 4.3: Scan

O método `scan` lê cada item da tabela inteira e retorna todos os dados dela. Você pode fornecer uma `filter_expression` opcional, para que apenas os itens que correspondem aos seus critérios sejam retornados. No entanto, o filtro é aplicado somente depois que a tabela inteira foi verificada.

O programa a seguir verifica a tabela `Movies` inteira, que contém aproximadamente 5.000 itens. O exemplo especifica o filtro opcional para recuperar somente os filmes de 1950 (aproximadamente 100 itens) e descartar todos os outros.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesScan.php`.

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

require 'vendor/autoload.php';

date_default_timezone_set('UTC');

use Aws\DynamoDb\Exception\DynamoDbException;
use Aws\DynamoDb\Marshaler;

$ sdk = new Aws\Sdk([
    'endpoint' => 'http://localhost:8000',
    'region'   => 'us-west-2',
    'version'  => 'latest'
]);

$dynamodb = $sdk->createDynamoDb();

$marshaller = new Marshaler();

//Expression attribute values
$eav = $marshaller->marshalJson('
```

```
{  
    ":start_yr": 1950,  
    ":end_yr": 1959  
}  
');  
  
$params = [  
    'TableName' => 'Movies',  
    'ProjectionExpression' => '#yr, title, info.rating',  
    'FilterExpression' => '#yr between :start_yr and :end_yr',  
    'ExpressionAttributeNames'=> [ '#yr' => 'year' ],  
    'ExpressionAttributeValues'=> $eav  
];  
  
echo "Scanning Movies table.\n";  
  
try {  
    while (true) {  
        $result = $dynamodb->scan($params);  
  
        foreach ($result['Items'] as $i) {  
            $movie = $marshaler->unmarshalItem($i);  
            echo $movie['year'] . ':' . $movie['title'];  
            echo ' ... ' . $movie['info']['rating']  
            . "\n";  
        }  
  
        if (isset($result['LastEvaluatedKey'])) {  
            $params['ExclusiveStartKey'] = $result['LastEvaluatedKey'];  
        } else {  
            break;  
        }  
    }  
}  
  
} catch (DynamoDbException $e) {  
    echo "Unable to scan:\n";  
    echo $e->getMessage() . "\n";  
}
```

No código, observe o seguinte:

- `ProjectionExpression` especifica os atributos que você deseja no resultado da verificação.
 - `FilterExpression` especifica uma condição que retorna apenas os itens que satisfazem à condição. Todos os outros itens são descartados.
2. Para executar o programa, digite o comando a seguir.

```
php MoviesScan.php
```

Note

Também é possível usar a operação `Scan` com quaisquer índices secundários que tenha criado na tabela. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 5: (Opcional) Excluir a tabela

Para excluir a tabela `Movies`:

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesDeleteTable.php`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
require 'vendor/autoload.php';  
  
date_default_timezone_set('UTC');  
  
use Aws\DynamoDb\Exception\DynamoDbException;  
  
$sdk = new Aws\Sdk([  
    'endpoint' => 'http://localhost:8000',  
    'region'   => 'us-west-2',  
    'version'   => 'latest'  
]);  
  
$dynamodb = $sdk->createDynamoDb();  
  
$params = [  
    'TableName' => 'Movies'  
];  
  
try {  
    $result = $dynamodb->deleteTable($params);  
    echo "Deleted table.\n";  
  
} catch (DynamoDbException $e) {  
    echo "Unable to delete table:\n";  
    echo $e->getMessage() . "\n";  
}
```

2. Para executar o programa, digite o comando a seguir.

```
php MoviesDeleteTable.php
```

Summary

Neste tutorial, você criou o `Movies` no Amazon DynamoDB no computador e realizou operações básicas. A versão para download do DynamoDB é útil durante o desenvolvimento e o teste de aplicativos. No entanto, quando estiver pronto para executar o aplicativo em um ambiente de produção, você precisará modificar o código para que ele use o serviço web do DynamoDB.

Modificação do código para usar o serviço DynamoDB

Para usar o web service do DynamoDB, altere o endpoint em seu aplicativo. Para fazer isso, localize as seguintes linhas no código.

```
$sdk = new Aws\Sdk([
    'endpoint' => 'http://localhost:8000',
    'region'   => 'us-west-2',
    'version'  => 'latest'
]);
```

Remova o parâmetro endpoint para que o código fique semelhante ao seguinte.

```
$sdk = new Aws\Sdk([
    'region'   => 'us-west-2',
    'version'  => 'latest'
]);
```

Depois que você remover essa linha, o código poderá acessar o web service do DynamoDB noAWSRegião especificada pelo regionValor de configuração. Por exemplo, a linha a seguir especifica que você deseja usar a região oeste dos EUA (Oregon).

```
'region' => 'us-west-2',
```

Em vez de usar a versão disponível para download do DynamoDB no seu computador, o programa agora usa o endpoint de serviço do DynamoDB na região oeste dos EUA (Oregon).

O DynamoDB está disponível em AWS Regiões do mundo todo. Para obter a lista completa, consulte [Regiões e endpoints](#) no AWS Referência geral. Para obter mais informações sobre como definir regiões e endpoints no seu código, consulte o [boto: Uma interface do Python para a Amazon Web Services](#).

Conceitos básicos do desenvolvimento com Python e DynamoDB

Neste tutorial, você usa o AWS SDK for Python (Boto3) para escrever programas simples a fim de realizar as seguintes operações do Amazon DynamoDB:

- Criar uma tabela chamada `Movies` e carregar dados de amostra no formato JSON.
- Realizar operações `create`, `read`, `update` e `delete` na tabela.
- Executar consultas simples.

Ao trabalhar com este tutorial, é possível consultar a documentação do AWS SDK for Python (Boto). As seções a seguir são específicas para o DynamoDB:

- [Tutorial do DynamoDB](#)
- [Cliente de baixo nível do DynamoDB](#)

Pré-requisitos do tutorial

- Faça download e execute o DynamoDB em seu computador. Para mais informações, consulte [Configuração do DynamoDB Local \(versão disponível para download\)](#) (p. 50).

Note

Você pode usar a versão para download do DynamoDB neste tutorial.

No [Summary \(p. 192\)](#) Explicamos como executar o mesmo código no serviço web do DynamoDB.

- Configurar um AWS Para usar a AWSSDKs do. Para mais informações, consulte [Configuração do DynamoDB \(serviço Web\) \(p. 57\)](#).
- Instalar o Python 2.6 ou posterior. Para obter mais informações, consulte <https://www.python.org/downloads>. Para obter instruções, consulte [Início rápido](#) na documentação do Boto 3.

Etapa 1: Criar uma tabela com Python

Nesta etapa, você cria uma tabela chamada `Movies`. A chave primária da tabela é composta dos seguintes atributos:

- `year`– A chave de partição. O `AttributeType` é `N` para número.
- `title`– A chave de classificação. O `AttributeType` é `S` para string.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesCreateTable.py`.

```
import boto3

def create_movie_table(dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")

    table = dynamodb.create_table(
        TableName='Movies',
        KeySchema=[
            {
                'AttributeName': 'year',
                'KeyType': 'HASH' # Partition key
            },
            {
                'AttributeName': 'title',
                'KeyType': 'RANGE' # Sort key
            }
        ],
        AttributeDefinitions=[
            {
                'AttributeName': 'year',
                'AttributeType': 'N'
            },
            {
                'AttributeName': 'title',
                'AttributeType': 'S'
            },
        ],
        ProvisionedThroughput={
            'ReadCapacityUnits': 10,
            'WriteCapacityUnits': 10
        }
    )
    return table

if __name__ == '__main__':
```

```
movie_table = create_movie_table()
print("Table status:", movie_table.table_status)
```

Note

- Defina o endpoint para indicar que está criando a tabela no[versão transferível do DynamoDB](#) em seu computador.
 - Na chamada `create_table`, especifique o nome da tabela, os atributos da chave primária e seus tipos de dados.
 - O parâmetro `ProvisionedThroughput` é obrigatório. No entanto, a versão para download do DynamoDB o ignora. (O throughput provisionado está além do escopo deste exercício.)
 - Estes exemplos usam a função de estilo `print` do Python 3. A linha `from __future__ import print_function` permite a impressão do Python 3 no Python 2.6 e posterior.
2. Para executar o programa, digite o comando a seguir.

```
python MoviesCreateTable.py
```

Para saber mais sobre como gerenciar tabelas, consulte [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#).

Etapa 2: Carregar dados de exemplo

Nesta etapa, você preenche a tabela `Movies` com dados de exemplo.

Tópicos

- [Etapa 2.1: Faça download do arquivo de dados de exemplo \(p. 181\)](#)
- [Etapa 2.2: Carregar os dados de exemplo para a tabela de filmes \(p. 181\)](#)

Esse cenário usa um arquivo de dados de exemplo que contém informações sobre milhares de filmes do Internet Movie Database (IMDb). Os dados de filme estão no formato JSON, conforme mostrado no exemplo a seguir. Para cada filme, existe um `year`, um `title` e um mapa JSON chamado `info`.

```
[  
  {  
    "year" : ... ,  
    "title" : ... ,  
    "info" : { ... }  
  },  
  {  
    "year" : ...,  
    "title" : ....,  
    "info" : { ... }  
  },  
  ...  
]
```

Nos dados JSON, observe o seguinte:

- Usamos `year` e `title` como os valores de atributo de chave primária da tabela `Movies`.
- O restante dos valores `info` é armazenado em um único atributo chamado `info`. Este programa ilustra como é possível armazenar o JSON em um atributo do Amazon DynamoDB.

Veja a seguir um exemplo de dados de filme.

```
{  
    "year" : 2013,  
    "title" : "Turn It Down, Or Else!",  
    "info" : {  
        "directors" : [  
            "Alice Smith",  
            "Bob Jones"  
        ],  
        "release_date" : "2013-01-18T00:00:00Z",  
        "rating" : 6.2,  
        "genres" : [  
            "Comedy",  
            "Drama"  
        ],  
        "image_url" : "http://ia.media-imdb.com/images/N/  
09ERWAU7FS797AJ7LU8HN09AMUP908RLl05JF90EWR7LJKQ7@._V1_SX400_.jpg",  
        "plot" : "A rock band plays their music at high volumes, annoying the neighbors.",  
        "rank" : 11,  
        "running_time_secs" : 5215,  
        "actors" : [  
            "David Matthewman",  
            "Ann Thomas",  
            "Jonathan G. Neff"  
        ]  
    }  
}
```

Etapa 2.1: Faça download do arquivo de dados de exemplo

1. Faça download do arquivo de dados de exemplo: [moviedata.zip](#)
2. Extraia o arquivo de dados (`moviedata.json`) do arquivo.
3. Copie o arquivo `moviedata.json` para o seu diretório atual.

Etapa 2.2: Carregar os dados de exemplo para a tabela de filmes

Depois de fazer download dos dados de exemplo, é possível executar o programa a seguir para preencher a tabela `Movies`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesLoadData.py`.

```
from decimal import Decimal  
import json  
import boto3  
  
def load_movies(movies, dynamodb=None):  
    if not dynamodb:  
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")  
  
    table = dynamodb.Table('Movies')  
    for movie in movies:  
        year = int(movie['year'])  
        title = movie['title']  
        print("Adding movie:", year, title)  
        table.put_item(Item=movie)
```

```
if __name__ == '__main__':
    with open("moviedata.json") as json_file:
        movie_list = json.load(json_file, parse_float=Decimal)
    load_movies(movie_list)
```

2. Para executar o programa, digite o comando a seguir.

```
python MoviesLoadData.py
```

Etapa 3: Criar, ler, atualizar e excluir um item com Python

Nesta etapa, você realiza operações de leitura e gravação em um item na tabela `Movies`.

Para saber mais sobre leitura e gravação de dados, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Tópicos

- [Etapa 3.1: Criar um novo item \(p. 182\)](#)
- [Etapa 3.2: Ler um item \(p. 183\)](#)
- [Etapa 3.3: Atualização de um item \(p. 184\)](#)
- [Etapa 3.4: Incrementar um Contador atômico \(p. 185\)](#)
- [Etapa 3.5: Atualização de um item \(condicionalmente\) \(p. 186\)](#)
- [Etapa 3.6: Exclusão de um item \(p. 187\)](#)

Etapa 3.1: Criar um novo item

Nesta etapa, você adiciona um novo item à tabela `Movies`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps01.py`.

```
from pprint import pprint
import boto3

def put_movie(title, year, plot, rating, dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")

    table = dynamodb.Table('Movies')
    response = table.put_item(
        Item={
            'year': year,
            'title': title,
            'info': {
                'plot': plot,
                'rating': rating
            }
        }
    )
    return response

if __name__ == '__main__':
    movie_resp = put_movie("The Big New Movie", 2015,
                           "Nothing happens at all.", 0)
```

```
print("Put movie succeeded:")
pprint(movie_resp, sort_dicts=False)
```

Note

- A chave primária é obrigatória. Este código adiciona um item que tem chave primária (`year`, `title`) e atributos `info`. O atributo `info` armazena o JSON de amostra que fornece mais informações sobre o filme.
 - A classe `DecimalEncoder` é usada para imprimir números armazenados usando a classe `Decimal`. O SDK da Boto usa o `Decimal` para manter valores numéricos do Amazon DynamoDB.
2. Para executar o programa, digite o comando a seguir.

```
python MoviesItemOps01.py
```

Etapa 3.2: Ler um item

No programa anterior, você adicionou os seguintes itens à tabela.

```
{
    year: 2015,
    title: "The Big New Movie",
    info: {
        plot: "Nothing happens at all.",
        rating: 0
    }
}
```

Você pode usar o método `get_item` para ler o item da tabela `Movies`. Você deve especificar os valores de chave primária para que possa ler qualquer item de `Movies`, caso saiba seu `year` e `title`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps02.py`.

```
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

def get_movie(title, year, dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")

    table = dynamodb.Table('Movies')

    try:
        response = table.get_item(Key={'year': year, 'title': title})
    except ClientError as e:
        print(e.response['Error']['Message'])
    else:
        return response['Item']

if __name__ == '__main__':
    movie = get_movie("The Big New Movie", 2015,)
    if movie:
        print("Get movie succeeded:")
        pprint(movie, sort_dicts=False)
```

2. Para executar o programa, digite o comando a seguir.

```
python MoviesItemOps02.py
```

Etapa 3.3: Atualização de um item

Você pode usar o método `update_item` para modificar um item existente. Você pode atualizar valores de atributos existentes, adicionar novos atributos ou remover atributos.

Neste exemplo, você executa as seguintes atualizações:

- Altere o valor dos atributos existentes (`rating`, `plot`).
- Adicione um novo atributo de lista (`actors`) ao mapa `info` existente.

Veja a seguir o item existente.

```
{  
    year: 2015,  
    title: "The Big New Movie",  
    info: {  
        plot: "Nothing happens at all.",  
        rating: 0  
    }  
}
```

O item é atualizado conforme o seguinte.

```
{  
    year: 2015,  
    title: "The Big New Movie",  
    info: {  
        plot: "Everything happens all at once.",  
        rating: 5.5,  
        actors: ["Larry", "Moe", "Curly"]  
    }  
}
```

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps03.py`.

```
from decimal import Decimal  
from pprint import pprint  
import boto3  
  
def update_movie(title, year, rating, plot, actors, dynamodb=None):  
    if not dynamodb:  
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")  
  
    table = dynamodb.Table('Movies')  
  
    response = table.update_item(  
        Key={  
            'year': year,  
            'title': title  
        },  
        UpdateExpression="set info.rating=:r, info.plot=:p, info.actors=:a",  
        ExpressionAttributeValues={  
            ':r': Decimal(rating),  
            ':p': plot,
```

```
        ':a': actors
    },
    ReturnValues="UPDATED_NEW"
)
return response

if __name__ == '__main__':
    update_response = update_movie(
        "The Big New Movie", 2015, 5.5, "Everything happens all at once.",
        ["Larry", "Moe", "Curly"])
    print("Update movie succeeded:")
    pprint(update_response, sort_dicts=False)
```

Note

Esse programa usa `UpdateExpression` para descrever todas as atualizações que você deseja realizar no item especificado. O parâmetro `ReturnValues` instrui o DynamoDB a retornar apenas os atributos atualizados (`UPDATED_NEW`).

2. Para executar o programa, digite o comando a seguir.

```
python MoviesItemOps03.py
```

Etapa 3.4: Incrementar um Contador atômico

O DynamoDB oferece suporte a contadores atômicos, que usam `update_item` para aumentar ou diminuir o valor de um atributo existente sem interferir em outras solicitações de gravação. (Todas as solicitações de gravação são aplicadas na ordem em que são recebidas.)

O programa a seguir mostra como aumentar a `rating` de um filme. Toda vez que o programa for executado, o atributo é incrementado uma vez.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps04.py`.

```
from decimal import Decimal
from pprint import pprint
import boto3

def increase_rating(title, year, rating_increase, dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")

    table = dynamodb.Table('Movies')

    response = table.update_item(
        Key={
            'year': year,
            'title': title
        },
        UpdateExpression="set info.rating = info.rating + :val",
        ExpressionAttributeValues={
            ':val': Decimal(rating_increase)
        },
        ReturnValues="UPDATED_NEW"
    )
    return response

if __name__ == '__main__':
```

```
update_response = increase_rating("The Big New Movie", 2015, 1)
print("Update movie succeeded:")
pprint(update_response, sort_dicts=False)
```

2. Para executar o programa, digite o comando a seguir.

```
python MoviesItemOps04.py
```

Etapa 3.5: Atualização de um item (condicionalmente)

O programa a seguir mostra como usar `UpdateItem` com uma condição. Se a condição for verdadeira, a atualização será bem-sucedida; caso contrário, a atualização não será realizada.

Neste caso, o item será atualizado somente se houver mais de três atores.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps05.py`.

```
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

def remove_actors(title, year, actor_count, dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")

    table = dynamodb.Table('Movies')

    try:
        response = table.update_item(
            Key={
                'year': year,
                'title': title
            },
            UpdateExpression="remove info.actors[0]",
            ConditionExpression="size(info.actors) > :num",
            ExpressionAttributeValues={':num': actor_count},
            ReturnValues="UPDATED_NEW"
        )
    except ClientError as e:
        if e.response['Error']['Code'] == "ConditionalCheckFailedException":
            print(e.response['Error']['Message'])
        else:
            raise
    else:
        return response

if __name__ == '__main__':
    print("Attempting conditional update (expecting failure)...")
    update_response = remove_actors("The Big New Movie", 2015, 3)
    if update_response:
        print("Update movie succeeded:")
        pprint(update_response, sort_dicts=False)
```

2. Para executar o programa, digite o comando a seguir.

```
python MoviesItemOps05.py
```

O programa deve falhar com a mensagem a seguir.

A solicitação condicional falhou

O programa falha porque o filme tem três atores, mas a condição está em busca de mais de três atores.

3. Modifique o programa para que `ConditionExpression` fique semelhante ao seguinte.

```
ConditionExpression="size(info.actors) >= :num",
```

A condição agora é maior ou igual a 3 em vez de maior que 3.

4. Execute o programa novamente. A operação `UpdateItem` agora deve ser bem-sucedida.

Etapa 3.6: Exclusão de um item

Você pode usar o método `delete_item` para excluir um item, especificando sua chave primária. Se desejar, é possível fornecer uma `ConditionExpression` para evitar a exclusão do item se a condição não for atendida.

No exemplo a seguir, você tenta excluir um item de filme específico se a sua classificação for 5 ou menos.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps06.py`.

```
from decimal import Decimal
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

def delete_underrated_movie(title, year, rating, dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")

    table = dynamodb.Table('Movies')

    try:
        response = table.delete_item(
            Key={
                'year': year,
                'title': title
            },
            ConditionExpression="info.rating <= :val",
            ExpressionAttributeValues={
                ":val": Decimal(rating)
            }
        )
    except ClientError as e:
        if e.response['Error']['Code'] == "ConditionalCheckFailedException":
            print(e.response['Error']['Message'])
        else:
            raise
    else:
        return response

if __name__ == '__main__':
    print("Attempting a conditional delete...")
    delete_response = delete_underrated_movie("The Big New Movie", 2015, 5)
    if delete_response:
        print("Delete movie succeeded:")
        pprint(delete_response, sort_dicts=False)
```

2. Para executar o programa, digite o comando a seguir.

```
python MoviesItemOps06.py
```

O programa deve falhar com a mensagem a seguir.

A solicitação condicional falhou

O programa falha porque a classificação desse filme em especial é maior que 5.

3. Modifique o programa para remover a condição em `table.delete_item`.

```
response = table.delete_item(  
    Key={  
        'year': year,  
        'title': title  
    }  
)
```

4. Execute o programa. Agora, a exclusão é bem-sucedida, pois você removeu a condição.

Etapa 4: Consultar e verificar os dados

Você pode usar o método `query` para recuperar os dados de uma tabela. Você deve especificar o valor de uma chave de partição. A chave de classificação é opcional.

A chave primária da tabela `Movies` é composta pelo seguinte:

- `year`— A chave de partição. O tipo de atributo é `number`.
- `title`— A chave de classificação. O tipo de atributo é `string`.

Para encontrar todos os filmes lançados durante um ano, você precisa especificar somente o `year`. Você também pode fornecer o `title` para recuperar um subconjunto de filmes baseados na mesma condição (na chave de classificação). Por exemplo, é possível encontrar filmes lançados em 2014 que têm um título começando com a letra "A".

Além do método `query`, é possível usar o método `scan` para recuperar todos os dados da tabela.

Para saber mais sobre como consultar e verificar dados, consulte [Como trabalhar com consultas no DynamoDB \(p. 490\)](#) e [Como trabalhar com verificações no DynamoDB \(p. 508\)](#), respectivamente.

Tópicos

- [Etapa 4.1: Consulta — Todos os filmes lançados em um ano \(p. 188\)](#)
- [Etapa 4.2: Consulta — Todos os filmes lançados em um ano com determinados títulos \(p. 189\)](#)
- [Etapa 4.3: Scan \(p. 190\)](#)

Etapa 4.1: Consulta — Todos os filmes lançados em um ano

O programa incluído nesta etapa recupera todos os filmes lançados no `year` 1985.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesQuery01.py`.

```
import boto3  
from boto3.dynamodb.conditions import Key  
  
def query_movies(year, dynamodb=None):  
    if not dynamodb:  
        dynamodb = boto3.resource('dynamodb')
```

```
dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")

table = dynamodb.Table('Movies')
response = table.query(
    KeyConditionExpression=Key('year').eq(year)
)
return response['Items']

if __name__ == '__main__':
    query_year = 1985
    print(f"Movies from {query_year}")
    movies = query_movies(query_year)
    for movie in movies:
        print(movie['year'], ":", movie['title'])
```

Note

O Boto 3 SDK cria uma `ConditionExpression` para você durante o uso das funções `Key` e `Attr` importadas de `boto3.dynamodb.conditions`. Você também pode especificar uma `ConditionExpression` como string.

Para obter uma lista de condições disponíveis para o Amazon DynamoDB, consulte [Condições do DynamoDB](#) em [AWS Conceitos básicos do SDK for Python \(Boto3\)](#). Para mais informações, consulte [Expressões de condição \(p. 422\)](#).

2. Para executar o programa, digite o comando a seguir.

```
python MoviesQuery01.py
```

Note

O programa anterior mostra como consultar uma tabela por seus atributos de chave primária. No DynamoDB, se desejar, é possível criar um ou mais índices secundários em uma tabela e consultá-los da mesma forma que consulta uma tabela. Os índices secundários oferecem aos seus aplicativos mais flexibilidade ao permitir consultas nos atributos que não são chave. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 4.2: Consulta — Todos os filmes lançados em um ano com determinados títulos

O programa incluído nesta etapa recupera todos os filmes lançados no year 1992, cujo title começa com a letra "A" até a letra "L".

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesQuery02.py`.

```
from pprint import pprint
import boto3
from boto3.dynamodb.conditions import Key

def query_and_project_movies(year, title_range, dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")

    table = dynamodb.Table('Movies')
    print(f"Get year, title, genres, and lead actor")

    # Expression attribute names can only reference items in the projection expression.
    response = table.query(
        ProjectionExpression="#yr, title, info.genres, info.actors[0]",
```

```
        ExpressionAttributeNames={"#yr": "year"},  
        KeyConditionExpression=  
            Key('year').eq(year) & Key('title').between(title_range[0], title_range[1])  
    )  
    return response['Items']  
  
if __name__ == '__main__':  
    query_year = 1992  
    query_range = ('A', 'L')  
    print(f"Get movies from {query_year} with titles from "  
        f"{query_range[0]} to {query_range[1]}")  
    movies = query_and_project_movies(query_year, query_range)  
    for movie in movies:  
        print(f"\n{movie['year']} : {movie['title']}")  
        pprint(movie['info'])
```

2. Para executar o programa, digite o comando a seguir.

```
python MoviesQuery02.py
```

Etapa 4.3: Scan

O método `scan` lê todos os itens da tabela inteira e retorna todos os dados nela contidos. Você pode fornecer uma `filter_expression` opcional, para que apenas os itens que correspondem aos seus critérios sejam retornados. No entanto, o filtro é aplicado somente depois que a tabela inteira foi verificada.

O programa a seguir verifica a tabela `Movies` inteira, que contém aproximadamente 5.000 itens. A verificação específica o filtro opcional para recuperar somente os filmes da década de 1950 (aproximadamente 100 itens) e descartar todos os outros.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesScan.py`.

```
from pprint import pprint  
import boto3  
from boto3.dynamodb.conditions import Key  
  
def scan_movies(year_range, display_movies, dynamodb=None):  
    if not dynamodb:  
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")  
  
    table = dynamodb.Table('Movies')  
    scan_kwargs = {  
        'FilterExpression': Key('year').between(*year_range),  
        'ProjectionExpression': "#yr, title, info.rating",  
        'ExpressionAttributeNames': {"#yr": "year"}  
    }  
  
    done = False  
    start_key = None  
    while not done:  
        if start_key:  
            scan_kwargs['ExclusiveStartKey'] = start_key  
        response = table.scan(**scan_kwargs)  
        display_movies(response.get('Items', []))  
        start_key = response.get('LastEvaluatedKey', None)  
        done = start_key is None  
  
if __name__ == '__main__':  
    def print_movies(movies):
```

```
for movie in movies:
    print(f"\n{movie['year']} : {movie['title']}")
    pprint(movie['info'])

query_range = (1950, 1959)
print(f"Scanning for movies released from {query_range[0]} to {query_range[1]}...")
scan_movies(query_range, print_movies)
```

No código, observe o seguinte:

- `ProjectionExpression` especifica os atributos que você deseja no resultado da verificação.
- `FilterExpression` especifica uma condição que retorna apenas os itens que satisfazem à condição. Todos os outros itens são descartados.
- O método `scan` retorna um subconjunto de itens por vez, chamado de página. O valor `LastEvaluatedKey` na resposta é passado para o método `scan` por meio do parâmetro `ExclusiveStartKey`. Quando a última página é retornada, `LastEvaluatedKey` não é parte da resposta.

Note

- `ExpressionAttributeNames` fornece substituição de nome. Usamos isso porque `year` É uma palavra reservada no DynamoDB. Você não pode usá-la diretamente em nenhuma expressão, incluindo `KeyConditionExpression`. Use o nome de atributo de expressão `#yr` para resolver essa questão.
- `ExpressionAttributeValues` fornece substituição de valor. Ele é usado porque você não pode usar literais em nenhuma expressão, incluindo `KeyConditionExpression`. Você pode usar o valor de atributo de expressão `:yyyy` para resolver essa questão.

2. Para executar o programa, digite o comando a seguir.

```
python MoviesScan.py
```

Note

Também é possível usar a operação `Scan` com quaisquer índices secundários criados na tabela. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 5: (Opcional) Excluir a tabela

Para excluir a tabela `Movies`:

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesDeleteTable.py`.

```
import boto3

def delete_movie_table(dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")

    table = dynamodb.Table('Movies')
    table.delete()

if __name__ == '__main__':
    delete_movie_table()
    print("Movies table deleted.")
```

2. Para executar o programa, digite o comando a seguir.

```
python MoviesDeleteTable.py
```

Summary

Neste tutorial, você criou o `Movies` na versão para download do Amazon DynamoDB no computador e realizou operações básicas. A versão para download do DynamoDB é útil durante o desenvolvimento e o teste de aplicativos. No entanto, quando você estiver pronto para executar o aplicativo em um ambiente de produção, precisará modificar o código para que ele use o serviço web do DynamoDB.

Modificação do código para usar o serviço DynamoDB

Para usar o serviço web do DynamoDB, altere o endpoint em seu aplicativo. Para fazer isso, modifique a seguinte linha.

```
dynamodb = boto3.resource('dynamodb', endpoint_url="http://localhost:8000")
```

Por exemplo, se você deseja usar a região `us-west-2`, altere o código para o seguinte.

```
dynamodb = boto3.resource('dynamodb', region_name='us-west-2')
```

Em vez de usar a versão para download do DynamoDB no computador, o programa agora usa o web service do DynamoDB na região oeste dos EUA (Oregon).

O DynamoDB está disponível em AWS Regiões do mundo todo. Para obter a lista completa, consulte [Regiões e endpoints](#) no [AWS Referência geral](#). Para obter mais informações sobre como definir regiões e endpoints no seu código, consulte [AWS Seleção de região](#) no [AWS SDK for Java Guia do desenvolvedor](#).

Ruby e DynamoDB

Neste tutorial, você usa o AWS SDK for Ruby para escrever programas simples a fim de realizar as seguintes operações do Amazon DynamoDB:

- Criar uma tabela chamada `Movies` e carregar dados de amostra no formato JSON.
- Realizar operações `create`, `read`, `update` e `delete` na tabela.
- Executar consultas simples.

À medida que você trabalha com este tutorial, consulte [AWS SDK for Ruby Referência de API](#). O [Seção do DynamoDB](#) descreve os parâmetros e resultados das operações do DynamoDB.

Pré-requisitos do tutorial

- Faça download e execute o DynamoDB no seu computador. Para mais informações, consulte [Configuração do DynamoDB Local \(versão disponível para download\) \(p. 50\)](#).

Note

Você pode usar as versões disponíveis para download do DynamoDB neste tutorial. Para obter informações sobre como executar o mesmo código para o serviço web do DynamoDB, consulte [o Summary \(p. 210\)](#).

- Configurar umAWSChave de acesso para usar aAWSSDKs. Para mais informações, consulte [Configuração do DynamoDB \(serviço Web\) \(p. 57\)](#).
- Configurar o AWS SDK for Ruby:
 - [InstallRuby](#).
 - [Instale o AWS SDK for Ruby.](#)

Para obter mais informações, consulte[Instalação](#)noAWS SDK for RubyReferência de API do.

Etapa 1: Criar uma tabela

Nesta etapa, você cria uma tabela chamadaMoviesNo Amazon DynamoDB. A chave primária da tabela é composta dos dois atributos a seguir:

- year– A chave de partição. O attribute_type é N para número.
- title– A chave de classificação. O attribute_type é S para string.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesCreateTable.rb`.

```
require 'aws-sdk-dynamodb'

def create_table(dynamodb_client, table_definition)
    response = dynamodb_client.create_table(table_definition)
    response.table_description.table_status
rescue StandardError => e
    puts "Error creating table: #{e.message}"
    'Error'
end

def run_me
    region = 'us-west-2'

    # To use the downloadable version of Amazon DynamoDB,
    # uncomment the endpoint statement.
    Aws.config.update(
        # endpoint: 'http://localhost:8000',
        region: region
    )

    dynamodb_client = Aws::DynamoDB::Client.new

    table_definition = {
        table_name: 'Movies',
        key_schema: [
            {
                attribute_name: 'year',
                key_type: 'HASH' # Partition key.
            },
            {
                attribute_name: 'title',
                key_type: 'RANGE' # Sort key.
            }
        ],
        attribute_definitions: [
            {
                attribute_name: 'year',
                attribute_type: 'N'
            },
            {
                attribute_name: 'title',
                attribute_type: 'S'
            }
        ]
    }
    response = dynamodb_client.create_table(table_definition)
    response.table_description.table_status
end
```

```
        attribute_type: 'S'
    }
],
provisioned_throughput: {
    read_capacity_units: 10,
    write_capacity_units: 10
}
}

puts "Creating the table named 'Movies'..."
create_table_result = create_table(dynamodb_client, table_definition)

if create_table_result == 'Error'
    puts 'Table not created.'
else
    puts "Table created with status '#{create_table_result}'."
end

run_me if $PROGRAM_NAME == __FILE__
```

Note

- Você define o endpoint para indicar que está criando a tabela na versão para download do Amazon DynamoDB no computador.
 - Na chamada `create_table`, você especifica o nome da tabela, os atributos da chave primária e seus tipos de dados.
 - O parâmetro `provisioned_throughput` é obrigatório. No entanto, a versão disponível para download do DynamoDB a ignora. (O throughput provisionado está além do escopo deste exercício.)

2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesCreateTable.rb
```

Para saber mais sobre como gerenciar tabelas, consulte [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#).

Etapa 2: Carregar dados de exemplo

Nesta etapa, você preenche o movies no Amazon DynamoDB com dados de amostra.

Tópicos

- Etapa 2.1: Faça download do arquivo de dados de exemplo (p. 195)
 - Etapa 2.2: Carregar os dados de exemplo para a tabela de filmes (p. 195)

Você usa um arquivo de dados de exemplo que contém informações sobre milhares de filmes do Internet Movie Database (IMDb). Os dados de filme estão no formato JSON, conforme mostrado no exemplo a seguir. Para cada filme, existe um `year`, um `title` e um mapa JSON chamado `info`.

```
[  
  {  
    "year" : ... ,  
    "title" : ... ,  
    "info" : { ... }  
  },  
  {  
    "year" : ... ,
```

```
        "title" : ....,
        "info" : { ... }
    },
    ...
]
```

Nos dados JSON, observe o seguinte:

- Usamos `year` e `title` como os valores de atributo de chave primária da tabela `Movies`.
- Você deve armazenar o resto dos valores `info` em um único atributo chamado `info`. Este programa ilustra como você pode armazenar o JSON em um atributo do DynamoDB.

Veja a seguir um exemplo de dados de filme.

```
{
    "year" : 2013,
    "title" : "Turn It Down, Or Else!",
    "info" : {
        "directors" : [
            "Alice Smith",
            "Bob Jones"
        ],
        "release_date" : "2013-01-18T00:00:00Z",
        "rating" : 6.2,
        "genres" : [
            "Comedy",
            "Drama"
        ],
        "image_url" : "http://ia.media-imdb.com/images/N/
09ERWAU7FS797AJ7LU8HN09AMUP908RLlo5JF90EWR7LJKQ7@._V1_SX400_.jpg",
        "plot" : "A rock band plays their music at high volumes, annoying the neighbors.",
        "rank" : 11,
        "running_time_secs" : 5215,
        "actors" : [
            "David Matthewman",
            "Ann Thomas",
            "Jonathan G. Neff"
        ]
    }
}
```

Etapa 2.1: Faça download do arquivo de dados de exemplo

1. Faça download do arquivo de dados de exemplo: [moviedata.zip](#)
2. Extraia o arquivo de dados (`moviedata.json`) do arquivo.
3. Copie o `moviedata.json` para o seu diretório atual.

Etapa 2.2: Carregar os dados de exemplo para a tabela de filmes

Depois de fazer download dos dados de exemplo, execute o programa a seguir para preencher a `MoviesTabela` INTO

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesLoadData.rb`.

```
require 'aws-sdk-dynamodb'
```

```
require 'json'

$movie_counter = 0
$total_movies = 0

def add_item_to_table(dynamodb_client, table_item)
    dynamodb_client.put_item(table_item)
    $movie_counter += 1
    puts "Added movie #{$movie_counter}/#{$total_movies}: " \
        "'#{table_item[:item]['title']}'" \
        "('#{table_item[:item]['year']}')."
rescue StandardError => e
    puts "Error adding movie '#{table_item[:item]['title']}' " \
        "('#{table_item[:item]['year']}'): #{e.message}"
    puts 'Program stopped.'
    exit 1
end

def run_me
    region = 'us-west-2'
    table_name = 'Movies'
    data_file = 'moviedata.json'

    # To use the downloadable version of Amazon DynamoDB,
    # uncomment the endpoint statement.
    Aws.config.update(
        # endpoint: 'http://localhost:8000',
        region: region
    )

    dynamodb_client = Aws::DynamoDB::Client.new
    file = File.read(data_file)
    movies = JSON.parse(file)
    $total_movies = movies.count

    puts "Adding #{$total_movies} movies from file '#{data_file}' " \
        "into table '#{table_name}'..."

    movies.each do |movie|
        table_item = {
            table_name: table_name,
            item: movie
        }
        add_item_to_table(dynamodb_client, table_item)
    end

    puts 'Done.'
end

run_me if $PROGRAM_NAME == __FILE__
```

2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesLoadData.rb
```

Etapa 3: Criar, ler, atualizar e excluir um item

Nesta etapa, você realiza operações de leitura e gravação em um item noMoviesTabela INTO DO Amazon DynamoDB.

Para saber mais sobre leitura e gravação de dados, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Tópicos

- [Etapa 3.1: Criar um novo item \(p. 197\)](#)
- [Etapa 3.2: Ler um item \(p. 198\)](#)
- [Etapa 3.3: Atualização de um item \(p. 199\)](#)
- [Etapa 3.4: Incrementar um contador atômico \(p. 201\)](#)
- [Etapa 3.5: Atualização de um item \(condicionalmente\) \(p. 202\)](#)
- [Etapa 3.6: Exclusão de um item \(p. 203\)](#)

Etapa 3.1: Criar um novo item

Nesta etapa, você adiciona um novo item à tabela .

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps01.rb`.

```
require 'aws-sdk-dynamodb'

def add_item_to_table(dynamodb_client, table_item)
  dynamodb_client.put_item(table_item)
  puts "Added movie '#{table_item[:item][:title]} " \
    "#(#{table_item[:item][:year]})'."
rescue StandardError => e
  puts "Error adding movie '#{table_item[:item][:title]} " \
    "#(#{table_item[:item][:year]})': #{e.message}"
end

def run_me
  region = 'us-west-2'
  table_name = 'Movies'
  title = 'The Big New Movie'
  year = 2015

  # To use the downloadable version of Amazon DynamoDB,
  # uncomment the endpoint statement.
  Aws.config.update(
    # endpoint: 'http://localhost:8000',
    region: region
  )

  dynamodb_client = Aws::DynamoDB::Client.new

  item = {
    year: year,
    title: title,
    info: {
      plot: 'Nothing happens at all.',
      rating: 0
    }
  }

  table_item = {
    table_name: table_name,
    item: item
  }

  puts "Adding movie '#{item[:title]} (#(item[:year]))' " \
    "to table '#{table_name}'..."
  add_item_to_table(dynamodb_client, table_item)
end

run_me if $PROGRAM_NAME == __FILE__
```

Note

A chave primária é obrigatória. Este código adiciona um item que tem chave primária (`year`, `title`) e atributos `info`. O atributo `info` armazena um mapa que fornece mais informações sobre o filme.

2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesItemOps01.rb
```

Etapa 3.2: Ler um item

No programa anterior, você adicionou os seguintes itens à tabela.

```
{  
    year: 2015,  
    title: "The Big New Movie",  
    info: {  
        plot: "Nothing happens at all.",  
        rating: 0  
    }  
}
```

Você pode usar o método `get_item` para ler o item da tabela `Movies`. Você deve especificar os valores de chave primária para que possa ler qualquer item de `Movies`, caso saiba seu `year` e `title`.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps02.rb`.

```
require 'aws-sdk-dynamodb'  
  
def get_item_from_table(dynamodb_client, table_item)  
    result = dynamodb_client.get_item(table_item)  
    puts "#{result.item['title']} (#{{result.item['year'].to_i}}):"  
    puts " Plot: #{result.item['info']['plot']}"  
    puts " Rating: #{result.item['info']['rating'].to_i}"  
rescue StandardError => e  
    puts "Error getting movie '#{table_item[:key][:title]} " \  
        "(#{table_item[:key][:year]}): #{e.message}"  
end  
  
def run_me  
    region = 'us-west-2'  
    table_name = 'Movies'  
    title = 'The Big New Movie'  
    year = 2015  
  
    # To use the downloadable version of Amazon DynamoDB,  
    # uncomment the endpoint statement.  
    Aws.config.update(  
        # endpoint: 'http://localhost:8000',  
        region: region  
    )  
  
    dynamodb_client = Aws::DynamoDB::Client.new  
  
    table_item = {  
        table_name: table_name,  
        key: {  
            year: year,  
            title: title  
        }  
    }
```

```
}

    puts "Getting information about '#{title} (#{year})' " \
        "from table '#{table_name}'..."
    get_item_from_table(dynamodb_client, table_item)
end

run_me if $PROGRAM_NAME == __FILE__
```

2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesItemOps02.rb
```

Etapa 3.3: Atualização de um item

Você pode usar o método `update_item` para modificar um item existente. Você pode atualizar valores de atributos existentes, adicionar novos atributos ou remover atributos.

Neste exemplo, você executa as seguintes atualizações:

- Altere o valor dos atributos existentes (`rating`, `plot`).
- Adicione um novo atributo de lista (`actors`) ao mapa `info` existente.

Veja a seguir o item existente.

```
{
  year: 2015,
  title: "The Big New Movie",
  info: {
    plot: "Nothing happens at all.",
    rating: 0
  }
}
```

O item é atualizado conforme o seguinte.

```
{
  year: 2015,
  title: "The Big New Movie",
  info: {
    plot: "Everything happens all at once.",
    rating: 5.5,
    actors: ["Larry", "Moe", "Curly"]
  }
}
```

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps03.rb`.

```
require 'aws-sdk-dynamodb'

def table_item_updated?(dynamodb_client, table_item)
  response = dynamodb_client.update_item(table_item)
  puts "Table item updated with the following attributes for 'info':"
  response.attributes['info'].each do |key, value|
    if key == 'rating'
      puts "#{key}: #{value.to_f}"
    else
      puts "#{key}: #{value}"
    end
  end
end
```

```
    end
  end
  true
rescue StandardError => e
  puts "Error updating item: #{e.message}"
  false
end

def run_me
  region = 'us-west-2'
  table_name = 'Movies'
  title = 'The Big New Movie'
  year = 2015

  # To use the downloadable version of Amazon DynamoDB,
  # uncomment the endpoint statement.
  Aws.config.update(
    # endpoint: 'http://localhost:8000',
    region: region
  )

  dynamodb_client = Aws::DynamoDB::Client.new

  table_item = {
    table_name: table_name,
    key: {
      year: year,
      title: title
    },
    update_expression: 'SET info.rating = :r, info.plot = :p, info.actors = :a',
    expression_attribute_values: {
      ':r': 5.5,
      ':p': 'Everything happens all at once.',
      ':a': [ 'Larry', 'Moe', 'Curly' ]
    },
    return_values: 'UPDATED_NEW'
  }

  puts "Updating table '#{table_name}' with information about " \
    "'#{title} (#{year})'..."

  if table_item_updated?(dynamodb_client, table_item)
    puts 'Table updated.'
  else
    puts 'Table not updated.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Note

Esse programa usa `update_expression` para descrever todas as atualizações que você deseja realizar no item especificado.

O `return_values` instrui o Amazon DynamoDB a retornar apenas os atributos atualizados (`UPDATED_NEW`).

2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesItemOps03.rb
```

Etapa 3.4: Incrementar um contador atômico

O DynamoDB oferece suporte a contadores atômicos, que usam `update_item` para aumentar ou diminuir o valor de um atributo existente sem interferir em outras solicitações de gravação. (Todas as solicitações de gravação são aplicadas na ordem em que são recebidas.)

O programa a seguir mostra como aumentar a `rating` de um filme. Toda vez que o programa for executado, o atributo é incrementado uma vez.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps04.rb`.

```
require 'aws-sdk-dynamodb'

def table_item_updated?(dynamodb_client, table_item)
    result = dynamodb_client.update_item(table_item)
    puts "Table item updated with the following attributes for 'info':"
    result.attributes['info'].each do |key, value|
        if key == 'rating'
            puts "#{key}: #{value.to_f}"
        else
            puts "#{key}: #{value}"
        end
    end
    true
rescue StandardError => e
    puts "Error updating item: #{e.message}"
    false
end

def run_me
    region = 'us-west-2'
    table_name = 'Movies'
    title = 'The Big New Movie'
    year = 2015

    # To use the downloadable version of Amazon DynamoDB,
    # uncomment the endpoint statement.
    Aws.config.update(
        # endpoint: 'http://localhost:8000',
        region: region
    )

    dynamodb_client = Aws::DynamoDB::Client.new

    table_item = {
        table_name: table_name,
        key: {
            year: year,
            title: title
        },
        update_expression: 'SET info.rating = info.rating + :val',
        expression_attribute_values: {
            ':val': 1
        },
        return_values: 'UPDATED_NEW'
    }

    puts "Updating table '#{table_name}' with information about " \
        "'#{title} (#{$year})'..."

    if table_item_updated?(dynamodb_client, table_item)
        puts 'Table updated.'
    else
        puts 'Table not updated.'
    end
end
```

```
    end
end

run_me if $PROGRAM_NAME == __FILE__
```

2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesItemOps04.rb
```

Etapa 3.5: Atualização de um item (condicionalmente)

O programa a seguir mostra como usar `update_item` com uma condição. Se a condição for verdadeira, a atualização será bem-sucedida; caso contrário, a atualização não será realizada.

Neste caso, o item será atualizado somente se o número de atores for maior que três.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps05.rb`.

```
require 'aws-sdk-dynamodb'

def table_item_updated?(dynamodb_client, table_item)
  result = dynamodb_client.update_item(table_item)
  puts "Table item updated with the following attributes for 'info':"
  result.attributes['info'].each do |key, value|
    if key == 'rating'
      puts "#{key}: #{value.to_f}"
    else
      puts "#{key}: #{value}"
    end
  end
  true
rescue StandardError => e
  puts "Error updating item: #{e.message}"
  false
end

def run_me
  region = 'us-west-2'
  table_name = 'Movies'
  title = 'The Big New Movie'
  year = 2015

  # To use the downloadable version of Amazon DynamoDB,
  # uncomment the endpoint statement.
  Aws.config.update(
    # endpoint: 'http://localhost:8000',
    region: region
  )

  dynamodb_client = Aws::DynamoDB::Client.new

  table_item = {
    table_name: table_name,
    key: {
      year: year,
      title: title
    },
    update_expression: 'REMOVE info.actors[0]',
    condition_expression: 'size(info.actors) > :num',
    expression_attribute_values: {
      ':num': 3
    },
    return_values: 'UPDATED_NEW'
  }
end
```

```
}

puts "Updating table '#{table_name}' with information about " \
  "'#{title} (#{year})'..."

if table_item_updated?(dynamodb_client, table_item)
  puts 'Table updated.'
else
  puts 'Table not updated.'
end
end

run_me if $PROGRAM_NAME == __FILE__
```

2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesItemOps05.rb
```

O programa deve falhar com a mensagem a seguir.

A solicitação condicional falhou

O programa falha porque o filme tem três atores, mas a condição está em busca de mais de três atores.

3. Modifique o programa para que `ConditionExpression` fique semelhante ao seguinte.

```
condition_expression: "size(info.actors) >= :num",
```

A condição agora é maior ou igual a 3 em vez de maior que 3.

4. Execute o programa novamente. O método `update_item` agora deve ser bem-sucedido.

Etapa 3.6: Exclusão de um item

Você pode usar o método `delete_item` para excluir um item, especificando sua chave primária. Se desejar, você pode fornecer uma `condition_expression` para evitar a exclusão do item, se a condição não for atendida.

No exemplo a seguir, você tenta excluir um item de filme específico se a sua classificação for 5 ou menos.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesItemOps06.rb`.

```
require 'aws-sdk-dynamodb'

def table_item_deleted?(dynamodb_client, table_item)
  dynamodb_client.delete_item(table_item)
  true
rescue StandardError => e
  puts "Error deleting item: #{e.message}"
  false
end

def run_me
  region = 'us-west-2'
  table_name = 'Movies'
  title = 'The Big New Movie'
  year = 2015

  # To use the downloadable version of Amazon DynamoDB,
  # uncomment the endpoint statement.
  Aws.config.update(
```

```
# endpoint: 'http://localhost:8000',
region: region
)

dynamodb_client = Aws::DynamoDB::Client.new

table_item = {
  table_name: table_name,
  key: {
    year: year,
    title: title
  },
  condition_expression: 'info.rating <= :val',
  expression_attribute_values: {
    ':val' => 5
  }
}

puts "Deleting item from table '#{table_name}' matching \" \
  '#{title} (#{{year}})' if specified criteria are met..."

if table_item_deleted?(dynamodb_client, table_item)
  puts 'Item deleted.'
else
  puts 'Item not deleted.'
end
end

run_me if $PROGRAM_NAME == __FILE__
```

2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesItemOps06.rb
```

O programa deve falhar com a mensagem a seguir.

A solicitação condicional falhou

O programa falha porque a classificação desse filme específico é maior que 5.

3. Modifique o programa para remover a condição.

```
params = {
  table_name: "Movies",
  key: {
    year: year,
    title: title
  }
}
```

4. Execute o programa. Agora, a exclusão é bem-sucedida, pois você removeu a condição.

Etapa 4: Consultar e verificar os dados

Você pode usar o método `query` para recuperar os dados de uma tabela. Você deve especificar o valor de uma chave de partição. A chave de classificação é opcional.

A chave primária da tabela `Movies` é composta pelo seguinte:

- `year`- A chave de partição. O tipo de atributo é `number`.
- `title`- A chave de classificação. O tipo de atributo é `string`.

Para encontrar todos os filmes lançados durante um ano, você precisa especificar somente o `year`. Você também pode fornecer o `title` para recuperar um subconjunto de filmes baseados na mesma condição (na chave de classificação). Por exemplo, é possível encontrar filmes lançados em 2014 que têm um título começando com a letra "A".

Além do método `query`, é possível usar o método `scan` para recuperar todos os dados da tabela.

Para saber mais sobre como consultar e verificar dados, consulte [Como trabalhar com consultas no DynamoDB \(p. 490\)](#) e [Como trabalhar com verificações no DynamoDB \(p. 508\)](#), respectivamente.

Tópicos

- [Etapa 4.1: Consulta — Todos os filmes lançados em um ano \(p. 205\)](#)
- [Etapa 4.2: Consulta — Todos os filmes lançados em um ano com determinados títulos \(p. 206\)](#)
- [Etapa 4.3: Scan \(p. 208\)](#)

Etapa 4.1: Consulta — Todos os filmes lançados em um ano

O programa a seguir recupera todos os filmes lançados no `year` de 1985.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesQuery01.rb`.

```
require 'aws-sdk-dynamodb'

def query_for_items_from_table(dynamodb_client, query_condition)
    # To display the elapsed time for the query operation,
    # uncomment the following three comments.
    # start = Time.now
    result = dynamodb_client.query(query_condition)
    # finish = Time.now
    # puts "Search took #{finish - start} seconds."
    if result.items.count.zero?
        puts 'No matching movies found.'
    else
        puts "Found #{result.items.count} matching movies:"
        result.items.each do |movie|
            puts "#{movie['title']} (#{movie['year'].to_i})"
        end
    end
rescue StandardError => e
    puts "Error querying for table items: #{e.message}"
end

def run_me
    region = 'us-west-2'
    table_name = 'Movies'
    year = 1985

    # To use the downloadable version of Amazon DynamoDB,
    # uncomment the endpoint statement.
    Aws.config.update(
        # endpoint: 'http://localhost:8000',
        region: region
    )

    dynamodb_client = Aws::DynamoDB::Client.new

    # To query on the 'title' range/sort key in addition to the 'year'
    # hash/partition key, uncomment the following three 'title' comments.
    query_condition = {
        table_name: table_name,
```

```
key_condition_expression: '#yr = :yyyy', # '#yr = :yyyy AND #t = :title'
expression_attribute_names: {
    '#t' => 'title',
    '#yr' => 'year'
},
expression_attribute_values: {
    ':title' => 'After Hours',
    ':yyyy' => year
}
}

puts "Searching for items in the '#{table_name}' table from '#{year}'..."

query_for_items_from_table(dynamodb_client, query_condition)
end

run_me if $PROGRAM_NAME == __FILE__
```

Note

- `expression_attribute_names` fornece substituição de nome. Usamos isso porque `year` é uma palavra reservada no Amazon DynamoDB. Não é possível usá-la diretamente em nenhuma expressão, incluindo `KeyConditionExpression`. Use o nome de atributo de expressão `#yr` para resolver essa questão.
- `expression_attribute_values` fornece substituição de valor. Ele é usado porque você não pode usar literais em nenhuma expressão, incluindo `key_condition_expression`. Você pode usar o valor de atributo de expressão `:yyyy` para resolver essa questão.

2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesItemQuery01.rb
```

Note

O programa anterior mostra como consultar uma tabela por seus atributos de chave primária. No DynamoDB, se desejar, é possível criar um ou mais índices secundários em uma tabela e consultá-los da mesma forma que consulta uma tabela. Os índices secundários oferecem aos seus aplicativos mais flexibilidade ao permitir consultas nos atributos que não são chave. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 4.2: Consulta — Todos os filmes lançados em um ano com determinados títulos

O programa a seguir recupera todos os filmes lançados no `year=1992` com `title` começando com a letra “A” até a letra “L”.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesQuery02.rb`.

```
require 'aws-sdk-dynamodb'

def query_for_items_from_table(dynamodb_client, query_condition)
    # To display the elapsed time for the query operation,
    # uncomment the following three comments.
    # start = Time.now
    result = dynamodb_client.query(query_condition)
    # finish = Time.now
    # puts "Search took #{finish - start} seconds."
    if result.items.count.zero?
```

```
    puts 'No matching movies found.'
else
  puts "Found #{result.items.count} matching movies:"
  result.items.each do |movie|
    puts "#{movie['title']} (#{movie['year'].to_i}):"
    if movie['info'].key?('genres') && movie['info']['genres'].count.positive?
      puts '  Genres:'
      movie['info']['genres'].each do |genre|
        puts "    #{genre}"
      end
    end
    if movie['info'].key?('actors') && movie['info']['actors'].count.positive?
      puts '  Actors:'
      movie['info']['actors'].each do |actor|
        puts "    #{actor}"
      end
    end
  end
rescue StandardError => e
  puts "Error querying for table items: #{e.message}"
end

def run_me
  region = 'us-west-2'
  table_name = 'Movies'
  year = 1982
  letter1 = 'A'
  letter2 = 'L'

  # To use the downloadable version of Amazon DynamoDB,
  # uncomment the endpoint statement.
  Aws.config.update(
    # endpoint: 'http://localhost:8000',
    region: region
  )

  dynamodb_client = Aws::DynamoDB::Client.new

  query_condition = {
    table_name: table_name,
    projection_expression: '#yr, title, info.genres, info.actors[0]',
    key_condition_expression: '#yr = :yyyy AND title BETWEEN :letter1 AND :letter2',
    expression_attribute_names: { '#yr' => 'year' },
    expression_attribute_values: {
      ':yyyy' => year,
      ':letter1' => letter1,
      ':letter2' => letter2
    }
  }

  puts "Searching for items in the '#{table_name}' table from '#{year}' and "
  puts "titles starting with the letters '#{letter1}' through '#{letter2}'..."

  query_for_items_from_table(dynamodb_client, query_condition)
end

run_me if $PROGRAM_NAME == __FILE__
```

2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesQuery02.rb
```

Etapa 4.3: Scan

O método `scan` lê todos os itens da tabela inteira e retorna todos os dados nela contidos. Você pode fornecer uma `filter_expression` opcional, para que apenas os itens que correspondem aos seus critérios sejam retornados. No entanto, observe que o filtro só é aplicado depois que toda a tabela é examinada.

O programa a seguir examina o `Movies`, que contém aproximadamente 5.000 itens. A verificação especifica o filtro opcional para recuperar somente os filmes da década de 1950 (aproximadamente 100 itens) e descartar todos os outros.

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesScan.rb`.

```
require 'aws-sdk-dynamodb'

def scan_for_items_from_table(dynamodb_client, scan_condition)
    # To display the elapsed time for the query operation,
    # uncomment the following three comments.
    #start = Time.now
    loop do
        result = dynamodb_client.scan(scan_condition)

        if result.items.count.zero?
            puts 'No matching movies found (yet)...'
        else
            puts "Found #{result.items.count} matching movies (so far):"
            result.items.each do |movie|
                puts "#{movie["title"]} (#{movie["year"].to_i}), " \
                    "Rating: #{movie["info"]["rating"].to_f}"
            end

            break if result.last_evaluated_key.nil?

            puts "Searching for more movies..."
            scan_condition[:exclusive_start_key] = result.last_evaluated_key
        end
    end
    puts 'Finished searching.'
    # finish = Time.now
    # puts "Search took #{finish - start} seconds."
rescue StandardError => e
    puts "Error scanning for table items: #{e.message}"
end

def run_me
    region = 'us-west-2'
    table_name = 'Movies'
    start_year = 1950
    end_year = 1959

    # To use the downloadable version of Amazon DynamoDB,
    # uncomment the endpoint statement.
    Aws.config.update(
        # endpoint: 'http://localhost:8000',
        region: region
    )

    dynamodb_client = Aws::DynamoDB::Client.new

    scan_condition = {
        table_name: table_name,
        projection_expression: '#yr, title, info.rating',
        filter_expression: '#yr between :start_yr and :end_yr',
        limit: 100
    }
    result = dynamodb_client.scan(scan_condition)
    puts "Scanned #{result.items.count} items from the table."
end
```

```
    expression_attribute_names: { '#yr' => 'year' },
    expression_attribute_values: {
        ':start_yr' => start_year,
        ':end_yr' => end_year
    }
}

puts "Searching for items in the '#{table_name}' table from #{start_year} "
      "through #{end_year}..."

scan_for_items_from_table(dynamodb_client, scan_condition)
end

run_me if $PROGRAM_NAME == __FILE__
```

No código, observe o seguinte:

- `projection_expression` especifica os atributos que você deseja no resultado da verificação.
 - `filter_expression` especifica uma condição que retorna apenas os itens que satisfazem à condição. Todos os outros itens são descartados.
2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesScan.rb
```

Note

Você também pode usar `oscan` com quaisquer índices secundários criados na tabela. Para mais informações, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Etapa 5: (Opcional) Excluir a tabela

Siga estas etapas para excluir a `Movies` Tabela INTO

1. Copie e cole o programa a seguir em um arquivo chamado `MoviesDeleteTable.rb`.

```
require 'aws-sdk-dynamodb'

def table_deleted?(dynamodb_client, table_name)
    dynamodb_client.delete_table(table_name: table_name)
    true
rescue StandardError => e
    puts "Error deleting table: #{e.message}"
    false
end

def run_me
    region = 'us-west-2'
    table_name = 'Movies'

    # To use the downloadable version of Amazon DynamoDB,
    # uncomment the endpoint statement.
    Aws.config.update(
        # endpoint: 'http://localhost:8000',
        region: region
    )

    dynamodb_client = Aws::DynamoDB::Client.new

    puts "Deleting table '#{table_name}'..."
```

```
if table_deleted?(dynamodb_client, table_name)
    puts 'Table deleted.'
else
    puts 'Table not deleted.'
end
end

run_me if $PROGRAM_NAME == __FILE__
```

2. Para executar o programa, digite o comando a seguir.

```
ruby MoviesDeleteTable.rb
```

Summary

Neste tutorial, você criou oMoviesNa versão para download do Amazon DynamoDB no computador e realizou operações básicas. A versão para download do DynamoDB é útil durante o desenvolvimento e o teste de aplicativos. No entanto, quando você estiver pronto para executar o aplicativo em um ambiente de produção, precisará modificar o código para que ele use o serviço web do DynamoDB.

Modificação do código para usar o serviço DynamoDB

Para usar o serviço do DynamoDB, você deve alterar o endpoint em seu aplicativo. Para fazer isso, localize as seguintes linhas no código.

```
Aws.config.update({
region: "us-west-2",
endpoint: "http://localhost:8000"
})
```

Remova o parâmetro endpoint para que o código fique semelhante ao seguinte.

```
Aws.config.update({
region: "us-west-2"
});
```

Depois que você remover essa linha, o código poderá acessar o serviço DynamoDB naAWSRegião especificada peloregionValor de configuração.

Em vez de usar a versão disponível do DynamoDB no seu computador, o programa agora usa o endpoint de serviço web do DynamoDB na região Oeste dos EUA (Oregon).

O DynamoDB está disponível emAWSRegiões do mundo todo. Para obter a lista completa, consulte[Regiões e endpoints do](#)[AWSReferência geral](#). Para obter mais informações, consulte o [AWS SDK for RubyGuia de conceitos básicos do](#).

Programação com o DynamoDB e o AWSSDKs da

Esta seção aborda tópicos relacionados a desenvolvedores. Se você deseja executar exemplos de código em vez disso, consulte [Executar os exemplos de código neste Guia do desenvolvedor \(p. 328\)](#).

Note

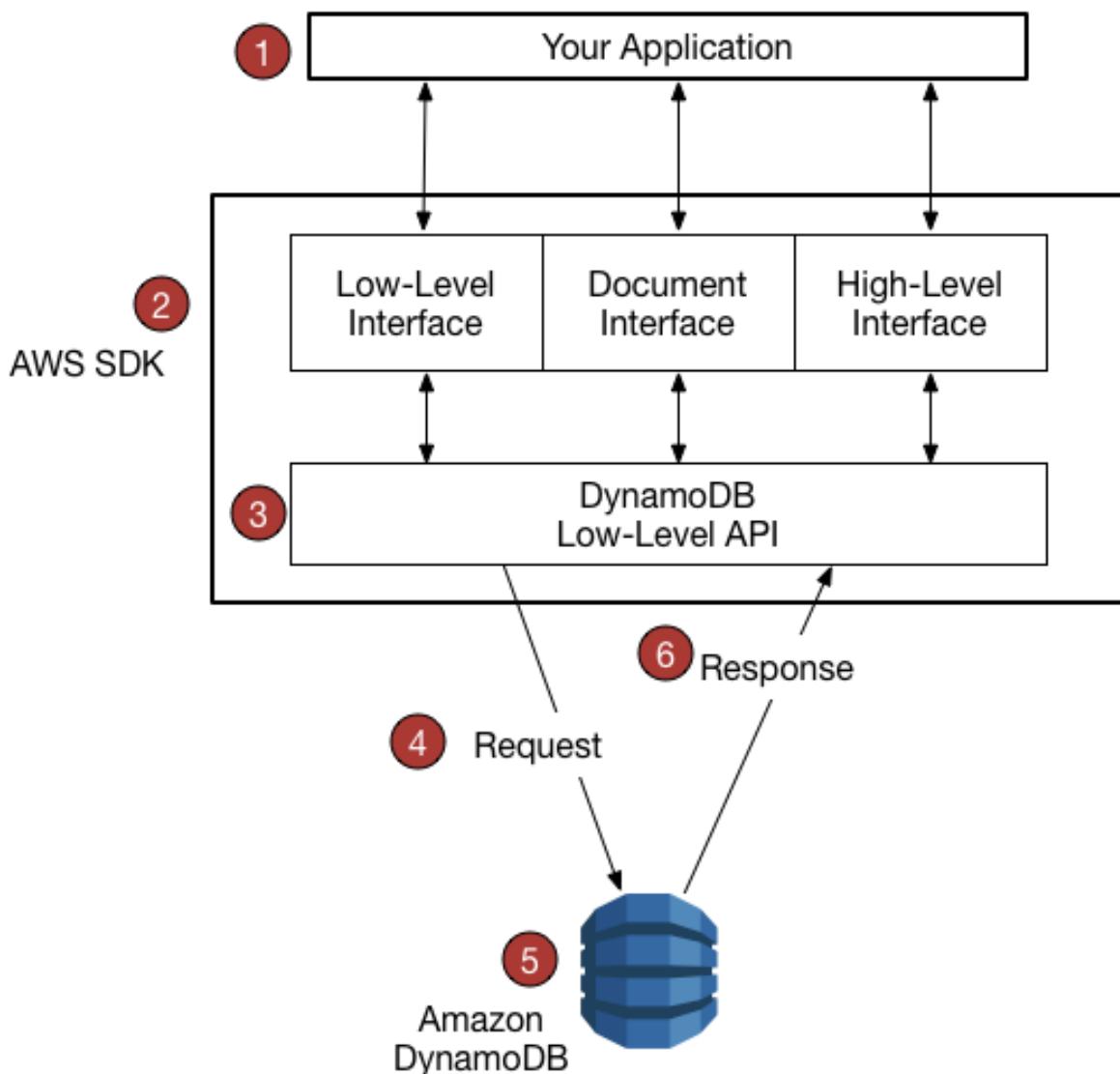
Em dezembro de 2017, o AWS iniciou o processo de migração de todos os endpoints do Amazon DynamoDB para usar certificados seguros emitidos pelo Amazon Trust Services (ATS). Para mais informações, consulte [Solução de problemas de estabelecimento de conexão SSL/TLS \(p. 1068\)](#).

Tópicos

- [Visão geral do AWSSupport do SDK da para o DynamoDB \(p. 211\)](#)
- [Interfaces programáticas \(p. 213\)](#)
- [API do DynamoDB de baixo nível \(p. 217\)](#)
- [Tratamento de erros com o DynamoDB \(p. 221\)](#)
- [Interfaces de programação de nível superior para o DynamoDB \(p. 227\)](#)
- [Executar os exemplos de código neste Guia do desenvolvedor \(p. 328\)](#)

Visão geral do AWSSupport do SDK da para o DynamoDB

O diagrama a seguir fornece uma visão geral de alto nível da programação de aplicativos do Amazon DynamoDB usando o AWSSDKs da.



1. Escreva um aplicativo usando um AWS SDK para a sua linguagem de programação.
2. Cada AWS SDK fornece uma ou mais interfaces programáticas para trabalhar com o DynamoDB. As interfaces específicas disponíveis dependem de qual linguagem de programação e AWS SDK que você usa.
3. O AWS SDK constrói solicitações HTTP (S) para uso com a API do DynamoDB de baixo nível.
4. O AWS SDK envia a solicitação ao endpoint do DynamoDB.
5. O DynamoDB executa a solicitação. Se a solicitação for bem-sucedida, o DynamoDB retornará um código de resposta HTTP 200 (OK). Se a solicitação não for bem-sucedida, o DynamoDB retornará um código de erro HTTP e uma mensagem de erro.
6. O AWS SDK processa a resposta e a propaga de volta ao seu aplicativo.

Cada um dos AWS SDKs fornecem serviços importantes ao seu aplicativo, incluindo os seguintes:

- Formatação de solicitações HTTP(S) e serialização de parâmetros de solicitação.
- Geração de uma assinatura criptográfica para cada solicitação.

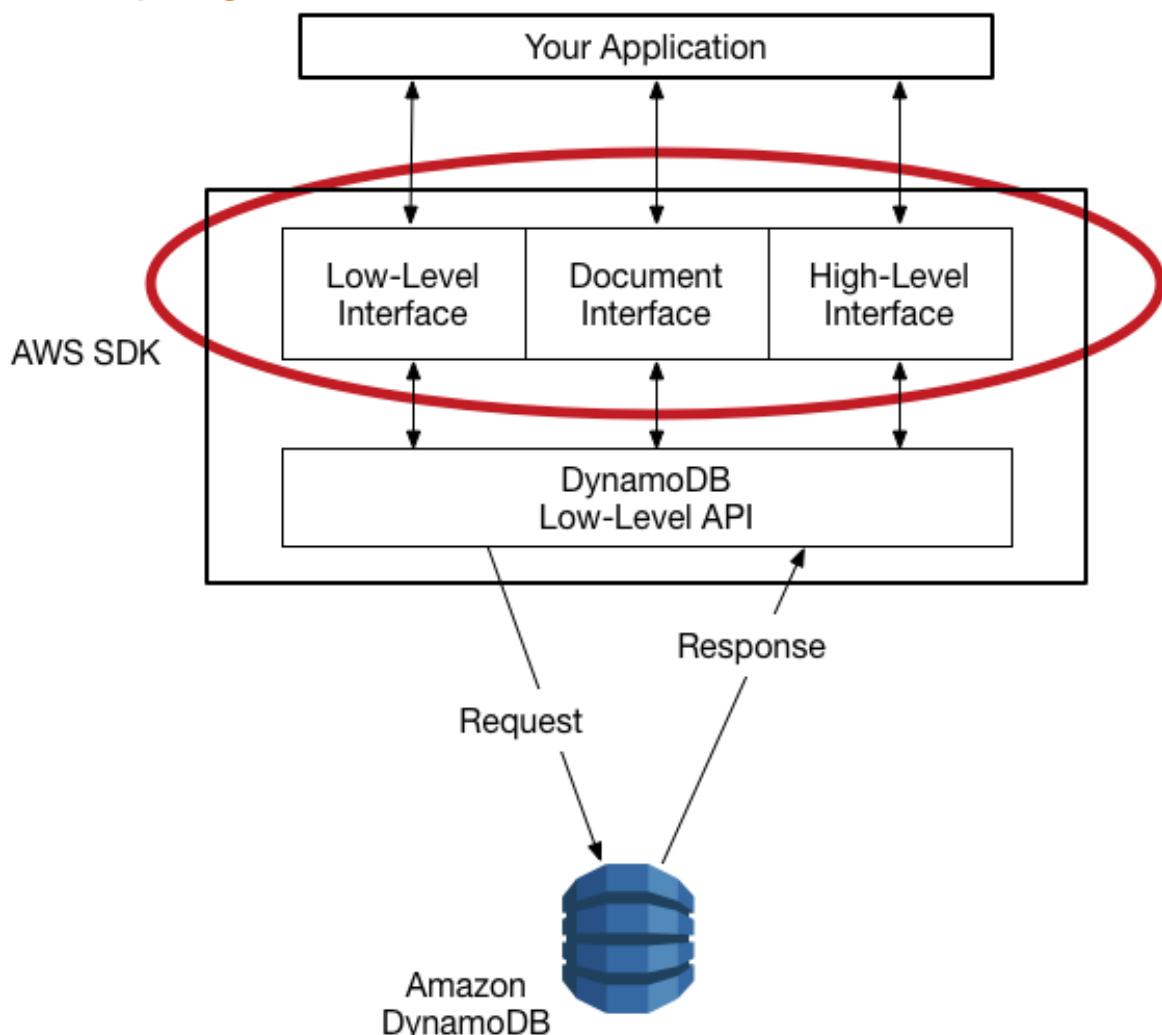
- Encaminhamento de solicitações a um endpoint do DynamoDB e recebimento de respostas do DynamoDB.
- Extração dos resultados dessas respostas.
- Implementação da lógica de novas tentativas básicas em caso de erros.

Não é necessário escrever código para qualquer uma dessas tarefas.

Note

Para obter mais informações sobre AWSSDKs, incluindo instruções de instalação e a documentação, consulte [Ferramentas para a Amazon Web Services](#).

Interfaces programáticas



Cada [AWSSDK](#) fornece uma ou mais interfaces programáticas para trabalhar com o Amazon DynamoDB. Essas interfaces variam de wrappers simples de baixo nível do DynamoDB a camadas de persistência orientadas a objetos. As interfaces disponíveis variam dependendo do [AWSSDK](#) e linguagem de programação que você usa.

A seguinte seção destaca algumas das interfaces disponíveis, usando o AWS SDK for JavaExemplo de da. (Nem todas as interfaces estão disponíveis em todos os AWSSDKs do.)

Tópicos

- [Interfaces de baixo nível \(p. 214\)](#)
- [Interfaces de documento \(p. 215\)](#)
- [Interface de persistência de objetos \(p. 215\)](#)

Interfaces de baixo nível

Cada idioma específico AWSO SDK fornece uma interface de baixo nível para o Amazon DynamoDB, com métodos que se assemelham a solicitações de API do DynamoDB de baixo nível.

Em alguns casos, você precisará identificar os tipos de dados dos atributos usando [Descritores de tipo de dados \(p. 220\)](#), como S para strings ou N para números.

Note

Uma interface de baixo nível está disponível em cada linguagem específica AWSSDKS DO.

O seguinte programa Java usa a interface de baixo nível do AWS SDK for Java. O programa emite umGetItemsolicitar uma música noMusicTabela e imprime o ano em que a canção foi lançada.

O `com.amazonaws.services.dynamodbv2.AmazonDynamoDB` implementa a interface de baixo nível do DynamoDB.

```
package com.amazonaws.codesamples;

import java.util.HashMap;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.GetItemRequest;
import com.amazonaws.services.dynamodbv2.model.GetItemResult;

public class MusicLowLevelDemo {

    public static void main(String[] args) {
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();

        HashMap<String, AttributeValue> key = new HashMap<String, AttributeValue>();
        key.put("Artist", new AttributeValue().withS("No One You Know"));
        key.put("SongTitle", new AttributeValue().withS("Call Me Today"));

        GetItemRequest request = new GetItemRequest()
            .withTableName("Music")
            .withKey(key);

        try {
            GetItemResult result = client.getItem(request);
            if (result && result.getItem() != null) {
                AttributeValue year = result.getItem().get("Year");
                System.out.println("The song was released in " + year.getN());
            } else {
                System.out.println("No matching song was found");
            }
        } catch (Exception e) {
```

```
        System.err.println("Unable to retrieve data: ");
        System.err.println(e.getMessage());
    }
}
```

Interfaces de documento

Muitos dos AWS SDKs fornecem uma interface de documento, permitindo que você realize operações de plano de dados (criar, ler, atualizar, excluir) em tabelas e índices. Com uma interface de documento, você não precisa especificar os [Descriptores de tipo de dados \(p. 220\)](#). Os tipos de dados estão implícitos pela semântica dos próprios dados. Estes AWS SDKs também fornecem métodos para converter facilmente documentos JSON em e a partir de tipo de dados nativos do Amazon DynamoDB.

Note

Interfaces de documento estão disponíveis no AWSSDKs para o [Java](#), [.NET](#), [Node.js](#), e [JavaScript no navegador](#).

O seguinte programa Java usa a interface de documento do AWS SDK for Java. O programa cria um `Table` que representa o `Music`, em seguida, pede a esse objeto para usar `GetItem` para recuperar uma música. Em seguida, o programa imprime o ano em que a canção foi lançada.

O `com.amazonaws.services.dynamodbv2.document.DynamoDB` implementa a interface de documento do DynamoDB. Observe como o DynamoDB atua como um wrapper em torno do cliente de baixo nível (`AmazonDynamoDB`).

```
package com.amazonaws.codesamples.gsg;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.GetItemOutcome;
import com.amazonaws.services.dynamodbv2.document.Table;

public class MusicDocumentDemo {

    public static void main(String[] args) {

        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
        DynamoDB docClient = new DynamoDB(client);

        Table table = docClient.getTable("Music");
        GetItemOutcome outcome = table.getItemOutcome(
            "Artist", "No One You Know",
            "SongTitle", "Call Me Today");

        int year = outcome.getItem().getInt("Year");
        System.out.println("The song was released in " + year);

    }
}
```

Interface de persistência de objetos

Alguns AWS SDKs fornecem uma interface de persistência de objetos em que você não realiza operações de plano de dados diretamente. Em vez disso, você cria objetos que representam itens em índices e tabelas do Amazon DynamoDB e interage apenas com esses objetos. Isso permite que você escreva um código centrado em objetos, em vez de um código centrado no banco de dados.

Note

Interfaces de persistência de objetos estão disponíveis no AWSSDKs para Java e .NET. Para mais informações, consulte [Interfaces de programação de nível superior para o DynamoDB \(p. 227\)](#).

O seguinte programa Java usa `DynamoDBMapper`, a interface de persistência de objetos do AWS SDK for Java. `MusicItem`classe representa um item no `MusicTable` do.

```
package com.amazonaws.codesamples;

import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBAttribute;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBRangeKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;

@DynamoDBTable(tableName="Music")
public class MusicItem {
    private String artist;
    private String songTitle;
    private String albumTitle;
    private int year;

    @DynamoDBHashKey(attributeName="Artist")
    public String getArtist() { return artist; }
    public void setArtist(String artist) {this.artist = artist; }

    @DynamoDBRangeKey(attributeName="SongTitle")
    public String getSongTitle() { return songTitle; }
    public void setSongTitle(String songTitle) {this.songTitle = songTitle; }

    @DynamoDBAttribute(attributeName = "AlbumTitle")
    public String getAlbumTitle() { return albumTitle; }
    public void setAlbumTitle(String albumTitle) {this.albumTitle = albumTitle; }

    @DynamoDBAttribute(attributeName = "Year")
    public int getYear() { return year; }
    public void setYear(int year) { this.year = year; }
}
```

Em seguida, você pode instanciar um `MusicItem` e recupere uma música usando o `load()` Método de `DynamoDBMapper`. Em seguida, o programa imprime o ano em que a canção foi lançada.

O `com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper` implementa a interface de persistência de objetos do DynamoDB. Observe como `DynamoDBMapper` atua como um wrapper em torno do cliente de baixo nível (`AmazonDynamoDB`).

```
package com.amazonaws.codesamples;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;

public class MusicMapperDemo {

    public static void main(String[] args) {
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
        DynamoDBMapper mapper = new DynamoDBMapper(client);
        MusicItem keySchema = new MusicItem();
```

```
keySchema.setArtist("No One You Know");
keySchema.setSongTitle("Call Me Today");

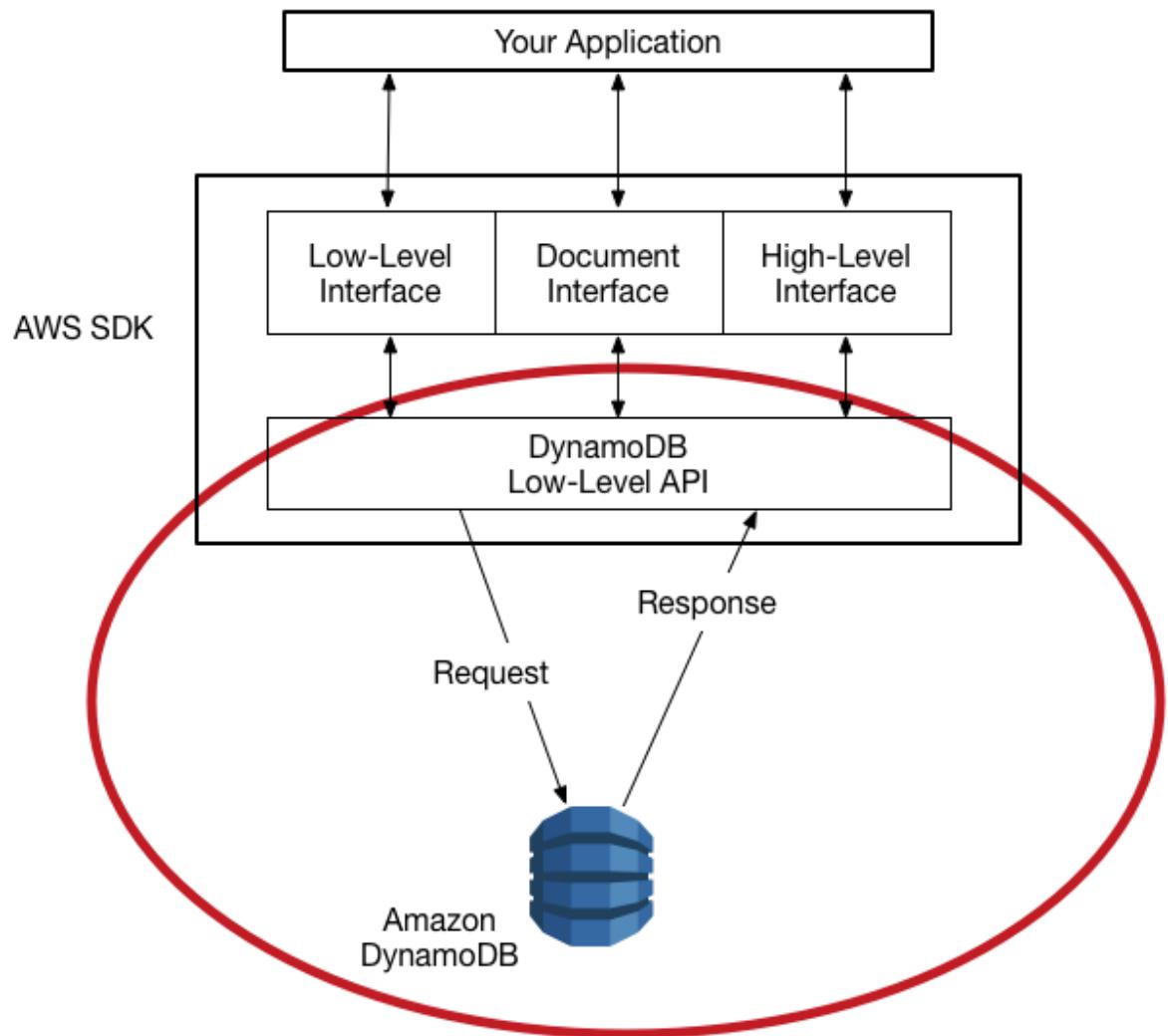
try {
    MusicItem result = mapper.load(keySchema);
    if (result != null) {
        System.out.println(
            "The song was released in " + result.getYear());
    } else {
        System.out.println("No matching song was found");
    }
} catch (Exception e) {
    System.err.println("Unable to retrieve data: ");
    System.err.println(e.getMessage());
}

}
```

API do DynamoDB de baixo nível

Tópicos

- [Formato de solicitação \(p. 219\)](#)
- [Formato de resposta \(p. 219\)](#)
- [Descritores de tipo de dados \(p. 220\)](#)
- [Dados numéricos \(p. 221\)](#)
- [Dados binários \(p. 221\)](#)



O Amazon DynamoDB API de baixo nível é a interface de nível de protocolo para o DynamoDB. Nesse nível, cada solicitação de HTTP(S) deve ser corretamente formatada e ter uma assinatura digital válida.

Os SDKs da AWS criam solicitações de API do DynamoDB de baixo nível em seu nome e processam as respostas do DynamoDB. Isso permite que você se concentre na lógica do seu aplicativo, em vez de detalhes de baixo nível. No entanto, você ainda pode se beneficiar de um conhecimento básico de como a API do DynamoDB de baixo nível funciona.

Para obter mais informações sobre a API do DynamoDB de baixo nível, consulte [Referência de API do Amazon DynamoDB](#).

Note

O DynamoDB Streams tem sua própria API de baixo nível, que é separada do DynamoDB e é totalmente compatível com os SDKs da AWS.

Para mais informações, consulte [Captura de dados de alteração para DynamoDB Streams \(p. 651\)](#). Para obter a API do DynamoDB Streams de baixo nível, consulte a [Referência de API do Amazon DynamoDB Streams](#).

A API do DynamoDB de baixo nível usa JavaScript Object Notation (JSON) como um formato de protocolo de conexão. O JSON apresenta dados em uma hierarquia de forma que os valores e a estrutura dos dados

sejam transmitidos simultaneamente. O pares de nome–valor são definidos no formato `name:value`. A hierarquia de dados é definida por colchetes aninhados de pares de nome–valor.

O DynamoDB usa JSON somente como um protocolo de transporte, não como um formato de armazenamento. OAWSOs SDKs usam JSON para enviar dados ao DynamoDB e o DynamoDB responde com JSON. O DynamoDB não armazena dados persistentemente no formato JSON.

Note

Para obter mais informações sobre JSON [Introdução ao JSON](#) site JSON.org.

Formato de solicitação

A API de baixo nível do DynamoDB aceita HTTP (S)POSTsolicitações como entrada. OAWSSDKs criam essas solicitações para você.

Vamos supor que você tenha uma tabela chamada `Pets`, com um esquema de chaves que consiste em `AnimalType` (chave de partição) e `Name` (chave de classificação). Ambos os atributos são do tipo `string`. Para recuperar um item do`Pets`, oAWSSDK cria a solicitação a seguir.

```
POST / HTTP/1.1
Host: dynamodb.<region>.<domain>;
Accept-Encoding: identity
Content-Length: <PayloadSizeBytes>
User-Agent: <UserAgentString>
Content-Type: application/x-amz-json-1.0
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
Signature=<Signature>
X-Amz-Date: <Date>
X-Amz-Target: DynamoDB_20120810.GetItem

{
    "TableName": "Pets",
    "Key": {
        "AnimalType": {"S": "Dog"},
        "Name": {"S": "Fido"}
    }
}
```

Observe o seguinte sobre essa solicitação:

- O `Authorization` cabeçalho contém as informações necessárias para o DynamoDB autenticar a solicitação. Para obter mais informações, consulte [Assinatura AWSSolicitações da API doeProcesso de assinatura do Signature versão 4](#)noReferência geral do Amazon Web Services Services.
- O `X-Amz-Target` contém o nome de uma operação do DynamoDB: `GetItem`. (Isso também é acompanhado pela versão da API de baixo nível, neste caso `20120810`.)
- A carga útil (corpo) da solicitação contém os parâmetros da operação, no formato JSON. Para a operação `GetItem`, os parâmetros são `TableName` e `Key`.

Formato de resposta

Após o recebimento da solicitação, o DynamoDB a processa e retorna uma resposta. Para a solicitação mostrada anteriormente, a carga de resposta HTTP(S) contém os resultados da operação, conforme mostrado no exemplo a seguir.

```
HTTP/1.1 200 OK
```

```
x-amzn-RequestId: <RequestId>
x-amz-crc32: <Checksum>
Content-Type: application/x-amz-json-1.0
Content-Length: <PayloadSizeBytes>
Date: <Date>
{
    "Item": {
        "Age": {"N": "8"},
        "Colors": {
            "L": [
                {"S": "White"},
                {"S": "Brown"},
                {"S": "Black"}
            ]
        },
        "Name": {"S": "Fido"},
        "Vaccinations": {
            "M": {
                "Rabies": {
                    "L": [
                        {"S": "2009-03-17"},
                        {"S": "2011-09-21"},
                        {"S": "2014-07-08"}
                    ]
                },
                "Distemper": {"S": "2015-10-13"}
            }
        },
        "Breed": {"S": "Beagle"},
        "AnimalType": {"S": "Dog"}
    }
}
```

Neste ponto, o AWS SDK retorna os dados da resposta ao seu aplicativo para processamento adicional.

Note

Se o DynamoDB não puder processar uma solicitação, ele retornará uma mensagem e um código de erro HTTP. O AWS SDK propaga esses elementos em seu aplicativo na forma de exceções.

Para mais informações, consulte [Tratamento de erros com o DynamoDB \(p. 221\)](#).

Descriidores de tipo de dados

O protocolo de API do DynamoDB de baixo nível exige que cada atributo seja acompanhado por um descritor de tipo de dados. Descrições de tipo de dados são tokens que informam ao DynamoDB como interpretar cada atributo.

Os exemplos em [Formato de solicitação \(p. 219\)](#) e [Formato de resposta \(p. 219\)](#) mostram exemplos de como os descritores de tipo de dados são usados. A solicitação `GetItem` especifica S para os atributos de esquema de chaves de Pets (`AnimalType` e `Name`), que são do tipo `string`. A resposta `GetItem` contém o item Pets com atributos do tipo `string` (S), `number` (N), `map` (M) e `list` (L).

Veja a seguir, uma lista completa dos descritores de tipo de dados do DynamoDB:

- **S** – String
- **N** – Number
- **B** – Binary
- **BOOL** – Boolean
- **NULL** – Null

- **M** – Map
- **L** – List
- **SS** – String Set
- **NS** – Number Set
- **BS** – Binary Set

Note

Para obter descrições detalhadas dos tipos de dados do DynamoDB, consulte [Tipos de dados \(p. 14\)](#).

Dados numéricos

As diferentes linguagens de programação oferecem diferentes níveis de suporte para JSON. Em alguns casos, é possível decidir usar uma biblioteca de terceiros para validar e analisar documentos JSON.

Algumas bibliotecas de terceiros se desenvolvem com base no tipo número do JSON, fornecendo seus próprios tipos, como `int`, `long` ou `double`. No entanto, o tipo de dados de número nativo no DynamoDB não é mapeado exatamente para esses outros tipos de dados, portanto, essas distinções de tipo podem causar conflitos. Além disso, muitas bibliotecas do JSON não manipulam valores numéricos de precisão fixa, e elas inferem automaticamente um tipo de dados duplo para sequências de dígitos que contêm um separador decimal.

Para solucionar esses problemas, o DynamoDB fornece um único tipo numérico sem perda de dados. Para evitar conversões implícitas indesejadas para um valor duplo, o DynamoDB usa strings para a transferência de dados de valores numéricos. Essa abordagem fornece flexibilidade para atualizar valores de atributo, sem deixar de manter a semântica de classificação adequada, como colocar os valores "01", "2" e "03" na sequência apropriada.

Se o número de precisão for importante para o seu aplicativo, você deverá converter valores numéricos em strings antes de passá-los para o DynamoDB.

Dados binários

O DynamoDB aceita atributos binários. No entanto, o JSON não é originalmente compatível com a codificação de dados binários. Para enviar dados binários em uma solicitação, será necessário codificá-los em formato base64. Ao receber a solicitação, o DynamoDB decodifica os dados em base64 de volta para binário.

O esquema de codificação base64 usado pelo DynamoDB é descrito em [RFC 4648](#) no website Internet Engineering Task Force (IETF).

Tratamento de erros com o DynamoDB

Esta seção descreve erros de tempo de execução e como lidar com eles. Ela também descreve códigos e mensagens de erro que são específicos para o Amazon DynamoDB.

Tópicos

- [Componentes de erros \(p. 222\)](#)
- [Mensagens e códigos de erro \(p. 222\)](#)
- [Tratamento de erros no seu aplicativo \(p. 225\)](#)

- Repetições de erro e recuo exponencial (p. 226)
- Operações em lote e tratamento de erros (p. 227)

Componentes de erros

Quando seu programa envia uma solicitação, o DynamoDB tenta processá-la. Se a solicitação for bem-sucedida, o DynamoDB retornará um código de status HTTP de êxito (200 OK), juntamente com os resultados da operação solicitada.

Se a solicitação for bem-sucedida, o DynamoDB retornará um erro. Cada erro tem três componentes:

- Um código de status HTTP (como 400).
- Um nome de exceção (como `ResourceNotFoundException`).
- Uma mensagem de erro (como `Requested resource not found: Table: tablename not found`).

OAWSOs SDKs cuidam da propagação de erros para o seu aplicativo para que você possa tomar as medidas apropriadas. Por exemplo, em um programa Java, você pode escrever a lógica `try-catch` para lidar com um `ResourceNotFoundException`.

Se você não estiver usando umAWSSDK, você precisa analisar o conteúdo da resposta de baixo nível do DynamoDB. Veja a seguir um exemplo dessa resposta.

```
HTTP/1.1 400 Bad Request
x-amzn-RequestId: LDM6CJP8RMQ1FHKSC1RBVJFPNVV4KQNSO5AEMF66Q9ASUAAJG
Content-Type: application/x-amz-json-1.0
Content-Length: 240
Date: Thu, 15 Mar 2012 23:56:23 GMT

{"__type": "com.amazonaws.dynamodb.v20120810#ResourceNotFoundException",
"message": "Requested resource not found: Table: tablename not found"}
```

Mensagens e códigos de erro

Veja a seguir uma lista de exceções retornadas pelo DynamoDB, agrupadas por código de status HTTP. Se OK para tentar novamente? for Sim, você poderá enviar a mesma solicitação novamente. Se OK to retry? (OK tentar novamente?) for No (Não), você deverá corrigir o problema no lado do cliente antes de enviar uma nova solicitação.

Código de status HTTP 400

Um código de status HTTP 400 indica um problema com sua solicitação, como falha de autenticação, parâmetros necessários ausentes ou limite excedido do throughput provisionado de uma tabela. Será necessário corrigir o problema no aplicativo antes de enviar a solicitação novamente.

`AccessDeniedException`

Mensagem: Acesso negado.

The client did not correctly sign the request. Se você estiver usando umAWSSDK, as solicitações serão assinadas para você automaticamente. Caso contrário, acesse a página do[Processo de assinatura do Signature versão 4](#)noAWSReferência geral.

OK para tentar novamente? Não

ConditionalCheckFailedException

Mensagem: The conditional request failed.

Você especificou uma condição avaliada como false. Por exemplo, você pode ter tentado realizar uma atualização condicional em um item, mas o valor real do atributo não correspondeu ao valor esperado na condição.

OK para tentar novamente? Não

IncompleteSignatureException

Mensagem: The request signature does not conform to AWS Padrões.

A assinatura da solicitação não incluía todos os componentes necessários. Se você estiver usando um AWSSDK, as solicitações serão assinadas para você automaticamente. Caso contrário, acesse a página do [Processo de assinatura do Signature versão 4](#) no AWS Referência geral.

OK para tentar novamente? Não

ItemCollectionSizeLimitExceededException

Mensagem: Tamanho da coleção excedido.

Para uma tabela com um índice secundário local, um grupo de itens com o mesmo valor de chave de partição excede o limite de tamanho máximo de 10 GB. Para obter mais informações sobre coleções de itens, consulte [Coleções de itens \(p. 612\)](#).

OK para tentar novamente? Sim

LimitExceededException

Mensagem: Muitas operações para um assinante específico.

Existem muitas operações simultâneas no plano de controle. O número cumulativo de tabelas e índices no CREATING, DELETING, ou UPDATING estado não pode exceder 50.

OK para tentar novamente? Sim

MissingAuthenticationTokenException

Mensagem: Solicitação deve conter um válido (registrado) AWS Access Key ID ID.

A solicitação não incluía o cabeçalho de autorização necessário ou estava mal formada. Consulte [API do DynamoDB de baixo nível \(p. 217\)](#).

OK para tentar novamente? Não

ProvisionedThroughputExceededException

Mensagem: Você excedeu seu throughput provisionado máximo permitido para uma tabela ou para um ou mais índices secundários globais. Para visualizar métricas de desempenho para throughput provisionado em relação ao throughput consumido, abra o [Console do Amazon CloudWatch](#).

Exemplo: Sua taxa de solicitação é muito alta. OAWSOs SDKs para o DynamoDB repetem automaticamente as solicitações que recebem essa exceção. Sua solicitação em algum momento terá

êxito, a menos que a fila de novas tentativas seja muito grande para concluir. Reduza a frequência de solicitações usando [Repetições de erro e recuo exponencial \(p. 226\)](#).

OK para tentar novamente? Sim

RequestLimitExceeded

Mensagem: Taxa de transferência excede o limite de taxa de transferência atual da sua conta. Para solicitar um aumento de limite, contate o AWS Support no <https://aws.amazon.com/support>.

Exemplo: Taxa de solicitações sob demanda excede a taxa de transferência permitida da conta.

OK para tentar novamente? Sim

ResourceInUseException

Mensagem: O recurso que você está tentando alterar está em uso.

Exemplo: Você tentou recriar uma tabela existente ou excluir uma tabela atualmente no CREATING Estado.

OK para tentar novamente? Não

ResourceNotFoundException

Mensagem: Recurso solicitado não encontrado.

Exemplo: A tabela que está sendo solicitada não existe ou está muito cedo no CREATING Estado.

OK para tentar novamente? Não

ThrottlingException

Mensagem: Rate of requests exceeds the allowed throughput.

Essa exceção é retornada como uma resposta da AmazonServiceException com um código de status THROTTLING_EXCEPTION. Essa exceção poderá ser retornada se você executar operações de API de [plano de controle](#) muito rapidamente.

Para as tabelas que usam o modo sob demanda, essa exceção poderá ser retornada para qualquer operação de API do [plano de dados](#) se a sua taxa de solicitação for muito alta. Para saber mais sobre a escalabilidade sob demanda, consulte [Tráfego de pico e propriedades de escalabilidade](#)

OK para tentar novamente? Sim

UnrecognizedClientException

Mensagem: Access Key ID ID ou token de segurança é inválido.

A assinatura da solicitação está incorreta. A causa mais provável é um inválido AWSID da chave de acesso ou chave secreta.

OK para tentar novamente? Sim

ValidationException

Mensagem: Varia dependendo do erro específico encontrado

Esse erro pode ocorrer por vários motivos, como um parâmetro necessário ausente, um valor fora do intervalo ou tipos de dados incompatíveis. A mensagem de erro contém detalhes sobre a parte específica da solicitação que causou o erro.

OK para tentar novamente? Não

Código de status HTTP 5xx

Um código de status HTTP 5xx indica um problema que deve ser resolvido pela AWS. Pode ser um erro transitório e, nesse caso, é possível repetir a solicitação até que ela tenha êxito. Caso contrário, acesse a seção [AWS Service Health Dashboard](#) para ver se há problemas operacionais com o serviço.

Internal Server Error (HTTP 500)

O DynamoDB não pôde processar a solicitação.

OK para tentar novamente? Sim

Note

Você pode encontrar erros de servidor internos enquanto trabalha com itens. Esses são esperados durante a vida útil de uma tabela. Todas as solicitações com falha podem ser repetidas imediatamente.

Service Unavailable (HTTP 503)

O DynamoDB está indisponível no momento. (Esse estado deve ser temporário.)

OK para tentar novamente? Sim

Tratamento de erros no seu aplicativo

Para que seu aplicativo seja executado sem problemas, é necessário adicionar uma lógica para detectar erros e reagir a eles. As abordagens comuns incluem o uso de blocos `try-catch` ou de uma instrução `if-then`.

Os SDKs da AWS realizam por conta própria novas tentativas e verificação de erros. Se você se deparar com um erro enquanto usa um dos AWSSDKs, o código de erro e a descrição podem ajudar a solucioná-lo.

Você também deve ver um `Request ID` na resposta. O `Request ID` pode ser útil se você precisa trabalhar com o `AWS Support` para diagnosticar um problema.

O exemplo de código Java a seguir tenta obter um item de uma tabela do DynamoDB e realiza um tratamento de erro rudimentar. (Nesse caso, ele simplesmente informa ao usuário que a solicitação falhou.)

```
Table table = dynamoDB.getTable("Movies");

try {
    Item item = table.getItem("year", 1978, "title", "Superman");
    if (item != null) {
        System.out.println("Result: " + item);
    } else {
        //No such item exists in the table
        System.out.println("Item not found");
    }
} catch (AmazonServiceException ase) {
```

```
System.err.println("Could not complete operation");
System.err.println("Error Message: " + ase.getMessage());
System.err.println("HTTP Status: " + ase.getStatusCode());
System.err.println("AWS Error Code: " + ase.getErrorCode());
System.err.println("Error Type: " + ase.getErrorType());
System.err.println("Request ID: " + ase.getRequestId());

} catch (AmazonClientException ace) {
    System.err.println("Internal error occurred communicating with DynamoDB");
    System.out.println("Error Message: " + ace.getMessage());
}
```

Neste exemplo de código, o construtor try-catch lida com dois tipos diferentes de exceções:

- `AmazonServiceException`— lançada se a solicitação do cliente foi transmitida corretamente ao DynamoDB, mas o DynamoDB não pôde processar a solicitação e retornou uma resposta de erro.
- `AmazonClientException`— Lançado se o cliente não conseguiu obter uma resposta de um serviço ou se ele não pôde analisar a resposta de um serviço.

Repetições de erro e recuo exponencial

Vários componentes em uma rede, como servidores DNS, switches, load balancers e outros, podem gerar erros em qualquer momento do ciclo de vida de uma determinada solicitação. A técnica usual para lidar com essas respostas de erro em um ambiente de rede é implementar novas tentativas no aplicativo cliente. Essa técnica aumenta a confiabilidade da aplicação.

Cada AWS SDK implementa a lógica de novas tentativas automaticamente. Você pode modificar os parâmetros de novas tentativas de acordo com as suas necessidades. Por exemplo, considere um aplicativo Java que exija uma estratégia rápida contra falhas, sem permitir novas tentativas em caso de erro. Com o AWS SDK for Java, você poderia usar a classe `ClientConfiguration` e fornecer um valor `maxErrorRetry` de 0 para desativar as novas tentativas. Para obter mais informações, consulte o [AWS Documentação do SDK](#) para sua linguagem de programação.

Se você não estiver usando um AWSSDK, você deve tentar novamente as solicitações originais que recebem erros do servidor (5xx). No entanto, erros de cliente (4xx, diferente de `ThrottlingException` ou `ProvisionedThroughputExceededException`) indicam que você precisa revisar a solicitação para corrigir o problema antes de tentar novamente.

Além de novas tentativas simples, cada AWS SDK implementa um algoritmo de recuo exponencial para um melhor controle de fluxo. O conceito por detrás do recuo exponencial é usar esperas progressivamente mais longas entre as novas tentativas para respostas de erro consecutivas. Por exemplo, até 50 milissegundos antes da primeira nova tentativa, até 100 milissegundos antes da segundo, até 200 milissegundos antes da terceira e assim por diante. No entanto, depois de um minuto, se a solicitação não tiver sido bem-sucedida, talvez o problema esteja relacionado ao tamanho da solicitação que excede o throughput provisionado e não à taxa de solicitação. Defina um tempo de interrupção de cerca de um minuto para o número máximo de novas tentativas. Se a solicitação não for bem-sucedida, investigue suas opções de throughput provisionado.

Note

O AWS SDKs da implementam uma lógica de novas tentativas automáticas e de recuo exponencial.

A maioria dos algoritmos de recuo exponencial usam variação (atraso randomizado) para evitar colisões sucessivas. Como você não está tentando evitar essas colisões nesses casos, não precisa usar esse número aleatório. No entanto, se você usar clientes simultâneos, a variação pode ajudar suas solicitações a serem bem-sucedidas mais depressa. Para obter mais informações, consulte a postagem no blog sobre [Recuo exponencial e variação](#).

Operações em lote e tratamento de erros

A API de baixo nível do DynamoDB oferece suporte a operações em lote para leituras e gravações. `BatchGetItem` leva itens de uma ou mais tabelas, e `BatchWriteItem` Coloca ou exclui itens em uma ou mais tabelas. Essas operações em lote são implementadas como wrappers em torno de outras operações do DynamoDB que não estão em lote. Em outras palavras, `BatchGetItem` invoca `GetItem` uma vez para cada item do lote. Da mesma forma, `BatchWriteItem` invoca `DeleteItem` ou `PutItem`, conforme apropriado, para cada item do lote.

Uma operação em lote pode tolerar a falha de solicitações individuais no lote. Por exemplo, considere uma solicitação `BatchGetItem` para ler cinco itens. Mesmo se algumas das solicitações `GetItem` subjacentes falharem, isso não faz com que toda a operação `BatchGetItem` falhe. Entretanto, se todas as cinco operações de leitura falharem, todo o `BatchGetItem` falhará.

As operações em lote retornam informações sobre solicitações individuais que apresentam falhas, para que você possa diagnosticar o problema e repetir a operação. Para `BatchGetItem`, as tabelas e chaves primárias em questão são retornadas no valor de `UnprocessedKeys` da resposta. Para `BatchWriteItem`, informações semelhantes são retornadas em `UnprocessedItems`.

A causa mais provável de uma falha de leitura ou gravação é a controle de utilização. Para `BatchGetItem`, uma ou mais das tabelas na solicitação em lote não tem capacidade de leitura provisionada suficiente para dar suporte à operação. Para `BatchWriteItem`, uma ou mais das tabelas não tem capacidade de gravação provisionada suficiente.

Se o DynamoDB retornar itens não processados, você deverá repetir a operação em lote nesses itens. No entanto, recomendamos que você use um algoritmo de recuo exponencial. Se você repetir a operação em lote imediatamente, solicitações subjacentes de leitura ou gravação ainda poderão falhar devido ao controle de utilização nas tabelas individuais. Se você atrasar a operação em lote usando o recuo exponencial, as solicitações individuais no lote terão muito mais chances de sucesso.

Interfaces de programação de nível superior para o DynamoDB

O AWS SDKs fornecem aplicativos com interfaces de baixo nível para trabalhar com o Amazon DynamoDB. Essas classes e métodos no lado do cliente correspondem diretamente à API do DynamoDB de baixo nível. No entanto, muitos desenvolvedores experimentam uma sensação de desconexão, ou divergência de impedância, quando precisam mapear tipos de dados complexos para itens em uma tabela de banco de dados. Com uma interface de banco de dados de baixo nível, os desenvolvedores precisam escrever métodos para a leitura ou a gravação de dados de objetos em tabelas de banco de dados, e vice-versa. A quantidade de código extra necessária para cada combinação de tipo de objeto e tabela de banco de dados pode parecer esmagadora.

Para simplificar o desenvolvimento, o AWS SDKs para Java e .NET fornecem interfaces adicionais com níveis mais altos de abstração. As interfaces de nível superior para o DynamoDB permitem que você defina as relações entre objetos no seu programa e as tabelas de banco de dados que armazenam os dados desses objetos. Depois de definir esse mapeamento, você chama métodos de objeto simples, como `save`, `load`, ou `delete` e as operações do DynamoDB de baixo nível subjacentes são invocadas automaticamente em seu nome. Isso permite que você escreva um código centrado em objetos, em vez de um código centrado no banco de dados.

Interfaces de programação de nível superior para o DynamoDB estão disponíveis no AWS SDKs para Java e .NET.

Java

- [Java: DynamoDBMapper \(p. 228\)](#)

.NET

- [.NET: Modelo de documento \(p. 276\)](#)
- [.NET: Modelo de persistência de objeto \(p. 299\)](#)

Java: DynamoDBMapper

Tópicos

- [Tipos de dados compatíveis \(p. 230\)](#)
- [Anotações Java para o DynamoDB \(p. 231\)](#)
- [Classe DynamoDBMapper \(p. 236\)](#)
- [Definições de configuração opcionais para DynamoDBMapper \(p. 244\)](#)
- [Exemplo: Operações de CRUD \(p. 245\)](#)
- [Exemplo: Operações de gravação em Batch \(p. 247\)](#)
- [Exemplo: Consulta e verificação \(p. 254\)](#)
- [Exemplo: Operações de transação \(p. 263\)](#)
- [Bloqueio otimista com número de versão \(p. 270\)](#)
- [Mapeamento de dados arbitrários \(p. 273\)](#)

O AWS SDK for Java fornece um `DynamoDBMapper` classe, permitindo mapear classes no lado do cliente para tabelas do Amazon DynamoDB. Para usar `DynamoDBMapper`, defina a relação entre os itens em uma tabela do DynamoDB e as instâncias de objeto correspondentes no seu código. O `DynamoDBMapper` classe permite acessar tabelas, realizar várias operações de criação, leitura, atualização e exclusão (CRUD) e executar consultas.

Note

A classe `DynamoDBMapper` não permite criar, atualizar ou excluir tabelas. Para realizar essas tarefas, use em vez disso o SDK de baixo nível para a interface Java. Para mais informações, consulte [Como trabalhar com tabelas do DynamoDB em Java \(p. 392\)](#).

O SDK for Java fornece um conjunto de tipos de anotações, para que você possa mapear suas classes para tabelas. Por exemplo, considere uma tabela `ProductCatalog` cujo `Id` seja a chave de partição.

```
ProductCatalog(Id, ...)
```

É possível mapear uma classe no seu aplicativo cliente para a tabela `ProductCatalog`, conforme mostrado no código Java a seguir. Este código define um objeto Java antigo simples (POJO) chamado `CatalogItem`, que usa anotações para mapear campos de objeto para nomes de atributos do DynamoDB.

Example

```
package com.amazonaws.codesamples;

import java.util.Set;

import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBAttribute;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBIgnore;
```

```
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;

@DynamoDBTable(tableName="ProductCatalog")
public class CatalogItem {

    private Integer id;
    private String title;
    private String ISBN;
    private Set<String> bookAuthors;
    private String someProp;

    @DynamoDBHashKey(attributeName="Id")
    public Integer getId() { return id; }
    public void setId(Integer id) { this.id = id; }

    @DynamoDBAttribute(attributeName="Title")
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    @DynamoDBAttribute(attributeName="ISBN")
    public String getISBN() { return ISBN; }
    public void setISBN(String ISBN) { this.ISBN = ISBN; }

    @DynamoDBAttribute(attributeName="Authors")
    public Set<String> getBookAuthors() { return bookAuthors; }
    public void setBookAuthors(Set<String> bookAuthors) { this.bookAuthors = bookAuthors; }

    @DynamoDBIgnore
    public String getSomeProp() { return someProp; }
    public void setSomeProp(String someProp) { this.someProp = someProp; }
}
```

No código anterior, a anotação `@DynamoDBTable` mapeia a classe `CatalogItem` para a tabela `ProductCatalog`. Você pode armazenar instâncias de classes individuais como itens na tabela. Na definição de classe, a anotação `@DynamoDBHashKey` mapeia a propriedade `Id` para a chave primária.

Por padrão, as propriedades da classe são mapeadas para os atributos com o mesmo nome na tabela. As propriedades `Title` e `ISBN` são mapeadas para os atributos com o mesmo nome na tabela.

O `@DynamoDBAttribute` é opcional quando o nome do atributo DynamoDB corresponde ao nome da propriedade declarada na classe. Quando esses nomes são diferentes, use essa anotação com `attributeName()` para especificar a qual atributo DynamoDB essa propriedade corresponde.

No exemplo anterior, a anotação `@DynamoDBAttribute` é adicionada a cada propriedade para garantir que os nomes de propriedades correspondam exatamente às tabelas criadas em [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#) e sejam consistentes com os nomes de atributos usados em outros exemplos de código neste guia.

Sua definição de classe pode ter propriedades que não são mapeadas para atributos na tabela. Para identificar essas propriedades, adicione a anotação `@DynamoDBIgnore`. No exemplo anterior, a propriedade `SomeProp` está marcada com a anotação `@DynamoDBIgnore`. Ao carregar uma instância `CatalogItem` na tabela, sua instância `DynamoDBMapper` não inclui a propriedade `SomeProp`. Além disso, o mapeador não retorna esse atributo quando você recupera um item da tabela.

Depois de definir sua classe de mapeamento, é possível usar métodos `DynamoDBMapper` para gravar uma instância dessa classe em um item correspondente na tabela `Catalog`. O exemplo de código a seguir demonstra essa técnica.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();

DynamoDBMapper mapper = new DynamoDBMapper(client);
```

```
CatalogItem item = new CatalogItem();
item.setId(102);
item.setTitle("Book 102 Title");
item.setISBN("222-2222222222");
item.setBookAuthors(new HashSet<String>(Arrays.asList("Author 1", "Author 2")));
item.setSomeProp("Test");

mapper.save(item);
```

O exemplo de código a seguir mostra como recuperar o item e acessar alguns dos seus atributos.

```
CatalogItem partitionKey = new CatalogItem();

partitionKey.setId(102);
DynamoDBQueryExpression<CatalogItem> queryExpression = new
    DynamoDBQueryExpression<CatalogItem>()
        .withHashKeyValues(partitionKey);

List<CatalogItem> itemList = mapper.query(CatalogItem.class, queryExpression);

for (int i = 0; i < itemList.size(); i++) {
    System.out.println(itemList.get(i).getTitle());
    System.out.println(itemList.get(i).getBookAuthors());
}
```

DynamoDBMapper oferece uma forma intuitiva e natural de trabalhar com dados do DynamoDB no Java. Ele também fornece uma série de recursos internos, como bloqueio otimista, transações ACID, valores de chave de partição e chave de classificação gerados automaticamente e versionamento de objetos.

Tipos de dados compatíveis

Esta seção descreve os tipos de dados arbitrários, as coleções e os tipos de dados Java primitivos com suporte no Amazon DynamoDB.

O Amazon DynamoDB oferece suporte aos seguintes tipos de dados Java primitivos e classes wrapper primitivas.

- `String`
- `Boolean, boolean`
- `Byte, byte`
- `Date` (como a string de precisão em milissegundos [ISO_8601](#), modificada para UTC)
- `Calendar` (como a string de precisão em milissegundos [ISO_8601](#), modificada para UTC)
- `Long, long`
- `Integer, int`
- `Double, double`
- `Float, float`
- `BigDecimal`
- `BigInteger`

Note

- Para obter mais informações sobre as regras de nomenclatura do DynamoDB e os vários tipos de dados compatíveis, consulte[Regras de nomeação e tipos de dados \(p. 13\)](#).
- O DynamoDBMapper oferece suporte a valores Binary vazios.

- O suporte a valores String vazios pelo AWS SDK for Java 2.x.

Dentro AWSSDK for Java 1.x, o DynamoDBMapper oferece suporte à leitura de valores de atributos String vazios. No entanto, ele não gravará valores de atributos String vazios porque esses atributos são descartados da solicitação.

O DynamoDB oferece suporte ao Java [Defina](#), [Lista](#), e [Mapa](#) Tipos de coleção. A tabela a seguir resume como esses tipos Java são mapeados para os tipos do DynamoDB.

Tipo Java	Tipo DynamoDB
Todos os tipos de número	N (tipo Número)
Strings	S (tipo String)
Booleano	BOOL (tipo booleano), 0 ou 1.
ByteBuffer	B (tipo Binário)
Data	s (tipo String). Os valores de Date são armazenados como strings formatadas em ISO-8601.
Tipos de coleção Set	Tipo ss (conjunto de strings), tipo ns (conjunto de números) e tipo bs (conjunto de binários).

O [DynamoDBTypeConverter](#) permite que você mapeie seus próprios tipos de dados arbitrários para um tipo de dados com suporte nativo pelo DynamoDB. Para mais informações, consulte [Mapeamento de dados arbitrários \(p. 273\)](#).

Anotações Java para o DynamoDB

Esta seção descreve as anotações que estão disponíveis para mapear suas classes e propriedades para tabelas e atributos no Amazon DynamoDB.

Para acessar a documentação Javadoc correspondente, consulte [Resumo dos tipos de anotação](#) no [AWS SDK for Java Referência de API](#).

Note

Nas seguintes anotações, apenas `DynamoDBTable` e `DynamoDBHashKey` são necessários.

Tópicos

- [DynamoDBAttribute \(p. 232\)](#)
- [DynamoDBAutoGeneratedKey \(p. 232\)](#)
- [DynamoDBDocument \(p. 232\)](#)
- [DynamoDBHashKey \(p. 234\)](#)
- [DynamoDBIgnore \(p. 234\)](#)
- [DynamoDBIndexHashKey \(p. 234\)](#)
- [DynamoDBIndexRangeKey \(p. 234\)](#)
- [DynamoDBRangeKey \(p. 234\)](#)
- [DynamoDBTable \(p. 235\)](#)
- [DynamoDBTypeConverted \(p. 235\)](#)

- [DynamoDBTyped \(p. 236\)](#)
- [DynamoDBVersionAttribute \(p. 236\)](#)

DynamoDBAttribute

Mapeia uma propriedade para um atributo de tabela. Por padrão, cada propriedade de classe é mapeada para um atributo de item com o mesmo nome. No entanto, se os nomes não forem os mesmos, você poderá usar essa anotação para mapear uma propriedade para o atributo. No seguinte trecho de código Java, `DynamoDBAttribute` mapeia a propriedade `BookAuthors` para o nome de atributo `Authors` na tabela.

```
@DynamoDBAttribute(attributeName = "Authors")
public List<String> getBookAuthors() { return BookAuthors; }
public void setBookAuthors(List<String> BookAuthors) { this.BookAuthors = BookAuthors; }
```

O `DynamoDBMapper` usa `Authors` como o nome do atributo ao salvar o objeto na tabela.

DynamoDBAutoGeneratedKey

Marca uma propriedade de chave de partição ou de chave de classificação como sendo gerada automaticamente. `DynamoDBMapper` gera um [UUID](#) aleatório ao salvar esses atributos. Apenas propriedades String podem ser marcadas como chaves geradas automaticamente.

O exemplo a seguir demonstra o uso de chaves geradas automaticamente.

```
@DynamoDBTable(tableName="AutoGeneratedKeysExample")
public class AutoGeneratedKeys {
    private String id;
    private String payload;

    @DynamoDBHashKey(attributeName = "Id")
    @DynamoDBAutoGeneratedKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    @DynamoDBAttribute(attributeName="payload")
    public String getPayload() { return this.payload; }
    public void setPayload(String payload) { this.payload = payload; }

    public static void saveItem() {
        AutoGeneratedKeys obj = new AutoGeneratedKeys();
        obj.setPayload("abc123");

        // id field is null at this point
        DynamoDBMapper mapper = new DynamoDBMapper(dynamoDBClient);
        mapper.save(obj);

        System.out.println("Object was saved with id " + obj.getId());
    }
}
```

DynamoDBDocument

Indica que uma classe pode ser serializada como um documento do Amazon DynamoDB.

Por exemplo, vamos supor que você queria mapear um documento JSON para um atributo DynamoDB do tipo Map (M). O exemplo de código a seguir define um item que contém um atributo aninhado (Pictures) do tipo Map.

```
public class ProductCatalogItem {  
  
    private Integer id; //partition key  
    private Pictures pictures;  
    /* ...other attributes omitted... */  
  
    @DynamoDBHashKey(attributeName="Id")  
    public Integer getId() { return id; }  
    public void setId(Integer id) {this.id = id;}  
  
    @DynamoDBAttribute(attributeName="Pictures")  
    public Pictures getPictures() { return pictures; }  
    public void setPictures(Pictures pictures) {this.pictures = pictures;}  
  
    // Additional properties go here.  
  
    @DynamoDBDocument  
    public static class Pictures {  
        private String frontView;  
        private String rearView;  
        private String sideView;  
  
        @DynamoDBAttribute(attributeName = "FrontView")  
        public String getFrontView() { return frontView; }  
        public void setFrontView(String frontView) { this.frontView = frontView; }  
  
        @DynamoDBAttribute(attributeName = "RearView")  
        public String getRearView() { return rearView; }  
        public void setRearView(String rearView) { this.rearView = rearView; }  
  
        @DynamoDBAttribute(attributeName = "SideView")  
        public String getSideView() { return sideView; }  
        public void setSideView(String sideView) { this.sideView = sideView; }  
    }  
}
```

É possível salvar um novo item `ProductCatalog`, com `Pictures`, conforme mostrado no exemplo a seguir.

```
ProductCatalogItem item = new ProductCatalogItem();  
  
Pictures pix = new Pictures();  
pix.setFrontView("http://example.com/products/123_front.jpg");  
pix.setRearView("http://example.com/products/123_rear.jpg");  
pix.setSideView("http://example.com/products/123_left_side.jpg");  
item.setPictures(pix);  
  
item.setId(123);  
  
mapper.save(item);
```

O item de `ProductCatalog` resultante seria semelhante ao seguinte (no formato JSON).

```
{  
    "Id" : 123  
    "Pictures" : {  
        "SideView" : "http://example.com/products/123_left_side.jpg",  
        "RearView" : "http://example.com/products/123_rear.jpg",  
        "FrontView" : "http://example.com/products/123_front.jpg"  
    }  
}
```

DynamoDBHashKey

Mapeia uma propriedade de classe para a chave de partição da tabela. A propriedade deve ser uma string escalar, número ou tipos binários. A propriedade não pode ser um tipo de coleção.

Vamos supor que você tenha uma tabela, `ProductCatalog`, com `Id` como chave primária. O código Java a seguir define uma classe `CatalogItem` e mapeia sua propriedade `Id` para a chave primária da tabela `ProductCatalog` usando a tag `@DynamoDBHashKey`.

```
@DynamoDBTable(tableName="ProductCatalog")
public class CatalogItem {
    private Integer Id;
    @DynamoDBHashKey(attributeName="Id")
    public Integer getId() {
        return Id;
    }
    public void setId(Integer Id) {
        this.Id = Id;
    }
    // Additional properties go here.
}
```

DynamoDBIgnore

Indica à instância de `DynamoDBMapper` que a propriedade associada deve ser ignorada. Quando salvar dados na tabela, o `DynamoDBMapper` não salva essa propriedade na tabela.

Aplicado ao método getter ou ao campo de classe de uma propriedade não modelada. Se a anotação for aplicada diretamente ao campo de classe, o getter e o setter correspondentes deverão ser declarados na mesma classe.

DynamoDBIndexHashKey

Mapeia uma propriedade de classe para a chave de partição de um índice secundário global. A propriedade deve ser uma string escalar, número ou tipos binários. A propriedade não pode ser um tipo de coleção.

Use essa anotação se você precisa realizar uma operação em um índice secundário global. É necessário especificar o nome de índice (`globalSecondaryIndexName`). Se o nome da propriedade da classe for diferente da chave de partição do índice, também será necessário especificar o nome desse atributo de índice (`attributeName`).

DynamoDBIndexRangeKey

Mapeia uma propriedade de classe para a chave de classificação de um índice secundário global ou um índice secundário local. A propriedade deve ser uma string escalar, número ou tipos binários. A propriedade não pode ser um tipo de coleção.

Use essa anotação se você precisa realizar uma operação em um índice secundário local ou um índice secundário global e deseja refinar seus resultados usando a chave de classificação de índice. É necessário especificar o nome de índice (`globalSecondaryIndexName` ou `localSecondaryIndexName`). Se o nome da propriedade da classe for diferente da chave de classificação do índice, você também deve especificar o nome desse atributo de índice (`attributeName`).

DynamoDBRangeKey

Mapeia uma propriedade de classe para a chave de classificação da tabela. A propriedade deve ser uma string escalar, número ou tipos binários. Não pode ser um tipo de coleção.

Se a chave primária for composta (chave de partição e chave de classificação), você poderá usar essa tag para mapear seu campo de classe para a chave de classificação. Por exemplo, vamos supor que você tenha uma tabela `Reply` que armazena respostas para tópicos de fórum. Cada tópico pode ter muitas respostas. Portanto, a chave primária dessa tabela é tanto `ThreadId` quanto `ReplyDateTime`. `ThreadId` é a chave de partição, e `ReplyDateTime` é a chave de classificação.

O código Java a seguir define uma classe `Reply` e a mapeia para a tabela `Reply`. Ele usa ambas as tags `@DynamoDBHashKey` e `@DynamoDBRangeKey` para identificar propriedades de classes que são mapeadas para a chave primária.

```
@DynamoDBTable(tableName="Reply")
public class Reply {
    private Integer id;
    private String replyDateTime;

    @DynamoDBHashKey(attributeName="Id")
    public Integer getId() { return id; }
    public void setId(Integer id) { this.id = id; }

    @DynamoDBRangeKey(attributeName="ReplyDateTime")
    public String getReplyDateTime() { return replyDateTime; }
    public void setReplyDateTime(String replyDateTime) { this.replyDateTime =
        replyDateTime; }

    // Additional properties go here.
}
```

DynamoDBTable

Identifica a tabela de destino no DynamoDB. Por exemplo, o seguinte código Java define uma classe `Developer` e mapeia-o para o `People` no DynamoDB.

```
@DynamoDBTable(tableName="People")
public class Developer { ...}
```

A anotação `@DynamoDBTable` pode ser herdada. Qualquer nova classe que herde da classe `Developer` também é mapeada para a tabela `People`. Por exemplo, suponha que você crie uma classe `Lead` que herde da classe `Developer`. Como você mapeou a classe `Developer` para a tabela `People`, e os objetos de classe `Lead` também são armazenados na mesma tabela.

Também é possível substituir `@DynamoDBTable`. Qualquer nova classe que herde da classe `Developer` por padrão é mapeada para a mesma tabela `People`. No entanto, você pode substituir esse mapeamento padrão. Por exemplo, se você criar uma classe que herda da classe `Developer`, será possível mapeá-la explicitamente para outra tabela adicionando a anotação `@DynamoDBTable`, conforme mostrado no seguinte exemplo de código Java.

```
@DynamoDBTable(tableName="Managers")
public class Manager extends Developer { ...}
```

DynamoDBTypeConverted

Uma anotação para marcar uma propriedade como usando um conversor de tipo personalizado. Pode ser anotada em uma anotação definida pelo usuário para transmitir propriedades adicionais a `DynamoDBTypeConverter`.

O `DynamoDBTypeConverter` permite que você mapeie seus próprios tipos de dados arbitrários para um tipo de dados com suporte nativo pelo DynamoDB. Para mais informações, consulte [Mapeamento de dados arbitrários \(p. 273\)](#).

DynamoDBTyped

Uma anotação para substituir a associação de tipo de atributo padrão. Tipos padrão não exigem a anotação se estiverem aplicando a associação de atributo padrão para esse tipo.

DynamoDBVersionAttribute

Identifica uma propriedade de classe para armazenar um número de versão de bloqueio otimista. `DynamoDBMapper` atribui um número de versão a essa propriedade ao salvar um novo item e o incrementa cada vez que você atualizar o item. Apenas há suporte para tipos de números escalares. Para obter informações sobre tipos de dados, consulte [Tipos de dados \(p. 14\)](#). Para obter mais informações sobre controle de versão, consulte [Bloqueio otimista com número de versão \(p. 270\)](#).

Classe DynamoDBMapper

`ODynamoDBMapper`A classe é o ponto de entrada para o Amazon DynamoDB. Ela fornece acesso a um endpoint do DynamoDB e permite acessar seus dados em várias tabelas. Ele também permite realizar várias operações de criação, leitura, atualização e exclusão (CRUD) em itens e executar consultas e verificações em tabelas. Essa classe fornece os seguintes métodos para trabalhar com o DynamoDB.

Para acessar a documentação Javadoc correspondente, consulte[DynamoDBMapper no AWS SDK for Java Referência de API](#) do.

Tópicos

- [save \(p. 236\)](#)
- [load \(p. 237\)](#)
- [delete \(p. 237\)](#)
- [query \(p. 237\)](#)
- [queryPage \(p. 239\)](#)
- [scan \(p. 239\)](#)
- [scanPage \(p. 240\)](#)
- [parallelScan \(p. 240\)](#)
- [batchSave \(p. 240\)](#)
- [batchLoad \(p. 241\)](#)
- [batchDelete \(p. 241\)](#)
- [batchWrite \(p. 241\)](#)
- [transactionWrite \(p. 242\)](#)
- [transactionLoad \(p. 242\)](#)
- [count \(p. 243\)](#)
- [generateCreateTableRequest \(p. 243\)](#)
- [createS3Link \(p. 243\)](#)
- [getS3ClientCache \(p. 244\)](#)

save

Salva o objeto especificado na tabela. O objeto que você deseja salvar é o único parâmetro necessário para esse método. É possível fornecer parâmetros de configuração opcionais usando o objeto `DynamoDBMapperConfig`.

Se um item com a mesma chave primária não existir, esse método criará um novo item na tabela. Se existir um item com a mesma chave primária, ele atualizará o item existente. Se a chave de partição e a chave de classificação forem do tipo String e estiverem anotadas com `@DynamoDBAutoGeneratedKey`, elas receberão um identificador universal exclusivo (UUID) aleatório se não forem inicializadas. Campos de versão anotados com `@DynamoDBVersionAttribute` são incrementados em um. Além disso, se um campo de versão for atualizado ou se uma chave for gerada, o objeto transmitido será atualizado como resultado da operação.

Por padrão, somente atributos correspondentes às propriedades de classe mapeadas são atualizados. Quaisquer atributos existentes em um item não são afetados. No entanto, se você especificar `SaveBehavior.CLOBBER`, poderá forçar o item a ser completamente substituído.

```
mapper.save(obj, new DynamoDBMapperConfig(DynamoDBMapperConfig.SaveBehavior.CLOBBER));
```

Se o versionamento estiver habilitado, as versões dos itens no lado do servidor e no lado do cliente deverão corresponder. No entanto, a versão não precisará corresponder se a opção `SaveBehavior.CLOBBER` for usada. Para obter mais informações sobre controle de versão, consulte [Bloqueio otimista com número de versão \(p. 270\)](#).

load

Recupera um item de uma tabela. É necessário fornecer a chave primária do item que você deseja recuperar. É possível fornecer parâmetros de configuração opcionais usando o objeto `DynamoDBMapperConfig`. Por exemplo, você pode solicitar opcionalmente leituras fortemente consistentes para garantir que esse método recupere apenas os valores de itens mais recentes, como mostra a seguinte instrução Java.

```
CatalogItem item = mapper.load(CatalogItem.class, item.getId(),  
    new DynamoDBMapperConfig(DynamoDBMapperConfig.ConsistentReads.CONSISTENT));
```

Por padrão, o DynamoDB retorna o item que possui valores eventualmente consistentes. Para obter informações sobre o modelo de consistência eventual do DynamoDB, consulte [Consistência de leituras \(p. 17\)](#).

delete

Exclui um item da tabela. Você deve transmitir uma instância do objeto da classe mapeada.

Se o versionamento estiver habilitado, as versões dos itens no lado do servidor e no lado do cliente deverão corresponder. No entanto, a versão não precisará corresponder se a opção `SaveBehavior.CLOBBER` for usada. Para obter mais informações sobre controle de versão, consulte [Bloqueio otimista com número de versão \(p. 270\)](#).

query

Consulta uma tabela ou um índice secundário. Você poderá consultar uma tabela ou um índice somente se ele tiver uma chave primária composta (chave de partição e chave de classificação). Esse método requer que você forneça um valor de chave de partição e um filtro de consulta que é aplicado à chave de classificação. Uma expressão de filtro inclui uma condição e um valor.

Vamos supor que você tenha uma tabela, `Reply`, que armazena as respostas de tópicos de fórum. Cada assunto de tópico pode ter 0 ou mais respostas. A chave primária da tabela `Reply` consiste nos campos `Id` e `ReplyDateTime`, em que `Id` é a chave de partição e `ReplyDateTime` é a chave de classificação da chave primária.

```
Reply ( Id, ReplyDateTime, ... )
```

Suponha que você tenha criado um mapeamento entre um `Reply` e a classe correspondente `Reply` no DynamoDB. O código Java a seguir usa `DynamoDBMapper` para localizar todas as respostas nas últimas duas semanas para um assunto de tópico específico.

Example

```
String forumName = "&DDB;";
String forumSubject = "&DDB; Thread 1";
String partitionKey = forumName + "#" + forumSubject;

long twoWeeksAgoMilli = (new Date()).getTime() - (14L*24L*60L*60L*1000L);
Date twoWeeksAgo = new Date();
twoWeeksAgo.setTime(twoWeeksAgoMilli);
SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");
String twoWeeksAgoStr = df.format(twoWeeksAgo);

Map<String,AttributeValue> eav = new HashMap<String,AttributeValue>();
eav.put(":v1", new AttributeValue().withS(partitionKey));
eav.put(":v2",new AttributeValue().withS(twoWeeksAgoStr.toString()));

DynamoDBQueryExpression<Reply> queryExpression = new DynamoDBQueryExpression<Reply>()
    .withKeyConditionExpression("Id = :v1 and ReplyDateTime > :v2")
    .withExpressionAttributeValues(eav);

List<Reply> latestReplies = mapper.query(Reply.class, queryExpression);
```

A consulta retorna uma coleção de objetos `Reply`.

Por padrão, o método `query` retorna uma coleção de "carregamento preguiçoso". Ele inicialmente retorna apenas uma página de resultados e, em seguida, faz uma chamada de serviço para a próxima página, se necessário. Para obter todos os itens correspondentes, faça uma iteração na coleção `latestReplies`.

Observe que chamar o método `size()` na coleção carregará todos os resultados para fornecer uma contagem precisa. Isso pode fazer com que uma grande quantidade de taxa de transferência provisionada seja consumida, e em uma tabela muito grande pode até esgotar toda a memória na JVM.

Para consultar um índice, você deve primeiro modelá-lo como uma classe de mapeador. Suponha que `Reply` a tabela tem um índice secundário global denominado `PostedBy-Message-Index`. A chave de partição para esse índice é `PostedBy`, e a chave de classificação é `Message`. A definição de classe para um item no índice seria semelhante ao seguinte.

```
@DynamoDBTable(tableName="Reply")
public class PostedByMessage {
    private String postedBy;
    private String message;

    @DynamoDBIndexHashKey(globalSecondaryIndexName = "PostedBy-Message-Index",
    attributeName = "PostedBy")
    public String getPostedBy() { return postedBy; }
    public void setPostedBy(String postedBy) { this.postedBy = postedBy; }

    @DynamoDBIndexRangeKey(globalSecondaryIndexName = "PostedBy-Message-Index",
    attributeName = "Message")
    public String getMessage() { return message; }
    public void setMessage(String message) { this.message = message; }

    // Additional properties go here.
}
```

A anotação `@DynamoDBTable` indica que esse índice está associado à tabela `Reply`. A anotação `@DynamoDBIndexHashKey` representa a chave de partição (`PostedBy`) do índice, enquanto `@DynamoDBIndexRangeKey` representa a chave de classificação (`Message`) do índice.

Agora, você pode usar `DynamoDBMapper` para consultar o índice, recuperando um subconjunto das mensagens que foram postadas por um usuário específico. É necessário especificar `withIndexName` para que o DynamoDB saiba qual índice deve consultar. O código a seguir consulta um índice secundário global. Como um índice secundário global oferece suporte para leituras eventualmente consistentes, mas não para leituras fortemente consistentes, é necessário especificar `withConsistentRead(false)`.

```
HashMap<String,AttributeValue> eav = new HashMap<String,AttributeValue>();
eav.put(":v1", new AttributeValue().withS("User A"));
eav.put(":v2", new AttributeValue().withS("DynamoDB"));

DynamoDBQueryExpression<PostedByMessage> queryExpression = new
    DynamoDBQueryExpression<PostedByMessage>()
        .withIndexName("PostedBy-Message-Index")
        .withConsistentRead(false)
        .withKeyConditionExpression("PostedBy = :v1 and begins_with(Message, :v2)")
        .withExpressionAttributeValues(eav);

List<PostedByMessage> iList = mapper.query(PostedByMessage.class, queryExpression);
```

A consulta retorna uma coleção de objetos `PostedByMessage`.

queryPage

Consulta uma tabela ou um índice secundário e retorna uma única página de resultados correspondentes. Como acontece com o método `query`, você deve especificar um valor de chave de partição e um filtro de consulta que seja aplicado ao atributo de chave de classificação. No entanto, `queryPage` retorna somente a primeira “página” de dados, ou seja, a quantidade de dados que caberá em 1 MB.

scan

Verifica uma tabela ou um índice secundário inteiro. Você tem a opção de especificar uma `FilterExpression` para filtrar o conjunto de resultados.

Vamos supor que você tenha uma tabela, `Reply`, que armazena as respostas de tópicos de fórum. Cada assunto de tópico pode ter 0 ou mais respostas. A chave primária da tabela `Reply` consiste nos campos `Id` e `ReplyDateTime`, em que `Id` é a chave de partição e `ReplyDateTime` é a chave de classificação da chave primária.

```
Reply ( Id, ReplyDateTime, ... )
```

Se você mapeou uma classe Java para a tabela `Reply`, será possível usar `DynamoDBMapper` para verificar essa tabela. Por exemplo, o código Java a seguir verifica a tabela `Reply` inteira, retornando somente as respostas de um determinado ano.

Example

```
HashMap<String,AttributeValue> eav = new HashMap<String,AttributeValue>();
eav.put(":v1", new AttributeValue().withS("2015"));

DynamoDBScanExpression scanExpression = new DynamoDBScanExpression()
    .withFilterExpression("begins_with(ReplyDateTime,:v1)")
    .withExpressionAttributeValues(eav);

List<Reply> replies = mapper.scan(Reply.class, scanExpression);
```

Por padrão, o método `scan` retorna uma coleção de “carregamento preguiçoso”. Ele inicialmente retorna apenas uma página de resultados e, em seguida, faz uma chamada de serviço para a próxima página, se necessário. Para obter todos os itens correspondentes, faça uma iteração na coleção `replies`.

Observe que chamar o método `size()` na coleção carregará todos os resultados para fornecer uma contagem precisa. Isso pode fazer com que uma grande quantidade de taxa de transferência provisionada seja consumida, e em uma tabela muito grande pode até esgotar toda a memória na JVM.

Para verificar um índice, você deve primeiro modelá-lo como uma classe de mapeador. Suponha que `oReplyA` tabela tem um índice secundário global denominado `PostedBy-Message-Index`. A chave de partição para esse índice é `PostedBy`, e a chave de classificação é `Message`. Uma classe de mapeador para esse índice é mostrada na seção [query \(p. 237\)](#). Ela usa as anotações `@DynamoDBIndexHashKey` e `@DynamoDBIndexRangeKey` para especificar a chave de classificação e a chave de partição do índice.

O código de exemplo a seguir verifica `PostedBy-Message-Index`. Ele não usa um filtro de verificação e, portanto, todos os itens no índice são retornados para você.

```
DynamoDBScanExpression scanExpression = new DynamoDBScanExpression()
    .withIndexName("PostedBy-Message-Index")
    .withConsistentRead(false);

List<PostedByMessage> iList = mapper.scan(PostedByMessage.class, scanExpression);
Iterator<PostedByMessage> indexItems = iList.iterator();
```

scanPage

Verifica uma tabela ou um índice secundário e retorna uma única página de resultados correspondentes. Como com o método `scan`, você pode especificar opcionalmente um `FilterExpression` para filtrar o conjunto de resultados. No entanto, `scanPage` retorna somente a primeira “página” de dados, ou seja, a quantidade de dados que caberá em 1 MB.

parallelScan

Realiza uma verificação paralela de uma tabela inteira ou de um índice secundário. Você especifica um número de segmentos lógicos para a tabela, juntamente com uma expressão de verificação para filtrar os resultados. O `parallelScan` divide a tarefa de verificação entre vários trabalhadores, um para cada segmento lógico. Por sua vez, esses trabalhadores processam os dados em paralelo e retornam os resultados.

O exemplo de código Java a seguir realiza uma verificação paralela na tabela `Product`.

```
int numberOfThreads = 4;

Map<String, AttributeValue> eav = new HashMap<String, AttributeValue>();
eav.put(":n", new AttributeValue().withN("100"));

DynamoDBScanExpression scanExpression = new DynamoDBScanExpression()
    .withFilterExpression("Price <= :n")
    .withExpressionAttributeValues(eav);

List<Product> scanResult = mapper.parallelScan(Product.class, scanExpression,
    numberOfThreads);
```

Para obter um exemplo de código Java ilustrando o uso de `parallelScan`, consulte [Exemplo: Consulta e verificação \(p. 254\)](#).

batchSave

Salva os objetos em uma ou mais tabelas usando uma ou mais chamadas para o método `AmazonDynamoDB.batchWriteItem`. Esse método não fornece garantias de transação.

O código Java a seguir salva dois itens (livros) na tabela `ProductCatalog`.

```
Book book1 = new Book();
book1.id = 901;
book1.productCategory = "Book";
book1.title = "Book 901 Title";

Book book2 = new Book();
book2.id = 902;
book2.productCategory = "Book";
book2.title = "Book 902 Title";

mapper.batchSave(Arrays.asList(book1, book2));
```

batchLoad

Recupera vários itens de uma ou mais tabelas usando suas chaves primárias.

O seguinte código Java recupera dois itens de duas tabelas diferentes.

```
ArrayList<Object> itemsToGet = new ArrayList<Object>();

ForumItem forumItem = new ForumItem();
forumItem.setForumName("Amazon DynamoDB");
itemsToGet.add(forumItem);

ThreadItem threadItem = new ThreadItem();
threadItem.setForumName("Amazon DynamoDB");
threadItem.setSubject("Amazon DynamoDB thread 1 message text");
itemsToGet.add(threadItem);

Map<String, List<Object>> items = mapper.batchLoad(itemsToGet);
```

batchDelete

Exclui objetos de uma ou mais tabelas usando uma ou mais chamadas para o método `AmazonDynamoDB.batchWriteItem`. Esse método não fornece garantias de transação.

O código Java a seguir exclui dois itens (livros) na tabela `ProductCatalog`.

```
Book book1 = mapper.load(Book.class, 901);
Book book2 = mapper.load(Book.class, 902);
mapper.batchDelete(Arrays.asList(book1, book2));
```

batchWrite

Salva e exclui objetos em/de uma ou mais tabelas usando uma ou mais chamadas para o método `AmazonDynamoDB.batchWriteItem`. Esse método não oferece garantias de transação ou suporte para versionamento (inserções ou exclusões condicionais).

O código Java a seguir grava um novo item na tabela `Forum`, grava um novo item na tabela `Thread` e exclui um item da tabela `ProductCatalog`.

```
// Create a Forum item to save
Forum forumItem = new Forum();
forumItem.name = "Test BatchWrite Forum";

// Create a Thread item to save
Thread threadItem = new Thread();
threadItem.forumName = "AmazonDynamoDB";
```

```
threadItem.subject = "My sample question";

// Load a ProductCatalog item to delete
Book book3 = mapper.load(Book.class, 903);

List<Object> objectsToWrite = Arrays.asList(forumItem, threadItem);
List<Book> objectsToDelete = Arrays.asList(book3);

mapper.batchWrite(objectsToWrite, objectsToDelete);
```

transactionWrite

Salva e exclui objetos em/de uma ou mais tabelas usando uma chamada para o método `AmazonDynamoDB.transactWriteItems`.

Para obter uma lista de exceções específicas de transação, consulte [Erros TransactWriteItems](#).

Para obter mais informações sobre transações do DynamoDB e as garantias de ACID (atomicidade, consistência, isolamento e durabilidade) fornecidas, consulte [Transações do Amazon DynamoDB](#).

Note

Esse método não oferece suporte ao seguinte:

- [DynamoDBMapperConfig.SaveBehavior](#).

O código Java a seguir grava um novo item em cada uma das tabelas `Forum` e `Thread`, de forma transacional.

```
Thread s3ForumThread = new Thread();
s3ForumThread.forumName = "S3 Forum";
s3ForumThread.subject = "Sample Subject 1";
s3ForumThread.message = "Sample Question 1";

Forum s3Forum = new Forum();
s3Forum.name = "S3 Forum";
s3Forum.category = "Amazon Web Services";
s3Forum.threads = 1;

TransactionWriteRequest transactionWriteRequest = new TransactionWriteRequest();
transactionWriteRequest.addPut(s3Forum);
transactionWriteRequest.addPut(s3ForumThread);
mapper.transactionWrite(transactionWriteRequest);
```

transactionLoad

Carrega objetos de uma ou mais tabelas usando uma chamada para o método `AmazonDynamoDB.transactGetItems`.

Para obter uma lista de exceções específicas de transação, consulte [Erros TransactGetItems](#).

Para obter mais informações sobre transações do DynamoDB e as garantias de ACID (atomicidade, consistência, isolamento e durabilidade) fornecidas, consulte [Transações do Amazon DynamoDB](#).

O código Java a seguir carrega um item de cada uma das tabelas `Forum` e `Thread`, de forma transacional.

```
Forum dynamodbForum = new Forum();
dynamodbForum.name = "DynamoDB Forum";
```

```
Thread dynamodbForumThread = new Thread();
dynamodbForumThread.forumName = "DynamoDB Forum";

TransactionLoadRequest transactionLoadRequest = new TransactionLoadRequest();
transactionLoadRequest.addLoad(dynamodbForum);
transactionLoadRequest.addLoad(dynamodbForumThread);
mapper.transactionLoad(transactionLoadRequest);
```

count

Avalia a expressão de verificação especificada e retorna a contagem de itens correspondentes. Dados de itens não são retornados.

generateCreateTableRequest

Analisa uma classe POJO que representa uma tabela do DynamoDB e retorna umCreateTableRequestPara essa tabela.

createS3Link

Cria um link para um objeto no Amazon S3. Você deve especificar um nome de bucket e um nome de chave que identifique exclusivamente o objeto no bucket.

Para usar `createS3Link`, a sua classe de mapeador deve definir métodos getter e setter. O exemplo de código a seguir ilustra isso, adicionando um novo atributo e métodos getter/setter à classe `CatalogItem`:

```
@DynamoDBTable(tableName="ProductCatalog")
public class CatalogItem {

    ...
    public S3Link productImage;
    ...
    @DynamoDBAttribute(attributeName = "ProductImage")
    public S3Link getProductImage() {
        return productImage;
    }
    public void setProductImage(S3Link productImage) {
        this.productImage = productImage;
    }
    ...
}
```

O código Java a seguir define um novo item a ser gravado na tabela `Product`. O item inclui um link para uma imagem do produto. Os dados da imagem são carregados no Amazon S3.

```
CatalogItem item = new CatalogItem();

item.id = 150;
item.title = "Book 150 Title";

String myS3Bucket = "myS3bucket";
String myS3Key = "productImages/book_150_cover.jpg";
item.setProductImage(mapper.createS3Link(myS3Bucket, myS3Key));

item.getProductImage().uploadFrom(new File("/file/path/book_150_cover.jpg"));
```

```
mapper.save(item);
```

OS3LinkA classe fornece muitos outros métodos para manipular objetos no Amazon S3. Para obter mais informações, consulte os [Javadoc para S3Link](#).

getS3ClientCache

Retorna o subjacenteS3ClientCachePara acessar o Amazon S3. Um s3ClientCache é um Mapa inteligente para objetos AmazonS3Client. Se você tiver vários clientes, ums3ClientCache pode ajudá-lo a manter os clientes organizados peloAWSRegião e pode criar novos clientes do Amazon S3 sob demanda.

Definições de configuração opcionais para DynamoDBMapper

Quando você cria uma instância de `DynamoDBMapper`, ela tem certos comportamentos padrão que podem ser substituídos com o uso da classe `DynamoDBMapperConfig`.

O seguinte trecho de código cria um `DynamoDBMapper` com configurações personalizadas:

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();

DynamoDBMapperConfig mapperConfig = DynamoDBMapperConfig.builder()
    .withSaveBehavior(DynamoDBMapperConfig.SaveBehavior.CLOBBER)
    .withConsistentReads(DynamoDBMapperConfig.ConsistentReads.CONSISTENT)
    .withTableNameOverride(null)

    .withPaginationLoadingStrategy(DynamoDBMapperConfig.PaginationLoadingStrategy.EAGER_LOADING)
    .build();

DynamoDBMapper mapper = new DynamoDBMapper(client, mapperConfig);
```

Para obter mais informações, consulte[DynamoDBMapperConfignoAWS SDK for JavaReferência de API do](#).

Como alternativa, você pode usar os seguintes argumentos para uma instância de `DynamoDBMapperConfig`:

- Um valor de enumeração `DynamoDBMapperConfig.ConsistentReads`:
 - `EVENTUAL` — a instância de mapeador usa uma solicitação de leitura eventualmente consistente.
 - `CONSISTENT`— a instância de recurso de mapeamento usa uma solicitação de leitura fortemente consistente. É possível usar essa configuração opcional com operações `load`, `query` ou `scan`. Leituras fortemente consistentes têm implicações de desempenho e cobrança. Consulte o [DynamoDB](#)[Página de detalhes do produto](#)Para obter mais informações.

Se você não especificar uma configuração de consistência de leitura para sua instância de mapeador, o padrão será `EVENTUAL`.

- Um valor de enumeração `DynamoDBMapperConfig.PaginationLoadingStrategy` — controla como a instância de mapeador processa uma lista paginada de dados, como os resultados de um `query` ou `scan`:
 - `LAZY_LOADING`— a instância de mapeador carrega dados quando possível e mantém todos os resultados carregados na memória.
 - `EAGER_LOADING` — a instância de mapeador carrega os dados assim que a lista é inicializada.
 - `ITERATION_ONLY`— você só pode usar um iterador para ler a partir da lista. Durante a iteração, a lista limpará todos os resultados anteriores antes de carregar a próxima página e, portanto, ela manterá no máximo uma página dos resultados carregados na memória. Isso também significa que a lista só

pode ser iterada uma vez. Essa estratégia é recomendada ao lidar com itens grandes, a fim de reduzir a sobrecarga de memória.

Se você não especificar uma estratégia de carregamento de paginação para a sua instância de mapeador, o padrão será `LAZY_LOADING`.

- Um valor de enumeração `DynamoDBMapperConfig.SaveBehavior` — especifica como a instância de mapeador deve lidar com atributos durante operações de salvamento:
 - `UPDATE`— durante uma operação de salvamento, todos os atributos modelados são atualizados, enquanto os atributos não modelados não são afetados. Tipos de números primitivos (`byte`, `int`, `long`) são definidos como 0. Tipos de objetos são definidos como nulos.
 - `CLOBBER`— limpa e substitui todos os atributos, incluindo os não modelados, durante uma operação de salvamento. Isso é feito excluindo-se o item e o recriando. Restrições de campo com versionamento também são desconsideradas.

Se você não especificar o comportamento de salvamento para sua instância de mapeador, o padrão será `UPDATE`.

Note

As operações transacionais do `DynamoDBMapper` não são compatíveis com a enumeração do `DynamoDBMapperConfig.SaveBehavior`.

- `ADynamoDBMapperConfig.TableNameOverride` — instrui a instância de mapeador a ignorar o nome de tabela especificado pelo `DynamoDBTable` Anotação e, em vez disso, usar um nome de tabela diferente que você fornece. Isso é útil ao particionar dados em várias tabelas no tempo de execução.

Você pode substituir o objeto de configuração padrão para `DynamoDBMapper` por operação, conforme necessário.

Exemplo: Operações de CRUD

O exemplo de código Java a seguir declara uma classe `CatalogItem` que tem as propriedades `Id`, `Title`, `ISBN` e `Authors`. Ele usa as anotações para mapear essas propriedades para o `ProductCatalog` no DynamoDB. O exemplo usa `DynamoDBMapper` para salvar um objeto de livro, recuperá-lo, atualizá-lo e excluir o item de livro.

Note

Este exemplo de código pressupõe que você já carregou dados no DynamoDB para sua conta seguindo as instruções no [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#) seção.

Para obter instruções detalhadas sobre como executar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */
```

```
package com.amazonaws.codesamples.datamodeling;

import java.io.IOException;
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBAttribute;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;

public class DynamoDBMapperCRUDExample {

    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();

    public static void main(String[] args) throws IOException {
        testCRUDOperations();
        System.out.println("Example complete!");
    }

    @DynamoDBTable(tableName = "ProductCatalog")
    public static class CatalogItem {
        private Integer id;
        private String title;
        private String ISBN;
        private Set<String> bookAuthors;

        // Partition key
        @DynamoDBHashKey(attributeName = "Id")
        public Integer getId() {
            return id;
        }

        public void setId(Integer id) {
            this.id = id;
        }

        @DynamoDBAttribute(attributeName = "Title")
        public String getTitle() {
            return title;
        }

        public void setTitle(String title) {
            this.title = title;
        }

        @DynamoDBAttribute(attributeName = "ISBN")
        public String getISBN() {
            return ISBN;
        }

        public void setISBN(String ISBN) {
            this.ISBN = ISBN;
        }

        @DynamoDBAttribute(attributeName = "Authors")
        public Set<String> getBookAuthors() {
            return bookAuthors;
        }

        public void setBookAuthors(Set<String> bookAuthors) {
            this.bookAuthors = bookAuthors;
        }
    }
}
```

```
    }

    @Override
    public String toString() {
        return "Book [ISBN=" + ISBN + ", bookAuthors=" + bookAuthors + ", id=" + id +
        ", title=" + title + "]";
    }
}

private static void testCRUDOperations() {

    CatalogItem item = new CatalogItem();
    item.setId(601);
    item.setTitle("Book 601");
    item.setISBN("611-111111111");
    item.setBookAuthors(new HashSet<String>(Arrays.asList("Author1", "Author2")));

    // Save the item (book).
    DynamoDBMapper mapper = new DynamoDBMapper(client);
    mapper.save(item);

    // Retrieve the item.
    CatalogItem itemRetrieved = mapper.load(CatalogItem.class, 601);
    System.out.println("Item retrieved:");
    System.out.println(itemRetrieved);

    // Update the item.
    itemRetrieved.setISBN("622-222222222");
    itemRetrieved.setBookAuthors(new HashSet<String>(Arrays.asList("Author1",
"Author3")));
    mapper.save(itemRetrieved);
    System.out.println("Item updated:");
    System.out.println(itemRetrieved);

    // Retrieve the updated item.
    DynamoDBMapperConfig config = DynamoDBMapperConfig.builder()
        .withConsistentReads(DynamoDBMapperConfig.ConsistentReads.CONSISTENT)
        .build();
    CatalogItem updatedItem = mapper.load(CatalogItem.class, 601, config);
    System.out.println("Retrieved the previously updated item:");
    System.out.println(updatedItem);

    // Delete the item.
    mapper.delete(updatedItem);

    // Try to retrieve deleted item.
    CatalogItem deletedItem = mapper.load(CatalogItem.class, updatedItem.getId(),
config);
    if (deletedItem == null) {
        System.out.println("Done - Sample item is deleted.");
    }
}
}
```

Exemplo: Operações de gravação em Batch

O exemplo de código Java a seguir declara Book, Forum, Thread, e Reply e os mapeia para as tabelas do Amazon DynamoDB usando as classes DynamoDBMapper Classe.

O código ilustra as seguintes operações de gravação em lote:

- batchSave para inserir itens de livro na tabela ProductCatalog.
- batchDelete para excluir itens da tabela ProductCatalog.

- batchWrite para inserir e excluir itens de livro das tabelas Forum e Thread.

Para obter mais informações sobre as tabelas usadas neste exemplo, consulte [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#). Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

Example

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples.datamodeling;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBAttribute;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBRangeKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;

public class DynamoDBMapperBatchWriteExample {

    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
    static SimpleDateFormat dateFormatter = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss.SSS'Z'");

    public static void main(String[] args) throws Exception {
        try {

            DynamoDBMapper mapper = new DynamoDBMapper(client);

            testBatchSave(mapper);
            testBatchDelete(mapper);
            testBatchWrite(mapper);

            System.out.println("Example complete!");

        }
        catch (Throwable t) {
            System.err.println("Error running the DynamoDBMapperBatchWriteExample: " + t);
            t.printStackTrace();
        }
    }
}
```

```
}

private static void testBatchSave(DynamoDBMapper mapper) {

    Book book1 = new Book();
    book1.id = 901;
    book1.inPublication = true;
    book1.ISBN = "902-11-11-1111";
    book1.pageCount = 100;
    book1.price = 10;
    book1.productCategory = "Book";
    book1.title = "My book created in batch write";

    Book book2 = new Book();
    book2.id = 902;
    book2.inPublication = true;
    book2.ISBN = "902-11-12-1111";
    book2.pageCount = 200;
    book2.price = 20;
    book2.productCategory = "Book";
    book2.title = "My second book created in batch write";

    Book book3 = new Book();
    book3.id = 903;
    book3.inPublication = false;
    book3.ISBN = "902-11-13-1111";
    book3.pageCount = 300;
    book3.price = 25;
    book3.productCategory = "Book";
    book3.title = "My third book created in batch write";

    System.out.println("Adding three books to ProductCatalog table.");
    mapper.batchSave(Arrays.asList(book1, book2, book3));
}

private static void testBatchDelete(DynamoDBMapper mapper) {

    Book book1 = mapper.load(Book.class, 901);
    Book book2 = mapper.load(Book.class, 902);
    System.out.println("Deleting two books from the ProductCatalog table.");
    mapper.batchDelete(Arrays.asList(book1, book2));
}

private static void testBatchWrite(DynamoDBMapper mapper) {

    // Create Forum item to save
    Forum forumItem = new Forum();
    forumItem.name = "Test BatchWrite Forum";
    forumItem.threads = 0;
    forumItem.category = "Amazon Web Services";

    // Create Thread item to save
    Thread threadItem = new Thread();
    threadItem.forumName = "AmazonDynamoDB";
    threadItem.subject = "My sample question";
    threadItem.message = "BatchWrite message";
    List<String> tags = new ArrayList<String>();
    tags.add("batch operations");
    tags.add("write");
    threadItem.tags = new HashSet<String>(tags);

    // Load ProductCatalog item to delete
    Book book3 = mapper.load(Book.class, 903);

    List<Object> objectsToWrite = Arrays.asList(forumItem, threadItem);
    List<Book> objectsToDelete = Arrays.asList(book3);
```

```
DynamoDBMapperConfig config = DynamoDBMapperConfig.builder()
    .withSaveBehavior(DynamoDBMapperConfig.SaveBehavior.CLOBBER)
    .build();

    mapper.batchWrite(objectsToWrite, objectsToDelete, config);
}

@DynamoDBTable(tableName = "ProductCatalog")
public static class Book {
    private int id;
    private String title;
    private String ISBN;
    private int price;
    private int pageCount;
    private String productCategory;
    private boolean inPublication;

    // Partition key
    @DynamoDBHashKey(attributeName = "Id")
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @DynamoDBAttribute(attributeName = "Title")
    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    @DynamoDBAttribute(attributeName = "ISBN")
    public String getISBN() {
        return ISBN;
    }

    public void setISBN(String ISBN) {
        this.ISBN = ISBN;
    }

    @DynamoDBAttribute(attributeName = "Price")
    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    @DynamoDBAttribute(attributeName = "PageCount")
    public int getPageCount() {
        return pageCount;
    }

    public void setPageCount(int pageCount) {
        this.pageCount = pageCount;
    }

    @DynamoDBAttribute(attributeName = "ProductCategory")
    public String getProductCategory() {
```

```
        return productCategory;
    }

    public void setProductCategory(String productCategory) {
        this.productCategory = productCategory;
    }

    @DynamoDBAttribute(attributeName = "InPublication")
    public boolean getInPublication() {
        return inPublication;
    }

    public void setInPublication(boolean inPublication) {
        this.inPublication = inPublication;
    }

    @Override
    public String toString() {
        return "Book [ISBN=" + ISBN + ", price=" + price + ", product category=" +
productCategory + ", id=" + id
        + ", title=" + title + "]";
    }

}

@dynamoDBTable(tableName = "Reply")
public static class Reply {
    private String id;
    private String replyDateTime;
    private String message;
    private String postedBy;

    // Partition key
    @DynamoDBHashKey(attributeName = "Id")
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    // Sort key
    @DynamoDBRangeKey(attributeName = "ReplyDateTime")
    public String getReplyDateTime() {
        return replyDateTime;
    }

    public void setReplyDateTime(String replyDateTime) {
        this.replyDateTime = replyDateTime;
    }

    @DynamoDBAttribute(attributeName = "Message")
    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    @DynamoDBAttribute(attributeName = "PostedBy")
    public String getPostedBy() {
        return postedBy;
    }
}
```

```
public void setPostedBy(String postedBy) {
    this.postedBy = postedBy;
}

}

@DynamoDBTable(tableName = "Thread")
public static class Thread {
    private String forumName;
    private String subject;
    private String message;
    private String lastPostedDateTime;
    private String lastPostedBy;
    private Set<String> tags;
    private int answered;
    private int views;
    private int replies;

    // Partition key
    @DynamoDBHashKey(attributeName = "ForumName")
    public String getForumName() {
        return forumName;
    }

    public void setForumName(String forumName) {
        this.forumName = forumName;
    }

    // Sort key
    @DynamoDBRangeKey(attributeName = "Subject")
    public String getSubject() {
        return subject;
    }

    public void setSubject(String subject) {
        this.subject = subject;
    }

    @DynamoDBAttribute(attributeName = "Message")
    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    @DynamoDBAttribute(attributeName = "LastPostedDateTime")
    public String getLastPostedDateTime() {
        return lastPostedDateTime;
    }

    public void setLastPostedDateTime(String lastPostedDateTime) {
        this.lastPostedDateTime = lastPostedDateTime;
    }

    @DynamoDBAttribute(attributeName = "LastPostedBy")
    public String getLastPostedBy() {
        return lastPostedBy;
    }

    public void setLastPostedBy(String lastPostedBy) {
        this.lastPostedBy = lastPostedBy;
    }

    @DynamoDBAttribute(attributeName = "Tags")
    public Set<String> getTags() {
```

```
        return tags;
    }

    public void setTags(Set<String> tags) {
        this.tags = tags;
    }

    @DynamoDBAttribute(attributeName = "Answered")
    public int getAnswered() {
        return answered;
    }

    public void setAnswered(int answered) {
        this.answered = answered;
    }

    @DynamoDBAttribute(attributeName = "Views")
    public int getViews() {
        return views;
    }

    public void setViews(int views) {
        this.views = views;
    }

    @DynamoDBAttribute(attributeName = "Replies")
    public int getReplies() {
        return replies;
    }

    public void setReplies(int replies) {
        this.replies = replies;
    }

}

{@DynamoDBTable(tableName = "Forum")}
public static class Forum {
    private String name;
    private String category;
    private int threads;

    // Partition key
    @DynamoDBHashKey(attributeName = "Name")
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @DynamoDBAttribute(attributeName = "Category")
    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    @DynamoDBAttribute(attributeName = "Threads")
    public int getThreads() {
        return threads;
    }
}
```

```
    public void setThreads(int threads) {
        this.threads = threads;
    }
}
```

Exemplo: Consulta e verificação

O exemplo Java nesta seção define as seguintes classes e as mapeia para as tabelas no Amazon DynamoDB. Para obter mais informações sobre como criar tabelas de exemplo, consulte [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#).

- A classe `Book` mapeia para a tabela `ProductCatalog`
- As classes `Forum`, `Thread` e `Reply` mapeiam para as tabelas com o mesmo nome.

Em seguida, o exemplo executa as seguintes operações de consulta e verificação usando `DynamoDBMapper` instância.

- Obtenha um livro por `Id`.

A tabela `ProductCatalog` tem `Id` como sua chave primária. Ela não tem uma chave de classificação como parte de sua chave primária. Portanto, você não pode consultar a tabela. É possível obter um item usando o valor `Id`.

- Execute as consultas a seguir no `ReplyTabela` do.

A chave primária da tabela `Reply` composta por atributos `Id` e `ReplyDateTime`. `ReplyDateTime` é uma chave de classificação. Portanto, é possível consultar essa tabela.

- Localizar respostas para um tópico de fórum postado nos últimos 15 dias.
- Localizar respostas para um tópico de fórum postado em um intervalo de datas específico.
- Verifique a tabela `ProductCatalog` para localizar livros cujo preço seja menor que um valor especificado.

Por motivos de desempenho, é necessário usar uma operação de consulta, em vez de uma operação de verificação. No entanto, em algumas ocasiões, talvez você precise verificar uma tabela. Vamos supor que tenha ocorrido um erro de entrada de dados e que um dos preços de livros tenha sido definido como menor que 0. Este exemplo verifica a tabela `ProductCategory` para localizar itens de livro (`ProductCategory` é livro) cujos preços são menores que 0.

- Realize uma verificação paralela da tabela `ProductCatalog` para localizar bicicletas de um tipo específico.

Note

Este exemplo de código pressupõe que você já carregou dados no DynamoDB para sua conta seguindo as instruções no [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#) seção.

Para obter instruções detalhadas sobre como executar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

Example

```
/**
```

```
* Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
*  
* This file is licensed under the Apache License, Version 2.0 (the "License").  
* You may not use this file except in compliance with the License. A copy of  
* the License is located at  
*  
* http://aws.amazon.com/apache2.0/  
*  
* This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
* CONDITIONS OF ANY KIND, either express or implied. See the License for the  
* specific language governing permissions and limitations under the License.  
*/  
  
package com.amazonaws.codesamples.datamodeling;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
import java.util.Set;  
import java.util.TimeZone;  
  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBAttribute;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBQueryExpression;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBRangeKey;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBScanExpression;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;  
import com.amazonaws.services.dynamodbv2.model.AttributeValue;  
  
public class DynamoDBMapperQueryScanExample {  
  
    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();  
  
    public static void main(String[] args) throws Exception {  
        try {  
  
            DynamoDBMapper mapper = new DynamoDBMapper(client);  
  
            // Get a book - Id=101  
            GetBook(mapper, 101);  
            // Sample forum and thread to test queries.  
            String forumName = "Amazon DynamoDB";  
            String threadSubject = "DynamoDB Thread 1";  
            // Sample queries.  
            FindRepliesInLast15Days(mapper, forumName, threadSubject);  
            FindRepliesPostedWithinTimePeriod(mapper, forumName, threadSubject);  
  
            // Scan a table and find book items priced less than specified  
            // value.  
            FindBooksPricedLessThanSpecifiedValue(mapper, "20");  
  
            // Scan a table with multiple threads and find bicycle items with a  
            // specified bicycle type  
            int numberOfThreads = 16;  
            FindBicyclesOfSpecificTypeWithMultipleThreads(mapper, numberOfThreads, "Road");  
  
            System.out.println("Example complete!");  
        }  
        catch (Throwable t) {
```

```
        System.err.println("Error running the DynamoDBMapperQueryScanExample: " + t);
        t.printStackTrace();
    }

    private static void GetBook(DynamoDBMapper mapper, int id) throws Exception {
        System.out.println("GetBook: Get book Id='101' ");
        System.out.println("Book table has no sort key. You can do GetItem, but not
Query.");
        Book book = mapper.load(Book.class, id);
        System.out.format("Id = %s Title = %s, ISBN = %s %n", book.getId(),
book.getTitle(), book.getISBN());
    }

    private static void FindRepliesInLast15Days(DynamoDBMapper mapper, String forumName,
String threadSubject)
        throws Exception {
        System.out.println("FindRepliesInLast15Days: Replies within last 15 days.");

        String partitionKey = forumName + "#" + threadSubject;

        long twoWeeksAgoMilli = (new Date()).getTime() - (15L * 24L * 60L * 60L * 1000L);
        Date twoWeeksAgo = new Date();
        twoWeeksAgo.setTime(twoWeeksAgoMilli);
        SimpleDateFormat dateFormatter = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss.SSS'Z'");
        dateFormatter.setTimeZone(TimeZone.getTimeZone("UTC"));
        String twoWeeksAgoStr = dateFormatter.format(twoWeeksAgo);

        Map<String, AttributeValue> eav = new HashMap<String, AttributeValue>();
        eav.put(":val1", new AttributeValue().withS(partitionKey));
        eav.put(":val2", new AttributeValue().withS(twoWeeksAgoStr.toString()));

        DynamoDBQueryExpression<Reply> queryExpression = new
DynamoDBQueryExpression<Reply>()
            .withKeyConditionExpression("Id = :val1 and ReplyDateTime
> :val2").withExpressionAttributeValues(eav);

        List<Reply> latestReplies = mapper.query(Reply.class, queryExpression);

        for (Reply reply : latestReplies) {
            System.out.format("Id=%s, Message=%s, PostedBy=%s %n, ReplyDateTime=%s %n",
reply.getId(),
                reply.getMessage(), reply.getPostedBy(), reply.getReplyDateTime());
        }
    }

    private static void FindRepliesPostedWithinTimePeriod(DynamoDBMapper mapper, String
forumName, String threadSubject)
        throws Exception {
        String partitionKey = forumName + "#" + threadSubject;

        System.out.println(
            "FindRepliesPostedWithinTimePeriod: Find replies for thread Message = 'DynamoDB
Thread 2' posted within a period.");
        long startDateMilli = (new Date()).getTime() - (14L * 24L * 60L * 60L * 1000L); // Two
                                                                 // weeks
        long endDateMilli = (new Date()).getTime() - (7L * 24L * 60L * 60L * 1000L); // One
                                                                 // week
        long ago = (new Date()).getTime() - (1L * 24L * 60L * 60L * 1000L); // One
                                                                 // day
    }
}
```

```
SimpleDateFormat dateFormatter = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss.SSS'Z'");
dateFormatter.setTimeZone(TimeZone.getTimeZone("UTC"));
String startDate = dateFormatter.format(startDateMilli);
String endDate = dateFormatter.format(endDateMilli);

Map<String,AttributeValue> eav = new HashMap<String,AttributeValue>();
eav.put(":val1", new AttributeValue().withS(partitionKey));
eav.put(":val2", new AttributeValue().withS(startDate));
eav.put(":val3", new AttributeValue().withS(endDate));

DynamoDBQueryExpression<Reply> queryExpression = new
DynamoDBQueryExpression<Reply>()
    .withKeyConditionExpression("Id = :val1 and ReplyDateTime between :val2
and :val3")
    .withExpressionAttributeValues(eav);

List<Reply> betweenReplies = mapper.query(Reply.class, queryExpression);

for (Reply reply : betweenReplies) {
    System.out.format("Id=%s, Message=%s, PostedBy=%s %n, PostedDateTime=%s %n",
    reply.getId(),
        reply.getMessage(), reply.getPostedBy(), reply.getReplyDateTime());
}

private static void FindBooksPricedLessThanSpecifiedValue(DynamoDBMapper mapper, String
value) throws Exception {

    System.out.println("FindBooksPricedLessThanSpecifiedValue: Scan ProductCatalog.");

    Map<String,AttributeValue> eav = new HashMap<String,AttributeValue>();
    eav.put(":val1", new AttributeValue().withN(value));
    eav.put(":val2", new AttributeValue().withS("Book"));

    DynamoDBScanExpression scanExpression = new DynamoDBScanExpression()
        .withFilterExpression("Price < :val1 and ProductCategory
= :val2").withExpressionAttributeValues(eav);

    List<Book> scanResult = mapper.scan(Book.class, scanExpression);

    for (Book book : scanResult) {
        System.out.println(book);
    }
}

private static void FindBicyclesOfSpecificTypeWithMultipleThreads(DynamoDBMapper
mapper, int numberOfThreads,
String bicycleType) throws Exception {

    System.out.println("FindBicyclesOfSpecificTypeWithMultipleThreads: Scan
ProductCatalog With Multiple Threads.");
    Map<String,AttributeValue> eav = new HashMap<String,AttributeValue>();
    eav.put(":val1", new AttributeValue().withS("Bicycle"));
    eav.put(":val2", new AttributeValue().withS(bicycleType));

    DynamoDBScanExpression scanExpression = new DynamoDBScanExpression()
        .withFilterExpression("ProductCategory = :val1 and BicycleType
= :val2").withExpressionAttributeValues(eav);

    List<Bicycle> scanResult = mapper.parallelScan(Bicycle.class, scanExpression,
numberOfThreads);
    for (Bicycle bicycle : scanResult) {
        System.out.println(bicycle);
    }
}
```

```
}  
  
@DynamoDBTable(tableName = "ProductCatalog")  
public static class Book {  
    private int id;  
    private String title;  
    private String ISBN;  
    private int price;  
    private int pageCount;  
    private String productCategory;  
    private boolean inPublication;  
  
    @DynamoDBHashKey(attributeName = "Id")  
    public int getId() {  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
  
    @DynamoDBAttribute(attributeName = "Title")  
    public String getTitle() {  
        return title;  
    }  
  
    public void setTitle(String title) {  
        this.title = title;  
    }  
  
    @DynamoDBAttribute(attributeName = "ISBN")  
    public String getISBN() {  
        return ISBN;  
    }  
  
    public void setISBN(String ISBN) {  
        this.ISBN = ISBN;  
    }  
  
    @DynamoDBAttribute(attributeName = "Price")  
    public int getPrice() {  
        return price;  
    }  
  
    public void setPrice(int price) {  
        this.price = price;  
    }  
  
    @DynamoDBAttribute(attributeName = "PageCount")  
    public int getPageCount() {  
        return pageCount;  
    }  
  
    public void setPageCount(int pageCount) {  
        this.pageCount = pageCount;  
    }  
  
    @DynamoDBAttribute(attributeName = "ProductCategory")  
    public String getProductCategory() {  
        return productCategory;  
    }  
  
    public void setProductCategory(String productCategory) {  
        this.productCategory = productCategory;  
    }
```

```
@DynamoDBAttribute(attributeName = "InPublication")
public boolean getInPublication() {
    return inPublication;
}

public void setInPublication(boolean inPublication) {
    this.inPublication = inPublication;
}

@Override
public String toString() {
    return "Book [ISBN=" + ISBN + ", price=" + price + ", product category=" +
productCategory + ", id=" + id
        + ", title=" + title + "]";
}

}

@DynamoDBTable(tableName = "ProductCatalog")
public static class Bicycle {
    private int id;
    private String title;
    private String description;
    private String bicycleType;
    private String brand;
    private int price;
    private List<String> color;
    private String productCategory;

    @DynamoDBHashKey(attributeName = "Id")
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @DynamoDBAttribute(attributeName = "Title")
    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    @DynamoDBAttribute(attributeName = "Description")
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    @DynamoDBAttribute(attributeName = "BicycleType")
    public String getBicycleType() {
        return bicycleType;
    }

    public void setBicycleType(String bicycleType) {
        this.bicycleType = bicycleType;
    }

    @DynamoDBAttribute(attributeName = "Brand")
```

```
public String getBrand() {
    return brand;
}

public void setBrand(String brand) {
    this.brand = brand;
}

@DynamoDBAttribute(attributeName = "Price")
public int getPrice() {
    return price;
}

public void setPrice(int price) {
    this.price = price;
}

@DynamoDBAttribute(attributeName = "Color")
public List<String> getColor() {
    return color;
}

public void setColor(List<String> color) {
    this.color = color;
}

@DynamoDBAttribute(attributeName = "ProductCategory")
public String getProductCategory() {
    return productCategory;
}

public void setProductCategory(String productCategory) {
    this.productCategory = productCategory;
}

@Override
public String toString() {
    return "Bicycle [Type=" + bicycleType + ", color=" + color + ", price=" + price
+ ", product category="
        + productCategory + ", id=" + id + ", title=" + title + "]";
}

}

@DynamoDBTable(tableName = "Reply")
public static class Reply {
    private String id;
    private String replyDateTime;
    private String message;
    private String postedBy;

    // Partition key
    @DynamoDBHashKey(attributeName = "Id")
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    // Range key
    @DynamoDBRangeKey(attributeName = "ReplyDateTime")
    public String getReplyDateTime() {
        return replyDateTime;
    }
}
```

```
public void setReplyDateTime(String replyDateTime) {
    this.replyDateTime = replyDateTime;
}

@DynamoDBAttribute(attributeName = "Message")
public String getMessage() {
    return message;
}

public void setMessage(String message) {
    this.message = message;
}

@DynamoDBAttribute(attributeName = "PostedBy")
public String getPostedBy() {
    return postedBy;
}

public void setPostedBy(String postedBy) {
    this.postedBy = postedBy;
}
}

@dynamoDBTable(tableName = "Thread")
public static class Thread {
    private String forumName;
    private String subject;
    private String message;
    private String lastPostedDateTime;
    private String lastPostedBy;
    private Set<String> tags;
    private int answered;
    private int views;
    private int replies;

    // Partition key
    @DynamoDBHashKey(attributeName = "ForumName")
    public String getForumName() {
        return forumName;
    }

    public void setForumName(String forumName) {
        this.forumName = forumName;
    }

    // Range key
    @DynamoDBRangeKey(attributeName = "Subject")
    public String getSubject() {
        return subject;
    }

    public void setSubject(String subject) {
        this.subject = subject;
    }

    @DynamoDBAttribute(attributeName = "Message")
    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    @DynamoDBAttribute(attributeName = "LastPostedDateTime")
```

```
public String getLastPostedDateTime() {
    return lastPostedDateTime;
}

public void setLastPostedDateTime(String lastPostedDateTime) {
    this.lastPostedDateTime = lastPostedDateTime;
}

@DynamoDBAttribute(attributeName = "LastPostedBy")
public String getLastPostedBy() {
    return lastPostedBy;
}

public void setLastPostedBy(String lastPostedBy) {
    this.lastPostedBy = lastPostedBy;
}

@DynamoDBAttribute(attributeName = "Tags")
public Set<String> getTags() {
    return tags;
}

public void setTags(Set<String> tags) {
    this.tags = tags;
}

@DynamoDBAttribute(attributeName = "Answered")
public int getAnswered() {
    return answered;
}

public void setAnswered(int answered) {
    this.answered = answered;
}

@DynamoDBAttribute(attributeName = "Views")
public int getViews() {
    return views;
}

public void setViews(int views) {
    this.views = views;
}

@DynamoDBAttribute(attributeName = "Replies")
public int getReplies() {
    return replies;
}

public void setReplies(int replies) {
    this.replies = replies;
}

}

@DynamoDBTable(tableName = "Forum")
public static class Forum {
    private String name;
    private String category;
    private int threads;

    @DynamoDBHashKey(attributeName = "Name")
    public String getName() {
        return name;
    }
}
```

```
public void setName(String name) {
    this.name = name;
}

@DynamoDBAttribute(attributeName = "Category")
public String getCategory() {
    return category;
}

public void setCategory(String category) {
    this.category = category;
}

@DynamoDBAttribute(attributeName = "Threads")
public int getThreads() {
    return threads;
}

public void setThreads(int threads) {
    this.threads = threads;
}
}
```

Exemplo: Operações de transação

O exemplo de código Java a seguir declara uma `Forum` e um `Thread` e os mapeia para as tabelas do DynamoDB usando a classe `DynamoDBMapper`.

O código ilustra as seguintes operações transacionais:

- `transactionWrite` para adicionar, atualizar e excluir vários itens de uma ou mais tabelas em uma transação.
- `transactionLoad` para recuperar vários itens de uma ou mais tabelas em uma transação.

Example

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

import java.io.IOException;
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;
import java.util.List;
import java.util.ArrayList;
import java.util.Map;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
```

```
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBAttribute;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapperConfig;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBRangeKey;
import com.amazonaws.services.dynamodbv2.datamodeling.TransactionWriteRequest;
import com.amazonaws.services.dynamodbv2.datamodeling.TransactionLoadRequest;
import com.amazonaws.services.dynamodbv2.model.TransactionCanceledException;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTransactionWriteExpression;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTransactionLoadExpression;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMappingException;
import com.amazonaws.services.dynamodbv2.model.ResourceNotFoundException;
import com.amazonaws.services.dynamodbv2.model.InternalServerErrorException;

public class DynamoDBMapperTransactionExample {

    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
    static DynamoDBMapper mapper;

    public static void main(String[] args) throws Exception {
        try {

            mapper = new DynamoDBMapper(client);

            testPutAndUpdateInTransactionWrite();
            testPutWithConditionalUpdateInTransactionWrite();
            testPutWithConditionCheckInTransactionWrite();
            testMixedOperationsInTransactionWrite();
            testTransactionLoadWithSave();
            testTransactionLoadWithTransactionWrite();
            System.out.println("Example complete");

        }
        catch (Throwable t) {
            System.err.println("Error running the DynamoDBMapperTransactionWriteExample: " +
+ t);
            t.printStackTrace();
        }
    }

    private static void testTransactionLoadWithSave() {
        // Create new Forum item for DynamoDB using save
        Forum dynamodbForum = new Forum();
        dynamodbForum.name = "DynamoDB Forum";
        dynamodbForum.category = "Amazon Web Services";
        dynamodbForum.threads = 0;
        mapper.save(dynamodbForum);

        // Add a thread to DynamoDB Forum
        Thread dynamodbForumThread = new Thread();
        dynamodbForumThread.forumName = "DynamoDB Forum";
        dynamodbForumThread.subject = "Sample Subject 1";
        dynamodbForumThread.message = "Sample Question 1";
        mapper.save(dynamodbForumThread);

        // Update DynamoDB Forum to reflect updated thread count
        dynamodbForum.threads = 1;
        mapper.save(dynamodbForum);

        // Read DynamoDB Forum item and Thread item at the same time in a serializable
manner
        TransactionLoadRequest transactionLoadRequest = new TransactionLoadRequest();

        // Read entire item for DynamoDB Forum
```

```
transactionLoadRequest.addLoad(dynamodbForum);

// Only read subject and message attributes from Thread item
DynamoDBTransactionLoadExpression loadExpressionForThread = new
DynamoDBTransactionLoadExpression()

.withProjectionExpression("Subject, Message");
transactionLoadRequest.addLoad(dynamodbForumThread, loadExpressionForThread);

// Loaded objects are guaranteed to be in same order as the order in which they are
// added to TransactionLoadRequest
List<Object> loadedObjects = executeTransactionLoad(transactionLoadRequest);
Forum loadedDynamoDBForum = (Forum) loadedObjects.get(0);
System.out.println("Forum: " + loadedDynamoDBForum.name);
System.out.println("Threads: " + loadedDynamoDBForum.threads);
Thread loadedDynamodbForumThread = (Thread) loadedObjects.get(1);
System.out.println("Subject: " + loadedDynamodbForumThread.subject);
System.out.println("Message: " + loadedDynamodbForumThread.message);
}

private static void testTransactionLoadWithTransactionWrite() {
    // Create new Forum item for DynamoDB using save
    Forum dynamodbForum = new Forum();
    dynamodbForum.name = "DynamoDB New Forum";
    dynamodbForum.category = "Amazon Web Services";
    dynamodbForum.threads = 0;
    mapper.save(dynamodbForum);

    // Update Forum item for DynamoDB and add a thread to DynamoDB Forum, in
    // an ACID manner using transactionWrite

    dynamodbForum.threads = 1;
    Thread dynamodbForumThread = new Thread();
    dynamodbForumThread.forumName = "DynamoDB New Forum";
    dynamodbForumThread.subject = "Sample Subject 2";
    dynamodbForumThread.message = "Sample Question 2";
    TransactionWriteRequest transactionWriteRequest = new TransactionWriteRequest();
    transactionWriteRequest.addPut(dynamodbForumThread);
    transactionWriteRequest.addUpdate(dynamodbForum);
    executeTransactionWrite(transactionWriteRequest);

    // Read DynamoDB Forum item and Thread item at the same time in a serializable
manner
    TransactionLoadRequest transactionLoadRequest = new TransactionLoadRequest();

    // Read entire item for DynamoDB Forum
    transactionLoadRequest.addLoad(dynamodbForum);

    // Only read subject and message attributes from Thread item
    DynamoDBTransactionLoadExpression loadExpressionForThread = new
DynamoDBTransactionLoadExpression()

.withProjectionExpression("Subject, Message");
transactionLoadRequest.addLoad(dynamodbForumThread, loadExpressionForThread);

    // Loaded objects are guaranteed to be in same order as the order in which they are
    // added to TransactionLoadRequest
    List<Object> loadedObjects = executeTransactionLoad(transactionLoadRequest);
    Forum loadedDynamoDBForum = (Forum) loadedObjects.get(0);
    System.out.println("Forum: " + loadedDynamoDBForum.name);
    System.out.println("Threads: " + loadedDynamoDBForum.threads);
    Thread loadedDynamodbForumThread = (Thread) loadedObjects.get(1);
    System.out.println("Subject: " + loadedDynamodbForumThread.subject);
    System.out.println("Message: " + loadedDynamodbForumThread.message);
}
```

```
private static void testPutAndUpdateInTransactionWrite() {
    // Create new Forum item for S3 using save
    Forum s3Forum = new Forum();
    s3Forum.name = "S3 Forum";
    s3Forum.category = "Core Amazon Web Services";
    s3Forum.threads = 0;
    mapper.save(s3Forum);

    // Update Forum item for S3 and Create new Forum item for DynamoDB using
    transactionWrite
    s3Forum.category = "Amazon Web Services";
    Forum dynamodbForum = new Forum();
    dynamodbForum.name = "DynamoDB Forum";
    dynamodbForum.category = "Amazon Web Services";
    dynamodbForum.threads = 0;
    TransactionWriteRequest transactionWriteRequest = new TransactionWriteRequest();
    transactionWriteRequest.addUpdate(s3Forum);
    transactionWriteRequest.addPut(dynamodbForum);
    executeTransactionWrite(transactionWriteRequest);
}

private static void testPutWithConditionalUpdateInTransactionWrite() {
    // Create new Thread item for DynamoDB forum and update thread count in DynamoDB
    forum
    // if the DynamoDB Forum exists
    Thread dynamodbForumThread = new Thread();
    dynamodbForumThread.forumName = "DynamoDB Forum";
    dynamodbForumThread.subject = "Sample Subject 1";
    dynamodbForumThread.message = "Sample Question 1";

    Forum dynamodbForum = new Forum();
    dynamodbForum.name = "DynamoDB Forum";
    dynamodbForum.category = "Amazon Web Services";
    dynamodbForum.threads = 1;

    DynamoDBTransactionWriteExpression transactionWriteExpression = new
    DynamoDBTransactionWriteExpression()

    .withConditionExpression("attribute_exists(Category)");

    TransactionWriteRequest transactionWriteRequest = new TransactionWriteRequest();
    transactionWriteRequest.addPut(dynamodbForumThread);
    transactionWriteRequest.addUpdate(dynamodbForum, transactionWriteExpression);
    executeTransactionWrite(transactionWriteRequest);
}

private static void testPutWithConditionCheckInTransactionWrite() {
    // Create new Thread item for DynamoDB forum and update thread count in DynamoDB
    forum if a thread already exists
    Thread dynamodbForumThread2 = new Thread();
    dynamodbForumThread2.forumName = "DynamoDB Forum";
    dynamodbForumThread2.subject = "Sample Subject 2";
    dynamodbForumThread2.message = "Sample Question 2";

    Thread dynamodbForumThread1 = new Thread();
    dynamodbForumThread1.forumName = "DynamoDB Forum";
    dynamodbForumThread1.subject = "Sample Subject 1";
    DynamoDBTransactionWriteExpression conditionExpressionForConditionCheck = new
    DynamoDBTransactionWriteExpression()

    .withConditionExpression("attribute_exists(Subject)");

    Forum dynamodbForum = new Forum();
    dynamodbForum.name = "DynamoDB Forum";
    dynamodbForum.category = "Amazon Web Services";
```

```
dynamodbForum.threads = 2;

TransactionWriteRequest transactionWriteRequest = new TransactionWriteRequest();
transactionWriteRequest.addPut(dynamodbForumThread2);
transactionWriteRequest.addConditionCheck(dynamodbForumThread1,
conditionExpressionForConditionCheck);
transactionWriteRequest.addUpdate(dynamodbForum);
executeTransactionWrite(transactionWriteRequest);
}

private static void testMixedOperationsInTransactionWrite() {
    // Create new Thread item for S3 forum and delete "Sample Subject 1" Thread from
DynamoDB forum if
    // "Sample Subject 2" Thread exists in DynamoDB forum
    Thread s3ForumThread = new Thread();
    s3ForumThread.forumName = "S3 Forum";
    s3ForumThread.subject = "Sample Subject 1";
    s3ForumThread.message = "Sample Question 1";

    Forum s3Forum = new Forum();
    s3Forum.name = "S3 Forum";
    s3Forum.category = "Amazon Web Services";
    s3Forum.threads = 1;

    Thread dynamodbForumThread1 = new Thread();
    dynamodbForumThread1.forumName = "DynamoDB Forum";
    dynamodbForumThread1.subject = "Sample Subject 1";

    Thread dynamodbForumThread2 = new Thread();
    dynamodbForumThread2.forumName = "DynamoDB Forum";
    dynamodbForumThread2.subject = "Sample Subject 2";
    DynamoDBTransactionWriteExpression conditionExpressionForConditionCheck = new
DynamoDBTransactionWriteExpression()

.withConditionExpression("attribute_exists(Subject)");

    Forum dynamodbForum = new Forum();
    dynamodbForum.name = "DynamoDB Forum";
    dynamodbForum.category = "Amazon Web Services";
    dynamodbForum.threads = 1;

    TransactionWriteRequest transactionWriteRequest = new TransactionWriteRequest();
    transactionWriteRequest.addPut(s3ForumThread);
    transactionWriteRequest.addUpdate(s3Forum);
    transactionWriteRequest.addDelete(dynamodbForumThread1);
    transactionWriteRequest.addConditionCheck(dynamodbForumThread2,
conditionExpressionForConditionCheck);
    transactionWriteRequest.addUpdate(dynamodbForum);
    executeTransactionWrite(transactionWriteRequest);
}

private static List<Object> executeTransactionLoad(TransactionLoadRequest
transactionLoadRequest) {
    List<Object> loadedObjects = new ArrayList<Object>();
    try {
        loadedObjects = mapper.transactionLoad(transactionLoadRequest);
    } catch (DynamoDBMappingException ddbme) {
        System.err.println("Client side error in Mapper, fix before retrying. Error: "
+ ddbme.getMessage());
    } catch (ResourceNotFoundException rnf) {
        System.err.println("One of the tables was not found, verify table exists before
retrying. Error: " + rnf.getMessage());
    } catch (InternalServerException ise) {
        System.err.println("Internal Server Error, generally safe to retry with back-
off. Error: " + ise.getMessage());
    }
}
```

```
        } catch (TransactionCanceledException tce) {
            System.err.println("Transaction Canceled, implies a client issue, fix before
retrying. Error: " + tce.getMessage());
        } catch (Exception ex) {
            System.err.println("An exception occurred, investigate and configure retry
strategy. Error: " + ex.getMessage());
        }
        return loadedObjects;
    }

    private static void executeTransactionWrite(TransactionWriteRequest
transactionWriteRequest) {
    try {
        mapper.transactionWrite(transactionWriteRequest);
    } catch (DynamoDBMappingException ddbme) {
        System.err.println("Client side error in Mapper, fix before retrying. Error: "
+ ddbme.getMessage());
    } catch (ResourceNotFoundException rnfe) {
        System.err.println("One of the tables was not found, verify table exists before
retrying. Error: " + rnfe.getMessage());
    } catch (InternalServerErrorException ise) {
        System.err.println("Internal Server Error, generally safe to retry with back-
off. Error: " + ise.getMessage());
    } catch (TransactionCanceledException tce) {
        System.err.println("Transaction Canceled, implies a client issue, fix before
retrying. Error: " + tce.getMessage());
    } catch (Exception ex) {
        System.err.println("An exception occurred, investigate and configure retry
strategy. Error: " + ex.getMessage());
    }
}

@DynamoDBTable(tableName = "Thread")
public static class Thread {
    private String forumName;
    private String subject;
    private String message;
    private String lastPostedDateTime;
    private String lastPostedBy;
    private Set<String> tags;
    private int answered;
    private int views;
    private int replies;

    // Partition key
    @DynamoDBHashKey(attributeName = "ForumName")
    public String getForumName() {
        return forumName;
    }

    public void setForumName(String forumName) {
        this.forumName = forumName;
    }

    // Sort key
    @DynamoDBRangeKey(attributeName = "Subject")
    public String getSubject() {
        return subject;
    }

    public void setSubject(String subject) {
        this.subject = subject;
    }

    @DynamoDBAttribute(attributeName = "Message")
    public String getMessage() {
        return message;
    }
}
```

```
}

public void setMessage(String message) {
    this.message = message;
}

@DynamoDBAttribute(attributeName = "LastPostedDateTime")
public String getLastPostedDateTime() {
    return lastPostedDateTime;
}

public void setLastPostedDateTime(String lastPostedDateTime) {
    this.lastPostedDateTime = lastPostedDateTime;
}

@DynamoDBAttribute(attributeName = "LastPostedBy")
public String getLastPostedBy() {
    return lastPostedBy;
}

public void setLastPostedBy(String lastPostedBy) {
    this.lastPostedBy = lastPostedBy;
}

@DynamoDBAttribute(attributeName = "Tags")
public Set<String> getTags() {
    return tags;
}

public void setTags(Set<String> tags) {
    this.tags = tags;
}

@DynamoDBAttribute(attributeName = "Answered")
public int getAnswered() {
    return answered;
}

public void setAnswered(int answered) {
    this.answered = answered;
}

@DynamoDBAttribute(attributeName = "Views")
public int getViews() {
    return views;
}

public void setViews(int views) {
    this.views = views;
}

@DynamoDBAttribute(attributeName = "Replies")
public int getReplies() {
    return replies;
}

public void setReplies(int replies) {
    this.replies = replies;
}

}

@DynamoDBTable(tableName = "Forum")
public static class Forum {
    private String name;
    private String category;
```

```
private int threads;

// Partition key
@DynamoDBHashKey(attributeName = "Name")
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

@DynamoDBAttribute(attributeName = "Category")
public String getCategory() {
    return category;
}

public void setCategory(String category) {
    this.category = category;
}

@DynamoDBAttribute(attributeName = "Threads")
public int getThreads() {
    return threads;
}

public void setThreads(int threads) {
    this.threads = threads;
}
}
```

Bloqueio otimista com número de versão

Bloqueio otimista É uma estratégia para garantir que o item no lado do cliente que você está atualizando (ou excluindo) seja o mesmo que o item no Amazon DynamoDB. Se você usar essa estratégia, suas gravações de banco de dados serão protegidas contra substituição pelas gravações de outros e vice-versa.

Com o bloqueio otimista, cada item tem um atributo que serve como um número de versão. Se você recuperar um item de uma tabela, o aplicativo registrará o número da versão desse item. Você poderá atualizar o item somente se o número de versão no lado do servidor não tiver sido alterado. Se há uma incompatibilidade de versão, isso significa que alguém modificou o item antes de você. A tentativa de atualização falha, porque você tem uma versão obsoleta do item. Se isso acontecer, basta tentar novamente ao recuperar o item e tentar atualizá-lo. O bloqueio otimista impede que você substitua acidentalmente as alterações que foram feitas por outros. Também impede que outros substituam acidentalmente suas alterações.

Para oferecer suporte ao bloqueio otimista, o AWS SDK for Java fornece a anotação `@DynamoDBVersionAttribute`. Na classe de mapeamento da sua tabela, você designa uma propriedade para armazenar o número da versão e a marca usando essa anotação. Ao salvar um objeto, o item correspondente na tabela do DynamoDB terá um atributo que armazena o número da versão. O `DynamoDBMapper` atribui um número de versão quando você salvar o objeto pela primeira vez e incrementa automaticamente o número da versão sempre que você atualiza o item. Suas solicitações de atualização ou exclusão só são bem-sucedidas se a versão do objeto no lado do cliente corresponder ao número de versão correspondente do item na tabela do DynamoDB.

`ConditionalCheckFailedException` será lançada se:

- Você usar bloqueio otimista com `@DynamoDBVersionAttribute` e o valor de versão no servidor for diferente do valor no lado do cliente.

- Você especificar suas próprias restrições condicionais ao salvar dados usando `DynamoDBMapper` com `DynamoDBSaveExpression` e ocorrer falha nessas restrições.

Note

- As tabelas globais do DynamoDB usam uma reconciliação “último gravador ganha” entre as atualizações simultâneas. Se você usa tabelas globais, a política de último gravador ganha. Portanto, neste caso, a estratégia de bloqueio não funciona como esperado.
- As operações de gravação transacional `DynamoDBMapper` não oferecem suporte a expressões de condição e anotação `@DynamoDBVersionAttribute` na mesma chamada de API. Se um objeto em uma gravação transacional for anotado com `@DynamoDBVersionAttribute` e também tiver uma expressão de condição, a `SdkClientException` será lançada.

Por exemplo, o código Java a seguir define uma classe `CatalogItem` que tem várias propriedades. A propriedade `Version` está marcada com a anotação `@DynamoDBVersionAttribute`.

Example

```
@DynamoDBTable(tableName="ProductCatalog")
public class CatalogItem {

    private Integer id;
    private String title;
    private String ISBN;
    private Set<String> bookAuthors;
    private String someProp;
    private Long version;

    @DynamoDBHashKey(attributeName="Id")
    public Integer getId() { return id; }
    public void setId(Integer Id) { this.id = Id; }

    @DynamoDBAttribute(attributeName="Title")
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    @DynamoDBAttribute(attributeName="ISBN")
    public String getISBN() { return ISBN; }
    public void setISBN(String ISBN) { this.ISBN = ISBN; }

    @DynamoDBAttribute(attributeName = "Authors")
    public Set<String> getBookAuthors() { return bookAuthors; }
    public void setBookAuthors(Set<String> bookAuthors) { this.bookAuthors = bookAuthors; }

    @DynamoDBIgnore
    public String getSomeProp() { return someProp; }
    public void setSomeProp(String someProp) { this.someProp = someProp; }

    @DynamoDBVersionAttribute
    public Long getVersion() { return version; }
    public void setVersion(Long version) { this.version = version; }
}
```

Você pode aplicar a anotação `@DynamoDBVersionAttribute` a tipos anuláveis fornecidos pelas classes wrapper primitivas que fornecem um tipo que permite valor nulo, como `Long` e `Integer`.

O bloqueio otimista tem o seguinte impacto sobre estes métodos `DynamoDBMapper`:

- `save`— Para um novo item, `DynamoDBMapper` atribui um número de versão inicial 1. Se você recuperar um item, atualizar uma ou mais das suas propriedades e tentar salvar as alterações, a operação de

salvamento será bem-sucedida somente se o número de versão no lado do cliente e no lado do servidor corresponderem. A classe `DynamoDBMapper` incrementa o número de versão automaticamente.

- `delete`— O `delete` método usa um objeto como parâmetro, e o método `DynamoDBMapper` realiza uma verificação de versão antes de excluir o item. A verificação da versão pode ser desabilitada se `DynamoDBMapperConfig.SaveBehavior.CLOBBER` for especificado na solicitação.

A implementação interna de bloqueio otimista dentro `DynamoDBMapper` usa o suporte para atualizações e exclusões condicionais fornecido pelo DynamoDB.

- `transactionWrite` —

- `Put`— Para um novo item, `DynamoDBMapper` atribui um número de versão inicial 1. Se você recuperar um item, atualizar uma ou mais das suas propriedades e tentar salvar as alterações, a operação `Put` será bem-sucedida somente se o número de versão no lado do cliente e no lado do servidor corresponder. A classe `DynamoDBMapper` incrementa o número de versão automaticamente.
- `Update`— Para um novo item, `DynamoDBMapper` atribui um número de versão inicial 1. Se você recuperar um item, atualizar uma ou mais das suas propriedades e tentar salvar as alterações, a operação `Update` será bem-sucedida somente se o número de versão no lado do cliente e no lado do servidor corresponder. A classe `DynamoDBMapper` incrementa o número de versão automaticamente.
- `Delete`— O `DynamoDBMapper` realiza uma verificação de versão antes de excluir o item. A operação `Delete` só será bem-sucedida se o número de versão no lado do cliente e no lado do servidor corresponder.
- `ConditionCheck`— O `@DynamoDBVersionAttribute` anotação não é suportada para `ConditionCheck` operações. Uma `SdkClientException` será lançada quando um item `ConditionCheck` for anotado com `@DynamoDBVersionAttribute`.

Como desabilitar o bloqueio otimista

Para desabilitar o bloqueio otimista, você pode alterar o valor de enumeração `DynamoDBMapperConfig.SaveBehavior` de `UPDATE` para `CLOBBER`. Você pode fazer isso criando uma instância de `DynamoDBMapperConfig` que ignora a verificação de versão e usando essa instância para todas as suas solicitações. Para obter informações sobre `DynamoDBMapperConfig.SaveBehavior` e outros parâmetros `DynamoDBMapper` opcionais, consulte [Definições de configuração opcionais para `DynamoDBMapper` \(p. 244\)](#).

Você também pode definir um comportamento de bloqueio apenas para uma operação específica. Por exemplo, o seguinte trecho de código Java usa `DynamoDBMapper` para salvar um item de catálogo. Ele especifica `DynamoDBMapperConfig.SaveBehavior` adicionando o parâmetro `DynamoDBMapperConfig` opcional ao método `save`.

Note

O método `transactionWrite` não oferece suporte à configuração `DynamoDBMapperConfig.SaveBehavior`. Não há suporte para a desabilitação do bloqueio otimista para `transactionWrite`.

Example

```
DynamoDBMapper mapper = new DynamoDBMapper(client);

// Load a catalog item.
CatalogItem item = mapper.load(CatalogItem.class, 101);
item.setTitle("This is a new title for the item");
...
// Save the item.
mapper.save(item,
    new DynamoDBMapperConfig(
        DynamoDBMapperConfig.SaveBehavior.CLOBBER));
```

Mapeamento de dados arbitrários

Além dos tipos Java suportados (consulte [Tipos de dados compatíveis \(p. 230\)](#)), você pode usar tipos no seu aplicativo para os quais não há um mapeamento direto para os tipos do Amazon DynamoDB. Para mapear esses tipos, você deve fornecer uma implementação que converta o tipo complexo em um tipo compatível com o DynamoDB e vice-versa, e anotar o método acessador de tipo complexo usando `@DynamoDBTypeConvertedAnotação`. O código de conversor transforma os dados quando os objetos são salvos ou carregados. Ele também é usado para todas as operações que consomem tipos complexos. Observe que, ao comparar dados durante operações de consulta e verificação, as comparações são feitas com os dados armazenados no DynamoDB.

Por exemplo, considere a seguinte classe `CatalogItem`, que define uma propriedade, `Dimension`, que é de `DimensionType`. Essa propriedade armazena as dimensões de itens, como altura, largura e espessura. Suponha que você decida armazenar essas dimensões de itens como uma string (como `8.5x11x.05`) no DynamoDB. O exemplo a seguir fornece o código de conversor que converte o objeto `DimensionType` em uma string e uma string em `DimensionType`.

Note

Este exemplo de código pressupõe que você já carregou dados no DynamoDB para sua conta seguindo as instruções na seção [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#) seção.

Para obter instruções detalhadas sobre como executar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
package com.amazonaws.codesamples.datamodeling;  
  
import java.io.IOException;  
import java.util.Arrays;  
import java.util.HashSet;  
import java.util.Set;  
  
import com.amazonaws.regions.Regions;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBAttribute;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTypeConverted;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTypeConverter;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;  
  
public class DynamoDBMapperExample {  
  
    static AmazonDynamoDB client;
```

```
public static void main(String[] args) throws IOException {
    // Set the AWS region you want to access.
    Regions usWest2 = Regions.US_WEST_2;
    client = AmazonDynamoDBClientBuilder.standard().withRegion(usWest2).build();

    DimensionType dimType = new DimensionType();
    dimType.setHeight("8.00");
    dimType.setLength("11.0");
    dimType.setThickness("1.0");

    Book book = new Book();
    book.setId(502);
    book.setTitle("Book 502");
    book.setISBN("555-5555555555");
    book.setBookAuthors(new HashSet<String>(Arrays.asList("Author1", "Author2")));
    book.setDimensions(dimType);

    DynamoDBMapper mapper = new DynamoDBMapper(client);
    mapper.save(book);

    Book bookRetrieved = mapper.load(Book.class, 502);
    System.out.println("Book info: " + "\n" + bookRetrieved);

    bookRetrieved.getDimensions().setHeight("9.0");
    bookRetrieved.getDimensions().setLength("12.0");
    bookRetrieved.getDimensions().setThickness("2.0");

    mapper.save(bookRetrieved);

    bookRetrieved = mapper.load(Book.class, 502);
    System.out.println("Updated book info: " + "\n" + bookRetrieved);
}

@DynamoDBTable(tableName = "ProductCatalog")
public static class Book {
    private int id;
    private String title;
    private String ISBN;
    private Set<String> bookAuthors;
    private DimensionType dimensionType;

    // Partition key
    @DynamoDBHashKey(attributeName = "Id")
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @DynamoDBAttribute(attributeName = "Title")
    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    @DynamoDBAttribute(attributeName = "ISBN")
    public String getISBN() {
        return ISBN;
    }
}
```

```
public void setISBN(String ISBN) {
    this.ISBN = ISBN;
}

@DynamoDBAttribute(attributeName = "Authors")
public Set<String> getBookAuthors() {
    return bookAuthors;
}

public void setBookAuthors(Set<String> bookAuthors) {
    this.bookAuthors = bookAuthors;
}

@DynamoDBTypeConverted(converter = DimensionTypeConverter.class)
@DynamoDBAttribute(attributeName = "Dimensions")
public DimensionType getDimensions() {
    return dimensionType;
}

@DynamoDBAttribute(attributeName = "Dimensions")
public void setDimensions(DimensionType dimensionType) {
    this.dimensionType = dimensionType;
}

@Override
public String toString() {
    return "Book [ISBN=" + ISBN + ", bookAuthors=" + bookAuthors + ",
dimensionType= "
        + dimensionType.getHeight() + " X " + dimensionType.getLength() + " X " +
dimensionType.getThickness()
        + ", Id=" + id + ", Title=" + title + "]";
}
}

static public class DimensionType {

    private String length;
    private String height;
    private String thickness;

    public String getLength() {
        return length;
    }

    public void setLength(String length) {
        this.length = length;
    }

    public String getHeight() {
        return height;
    }

    public void setHeight(String height) {
        this.height = height;
    }

    public String getThickness() {
        return thickness;
    }

    public void setThickness(String thickness) {
        this.thickness = thickness;
    }
}

// Converts the complex type DimensionType to a string and vice-versa.
```

```
static public class DimensionTypeConverter implements DynamoDBTypeConverter<String, DimensionType> {

    @Override
    public String convert(DimensionType object) {
        DimensionType itemDimensions = (DimensionType) object;
        String dimension = null;
        try {
            if (itemDimensions != null) {
                dimension = String.format("%s x %s x %s", itemDimensions.getLength(),
                itemDimensions.getHeight(),
                itemDimensions.getThickness());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return dimension;
    }

    @Override
    public DimensionType unconvert(String s) {

        DimensionType itemDimension = new DimensionType();
        try {
            if (s != null && s.length() != 0) {
                String[] data = s.split("x");
                itemDimension.setLength(data[0].trim());
                itemDimension.setHeight(data[1].trim());
                itemDimension.setThickness(data[2].trim());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return itemDimension;
    }
}
}
```

.NET: Modelo de documento

Tópicos

- [Operações sem suporte pelo modelo de documento \(p. 277\)](#)
- [Tipos de dados compatíveis \(p. 277\)](#)
- [Trabalho com itens no DynamoDB usando o AWS SDK for .NETModelo de documento \(p. 278\)](#)
- [Como obter um item - Table.GetItem \(p. 281\)](#)
- [Como excluir um item - Table.DeleteItem \(p. 282\)](#)
- [Como atualizar um item - Table.UpdateItem \(p. 283\)](#)
- [Gravação em lote: como inserir e excluir vários itens \(p. 285\)](#)
- [Exemplo: Operações CRUD usando o AWS SDK for .NETModelo de documento \(p. 286\)](#)
- [Exemplo: Operações em Batch usando o AWS SDK for .NETAPI de modelo de documento \(p. 289\)](#)
- [Consultando tabelas no DynamoDB usando o AWS SDK for .NETModelo de documento \(p. 291\)](#)

O AWS SDK for .NET fornece classes de modelos de documento que encapsulam algumas das operações do Amazon DynamoDB de baixo nível, simplificando ainda mais sua codificação. No modelo de documento, as classes primárias são `Table` e `Document`. A classe `Table` fornece métodos de operação

de dados, como `PutItem`, `GetItem` e `DeleteItem`. Ela também fornece os métodos `Query` e `Scan`. A classe `Document` representa um único item em uma tabela.

As classes do modelo de documento anteriores estão disponíveis no namespace `Amazon.DynamoDBv2.DocumentModel`.

Operações sem suporte pelo modelo de documento

Não é possível usar as classes do modelo de documento para criar, atualizar e excluir tabelas. Por outro lado, o modelo de documento oferece suporte à maioria das operações de dados comuns.

Tipos de dados compatíveis

O modelo de documento oferece suporte a um conjunto de tipos de dados .NET primitivos e tipos de dados de coleções. O modelo é compatível com os seguintes tipos de dados primitivos.

- `bool`
- `byte`
- `char`
- `DateTime`
- `decimal`
- `double`
- `float`
- `Guid`
- `Int16`
- `Int32`
- `Int64`
- `SByte`
- `string`
- `UInt16`
- `UInt32`
- `UInt64`

A tabela a seguir resume o mapeamento de tipos .NET anteriores para os tipos do DynamoDB.

Tipo primitivo .NET	Tipo DynamoDB
Todos os tipos de número	<code>N</code> (tipo Número)
Todos os tipos de string	<code>S</code> (tipo String)
<code>MemoryStream</code> , <code>byte[]</code>	<code>B</code> (tipo Binário)
<code>bool</code>	<code>N</code> (tipo Número). 0 representa false e 1 representa true.
<code>DateTime</code>	<code>S</code> (tipo String). Os valores do são armazenados como strings formatadas em ISO-8601.
<code>Guid</code>	<code>S</code> (tipo String).
Tipos de coleção (<code>List</code> , <code>HashSet</code> e <code>array</code>)	Tipo <code>BS</code> (conjunto binário), tipo <code>SS</code> (conjunto de strings) e tipo <code>NS</code> (conjunto de números)

AWS SDK for .NET define tipos para mapear os tipos booleano, nulo, lista e mapa do DynamoDB para a API do modelo de documento .NET:

- Use `DynamoDBBool` para o tipo booleano.
- Use `DynamoDBNull` para o tipo nulo.
- Use `DynamoDBList` para o tipo lista.
- Use `Document` para o tipo mapa.

Note

- Valores binários vazios são compatíveis.
- A leitura de valores string vazios é compatível. Os valores de atributos string vazios são compatíveis nos valores de atributos do tipo de conjunto string durante a gravação no DynamoDB. Os valores de atributos string vazios do tipo string e os valores string vazios contidos no tipo Lista ou Mapa são descartados das solicitações de gravação

Trabalho com itens no DynamoDB usando o AWS SDK for .NETModelo de documento

Tópicos

- [Como inserir um item - método Table.PutItem \(p. 279\)](#)
- [Especificação de parâmetros opcionais \(p. 280\)](#)

Para realizar operações de dados usando o modelo de documento, você deve primeiro chamar o método `Table.LoadTable`, que cria uma instância da classe `Table` que representa uma tabela específica. O exemplo de código C# a seguir cria um `Table` que representa a `ProductCatalog` tabela no Amazon DynamoDB.

Example

```
Table table = Table.LoadTable(client, "ProductCatalog");
```

Note

Em geral, você usa a propriedade `LoadTable` método uma vez no início do seu aplicativo, pois ele faz um `DescribeTable` que adiciona ao percurso de ida e volta para o DynamoDB.

É possível usar o objeto `Table` para realizar várias operações de dados. Cada operação de dados tem dois tipos de sobrecargas: Um que usa os parâmetros mínimos necessários, e o outro que usa informações de configuração opcionais específicas da operação. Por exemplo, para recuperar um item, é necessário fornecer o valor da chave primária da tabela. Nesse caso, é possível usar a seguinte sobrecarga de `GetItem`.

Example

```
// Get the item from a table that has a primary key that is composed of only a partition key.  
Table.GetItem(Primitive partitionKey);  
// Get the item from a table whose primary key is composed of both a partition key and sort key.  
Table.GetItem(Primitive partitionKey, Primitive sortKey);
```

Também é possível transmitir parâmetros opcionais para esses métodos. Por exemplo, a `GetItem` anterior retorna o item inteiro, incluindo todos os seus atributos. Como opção, é possível especificar uma lista de atributos a serem recuperados. Nesse caso, use a seguinte sobrecarga de `GetItem`, que usa o parâmetro de objeto de configuração específico da operação.

Example

```
// Configuration object that specifies optional parameters.  
GetItemOperationConfig config = new GetItemOperationConfig()  
{  
    AttributesToGet = new List<string>() { "Id", "Title" },  
};  
// Pass in the configuration to the GetItem method.  
// 1. Table that has only a partition key as primary key.  
Table.GetItem(Primitive partitionKey, GetItemOperationConfig config);  
// 2. Table that has both a partition key and a sort key.  
Table.GetItem(Primitive partitionKey, Primitive sortKey, GetItemOperationConfig config);
```

É possível usar o objeto de configuração para especificar vários parâmetros opcionais, como solicitar uma lista específica de atributos ou especificar o tamanho da página (número de itens por página). Cada método de operação de dados tem sua própria classe de configuração. Por exemplo é possível usar a classe `GetItemOperationConfig` para fornecer opções para a operação `GetItem`. É possível usar a classe `PutItemOperationConfig` a fim de fornecer parâmetros opcionais para a operação `PutItem`.

As seções a seguir discutem cada uma das operações de dados com suporte pela classe `Table`.

Como inserir um item - método `Table.PutItem`

O método `PutItem` faz upload da instância `Document` de entrada na tabela. Se um item com uma chave primária especificada na entrada `Document` existir na tabela, a operação `PutItem` substituirá esse item inteiro. O novo item será idêntico ao objeto `Document` que você forneceu para o método `PutItem`. Se o seu item original tinha atributos extras, eles não estão mais presentes no novo item.

Veja a seguir as etapas para inserir um novo item em uma tabela usando o modelo de documento do AWS SDK for .NET.

1. Execute a `Table.LoadTable` método que fornece o nome da tabela na qual você deseja inserir um item.
2. Crie um objeto `Document` que tenha uma lista de nomes de atributos e seus valores.
3. Execução do `Table.PutItem`, fornecendo o `Document` instância como um parâmetro.

O exemplo de código C# a seguir demonstra as tarefas anteriores. O exemplo faz upload de um item para a tabela `ProductCatalog`.

Example

```
Table table = Table.LoadTable(client, "ProductCatalog");  
  
var book = new Document();  
book["Id"] = 101;  
book["Title"] = "Book 101 Title";  
book["ISBN"] = "11-11-11-11";  
book["Authors"] = new List<string> { "Author 1", "Author 2" };  
book["InStock"] = new DynamoDBBool(true);  
book["QuantityOnHand"] = new DynamoDBNull();  
  
table.PutItem(book);
```

No exemplo anterior, a instância Document cria um item que tem atributos Number, String, String Set, Boolean, e Null. (Null é usado para indicar que o QuantityOnHand para essa operação é desconhecido.) Para Boolean e Null, use os métodos de construtor DynamoDBBool e DynamoDBNull.

No DynamoDB, o `ListeMap` tipos de dados podem conter elementos compostos por outros tipos de dados. Veja a seguir como mapear esses tipos de dados para a API do modelo de documento:

- List — use o construtor `DynamoDBList`.
- Map — use o construtor `Document`.

É possível modificar o exemplo anterior para adicionar um atributo List ao item. Para fazer isso, use um construtor `DynamoDBList`, conforme mostrado no exemplo de código a seguir.

Example

```
Table table = Table.LoadTable(client, "ProductCatalog");

var book = new Document();
book["Id"] = 101;

/*other attributes omitted for brevity...*/

var relatedItems = new DynamoDBList();
relatedItems.Add(341);
relatedItems.Add(472);
relatedItems.Add(649);
book.Add("RelatedItems", relatedItems);

table.PutItem(book);
```

Para adicionar um atributo Map ao livro, defina outro Document. O exemplo de código a seguir ilustra como fazer isso.

Example

```
Table table = Table.LoadTable(client, "ProductCatalog");

var book = new Document();
book["Id"] = 101;

/*other attributes omitted for brevity...*/

var pictures = new Document();
pictures.Add("FrontView", "http://example.com/products/101_front.jpg" );
pictures.Add("RearView", "http://example.com/products/101_rear.jpg" );

book.Add("Pictures", pictures);

table.PutItem(book);
```

Esses exemplos se baseiam no item mostrado em [Especificar atributos de item ao usar expressões \(p. 414\)](#). O modelo de documento permite criar atributos complexos aninhados, como o atributo `ProductReviews` mostrado no estudo de caso.

Especificação de parâmetros opcionais

É possível configurar parâmetros opcionais para a operação `PutItem`, adicionando o parâmetro `PutItemOperationConfig`. Para obter uma lista completa de parâmetros opcionais, consulte [PutItem](#).

O exemplo de código C# a seguir insere um item na tabela `ProductCatalog`. Ele especifica o parâmetro opcional a seguir:

- O parâmetro `ConditionalExpression` faz com que essa solicitação seja uma inserção condicional. O exemplo cria uma expressão que especifica que o atributo `ISBN` deve ter um valor específico, que precisa estar presente no item que você está substituindo.

Example

```
Table table = Table.LoadTable(client, "ProductCatalog");

var book = new Document();
book["Id"] = 555;
book["Title"] = "Book 555 Title";
book["Price"] = "25.00";
book["ISBN"] = "55-55-55-55";
book["Name"] = "Item 1 updated";
book["Authors"] = new List<string> { "Author x", "Author y" };
book["InStock"] = new DynamoDBBool(true);
book["QuantityOnHand"] = new DynamoDBNull();

// Create a condition expression for the optional conditional put operation.
Expression expr = new Expression();
expr.ExpressionStatement = "ISBN = :val";
expr.ExpressionAttributeValues[":val"] = "55-55-55-55";

PutItemOperationConfig config = new PutItemOperationConfig()
{
    // Optional parameter.
    ConditionalExpression = expr
};

table.PutItem(book, config);
```

Como obter um item - Table.GetItem

A operação `GetItem` recupera um item como uma instância de `Document`. É necessário fornecer a chave primária do item que você deseja recuperar, conforme mostrado no seguinte exemplo de código C#.

Example

```
Table table = Table.LoadTable(client, "ProductCatalog");
Document document = table.GetItem(101); // Primary key 101.
```

A operação `GetItem` retorna todos os atributos do item e realiza uma leitura eventualmente consistente (consulte [Consistência de leituras \(p. 17\)](#)) por padrão.

Especificação de parâmetros opcionais

Você pode configurar opções adicionais para a operação `GetItem`, adicionando o parâmetro `GetItemOperationConfig`. Para obter uma lista completa de parâmetros opcionais, consulte [GetItem](#). O exemplo de código do C# a seguir recupera um item da tabela `ProductCatalog`. Ele especifica a `GetItemOperationConfig` para fornecer os seguintes parâmetros opcionais:

- O parâmetro `AttributesToGet` para recuperar somente os atributos especificados.
- O parâmetro `ConsistentRead` para solicitar os valores mais recentes para todos os atributos especificados. Para saber mais sobre consistência de dados, consulte [Consistência de leituras \(p. 17\)](#).

Example

```
Table table = Table.LoadTable(client, "ProductCatalog");

GetItemOperationConfig config = new GetItemOperationConfig()
{
    AttributesToGet = new List<string>() { "Id", "Title", "Authors", "InStock",
    "QuantityOnHand" },
    ConsistentRead = true
};
Document doc = table.GetItem(101, config);
```

Ao recuperar um item usando a API do modelo de documento, é possível acessar os elementos individuais dentro do objeto Document que é retornado, conforme mostrado no exemplo a seguir.

Example

```
int id = doc["Id"].AsInt();
string title = doc["Title"].AsString();
List<string> authors = doc["Authors"].AsListOfString();
bool inStock = doc["InStock"].AsBoolean();
DynamoDBNull quantityOnHand = doc["QuantityOnHand"].AsDynamoDBNull();
```

Para atributos que são do tipo List ou Map, veja a seguir como mapear esses atributos para a API do modelo de documento:

- List— Uso do AsDynamoDBList Método do.
- Map— Uso do AsDocument Método do.

O exemplo de código a seguir mostra como recuperar um List (RelatedItems) e um Map (Pictures) do objeto Document:

Example

```
DynamoDBList relatedItems = doc["RelatedItems"].AsDynamoDBList();

Document pictures = doc["Pictures"].AsDocument();
```

Como excluir um item - Table.DeleteItem

A operação DeleteItem exclui um item de uma tabela. Você pode transmitir a chave primária do item como um parâmetro. Ou, se você já tiver lido um item e tiver o objeto Document correspondente, será possível transmiti-lo como um parâmetro para o método DeleteItem, conforme mostrado no exemplo de código C# a seguir.

Example

```
Table table = Table.LoadTable(client, "ProductCatalog");

// Retrieve a book (a Document instance)
Document document = table.GetItem(111);

// 1) Delete using the Document instance.
table.DeleteItem(document);

// 2) Delete using the primary key.
int partitionKey = 222;
table.DeleteItem(partitionKey)
```

Especificação de parâmetros opcionais

Você pode configurar opções adicionais para a operação `Delete`, adicionando o parâmetro `DeleteItemOperationConfig`. Para obter uma lista completa de parâmetros opcionais, consulte [DeleteTable](#). O exemplo de código C# a seguir especifica os dois parâmetros opcionais a seguir:

- O parâmetro `ConditionalExpression`, para garantir que o item de livro que está sendo excluído tenha um valor específico para o atributo ISBN.
- O parâmetro `ReturnValues`, para solicitar que o método `Delete` retorne o item excluído.

Example

```
Table table = Table.LoadTable(client, "ProductCatalog");
int partitionKey = 111;

Expression expr = new Expression();
expr.ExpressionStatement = "ISBN = :val";
expr.ExpressionAttributeValues[":val"] = "11-11-11-11";

// Specify optional parameters for Delete operation.
DeleteItemOperationConfig config = new DeleteItemOperationConfig
{
    ConditionalExpression = expr,
    ReturnValues = ReturnValues.AllOldAttributes // This is the only supported value when
        using the document model.
};

// Delete the book.
Document d = table.DeleteItem(partitionKey, config);
```

Como atualizar um item - Table.UpdateItem

A operação `UpdateItem` atualiza um item existente, se estiver presente. Se o item que tem a chave primária especificada não for encontrado, a operação `UpdateItem` adicionará um novo item.

É possível usar a operação `UpdateItem` para atualizar valores de atributos existentes, adicionar novos atributos à coleção existente ou excluir atributos da coleção existente. Forneça essas atualizações ao criar uma instância `Document` que descreve as atualizações que você deseja realizar.

A ação `UpdateItem` usa as seguintes diretrizes:

- Se o item não existir, o `UpdateItem` adicionará um novo item usando a chave primária especificada na entrada.
- Se o item existir, o `UpdateItem` aplicará as atualizações da seguinte maneira:
 - Substitui os valores de atributos existentes pelos os valores na atualização.
 - Se um atributo que você fornecer na entrada não existir, ele adicionará um novo atributo ao item.
 - Se o valor do atributo de entrada for nulo, ele excluirá os atributos, se estiver presente.

Note

Este nível médio `UpdateItem` operação não é compatível com o `Add`ação (consulte [UpdateItem](#)) com suporte pela operação subjacente do DynamoDB.

Note

A operação `PutItem` operation ([Como inserir um item - método Table.PutItem \(p. 279\)](#)) também pode realizar uma atualização. Se você chamar `PutItem` para fazer upload de um item, e a chave primária existir, a operação `PutItem` substituirá o item inteiro. Se houver atributos no item

existente e eles não forem especificados no `Document` que está sendo inserido, a operação `PutItem` excluirá esses atributos. No entanto, `UpdateItem` só atualiza os atributos de entrada especificados. Outros atributos existentes desse item permanecerão inalterados.

Veja a seguir as etapas para atualizar um item usando o modelo de documento do AWS SDK for .NET:

1. Execute `aTable.LoadTable` fornecendo o nome da tabela na qual você deseja executar a operação de atualização.
2. Crie uma instância de `Document`, fornecendo todas as atualizações que você deseja realizar.
Para excluir um atributo existente, especifique o valor desse atributo como nulo.
3. Chame o método `Table.UpdateItem` e forneça a instância `Document` como um parâmetro de entrada.

Você deve fornecer a chave primária na instância de `Document` ou explicitamente como um parâmetro.

O exemplo de código C# a seguir demonstra as tarefas anteriores. O exemplo de código atualiza um item na tabela `Book`. A operação `UpdateItem` atualiza o atributo `Authors` existente, exclui o atributo `PageCount` e adiciona um novo atributo `XYZ`. A instância `Document` inclui a chave primária do livro a ser atualizado.

Example

```
Table table = Table.LoadTable(client, "ProductCatalog");

var book = new Document();

// Set the attributes that you wish to update.
book["Id"] = 111; // Primary key.
// Replace the authors attribute.
book["Authors"] = new List<string> { "Author x", "Author y" };
// Add a new attribute.
book["XYZ"] = 12345;
// Delete the existing PageCount attribute.
book["PageCount"] = null;

table.Update(book);
```

Especificação de parâmetros opcionais

Você pode configurar opções adicionais para a operação `UpdateItem`, adicionando o parâmetro `UpdateItemOperationConfig`. Para obter uma lista completa de parâmetros opcionais, consulte [UpdateItem](#).

O exemplo de código C# a seguir atualiza um preço de item de livro para 25. Ele especifica os dois parâmetros opcionais a seguir:

- O parâmetro `ConditionalExpression`, que identifica o atributo `Price` com o valor 20 que você espera que esteja presente.
- O parâmetro `ReturnValues` para solicitar que a operação `UpdateItem` retorne o item que está sendo atualizado.

Example

```
Table table = Table.LoadTable(client, "ProductCatalog");
string partitionKey = "111";
```

```
var book = new Document();
book["Id"] = partitionKey;
book["Price"] = 25;

Expression expr = new Expression();
expr.ExpressionStatement = "Price = :val";
expr.ExpressionAttributeValues[":val"] = "20";

UpdateItemOperationConfig config = new UpdateItemOperationConfig()
{
    ConditionalExpression = expr,
    ReturnValue = ReturnValue.AllOldAttributes
};

Document d1 = table.Update(book, config);
```

Gravação em lote: como inserir e excluir vários itens

Gravação em lote se refere a inserir e excluir vários itens em um lote. A operação permite que você insira e exclua vários itens de uma ou mais tabelas em uma única chamada. Veja a seguir as etapas para inserir ou excluir vários itens de uma tabela usando o AWS SDK for .NET API do modelo de documento.

1. Crie um objeto `Table` executando o método `Table.LoadTable` ao fornecer o nome da tabela na qual você deseja executar a operação em lote.
2. Execute a `CreateBatchWrite` na instância de tabela que você criou na etapa anterior e crie um `DocumentBatchWrite`.
3. Use os métodos de objeto `DocumentBatchWrite` para especificar os documentos que você deseja carregar ou excluir.
4. Chame a `DocumentBatchWrite.Execute` para executar a operação em lote.

Ao usar a API do modelo de documento, você pode especificar qualquer número de operações em um lote. No entanto, o DynamoDB limita o número de operações em lote e o tamanho total do lote em uma operação em lote. Para obter mais informações sobre os limites específicos, consulte [BatchWriteItem](#). Se a API do modelo de documento detectar que sua solicitação de gravação em lote excedeu o número de solicitações de gravação permitidas ou que o tamanho da carga HTTP de um lote excedeu o limite permitido por `BatchWriteItem`, ela fragmentará esse lote em vários lotes menores. Além disso, se uma resposta a uma gravação em lote retornar itens não processados, a API do modelo de documento enviará automaticamente outra solicitação em lote com esses itens não processados.

O exemplo de código C# a seguir demonstra as etapas anteriores. O exemplo usa a operação de gravação em lote para realizar duas gravações: fazer upload de um item de livro e excluir outro item de livro.

```
Table productCatalog = Table.LoadTable(client, "ProductCatalog");
var batchWrite = productCatalog.CreateBatchWrite();

var book1 = new Document();
book1["Id"] = 902;
book1["Title"] = "My book1 in batch write using .NET document model";
book1["Price"] = 10;
book1["Authors"] = new List<string> { "Author 1", "Author 2", "Author 3" };
book1["InStock"] = new DynamoDBBool(true);
book1["QuantityOnHand"] = 5;

batchWrite.AddDocumentToPut(book1);
// specify delete item using overload that takes PK.
batchWrite.AddKeyToDelete(12345);

batchWrite.Execute();
```

Para obter um exemplo de trabalho, consulte [Exemplo: Operações em Batch usando oAWS SDK for .NETAPI de modelo de documento \(p. 289\)](#).

É possível usar a operação `batchWrite` para realizar operações de inserção e exclusão em várias tabelas. Veja a seguir as etapas para inserir ou excluir vários itens de várias tabelas usando o modelo de documento do AWS SDK for .NET.

1. Crie a instância de `DocumentBatchWrite` para cada tabela na qual deseja inserir ou excluir vários itens, conforme descrito no procedimento anterior.
2. Crie uma instância de `MultiTableDocumentBatchWrite` e adicione os objetos `DocumentBatchWrite` individuais nela.
3. Execute o método `MultiTableDocumentBatchWrite.Execute`.

O exemplo de código C# a seguir demonstra as etapas anteriores. O exemplo usa a operação de gravação em lote para realizar as seguintes operações de gravação:

- Inserir um novo item no item de tabela `Forum`.
- Inserir um item na tabela `Thread` e excluir um item da mesma tabela.

```
// 1. Specify item to add in the Forum table.  
Table forum = Table.LoadTable(client, "Forum");  
var forumBatchWrite = forum.CreateBatchWrite();  
  
var forum1 = new Document();  
forum1["Name"] = "Test BatchWrite Forum";  
forum1["Threads"] = 0;  
forumBatchWrite.AddDocumentToPut(forum1);  
  
// 2a. Specify item to add in the Thread table.  
Table thread = Table.LoadTable(client, "Thread");  
var threadBatchWrite = thread.CreateBatchWrite();  
  
var thread1 = new Document();  
thread1["ForumName"] = "Amazon S3 forum";  
thread1["Subject"] = "My sample question";  
thread1["Message"] = "Message text";  
thread1["KeywordTags"] = new List<string>{ "Amazon S3", "Bucket" };  
threadBatchWrite.AddDocumentToPut(thread1);  
  
// 2b. Specify item to delete from the Thread table.  
threadBatchWrite.AddKeyToDelete("someForumName", "someSubject");  
  
// 3. Create multi-table batch.  
var superBatch = new MultiTableDocumentBatchWrite();  
superBatch.AddBatch(forumBatchWrite);  
superBatch.AddBatch(threadBatchWrite);  
  
superBatch.Execute();
```

Exemplo: Operações CRUD usando oAWS SDK for .NETModelo de documento

O exemplo de código C# a seguir realiza as seguintes ações:

- Cria um item de livro na tabela `ProductCatalog`.

- Recupera o item de livro.
- Atualiza o item de livro. O exemplo de código mostra uma atualização normal que adiciona novos atributos e atualiza atributos existentes. Ele também mostra uma atualização condicional que atualiza o preço do livro somente quando o valor de preço existente é igual ao especificado no código.
- Exclui o item de livro.

Para ver as instruções passo a passo para testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.DocumentModel;  
using Amazon.Runtime;  
  
namespace com.amazonaws.codesamples  
{  
    class MidlevelItemCRUD  
    {  
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();  
        private static string tableName = "ProductCatalog";  
        // The sample uses the following id PK value to add book item.  
        private static int sampleBookId = 555;  
  
        static void Main(string[] args)  
        {  
            try  
            {  
                Table productCatalog = Table.LoadTable(client, tableName);  
                CreateBookItem(productCatalog);  
                RetrieveBook(productCatalog);  
                // Couple of sample updates.  
                UpdateMultipleAttributes(productCatalog);  
                UpdateBookPriceConditionally(productCatalog);  
  
                // Delete.  
                DeleteBook(productCatalog);  
                Console.WriteLine("To continue, press Enter");  
                Console.ReadLine();  
            }  
            catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message); }  
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }  
            catch (Exception e) { Console.WriteLine(e.Message); }  
        }  
  
        // Creates a sample book item.
```

```
private static void CreateBookItem(Table productCatalog)
{
    Console.WriteLine("\n*** Executing CreateBookItem() ***");
    var book = new Document();
    book["Id"] = sampleBookId;
    book["Title"] = "Book " + sampleBookId;
    book["Price"] = 19.99;
    book["ISBN"] = "111-1111111111";
    book["Authors"] = new List<string> { "Author 1", "Author 2", "Author 3" };
    book["PageCount"] = 500;
    book["Dimensions"] = "8.5x11x.5";
    book["InPublication"] = new DynamoDBBool(true);
    book["InStock"] = new DynamoDBBool(false);
    book["QuantityOnHand"] = 0;

    productCatalog.PutItem(book);
}

private static void RetrieveBook(Table productCatalog)
{
    Console.WriteLine("\n*** Executing RetrieveBook() ***");
    // Optional configuration.
    GetItemOperationConfig config = new GetItemOperationConfig
    {
        AttributesToGet = new List<string> { "Id", "ISBN", "Title", "Authors",
"Price" },
        ConsistentRead = true
    };
    Document document = productCatalog.GetItem(sampleBookId, config);
    Console.WriteLine("RetrieveBook: Printing book retrieved...");
    PrintDocument(document);
}

private static void UpdateMultipleAttributes(Table productCatalog)
{
    Console.WriteLine("\n*** Executing UpdateMultipleAttributes() ***");
    Console.WriteLine("\nUpdating multiple attributes....");
    int partitionKey = sampleBookId;

    var book = new Document();
    book["Id"] = partitionKey;
    // List of attribute updates.
    // The following replaces the existing authors list.
    book["Authors"] = new List<string> { "Author x", "Author y" };
    book["newAttribute"] = "New Value";
    book["ISBN"] = null; // Remove it.

    // Optional parameters.
    UpdateItemOperationConfig config = new UpdateItemOperationConfig
    {
        // Get updated item in response.
        ReturnValues = ReturnValues.AllNewAttributes
    };
    Document updatedBook = productCatalog.UpdateItem(book, config);
    Console.WriteLine("UpdateMultipleAttributes: Printing item after updates ...");
    PrintDocument(updatedBook);
}

private static void UpdateBookPriceConditionally(Table productCatalog)
{
    Console.WriteLine("\n*** Executing UpdateBookPriceConditionally() ***");

    int partitionKey = sampleBookId;

    var book = new Document();
    book["Id"] = partitionKey;
```

```
book["Price"] = 29.99;

// For conditional price update, creating a condition expression.
Expression expr = new Expression();
expr.ExpressionStatement = "Price = :val";
expr.ExpressionAttributeValues[":val"] = 19.00;

// Optional parameters.
UpdateItemOperationConfig config = new UpdateItemOperationConfig
{
    ConditionalExpression = expr,
    ReturnValues = ReturnValues.AllNewAttributes
};
Document updatedBook = productCatalog.UpdateItem(book, config);
Console.WriteLine("UpdateBookPriceConditionally: Printing item whose price was
conditionally updated");
PrintDocument(updatedBook);
}

private static void DeleteBook(Table productCatalog)
{
    Console.WriteLine("\n*** Executing DeleteBook() ***");
    // Optional configuration.
    DeleteItemOperationConfig config = new DeleteItemOperationConfig
    {
        // Return the deleted item.
        ReturnValues = ReturnValues.AllOldAttributes
    };
    Document document = productCatalog.DeleteItem(sampleBookId, config);
    Console.WriteLine("DeleteBook: Printing deleted just deleted...");
    PrintDocument(document);
}

private static void PrintDocument(Document updatedDocument)
{
    foreach (var attribute in updatedDocument.GetAttributeNames())
    {
        string stringValue = null;
        var value = updatedDocument[attribute];
        if (value is Primitive)
            stringValue = value.AsPrimitive().Value.ToString();
        else if (value is PrimitiveList)
            stringValue = string.Join(", ", (from primitive
                in value.AsPrimitiveList().Entries
                select primitive.Value).ToArray());
        Console.WriteLine("{0} - {1}", attribute, stringValue);
    }
}
}
```

Exemplo: Operações em Batch usando oAWS SDK for .NETAPI de modelo de documento

Tópicos

- Exemplo: Gravação em Batch usando oAWS SDK for .NETModelo de documento (p. 289)

Exemplo: Gravação em Batch usando oAWS SDK for .NETModelo de documento

O exemplo de código C# a seguir ilustra operações de gravação em lote em uma ou várias tabelas. O exemplo realiza as seguintes tarefas:

- Ilustra uma gravação em lote em uma única tabela. Adiciona dois itens à tabela `ProductCatalog`.
- Ilustra uma gravação em lote em várias tabelas. Adiciona um item às tabelas `Forum` e `Thread` e exclui um item da tabela `Thread`.

Se você seguiu as etapas em [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#), já tem as tabelas `ProductCatalog`, `Forum` e `Thread` criadas. Você também pode criar essas tabelas de amostra de forma programática. Para mais informações, consulte [Como criar tabelas de exemplo e carregar dados usando o AWS SDK for .NET \(p. 1086\)](#). Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
using System;  
using System.Collections.Generic;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.DocumentModel;  
using Amazon.Runtime;  
  
namespace com.amazonaws.codesamples  
{  
    class MidLevelBatchWriteItem  
    {  
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();  
        static void Main(string[] args)  
        {  
            try  
            {  
                SingleTableBatchWrite();  
                MultiTableBatchWrite();  
            }  
            catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message); }  
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }  
            catch (Exception e) { Console.WriteLine(e.Message); }  
  
            Console.WriteLine("To continue, press Enter");  
            Console.ReadLine();  
        }  
  
        private static void SingleTableBatchWrite()  
        {  
            Table productCatalog = Table.LoadTable(client, "ProductCatalog");  
            var batchWrite = productCatalog.CreateBatchWrite();  
  
            var book1 = new Document();  
            book1["Id"] = 902;  
            book1["Title"] = "My book1 in batch write using .NET helper classes";  
            book1["ISBN"] = "902-11-1111-1";  
            book1["Price"] = 10;  
            book1["ProductCategory"] = "Book";
```

```
book1["Authors"] = new List<string> { "Author 1", "Author 2", "Author 3" };
book1["Dimensions"] = "8.5x11x.5";
book1["InStock"] = new DynamoDBBool(true);
book1["QuantityOnHand"] = new DynamoDBNull(); //Quantity is unknown at this
time

batchWrite.AddDocumentToPut(book1);
// Specify delete item using overload that takes PK.
batchWrite.AddKeyToDelete(12345);
Console.WriteLine("Performing batch write in SingleTableBatchWrite()");
batchWrite.Execute();
}

private static void MultiTableBatchWrite()
{
    // 1. Specify item to add in the Forum table.
    Table forum = Table.LoadTable(client, "Forum");
    var forumBatchWrite = forum.CreateBatchWrite();

    var forum1 = new Document();
    forum1["Name"] = "Test BatchWrite Forum";
    forum1["Threads"] = 0;
    forumBatchWrite.AddDocumentToPut(forum1);

    // 2a. Specify item to add in the Thread table.
    Table thread = Table.LoadTable(client, "Thread");
    var threadBatchWrite = thread.CreateBatchWrite();

    var thread1 = new Document();
    thread1["ForumName"] = "S3 forum";
    thread1["Subject"] = "My sample question";
    thread1["Message"] = "Message text";
    thread1["KeywordTags"] = new List<string> { "S3", "Bucket" };
    threadBatchWrite.AddDocumentToPut(thread1);

    // 2b. Specify item to delete from the Thread table.
    threadBatchWrite.AddKeyToDelete("someForumName", "someSubject");

    // 3. Create multi-table batch.
    var superBatch = new MultiTableDocumentBatchWrite();
    superBatch.AddBatch(forumBatchWrite);
    superBatch.AddBatch(threadBatchWrite);
    Console.WriteLine("Performing batch write in MultiTableBatchWrite()");
    superBatch.Execute();
}
}
```

Consultando tabelas no DynamoDB usando o AWS SDK for .NETModelo de documento

Tópicos

- [Método Table.Query na classeAWS SDK for .NET \(p. 291\)](#)
- [Método Table.ScanAWS SDK for .NET \(p. 296\)](#)

Método Table.Query na classeAWS SDK for .NET

O método `Query` permite que você consulte suas tabelas. Apenas é possível consultar tabelas que tenham uma chave primária composta (chave de partição e chave de classificação). Se a chave primária da sua tabela for composta por apenas uma chave de partição, a operação `Query` não terá suporte. Por padrão,

`Query` realiza internamente consultas que são eventualmente consistentes. Para saber mais sobre o modelo de consistência, consulte [Consistência de leituras \(p. 17\)](#).

O método `Query` fornece duas sobrecargas. Os parâmetros mínimos necessários para o método `Query` são um valor de chave de partição e um filtro de chave de classificação. É possível usar a seguinte sobrecarga para fornecer esses parâmetros mínimos necessários.

Example

```
Query(Primitive partitionKey, RangeFilter Filter);
```

Por exemplo, o código C# a seguir consulta todas as respostas de fórum que foram publicadas nos últimos 15 dias.

Example

```
string tableName = "Reply";
Table table = Table.LoadTable(client, tableName);

DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
RangeFilter filter = new RangeFilter(QueryOperator.GreaterThan, twoWeeksAgoDate);
Search search = table.Query("DynamoDB Thread 2", filter);
```

Isso cria um objeto `Search`. Agora, é possível chamar o método `Search.GetNextSet` iterativamente para recuperar uma página de resultados por vez, conforme mostrado no exemplo de código C# a seguir. O código imprime os valores de atributo para cada item que a consulta retorna.

Example

```
List<Document> documentSet = new List<Document>();
do
{
    documentSet = search.GetNextSet();
    foreach (var document in documentSet)
        PrintDocument(document);
} while (!search.IsDone());

private static void PrintDocument(Document document)
{
    Console.WriteLine();
    foreach (var attribute in document.GetAttributeNames())
    {
        string stringValue = null;
        var value = document[attribute];
        if (value is Primitive)
            stringValue = value.AsPrimitive().Value;
        else if (value is PrimitiveList)
            stringValue = string.Join(", ", (from primitive
                                              in value.AsPrimitiveList().Entries
                                              select primitive.Value).ToArray());
        Console.WriteLine("{0} - {1}", attribute, stringValue);
    }
}
```

Especificação de parâmetros opcionais

Também é possível especificar os parâmetros opcionais de `Query`, como especificar uma lista de atributos a serem recuperados, leituras fortemente consistentes, tamanho das páginas e o número

de itens retornados por página. Para obter uma lista completa de parâmetros, consulte[Consulte](#). Para especificar parâmetros opcionais, você deve usar a seguinte sobrecarga, na qual você fornece o objeto `QueryOperationConfig`.

Example

```
Query(QueryOperationConfig config);
```

Suponha que você queira executar a consulta no exemplo anterior (recuperar respostas de fórum postadas nos últimos 15 dias). No entanto, suponha que você queira fornecer parâmetros de consulta opcionais para recuperar apenas atributos específicos e também solicitar uma leitura fortemente consistente. O exemplo de código C# a seguir cria a solicitação usando o objeto `QueryOperationConfig`.

Example

```
Table table = Table.LoadTable(client, "Reply");
DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
QueryOperationConfig config = new QueryOperationConfig()
{
    HashKey = "DynamoDB Thread 2", //Partition key
    AttributesToGet = new List<string>
    {
        "Subject", "ReplyDateTime", "PostedBy"
    },
    ConsistentRead = true,
    Filter = new RangeFilter(QueryOperator.GreaterThan, twoWeeksAgoDate)
};

Search search = table.Query(config);
```

Exemplo: Consultar usando o método `Table.Query`

O exemplo de código C# a seguir usa o `Table.Query` para executar as seguintes consultas de exemplo.

- As consultas a seguir são executadas no `ReplyTabela` do.
- Localizar respostas de tópicos de fórum que foram postadas nos últimos 15 dias.

Essa consulta é executada duas vezes. Na primeira chamada de `Table.Query`, o exemplo fornece somente os parâmetros de consulta necessários. Na segunda chamada de `Table.Query`, forneça parâmetros de consulta opcionais para solicitar uma leitura fortemente consistente e uma lista de atributos para recuperar.

- Localizar respostas de tópicos de fórum postadas durante um certo período.

Essa consulta usa o operador de consulta `Between` para localizar respostas publicadas entre duas datas.

- Obtenha um produto da tabela `ProductCatalog`.

Como a tabela `ProductCatalog` tem uma chave primária que é somente uma chave de partição, só é possível obter itens, mas não consultar a tabela. O exemplo recupera um item de produto específico usando o `Id` do item.

Example

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 */
```

```
* This file is licensed under the Apache License, Version 2.0 (the "License").  
* You may not use this file except in compliance with the License. A copy of  
* the License is located at  
*  
* http://aws.amazon.com/apache2.0/  
*  
* This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
* CONDITIONS OF ANY KIND, either express or implied. See the License for the  
* specific language governing permissions and limitations under the License.  
*/  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.DocumentModel;  
using Amazon.Runtime;  
using Amazon.SecurityToken;  
  
namespace com.amazonaws.codesamples  
{  
    class MidLevelQueryAndScan  
    {  
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();  
  
        static void Main(string[] args)  
        {  
            try  
            {  
                // Query examples.  
                Table replyTable = Table.LoadTable(client, "Reply");  
                string forumName = "Amazon DynamoDB";  
                string threadSubject = "DynamoDB Thread 2";  
                FindRepliesInLast15Days(replyTable, forumName, threadSubject);  
                FindRepliesInLast15DaysWithConfig(replyTable, forumName, threadSubject);  
                FindRepliesPostedWithinTimePeriod(replyTable, forumName, threadSubject);  
  
                // Get Example.  
                Table productCatalogTable = Table.LoadTable(client, "ProductCatalog");  
                int productId = 101;  
                GetProduct(productCatalogTable, productId);  
  
                Console.WriteLine("To continue, press Enter");  
                Console.ReadLine();  
            }  
            catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message); }  
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }  
            catch (Exception e) { Console.WriteLine(e.Message); }  
        }  
  
        private static void GetProduct(Table tableName, int productId)  
        {  
            Console.WriteLine("**** Executing GetProduct() ****");  
            Document productDocument = tableName.GetItem(productId);  
            if (productDocument != null)  
            {  
                PrintDocument(productDocument);  
            }  
            else  
            {  
                Console.WriteLine("Error: product " + productId + " does not exist");  
            }  
        }  
  
        private static void FindRepliesInLast15Days(Table table, string forumName, string  
threadSubject)  
        {
```

```
string Attribute = forumName + "#" + threadSubject;

DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
QueryFilter filter = new QueryFilter("Id", QueryOperator.Equal, partitionKey);
filter.AddCondition("ReplyDateTime", QueryOperator.GreaterThan,
twoWeeksAgoDate);

// Use Query overloads that takes the minimum required query parameters.
Search search = table.Query(filter);

List<Document> documentSet = new List<Document>();
do
{
    documentSet = search.GetNextSet();
    Console.WriteLine("\nFindRepliesInLast15Days: printing .....");
    foreach (var document in documentSet)
        PrintDocument(document);
} while (!search.IsDone);

private static void FindRepliesPostedWithinTimePeriod(Table table, string
forumName, string threadSubject)
{
    DateTime startDate = DateTime.UtcNow.Subtract(new TimeSpan(21, 0, 0, 0));
    DateTime endDate = DateTime.UtcNow.Subtract(new TimeSpan(1, 0, 0, 0));

    QueryFilter filter = new QueryFilter("Id", QueryOperator.Equal, forumName + "#"
+ threadSubject);
    filter.AddCondition("ReplyDateTime", QueryOperator.Between, startDate,
endDate);

    QueryOperationConfig config = new QueryOperationConfig()
    {
        Limit = 2, // 2 items/page.
        Select = SelectValues.SpecificAttributes,
        AttributesToGet = new List<string> { "Message",
                                             "ReplyDateTime",
                                             "PostedBy" },
        ConsistentRead = true,
        Filter = filter
    };

    Search search = table.Query(config);

    List<Document> documentList = new List<Document>();

    do
    {
        documentList = search.GetNextSet();
        Console.WriteLine("\nFindRepliesPostedWithinTimePeriod: printing replies
posted within dates: {0} and {1} .....", startDate, endDate);
        foreach (var document in documentList)
        {
            PrintDocument(document);
        }
    } while (!search.IsDone);
}

private static void FindRepliesInLast15DaysWithConfig(Table table, string
forumName, string threadName)
{
    DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
    QueryFilter filter = new QueryFilter("Id", QueryOperator.Equal, forumName + "#"
+ threadName);
    filter.AddCondition("ReplyDateTime", QueryOperator.GreaterThan,
twoWeeksAgoDate);
```

```
// You are specifying optional parameters so use QueryOperationConfig.  
QueryOperationConfig config = new QueryOperationConfig()  
{  
    Filter = filter,  
    // Optional parameters.  
    Select = SelectValues.SpecificAttributes,  
    AttributesToGet = new List<string> { "Message", "ReplyDateTime",  
        "PostedBy" },  
    ConsistentRead = true  
};  
  
Search search = table.Query(config);  
  
List<Document> documentSet = new List<Document>();  
do  
{  
    documentSet = search.GetNextSet();  
    Console.WriteLine("\nFindRepliesInLast15DaysWithConfig:  
printing .....");  
    foreach (var document in documentSet)  
        PrintDocument(document);  
} while (!search.IsDone());  
}  
  
private static void PrintDocument(Document document)  
{  
    // count++;  
    Console.WriteLine();  
    foreach (var attribute in document.GetAttributeNames())  
    {  
        string stringValue = null;  
        var value = document[attribute];  
        if (value is Primitive)  
            stringValue = value.AsPrimitive().Value.ToString();  
        else if (value is PrimitiveList)  
            stringValue = string.Join(", ", (from primitive  
                in value.AsPrimitiveList().Entries  
                select primitive.Value).ToArray());  
        Console.WriteLine("{0} - {1}", attribute, stringValue);  
    }  
}
```

Método Table.ScanAWS SDK for .NET

O método Scan realiza uma verificação de tabela completa. Ele fornece duas sobrecargas. O único parâmetro necessário pelo método Scan é o filtro de verificação, que é possível fornecer usando a seguinte sobrecarga.

Example

```
Scan(ScanFilter filter);
```

Por exemplo, vamos supor que você mantenha uma tabela de informações de acompanhamento de tópicos de fórum, como o assunto do tópico (primário), a mensagem relacionada, o Id do fórum ao qual o tópico pertence, Tags e outras informações. Suponha que o assunto seja a chave primária.

Example

```
Thread(Subject, Message, ForumId, Tags, LastPostedDateTime, .... )
```

Esta é uma versão simplificada de fóruns e tópicos que você visualiza no AWSfóruns (consulte [Fóruns de discussão da](#)). O exemplo de código C# a seguir consulta todos os tópicos em um fórum específico (ForumId = 101) marcado com "sortkey". Como ForumId não é uma chave primária, o exemplo verifica a tabela. O ScanFilter inclui duas condições. A consulta retorna todos os tópicos que atendem ambas as condições.

Example

```
string tableName = "Thread";
Table ThreadTable = Table.LoadTable(client, tableName);

ScanFilter scanFilter = new ScanFilter();
scanFilter.AddCondition("ForumId", ScanOperator.Equal, 101);
scanFilter.AddCondition("Tags", ScanOperator.Contains, "sortkey");

Search search = ThreadTable.Scan(scanFilter);
```

Especificação de parâmetros opcionais

Também é possível especificar parâmetros opcionais para Scan, como uma lista específica de atributos para recuperar ou se deseja realizar uma leitura fortemente consistente. Para especificar parâmetros opcionais, você deve criar um objeto ScanOperationConfig, que inclua ambos os parâmetros necessários e opcionais, e usar a seguinte a sobrecarga.

Example

```
Scan(ScanOperationConfig config);
```

O exemplo de código C# a seguir executa a mesma consulta anterior (localizar tópicos do fórum no qual o ForumId é 101 e a Tag atributo contém a palavra-chave "sortkey"). Vamos supor que você queira adicionar um parâmetro opcional para recuperar somente uma lista de atributos específica. Nesse caso, é necessário criar um objeto ScanOperationConfig, fornecendo todos os parâmetros, necessários e opcionais, conforme mostrado no exemplo de código a seguir.

Example

```
string tableName = "Thread";
Table ThreadTable = Table.LoadTable(client, tableName);

ScanFilter scanFilter = new ScanFilter();
scanFilter.AddCondition("ForumId", ScanOperator.Equal, forumId);
scanFilter.AddCondition("Tags", ScanOperator.Contains, "sortkey");

ScanOperationConfig config = new ScanOperationConfig()
{
    AttributesToGet = new List<string> { "Subject", "Message" } ,
    Filter = scanFilter
};

Search search = ThreadTable.Scan(config);
```

Exemplo: Analisar usando o método Table.Scan

A operação Scan realiza uma verificação completa da tabela, o que a torna uma operação potencialmente cara. Em vez disso, você deve usar consultas. No entanto, em algumas ocasiões, talvez você precise executar uma verificação em uma tabela. Por exemplo, é possível ter um erro de entrada de dados nos preços de produtos, e é necessário examinar a tabela, conforme mostrado no exemplo de código C# a seguir. O exemplo faz a verificação da tabela ProductCatalog para localizar produtos cujo preço é menor que 0. O exemplo ilustra o uso das duas sobrecargas de Table.Scan.

- `Table.Scan`, que usa o objeto `ScanFilter` como um parâmetro.

Você pode transmitir o parâmetro `ScanFilter` ao transmitir apenas os parâmetros necessários.

- `Table.Scan`, que usa o objeto `ScanOperationConfig` como um parâmetro.

Você deverá usar o parâmetro `ScanOperationConfig` se quiser transmitir parâmetros opcionais para o método `Scan`.

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.DocumentModel;  
  
namespace com.amazonaws.codesamples  
{  
    class MidLevelScanOnly  
    {  
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();  
  
        static void Main(string[] args)  
        {  
            Table productCatalogTable = Table.LoadTable(client, "ProductCatalog");  
            // Scan example.  
            FindProductsWithNegativePrice(productCatalogTable);  
            FindProductsWithNegativePriceWithConfig(productCatalogTable);  
  
            Console.WriteLine("To continue, press Enter");  
            Console.ReadLine();  
        }  
  
        private static void FindProductsWithNegativePrice(Table productCatalogTable)  
        {  
            // Assume there is a price error. So we scan to find items priced < 0.  
            ScanFilter scanFilter = new ScanFilter();  
            scanFilter.AddCondition("Price", ScanOperator.LessThan, 0);  
  
            Search search = productCatalogTable.Scan(scanFilter);  
  
            List<Document> documentList = new List<Document>();  
            do  
            {  
                documentList = search.GetNextSet();  
                Console.WriteLine("\nFindProductsWithNegativePrice:  
printing .....");  
                foreach (var document in documentList)  
                    PrintDocument(document);  
            } while (documentList.Count > 0);  
        }  
    }  
}
```

```
        } while (!search.IsDone);

    }

    private static void FindProductsWithNegativePriceWithConfig(Table
productCatalogTable)
{
    // Assume there is a price error. So we scan to find items priced < 0.
    ScanFilter scanFilter = new ScanFilter();
    scanFilter.AddCondition("Price", ScanOperator.LessThan, 0);

    ScanOperationConfig config = new ScanOperationConfig()
    {
        Filter = scanFilter,
        Select = SelectValues.SpecificAttributes,
        AttributesToGet = new List<string> { "Title", "Id" }
    };

    Search search = productCatalogTable.Scan(config);

    List<Document> documentList = new List<Document>();
    do
    {
        documentList = search.GetNextSet();
        Console.WriteLine("\nFindProductsWithNegativePriceWithConfig:
printing .....");
        foreach (var document in documentList)
            PrintDocument(document);
    } while (!search.IsDone);
}

private static void PrintDocument(Document document)
{
    // count++;
    Console.WriteLine();
    foreach (var attribute in document.GetAttributeNames())
    {
        string stringValue = null;
        var value = document[attribute];
        if (value is Primitive)
            stringValue = value.AsPrimitive().Value.ToString();
        else if (value is PrimitiveList)
            stringValue = string.Join(", ", (from primitive
                in value.AsPrimitiveList().Entries
                select primitive.Value).ToArray());
        Console.WriteLine("{0} - {1}", attribute, stringValue);
    }
}
}
```

.NET: Modelo de persistência de objeto

Tópicos

- [Attribute DynamoDB \(p. 301\)](#)
- [Classe DynamoDBContext \(p. 303\)](#)
- [Tipos de dados compatíveis \(p. 308\)](#)
- [Bloqueio otimista usando um número de versão com o DynamoDB usando o AWS SDK for .NETModelo de persistência de objeto \(p. 309\)](#)
- [Mapeamento de dados arbitrários com o DynamoDB usando o AWS SDK for .NETModelo de persistência de objeto \(p. 311\)](#)
- [Operações em Batch usando o AWS SDK for .NETModelo de persistência de objeto \(p. 314\)](#)

- Exemplo: Operações CRUD usando oAWS SDK for .NETModelo de persistência de objeto (p. 317)
- Exemplo: Operação de gravação em Batch usando oAWS SDK for .NETModelo de persistência de objeto (p. 319)
- Exemplo: Consulta e digitalização no DynamoDB usando oAWS SDK for .NETModelo de persistência de objeto (p. 323)

OAWS SDK for .NETO fornece um modelo de persistência de objeto que permite que você mapeie suas classes do lado do cliente para tabelas do Amazon DynamoDB. Em seguida, cada instância de objeto é mapeada para um item nas tabelas correspondentes. Para salvar objetos no lado do cliente nas tabelas, o modelo de persistência de objeto fornece oDynamoDBContext, um ponto de entrada para o DynamoDB. Esta classe fornece uma conexão ao DynamoDB e permite que você acesse tabelas, realize várias operações CRUD e realize consultas.

O modelo de persistência de objeto fornece um conjunto de atributos para mapear classes no lado do cliente para tabelas e propriedades/campos para atributos de tabela.

Note

O modelo de persistência de objeto não fornece uma API para criar, atualizar ou excluir tabelas. Ele fornece apenas operações de dados. Você pode usar oAWS SDK for .NETAPI de baixo nível para criar, atualizar e excluir tabelas. Para mais informações, consulte [Como trabalhar com tabelas do DynamoDB no .NET](#) (p. 397).

O exemplo a seguir mostra como o modelo de persistência de objeto funciona. Começa com oProductCatalogTabela do. Ele temIdcomo chave primária.

```
ProductCatalog(Id, ...)
```

Suponha que você tenha umBookclasse comTitle,ISBN, eAuthors.properties Você pode mapear oBookclasse para aProductCatalogAdicionando os atributos definidos pelo modelo de persistência de objeto, conforme mostrado no seguinte exemplo de código C#.

Example

```
[DynamoDBTable("ProductCatalog")]
public class Book
{
    [DynamoDBHashKey]
    public int Id { get; set; }

    public string Title { get; set; }
    public int ISBN { get; set; }

    [DynamoDBProperty("Authors")]
    public List<string> BookAuthors { get; set; }

    [DynamoDBIgnore]
    public string CoverPage { get; set; }
}
```

No exemplo anterior, oDynamoDBTablemapeia o atributoBookclasse para aProductCatalogTabela do.

O modelo de persistência objeto oferece suporte a tipos de mapeamento explícito e padrão entre propriedades de classe e atributos de tabela.

- Mapeamento explícito...Para mapear uma propriedade até uma chave primária, você deve usar oDynamoDBHashKeyeDynamoDBRangeKeyatributos de modelo de persistência de objeto. Além disso, referente aos atributos de chave não primária, se um nome de uma propriedade em sua classe e no

atributo da tabela correspondente ao qual você deseja mapear não forem os mesmos, você deve definir o mapeamento adicionando explicitamente o `DynamoDBPropertyAttribute`.

No exemplo anterior, o `Id` mapeia para a chave primária com o mesmo nome e a `propriedadeBookAuthors` mapeia para a `propriedadeAuthors` atributo no `ProductCatalogTabela` do.

- Mapeamento padrão — por padrão, o modelo de persistência objeto mapeia as propriedades da classe para os atributos com o mesmo nome na tabela.

No exemplo anterior, as propriedades `Title` e `ISBN` mapeiam para os atributos com o mesmo nome no `ProductCatalogTabela` do.

Não é necessário mapear cada propriedade de classe. Para identificar essas propriedades, adicione o `DynamoDBIgnoreAttribute`. Quando você salva um `Book` para a tabela, a instância `DynamoDBContext` não inclui o `CoverPage` propriedade. Ele também não retorna essa propriedade quando você recupera a instância de livro.

É possível mapear propriedades de tipos primitivos .NET, como `int` e `string`. Também é possível mapear qualquer tipo de dados arbitrário, desde que você forneça um conversor apropriado para mapear os dados arbitrários para um dos tipos do DynamoDB. Para saber mais sobre o mapeamento de tipos arbitrários, consulte [Mapeamento de dados arbitrários com o DynamoDB usando o AWS SDK for .NET](#) [Modelo de persistência de objeto](#) (p. 311).

O modelo de persistência objeto oferece suporte para bloqueio otimista. Durante uma operação de atualização, isso garante que você tenha a cópia mais recente do item que está prestes a atualizar. Para mais informações, consulte [Bloqueio otimista usando um número de versão com o DynamoDB usando o AWS SDK for .NET](#) [Modelo de persistência de objeto](#) (p. 309).

Attribute DynamoDB

Esta seção descreve os atributos que o modelo de persistência objeto oferece, para que você possa mapear suas classes e propriedades para tabelas e atributos do DynamoDB do.

Note

Nos seguintes atributos, apenas `DynamoDBTable` e `DynamoDBHashKey` são necessários.

DynamoDBGlobalSecondaryIndexHashKey

Mapeia uma propriedade de classe para a chave de partição de um índice secundário global. Use esse atributo se você precisa realizar uma operação em um `Query` um índice secundário global.

DynamoDBGlobalSecondaryIndexRangeKey

Mapeia uma propriedade de classe para a chave de classificação de um índice secundário global. Use esse atributo se você precisa realizar uma operação em um `Query` um índice secundário global e deseja refinar seus resultados usando a chave de classificação de índice.

DynamoDBHashKey

Mapeia uma propriedade de classe para a chave de partição da chave primária da tabela. Os atributos de chave primária não podem ser um tipo de coleção.

O exemplo de código C# a seguir mapeia a `Book` classe para a `ProductCatalog`, e a tabela `Id` para acessar a chave de partição de chave primária da tabela.

```
[DynamoDBTable("ProductCatalog")]
public class Book {
```

```
[DynamoDBHashKey]
public int Id { get; set; }

// Additional properties go here.
}
```

DynamoDBIgnore

Indica que a propriedade associada deve ser ignorada. Se não quiser salvar nenhuma das suas propriedades de classe, você poderá adicionar esse atributo para instruir a `DynamoDBContext` para não incluir essa propriedade ao salvar objetos na tabela.

DynamoDBLocalSecondaryIndexRangeKey

Mapeia uma propriedade de classe para a chave de classificação de um índice secundário local. Use esse atributo se você precisa realizar uma operação em um `Query` em um índice secundário local e deseja refinar seus resultados usando a chave de classificação de índice.

DynamoDBProperty

Mapeia uma propriedade de classe para um atributo de tabela. Se a propriedade de classe for mapeada para um atributo de tabela com o mesmo nome, você não precisará especificar esse atributo. No entanto, se os nomes não forem iguais, será possível usar essa tag para fornecer o mapeamento. Na seguinte instrução C#, o `DynamoDBProperty` mapeia a `BookAuthors` para a propriedade `Authors` atributo na tabela.

```
[DynamoDBProperty("Authors")]
public List<string> BookAuthors { get; set; }
```

`DynamoDBContext` usa essas informações de mapeamento para criar o `Authors` ao salvar dados de objeto para a tabela correspondente.

DynamoDBRenamable

Especifica um nome alternativo para uma propriedade de classe. Isso é útil quando você está escrevendo um conversor personalizado para o mapeamento de dados arbitrários para uma tabela do DynamoDB na qual o nome de uma propriedade de classe é diferente de um atributo da tabela.

DynamoDBRangeKey

Mapeia uma propriedade de classe para a chave de classificação da chave primária da tabela. Se a tabela tiver uma chave primária composta (chave de partição e chave de classificação), você deverá especificar tanto o `DynamoDBHashKey` e `DynamoDBRangeKey` atributos em seu mapeamento de classe.

Por exemplo, a tabela de amostra `ReplyItem` tem uma chave primária feita do `Id` da chave de partição e `Replenishment` da chave de classificação. O exemplo de código C# a seguir mapeia a `Reply` classe para a `ReplyTabela` do. A definição de classe também indica que duas de suas propriedades são mapeadas para a chave primária.

Para obter mais informações sobre tabelas de exemplo, consulte [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#).

```
[DynamoDBTable("Reply")]
public class Reply {
    [DynamoDBHashKey]
    public int ThreadId { get; set; }
    [DynamoDBRangeKey]
```

```
    public string Replenishment { get; set; }
    // Additional properties go here.
}
```

DynamoDBTable

Identifica a tabela de destino no DynamoDB para a qual a classe é mapeada. Por exemplo, o exemplo de código C# a seguir mapeia a propriedade `Developer` para a `People` no DynamoDB.

```
[DynamoDBTable("People")]
public class Developer { ... }
```

Esse atributo pode ser herdado ou substituído.

- O `DynamoDBTable` atributo pode ser herdado. No exemplo anterior, se você adicionar uma nova classe, `Lead`, que herda da `Developer`, ele também mapeia para a classe `People` tabela do. Ambos os `Developer` e `Lead` objetos são armazenados na `People` tabela do.
- O `DynamoDBTable` também é possível substituir. No exemplo de código C# a seguir, o `Manager` herda da classe `Developer`. No entanto, a adição explícita do `DynamoDBTable` mapeia a classe para outra tabela (`Managers`).

```
[DynamoDBTable("Managers")]
public class Manager : Developer { ... }
```

Você pode adicionar o parâmetro opcional `LowerCamelCaseProperties` para solicitar que o DynamoDB coloque em minúscula a primeira letra do nome da propriedade ao armazenar os objetos em uma tabela, como mostra o exemplo C# a seguir.

```
[DynamoDBTable("People", LowerCamelCaseProperties=true)]
public class Developer {
    string developerName;
    ...
}
```

Ao salvar instâncias do `Developer` classe, `DynamoDBContext` save a `developerName` como `adeveloperName`.

DynamoDBVersion

Identifica uma propriedade de classe para armazenar o número de versão do item. Para obter mais informações sobre controle de versão, consulte [Bloqueio otimista usando um número de versão com o DynamoDB usando o AWS SDK for .NET](#) [Modelo de persistência de objeto](#) (p. 309).

Classe DynamoDBContext

O `DynamoDBContext` classe é o ponto de entrada para o banco de dados do Amazon DynamoDB. Ele fornece uma conexão ao DynamoDB e permite que você acesse seus dados em várias tabelas, realize várias operações CRUD e realize consultas. O `DynamoDBContext` classe fornece os seguintes métodos.

CreateMultiTableBatchGet

Cria um `MultiTableBatchGet` objeto, formado por vários indivíduos `BatchGet` objetos. Cada um destes `BatchGet` objetos podem ser usados para recuperar itens de uma única tabela do DynamoDB.

Para recuperar os itens de tabelas, use o `ExecuteBatchGet`, passando o método `MultiTableBatchGet` objeto como um parâmetro.

CreateMultitableBatchWrite

Cria um `MultiTableBatchWriteObject` que é formado por vários `BatchWriteObject`s. Cada um desses `BatchWriteObject`s podem ser usados para gravar ou excluir itens em uma única tabela do DynamoDB.

Para escrever em tabelas, use a opção `ExecuteBatchWrite`, passando o método `MultiTableBatchWriteObject` como um parâmetro.

CreateBatchGet

Cria um `BatchGetObject` que você pode usar para recuperar vários itens de uma tabela do. Para mais informações, consulte [Obter Batch: Obtendo vários itens](#) (p. 316).

CreateBatchWrite

Cria um `BatchWriteObject` que você pode usar para inserir vários itens em uma tabela ou para excluir vários itens de uma tabela. Para mais informações, consulte [Gravação em Batch: Como inserir e excluir vários itens](#) (p. 314).

Delete

Exclui um item da tabela. O método requer a chave primária do item que você deseja excluir. É possível fornecer o valor da chave primária ou um objeto no lado do cliente que contém um valor de chave primária como um parâmetro para esse método.

- Se você especificar um objeto no lado do cliente como um parâmetro e tiver habilitado o bloqueio otimista, a exclusão apenas será bem-sucedida se as versões no lado do cliente e no lado do servidor desse objeto corresponderem.
- Se você especificar somente o valor da chave primária como parâmetro, a exclusão será bem-sucedida, independentemente de você ter habilitado ou não o bloqueio otimista.

Note

Para realizar essa operação em segundo plano, use o método `DeleteAsync` em vez disso.

Dispose

Descarta todos os recursos gerenciados e não gerenciados.

ExecuteBatchGet

Lê dados de uma ou mais tabelas, processando todos os objetos `BatchGetObject` em um `MultiTableBatchGetObject`.

Note

Para realizar essa operação em segundo plano, use o método `ExecuteBatchGetAsync` em vez disso.

ExecuteBatchWrite

Grava ou exclui dados em uma ou mais tabelas, processando todos os objetos `BatchWriteObject` em um `MultiTableBatchWriteObject`.

Note

Para realizar essa operação em segundo plano, use o método `ExecuteBatchWriteAsync` em vez disso.

FromDocument

Considerando uma instância de `Document`, o método `FromDocument` retorna uma instância de uma classe no lado do cliente.

Isso será útil se você quiser usar as classes de modelo de documento junto com o modelo de persistência de objeto para realizar qualquer operação de dados. Para obter mais informações sobre as classes do modelo de documento fornecidas pelo AWS SDK for .NET, consulte [.NET: Modelo de documento \(p. 276\)](#).

Suponha que você tenha um `Document` chamado `doc`, que contém uma representação de `ForumItem`. (Para ver como construir esse objeto, consulte a descrição de `ToDocument` mais adiante neste tópico.) Você pode usar `FromDocument` para recuperar o `Forum` a partir de `doc`, como mostrado no exemplo de código C# a seguir.

Example

```
forum101 = context.FromDocument<Forum>(101);
```

Note

Se suas `Document` implementa o objeto `IEnumerable`, você pode usar `FromDocuments` em vez disso. Isso permite uma iteração sobre todas as instâncias de classe `Document`.

FromQuery

Executa um `Query`, com os parâmetros de consulta definidos em `QueryOperationConfig` objeto.

Note

Para realizar essa operação em segundo plano, use o método `FromQueryAsync` em vez disso.

FromScan

Executa um `Scan`, com os parâmetros de verificação definidos em `ScanOperationConfig` objeto.

Note

Para realizar essa operação em segundo plano, use o método `FromScanAsync` em vez disso.

GetTargetTable

Recupera a tabela de destino para o tipo especificado. Isso é útil quando você está escrevendo um conversor personalizado para o mapeamento de dados arbitrários para uma tabela do DynamoDB e precisa determinar qual tabela está associada a um tipo de dados personalizado.

Load

Recupera um item de uma tabela. O método requer somente a chave primária do item que você deseja recuperar.

Por padrão, o DynamoDB retorna o item com valores que são eventualmente consistentes. Para obter informações sobre o modelo de consistência eventual, consulte [Consistência de leituras \(p. 17\)](#).

Note

Para realizar essa operação em segundo plano, use o método `LoadAsync` em vez disso.

Query

Consulta uma tabela com base em parâmetros de consulta que você fornece.

Você poderá consultar uma tabela somente se ela tiver uma chave primária composta (chave de partição e chave de classificação). Ao consultar, você deve especificar uma chave de partição e uma condição que se aplique à chave de classificação.

Suponha que você tenha um lado do cliente Reply mapeada para a classe Reply no DynamoDB. O exemplo de código C# a seguir consulta o Reply Para localizar respostas a tópicos do fórum postadas nos últimos 15 dias. O Reply tem uma chave primária que tem a propriedade Id da chave de partição Reply Date Time como chave de classificação. Para obter mais informações sobre o Reply, consulte [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#).

Example

```
DynamoDBContext context = new DynamoDBContext(client);

string replyId = "DynamoDB#DynamoDB Thread 1"; //Partition key
DateTime twoWeeksAgoDate = DateTime.UtcNow.Subtract(new TimeSpan(14, 0, 0, 0)); // Date to
// compare.
IEnumerable<Reply> latestReplies = context.Query<Reply>(replyId, QueryOperator.GreaterThan,
twoWeeksAgoDate);
```

Isso retorna uma coleção de objetos Reply.

O `Query` método retorna um “carregamento preguiçoso” `IEnumerable` Coleção. Ele inicialmente retorna apenas uma página de resultados e, em seguida, faz uma chamada de serviço para a próxima página, se necessário. Para obter todos os itens correspondentes, você precisa fazer uma iteração somente na `IEnumerable`.

Se a sua tabela tiver uma chave primária simples (chave de partição), você não poderá usar o `Query` Método do. Em vez disso, poderá usar o método `Load` e fornecer a chave de partição para recuperar o item.

Note

Para realizar essa operação em segundo plano, use o método `QueryAsync` em vez disso.

Save (Salvar)

Salva o objeto especificado na tabela. Se a chave primária especificada no objeto de entrada não existir na tabela, o método adicionará um novo item à tabela. Se a chave primária existir, o método atualizará o item existente.

Se você tiver o bloqueio otimista configurado, a atualização será bem-sucedida somente se as versões do item no lado do cliente e no lado do servidor corresponderem. Para mais informações, consulte [Bloqueio otimista usando um número de versão com o DynamoDB usando o AWS SDK for .NET](#) Modelo de persistência de objeto (p. 309).

Note

Para realizar essa operação em segundo plano, use o método `SaveAsync` em vez disso.

Scan

Realiza uma verificação de tabela inteira.

É possível filtrar os resultados da verificação, especificando uma condição de verificação. A condição pode ser avaliada em qualquer atributo da tabela. Suponha que você tenha uma classe no lado do

clienteBook mapeado para o ProductCatalog no DynamoDB. O exemplo C# a seguir verifica a tabela e retorna apenas os itens de livros com preços inferiores a 0.

Example

```
IEnumerable<Book> itemsWithWrongPrice = context.Scan<Book>(
    new ScanCondition("Price", ScanOperator.LessThan, price),
    new ScanCondition("ProductCategory", ScanOperator.Equal, "Book")
);
```

O `Scan` método retorna um “carregamento preguiçoso” `IEnumerable` Coleção. Ele inicialmente retorna apenas uma página de resultados e, em seguida, faz uma chamada de serviço para a próxima página, se necessário. Para obter todos os itens correspondentes, você só precisa fazer uma iteração na `IEnumerable`.

Por motivos de desempenho, você deve consultar suas tabelas e evitar uma verificação de tabela.

Note

Para realizar essa operação em segundo plano, use o método `ScanAsync` em vez disso.

ToDocument

Retorna uma instância da classe de modelo de documento `Document` da sua instância de classe.

Isso será útil se você quiser usar as classes de modelo de documento junto com o modelo de persistência de objeto para realizar qualquer operação de dados. Para obter mais informações sobre as classes do modelo de documento fornecidas pelo AWS SDK for .NET, consulte [.NET: Modelo de documento \(p. 276\)](#).

Suponha que você tenha uma classe no lado do cliente mapeada para o exemplo do `Forum` Tabela do. Em seguida, você pode usar um `DynamoDBContext` para obter um item como `Document` a partir do objeto `Forum`, conforme mostrado no exemplo de código C# a seguir.

Example

```
DynamoDBContext context = new DynamoDBContext(client);

Forum forum101 = context.Load<Forum>(101); // Retrieve a forum by primary key.
Document doc = context.ToDocument<Forum>(forum101);
```

Como especificar parâmetros opcionais para `DynamoDBContext`

Ao usar o modelo de persistência de objeto, você pode especificar os seguintes parâmetros opcionais para `DynamoDBContext`.

- **ConsistentRead**—Ao recuperar dados usando o `Load`, `Query`, ou `Scan` operações, você pode adicionar esse parâmetro opcional para solicitar os valores mais recentes para os dados.
- **IgnoreNullValues**—Este parâmetro informa `DynamoDBContext` que ignore valores nulos em atributos durante uma `Save` operação. Se esse parâmetro for `false` (ou se não estiver definido), um valor nulo será interpretado como uma diretiva para excluir o atributo específico.
- **SkipVersionCheck**—Este parâmetro informa `DynamoDBContext` para não comparar versões ao salvar ou excluir um item. Para obter mais informações sobre controle de versão, consulte [Bloqueio otimista usando um número de versão com o DynamoDB usando o AWS SDK for .NET](#) [Modelo de persistência de objeto \(p. 309\)](#).
- **TableNamePrefix**—Prefixa todos os nomes de tabelas com uma string específica. Se esse parâmetro for nulo (ou se não estiver definido), nenhum prefixo será usado.

O exemplo de código C# a seguir cria um novo `DynamoDBContext` especificando dois dos parâmetros opcionais anteriores.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
...
DynamoDBContext context =
    new DynamoDBContext(client, new DynamoDBContextConfig { ConsistentRead = true,
    SkipVersionCheck = true});
```

`DynamoDBContext` inclui esses parâmetros opcionais com cada solicitação enviada usando esse contexto.

Em vez de definir esses parâmetros no `DynamoDBContext`, você pode especificá-los para operações individuais que você executa usando `DynamoDBContext`, como mostrado no exemplo de código C# a seguir. O exemplo carrega um item de livro específico. O método `Load` de `DynamoDBContext` especifica os parâmetros opcionais anteriores.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
...
DynamoDBContext context = new DynamoDBContext(client);
Book bookItem = context.Load<Book>(productId,new DynamoDBContextConfig{ ConsistentRead =
    true, SkipVersionCheck = true });
```

Neste caso, `DynamoDBContext` inclui esses parâmetros somente ao enviar o `Get` solicitação.

Tipos de dados compatíveis

O modelo de persistência de objeto oferece suporte a um conjunto de tipos de dados .NET, coleções e tipos de dados arbitrários primitivos. O modelo é compatível com os seguintes tipos de dados primitivos.

- `bool`
- `byte`
- `char`
- `DateTime`
- `decimal`
- `double`
- `float`
- `Int16`
- `Int32`
- `Int64`
- `SByte`
- `string`
- `UInt16`
- `UInt32`
- `UInt64`

O modelo de persistência objeto também oferece suporte aos tipos de coleção .NET. `DynamoDBContext` é capaz de converter tipos de coleção concretos e objetos CLR básicos (POCOs).

A tabela a seguir resume o mapeamento de tipos .NET anteriores para os tipos do DynamoDB.

Tipo primitivo .NET	Tipo DynamoDB
Todos os tipos de número	N (tipo Número)
Todos os tipos de string	S (tipo String)
MemoryStream, byte[]	B (tipo Binário)
bool	N(tipo Número). 0 representa false e 1 representa true.
Tipos de coleção	Tipo BS (conjunto binário), tipo SS (conjunto de strings) e tipo NS (conjunto de números)
DateTime	S (tipo String). O DateTimeOs valores do são armazenados como strings formatadas em ISO-8601.

O modelo de persistência objeto também oferece suporte a tipos de dados arbitrários. No entanto, você deve fornecer o código de conversor para mapear os tipos complexos para tipos do DynamoDB.

Note

- Valores binários vazios são compatíveis.
- A leitura de valores string vazios é compatível. Os valores de atributos string vazios são compatíveis nos valores de atributos do tipo de conjunto string durante a gravação no DynamoDB. Os valores de atributos string vazios do tipo string e os valores string vazios contidos no tipo Lista ou Mapa são descartados das solicitações de gravação

Bloqueio otimista usando um número de versão com o DynamoDB usando o AWS SDK for .NETModelo de persistência de objeto

O suporte para bloqueio otimista no modelo de persistência de objeto garante que a versão do item para o seu aplicativo seja igual à versão do item no lado do servidor antes que esse item seja atualizado ou excluído. Suponha que você recupere um item para atualização. No entanto, antes de você retornar suas atualizações, outro aplicativo atualiza o mesmo item. Agora, o aplicativo tem uma cópia obsoleta do item. Sem o bloqueio otimista, qualquer atualização que você realizar substituirá a atualização feita pelo outro aplicativo.

O recurso de bloqueio otimista do modelo de persistência de objeto fornece o `DynamoDBVersionTag` que você pode usar para habilitar o bloqueio otimista. Para usar esse recurso, adicione uma propriedade à sua classe para armazenar o número de versão. Você adiciona o `DynamoDBVersion` atributo para a propriedade. Quando você salva o objeto pela primeira vez, o `DynamoDBContext` atribui um número de versão e incrementa esse valor cada vez que você atualizar o item.

Sua solicitação de atualização ou exclusão só será bem-sucedida se a versão do objeto no lado do cliente corresponder ao número de versão correspondente do item no lado do servidor. Se o seu aplicativo tiver uma cópia obsoleta, ele deverá obter a versão mais recente do servidor antes de poder atualizar ou excluir o item.

O exemplo de código C# a seguir define um `BookClasse` com atributos de persistência de objeto, mapeando-a para o `ProductCatalogTabela` do. A propriedade `VersionNumber` na classe decorada com o atributo `DynamoDBVersion` armazena o valor do número de versão.

Example

```
[DynamoDBTable("ProductCatalog")]
public class Book
{
    [DynamoDBHashKey] //Partition key
    public int Id { get; set; }
    [DynamoDBProperty]
    public string Title { get; set; }
    [DynamoDBProperty]
    public string ISBN { get; set; }
    [DynamoDBProperty("Authors")]
    public List<string> BookAuthors { get; set; }
    [DynamoDBVersion]
    public int? VersionNumber { get; set; }
}
```

Note

Você pode aplicar o atributo `DynamoDBVersion` apenas a um tipo primitivo numérico anulável (como `int?`).

O bloqueio otimista tem o seguinte impacto sobre operações `DynamoDBContext`:

- Para um novo item, `DynamoDBContext` atribui o número de versão inicial 0. Se você recuperar um item existente, atualizar uma ou mais das suas propriedades e tentar salvar as alterações, a operação Salvar será bem-sucedida somente se o número de versão no lado do cliente e no lado do servidor corresponder. `DynamoDBContext` incrementa o número da versão. Você não precisa definir o número de versão.
- O `Delete` método fornece sobrecargas que podem usar um valor de chave primária ou um objeto como parâmetro, como mostra o exemplo de código C# a seguir.

Example

```
DynamoDBContext context = new DynamoDBContext(client);
...
// Load a book.
Book book = context.Load<ProductCatalog>(111);
// Do other operations.
// Delete 1 - Pass in the book object.
context.Delete<ProductCatalog>(book);

// Delete 2 - Pass in the Id (primary key)
context.Delete<ProductCatalog>(222);
```

Se você fornecer um objeto como parâmetro, a exclusão só será bem-sucedida se a versão do objeto corresponder à versão de item no lado do servidor correspondente. No entanto, se você fornecer um valor de chave primária como parâmetro, o `DynamoDBContext` não tem conhecimento de números de versão, e exclui o item sem fazer a verificação de versão.

Observe que a implementação interna do bloqueio otimista no código do modelo de persistência de objeto usa as ações de API de exclusão condicional e exclusão condicional no DynamoDB.

Como desabilitar o bloqueio otimista

Para desabilitar o bloqueio otimista, use o `skipVersionCheck` propriedade de configuração. Você pode definir essa propriedade ao criar `DynamoDBContext`. Nesse caso, o bloqueio otimista está desabilitado para solicitações feitas usando o contexto. Para mais informações, consulte [Como especificar parâmetros opcionais para `DynamoDBContext` \(p. 307\)](#).

Em vez de definir a propriedade no nível do contexto, você pode desabilitar o bloqueio otimista para uma operação específica, como mostra o exemplo de código C# a seguir. O exemplo usa o contexto para excluir um item de livro. O `Deleted` define o método opcional `SkipVersionCheck` para `true`, desabilitando a verificação de versão.

Example

```
DynamoDBContext context = new DynamoDBContext(client);
// Load a book.
Book book = context.Load<ProductCatalog>(111);
...
// Delete the book.
context.Delete<Book>(book, new DynamoDBContextConfig { SkipVersionCheck = true });
```

Mapeamento de dados arbitrários com o DynamoDB usando o AWS SDK for .NET Modelo de persistência de objeto

Além dos tipos de .NET suportados (consulte [Tipos de dados compatíveis \(p. 308\)](#)), você pode usar tipos no seu aplicativo para os quais não há um mapeamento direto para os tipos do Amazon DynamoDB. O modelo de persistência de objeto oferece suporte ao armazenamento de dados de tipos arbitrários, desde que você forneça o conversor para converter dados do tipo arbitrário no tipo do DynamoDB, e vice-versa. O código de conversor transforma os dados durante os processos de salvar e carregar os objetos.

Você pode criar qualquer tipo no lado do cliente. No entanto, os dados armazenados nas tabelas são um dos tipos do DynamoDB e, durante a consulta e a verificação, qualquer comparação de dados feita se baseia nos dados armazenados no DynamoDB.

O exemplo de código C# a seguir define um `Book` classe com `Id`, `Title`, `ISBN`, `eDimension`.`properties` `ODimension` é da propriedade `DimensionType` que descreve `Height`, `Width`, `eThickness`.`properties` O código de exemplo fornece os métodos de conversor `ToEntry` e `FromEntry` para converter dados entre `ODimensionType` e os tipos de string do DynamoDB. Por exemplo, ao salvar um `Book` instância, o conversor cria um `livro` `ODimension` string como "8.5x11x.05". Quando você recupera um livro, converte a string em um `ODimensionType` instância.

O exemplo mapeia o `Book` para o `ProductCatalog` Tabela do. Ele salva uma amostra `Book` instância, recupera essa instância, atualiza suas dimensões e salva o `Book` novamente.

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

Example

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
using System;
using System.Collections.Generic;
```

```
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.DataModel;
using Amazon.DynamoDBv2.DocumentModel;
using Amazon.Runtime;
using Amazon.SecurityToken;

namespace com.amazonaws.codesamples
{
    class HighLevelMappingArbitraryData
    {
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();

        static void Main(string[] args)
        {
            try
            {
                DynamoDBContext context = new DynamoDBContext(client);

                // 1. Create a book.
                DimensionType myBookDimensions = new DimensionType()
                {
                    Length = 8M,
                    Height = 11M,
                    Thickness = 0.5M
                };

                Book myBook = new Book
                {
                    Id = 501,
                    Title = "AWS SDK for .NET Object Persistence Model Handling Arbitrary
Data",
                    ISBN = "999-9999999999",
                    BookAuthors = new List<string> { "Author 1", "Author 2" },
                    Dimensions = myBookDimensions
                };

                context.Save(myBook);

                // 2. Retrieve the book.
                Book bookRetrieved = context.Load<Book>(501);

                // 3. Update property (book dimensions).
                bookRetrieved.Dimensions.Height += 1;
                bookRetrieved.Dimensions.Length += 1;
                bookRetrieved.Dimensions.Thickness += 0.2M;
                // Update the book.
                context.Save(bookRetrieved);

                Console.WriteLine("To continue, press Enter");
                Console.ReadLine();
            }
            catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message); }
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
            catch (Exception e) { Console.WriteLine(e.Message); }
        }
    }
}

[DynamoDBTable("ProductCatalog")]
public class Book
{
    [DynamoDBHashKey] //Partition key
    public int Id
    {
        get; set;
    }
    [DynamoDBProperty]
    public string Title
```

```
{  
    get; set;  
}  
[DynamoDBProperty]  
public string ISBN  
{  
    get; set;  
}  
// Multi-valued (set type) attribute.  
[DynamoDBProperty("Authors")]  
public List<string> BookAuthors  
{  
    get; set;  
}  
// Arbitrary type, with a converter to map it to DynamoDB type.  
[DynamoDBProperty(typeof(DimensionTypeConverter))]  
public DimensionType Dimensions  
{  
    get; set;  
}  
}  
  
public class DimensionType  
{  
    public decimal Length  
    {  
        get; set;  
    }  
    public decimal Height  
    {  
        get; set;  
    }  
    public decimal Thickness  
    {  
        get; set;  
    }  
}  
  
// Converts the complex type DimensionType to string and vice-versa.  
public class DimensionTypeConverter : IPropertyConverter  
{  
    public DynamoDBEntry ToEntry(object value)  
    {  
        DimensionType bookDimensions = value as DimensionType;  
        if (bookDimensions == null) throw new ArgumentOutOfRangeException();  
  
        string data = string.Format("{1}{0}{2}{0}{3}", " x ",  
                                     bookDimensions.Length, bookDimensions.Height,  
                                     bookDimensions.Thickness);  
  
        DynamoDBEntry entry = new Primitive  
        {  
            Value = data  
        };  
        return entry;  
    }  
  
    public object FromEntry(DynamoDBEntry entry)  
    {  
        Primitive primitive = entry as Primitive;  
        if (primitive == null || !(primitive.Value is String) ||  
string.IsNullOrEmpty((string)primitive.Value))  
            throw new ArgumentOutOfRangeException();  
  
        string[] data = ((string)(primitive.Value)).Split(new string[] { " x " },  
StringSplitOptions.None);  
    }  
}
```

```
        if (data.Length != 3) throw new ArgumentOutOfRangeException();

        DimensionType complexData = new DimensionType
        {
            Length = Convert.ToDecimal(data[0]),
            Height = Convert.ToDecimal(data[1]),
            Thickness = Convert.ToDecimal(data[2])
        };
        return complexData;
    }
}
```

Operações em Batch usando o AWS SDK for .NETModelo de persistência de objeto

Gravação em Batch: Como inserir e excluir vários itens

Para inserir ou excluir vários objetos de uma tabela em uma única solicitação, faça o seguinte:

- Execute o `CreateBatchWrite` método do `DynamoDBContext`, e crie uma instância `doBatchWriteClasse`.
- Especifique os itens que você deseja inserir ou excluir.
 - Para inserir um ou mais itens, use o método `AddPutItem` ou `AddPutItems`.
 - Para excluir um ou mais itens, você pode especificar a chave primária do item ou um objeto no lado do cliente que é mapeado para o item que você deseja excluir. Use os métodos `AddDeleteItem`, `AddDeleteItems` e `AddDeleteKey` para especificar a lista de itens para exclusão.
- Chame o método `BatchWrite.Execute` para inserir e excluir todos os itens especificados da tabela.

Note

Ao usar o modelo de persistência de objeto, você pode especificar qualquer número de operações em um lote. No entanto, observe que o Amazon DynamoDB limita o número de operações em lote e o tamanho total do lote em uma operação em lote. Para obter mais informações sobre os limites específicos, consulte [BatchWriteItem](#). Se a API detectar que sua solicitação de gravação em lote excedeu o número permitido de solicitações de gravação ou excedeu o tamanho máximo de carga útil HTTP permitido, ela dividirá esse lote em vários lotes menores. Além disso, se uma resposta a uma gravação em lote retornar itens não processados, a API enviará automaticamente outra solicitação em lote com esses itens não processados.

Suponha que você definiu uma classe C# `Book` que mapeia para a classe `ProductCatalog` no DynamoDB. O exemplo de código C# a seguir usa o `BatchWrite` para carregar dois itens e excluir um item da `ProductCatalog` tabela do.

Example

```
DynamoDBContext context = new DynamoDBContext(client);
var bookBatch = context.CreateBatchWrite<Book>();

// 1. Specify two books to add.
Book book1 = new Book
{
    Id = 902,
    ISBN = "902-11-11111",
    ProductCategory = "Book",
    Title = "My book3 in batch write"
};
```

```
Book book2 = new Book
{
    Id = 903,
    ISBN = "903-11-11-1111",
    ProductCategory = "Book",
    Title = "My book4 in batch write"
};

bookBatch.AddPutItems(new List<Book> { book1, book2 });

// 2. Specify one book to delete.
bookBatch.AddDeleteKey(111);

bookBatch.Execute();
```

Para inserir ou excluir objetos de várias tabelas, faça o seguinte:

- Crie uma instância da classe `BatchWrite` para cada tipo e especifique os itens que você deseja inserir ou excluir, conforme descrito na seção anterior.
- Crie uma instância de `MultiTableBatchWrite` usando um dos seguintes métodos:
 - Execute a `Combine` em um dos `BatchWrite`s os objetos que você criou na etapa anterior.
 - Crie uma instância do tipo `MultiTableBatchWrite`, fornecendo uma lista de objetos `BatchWrite`.
 - Execute a `CreateMultiTableBatchWrite` Método do `DynamoDBContext` transmite sua lista de `BatchWrite` Objetos.
- Chame o método `Execute` de `MultiTableBatchWrite`, que realiza as operações de inserção e exclusão especificadas em várias tabelas.

Suponha que você definiu `Forum` e `Thread` Classes C# que mapeiam para o `Forum` e `Thread` tabelas no DynamoDB. Além disso, suponha que o `Thread` classe tem o versionamento habilitado. Como o versionamento não tem suporte ao usar operações em lote, você deve desabilitar explicitamente o versionamento, como mostra o exemplo de código C# a seguir. O exemplo usa `MultiTableBatchWrite` O objeto para realizar uma atualização em várias tabelas.

Example

```
DynamoDBContext context = new DynamoDBContext(client);
// Create BatchWrite objects for each of the Forum and Thread classes.
var forumBatch = context.CreateBatchWrite<Forum>();

DynamoDBOperationConfig config = new DynamoDBOperationConfig();
config.SkipVersionCheck = true;
var threadBatch = context.CreateBatchWrite<Thread>(config);

// 1. New Forum item.
Forum newForum = new Forum
{
    Name = "Test BatchWrite Forum",
    Threads = 0
};
forumBatch.AddPutItem(newForum);

// 2. Specify a forum to delete by specifying its primary key.
forumBatch.AddDeleteKey("Some forum");

// 3. New Thread item.
Thread newThread = new Thread
{
    ForumName = "Amazon S3 forum",
    Subject = "My sample question",
```

```
    KeywordTags = new List<string> { "Amazon S3", "Bucket" },
    Message = "Message text"
};

threadBatch.AddPutItem(newThread);

// Now run multi-table batch write.
var superBatch = new MultiTableBatchWrite(forumBatch, threadBatch);
superBatch.Execute();
```

Para obter um exemplo de trabalho, consulte [Exemplo: Operação de gravação em Batch usando o AWS SDK for .NET](#)[Modelo de persistência de objeto \(p. 319\)](#).

Note

A API de lote do DynamoDB limita o número de gravações em lote e também o tamanho do lote. Para obter mais informações, consulte [BatchWriteItem](#). Ao usar a API do modelo de persistência de objeto .NET, é possível especificar qualquer número de operações. No entanto, se o número de operações em um lote ou o tamanho exceder o limite, a API .NET dividirá a solicitação de gravação em lote em lotes menores e enviará várias solicitações de gravação em lote ao DynamoDB.

Obter Batch: Obtendo vários itens

Para recuperar vários itens de uma tabela em uma única solicitação, faça o seguinte:

- Crie uma instância da classe `CreateBatchGet`.
- Especifique uma lista de chaves primárias para recuperar.
- Chame o método `Execute`. A resposta retorna os itens na propriedade `Results`.

O exemplo de código C# a seguir recupera três itens do `ProductCatalogTabela`. Os itens no resultado não estão necessariamente na mesma ordem em que você especificou as chaves primárias.

Example

```
DynamoDBContext context = new DynamoDBContext(client);
var bookBatch = context.CreateBatchGet<ProductCatalog>();
bookBatch.AddKey(101);
bookBatch.AddKey(102);
bookBatch.AddKey(103);
bookBatch.Execute();
// Process result.
Console.WriteLine(bookBatch.Results.Count);
Book book1 = bookBatch.Results[0];
Book book2 = bookBatch.Results[1];
Book book3 = bookBatch.Results[2];
```

Para recuperar objetos de várias tabelas, faça o seguinte:

- Para cada tipo, criar uma instância do tipo `CreateBatchGet` e forneça os valores de chave primária que você deseja recuperar de cada tabela.
- Crie uma instância da classe `MultiTableBatchGet` usando um dos seguintes métodos:
 - Execute a `Combine` em um dos `BatchGet` que você criou na etapa anterior.
 - Crie uma instância do tipo `MultiBatchGet`, fornecendo uma lista de objetos `BatchGet`.
 - Execute a `CreateMultiTableBatchGet` Método do `DynamoDBContext` transmite sua lista de `BatchGet` objetos.
- Chame a `Execute` Método do `MultiTableBatchGet`, que retorna os resultados com tipo definido no `BatchGet` objetos.

O exemplo de código C# a seguir recupera vários itens do `OrderDetail` usando as tabelas `CreateBatchGet` método do.

Example

```
var orderBatch = context.CreateBatchGet<Order>();
orderBatch.AddKey(101);
orderBatch.AddKey(102);

var orderDetailBatch = context.CreateBatchGet<OrderDetail>();
orderDetailBatch.AddKey(101, "P1");
orderDetailBatch.AddKey(101, "P2");
orderDetailBatch.AddKey(102, "P3");
orderDetailBatch.AddKey(102, "P1");

var orderAndDetailSuperBatch = orderBatch.Combine(orderDetailBatch);
orderAndDetailSuperBatch.Execute();

Console.WriteLine(orderBatch.Results.Count);
Console.WriteLine(orderDetailBatch.Results.Count);

Order order1 = orderBatch.Results[0];
Order order2 = orderDetailBatch.Results[1];
OrderDetail orderDetail1 = orderDetailBatch.Results[0];
```

Exemplo: Operações CRUD usando o AWS SDK for .NET Modelo de persistência de objeto

O exemplo de código C# a seguir declara um `Book` classe com `Id`, `Title`, `ISBN`, e `Authors`. `properties` O exemplo usa atributos de persistência de objeto para mapear essas propriedades para o `ProductCatalog` Tabela no Amazon DynamoDB. O exemplo usa, em seguida, o `DynamoDBContext` para ilustrar operações típicas de criação, leitura, atualização e exclusão (CRUD). O exemplo cria uma amostra `Book` a salva no `ProductCatalog` Tabela do. Em seguida, ele recupera o item do livro e atualiza sua `ISBN` e `Authors`. Observe que a atualização substitui a lista de autores existentes. Por fim, o exemplo exclui o item de livro.

Para obter mais informações sobre o `ProductCatalog` usada neste exemplo, consulte [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#). Para ver as instruções passo a passo para testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

Note

O exemplo a seguir não funciona com o .NET Core, pois ele não é compatível com métodos síncronos. Para obter mais informações, consulte [AWS APIs assíncronas da para .NET](#).

Example

```
/** 
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
```

```
using System;
using System.Collections.Generic;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.DataModel;
using Amazon.Runtime;

namespace com.amazonaws.codesamples
{
    class HighLevelItemCRUD
    {
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();

        static void Main(string[] args)
        {
            try
            {
                DynamoDBContext context = new DynamoDBContext(client);
                TestCRUDOperations(context);
                Console.WriteLine("To continue, press Enter");
                Console.ReadLine();
            }
            catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message); }
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
            catch (Exception e) { Console.WriteLine(e.Message); }
        }

        private static void TestCRUDOperations(DynamoDBContext context)
        {
            int bookID = 1001; // Some unique value.
            Book myBook = new Book
            {
                Id = bookID,
                Title = "object persistence-AWS SDK for.NET SDK-Book 1001",
                ISBN = "111-1111111001",
                BookAuthors = new List<string> { "Author 1", "Author 2" },
            };

            // Save the book.
            context.Save(myBook);
            // Retrieve the book.
            Book bookRetrieved = context.Load<Book>(bookID);

            // Update few properties.
            bookRetrieved.ISBN = "222-2222221001";
            bookRetrieved.BookAuthors = new List<string> { " Author 1", "Author x" }; // Replace existing authors list with this.
            context.Save(bookRetrieved);

            // Retrieve the updated book. This time add the optional ConsistentRead parameter using DynamoDBContextConfig object.
            Book updatedBook = context.Load<Book>(bookID, new DynamoDBContextConfig
            {
                ConsistentRead = true
            });

            // Delete the book.
            context.Delete<Book>(bookID);
            // Try to retrieve deleted book. It should return null.
            Book deletedBook = context.Load<Book>(bookID, new DynamoDBContextConfig
            {
                ConsistentRead = true
            });
            if (deletedBook == null)
                Console.WriteLine("Book is deleted");
        }
    }
}
```

```
[DynamoDBTable("ProductCatalog")]
public class Book
{
    [DynamoDBHashKey] //Partition key
    public int Id
    {
        get; set;
    }
    [DynamoDBProperty]
    public string Title
    {
        get; set;
    }
    [DynamoDBProperty]
    public string ISBN
    {
        get; set;
    }
    [DynamoDBProperty("Authors")] //String Set datatype
    public List<string> BookAuthors
    {
        get; set;
    }
}
```

Exemplo: Operação de gravação em Batch usando oAWS SDK for .NETModelo de persistência de objeto

O exemplo de código C# a seguir declara `Book`, `Forum`, `Thread`, e `Reply` e as mapeia para tabelas do Amazon DynamoDB usando os atributos do modelo de persistência de objeto.

O exemplo usa, em seguida, o `DynamoDBContext` para ilustrar as seguintes operações de gravação em lote:

- `BatchWrite` o objeto para inserir e excluir itens de livro da `ProductCatalog` tabela do.
- `MultiTableBatchWrite` o objeto para inserir e excluir itens da `Forum` e a `Thread` tabelas.

Para obter mais informações sobre as tabelas usadas neste exemplo, consulte [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#). Para ver as instruções passo a passo para testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

Note

O exemplo a seguir não funciona com o .NET Core, pois ele não é compatível com métodos síncronos. Para obter mais informações, consulte [AWS APIs assíncronas da para .NET](#).

Example

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
```

```
* This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
* CONDITIONS OF ANY KIND, either express or implied. See the License for the
* specific language governing permissions and limitations under the License.
*/
using System;
using System.Collections.Generic;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.DataModel;
using Amazon.Runtime;
using Amazon.SecurityToken;

namespace com.amazonaws.codesamples
{
    class HighLevelBatchWriteItem
    {
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();

        static void Main(string[] args)
        {
            try
            {
                DynamoDBContext context = new DynamoDBContext(client);
                SingleTableBatchWrite(context);
                MultiTableBatchWrite(context);
            }
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
            catch (Exception e) { Console.WriteLine(e.Message); }

            Console.WriteLine("To continue, press Enter");
            Console.ReadLine();
        }

        private static void SingleTableBatchWrite(DynamoDBContext context)
        {
            Book book1 = new Book
            {
                Id = 902,
                InPublication = true,
                ISBN = "902-11-11-1111",
                PageCount = "100",
                Price = 10,
                ProductCategory = "Book",
                Title = "My book3 in batch write"
            };
            Book book2 = new Book
            {
                Id = 903,
                InPublication = true,
                ISBN = "903-11-11-1111",
                PageCount = "200",
                Price = 10,
                ProductCategory = "Book",
                Title = "My book4 in batch write"
            };

            var bookBatch = context.CreateBatchWrite<Book>();
            bookBatch.AddPutItems(new List<Book> { book1, book2 });

            Console.WriteLine("Performing batch write in SingleTableBatchWrite()");
            bookBatch.Execute();
        }

        private static void MultiTableBatchWrite(DynamoDBContext context)
        {
            // 1. New Forum item.
            Forum newForum = new Forum
```

```
{  
    Name = "Test BatchWrite Forum",  
    Threads = 0  
};  
var forumBatch = context.CreateBatchWrite<Forum>();  
forumBatch.AddPutItem(newForum);  
  
// 2. New Thread item.  
Thread newThread = new Thread  
{  
    ForumName = "S3 forum",  
    Subject = "My sample question",  
    KeywordTags = new List<string> { "S3", "Bucket" },  
    Message = "Message text"  
};  
  
DynamoDBOperationConfig config = new DynamoDBOperationConfig();  
config.SkipVersionCheck = true;  
var threadBatch = context.CreateBatchWrite<Thread>(config);  
threadBatch.AddPutItem(newThread);  
threadBatch.AddDeleteKey("some partition key value", "some sort key value");  
  
var superBatch = new MultiTableBatchWrite(forumBatch, threadBatch);  
Console.WriteLine("Performing batch write in MultiTableBatchWrite().");  
superBatch.Execute();  
}  
}  
  
[DynamoDBTable("Reply")]  
public class Reply  
{  
    [DynamoDBHashKey] //Partition key  
    public string Id  
    {  
        get; set;  
    }  
  
    [DynamoDBRangeKey] //Sort key  
    public DateTime ReplyDateTime  
    {  
        get; set;  
    }  
  
    // Properties included implicitly.  
    public string Message  
    {  
        get; set;  
    }  
    // Explicit property mapping with object persistence model attributes.  
    [DynamoDBProperty("LastPostedBy")]  
    public string PostedBy  
    {  
        get; set;  
    }  
    // Property to store version number for optimistic locking.  
    [DynamoDBVersion]  
    public int? Version  
    {  
        get; set;  
    }  
}  
  
[DynamoDBTable("Thread")]  
public class Thread  
{  
    // PK mapping.
```

```
[DynamoDBHashKey]      //Partition key
public string ForumName
{
    get; set;
}
[DynamoDBRangeKey]      //Sort key
public String Subject
{
    get; set;
}
// Implicit mapping.
public string Message
{
    get; set;
}
public string LastPostedBy
{
    get; set;
}
public int Views
{
    get; set;
}
public int Replies
{
    get; set;
}
public bool Answered
{
    get; set;
}
public DateTime LastPostedDateTime
{
    get; set;
}
// Explicit mapping (property and table attribute names are different.
[DynamoDBProperty("Tags")]
public List<string> KeywordTags
{
    get; set;
}
// Property to store version number for optimistic locking.
[DynamoDBVersion]
public int? Version
{
    get; set;
}

[DynamoDBTable("Forum")]
public class Forum
{
    [DynamoDBHashKey]      //Partition key
    public string Name
    {
        get; set;
    }
    // All the following properties are explicitly mapped,
    // only to show how to provide mapping.
    [DynamoDBProperty]
    public int Threads
    {
        get; set;
    }
    [DynamoDBProperty]
    public int Views
```

```
{  
    get; set;  
}  
[DynamoDBProperty]  
public string LastPostBy  
{  
    get; set;  
}  
[DynamoDBProperty]  
public DateTime LastPostDateTime  
{  
    get; set;  
}  
[DynamoDBProperty]  
public int Messages  
{  
    get; set;  
}  
}  
  
[DynamoDBTable("ProductCatalog")]  
public class Book  
{  
    [DynamoDBHashKey] //Partition key  
    public int Id  
    {  
        get; set;  
    }  
    public string Title  
    {  
        get; set;  
    }  
    public string ISBN  
    {  
        get; set;  
    }  
    public int Price  
    {  
        get; set;  
    }  
    public string PageCount  
    {  
        get; set;  
    }  
    public string ProductCategory  
    {  
        get; set;  
    }  
    public bool InPublication  
    {  
        get; set;  
    }  
}
```

Exemplo: Consulta e digitalização no DynamoDB usando o AWS SDK for .NETModelo de persistência de objeto

O exemplo C# nesta seção define as seguintes classes e as mapeia para as tabelas no DynamoDB. Para obter mais informações sobre como criar as tabelas usadas neste exemplo, consulte [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#).

- O `Book` classe mapeia para o `ProductCatalog`Tabela do.

- As classes `Forum`, `Thread` e `Reply` mapeiam para as tabelas com o mesmo nome.

Em seguida, o exemplo executa as seguintes operações de consulta e verificação usando `DynamoDBContext`.

- Obtenha um livro por `Id`.

A tabela `ProductCatalog` tem `Id` como sua chave primária. Ela não tem uma chave de classificação como parte de sua chave primária. Portanto, você não pode consultar a tabela. É possível obter um item usando o valor `Id`.

- Execute as consultas a seguir no `ReplyTabela` do. (`OReplyA` chave primária da tabela é formada por `Id` e `ReplyDateTimeAtributos`. `OReplyDateTime` é uma chave de classificação. Portanto, você pode consultar essa tabela.)
 - Localizar respostas para um tópico de fórum postado nos últimos 15 dias.
 - Localizar respostas para um tópico de fórum postado em um intervalo de datas específico.
- Scan can a `ProductCatalog` para localizar livros cujo preço seja menor que zero.

Por motivos de desempenho, você deve usar uma operação de consulta em vez de uma operação de verificação. No entanto, em algumas ocasiões, talvez você precise verificar uma tabela. Suponha que tenha havido um erro de entrada de dados, e um dos preços de livros esteja definido como menor que 0. Este exemplo examina o `ProductCategory` para encontrar itens do livro (o `ProductCategory` é livro) com preços menores que 0.

Para obter instruções sobre como criar um exemplo funcional, consulte [Exemplos de código .NET \(p. 336\)](#).

Note

O exemplo a seguir não funciona com o .NET Core, pois ele não é compatível com métodos síncronos. Para obter mais informações, consulte [AWS APIs assíncronas da para .NET](#).

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
using System;  
using System.Collections.Generic;  
using System.Configuration;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.DataModel;  
using Amazon.DynamoDBv2.DocumentModel;  
using Amazon.Runtime;  
  
namespace com.amazonaws.codesamples  
{  
    class HighLevelQueryAndScan  
    {  
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();  
    }  
}
```

```
static void Main(string[] args)
{
    try
    {
        DynamoDBContext context = new DynamoDBContext(client);
        // Get an item.
        GetBook(context, 101);

        // Sample forum and thread to test queries.
        string forumName = "Amazon DynamoDB";
        string threadSubject = "DynamoDB Thread 1";
        // Sample queries.
        FindRepliesInLast15Days(context, forumName, threadSubject);
        FindRepliesPostedWithinTimePeriod(context, forumName, threadSubject);

        // Scan table.
        FindProductsPricedLessThanZero(context);
        Console.WriteLine("To continue, press Enter");
        Console.ReadLine();
    }
    catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message); }
    catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
    catch (Exception e) { Console.WriteLine(e.Message); }
}

private static void GetBook(DynamoDBContext context, int productId)
{
    Book bookItem = context.Load<Book>(productId);

    Console.WriteLine("\nGetBook: Printing result.....");
    Console.WriteLine("Title: {0} \n No.Of threads:{1} \n No. of messages: {2}",
                    bookItem.Title, bookItem.ISBN, bookItem.PageCount);
}

private static void FindRepliesInLast15Days(DynamoDBContext context,
                                             string forumName,
                                             string threadSubject)
{
    string replyId = forumName + "#" + threadSubject;
    DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
    IEnumerable<Reply> latestReplies =
        context.Query<Reply>(replyId, QueryOperator.GreaterThan, twoWeeksAgoDate);
    Console.WriteLine("\nFindRepliesInLast15Days: Printing result.....");
    foreach (Reply r in latestReplies)
        Console.WriteLine("{0}\t{1}\t{2}\t{3}", r.Id, r.PostedBy, r.Message,
r.ReplyDateTime);
}

private static void FindRepliesPostedWithinTimePeriod(DynamoDBContext context,
                                                       string forumName,
                                                       string threadSubject)
{
    string forumId = forumName + "#" + threadSubject;
    Console.WriteLine("\nFindRepliesPostedWithinTimePeriod: Printing result.....");

    DateTime startDate = DateTime.UtcNow - TimeSpan.FromDays(30);
    DateTime endDate = DateTime.UtcNow - TimeSpan.FromDays(1);

    IEnumerable<Reply> repliesInAPeriod = context.Query<Reply>(forumId,
                                                               QueryOperator.Between, startDate, endDate);
    foreach (Reply r in repliesInAPeriod)
        Console.WriteLine("{0}\t{1}\t{2}\t{3}", r.Id, r.PostedBy, r.Message,
r.ReplyDateTime);
}
```

```
private static void FindProductsPricedLessThanZero(DynamoDBContext context)
{
    int price = 0;
    IEnumerable<Book> itemsWithWrongPrice = context.Scan<Book>(
        new ScanCondition("Price", ScanOperator.LessThan, price),
        new ScanCondition("ProductCategory", ScanOperator.Equal, "Book")
    );
    Console.WriteLine("\nFindProductsPricedLessThanZero: Printing result.....");
    foreach (Book r in itemsWithWrongPrice)
        Console.WriteLine("{0}\t{1}\t{2}\t{3}", r.Id, r.Title, r.Price, r.ISBN);
}

[DynamoDBTable("Reply")]
public class Reply
{
    [DynamoDBHashKey] //Partition key
    public string Id
    {
        get; set;
    }

    [DynamoDBRangeKey] //Sort key
    public DateTime ReplyDateTime
    {
        get; set;
    }

    // Properties included implicitly.
    public string Message
    {
        get; set;
    }
    // Explicit property mapping with object persistence model attributes.
    [DynamoDBProperty("LastPostedBy")]
    public string PostedBy
    {
        get; set;
    }
    // Property to store version number for optimistic locking.
    [DynamoDBVersion]
    public int? Version
    {
        get; set;
    }
}

[DynamoDBTable("Thread")]
public class Thread
{
    // Partition key mapping.
    [DynamoDBHashKey] //Partition key
    public string ForumName
    {
        get; set;
    }
    [DynamoDBRangeKey] //Sort key
    public DateTime Subject
    {
        get; set;
    }
    // Implicit mapping.
    public string Message
    {
        get; set;
    }
}
```

```
public string LastPostedBy
{
    get; set;
}
public int Views
{
    get; set;
}
public int Replies
{
    get; set;
}
public bool Answered
{
    get; set;
}
public DateTime LastPostedDateTime
{
    get; set;
}
// Explicit mapping (property and table attribute names are different).
[DynamoDBProperty("Tags")]
public List<string> KeywordTags
{
    get; set;
}
// Property to store version number for optimistic locking.
[DynamoDBVersion]
public int? Version
{
    get; set;
}

[DynamoDBTable("Forum")]
public class Forum
{
    [DynamoDBHashKey]
    public string Name
    {
        get; set;
    }
    // All the following properties are explicitly mapped
    // to show how to provide mapping.
    [DynamoDBProperty]
    public int Threads
    {
        get; set;
    }
    [DynamoDBProperty]
    public int Views
    {
        get; set;
    }
    [DynamoDBProperty]
    public string LastPostBy
    {
        get; set;
    }
    [DynamoDBProperty]
    public DateTime LastPostDateTime
    {
        get; set;
    }
    [DynamoDBProperty]
    public int Messages
```

```
        {
            get; set;
        }

    [DynamoDBTable("ProductCatalog")]
    public class Book
    {
        [DynamoDBHashKey] //Partition key
        public int Id
        {
            get; set;
        }
        public string Title
        {
            get; set;
        }
        public string ISBN
        {
            get; set;
        }
        public int Price
        {
            get; set;
        }
        public string PageCount
        {
            get; set;
        }
        public string ProductCategory
        {
            get; set;
        }
        public bool InPublication
        {
            get; set;
        }
    }
}
```

Executar os exemplos de código neste Guia do desenvolvedor

OAWSOs SDKs fornecem um amplo suporte ao Amazon DynamoDB nas seguintes linguagens:

- Java
- JavaScript no navegador
- .NET
- Node.js
- PHP
- Python
- Ruby
- C++
- Go
- Android
- iOS

Para começar a trabalhar rapidamente com essas linguagens, consulte [Conceitos básicos do DynamoDB e do AWSSDKs da \(p. 81\)](#).

Os exemplos de código neste guia do desenvolvedor fornecem uma cobertura mais detalhada das operações do DynamoDB, usando as seguintes linguagens de programação:

- [Exemplos de código Java \(p. 334\)](#)
- [Exemplos de código .NET \(p. 336\)](#)

Antes de começar com este exercício, você precisa criar um AWS, obtenha sua chave de acesso e chave secreta e configure o AWS Command Line Interface(AWS CLI) no computador. Para mais informações, consulte [Configuração do DynamoDB \(serviço Web\) \(p. 57\)](#).

Note

Se estiver usando a versão para download do DynamoDB, você precisará usar o AWS CLI para criar as tabelas e dados de amostra. Também precisará especificar o parâmetro `--endpoint-url` com cada comando da AWS CLI. Para mais informações, consulte [Definição do endpoint local \(p. 56\)](#).

Criar tabelas e carregar dados para exemplos de código no DynamoDB

Neste tutorial, use a AWS Management Console para criar tabelas no Amazon DynamoDB. Depois, carrega dados nessas tabelas usando a AWS Command Line Interface (AWS CLI).

Essas tabelas e seus dados são usados como exemplos ao longo deste guia do desenvolvedor.

Note

Se você é desenvolvedor de aplicativos, recomendamos também a leitura do [Conceitos básicos do DynamoDB e do AWSSDKs da \(p. 81\)](#), que usa a versão para download do DynamoDB. Com ele, é possível saber mais sobre a API de baixo nível do DynamoDB sem precisar pagar taxas de throughput, armazenamento ou transferência de dados.

Tópicos

- [Etapa 1: Criar tabelas de exemplo \(p. 329\)](#)
- [Etapa 2: Carregar dados em tabelas \(p. 332\)](#)
- [Etapa 3: Consultar os dados \(p. 333\)](#)
- [Etapa 4: Limpar \(opcional\) \(p. 334\)](#)
- [Summary \(p. 334\)](#)

Etapa 1: Criar tabelas de exemplo

Nesta seção, use a AWS Management Console para criar tabelas no Amazon DynamoDB para dois casos de uso simples.

Caso de uso 1: Catálogo de produtos

Vamos supor que você queira armazenar informações sobre produtos no DynamoDB. Cada produto tem seus próprios atributos distintos e, portanto, você precisa armazenar informações diferentes sobre cada um deles.

É possível criar uma tabela `ProductCatalog`, na qual cada item é identificado exclusivamente por um único atributo numérico: `Id`.

Nome da tabela	Chave primária
ProductCatalog	Chave de partição: Id (telefone)

Caso de uso 2: Aplicativo de fórum

Vamos supor que você queira criar um aplicativo para quadros de mensagens ou fóruns de discussão. AWS Fóruns de discussão da Representa um exemplo desse tipo de aplicativo. Os clientes podem se envolver com a comunidade de desenvolvedores, fazer perguntas ou responder a publicações de outros clientes. Cada AWSO serviço tem um fórum dedicado. Qualquer um pode iniciar um novo tópico de discussão, postando uma mensagem em um fórum. Cada tópico pode receber qualquer número de respostas.

É possível modelar esse aplicativo criando três tabelas: Forum, Thread e Reply.

Nome da tabela	Chave primária
Forum	Chave de partição: Name (String)
Thread	Chave de partição: ForumName (String) Chave de classificação: Subject (String)
Reply	Chave de partição: Id (String) Chave de classificação: ReplyDateTime (String)

O ReplyA tabela tem um índice secundário global chamado PostedBy-Message-Index. Esse índice facilita as consultas em dois atributos que não são de chave da tabela Reply.

Nome do índice	Chave primária
PostedBy-Message-Index	Chave de partição: PostedBy (String) Chave de classificação: Message (String)

Criar a tabela ProductCatalog

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. Selecione Create Table (Criar tabela).
3. Na tela Create DynamoDB table (Criar tabela do DynamoDB), faça o seguinte:
 - a. Na caixa de nome Table (Tabela), digite ProductCatalog.
 - b. Em Primary key (Chave primária), na caixa Partition key (Chave de partição), digite Id. Defina o tipo de dados como Number (Número).
4. Quando estiver satisfeito com as configurações, escolha Create.

Criar a tabela Forum

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. Selecione Create Table (Criar tabela).

3. Na tela Create DynamoDB table (Criar tabela do DynamoDB), faça o seguinte:
 - a. Na caixa Table name (Nome da tabela), digite **Forum**.
 - b. Em Primary key (Chave primária), na caixa Partition key (Chave de partição), digite **Name**. Defina o tipo de dados como String.
4. Quando estiver satisfeito com as configurações, escolha Create.

Criar a tabela Thread

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. Selecione Create Table (Criar tabela).
3. Na tela Create DynamoDB table (Criar tabela do DynamoDB), faça o seguinte:
 - a. Na caixa Table name (Nome da tabela), digite **Thread**.
 - b. Em Primary key (Chave primária), faça o seguinte:
 - Na caixa Partition key (Chave de partição), digite **ForumName**. Defina o tipo de dados como String.
 - Escolha Add sort key (Adicionar chave de classificação).
 - Na caixa Sort key (Chave de classificação), digite **Subject**. Defina o tipo de dados como String.
4. Quando estiver satisfeito com as configurações, escolha Create.

Criar a tabela Reply

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. Selecione Create Table (Criar tabela).
3. Na tela Create DynamoDB table (Criar tabela do DynamoDB), faça o seguinte:
 - a. Na caixa Table name (Nome da tabela), digite **Reply**.
 - b. Em Primary key (Chave primária), faça o seguinte:
 - Na caixa Partition key (Chave de partição), digite **Id**. Defina o tipo de dados como String.
 - Escolha Add sort key (Adicionar chave de classificação).
 - Na caixa Sort key (Chave de classificação), digite **ReplyDateTime**. Defina o tipo de dados como String.
 - c. Na seção Table settings (Configurações da tabela), desmarque Use default settings (Usar configurações padrão).
 - d. Na seção Secondary indexes (Índices secundários), escolha Add index (Adicionar índice).
 - e. Na janela Add index (Adicionar índice), faça o seguinte:
 - Em Primary key (Chave primária), faça o seguinte:
 - Na caixa Partition key (Chave de partição), digite **PostedBy**. Defina o tipo de dados como String.
 - Escolha Add sort key (Adicionar chave de classificação).
 - Na caixa Sort key (Chave de classificação), digite **Message**. Defina o tipo de dados como String.
 - Na caixa Index name (Nome do índice), digite **PostedBy-Message-Index**.
 - Defina Projected attributes (Atributos projetados) como All (Tudo).
 - Escolha Add index (Adicionar índice).
4. Quando estiver satisfeito com as configurações, escolha Create.

Etapa 2: Carregar dados em tabelas

Nesta etapa, carregue dados de exemplo nas tabelas que criou. É possível inserir os dados manualmente no console do Amazon DynamoDB. Entretanto, para economizar tempo, use a AWS Command Line Interface (AWS CLI).

Note

Se ainda não tiver configurado a AWS CLI, consulte [Como usar AWS CLI \(p. 62\)](#) para obter instruções.

Você fará download de um arquivamento .zip que contém arquivos JSON com dados de exemplo para cada tabela. Para cada arquivo, use a AWS CLI para carregar os dados no DynamoDB. Cada carregamento de dados bem-sucedido produzi a seguinte saída.

```
{  
    "UnprocessedItems": {}  
}
```

Fazer download do arquivamento de arquivo de dados de exemplo

1. Faça download do arquivamento de dados de exemplo (`sampleddata.zip`) usando este link:
 - [sampledata.zip](#)
2. Extraia os arquivo de dados .json do arquivamento.
3. Copie o arquivo de dados .json para o seu diretório atual.

Carregar os dados de exemplo nas tabelas do DynamoDB

1. Para carregar a tabela `ProductCatalog` com dados, insira o comando a seguir.

```
aws dynamodb batch-write-item --request-items file://ProductCatalog.json
```

2. Para carregar a tabela `Forum` com dados, insira o comando a seguir.

```
aws dynamodb batch-write-item --request-items file://Forum.json
```

3. Para carregar a tabela `Thread` com dados, insira o comando a seguir.

```
aws dynamodb batch-write-item --request-items file://Thread.json
```

4. Para carregar a tabela `Reply` com dados, insira o comando a seguir.

```
aws dynamodb batch-write-item --request-items file://Reply.json
```

Verificar o carregamento dos dados

Você pode usar o AWS Management Console para verificar os dados carregados nas tabelas.

Para verificar os dados usando o AWS Management Console

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, selecione Tables (Tabelas).
3. Na lista de tabelas, escolha `ProductCatalog`.
4. Escolha a guia Items (Itens) para visualizar os dados carregados na tabela.
5. Para exibir um item na tabela, escolha seu Id. (Se quiser, você também pode editar o item.)
6. Para retornar à lista de tabelas, escolha Cancel (Cancelar).

Repita esse procedimento para cada uma das outras tabelas que você criou:

- Forum
- Thread
- Reply

Etapa 3: Consultar os dados

Nesta etapa, tente algumas consultas simples nas tabelas que criou, usando o console do Amazon DynamoDB.

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, selecione Tables (Tabelas).
3. Na lista de tabelas, escolha Reply.
4. Escolha a guia Items (Itens) para visualizar os dados carregados na tabela.
5. Escolha o link de filtragem de dados localizado logo abaixo do botão Create item (Criar item).

The screenshot shows the AWS DynamoDB console interface for the 'Reply' table. At the top, there's a navigation bar with tabs: Overview, Metrics, Alarms, Capacity, Items (which is highlighted with an orange border), and Indexes. Below the tabs, there are two main buttons: 'Create item' (blue) and 'Actions ▾'. Underneath these, a red oval highlights a link labeled 'Scan: [Table] Reply: Id, ReplyDateTime ▾'. This link is part of a larger panel for filtering data.

Quando você escolhe o link, o console revela um painel de filtragem de dados.

This screenshot shows the 'Scan' operation selected in the filter panel. The 'Table' dropdown is set to '[Table] Reply: Id, ReplyDateTime'. There are buttons for 'Start' and 'Begins a new search'.

6. No painel de filtragem de dados, faça o seguinte:
 - a. Altere a operação de Scan (Verificar) para Query (Consultar).
 - b. Em Partition key (Chave de partição), digite o valor **Amazon DynamoDB#DynamoDB Thread 1**.
 - c. Escolha Iniciar. Somente os itens que corresponderem a seus critérios de consulta serão retornados da tabela Reply.
7. O ReplyA tabela tem um índice secundário global no PostedByMessageAtributos. É possível usar o painel de filtragem de dados para consultar o índice. Faça o seguinte:
 - a. Altere a origem da consulta do seguinte:

[Table] Reply: Id, ReplyDateTime

Para o seguinte:

[Index] PostedBy-Message-Index: PostedBy, Message

- b. Em Partition key (Chave de partição), digite o valor **User A**.

- c. Escolha Iniciar. Somente os itens que corresponderem a seus critérios de consulta serão retornados de `PostedBy-Message-Index`.

Aproveite um tempo para explorar suas outras tabelas usando o console do DynamoDB:

- ProductCatalog
- Forum
- Thread

Etapa 4: Limpar (opcional)

Essas tabelas de exemplo são usadas em todo o Guia do desenvolvedor do Amazon DynamoDB Para ajudar a ilustrar operações de tabelas e itens usando a API de baixo nível do e vários AWSSDKs. Se você pretende ler o resto deste guia, talvez as tabelas sejam úteis para referência. No entanto, se você não quiser manter essas tabelas, será necessário excluí-las para evitar cobranças por recursos desnecessários.

Como excluir as tabelas de exemplo

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, selecione Tables (Tabelas).
3. Na lista de tabelas, escolha ProductCatalog.
4. Selecione Delete Table (Excluir tabela). Será solicitado que você confirme sua seleção.

Reita esse procedimento para cada uma das outras tabelas que você criou:

- Forum
- Thread
- Responder

Summary

Neste exercício, você usou o console do DynamoDB para criar várias tabelas no DynamoDB. Você usou a AWS CLI para carregar dados nas tabelas e executou algumas operações básicas nos dados usando o console.

O console e a AWS CLI são úteis para começar a usar rapidamente. No entanto, você provavelmente quer saber mais sobre como o DynamoDB funciona e como escrever programas aplicativos com o DynamoDB. O restante deste guia do desenvolvedor aborda esses tópicos.

Exemplos de código Java

Tópicos

- [Java: Configurar o AWS Credenciais \(p. 335\)](#)
- [Java: Configurar o AWS Região e endpoints \(p. 336\)](#)

Este Guia do Desenvolvedor contém trechos de código Java e programas prontos para serem executados. Você encontrará esses exemplos de código nas seguintes seções:

- [Trabalho com itens e atributos \(p. 404\)](#)

- Como trabalhar com tabelas e dados no DynamoDB (p. 339)
- Como trabalhar com consultas no DynamoDB (p. 490)
- Como trabalhar com verificações no DynamoDB (p. 508)
- Como melhorar o acesso a dados com índices secundários (p. 559)
- Java: DynamoDBMapper (p. 228)
- Captura de dados de alteração para DynamoDB Streams (p. 651)

É possível começar rapidamente usando o Eclipse com o [AWS Toolkit for Eclipse](#). Além de um IDE completo, você também terá o AWS SDK for Java com atualizações automáticas e modelos pré-configurados para criar aplicativos.

Como executar os exemplos de código Java (usando o Eclipse)

1. Baixe e instale o IDE do [Eclipse](#).
2. Faça download e instale o [AWS Toolkit for Eclipse](#).
3. Inicie o Eclipse e, no menu Eclipse, escolha File (Arquivo), New (Novo) e Other (Outro).
4. Dentro de Selecionar um assistente, escolha AWS, escolha AWS Projeto Java e, depois, escolha Próximo.
5. Dentro de Criar um AWS Java, faça o seguinte:
 - a. Em Nome do projeto, digite um nome para o projeto.
 - b. Em Selecionar conta (Selecionar conta), escolha o perfil de suas credenciais na lista.

Se esta for a primeira vez que você está usando o [AWS Toolkit for Eclipse](#),
escolha Configure AWS Contas Para configurar a AWS Credenciais da .
6. Escolha Finish (Concluir) para criar o projeto.
7. No menu do Eclipse, escolha File (Arquivo), New (Novo) e, em seguida, Class (Classe).
8. Em Java Class (Classe Java), digite um nome para a sua classe em Name (Nome) (use o mesmo nome que o exemplo de código que você deseja executar) e escolha Finish (Concluir) para criar a classe.
9. Copie o exemplo de código da página de documentação no editor do Eclipse.
10. Para executar o código, escolha Run (Executar) no menu do Eclipse.

O SDK for Java fornece clientes thread-safe para trabalhar com o DynamoDB. De acordo com as melhores práticas, seus aplicativos devem criar um único cliente e reutilizá-lo entre os threads.

Para mais informações, consulte o [AWS SDK for Java](#).

Note

Os exemplos de código neste guia são destinados para uso com a versão mais recente do AWS SDK for Java.

Se você estiver usando o AWS Toolkit for Eclipse, você pode configurar atualizações automáticas para o SDK for Java. Para fazer isso no Eclipse, vá até o Preferences (Preferências) e escolha AWS Toolkit , AWS SDK for Java, Baixar novos SDKs automaticamente.

Java: Configurar o AWS Credenciais

O SDK for Java exige que você forneça AWS para o seu aplicativo em tempo de execução. Os exemplos de código neste guia pressupõem que você esteja usando um AWS, conforme descrito em [Configurar seu AWS Credenciais](#) no AWS SDK for Java Guia do desenvolvedor.

Veja a seguir um exemplo de um AWS arquivo de credenciais chamado ~ / .aws / credentials, onde o caractere til (~) representa o diretório inicial.

```
[default]
aws_access_key_id = AWS access key ID goes here
aws_secret_access_key = Secret key goes here
```

Java: Configurar oAWSRegião e endpoints

Por padrão, os exemplos de código acessam o DynamoDB na região Oeste dos EUA (Oregon). É possível alterar a região ao modificar as propriedades AmazonDynamoDB.

O exemplo de código a seguir instancia uma nova AmazonDynamoDB.

```
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.regions.Regions;
...
// This client will default to US West (Oregon)
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
.withRegion(Regions.US_WEST_2)
.build();
```

Você pode usar owithRegionPara executar seu código no DynamoDB em qualquer região em que ele esteja disponível. Para obter uma lista completa, consulte[AWSRegiões e endpoints](#) donoReferência geral da Amazon Web Services.

Se quiser executar os exemplos de código usando o DynamoDB localmente no seu computador, defina o endpoint da seguinte maneira.

AWSSDK V1

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().withEndpointConfiguration(
new AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))
.build();
```

AWSSDK V2

```
DynamoDbClient client = DynamoDbClient.builder()
.endpointOverride(URI.create("http://localhost:8000"))
// The region is meaningless for local DynamoDb but required for client builder
validation
.region(Region.US_EAST_1)
.credentialsProvider(StaticCredentialsProvider.create(
AwsBasicCredentials.create("dummy-key", "dummy-secret")))
.build();
```

Exemplos de código .NET

Tópicos

- [.NET: Configurar oAWSCredenciais \(p. 337\)](#)
- [.NET: Configurar oAWSRegião e endpoints \(p. 338\)](#)

Este guia contém trechos de código .NET e programas prontos para serem executados. Você encontrará esses exemplos de código nas seguintes seções:

- [Trabalho com itens e atributos \(p. 404\)](#)
- [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#)

- Como trabalhar com consultas no DynamoDB (p. 490)
- Como trabalhar com verificações no DynamoDB (p. 508)
- Como melhorar o acesso a dados com índices secundários (p. 559)
- .NET: Modelo de documento (p. 276)
- .NET: Modelo de persistência de objeto (p. 299)
- Captura de dados de alteração para DynamoDB Streams (p. 651)

É possível começar rapidamente usando o AWS SDK for .NETCom o Toolkit for Visual Studio.

Como executar os exemplos de código .NET (usando o Visual Studio)

1. Baixe e instale o [Microsoft Visual Studio](#).
2. Faça download e instale o[Toolkit for Visual Studio](#).
3. Inicie o Visual Studio. Escolha File (Arquivo), New (Novo), Project (Projeto).
4. DentroNovo projeto do, escolhaAWSProjeto vazioe, depois, escolhaOK.
5. DentroAWSCredenciais de acesso, escolhaUsar perfil existente, escolha seu perfil de credenciais na lista e depois escolhaOK.

Se esta for sua primeira vez usando o Toolkit for Visual Studio, escolhaUsar um novo perfilPara configurar aAWSCredenciais da .

6. No projeto do Visual Studio, escolha a guia para o código-fonte do seu programa (`Program.cs`). Copie o exemplo de código da página da documentação no editor do Visual Studio, substituindo qualquer outro código que você visualize no editor.
7. Se você vir mensagens de erro do formulárioO nome do tipo ou do espaço de nomes... não pôde ser encontrado, você precisa instalar oAWSMontagem do SDK para o DynamoDB da seguinte forma:
 - a. No Solution Explorer, abra o menu de contexto (clique com o botão direito do mouse) do seu projeto e escolha Manage NuGet Packages (Gerenciar pacotes do NuGet).
 - b. No Gerenciador de pacotes NuGet, escolha Browse (Procurar).
 - c. Na caixa de pesquisa, digite **AWSSDK.DynamoDBv2** e aguarde a conclusão da pesquisa.
 - d. Escolha AWSSDK.DynamoDBv2 e Install (Instalar).
 - e. Quando a instalação estiver concluída, escolha a guia Program.cs para retornar ao seu programa.
8. Para executar o código, escolha Start (Iniciar) na barra de ferramentas do Visual Studio.

OAWS SDK for .NETO fornece clientes thread-safe para trabalhar com o DynamoDB. De acordo com as melhores práticas, seus aplicativos devem criar um único cliente e reutilizá-lo entre os threads.

Para obter mais informações, consulte [AWS SDK para .NET](#).

Note

Os exemplos de código neste guia são destinados para uso com a versão mais recente do AWS SDK for .NET.

.NET: Configurar oAWSCredenciais

OAWS SDK for .NETrequer que você forneçaAWSpara o seu aplicativo em tempo de execução. Os exemplos de código neste guia pressupõem que você esteja usando o SDK Store para gerenciar seuAWS, conforme descrito em[Usar o SDK Store](#)noAWS SDK for .NETGuia do desenvolvedor.

O Toolkit for Visual Studio oferece suporte a vários conjuntos de credenciais de qualquer número de contas. Cada conjunto é chamado de perfil. O Visual Studio adiciona entradas ao`App.config`para que o aplicativo possa encontrar oAWSCredenciais no tempo de execução.

O exemplo a seguir mostra o padrão do `App.config` que é gerado quando você cria um novo projeto usando o Toolkit for Visual Studio.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <appSettings>
        <add key="AWSProfileName" value="default"/>
        <add key="AWSRegion" value="us-west-2" />
    </appSettings>
</configuration>
```

Em tempo de execução, o programa usa a função `default` do conjunto de AWS Credenciais, conforme especificado pelo `AWSProfileName` Entrada. As credenciais em si são mantidas no SDK Store em formato criptografado. O Toolkit for Visual Studio fornece uma interface gráfica do usuário para gerenciar suas credenciais no Visual Studio. Para obter mais informações, consulte [Especificação de credenciais no AWS Toolkit for Visual Studio](#).

Note

Por padrão, os exemplos de código acessam o DynamoDB na região Oeste dos EUA (Oregon). É possível alterar a região ao modificar a entrada `AWSRegion` no arquivo `App.config`. Você pode definir o `AWSRegion` para qualquer região em que o DynamoDB esteja disponível. Para obter uma lista completa, consulte [AWS Regiões e endpoints](#) na Referência geral da Amazon Web Services.

.NET: Configurar o AWS Region e endpoints

Por padrão, os exemplos de código acessam o DynamoDB na região Oeste dos EUA (Oregon). É possível alterar a região ao modificar a entrada `AWSRegion` no arquivo `App.config`. É possível alterar a região ao modificar as propriedades `AmazonDynamoDBClient`.

O exemplo de código a seguir instancia uma nova `AmazonDynamoDBClient`. O cliente é modificado de forma que o código seja executado no DynamoDB em uma região diferente.

```
AmazonDynamoDBConfig clientConfig = new AmazonDynamoDBConfig();
// This client will access the US East 1 region.
clientConfig.RegionEndpoint = RegionEndpoint.USEast1;
AmazonDynamoDBClient client = new AmazonDynamoDBClient(clientConfig);
```

Para obter uma lista completa de regiões, consulte [AWS Regiões e endpoints](#) na Referência geral da Amazon Web Services.

Se quiser executar os exemplos de código usando o DynamoDB localmente no seu computador, defina o endpoint da seguinte maneira.

```
AmazonDynamoDBConfig clientConfig = new AmazonDynamoDBConfig();
// Set the endpoint URL
clientConfig.ServiceURL = "http://localhost:8000";
AmazonDynamoDBClient client = new AmazonDynamoDBClient(clientConfig);
```

Trabalhar com tabelas, itens, consultas, varreduras e índices

Esta seção fornece detalhes sobre como trabalhar com tabelas, itens, consultas e muito mais no Amazon DynamoDB.

Tópicos

- [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#)
- [Trabalho com itens e atributos \(p. 404\)](#)
- [Como trabalhar com consultas no DynamoDB \(p. 490\)](#)
- [Como trabalhar com verificações no DynamoDB \(p. 508\)](#)
- [PartiQL - Uma linguagem de consulta compatível com SQL para o Amazon DynamoDB \(p. 529\)](#)
- [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#)
- [Captura de dados de alteração com o Amazon DynamoDB \(p. 641\)](#)
- [Gerenciar fluxos de trabalho complexos com transações do DynamoDB \(p. 680\)](#)
- [Backups do DynamoDB \(p. 691\)](#)

Como trabalhar com tabelas e dados no DynamoDB

Esta seção descreve como usar o AWS Command Line Interface(AWS CLI) e o AWSSDKs para criar, atualizar e excluir tabelas no Amazon DynamoDB.

Note

Você também pode executar essas mesmas tarefas usando o AWS Management Console. Para mais informações, consulte [Usar o console \(p. 60\)](#).

Esta seção também fornece mais informações sobre a capacidade de throughput, usando o Auto Scaling do DynamoDB ou configurando manualmente o throughput provisionado.

Tópicos

- [Operações básicas nas tabelas do DynamoDB \(p. 340\)](#)
- [Considerações ao mudar o modo de capacidade de leitura/gravação \(p. 344\)](#)
- [Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB \(p. 345\)](#)
- [Tamanhos e formatos de item do DynamoDB \(p. 349\)](#)
- [Como gerenciar a capacidade de throughput automaticamente com o Auto Scaling do DynamoDB \(p. 350\)](#)
- [Tabelas globais Replicação em várias regiões com o DynamoDB \(p. 365\)](#)
- [Adicionar tags e rótulos a recursos \(p. 388\)](#)
- [Como trabalhar com tabelas do DynamoDB em Java \(p. 392\)](#)
- [Como trabalhar com tabelas do DynamoDB no .NET \(p. 397\)](#)

Operações básicas nas tabelas do DynamoDB

Semelhante a outros sistemas de banco de dados, o Amazon DynamoDB armazena dados em tabelas. É possível gerenciar suas tabelas usando algumas operações básicas.

Tópicos

- [Criação de uma tabela \(p. 340\)](#)
- [Descrição de uma tabela \(p. 342\)](#)
- [Atualização de uma tabela \(p. 343\)](#)
- [Exclusão de uma tabela \(p. 343\)](#)
- [Nomes de tabela de listagem \(p. 344\)](#)
- [Descrição das cotas de taxa de transferência provisionada \(p. 344\)](#)

Criação de uma tabela

Usar a `CreateTable` para criar uma tabela no Amazon DynamoDB. Para criar a tabela, você deve fornecer as seguintes informações:

- Nome da tabela. O nome deve estar de acordo com as regras de nomeação do DynamoDB e deve ser exclusivo para o AWS conta e região. Por exemplo, você poderia criar um `PeopleTabela` no Leste dos EUA (Norte da Virgínia) e outro `PeopleTabela` na Europa (Irlanda). No entanto, essas duas tabelas devem ser inteiramente diferentes uma da outra. Para mais informações, consulte [Regras de nomeação e tipos de dados \(p. 13\)](#).
- Chave primária. A chave primária pode consistir em um atributo (chave de partição) ou de dois atributos (chave de partição e chave de classificação). Você precisa fornecer os nomes de atributos, os tipos de dados e a função de cada um: `HASH` (para uma chave de partição) e `RANGE` (para uma chave de classificação). Para mais informações, consulte [Chave primária \(p. 6\)](#).
- Configurações de taxa de rendimento (para tabelas provisionadas). Se estiver usando o modo provisionado, você deve especificar as configurações de throughput de leitura e gravação inicial da tabela. Você pode modificar essas configurações mais tarde ou habilitar o Auto Scaling do DynamoDB para gerenciar as configurações para você. Para mais informações, consulte [.](#)

Exemplo 1: Criar uma tabela provisionada

O exemplo da AWS CLI a seguir mostra como criar uma tabela (`Music`). A chave primária consiste em `Artist` (chave de partição) e `SongTitle` (chave de classificação), cada uma delas tem um tipo de dados de `String`. O throughput máximo da tabela é 10 unidades de capacidade de leitura e 5 unidades de capacidade de gravação.

```
aws dynamodb create-table \
    --table-name Music \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput \
        ReadCapacityUnits=10,WriteCapacityUnits=5
```

A operação `CreateTable` retorna metadados para a tabela, conforme mostrado a seguir.

```
{
```

```
"TableDescription": {  
    "TableArn": "arn:aws:dynamodb:us-east-1:123456789012:table/Music",  
    "AttributeDefinitions": [  
        {  
            "AttributeName": "Artist",  
            "AttributeType": "S"  
        },  
        {  
            "AttributeName": "SongTitle",  
            "AttributeType": "S"  
        }  
    ],  
    "ProvisionedThroughput": {  
        "NumberOfDecreasesToday": 0,  
        "WriteCapacityUnits": 5,  
        "ReadCapacityUnits": 10  
    },  
    "TableSizeBytes": 0,  
    "TableName": "Music",  
    "TableStatus": "CREATING",  
    "TableId": "12345678-0123-4567-a123-abcdefghijkl",  
    "KeySchema": [  
        {  
            "KeyType": "HASH",  
            "AttributeName": "Artist"  
        },  
        {  
            "KeyType": "RANGE",  
            "AttributeName": "SongTitle"  
        }  
    ],  
    "ItemCount": 0,  
    "CreationDateTime": 1542397215.37  
}
```

O elemento `TableStatus` indica o estado atual da tabela (`CREATING`). Pode demorar um pouco para criar a tabela, dependendo dos valores que você especificar para `ReadCapacityUnits` e `WriteCapacityUnits`. Valores maiores exigem que o DynamoDB aloque mais recursos para a tabela.

Exemplo 2: Criar uma tabela sob demanda

Para criar a mesma tabela `Music` usando modo sob demanda.

```
aws dynamodb create-table \  
    --table-name Music \  
    --attribute-definitions \  
        AttributeName=Artist,AttributeType=S \  
        AttributeName=SongTitle,AttributeType=S \  
    --key-schema \  
        AttributeName=Artist,KeyType=HASH \  
        AttributeName=SongTitle,KeyType=RANGE \  
    --billing-mode=PAY_PER_REQUEST
```

A operação `CreateTable` retorna metadados para a tabela, conforme mostrado a seguir.

```
{  
    "TableDescription": {  
        "TableArn": "arn:aws:dynamodb:us-east-1:123456789012:table/Music",  
        "AttributeDefinitions": [  
            {  
                "AttributeName": "Artist",  
                "AttributeType": "S"
```

```
        },
        {
            "AttributeName": "SongTitle",
            "AttributeType": "S"
        }
    ],
    "ProvisionedThroughput": {
        "NumberOfDecreasesToday": 0,
        "WriteCapacityUnits": 0,
        "ReadCapacityUnits": 0
    },
    "TableSizeBytes": 0,
    "TableName": "Music",
    "BillingModeSummary": {
        "BillingMode": "PAY_PER_REQUEST"
    },
    "TableStatus": "CREATING",
    "TableId": "12345678-0123-4567-a123-abcdefghijkl",
    "KeySchema": [
        {
            "KeyType": "HASH",
            "AttributeName": "Artist"
        },
        {
            "KeyType": "RANGE",
            "AttributeName": "SongTitle"
        }
    ],
    "ItemCount": 0,
    "CreationDateTime": 1542397468.348
}
```

Important

Ao chamar `DescribeTable` em uma tabela sob demanda, as unidades de capacidade de leitura e unidades de capacidade de gravação são definidas como 0.

Descrição de uma tabela

Para visualizar detalhes sobre uma tabela, use a operação `DescribeTable`. Você deve fornecer o nome da tabela. A saída de `DescribeTable` está no mesmo formato de `CreateTable`. Ela inclui o carimbo de data e hora em que a tabela foi criada, o esquema de chaves, as configurações de throughput provisionado, o tamanho estimado e os índices secundários que estão presentes.

Important

Ao chamar `DescribeTable` em uma tabela sob demanda, as unidades de capacidade de leitura e unidades de capacidade de gravação são definidas como 0.

Example

```
aws dynamodb describe-table --table-name Music
```

A tabela estará pronta para uso quando `TableStatus` tiver sido alterado de `CREATING` para `ACTIVE`.

Note

Se você emitir um `DescribeTable` imediatamente após um `CreateTable`, o DynamoDB pode retornar um erro (`ResourceNotFoundException`). Isso ocorre porque `DescribeTable` usa uma consulta eventualmente consistente e os metadados da sua tabela podem não estar

disponíveis nesse momento. Aguarde alguns segundos e, em seguida, tente a solicitação `DescribeTable` novamente.

Para fins de faturamento, os custos de armazenamento do DynamoDB incluem uma sobrecarga por item de 100 bytes. (Para obter mais informações, acesse[Definição de preços do DynamoDB](#).) Esses 100 bytes extras por item não são usados nos cálculos da unidade de capacidade ou pelo[DescribeTable](#)opération.

Atualização de uma tabela

A operação `UpdateTable` permite que você execute uma das seguintes ações:

- Modificar as configurações de taxa de rendimento provisionada de uma tabela (para tabelas de modo provisionadas).
- Alterar o modo de capacidade de leitura/gravação de tabela.
- Manipular índices secundários globais na tabela (consulte [Como usar índices secundários globais no DynamoDB \(p. 562\)](#)).
- Habilite ou desabilite os DynamoDB Streams na tabela (consulte[Captura de dados de alteração para DynamoDB Streams \(p. 651\)](#)).

Example

O exemplo de AWS CLI a seguir mostra como modificar as configurações de taxa de transferência provisionada de uma tabela.

```
aws dynamodb update-table --table-name Music \
    --provisioned-throughput ReadCapacityUnits=20,WriteCapacityUnits=10
```

Note

Quando você emite uma solicitação `UpdateTable`, o status da tabela muda de `AVAILABLE` para `UPDATING`. A tabela permanecerá totalmente disponível para uso enquanto estiver `UPDATING`.

Quando esse processo for concluído, o status da tabela mudará de `UPDATING` para `AVAILABLE`.

Example

O exemplo de AWS CLI a seguir mostra como modificar o modo de capacidade de leitura/gravação de uma tabela para o modo sob demanda.

```
aws dynamodb update-table --table-name Music \
    --billing-mode PAY_PER_REQUEST
```

Exclusão de uma tabela

Você pode remover uma tabela não utilizada com a operação `DeleteTable`. Excluir uma tabela é uma operação irrecuperável.

Example

O exemplo de AWS CLI a seguir mostra como excluir uma tabela.

```
aws dynamodb delete-table --table-name Music
```

Quando você emite uma solicitação `DeleteTable`, o status da tabela muda de `ACTIVE` para `DELETING`. A exclusão da tabela pode demorar um pouco, dependendo dos recursos que ela usa (como os dados armazenados e os streams ou índices da tabela).

Quando a `DELETE TABLE` Conclua a operação, a tabela não existe mais no DynamoDB.

Nomes de tabela de listagem

O `LIST TABLES` A operação retorna os nomes das tabelas do DynamoDB para o AWS conta e região.

Example

Os seguintes exemplos de AWS CLI O exemplo mostra como listar os nomes de tabelas do DynamoDB.

```
aws dynamodb list-tables
```

Descrição das cotas de taxa de transferência provisionada

O `DESCRIBE LIMITS` Retorna as cotas de capacidade de leitura e gravação atuais do AWS conta e região.

Example

O exemplo de AWS CLI a seguir mostra como descrever as cotas atuais de taxa de transferência provisionada.

```
aws dynamodb describe-limits
```

A saída mostra as cotas superiores das unidades de capacidade de leitura e gravação do atual AWS conta e região.

Para obter mais informações sobre essas cotas e como solicitar aumentos de cota, consulte [Cotas padrão de taxa de transferência \(p. 1057\)](#).

Considerações ao mudar o modo de capacidade de leitura/gravação

Você pode alternar entre os modos de capacidade de leitura/gravação uma vez a cada 24 horas. Considere o seguinte ao atualizar seu modo de capacidade de leitura/gravação no Amazon DynamoDB.

Gerenciamento da capacidade

Quando você atualiza uma tabela de um modo sob demanda provisionado, não é necessário especificar a taxa de transferência de leitura e gravação que você espera que seu aplicativo execute.

Considere o seguinte ao atualizar uma tabela de modo sob demanda para provisionado:

- Se estiver usando o AWS Management Console, o console estima os valores iniciais de capacidade provisionada com base na capacidade consumida de leitura e gravação da sua tabela e índices secundários globais nos últimos 30 minutos. Para sobrepor esses valores recomendados, escolha `Override recommended values`.
- Se você estiver usando o AWS CLI ou AWSSDK, escolha as configurações de capacidade provisionadas certas de sua tabela e índices secundários globais usando o Amazon CloudWatch para procurar seu consumo histórico (`ConsumedWriteCapacityUnits` e `ConsumedReadCapacityUnits`) para determinar as novas configurações de taxa de transferência.

Note

Se você estiver alternando uma tabela global para o modo provisionado, examine o consumo máximo entre todas as suas réplicas regionais para tabelas de base e índices secundários globais ao determinar as novas configurações de taxa de transferência.

Gerenciamento de Auto Scaling

Quando você atualiza uma tabela do modo provisionado para sob demanda:

- Se estiver usando o console, todas as suas configurações de Auto Scaling (se houver alguma) serão excluídas.
- Se você estiver usando o AWS CLI ou o AWSSDK, todas as suas configurações de Auto Scaling serão preservadas. Essas configurações podem ser aplicadas quando você atualiza sua tabela para o modo de cobrança provisionado novamente.

Quando você atualiza uma tabela do modo sob demanda para provisionado:

- Se estiver usando o console, o DynamoDB recomenda habilitar Auto Scaling com os seguintes padrões:
 - Utilização pretendida: 70%
 - Capacidade provisionada mínima: 5 unidades
 - Capacidade máxima provisionada: O máximo da região
- Se estiver usando a AWS CLI ou o SDK, suas configurações anteriores de Auto Scaling (se houver) serão preservadas.

Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB

Ao criar uma nova tabela provisionada no Amazon DynamoDB, especifique seu **Capacidade de throughput provisionada**. Essa é a quantidade de atividades de leitura e gravação que a tabela pode suportar. O DynamoDB usa essas informações para reservar recursos suficientes do sistema para atender aos seus requisitos de throughput.

Note

Você pode criar uma tabela de modo sob demanda em vez disso, para que não tenha que gerenciar quaisquer configurações de capacidade para servidores, armazenamento ou taxa de transferência. O DynamoDB acomoda instantaneamente o crescimento e a redução das cargas de trabalho para qualquer nível de tráfego previamente registrado. Se o nível de tráfego de uma carga de trabalho atingir um novo pico, o DynamoDB se adapta rapidamente para acomodar a carga de trabalho. Para mais informações, consulte [Modo sob demanda \(p. 18\)](#).

Opcionalmente, você pode permitir Auto Scaling do DynamoDB para gerenciar a capacidade de taxa de rendimento da tabela. No entanto, você ainda deve fornecer configurações iniciais para a capacidade de leitura e de gravação ao criar a tabela. O Auto Scaling do DynamoDB usa essas configurações iniciais como um ponto de partida e, em seguida, as ajusta dinamicamente em resposta às exigências do seu aplicativo. Para mais informações, consulte [Como gerenciar a capacidade de throughput automaticamente com o Auto Scaling do DynamoDB \(p. 350\)](#).

À medida que os dados do seu aplicativo e os requisitos de acesso mudam, talvez seja necessário ajustar as configurações de throughput da tabela. Se você estiver usando o Auto Scaling do DynamoDB, as configurações de throughput serão automaticamente ajustadas em resposta às cargas de trabalho reais. Também é possível usar a operação `UpdateTable` para ajustar manualmente a capacidade de throughput da tabela. Você pode optar por fazer isso se precisar fazer o carregamento em massa de dados de um armazenamento de dados existentes para sua nova tabela do DynamoDB. Você pode criar a tabela com uma configuração alta de throughput de gravação e, em seguida, reduzir essa configuração depois que o carregamento de dados em massa for concluído.

Você especifica os requisitos de taxa de transferência em termos de unidades de capacidade—a quantidade de dados que o seu aplicativo precisa ler ou gravar por segundo. Você pode modificar essas

configurações mais tarde, se necessário, ou permitir que o Auto Scaling do DynamoDB modifique-os automaticamente.

Tópicos

- [unidades de capacidade de leitura \(p. 346\)](#)
- [Unidades de capacidade de gravação \(p. 347\)](#)
- [Limitação de solicitações e capacidade de intermitência \(p. 348\)](#)
- [Limitação de solicitações e capacidade adaptável \(p. 348\)](#)
- [Escolha de configurações iniciais de throughput \(p. 348\)](#)
- [Modificação das configurações de throughput \(p. 349\)](#)

unidades de capacidade de leitura

A Unidade de capacidade de leitura representa uma leitura fortemente consistente por segundo, ou duas leituras eventualmente consistentes por segundo, para um item de até 4 KB de tamanho.

Note

Para saber mais sobre os modelos de consistência de leitura do DynamoDB, consulte [Consistência de leituras \(p. 17\)](#).

Por exemplo, suponha que você crie uma tabela com 10 unidades de capacidade de leitura provisionadas. Isso permite que você execute 10 leituras fortemente consistentes por segundo, ou 20 leituras eventualmente consistentes por segundo para itens de até 4 KB.

Ler um item maior que 4 KB consome mais unidades de capacidade de leitura. Por exemplo, uma leitura fortemente consistente de um item que tem 8 KB ($4\text{ KB} \times 2$) consome duas unidades de capacidade de leitura. Uma leitura eventualmente consistente nesse mesmo item consome apenas uma unidade de capacidade de leitura.

Os tamanhos de item para leituras são arredondados até o próximo múltiplo de 4 KB. Por exemplo, ler um item com 3.500 bytes consome a mesma taxa de rendimento que a leitura de um item de 4 KB.

Consumo de unidade de capacidade para leituras

Veja a seguir como as operações de leitura do DynamoDB consomem unidades de capacidade de leitura:

- **GetItem**- Lê um único item de uma tabela. Para determinar o número de unidades de capacidade que o `GetItem` consumirá, arredonde o tamanho do item até o próximo limite de 4 KB. Se você especificou uma leitura fortemente consistente, este é o número de unidades de capacidade necessárias. Para uma leitura eventualmente consistente (o padrão), divida esse número por dois.

Por exemplo, se você ler um item que tem 3,5 KB, o DynamoDB arredonda o tamanho do item para 4 KB. Se você ler um item de 10 KB, o DynamoDB arredonda o tamanho do item para 12 KB.

- **BatchGetItem**- Lê até 100 itens, de uma ou mais tabelas. O DynamoDB processa cada item no lote como um indivíduo `GetItem`. Então, o DynamoDB arredonda o tamanho de cada item para o próximo limite de 4 KB e, em seguida, calcula o tamanho total. O resultado não é necessariamente o mesmo que o tamanho total de todos os itens. Por exemplo, se `BatchGetItem` ler um item de 1,5 KB e outro de 6,5 KB, o DynamoDB calcula o tamanho como 12 KB ($4\text{ KB} + 8\text{ KB}$), não 8 KB ($1,5\text{ KB} + 6,5\text{ KB}$).
- **Query**- Lê vários itens que têm o mesmo valor de chave de partição. Todos os itens retornados são tratados como uma única operação de leitura, em que o DynamoDB calcula o tamanho total de todos os itens e, em seguida, arredonda para o próximo limite de 4 KB. Por exemplo, suponha que a consulta retorne 10 itens cujo tamanho combinado seja 40,8 KB. O DynamoDB arredonda o tamanho do item da operação para 44 KB. Se uma consulta retornar 1.500 itens de 64 bytes cada, o tamanho cumulativo será 96 KB.
- **Scan**- Lê todos os itens em uma tabela. O DynamoDB considera o tamanho dos itens que são avaliados, não o tamanho dos itens retornados pela verificação.

Se você executar uma operação de leitura em um item que não existe, o DynamoDB ainda consome o throughput de leitura provisionado: Uma solicitação de leitura fortemente consistente consome uma unidade de capacidade, enquanto uma solicitação de leitura eventualmente consistente consome 0,5 de uma unidade de capacidade.

Para qualquer operação que retorna itens, você pode solicitar um subconjunto de atributos a serem recuperados. No entanto, isso não tem nenhum impacto sobre os cálculos de tamanho dos itens. Além disso, `Query` e `Scan` podem retornar as contagens de itens, em vez de valores de atributo. A obtenção da contagem de itens usa a mesma quantidade de unidades de capacidade de leitura e está sujeita aos mesmos cálculos de tamanho de item. Isso ocorre porque o DynamoDB precisa ler cada item para aumentar a contagem.

Operações de leitura e consistência de leitura

Os cálculos anteriores presumem solicitações de leitura fortemente consistente. Para uma solicitação de leitura eventualmente consistente, a operação consome apenas metade das unidades de capacidade. Para uma leitura eventualmente consistente, se o tamanho total do item for 80 KB, a operação consumirá apenas 10 unidades de capacidade.

Unidades de capacidade de gravação

A unidade de capacidade de gravação representa uma gravação por segundo para um item de até 1 KB de tamanho.

Por exemplo, suponha que você crie uma tabela com 10 unidades de capacidade de gravação. Isso permite que você execute 10 gravações por segundo para itens de até 1 KB de tamanho por segundo.

Os tamanhos de item para gravações são arredondados até o próximo múltiplo de 1 KB. Por exemplo, gravar um item de 500 bytes consome a mesma taxa de rendimento que gravar um item de 1 KB.

Consumo de unidade de capacidade para gravações

Veja a seguir como as operações de gravação do DynamoDB consomem unidades de capacidade de gravação:

- `PutItem`- grava um único item em uma tabela. Se um item com a mesma chave primária existe na tabela, a operação substitui o item. Para calcular o consumo de taxa de throughput provisionado, o tamanho do item que importa é o maior dos dois.
- `UpdateItem`- Modifica um único item na tabela. O DynamoDB considera o tamanho do item como ele aparece antes e depois da atualização. O throughput provisionado consumido reflete o maior desses tamanhos de item. Mesmo se você atualizar apenas um subconjunto dos atributos do item, `UpdateItem` ainda consumirá a quantidade total do throughput provisionado (o maior dos tamanhos de item de "antes" e "depois").
- `DeleteItem`- remove um único item de uma tabela. O consumo de throughput provisionado é baseado no tamanho do item excluído.
- `BatchWriteItem`— grava até 25 itens em uma ou mais tabelas. O DynamoDB processa cada item no lote como um `indivíduoPutItem` ou `DeleteItem` (atualizações não são suportadas). Então, o DynamoDB arredonda o tamanho de cada item para o próximo limite de 1 KB e, em seguida, calcula o tamanho total. O resultado não é necessariamente o mesmo que o tamanho total de todos os itens. Por exemplo, se `BatchWriteItem` grava um item de 500 bytes e um item de 3,5 KB, o DynamoDB calcula o tamanho como 5 KB (1 KB + 4 KB), não 4 KB (500 bytes + 3,5 KB).

para o `PutItem`, `UpdateItem`, `DeleteItem`, o DynamoDB arredonda o tamanho do item até o próximo 1 KB. Por exemplo, se você inserir ou excluir um item de 1,6 KB, o DynamoDB arredondará o tamanho do item até 2 KB.

`PutItem`, `UpdateItem` e `DeleteItem` permitem gravações condicionais, onde você especifica uma expressão que deve ser verdadeira para a operação ser bem-sucedida. Se a expressão for falsa, o DynamoDB ainda consumirá unidades de capacidade de gravação da tabela:

- Para um item existente, o número de unidades de capacidade de gravação consumidas depende do tamanho do novo item. (Por exemplo, uma gravação condicional falha de um item de 1 KB consumiria uma unidade de capacidade de gravação. Se os novos itens forem duas vezes esse tamanho, a gravação condicional falha consumiria duas unidades de capacidade de gravação.)
- Para um novo item, o DynamoDB consome uma unidade de capacidade de gravação.

Limitação de solicitações e capacidade de intermitência

Se seu aplicativo executar leituras ou gravações a uma taxa mais alta do que a com a qual sua tabela é compatível, o DynamoDB começará a Acelerador de pedidos. Quando o DynamoDB limita uma leitura ou gravação, ele retorna uma `ProvisionedThroughputExceededException` para o chamador. Em seguida, o aplicativo pode executar a ação apropriada, como esperar por um curto intervalo antes de repetir a solicitação.

Note

Recomendamos usar o AWS SDKs para desenvolvimento de software. O AWS SDKs fornecem suporte integrado para repetir solicitações limitadas; você não precisa gravar essa lógica manualmente. Para mais informações, consulte [Repetições de erro e recuo exponencial \(p. 226\)](#).

O console do DynamoDB exibe as métricas do Amazon CloudWatch para as tabelas, portanto, você pode monitorar as solicitações de leitura e de gravação limitadas. Se você encontrar um número excessivo de controle de utilização, considere aumentar suas configurações de taxa de transferência provisionada da tabela.

Em alguns casos, o DynamoDB usa Capacidade de intermitência para acomodar leituras ou gravações em excesso das configurações de throughput da tabela. Com a capacidade de intermitência, as solicitações de leitura ou gravação inesperadas podem ser bem-sucedidas onde de outra forma seriam limitadas. Para mais informações, consulte [Uso efetivo da capacidade de intermitência \(p. 986\)](#).

Limitação de solicitações e capacidade adaptável

O DynamoDB distribui automaticamente seus dados entre as partições, que são armazenados em vários servidores no AWS Cloud (Para obter mais informações, consulte [Partições e distribuição de dados \(p. 23\)](#)). Não é sempre possível distribuir a atividade de leitura e gravação uniformemente o tempo todo. Quando o acesso aos dados é desequilibrado, uma partição “dinâmica” pode receber um volume mais alto de tráfego de leitura e gravação em comparação com outras partições. A capacidade adaptável funciona por meio do aumento automático da capacidade de throughput para as partições que recebem mais tráfego. Para mais informações, consulte [Noções básicas da capacidade adaptável do DynamoDB \(p. 987\)](#).

Escolha de configurações iniciais de throughput

Cada aplicativo tem diferentes requisitos para ler e gravar a partir de um banco de dados. Quando você estiver determinando as configurações iniciais de throughput de uma tabela do DynamoDB, leve as seguintes entradas em consideração:

- Tamanhos de itens. Alguns itens são pequenos o suficiente para que possam ser lidos ou gravados usando uma única unidade de capacidade. Itens maiores exigem várias unidades de capacidade. Ao estimar os tamanhos dos itens que estarão em sua tabela, você pode especificar configurações precisas para seu throughput provisionado.
- Taxas de solicitações de leitura e de gravação esperadas. Além do tamanho do item, você deve estimar o número de leituras e gravações que você precisa executar por segundo.

- Leia os requisitos de consistência. As unidades de capacidade de leitura se baseiam em operações de leitura fortemente consistentes, que consomem duas vezes mais recursos de banco de dados que as leituras eventualmente consistentes. Você deve determinar se o seu aplicativo requer leituras fortemente consistentes, ou se ele pode ignorar essa exigência e executar leituras eventualmente consistentes em vez disso. (Operações de leitura no DynamoDB são eventualmente consistentes por padrão. Você pode solicitar ações leituras fortemente consistentes com essas operações, se necessário.)

Por exemplo, suponha que você queira ler 80 itens por segundo de uma tabela. Os itens têm 3 KB de tamanho e você deseja leituras fortemente consistentes. Para esse cenário, cada leitura requer uma unidade de capacidade de leitura provisionada. Para determinar esse número, divida o tamanho do item da operação por 4 KB e depois arredonde o resultado até o número inteiro mais próximo, como neste exemplo:

- $3 \text{ KB}/4 \text{ KB} = 0,75$, ou 1 Unidade de capacidade de leitura

Para esse cenário, você precisa definir o throughput de leitura provisionado da tabela como 80 unidades de capacidade de leitura:

- 1 unidade de capacidade de leitura por item \times 80 leituras por segundo = 80 unidades de capacidade de leitura

Agora, suponhamos que você queira gravar 100 itens por segundo na tabela e que esses itens tenham 512 bytes. Para esse cenário, cada gravação requer uma unidade de capacidade de gravação provisionada. Para determinar esse número, divida o tamanho do item da operação por 1 KB e depois arredonde o resultado até o número inteiro mais próximo:

- $512 \text{ bytes}/1 \text{ KB} = 0,5$ ou 1

Para esse cenário, você gostaria de definir o throughput provisionado de gravação da tabela como 100 unidades de capacidade de gravação:

- 1 unidade de capacidade de gravação por item \times 100 gravações por segundo = 100 unidades de capacidade de gravação

Note

Para obter recomendações sobre throughput provisionado e tópicos relacionados, consulte [Práticas recomendadas para projetar chaves de partição e usá-las efetivamente \(p. 986\)](#).

Modificação das configurações de throughput

Se você tiver habilitado o Auto Scaling do DynamoDB para uma tabela, sua capacidade de throughput será ajustada dinamicamente em resposta ao uso real. Nenhuma intervenção manual é necessária.

Você pode modificar as configurações de throughput provisionado de sua tabela usando o AWS Management Console ou a operação `UpdateTable`. Para obter mais informações sobre aumentos e diminuições da taxa de transferência por dia, consulte [Cotas de serviço, conta e tabela no Amazon DynamoDB \(p. 1056\)](#).

Tamanhos e formatos de item do DynamoDB

As tabelas do DynamoDB são sem esquema, exceto para a chave primária, portanto, os itens em uma tabela podem ter atributos, tamanhos e tipos de dados diferentes.

O tamanho total de um item é a soma dos tamanhos de seus nomes e valores de atributos. Você pode usar as seguintes diretrizes para estimar os tamanhos de atributo:

- Strings são Unicode com codificação binária UTF-8. O tamanho de uma string é (tamanho do nome do atributo) + (número de bytes codificados em UTF-8).
- Os números são de tamanho variável, com até 38 dígitos significativos. Zeros iniciais e finais são cortados. O tamanho de um número é aproximadamente (tamanho do nome do atributo) + (1 byte por dois dígitos significativos) + (1 byte).
- Um valor binário deve ser codificado no formato base64 antes que possa ser enviado para o DynamoDB, mas o tamanho de byte bruto do valor é usado para calcular o tamanho. O tamanho de um atributo binário é (tamanho do nome do atributo) + (número de bytes brutos).
- O tamanho de um atributo nulo ou de um atributo booleano é (tamanho do nome do atributo) + (1 byte).
- Um atributo do tipo `List` ou `Map` requer 3 bytes de sobrecarga, independentemente de seu conteúdo. O tamanho de uma `List` ou `Map` é (tamanho do nome do atributo) + soma (tamanho dos elementos aninhados) + (3 bytes). O tamanho de uma `List` ou `Map` vazio é (tamanho do nome do atributo) + (3 bytes).

Note

Recomendamos que você escolha nomes mais curtos para atributo. Isso ajuda a reduzir a quantidade de armazenamento necessária, mas também pode reduzir a quantidade de RCU/WCUs usada.

Como gerenciar a capacidade de throughput automaticamente com o Auto Scaling do DynamoDB

Muitas cargas de trabalho de banco de dados são cíclicas por natureza ou são difíceis de prever com antecedência. Por exemplo, considere um aplicativo de rede social na qual a maioria dos usuários está ativa durante o horário diurno. O banco de dados deve ser capaz de lidar com a atividade durante o dia, mas não há necessidade dos mesmos níveis de throughput à noite. Outro exemplo pode ser um novo aplicativo de jogos para celular que está passando por uma rápida adoção. Se o jogo se tornar muito popular, talvez ele exceda os recursos de banco de dados disponíveis, resultando em desempenho lento e clientes insatisfeitos. Esses tipos de cargas de trabalho muitas vezes exigem intervenção manual para dimensionar recursos de banco de dados, aumentando-os ou diminuindo-os em resposta a diferentes níveis de uso.

O dimensionamento automático do Amazon DynamoDB usa o AWS Serviço de Application Auto Scaling para ajustar dinamicamente a capacidade de throughput provisionado em seu nome, em resposta aos padrões de tráfego reais. Isso permite que uma tabela ou um índice secundário global aumente a capacidade provisionada de leitura e gravação para lidar com aumentos repentinos no tráfego, sem restrições. Quando a carga de trabalho diminuir, o Application Auto Scaling diminuirá o throughput, para que você não precise pagar por uma capacidade provisionada não utilizada.

Note

Se você usar a AWS Management Console para criar uma tabela ou um índice secundário global, o Auto Scaling do DynamoDB está habilitada por padrão. É possível modificar as configurações de Auto Scaling a qualquer momento. Para mais informações, consulte [Usar o AWS Management Console Com o DynamoDB Auto Scaling \(p. 353\)](#).

Com o Application Auto Scaling, você cria uma política de escalabilidade para obter uma tabela ou um índice secundário global. A política de escalabilidade especifica se você deseja dimensionar a capacidade de leitura ou a capacidade de gravação (ou ambas), bem como as configurações mínimas e máximas de unidades de capacidade provisionadas para a tabela ou o índice.

A política de dimensionamento também contém uma utilização pretendida—a porcentagem de throughput provisionado consumido em um determinado ponto no tempo. Application Auto Scaling usa um rastreamento de destino para ajustar o throughput provisionado da tabela (ou índice) para cima ou para baixo em resposta a cargas de trabalho reais, para que a utilização da capacidade real permaneça em ou perto de sua utilização prevista.

É possível definir os valores de utilização do destino de Auto Scaling entre 20% e 90% como sua capacidade de gravação e leitura.

Note

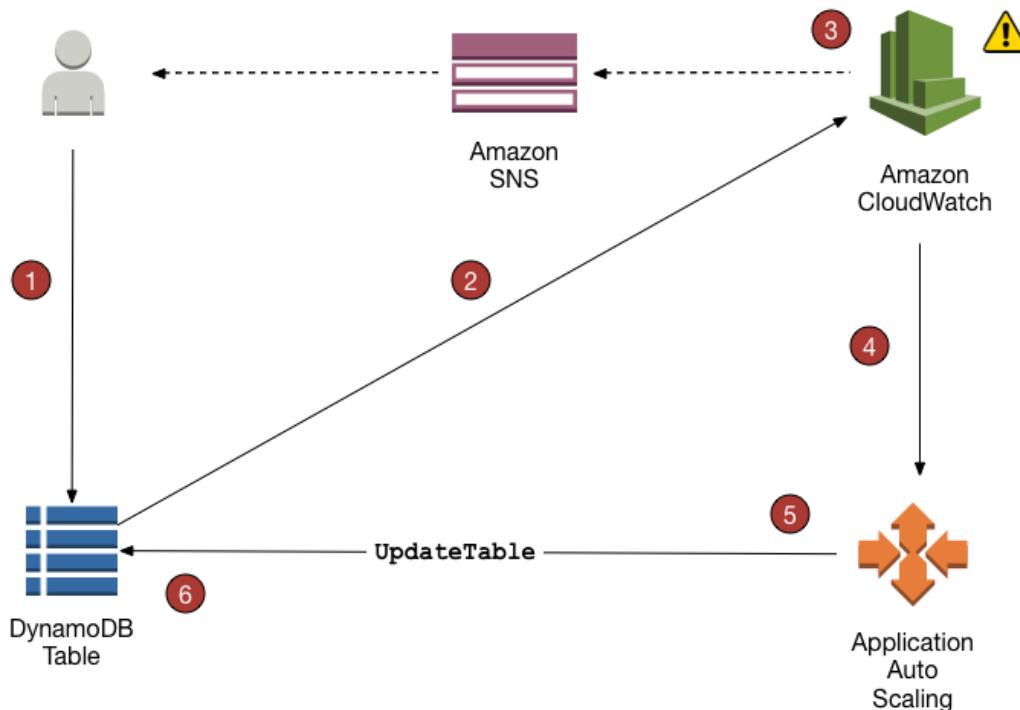
Além de tabelas, o Auto Scaling do DynamoDB também oferece suporte a índices secundários locais. Cada índice secundário global tem sua própria capacidade de throughput provisionado, à parte daquela da tabela base. Quando você cria uma política de escalabilidade para um índice secundário global, o Application Auto Scaling ajusta as configurações de throughput provisionado do índice para garantir que sua utilização real permaneça em ou perto da proporção de utilização desejada.

Como o DynamoDB Auto Scaling funciona

Note

Para começar a trabalhar rapidamente com o Auto Scaling do DynamoDB, consulte [Usar o AWS Management Console Com o DynamoDB Auto Scaling \(p. 353\)](#).

O diagrama a seguir fornece uma visão geral de alto nível de como o Auto Scaling do DynamoDB gerencia a capacidade de throughput para uma tabela.



As etapas a seguir resumem o processo de Auto Scaling, conforme mostrado no diagrama anterior:

1. Você cria uma política de Application Auto Scaling para sua tabela do DynamoDB.
2. O DynamoDB publica métricas de capacidade consumida no Amazon CloudWatch.

3. Se a capacidade consumida da tabela exceder sua utilização prevista (ou cair abaixo desse alvo) por um período específico, o Amazon CloudWatch disparará um alarme. Você pode visualizar o alarme no console e receber notificações usando o Amazon Simple Notification Service (Amazon SNS).
4. O alarme do CloudWatch chama o Application Auto Scaling para avaliar sua política de escalabilidade.
5. Application Auto Scaling emite um `UpdateTable` para ajustar o throughput provisionado da sua tabela.
6. DynamoDB processa uma solicitação `UpdateTable`, aumentando (ou diminuindo) dinamicamente a capacidade de throughput provisionado da tabela de forma que ela se aproxime da sua utilização prevista.

Para entender como o Auto Scaling do DynamoDB funciona, suponha que você tenha uma tabela chamada `ProductCatalog`. A tabela é raramente carregada em massa com dados e, por isso, não provoca muita atividade de gravação. No entanto, ela é submetida a um alto grau de atividades de leitura, que variam ao longo do tempo. Ao monitorar as métricas do Amazon CloudWatch para `ProductCatalog`, você determina que a tabela requer 1.200 unidades de capacidade de leitura (para evitar que o DynamoDB limite solicitações de leitura no pico das atividades). Você também determina que o `ProductCatalog` requer no mínimo 150 unidades de capacidade de leitura quando o tráfego de leitura está em seu ponto mais baixo.

No intervalo de 150 a 1.200 unidades de capacidade de leitura, você decide que uma utilização de destino de 70% seria adequada para a tabela `ProductCatalog`. Utilização pretendida é a proporção entre unidades de capacidade consumidas e unidades de capacidade provisionadas, expressa como porcentagem. O Application Auto Scaling usa seu algoritmo de rastreamento previsto para garantir que a capacidade de leitura provisionada de `ProductCatalog` é ajustada conforme necessário, de forma que a utilização permaneça em ou perto de 70%.

Note

O Auto Scaling do DynamoDB modifica as configurações de throughput provisionado somente quando a carga de trabalho real permanece elevada (ou baixa) por um período prolongado de vários minutos. O algoritmo de rastreamento previsto previsto do Application Auto Scaling do procura manter a utilização prevista em ou perto do seu valor escolhido em longo prazo. Picos de atividade súbitos de curta duração são acomodados pela capacidade de explosão interna da tabela. Para mais informações, consulte [Uso efetivo da capacidade de intermitência \(p. 986\)](#).

Para habilitar o dimensionamento automático do DynamoDB para o `ProductCatalog`, crie uma política de escalabilidade. A política especifica o seguinte:

- A tabela ou o índice secundário global que você deseja gerenciar
- Qual tipo de capacidade gerenciar (capacidade de leitura ou capacidade de gravação)
- Os limites superior e inferior das configurações da taxa de transferência provisionada
- Utilização de destino

Quando você criar uma política de escalabilidade, o Application Auto Scaling cria um par de alarmes do Amazon CloudWatch em seu nome. Cada par representa seus limites superiores e inferiores para configurações de throughput provisionado. Esses alarmes do CloudWatch são disparados quando a utilização real da tabela se desvia da utilização prevista por um período prolongado.

Quando um dos alarmes do CloudWatch for disparado, o Amazon SNS enviará uma notificação (caso você tenha habilitado esse recurso). O alarme do CloudWatch então invoca o Application Auto Scaling, que por sua vez notifica o DynamoDB para ajustar o `ProductCatalog` a capacidade provisionada da tabela para cima ou para baixo, conforme apropriado.

Observações sobre o uso

Antes de começar a usar o Auto Scaling do DynamoDB, você deve estar ciente do seguinte:

- O Auto Scaling do DynamoDB pode aumentar a capacidade de leitura ou gravação sempre que necessário, de acordo com a sua política de Auto Scaling. Todas as cotas do DynamoDB permanecem em vigor, conforme descrito em [Cotas de serviço, conta e tabela no Amazon DynamoDB \(p. 1056\)](#).
- O Auto Scaling do DynamoDB não impede a modificação manual de configurações do throughput provisionado. Esses ajustes manuais não afetam alarmes existentes do CloudWatch do que estejam relacionados ao Auto Scaling do DynamoDB.
- Se você habilitar o Auto Scaling do DynamoDB para uma tabela que tenha um ou mais índices secundários globais, é altamente recomendável que você também aplique o Auto Scaling uniformemente e esses índices. Você pode fazer isso escolhendo Aplicar as mesmas configurações a índices secundários globais no AWS Management Console. Para mais informações, consulte [Como habilitar o Auto Scaling do DynamoDB em tabelas existentes \(p. 354\)](#).

Usar oAWS Management ConsoleCom o DynamoDB Auto Scaling

Quando você usa oAWS Management ConsolePara criar uma nova tabela, o Auto Scaling do Amazon DynamoDB está habilitada para essa tabela por padrão. Você também pode usar o console para habilitar o Auto Scaling de tabelas existentes, modificar as configurações de Auto Scaling ou desabilitar o Auto Scaling.

Note

Para obter recursos mais avançados, como a definição de redução e expansão de períodos de desaquecimento, use aAWS Command Line Interface(AWS CLI) para gerenciar o dimensionamento automático do DynamoDB. Para mais informações, consulte [Usar oAWS CLIPara gerenciar o DynamoDB Auto Scaling \(p. 355\)](#).

Tópicos

- [Antes de começar: Concedendo permissões de usuário para o DynamoDB Auto Scaling \(p. 353\)](#)
- [Como criar uma nova tabela com o Auto Scaling habilitada \(p. 354\)](#)
- [Como habilitar o Auto Scaling do DynamoDB em tabelas existentes \(p. 354\)](#)
- [Como visualizar atividades de Auto Scaling no Console do \(p. 355\)](#)
- [Como modificar ou desabilitar configurações de Auto Scaling do DynamoDB \(p. 355\)](#)

Antes de começar: Concedendo permissões de usuário para o DynamoDB Auto Scaling

DentroAWS Identity and Access Management(IAM), a política gerenciada da `AWSxDynamoDBFullAccess`O fornece as permissões necessárias para usar o console do DynamoDB. No entanto, para o dimensionamento automático do DynamoDB, os usuários do IAM exigem privilégios adicionais.

Important

`application-autoscaling:`*Permissões são necessárias para excluir uma tabela ativada para escalabilidade automática. A política gerenciada da `AWSxDynamoDBFullAccess`inclusasessas permissões.

Para configurar um usuário do IAM para acesso ao console do DynamoDB e Auto Scaling do DynamoDB, adicione a política a seguir.

Para anexar a política `AmazonDynamoDBFullAccess`

1. Faça login no AWS Management Console e abra o console do IAM em <https://console.aws.amazon.com/iam/>.

2. No painel do console do IAM, escolha Usuários e depois escolha seu usuário do IAM na lista.
3. Na página Summary (Resumo), escolha Add permissions (Adicionar permissões).
4. Selecione Attach existing policies directly (Vincular diretamente políticas existentes).
5. Na lista de políticas, escolha AmazonDynamoDBFullAccess, depois, escolha Próximo: Análise.
6. Selecione Add permissions.

Como criar uma nova tabela com o Auto Scaling habilitada

Note

O Auto Scaling do DynamoDB requer a presença de uma função vinculada a serviços (`AWSServiceRoleForApplicationAutoScaling_DynamoDBTable`) que realiza ações de Auto Scaling em seu nome. Esta função é criada automaticamente para você. Para obter mais informações, consulte [Funções vinculadas a serviço do aplicativo Auto Scaling](#), no Guia do usuário do Aplicativo Auto Scaling.

Para criar uma nova tabela com Auto Scaling habilitada

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. Selecione Create Table (Criar tabela).
3. Na página Create DynamoDB table, insira um Table name e os detalhes de Primary key.
4. Certifique-se de que Usar configurações padrão estiver selecionada. (Sua AWS conta já tem `AWSServiceRoleForApplicationAutoScaling_DynamoDBTable`.)

Caso contrário, para configurações personalizadas:

1. Clear Usar configurações padrão.
2. No Auto Scaling, defina as configurações de parâmetros e certifique-se de que `AWSServiceRoleForApplicationAutoScaling_DynamoDBTable` estiver selecionada.
5. Quando estiver satisfeito com as configurações, escolha Create. Sua tabela é criada com os parâmetros de Auto Scaling.

Como habilitar o Auto Scaling do DynamoDB em tabelas existentes

Note

O Auto Scaling do DynamoDB requer a presença de uma função vinculada a serviços (`AWSServiceRoleForApplicationAutoScaling_DynamoDBTable`) que realiza ações de Auto Scaling em seu nome. Esta função é criada automaticamente para você. Para obter mais informações, consulte [Funções vinculadas ao serviço para o Application Auto Scaling](#).

Se você nunca usou o Auto Scaling do DynamoDB antes, consulte [Como criar uma nova tabela com o Auto Scaling habilitada \(p. 354\)](#).

Para habilitar o Auto Scaling do DynamoDB para uma tabela existente

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. Escolha a tabela com a qual você deseja trabalhar e depois escolha Capacity.
3. Na seção Auto Scaling, faça o seguinte:
 - Select Capacidade de leitura, Capacidade de gravação ou ambos. (Para capacidade de gravação, escolha Mesmas configurações que lidas.) Para cada um deles, faça o seguinte:
 - Utilização pretendida— Informe a porcentagem de utilização pretendida para a tabela.
 - Capacidade provisionada mínima— Digite o limite inferior para o intervalo de Auto Scaling.

- Capacidade máxima provisionada— Digite o limite superior para o intervalo de Auto Scaling.
- Aplicar as mesmas configurações a índices secundários globais- Mantenha essa opção na configuração padrão (habilitada).

Note

Para o melhor desempenho, recomendamos que você habilite oAplicar as mesmas configurações a índices secundários globais. Essa opção permite que o Auto Scaling do DynamoDB dimensione uniformemente todos os índices secundários globais na tabela base. Isso inclui índices secundários globais existentes e quaisquer outros que você crie no futuro para essa tabela.

Com essa opção habilitada, não é possível definir uma política de escalabilidade em um índice secundário global individual.

(ParaCapacidade de gravação, você pode escolherMesmas configurações que lidas.)

NoIAM Role, certifique-se de queAWSServiceRoleForApplicationAutoScaling_DynamoDBTableestiver selecionada.

4. Quando estiver satisfeito com as configurações, clique em Salvar.

Como visualizar atividades de Auto Scaling no Console do

À medida que o seu aplicativo direciona o tráfego de leitura e gravação para a sua tabela, o Auto Scaling do DynamoDB modifica dinamicamente as configurações de throughput da tabela.

Para visualizar essas atividades de Auto Scaling no console do DynamoDB, escolha a tabela com a qual você deseja trabalhar. SelecioneCapacidadee, em seguida, expanda aAtividades de escalabilidadeseção. Quando as configurações de throughput da sua tabela forem modificadas, você verá mensagens informativas aqui.

Como modificar ou desabilitar configurações de Auto Scaling do DynamoDB

Você pode usar oAWS Management ConsolePara modificar configurações de Auto Scaling do DynamoDB. Para fazer isso, vá até oCapacidadePara sua tabela e modifique as configurações no na listaAuto Scalingseção. Para obter mais informações sobre essas configurações, consulte [Como habilitar o Auto Scaling do DynamoDB em tabelas existentes \(p. 354\)](#).

Para desativar o Auto Scaling do DynamoDB, vá para oCapacidadePara sua tabela e desmarqueCapacidade de leitura,Capacidade de gravaçãou ambos.

Usar oAWS CLIpara gerenciar o DynamoDB Auto Scaling

Em vez de usar oAWS Management Console, você pode usar oAWS Command Line Interface(AWS CLI) para gerenciar o dimensionamento automático do Amazon DynamoDB. O tutorial desta seção demonstra como instalar e configurar oAWS CLIpara gerenciar o dimensionamento automático do DynamoDB. Neste tutorial, você faz o seguinte:

- Criar uma tabela do DynamoDB chamadaTestTable. As configurações de throughput iniciais são 5 unidades de capacidade de leitura e 5 unidades de capacidade de gravação.
- Crie uma política Application Auto Scaling para oTestTable. A política procura manter uma taxa de destino de 50% entre a capacidade de gravação consumida e a capacidade de gravação provisionada. O intervalo dessa métrica é entre 5 e 10 unidades de capacidade de gravação. (O Application Auto Scaling não tem permissão para ajustar o throughput além desse intervalo.)
- Execute um programa Python para conduzir o tráfego de gravação para oTestTable. Quando a taxa prevista ultrapassar 50% por um tempo contínuo, o Application Auto Scaling do notificará o DynamoDB para ajustar o throughput deTestTablepara cima para manter a utilização alvo de 50%.

- Verifique se o DynamoDB ajustou com sucesso a capacidade de gravação provisionada de `TestTable`.

Tópicos

- [Antes de começar \(p. 356\)](#)
- [Etapa 1: Criação de uma tabela do DynamoDB \(p. 356\)](#)
- [Etapa 2: Registrar um destino escalável \(p. 357\)](#)
- [Etapa 3: Criar uma política de escalabilidade \(p. 357\)](#)
- [Etapa 4: Direcionar o tráfego de gravação para `TestTable` \(p. 359\)](#)
- [Etapa 5: Exibir Application Auto Scaling \(p. 359\)](#)
- [\(Opcional\) Etapa 6: Limpar \(p. 360\)](#)

Antes de começar

Conclua as seguintes tarefas antes de iniciar o tutorial.

Instalar a AWS CLI

Caso ainda não tenha feito isso, você deve instalar e configurar a AWS CLI. Para isso, siga estas instruções no [AWS Command Line Interface Guia do usuário](#):

- [Instalar a AWS CLI](#)
- [Configuração do AWS CLI](#)

Instalar o Python

Parte deste tutorial requer que você execute um programa Python (consulte [Etapa 4: Direcionar o tráfego de gravação para `TestTable` \(p. 359\)](#)). Se ainda não o tiver instalado, você poderá [baixar Python](#).

Etapa 1: Criação de uma tabela do DynamoDB

Nesta etapa, você deve usar o AWS CLI para criar o `TestTable`. A chave primária consiste em `pk` (chave de partição) e `sk` (chave de classificação). Ambos os atributos são do tipo `Number`. As configurações de throughput iniciais são 5 unidades de capacidade de leitura e 5 unidades de capacidade de gravação.

1. Use o seguinte AWS CLI para criar a tabela.

```
aws dynamodb create-table \
--table-name TestTable \
--attribute-definitions \
  AttributeName=pk,AttributeType=N \
  AttributeName=sk,AttributeType=N \
--key-schema \
  AttributeName=pk,KeyType=HASH \
  AttributeName=sk,KeyType=RANGE \
--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5
```

2. Para verificar o status da tabela do, use o comando a seguir.

```
aws dynamodb describe-table \
--table-name TestTable \
--query "Table.[TableName,TableStatus,ProvisionedThroughput]"
```

A tabela está pronta para uso quando seu status é `ACTIVE`.

Etapa 2: Registrar um destino escalável

Em seguida, você registra a capacidade de gravação da tabela como um destino escalável com o Application Auto Scaling. Isso permite que o Application Auto Scaling ajuste a capacidade de gravação provisionada de `TestTable`, mas apenas no intervalo de 5 a 10 unidades de capacidade.

Note

O Auto Scaling do DynamoDB requer a presença de uma função vinculada a serviços (`AWSServiceRoleForApplicationAutoScaling_DynamoDBTable`) que realiza ações de Auto Scaling em seu nome. Esta função é criada automaticamente para você. Para obter mais informações, consulte [Funções vinculadas a serviço do aplicativo Auto Scaling](#), no Guia do usuário do Aplicativo Auto Scaling.

1. Digite o comando a seguir para registrar o destino dimensionável.

```
aws application-autoscaling register-scalable-target \  
  --service-namespace dynamodb \  
  --resource-id "table/TestTable" \  
  --scalable-dimension "dynamodb:table:WriteCapacityUnits" \  
  --min-capacity 5 \  
  --max-capacity 10
```

2. Para verificar o registro, use o comando a seguir.

```
aws application-autoscaling describe-scalable-targets \  
  --service-namespace dynamodb \  
  --resource-id "table/TestTable"
```

Note

Você também pode registrar um alvo escalável em relação a um índice secundário global. Por exemplo, para um índice secundário global ("test-index"), o ID do recurso e os argumentos de dimensão escalável são atualizados adequadamente.

```
aws application-autoscaling register-scalable-target \  
  --service-namespace dynamodb \  
  --resource-id "table/TestTable/index/test-index" \  
  --scalable-dimension "dynamodb:index:WriteCapacityUnits" \  
  --min-capacity 5 \  
  --max-capacity 10
```

Etapa 3: Criar uma política de escalabilidade

Nesta etapa, você cria uma política de escalabilidade para `TestTable`. A política define os detalhes sobre os quais o Auto Scaling do Aplicativo pode ajustar o throughput provisionado da tabela, e as ações que serão executadas quando ele fizer isso. Você associa essa política ao destino escalável que definiu na etapa anterior (unidades de capacidade de gravação da `TestTable`).

A política contém os elementos a seguir:

- `PredefinedMetricSpecification`— A métrica que o Application Auto Scaling tem permissão para ajustar. Para o DynamoDB, os valores a seguir são válidos para o `PredefinedMetricType`:
 - `DynamoDBReadCapacityUtilization`
 - `DynamoDBWriteCapacityUtilization`
- `ScaleOutCooldown`— A quantidade mínima de tempo (em segundos) entre cada evento Application Auto Scaling do que aumenta o throughput provisionado. Esse parâmetro permite que o Application

Auto Scaling aumenta o throughput continuamente, mas não agressivamente, em resposta às cargas de trabalho do mundo real. A configuração padrão de `ScaleOutCooldown` é 0.

- `ScaleInCooldown`- A quantidade mínima de tempo (em segundos) entre cada evento Application Auto Scaling do que diminui o throughput provisionado. Esse parâmetro permite que o Application Auto Scaling diminua o throughput de forma gradual e previsível. A configuração padrão de `ScaleInCooldown` é 0.
- `TargetValue`— O Auto Scaling do aplicativo garante que o índice de capacidade consumida para capacidade provisionada permaneça nesse valor ou próximo a ele. Você define `TargetValue` como uma porcentagem.

Note

Para entender melhor como `TargetValue` funciona, suponha que você tenha uma tabela com uma configuração de throughput provisionado de 200 unidades de capacidade de gravação. Você decide criar uma política de escalabilidade para essa tabela, com um `TargetValue` de 70%. Agora, suponha que você comece a direcionar o tráfego de gravação para a tabela, de forma que o throughput de gravação real seja de 150 unidades de capacidade. A taxa provisionada consumida-para agora é $(150/200)$ ou 75%. Essa taxa excede o seu destino, portanto, o Auto Scaling do Aplicativo aumenta a capacidade de gravação provisionada para 215, de modo que a taxa seja $(150/215)$ ou 69,77% - tão próxima ao seu `TargetValue` quanto possível, mas não excedê-lo.

para o `TestTable`, você define `TargetValue` para 50 por cento. O Application Auto Scaling ajusta o throughput provisionado da tabela dentro do intervalo de 5 a 10 unidades de capacidade (consulte [Etapa 2: Registrar um destino escalável \(p. 357\)](#)), para que a taxa consumida-para-provisionada permaneça em 50% ou próxima a isso. Você define os valores de `ScaleOutCooldown` e `ScaleInCooldown` para 60 segundos.

1. Crie um arquivo denominado `scaling-policy.json` com o seguinte conteúdo.

```
{  
    "PredefinedMetricSpecification": {  
        "PredefinedMetricType": "DynamoDBWriteCapacityUtilization"  
    },  
    "ScaleOutCooldown": 60,  
    "ScaleInCooldown": 60,  
    "TargetValue": 50.0  
}
```

2. Use o seguinte AWS CLI para criar a política.

```
aws application-autoscaling put-scaling-policy \  
    --service-namespace dynamodb \  
    --resource-id "table/TestTable" \  
    --scalable-dimension "dynamodb:table:WriteCapacityUnits" \  
    --policy-name "MyScalingPolicy" \  
    --policy-type "TargetTrackingScaling" \  
    --target-tracking-scaling-policy-configuration file://scaling-policy.json
```

3. Na saída, observe que o Application Auto Scaling criou dois alarmes do Amazon CloudWatch - um para o limite superior e outro para o limite inferior da faixa do destino de escalabilidade.
4. Use o seguinte AWS CLI para exibir mais detalhes sobre a política de escalabilidade do.

```
aws application-autoscaling describe-scaling-policies \  
    --service-namespace dynamodb \  
    --resource-id "table/TestTable" \  
    --policy-name "MyScalingPolicy"
```

5. Na saída, verifique se as configurações da política correspondem às suas especificações de [Etapa 2: Registrar um destino escalável \(p. 357\)](#) e [Etapa 3: Criar uma política de escalabilidade \(p. 357\)](#).

Etapa 4: Direcionar o tráfego de gravação para TestTable

Agora, você pode testar a política de escalabilidade ao gravar dados em `TestTable`. Para fazer isso, execute um programa Python.

1. Crie um arquivo denominado `bulk-load-test-table.py` com o seguinte conteúdo.

```
import boto3
dynamodb = boto3.resource('dynamodb')

table = dynamodb.Table("TestTable")

filler = "x" * 100000

i = 0
while (i < 10):
    j = 0
    while (j < 10):
        print (i, j)

        table.put_item(
            Item={
                'pk':i,
                'sk':j,
                'filler':{'S':filler}
            }
        )
        j += 1
    i += 1
```

2. Digite o comando a seguir para executar o programa.

```
python bulk-load-test-table.py
```

A capacidade de gravação provisionada do `TestTable` é muito baixa (5 unidades de capacidade de gravação), para que o programa pare ocasionalmente devido a restrições de gravação. Esse comportamento é esperado.

Deixe o programa continuar em execução enquanto você passa para a próxima etapa.

Etapa 5: Exibir Application Auto Scaling

Nesta etapa, você visualiza as ações do Auto Scaling do Aplicativo que são iniciadas em seu nome. Você também pode verificar se Application Auto Scaling do atualizou a capacidade de gravação provisionada de `TestTable`.

1. Insira o comando a seguir para visualizar as ações do Application Auto Scaling.

```
aws application-autoscaling describe-scaling-activities \
--service-name dynamodb
```

Execute esse comando ocasionalmente, enquanto o programa Python está em execução. (Demora alguns minutos antes que sua política de escalabilidade seja invocada.) Você deve finalmente ver a saída a seguir.

```
...
```

```
{  
    "ScalableDimension": "dynamodb:table:WriteCapacityUnits",  
    "Description": "Setting write capacity units to 10.",  
    "ResourceId": "table/TestTable",  
    "ActivityId": "0cc6fb03-2a7c-4b51-b67f-217224c6b656",  
    "StartTime": 1489088210.175,  
    "ServiceNamespace": "dynamodb",  
    "EndTime": 1489088246.85,  
    "Cause": "monitor alarm AutoScaling-table/TestTable-AlarmHigh-1bb3c8db-1b97-4353-  
baf1-4def76f4e1b9 in state ALARM triggered policy MyScalingPolicy",  
    "StatusMessage": "Successfully set write capacity units to 10. Change successfully  
fulfilled by dynamodb.",  
    "StatusCode": "Successful"  
},  
...  
}
```

Isso indica que o Application Auto Scaling emitiu um `UpdateTable` solicitação ao DynamoDB.

2. Digite o comando a seguir para verificar se o DynamoDB aumentou a capacidade de gravação da tabela.

```
aws dynamodb describe-table \  
  --table-name TestTable \  
  --query "Table.[TableName,TableStatus,ProvisionedThroughput]"
```

As `WriteCapacityUnits` devem ter sido dimensionadas de 5 para 10.

(Opcional) Etapa 6: Limpar

Neste tutorial, você criou vários recursos. Você pode excluir esses recursos se eles não forem mais necessários.

1. Excluir a política de escalabilidade do `TestTable`.

```
aws application-autoscaling delete-scaling-policy \  
  --service-namespace dynamodb \  
  --resource-id "table/TestTable" \  
  --scalable-dimension "dynamodb:table:WriteCapacityUnits" \  
  --policy-name "MyScalingPolicy"
```

2. Cancelar o registro do alvo escalável.

```
aws application-autoscaling deregister-scalable-target \  
  --service-namespace dynamodb \  
  --resource-id "table/TestTable" \  
  --scalable-dimension "dynamodb:table:WriteCapacityUnits"
```

3. Exclua a tabela `TestTable`.

```
aws dynamodb delete-table --table-name TestTable
```

Usar o AWSSDK para configurar o Auto Scaling em tabelas do Amazon DynamoDB

Além de usar o AWS Management Console e a AWS Command Line Interface(AWS CLI), você pode escrever aplicativos que interagem com o Auto Scaling do Amazon DynamoDB. Esta seção contém dois programas Java que podem ser usados para testar essa funcionalidade:

- `EnableDynamoDBAutoscaling.java`
- `DisableDynamoDBAutoscaling.java`

Como habilitar o Application Auto Scaling para uma tabela

O programa a seguir mostra um exemplo de configuração de uma política de Auto Scaling para uma tabela do DynamoDB (`TestTable`). Ele procede da seguinte maneira:

- O programa registra unidades de capacidade de gravação como um destino dimensionável para a `TestTable`. O intervalo dessa métrica é entre 5 e 10 unidades de capacidade de gravação.
- Depois que o destino escalável é criado, o programa cria uma configuração de rastreamento de destino. A política procura manter uma taxa de destino de 50% entre a capacidade de gravação consumida e a capacidade de gravação provisionada.
- Em seguida, o programa cria a política de escalabilidade com base na configuração de rastreamento de destino.

O programa requer que você forneça um nome de recurso da Amazon (ARN) para uma função vinculada ao serviço Application Auto Scaling do válida. (Por exemplo: `arn:aws:iam::122517410325:role/AWSServiceRoleForApplicationAutoScaling_DynamoDBTable`.) No seguinte programa, substitua `SERVICE_ROLE_ARN_Goes_Here` pelo ARN real.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
package com.amazonaws.codesamples.autoscaling;  
  
import com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClient;  
import  
    com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClientBuilder;  
import com.amazonaws.services.applicationautoscaling.model.DescribeScalableTargetsRequest;  
import com.amazonaws.services.applicationautoscaling.model.DescribeScalableTargetsResult;  
import com.amazonaws.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;  
import com.amazonaws.services.applicationautoscaling.model.DescribeScalingPoliciesResult;  
import com.amazonaws.services.applicationautoscaling.model.MetricType;  
import com.amazonaws.services.applicationautoscaling.model.PolicyType;  
import com.amazonaws.services.applicationautoscaling.model.PredefinedMetricSpecification;  
import com.amazonaws.services.applicationautoscaling.model.PutScalingPolicyRequest;  
import com.amazonaws.services.applicationautoscaling.model.RegisterScalableTargetRequest;  
import com.amazonaws.services.applicationautoscaling.model.ScalableDimension;  
import com.amazonaws.services.applicationautoscaling.model.ServiceNamespace;  
import  
    com.amazonaws.services.applicationautoscaling.model.TargetTrackingScalingPolicyConfiguration;  
  
public class EnableDynamoDBAutoscaling {  
  
    static AWSApplicationAutoScalingClient aaClient = (AWSApplicationAutoScalingClient)  
AWSApplicationAutoScalingClientBuilder.standard().build();  
  
    public static void main(String args[]) {
```

```
ServiceNamespace ns = ServiceNamespace.Dynamodb;
ScalableDimension tableWCUs = ScalableDimension.DynamodbTableWriteCapacityUnits;
String resourceID = "table/TestTable";

// Define the scalable target
RegisterScalableTargetRequest rstRequest = new RegisterScalableTargetRequest()
    .withServiceNamespace(ns)
    .withResourceId(resourceID)
    .withScalableDimension(tableWCUs)
    .withMinCapacity(5)
    .withMaxCapacity(10)
    .withRoleARN("SERVICE_ROLE_ARN_Goes_Here");

try {
    aaClient.registerScalableTarget(rstRequest);
} catch (Exception e) {
    System.err.println("Unable to register scalable target: ");
    System.err.println(e.getMessage());
}

// Verify that the target was created
DescribeScalableTargetsRequest dscRequest = new DescribeScalableTargetsRequest()
    .withServiceNamespace(ns)
    .withScalableDimension(tableWCUs)
    .withResourceIds(resourceID);
try {
    DescribeScalableTargetsResult dsaResult =
        aaClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(dsaResult);
    System.out.println();
} catch (Exception e) {
    System.err.println("Unable to describe scalable target: ");
    System.err.println(e.getMessage());
}

System.out.println();

// Configure a scaling policy
TargetTrackingScalingPolicyConfiguration targetTrackingScalingPolicyConfiguration =
    new TargetTrackingScalingPolicyConfiguration()
        .withPredefinedMetricSpecification(
            new PredefinedMetricSpecification()
                .withPredefinedMetricType(MetricType.DynamoDBWriteCapacityUtilization))
        .withTargetValue(50.0)
        .withScaleInCooldown(60)
        .withScaleOutCooldown(60);

// Create the scaling policy, based on your configuration
PutScalingPolicyRequest pspRequest = new PutScalingPolicyRequest()
    .withServiceNamespace(ns)
    .withScalableDimension(tableWCUs)
    .withResourceId(resourceID)
    .withPolicyName("MyScalingPolicy")
    .withPolicyType(PolicyType.TargetTrackingScaling)
    .withTargetTrackingScalingPolicyConfiguration(targetTrackingScalingPolicyConfiguration);

try {
    aaClient.putScalingPolicy(pspRequest);
} catch (Exception e) {
    System.err.println("Unable to put scaling policy: ");
    System.err.println(e.getMessage());
}

// Verify that the scaling policy was created
DescribeScalingPoliciesRequest dspRequest = new DescribeScalingPoliciesRequest()
```

```
.withServiceNamespace(ns)
.withScalableDimension(tableWCUs)
.withResourceId(resourceID);

try {
    DescribeScalingPoliciesResult dspResult =
aaClient.describeScalingPolicies(dspRequest);
    System.out.println("DescribeScalingPolicies result: ");
    System.out.println(dspResult);
} catch (Exception e) {
    e.printStackTrace();
    System.err.println("Unable to describe scaling policy: ");
    System.err.println(e.getMessage());
}

}
```

Como desabilitar o Application Auto Scaling para uma tabela

O programa a seguir inverte o processo anterior. Ele remove a política de Auto Scaling e cancela o registro do destino dimensionável.

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
package com.amazonaws.codesamples.autoscaling;

import com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClient;
import com.amazonaws.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import com.amazonaws.services.applicationautoscaling.model.DeregisterScalableTargetRequest;
import com.amazonaws.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import com.amazonaws.services.applicationautoscaling.model.DescribeScalableTargetsResult;
import com.amazonaws.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import com.amazonaws.services.applicationautoscaling.model.DescribeScalingPoliciesResult;
import com.amazonaws.services.applicationautoscaling.model.ScalableDimension;
import com.amazonaws.services.applicationautoscaling.model.ServiceNamespace;

public class DisableDynamoDBAutoscaling {

    static AWSApplicationAutoScalingClient aaClient = new
AWSApplicationAutoScalingClient();

    public static void main(String args[]) {

        ServiceNamespace ns = ServiceNamespace.Dynamodb;
        ScalableDimension tableWCUs = ScalableDimension.DynamodbTableWriteCapacityUnits;
        String resourceID = "table/TestTable";

        // Delete the scaling policy
        DeleteScalingPolicyRequest delSPRequest = new DeleteScalingPolicyRequest()
            .withServiceNamespace(ns)
```

```
.withScalableDimension(tableWCUs)
.withResourceId(resourceID)
.withPolicyName("MyScalingPolicy");

try {
    aaClient.deleteScalingPolicy(delSPRequest);
} catch (Exception e) {
    System.err.println("Unable to delete scaling policy: ");
    System.err.println(e.getMessage());
}

// Verify that the scaling policy was deleted
DescribeScalingPoliciesRequest descSPRequest = new DescribeScalingPoliciesRequest()
.withServiceNamespace(ns)
.withScalableDimension(tableWCUs)
.withResourceId(resourceID);

try {
    DescribeScalingPoliciesResult dspResult =
aaClient.describeScalingPolicies(descSPRequest);
    System.out.println("DescribeScalingPolicies result: ");
    System.out.println(dspResult);
} catch (Exception e) {
    e.printStackTrace();
    System.err.println("Unable to describe scaling policy: ");
    System.err.println(e.getMessage());
}

System.out.println();

// Remove the scalable target
DeregisterScalableTargetRequest delSTRequest = new DeregisterScalableTargetRequest()
.withServiceNamespace(ns)
.withScalableDimension(tableWCUs)
.withResourceId(resourceID);

try {
    aaClient.deregisterScalableTarget(delSTRequest);
} catch (Exception e) {
    System.err.println("Unable to deregister scalable target: ");
    System.err.println(e.getMessage());
}

// Verify that the scalable target was removed
DescribeScalableTargetsRequest dscRequest = new DescribeScalableTargetsRequest()
.withServiceNamespace(ns)
.withScalableDimension(tableWCUs)
.withResourceIds(resourceID);

try {
    DescribeScalableTargetsResult dsaResult =
aaClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(dsaResult);
    System.out.println();
} catch (Exception e) {
    System.err.println("Unable to describe scalable target: ");
    System.err.println(e.getMessage());
}

}
```

Tabelas globais Replicação em várias regiões com o DynamoDB

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte [Determinar a versão \(p. 366\)](#).

As tabelas globais do Amazon DynamoDB são uma solução totalmente gerenciada para a implantação de banco de dados com várias regiões e não exigem que você crie e mantenha sua própria solução de replicação. Com tabelas globais, você pode especificar oAWSRegiões onde deseja que a tabela esteja disponível. O DynamoDB executa todas as tarefas necessárias para criar tabelas idênticas nessas regiões e propaga continuamente as alterações de dados para todas elas.

Por exemplo, suponha que você tenha uma ampla base de clientes distribuídos em três áreas geográficas: a Costa Leste dos EUA, a Costa Oeste dos EUA e a Europa Ocidental. Os clientes podem atualizar suas informações de perfil usando seu aplicativo. Para satisfazer esse caso de uso, é necessário criar três tabelas do DynamoDB idênticas, chamadasCustomerProfiles, em três diferentesAWSRegiões onde os clientes estão localizados. Essas três tabelas seriam totalmente separadas umas das outras. As alterações nos dados de uma tabela não seriam refletidas nas outras tabelas. Sem uma solução de replicação gerenciada, você poderia gravar código para replicar alterações nos dados entre essas tabelas. No entanto, isso seria um esforço demorado e trabalhoso.

Em vez de escrever seu próprio código, você pode criar uma tabela global que englobe as três regiões específicas à região.CustomerProfilesTabelas do. O DynamoDB pode então replicar automaticamente as alterações de dados entre essas tabelas para que o mude para oCustomerProfilesdados em uma região se propagariam perfeitamente para as outras regiões. Além disso, se um dosAWSAs regiões se tornassem temporariamente indisponíveis, os clientes ainda poderiam acessar o mesmoCustomerProfilesnas outras regiões.

As tabelas globais do DynamoDB são ideais para aplicativos muito dimensionados e com usuários globalmente dispersos. Nesse ambiente, os usuários esperam aplicativos de altíssimo desempenho. As tabelas globais fornecem replicação automática multiativa para oAWSRegiões do mundo inteiro. Elas permitem fornecer acesso de baixa latência aos usuários independentemente de onde estão localizados.

Note

- Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Recomendamos o uso de [Versão 2019.11.21 \(atual\) \(p. 367\)](#) das tabelas globais, que permite a adição dinâmica de notas tabelas-réplica de uma tabela preenchida com dados. O [Versão 2019.11.21 \(atual\) \(p. 367\)](#) é mais eficiente e consome menos capacidade de gravação que [Versão 2017.11.29 \(p. 379\)](#).
- O suporte de região para tabelas globais [Versão 2017.11.29 \(p. 379\)](#) é limitado ao Leste dos EUA (Norte da Virgínia), Leste dos EUA (Ohio), Oeste dos EUA (Norte da Califórnia), Oeste dos EUA (Oregon), Europa (Irlanda), Europa (Londres), Europa (Frankfurt), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio) e Ásia-Pacífico (Seul).
- Se estiver usando o [Versão 2019.11.21 \(atual\) \(p. 367\)](#) das tabelas globais e também usar o recurso [vida útil](#), o DynamoDB replicará as exclusões do TTL em todas as tabelas-réplica. A exclusão inicial do TTL não consome capacidade de gravação na região onde a expiração do TTL ocorre. No entanto, a exclusão do TTL replicada para as tabelas-réplica consome uma unidade de capacidade de gravação replicada ao usar a capacidade provisionada ou a gravação replicada ao usar o modo de capacidade sob demanda, em cada uma das regiões de réplica, e serão aplicadas cobranças.
- As operações transacionais fornecem garantia de atomicidade, consistência, isolamento e durabilidade (ACID) somente na região onde a gravação é realizada originalmente. As

transações não são compatíveis entre as regiões em tabelas globais. Por exemplo, se você tiver uma tabela global com réplicas nas regiões Leste dos EUA (Ohio) e Oeste dos EUA (Oregon) e realizar uma operação TransactWriteItems na região Leste dos EUA (Norte da Virgínia), poderá observar transações parcialmente concluídas na região Oeste dos EUA (Oregon) à medida que as alterações forem replicadas. As alterações só serão replicadas para outras regiões quando forem confirmadas na região de origem.

- Se o CMK gerenciado pelo cliente usado para criptografar uma réplica estiver inacessível, o DynamoDB removerá essa réplica do grupo de replicação. A réplica não será excluída e a replicação será interrompida de e para esta região, 20 horas após a detecção do AWS KMS como inacessível.
- Se você [Desabilite um AWS Região](#), o DynamoDB removerá essa réplica do grupo de replicação, 20 horas após a detecção do AWS Região como inacessível. A réplica não será excluída e a replicação será interrompida de e para esta região.
- Se quiser desativar o [CMK gerenciada pelo cliente](#) que é usado para criptografar uma tabela de réplica, você deve fazer isso somente se a chave não for mais usada para criptografar uma tabela de réplica. Depois de emitir um comando para excluir uma tabela de réplica, você deve aguardar a conclusão da operação de exclusão e para que a tabela global se torne [Active](#) antes de desabilitar a chave. Não fazer isso pode resultar em replicação parcial de dados de e para a tabela de réplicas.
- Se você deseja modificar ou excluir sua política de função do IAM para a tabela de réplica, você deve fazê-lo quando a tabela de réplica estiver no [Active](#) estado. Caso contrário, a criação, atualização ou exclusão da tabela de réplica poderá falhar.
- Tabelas globais podem executar AWS KMS usando o comando `AWSServiceRoleForDynamoDBReplication` função vinculada ao serviço do [CMK gerenciada pelo cliente](#) ou o [AWS CMK gerenciada pela](#) usado para criptografar a réplica.

Para obter informações sobre o AWS Disponibilidade e preços da região, consulte [Definição de preço do Amazon DynamoDB](#).

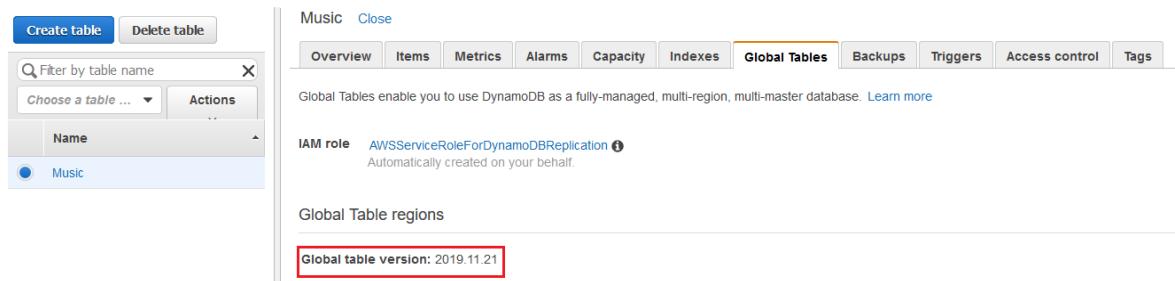
Tópicos

- [Determinar a versão das tabelas globais que estão sendo usadas \(p. 366\)](#)
- [Versão 2019.11.21 \(atual\) \(p. 367\)](#)
- [Versão 2017.11.29 \(p. 379\)](#)

Determinar a versão das tabelas globais que estão sendo usadas

Para descobrir qual versão das tabelas globais você está usando, faça o seguinte:

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/home>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Escolha a tabela que deseja usar.
4. Selecione a guia Global Tables (Tabelas globais).
5. A Global Table version (Versão da tabela global) exibe a versão das tabelas globais em uso.



Versão 2019.11.21 (atual)

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte[Determinar a versão \(p. 366\)](#).

Tópicos

- [Tabelas globais Como ele funciona \(p. 367\)](#)
- [Melhores práticas e requisitos do gerenciamento de tabelas globais \(p. 368\)](#)
- [Tutorial: Como criar uma tabela global \(p. 369\)](#)
- [Monitorar as tabelas globais \(p. 373\)](#)
- [Como usar o IAM com tabelas globais \(p. 374\)](#)
- [Atualizar tabelas globais para a versão 2019.11.21 \(Atual\) \(p. 376\)](#)

Tabelas globais Como ele funciona

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte[Determinar a versão \(p. 366\)](#).

As seções a seguir ajudam você a entender os conceitos e o comportamento das tabelas globais no Amazon DynamoDB.

Conceitos de tabela global

A Tabela global é uma coleção de uma ou mais tabelas de réplica, todas pertencentes a uma única AWS conta.

A Tabela de réplica (ou Réplica Em resumo) é uma única tabela do DynamoDB que funciona como parte de uma tabela global. Cada réplica armazena o mesmo conjunto de itens de dados. Qualquer tabela global só pode ter uma tabela de réplica por AWS Região. Para obter mais informações sobre como começar a usar as tabelas globais, consulte [Tutorial: Como criar uma tabela global \(p. 369\)](#).

Quando você criar uma tabela global do DynamoDB, ela consiste em várias tabelas de réplica (uma por região) que o DynamoDB trata como uma única unidade. Cada réplica possui o mesmo nome de tabela e o mesmo esquema de chave primária. Quando um aplicativo grava dados em uma tabela de réplica em uma região, o DynamoDB propaga a gravação para as outras tabelas de réplica nas outras tabelas de réplica nas outras AWS Regiões automaticamente.

Você pode adicionar tabelas de réplica à tabela global para que ela fique disponível em outras regiões.

Consistência e resolução de conflitos

Todas as alterações feitas em qualquer item de qualquer tabela de réplica serão replicadas para todas as outras réplicas dentro da mesma tabela global. Em uma tabela global, um item recém-gravado geralmente é propagado para todas as tabelas de réplica dentro de um segundo.

Com uma tabela global, cada tabela de réplica armazena o mesmo conjunto de itens de dados. O DynamoDB não oferece suporte à replicação parcial de apenas alguns dos itens.

Um aplicativo pode ler e gravar dados em qualquer tabela de réplica. Se seu aplicativo usar somente leituras eventualmente consistentes e somente leituras de problemas em um AWS Região, ele funcionará sem qualquer modificação. No entanto, se seu aplicativo exigir leituras muito consistentes, ele precisará executar todas as suas leituras e gravações muito consistentes na mesma região. O DynamoDB não oferece suporte a leituras altamente consistentes entre regiões. Portanto, se você gravar em uma região e ler em outra região, a resposta lida poderá incluir dados obsoletos que não refletem os resultados das gravações concluídas recentemente na outra região.

Se os aplicativos atualizarem o mesmo item em diferentes regiões e quase no mesmo tempo, os conflitos podem surgir. Para ajudar a garantir uma consistência eventual, as tabelas globais do DynamoDB usam um último escritor vence Reconciliação entre atualizações simultâneas, na qual o DynamoDB faz o melhor esforço para determinar o último gravador. Com esse mecanismo de resolução de conflitos, todas as réplicas concordarão com a atualização mais recente e serão convertidas para um estado em que todas tenham dados idênticos.

Disponibilidade e durabilidade

Se um único AWS A region se tornar isolada ou degradada, seu aplicativo poderá realizar redirecionamentos para uma região diferente e executar leituras e gravações em uma tabela de réplica diferente. Você pode aplicar lógica de negócios personalizada para determinar quando redirecionar solicitações para outras regiões.

Se uma região se tornar isolada ou degradada, o DynamoDB acompanhará as gravações executadas que ainda não foram propagadas para todas as tabelas de réplica. Quando a região ficar online novamente, o DynamoDB retomará a propagação de todas as gravações pendentes dessa região para as tabelas de réplica nas outras regiões. Ele também retoma a propagação de gravações de outras tabelas de réplica para a região que está online novamente.

Melhores práticas e requisitos do gerenciamento de tabelas globais

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte [Determinar a versão \(p. 366\)](#).

Ao usar as tabelas globais do Amazon DynamoDB, você pode replicar os dados da tabela no AWS Regiões. É importante que as tabelas-réplica e índices secundários na tabela global tenham configurações idênticas de capacidade de gravação para garantir a replicação apropriada dos dados.

Práticas recomendadas e requisitos de gerenciamento de capacidade

Considere o seguinte ao gerenciar configurações de capacidade de tabelas-réplica no DynamoDB.

Ao usar a versão mais recente das tabelas globais, você deve usar a capacidade sob demanda ou habilitar o dimensionamento automático na tabela. O uso do dimensionamento automático ou da capacidade sob demanda garante que você sempre tenha capacidade suficiente para executar gravações replicadas em todas as regiões da tabela global. O número de unidades de capacidade de gravação replicadas (RWCUs) ou unidades de solicitação de gravação replicada será igual em todas as regiões da tabela global. Por exemplo, suponha que você espera 5 gravações por segundo em sua tabela-réplica em Ohio e 5 gravações por segundo em sua tabela-réplica no Norte da Virgínia. Nesse caso, você deve

esperar consumir 10 RWCUs (ou 10 unidades de solicitação de gravação replicadas, se estiver usando capacidade sob demanda) em cada região, Ohio e Norte da Virgínia. Em outras palavras, você deve esperar consumir 20 RWCUs total em Ohio e N. Virginia. Ao usar o modo de capacidade provisionada, você gerencia sua política de auto scaling com UpdateTableReplicaAutoScaling. O throughput mínimo e máximo e a utilização do destino são estabelecidos globalmente para a tabela e passados para todas as réplicas da tabela.

Para obter detalhes sobre o dimensionamento automático e o DynamoDB, consulte [Como gerenciar a capacidade de throughput automaticamente com o Auto Scaling do DynamoDB \(p. 350\)](#).

Tutorial: Como criar uma tabela global

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte [Determinar a versão \(p. 366\)](#).

Esta seção descreve como criar uma tabela global usando o console do Amazon DynamoDB ou a AWS Command Line Interface(AWS CLI).

Tópicos

- [Como criar uma tabela global \(console\) \(p. 369\)](#)
- [Criar uma tabela global \(AWS CLI\) \(p. 370\)](#)
- [Criar uma tabela global \(Java\) \(p. 372\)](#)

Como criar uma tabela global (console)

Siga estas etapas para criar uma tabela global usando o console. O exemplo a seguir cria uma tabela global com tabelas-réplica nos Estados Unidos e na Europa.

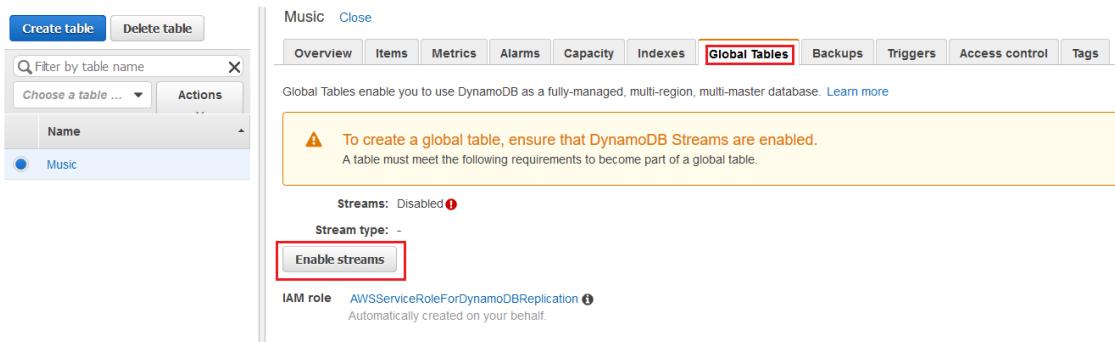
1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/home>. Para este exemplo, escolha a região us-east-2 (Leste dos EUA (Ohio)).
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Selecione Criar tabela.

Em Table name (Nome da tabela), insira **Music**.

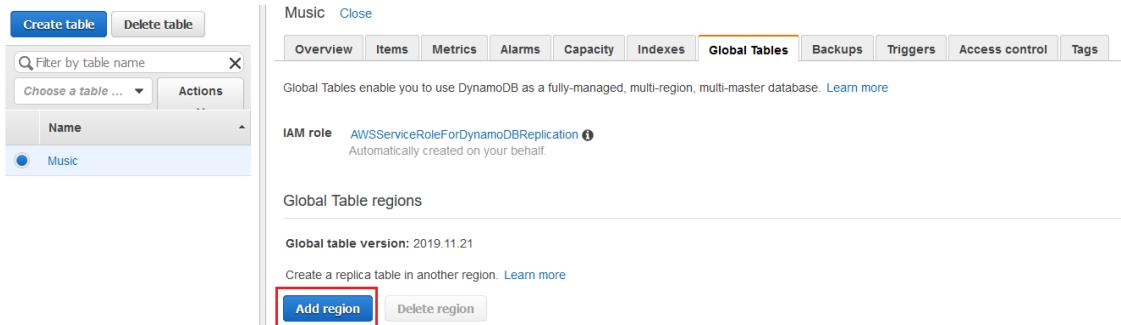
Em Primary key (Chave primária) insira **Artist**. Escolha Add sort key (Adicionar chave de classificação) e insira **SongTitle** (**Artist** e **SongTitle** devem ser strings).

Para criar a tabela, selecione Create (Criar). Essa tabela serve como a primeira tabela-réplica em uma nova tabela global. Ela é o protótipo das outras tabelas-réplica que serão adicionadas posteriormente.

4. Selecione a guia Global Tables (Tabelas globais) e Enable streams (Habilitar streams). Deixe View type (Tipo de visualização) com o valor padrão (imagens novas e antigas).



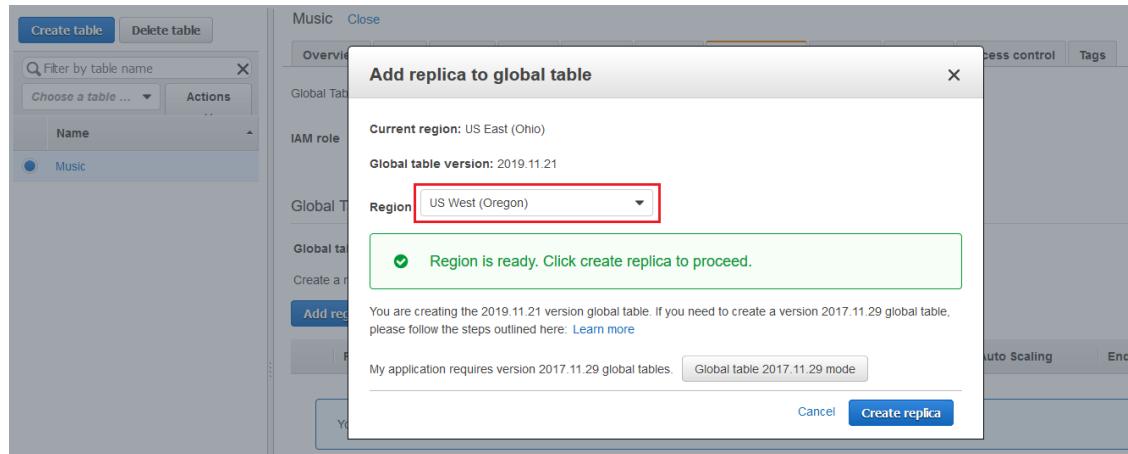
5. Selecione Add region (Adicionar região).



6. DentroRegião, escolhaOeste dos EUA (Oregon).

O console faz uma verificação para garantir que não exista uma tabela com o mesmo nome na região selecionada. Se existir uma tabela com o mesmo nome, será necessário excluir a tabela existente para criar outra tabela-réplica nessa região.

SelecionarCriar réplicasso iniciará o processo de criação da tabela no Oeste dos EUA (Oregon).



A guia Global Table (Tabela global) da tabela selecionada (e de qualquer outra tabela-réplica) mostra que a tabela foi replicada em várias regiões.

7. Agora, adicione outra região para que a tabela global seja replicada e sincronizada nos Estados Unidos e na Europa. Para fazer isso, repita a etapa 5, mas, desta vez, especifiqueUE (Frankfurt)Em vez deOeste dos EUA (Oregon).
8. Você ainda deve estar usando o AWS Management Console na região us-east-2 (Leste dos EUA (Ohio)). Para a tabela Music (Música), selecione a guia Items (Itens) e escolha Create Item (Criar item). para oArtista, insiraitem_1. Em SongTitle (Nome da música), insira Song Value 1. Para gravar o item, selecioneSave (Salvar).

Pouco tempo depois o item será replicado em todas as três regiões da tabela global. Para confirmar isso, no seletor de regiões no canto superior direito do console, selecioneEuropa (Frankfurt). OMusicTabela na Europa (Frankfurt) deve conter o novo item.

Repita isso para Oeste dos EUA (Oregon).

Criar uma tabela global (AWS CLI)

Siga estas etapas para criar uma tabela global Music usando a AWS CLI. O exemplo a seguir cria uma tabela global com tabelas-réplica nos Estados Unidos e na Europa.

- Criar uma nova tabela (`Music`) no Leste dos EUA (Ohio), com os DynamoDB Streams habilitados (`NEW_AND_OLD_IMAGES`).

```
aws dynamodb create-table \
--table-name Music \
--attribute-definitions \
  AttributeName=Artist,AttributeType=S \
  AttributeName=SongTitle,AttributeType=S \
--key-schema \
  AttributeName=Artist,KeyType=HASH \
  AttributeName=SongTitle,KeyType=RANGE \
--billing-mode PAY_PER_REQUEST \
--stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES \
--region us-east-2
```

- Crie um `MusicTabela` em Leste dos EUA (Norte da Virgínia).

```
aws dynamodb update-table --table-name Music --cli-input-json \
'{
  "ReplicaUpdates": [
    {
      "Create": {
        "RegionName": "us-east-1"
      }
    }
  ]
}'
```

- Repita a etapa 2 para criar uma tabela na Europa (Irlanda) (`eu-west-1`).
- É possível visualizar a lista de réplicas criadas usando `describe-table`.

```
aws dynamodb describe-table --table-name Music --region us-east-2
```

- Para verificar se a replicação está funcionando, adicione um novo item à propriedade `Music` tabela na região Leste dos EUA (Ohio).

```
aws dynamodb put-item \
--table-name Music \
--item '{"Artist": {"S":"item_1"}, "SongTitle": {"S":"Song Value 1"}}' \
--region us-east-2
```

- Aguarde alguns segundos e, em seguida, verifique se o item foi replicado com êxito em Leste dos EUA (Norte da Virgínia) e Europa (Irlanda).

```
aws dynamodb get-item \
--table-name Music \
--key '{"Artist": {"S":"item_1"}, "SongTitle": {"S":"Song Value 1"}}' \
--region us-east-1
```

```
aws dynamodb get-item \
--table-name Music \
--key '{"Artist": {"S":"item_1"}, "SongTitle": {"S":"Song Value 1"}}' \
--region eu-west-1
```

- Exclua a tabela de réplica na região Europa (Irlanda).

```
aws dynamodb update-table --table-name Music --cli-input-json \
'{'
  "ReplicaUpdates":
```

```
[  
  {  
    "Delete": {  
      "RegionName": "eu-west-1"  
    }  
  }  
]'  
'
```

Criar uma tabela global (Java)

O exemplo de código Java a seguir cria um Music na região Europa (Irlanda), em seguida, cria uma réplica na região da Ásia-Pacífico (Seul).

```
package com.amazonaws.codesamples.gtv2  
import java.util.logging.Logger;  
import com.amazonaws.auth.profile.ProfileCredentialsProvider;  
import com.amazonaws.regions.Regions;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.model.AmazonDynamoDBException;  
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;  
import com.amazonaws.services.dynamodbv2.model.BillingMode;  
import com.amazonaws.services.dynamodbv2.model.CreateReplicationGroupMemberAction;  
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;  
import com.amazonaws.services.dynamodbv2.model.DescribeTableRequest;  
import com.amazonaws.services.dynamodbv2.model.GlobalSecondaryIndex;  
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;  
import com.amazonaws.services.dynamodbv2.model.KeyType;  
import com.amazonaws.services.dynamodbv2.model.Projection;  
import com.amazonaws.services.dynamodbv2.model.ProjectionType;  
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;  
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughputOverride;  
import com.amazonaws.services.dynamodbv2.model.ReplicaGlobalSecondaryIndex;  
import com.amazonaws.services.dynamodbv2.model.ReplicationGroupUpdate;  
import com.amazonaws.services.dynamodbv2.model.ScalarAttributeType;  
import com.amazonaws.services.dynamodbv2.model.StreamSpecification;  
import com.amazonaws.services.dynamodbv2.model.StreamViewType;  
import com.amazonaws.services.dynamodbv2.model.UpdateTableRequest;  
import com.amazonaws.waiters.WaiterParameters;  
  
public class App  
{  
    private final static Logger LOGGER = Logger.getLogger(Logger.GLOBAL_LOGGER_NAME);  
  
    public static void main( String[] args )  
    {  
  
        String tableName = "Music";  
        String indexName = "index1";  
  
        Regions calledRegion = Regions.EU_WEST_1;  
        Regions destRegion = Regions.AP_NORTHEAST_2;  
  
        AmazonDynamoDB ddbClient = AmazonDynamoDBClientBuilder.standard()  
            .withCredentials(new ProfileCredentialsProvider("default"))  
            .withRegion(calledRegion)  
            .build();  
  
        LOGGER.info("Creating a regional table - TableName: " + tableName + ", IndexName: "  
            + indexName + " ....");  
    }  
}
```

```
ddbClient.createTable(new CreateTableRequest()
    .withTableName(tableName)
    .withAttributeDefinitions(
        new AttributeDefinition()

    .withAttributeName("Artist").withAttributeType(ScalarAttributeType.S),
        new AttributeDefinition()

    .withAttributeName("SongTitle").withAttributeType(ScalarAttributeType.S))
        .withKeySchema(
            new
KeySchemaElement().withAttributeName("Artist").withKeyType(KeyType.HASH),
            new
KeySchemaElement().withAttributeName("SongTitle").withKeyType(KeyType.RANGE))
        .withBillingMode(BillingMode.PAY_PER_REQUEST)
        .withGlobalSecondaryIndexes(new GlobalSecondaryIndex()
            .withIndexName(indexName)
            .withKeySchema(new KeySchemaElement()
                .withAttributeName("SongTitle")
                .withKeyType(KeyType.HASH))
            .withProjection(new
Projection().withProjectionType(ProjectionType.ALL)))
        .withStreamSpecification(new StreamSpecification()
            .withStreamEnabled(true)
            .withStreamViewType(StreamViewType.NEW_AND_OLD_IMAGES)));

    LOGGER.info("Waiting for ACTIVE table status .....");
    ddbClient.waiters().tableExists().run(new WaiterParameters<>(new
DescribeTableRequest(tableName)));

    LOGGER.info("Testing parameters for adding a new Replica in " + destRegion +
" ....");

    CreateReplicationGroupMemberAction createReplicaAction = new
CreateReplicationGroupMemberAction()
    .withRegionName(destRegion.getName())
    .withGlobalSecondaryIndexes(new ReplicaGlobalSecondaryIndex()
        .withIndexName(indexName)
        .withProvisionedThroughputOverride(new
ProvisionedThroughputOverride()
            .withReadCapacityUnits(15L))));

    ddbClient.updateTable(new UpdateTableRequest()
        .withTableName(tableName)
        .withReplicaUpdates(new ReplicationGroupUpdate()
            .withCreate(createReplicaAction.withKMSMasterKeyId(null))));

}

}
```

Monitorar as tabelas globais

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte [Determinar a versão \(p. 366\)](#).

Você pode usar o Amazon CloudWatch para monitorar o comportamento e o desempenho de uma tabela global. Amazon DynamoDB publica `ReplicationLatency` para cada réplica na tabela global.

- **ReplicationLatency**— O tempo decorrido entre quando um item é gravado em uma tabela de réplica e quando esse item aparece em outra réplica na tabela global. `ReplicationLatency` é expressa em milissegundos e emitida para cada par de regiões de origem e destino.

Durante o funcionamento normal, `ReplicationLatency` deve ser constante. Um valor elevado para `ReplicationLatency` pode indicar que as atualizações de uma réplica não se propagaram para outras tabelas-réplica em tempo hábil. Com o tempo, isso pode fazer com que outras tabelas-réplica fiquem para trás já que elas não recebem atualizações de forma consistente. Nesse caso, você deve verificar se as unidades de capacidade de leitura (RCUs) e as unidades de capacidade de gravação (WCUs) são idênticas em cada uma das tabelas-réplica. Além disso, ao escolher as configurações da WCU, siga as recomendações em [Práticas recomendadas e requisitos de gerenciamento de capacidade \(p. 368\)](#).

`ReplicationLatency` pode aumentar se uma AWSA região se torna degradada e houver uma tabela-réplica nessa região. Nesse caso, você pode redirecionar temporariamente as atividades de leitura e gravação do seu aplicativo para um diferente AWSRegião :

Para mais informações, consulte [Métricas e dimensões do DynamoDB \(p. 930\)](#).

Como usar o IAM com tabelas globais

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte [Determinar a versão \(p. 366\)](#).

Na primeira vez em que você cria uma tabela global, o Amazon DynamoDB cria automaticamente uma AWS Identity and Access ManagementFunção vinculada ao serviço (IAM) para você. Essa função é nomeada como `AWSServiceRoleForDynamoDBReplication` e permite que o DynamoDB gerencie para você a replicação para tabelas globais entre as regiões. Não exclua essa função vinculada ao serviço. Se fizer isso, todas as suas tabelas globais não funcionarão mais.

Para obter mais informações sobre funções vinculadas a serviços, consulte [Usar funções vinculadas a serviços](#) no Guia do usuário do IAM.

Para criar tabelas-réplica no DynamoDB, é necessário ter as permissões a seguir na região de origem.

- `dynamodb:UpdateTable`

Para criar tabelas-réplica no DynamoDB, é necessário ter as permissões a seguir nas regiões de destino.

- `dynamodb:CreateTable`
- `dynamodb:CreateTableReplica`
- `dynamodb:Scan`
- `dynamodb:Query`
- `dynamodb:UpdateItem`
- `dynamodb:PutItem`
- `dynamodb:GetItem`
- `dynamodb:DeleteItem`
- `dynamodb:BatchWriteItem`

Para excluir tabelas-réplica no DynamoDB, é necessário ter as permissões a seguir nas regiões de destino.

- dynamodb>DeleteTable
- dynamodb>DeleteTableReplica

Para atualizar a política de escalabilidade automática da réplica usando UpdateTableReplicaAutoScaling, é necessário ter as permissões a seguir em todas as regiões onde existam réplicas da tabela.

- application-autoscaling>DeleteScalingPolicy
- application-autoscaling>DeleteScheduledAction
- application-autoscaling>DeregisterScalableTarget
- application-autoscaling>DescribeScalableTargets
- application-autoscaling>DescribeScalingActivities
- application-autoscaling>DescribeScalingPolicies
- application-autoscaling>DescribeScheduledActions
- application-autoscaling>PutScalingPolicy
- application-autoscaling>PutScheduledAction
- application-autoscaling/RegisterScalableTarget

Para usar UpdateTimeToLive, é necessário ter permissão para dynamodb:UpdateTimeToLive em todas as regiões onde existam réplicas da tabela.

Exemplo: Adicionar réplica

A política do IAM a seguir concede permissões para você adicionar réplicas a uma tabela global.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "VisualEditor0",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb>CreateTable",  
                "dynamodb>DescribeTable",  
                "dynamodb>UpdateTable",  
                "dynamodb>CreateTableReplica",  
                "iam>CreateServiceLinkedRole"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

Exemplo: Atualizar política de AutoScaling

A política do IAM a seguir concede permissões para que você atualize a política de auto-scaling da réplica.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "VisualEditor0",  
            "Effect": "Allow",  
            "Action": [  
                "application-autoscaling>DeleteScalingPolicy",  
                "application-autoscaling>DeleteScheduledAction",  
                "application-autoscaling>DeregisterScalableTarget",  
                "application-autoscaling>DescribeScalableTargets",  
                "application-autoscaling>DescribeScalingActivities",  
                "application-autoscaling>DescribeScalingPolicies",  
                "application-autoscaling>DescribeScheduledActions",  
                "application-autoscaling>PutScalingPolicy",  
                "application-autoscaling>PutScheduledAction",  
                "application-autoscaling/RegisterScalableTarget"  
            ]  
        }  
    ]  
}
```

```
        "application-autoscaling:RegisterScalableTarget",
        "application-autoscaling>DeleteScheduledAction",
        "application-autoscaling>DescribeScalableTargets",
        "application-autoscaling>DescribeScalingActivities",
        "application-autoscaling>DescribeScalingPolicies",
        "application-autoscaling>PutScalingPolicy",
        "application-autoscaling>DescribeScheduledActions",
        "application-autoscaling>DeleteScalingPolicy",
        "application-autoscaling>PutScheduledAction",
        "application-autoscaling>DeregisterScalableTarget"
    ],
    "Resource": "*"
}
]
```

Exemplo: Permitir criações de réplicas para um nome e uma região específicos de uma tabela

A política do IAM a seguir concede permissões para a criação de tabelas e de réplicas para a tabela `Customers` com réplicas em três regiões.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "dynamodb>CreateTable",
                "dynamodb>DescribeTable",
                "dynamodb>UpdateTable"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:123456789012:table/Customers",
                "arn:aws:dynamodb:us-west-1:123456789012:table/Customers",
                "arn:aws:dynamodb:eu-east-2:123456789012:table/Customers"
            ]
        }
    ]
}
```

Atualizar tabelas globais para a versão 2019.11.21 (Atual)

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte [Determinar a versão \(p. 366\)](#).

Esta seção descreve como atualizar suas tabelas globais para a versão 2019.11.21 (atual).

Tópicos

- [Antes de começar \(p. 376\)](#)
- [Atualizar para a versão 2019.11.21 \(atual\) \(p. 378\)](#)

Antes de começar

As tabelas globais do DynamoDB versão 2019.11.21 (atual) apresentam os seguintes requisitos:

- Os [índices secundários globais](#) nas tabelas-réplica devem ser consistentes entre regiões.

- A configuração de [criptografia](#) nas tabelas-réplica deve ser consistente em todas as regiões.
- As configurações de [Vida útil \(TTL\)](#) nas tabelas-réplica devem ser consistentes em todas as regiões. Se o TTL estiver habilitado em uma tabela-réplica, as exclusões do TTL serão replicadas.
- Auto Scaling do DynamoDB ou [Sob demanda](#)A capacidade deve estar habilitada para unidades de capacidade de gravação em todas as tabelas de réplica.
- As APIs das operações de plano de controle das tabelas globais (criar, excluir, atualizar e descrever) são diferentes. Para obter mais informações, consulte[Tabelas globais: Replicação em várias regiões com o DynamoDB](#).

As tabelas globais do DynamoDB versão 2019.11.21 (atual) apresentam as seguintes alterações no comportamento:

- O DynamoDB Streams publica apenas um registro (em vez de dois) para cada gravação.
- Para novas inserções, não são adicionados atributos `aws:rep:*` no registro do item.
- Para atualizações de itens que contêm atributos `aws:rep:*`, esses atributos não são atualizados.
- O mapeador do DynamoDB não deve exigir `aws:rep:*`Atributos da tabela.
- Na atualização da versão 2017.11.29 para a versão 2019.11.21, talvez você perceba um aumento na `ReplicationLatency`Métrica do. Isso é esperado porque a versão 2019.11.21 (atual) captura a medição completa do atraso de replicação entre regiões de tabelas globais. Para obter mais informações, consulte o `.ReplicationLatency`documentação para a versão[Versão 2017.11.29](#)e[Versão 2019.11.21 \(atual\)](#).

Permissões obrigatórias

Para atualizar para a versão 2019.11.21 (atual), é necessário ter permissões `dynamodb:UpdateGlobalTableVersion` em todas as regiões de réplica. Essas permissões vão além das permissões necessárias para acessar o console do DynamoDB e visualizar tabelas.

A política do IAM a seguir concede permissões para atualizar qualquer tabela global para a versão 2019.11.21 (atual).

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "dynamodb:UpdateGlobalTableVersion",  
            "Resource": "*"  
        }  
    ]  
}
```

A política do IAM a seguir concede permissões para atualizar apenas a `Music`Tabela global, com réplicas em duas regiões, para a versão 2019.11.21 (atual).

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "dynamodb:UpdateGlobalTableVersion",  
            "Resource": [  
                "arn:aws:dynamodb:123456789012:global-table/Music",  
                "arn:aws:dynamodb:ap-southeast-1:123456789012:table/Music",  
                "arn:aws:dynamodb:us-east-2:123456789012:table/Music"  
            ]  
        }  
    ]  
}
```

}]

Visão geral do processo de atualização

Durante o processo de atualização, o estado da tabela global muda de ACTIVE (ATIVO) para UPDATING (ATUALIZANDO).

O processo leva vários minutos, mas deve ser concluído em menos de uma hora. Durante o processo de atualização, sua tabela permanece disponível para tráfego de leitura e gravação. No entanto, a escalabilidade automática não altera a capacidade provisionada na tabela durante a atualização. Portanto, antes de iniciar a atualização, recomendamos mudar sua tabela para a capacidade [sob demanda](#).

Se você optar por usar a capacidade [provisionada](#) com escalabilidade automática durante a atualização, será necessário aumentar a taxa de transferência mínima de leitura e gravação em suas políticas para acomodar os aumentos esperados no tráfego durante a atualização.

Quando o processo de atualização estiver concluído, o status da tabela retornará a ACTIVE (ATIVO). É possível verificar o estado da tabela usando `DescribeTable`, ou verifique-o usando o [TabelasNo](#) console do DynamoDB.

Important

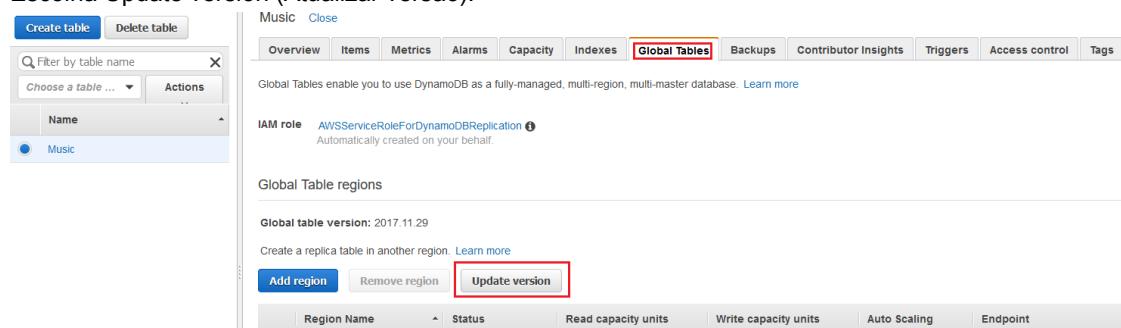
- Atualizar da versão 2017.11.29 para a versão 2019.11.21 (atual) é uma ação única e não pode ser revertida. Antes de prosseguir com a atualização, verifique se você executou todos os testes necessários. Aguarde até 60 minutos antes de tentar atualizar uma tabela global recém-criada.
- Na versão 2017.11.29 de tabelas globais, o DynamoDB executou uma operação de gravação para inserir três atributos no registro do item. Esses atributos (`aws:rep:*`) foram usados para realizar a replicação e gerenciar a resolução de conflitos. Na versão 2019.11.21 (atual) das tabelas globais, a atividade de replicação é gerenciada nativamente e não é exposta aos usuários.
- A atualização para a versão 2019.11.21 (atual) está disponível somente por meio do console do DynamoDB.

Atualizar para a versão 2019.11.21 (atual)

Siga estas etapas para atualizar sua versão das tabelas globais do DynamoDB usando o AWS Management Console.

Como atualizar tabelas globais para a versão 2019.11.21

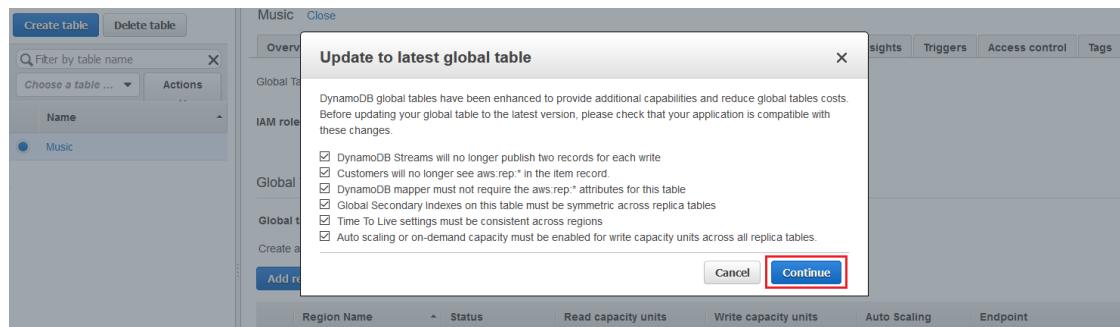
1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/home>.
2. No painel de navegação no lado esquerdo do console, escolha Tables (Tabelas) e selecione a tabela global que você deseja atualizar para 2019.11.21 (atual).
3. Selecione a guia Global Tables (Tabelas globais).
4. Escolha Update version (Atualizar versão).



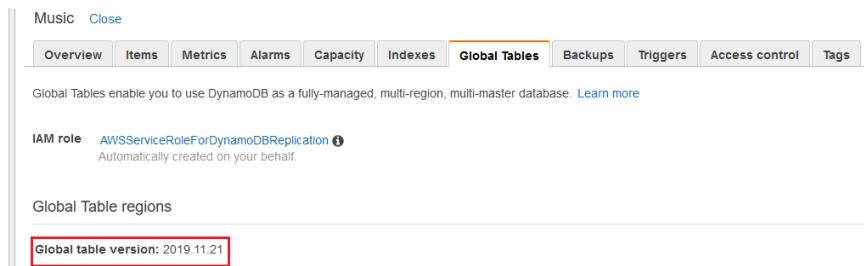
5. Leia e concorde com os novos requisitos e escolha Continuar (Continuar).

Important

Atualizar da versão 2017.11.29 para a versão 2019.11.21 (atual) é uma ação única e não pode ser revertida. Antes de iniciar a atualização, verifique se você executou todos os testes necessários.



6. Após a conclusão do processo de atualização, a versão de tabelas globais exibida no console muda para 2019.11.21.



Versão 2017.11.29

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\)](#) (p. 367) e [Versão 2017.11.29](#) (p. 379). Para descobrir qual versão você está usando, consulte [Determinar a versão](#) (p. 366).

Tópicos

- [Tabelas globais: Como ele funciona](#) (p. 379)
- [Melhores práticas e requisitos do gerenciamento de tabelas globais](#) (p. 381)
- [Como criar uma tabela global](#) (p. 383)
- [Monitorar as tabelas globais](#) (p. 386)
- [Como usar o IAM com tabelas globais](#) (p. 387)

Tabelas globais: Como ele funciona

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\)](#) (p. 367) e [Versão 2017.11.29](#) (p. 379). Para descobrir qual versão você está usando, consulte [Determinar a versão](#) (p. 366).

As seções a seguir ajudam você a entender os conceitos e o comportamento das tabelas globais no Amazon DynamoDB.

Conceitos de tabela global

A tabela global é uma coleção de uma ou mais tabelas de réplica, todas pertencentes a uma única AWS conta.

A tabela de réplica (ou Réplica Em resumo) é uma única tabela do DynamoDB que funciona como parte de uma tabela global. Cada réplica armazena o mesmo conjunto de itens de dados. Qualquer tabela global só pode ter uma tabela de réplica por AWS Região:

Veja a seguir uma visão geral conceitual de como uma tabela global é criada.

1. Crie uma tabela comum do DynamoDB, com o DynamoDB Streams habilitado, em um AWS Região:
2. Repita a etapa 1 para todas as outras regiões em que você quer replicar seus dados.
3. Defina uma tabela global do DynamoDB com base nas tabelas que você criou.

O AWS Management Console automatiza essas tarefas para que você possa criar uma tabela global de forma mais rápida e fácil. Para mais informações, consulte [Como criar uma tabela global \(p. 383\)](#).

A tabela global resultante do DynamoDB consiste em várias tabelas de réplica, uma por região, que o DynamoDB trata como uma única unidade. Cada réplica possui o mesmo nome de tabela e o mesmo esquema de chave primária. Quando um aplicativo grava dados em uma tabela de réplica em uma região, o DynamoDB propaga automaticamente a gravação para as outras tabelas de réplica no outro AWS Regiões.

Important

Para manter os dados da tabela sincronizados, as tabelas globais criam automaticamente os seguintes atributos para cada item:

- `aws:rep:deleting`
- `aws:rep:updatetime`
- `aws:rep:updateregion`

Não modifique esses atributos ou crie atributos com o mesmo nome.

Você pode adicionar tabelas de réplica à tabela global para que ela fique disponível em outras regiões. (Para fazer isso, a tabela global deve estar vazia. Em outras palavras, nenhuma das tabelas de réplica pode conter informações.)

Você também pode remover uma tabela de réplica de uma tabela global. Se você fizer isso, a tabela será completamente desassociada da tabela global. Essa tabela que se tornou independente não interagirá mais com a tabela global e os dados não serão mais propagados de nem para a tabela global.

Consistência e resolução de conflitos

Todas as alterações feitas em qualquer item de qualquer tabela de réplica serão replicadas para todas as outras réplicas dentro da mesma tabela global. Em uma tabela global, um item recém-gravado geralmente é propagado para todas as tabelas de réplica dentro de poucos segundos.

Com uma tabela global, cada tabela de réplica armazena o mesmo conjunto de itens de dados. O DynamoDB não oferece suporte à replicação parcial de apenas alguns dos itens.

Um aplicativo pode ler e gravar dados em qualquer tabela de réplica. Se seu aplicativo usar somente leituras eventualmente consistentes e somente leituras de problemas em um AWS Região, ele funcionará sem qualquer modificação. No entanto, se seu aplicativo exigir leituras muito consistentes, ele precisará

executar todas as suas leituras e gravações muito consistentes na mesma região. O DynamoDB não oferece suporte a leituras altamente consistentes entre regiões. Portanto, se você gravar em uma região e ler em outra região, a resposta lida poderá incluir dados obsoletos que não refletem os resultados das gravações concluídas recentemente na outra região.

Podem surgir conflitos se os aplicativos atualizarem o mesmo item em diferentes regiões e quase no mesmo tempo. Para ajudar a garantir uma consistência eventual, as tabelas globais do DynamoDB usam um último escritor venceReconciliação entre atualizações simultâneas, na qual o DynamoDB faz o melhor esforço para determinar o último gravador. Com esse mecanismo de resolução de conflitos, todas as réplicas concordarão com a atualização mais recente e serão convertidas para um estado em que todas tenham dados idênticos.

Disponibilidade e durabilidade

Se um único AWSA região se tornar isolada ou degradada, seu aplicativo poderá realizar redirecionamentos para uma região diferente e executar leituras e gravações em uma tabela de réplica diferente. Você pode aplicar lógica de negócios personalizada para determinar quando redirecionar solicitações para outras regiões.

Se uma região se tornar isolada ou degradada, o DynamoDB acompanhará as gravações executadas que ainda não foram propagadas para todas as tabelas de réplica. Quando a região ficar online novamente, o DynamoDB retomará a propagação de todas as gravações pendentes dessa região para as tabelas de réplica nas outras regiões. Ele também retoma a propagação de gravações de outras tabelas de réplica para a região que está online novamente.

Melhores práticas e requisitos do gerenciamento de tabelas globais

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte [Determinar a versão \(p. 366\)](#).

Ao usar as tabelas globais do Amazon DynamoDB, você pode replicar os dados da tabela no AWSRegiões. É importante que as tabelas-réplica e índices secundários na tabela global tenham configurações idênticas de capacidade de gravação para garantir a replicação adequada dos dados.

Tópicos

- [Requisitos para adicionar uma nova tabela-réplica \(p. 381\)](#)
- [Práticas recomendadas e requisitos de gerenciamento de capacidade \(p. 382\)](#)

Requisitos para adicionar uma nova tabela-réplica

Se você quiser adicionar uma nova tabela-réplica a uma tabela global, todas estas condições precisarão ser verdadeiras:

- A tabela precisa ter a mesma chave de partição que todas as outras réplicas.
- A tabela deve ter as mesmas configurações de gerenciamento de capacidade de gravação especificadas.
- A tabela precisa ter o mesmo nome que todas as outras réplicas.
- A tabela precisa estar com o DynamoDB Streams ativado, e o fluxo precisa conter as imagens novas e antigas do item.
- Nenhuma das tabelas-réplica novas ou existentes na tabela global pode conter dados.

Se os índices secundários globais forem especificados, as seguintes condições também deverão ser atendidas:

- Os índices secundários globais devem ter o mesmo nome.
- Os índices secundários globais devem ter a mesma chave de partição e chave de classificação (se presente).

Important

As configurações de capacidade de gravação devem ser definidas de maneira consistente em todas as tabelas-réplica das tabelas globais e corresponder aos índices secundários.

Para atualizar as configurações de capacidade de gravação para a tabela global, é altamente recomendável usar o console do DynamoDB ou o `UpdateGlobalTableSettings` operação da API. `UpdateGlobalTableSettings` aplica automaticamente alterações nas configurações de capacidade de gravação para todas as tabelas-réplica e índices secundários correspondentes em uma tabela global. Se você usar as operações `UpdateTable`, `RegisterScalableTarget` ou `PutScalingPolicy`, deverá aplicar a alteração em cada tabela-réplica e índice secundário correspondente individualmente. Para obter mais informações, consulte [UpdateGlobalTableSettings na Referência de API do Amazon DynamoDB](#).

Recomendamos enfaticamente habilitar o Auto Scaling para gerenciar as configurações de capacidade de gravação provisionadas. Se preferir gerenciar as configurações de capacidade de gravação manualmente, você deverá provisionar unidades de capacidade de gravação replicadas iguais para todas as suas tabelas-réplicas. Provisione também unidades de capacidade de gravação replicadas iguais para fazer a correspondência dos índices secundários na tabela global.

Você também precisa ter AWS Identity and Access Management (IAM) permissões. Para mais informações, consulte [Como usar o IAM com tabelas globais \(p. 387\)](#).

Práticas recomendadas e requisitos de gerenciamento de capacidade

Considere o seguinte ao gerenciar configurações de capacidade de tabelas-réplica no DynamoDB.

Usar o DynamoDB Auto Scaling

Usar o Auto Scaling do DynamoDB é a maneira recomendada de gerenciar as configurações de capacidade de throughput para tabelas de réplica que usam o modo provisionado. O Auto Scaling do DynamoDB ajusta automaticamente as unidades de capacidade de leitura (RCUs) e as unidades de capacidade de gravação (WCUs) para cada tabela de réplica com base na carga de trabalho real do aplicativo. Para mais informações, consulte [Como gerenciar a capacidade de throughput automaticamente com o Auto Scaling do DynamoDB \(p. 350\)](#).

Se você criar suas tabelas-réplica usando o AWS Management Console, o Auto Scaling ficará ativado por padrão em cada tabela-réplica com as configurações padrão para o gerenciamento das unidades de capacidade de leitura e de gravação.

Alterações nas configurações de Auto Scaling para uma tabela-réplica ou um índice secundário feitas por meio do console do DynamoDB ou usando o `UpdateGlobalTableSettings` chamada são aplicadas a todas as tabelas-réplica e índices secundários correspondentes na tabela global automaticamente. Essas alterações substituem todas as configurações de Auto Scaling existentes. Isso garante que as configurações de capacidade de gravação provisionadas fiquem consistentes nas tabelas-réplica e nos índices secundários na tabela global. Se usar chamadas `UpdateTable`, `RegisterScalableTarget` ou `PutScalingPolicy`, você deverá aplicar a alteração a cada tabela-réplica e índice secundário correspondente individualmente.

Note

Se o Auto Scaling não satisfizer as alterações de capacidade de seu aplicativo (carga de trabalho imprevisível) ou se você não quiser definir suas configurações (configurações de destino para limite mínimo, máximo ou de utilização), você poderá usar o modo sob demanda para gerenciar a capacidade das tabelas globais. Para mais informações, consulte [Modo sob demanda \(p. 18\)](#).

Se você habilitar modo sob demanda em uma tabela global, seu consumo de unidades de solicitação de gravação replicadas (rWCUs) será consistente com como rWCUs são provisionados. Por exemplo, se você realizar 10 gravações em uma tabela local que é replicada em duas regiões adicionais, você consumirá 60 unidades de solicitações de gravação ($10 + 10 + 10 = 30; 30 \times 2 = 60$). As 60 unidades de solicitação de gravação consumidas incluem a gravação extra consumida pelas tabelas globais Versão 2017.11.29 para atualizar os atributos `aws:rep:deleting`, `aws:rep:updatetime` e `aws:rep:updateregion`.

Gerenciar capacidade manualmente

Se você decidir não usar o Auto Scaling do DynamoDB, deverá definir manualmente as configurações da capacidade de leitura e da capacidade de gravação em cada tabela de réplica e no índice secundário.

As unidades de capacidade de gravação replicadas (rWCUs - Replicated write capacity units) provisionadas em cada tabela-réplica devem ser definidas para o número total de rWCUs necessárias para as gravações de aplicativo em todas as regiões multiplicado por dois. Isso acomoda as gravações do aplicativo que ocorrem na região local e as gravações replicadas do aplicativo vindas de outras regiões. Por exemplo, suponha que você espera 5 gravações por segundo em sua tabela-réplica em Ohio e 5 gravações por segundo em sua tabela-réplica no Norte da Virginia. Nesse caso, você deve provisionar 20 WCUs para cada tabela-réplica ($5 + 5 = 10; 10 \times 2 = 20$).

Para atualizar as configurações de capacidade de gravação para a tabela global, é altamente recomendável usar o console do DynamoDB ou o `UpdateGlobalTableSettings` Operação da API. `UpdateGlobalTableSettings` O aplica automaticamente alterações nas configurações de capacidade de gravação para todas as tabelas-réplica e índices secundários correspondentes em uma tabela global. Se você usar as operações `UpdateTable`, `RegisterScalableTarget` ou `PutScalingPolicy`, deverá aplicar a alteração em cada tabela-réplica e índice secundário correspondente individualmente. Para obter mais informações, consulte [Referência de API do Amazon DynamoDB](#).

Note

Para atualizar as configurações (`UpdateGlobalTableSettings`) para uma tabela global no DynamoDB, você deve ter `odynamodb:UpdateGlobalTable`, `dynamodb:DescribeLimits`, `application-autoscaling:DeleteScalingPolicy`, `eapplication-autoscaling:DeregisterScalableTarget` Permissões Para mais informações, consulte [Como usar o IAM com tabelas globais \(p. 387\)](#).

Como criar uma tabela global

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte [Determinar a versão \(p. 366\)](#).

Esta seção descreve como criar uma tabela global usando o console do Amazon DynamoDB ou a AWS Command Line Interface(AWS CLI).

Tópicos

- [Como criar uma tabela global \(console\) \(p. 383\)](#)
- [Criar uma tabela global \(AWS CLI\) \(p. 384\)](#)

Como criar uma tabela global (console)

Siga estas etapas para criar uma tabela global usando o console. O exemplo a seguir cria uma tabela global com tabelas-réplica nos Estados Unidos e na Europa.

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/home>. Para este exemplo, escolha a região us-east-2 (Leste dos EUA (Ohio)).
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Selecione Create Table (Criar tabela).

Em Table name (Nome da tabela), insira **Music**.

Em Primary key (Chave primária) insira **Artist**. Escolha Add sort key (Adicionar chave de classificação) e insira **SongTitle** (**Artist** e **SongTitle** devem ser strings).

Para criar a tabela, selecione Create (Criar). Essa tabela serve como a primeira tabela-réplica em uma nova tabela global. Ela é o protótipo das outras tabelas-réplica que serão adicionadas posteriormente.

4. Selecione a guia Global Tables (Tabelas globais) e Enable streams (Habilitar streams). Deixe View type (Tipo de visualização) com o valor padrão (imagens novas e antigas).
5. Selecione Adicionar região, e escolha Modo da tabela global 2017.11.29. Agora você pode escolher outra região na qual você deseja implantar outra tabela de réplica. Nesse caso, escolha Oeste dos EUA (Oregon) e, depois, escolha Continuar. Isso inicia o processo de criação da tabela no Oeste dos EUA (Oregon).

O console faz uma verificação para garantir que não haja nenhuma tabela com o mesmo nome na região selecionada. (Se de fato houver uma tabela com o mesmo nome, será necessário excluir a tabela existente para criar uma nova tabela-réplica na região.)

O Tabela global Para a tabela selecionada (e para qualquer outra tabela-réplica) mostrará que a tabela foi replicada em várias regiões.

6. Agora, adicione outra região para que a tabela global seja replicada e sincronizada nos Estados Unidos e na Europa. Para fazer isso, repita a etapa 5, mas, desta vez, especifique UE (Frankfurt) Em vez de Oeste dos EUA (Oregon).
7. Você ainda deve estar usando o AWS Management Console na região us-east-2 (Leste dos EUA (Ohio)). Para a tabela Music (Música), selecione a guia Items (Itens) e escolha Create Item (Criar item). Em Artist (Artista), insira **item_1**. Em SongTitle (Nome da música), insira **Song Value 1**. Para gravar o item, selecione Save (Salvar).

Pouco tempo depois o item será replicado em todas as três regiões da tabela global. Para confirmar isso, no seletor de regiões no canto superior direito do console, selecione Europa (Frankfurt). O Music Tabela na Europa (Frankfurt) deve conter o novo item.

Repita isso para Oeste dos EUA (Oregon).

Criar uma tabela global (AWS CLI)

Siga estas etapas para criar uma tabela global Music usando a AWS CLI. O exemplo a seguir cria uma tabela global com tabelas-réplica nos Estados Unidos e na Europa.

1. Criar uma nova tabela (Music) no Leste dos EUA (Ohio), com os DynamoDB Streams habilitados (NEW_AND_OLD_IMAGES).

```
aws dynamodb create-table \
    --table-name Music \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput \
        ReadCapacityUnits=10,WriteCapacityUnits=5 \
```

```
--stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES \
--region us-east-2
```

- Crie um **Music** Tabela no Leste dos EUA (Norte da Virgínia).

```
aws dynamodb create-table \
--table-name Music \
--attribute-definitions \
  AttributeName=Artist,AttributeType=S \
  AttributeName=SongTitle,AttributeType=S \
--key-schema \
  AttributeName=Artist,KeyType=HASH \
  AttributeName=SongTitle,KeyType=RANGE \
--provisioned-throughput \
  ReadCapacityUnits=10,WriteCapacityUnits=5 \
--stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES \
--region us-east-1
```

- Crie uma tabela global (**Music**) consistindo em tabelas de réplica `nous-east-2`/`us-east-1` Regiões.

```
aws dynamodb create-global-table \
--global-table-name Music \
--replication-group RegionName=us-east-2 RegionName=us-east-1 \
--region us-east-2
```

Note

O nome da tabela global (**Music**) deve corresponder ao nome de cada uma das tabelas de réplica (**Music**). Para mais informações, consulte [Melhores práticas e requisitos do gerenciamento de tabelas globais \(p. 381\)](#).

- Crie outra tabela na Europa (Irlanda), com as mesmas configurações daquela que você criou na etapa 1 e na etapa 2.

```
aws dynamodb create-table \
--table-name Music \
--attribute-definitions \
  AttributeName=Artist,AttributeType=S \
  AttributeName=SongTitle,AttributeType=S \
--key-schema \
  AttributeName=Artist,KeyType=HASH \
  AttributeName=SongTitle,KeyType=RANGE \
--provisioned-throughput \
  ReadCapacityUnits=10,WriteCapacityUnits=5 \
--stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES \
--region eu-west-1
```

Depois de fazer esta etapa, adicione a nova tabela ao **Music** tabela global.

```
aws dynamodb update-global-table \
--global-table-name Music \
--replica-updates 'Create={RegionName=eu-west-1}' \
--region us-east-2
```

- Para verificar se a replicação está funcionando, adicione um novo item à propriedade **Music** tabela na região Leste dos EUA (Ohio).

```
aws dynamodb put-item \
--table-name Music \
--item '{"Artist": {"S":"item_1"}, "SongTitle": {"S":"Song Value 1"}}'
```

```
--region us-east-2
```

6. Aguarde alguns segundos e, em seguida, verifique se o item foi replicado com êxito em Leste dos EUA (Norte da Virgínia) e Europa (Irlanda).

```
aws dynamodb get-item \  
--table-name Music \  
--key '{"Artist": {"S":"item_1"}, "SongTitle": {"S":"Song Value 1"}}' \  
--region us-east-1
```

```
aws dynamodb get-item \  
--table-name Music \  
--key '{"Artist": {"S":"item_1"}, "SongTitle": {"S":"Song Value 1"}}' \  
--region eu-west-1
```

Monitorar as tabelas globais

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte [Determinar a versão \(p. 366\)](#).

Você pode usar o Amazon CloudWatch para monitorar o comportamento e o desempenho de uma tabela global. Amazon DynamoDB publica `ReplicationLatency` e `PendingReplicationCount` para cada réplica na tabela global.

- **ReplicationLatency**— O tempo decorrido entre quando um item atualizado aparece no fluxo do DynamoDB para uma tabela de réplica e quando esse item aparece em outra réplica na tabela global. `ReplicationLatency` é expressa em milissegundos e emitida para cada par de regiões de origem e destino.

Durante o funcionamento normal, `ReplicationLatency` deve ser constante. Um valor elevado para `ReplicationLatency` pode indicar que as atualizações de uma réplica não se propagaram para outras tabelas-réplica em tempo hábil. Com o tempo, isso pode fazer com que outras tabelas-réplica fiquem para trás já que elas não recebem atualizações de forma consistente. Nesse caso, você deve verificar se as unidades de capacidade de leitura (RCUs) e as unidades de capacidade de gravação (WCUs) são idênticas em cada uma das tabelas-réplica. Além disso, ao escolher as configurações da WCU, siga as recomendações em [Práticas recomendadas e requisitos de gerenciamento de capacidade \(p. 382\)](#).

`ReplicationLatency` pode aumentar se um AWS A regional se torna degradada e houver uma tabela-réplica nessa região. Nesse caso, você pode redirecionar temporariamente as atividades de leitura e gravação do seu aplicativo para um diferente AWS Região :

- **PendingReplicationCount**— O número de atualizações de itens que foram gravadas em uma tabela de réplica, mas ainda não foram gravadas em outra réplica na tabela global. `PendingReplicationCount` é expressa em número de itens e emitida para cada par de regiões de origem e destino.

Durante o funcionamento normal, `PendingReplicationCount` deve ser muito baixa. Se `PendingReplicationCount` aumente por períodos prolongados, investigue se as configurações de capacidade de gravação provisionadas das tabelas de réplica são suficientes para a carga de trabalho atual.

`PendingReplicationCount` pode aumentar se um AWS A regional se torna degradada e houver uma tabela-réplica nessa região. Nesse caso, você pode redirecionar temporariamente as atividades de leitura e gravação do seu aplicativo para um diferente AWS Região :

Para mais informações, consulte [Métricas e dimensões do DynamoDB \(p. 930\)](#).

Como usar o IAM com tabelas globais

Há duas versões das tabelas globais do DynamoDB disponíveis: [Versão 2019.11.21 \(atual\) \(p. 367\)](#) e [Versão 2017.11.29 \(p. 379\)](#). Para descobrir qual versão você está usando, consulte [Determinar a versão \(p. 366\)](#).

Na primeira vez em que você cria uma tabela global, o Amazon DynamoDB cria automaticamente uma AWS Identity and Access Management Function vinculada ao serviço (IAM) para você. Essa função é nomeada como `AWSServiceRoleForDynamoDBReplication` e permite que o DynamoDB gerencie para você a replicação para tabelas globais entre as regiões. Não exclua essa função vinculada ao serviço. Se fizer isso, todas as suas tabelas globais não funcionarão mais.

Para obter mais informações sobre funções vinculadas a serviços, consulte [Usar funções vinculadas a serviços](#) no Guia do usuário do IAM.

Para criar e manter tabelas globais no DynamoDB, é preciso ter `dynamodb:CreateGlobalTable` permissão para acessar cada um dos seguintes itens:

- A tabela de réplica que você deseja adicionar.
- Toda réplica existente que já faça parte da tabela global.
- A própria tabela global.

Para atualizar as configurações (`UpdateGlobalTableSettings`) para uma tabela global no DynamoDB, você deve ter `dynamodb:UpdateGlobalTable`, `dynamodb:DescribeLimits`, `application-autoscaling:DeleteScalingPolicy`, `eapplication-autoscaling:DeregisterScalableTarget` permissões.

O `application-autoscaling:DeleteScalingPolicy` e `eapplication-autoscaling:DeregisterScalableTargets` são necessários ao atualizar uma política de dimensionamento existente. Isto é para que o serviço de tabelas globais possa remover a política de dimensionamento antiga antes de anexar a nova política à tabela ou ao índice secundário.

Se você usar uma política do IAM para gerenciar o acesso a uma tabela-réplica, deverá aplicar uma política idêntica a todas as outras réplicas dentro da tabela global. Esse procedimento pode ajudá-lo a manter um modelo de permissões consistente em todas as tabelas-réplica.

Ao usar políticas idênticas do IAM em todas as réplicas em uma tabela global, você pode também evitar a concessão não intencional de acesso de leitura e gravação a seus dados na tabela global. Por exemplo, considere um usuário que tem acesso a uma única réplica em uma tabela global. Se esse usuário puder gravar nessa réplica, o DynamoDB propagará a gravação para todas as outras tabelas-réplica. Na verdade, o usuário poderá gravar (indiretamente) em todas as outras réplicas na tabela global. Essa situação pode ser evitada usando políticas consistentes do IAM em todas as tabelas-réplica.

Exemplo: Permitir a ação `CreateGlobalTable`

Para adicionar uma réplica a uma tabela global, você precisa ter a permissão `dynamodb:CreateGlobalTable` para a tabela global e para cada uma de suas tabelas-réplica.

A política do IAM a seguir concede permissões para o `CreateGlobalTable` em todas as tabelas.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {
```

```
        "Effect": "Allow",
        "Action": ["dynamodb>CreateGlobalTable"],
        "Resource": "*"
    }
]
```

Exemplo: Permitir as ações UpdateGlobalTable, DescribeLimits, Application-AutoScaling:DeleteScalingPolicy e Application-AutoScaling:Cancelar RegisterScalableTarget

Para atualizar as configurações (UpdateGlobalTableSettings) para uma tabela global no DynamoDB, você deve ter o dynamodb:UpdateGlobalTable,dynamodb:DescribeLimits,application-autoscaling:DeleteScalingPolicy, application-autoscaling:DeregisterScalableTargetPermissões

A política do IAM a seguir concede permissões para o UpdateGlobalTableSettingsação em todas as tabelas.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:UpdateGlobalTable",
                "dynamodb:DescribeLimits",
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:DeregisterScalableTarget"
            ],
            "Resource": "*"
        }
    ]
}
```

Exemplo: Permitir a ação CreateGlobalTable para um nome de tabela global específico com réplicas permitidas somente em determinadas regiões

A política do IAM a seguir concede permissões para o CreateGlobalTableAção para criar uma tabela global chamadaCustomerscom réplicas em duas regiões.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb>CreateGlobalTable",
            "Resource": [
                "arn:aws:dynamodb::123456789012:global-table/Customers",
                "arn:aws:dynamodb:us-east-1:123456789012:table/Customers",
                "arn:aws:dynamodb:us-west-1:123456789012:table/Customers"
            ]
        }
    ]
}
```

Adicionar tags e rótulos a recursos

É possível rotular recursos do Amazon DynamoDB usandotags. As tags permitem categorizar recursos de diferentes maneiras, por exemplo, por finalidade, proprietário, ambiente ou outros critérios. As tags podem ajudar a fazer o seguinte:

- Identificar rapidamente um recurso com base nas tags que você atribuiu a ele.
- Consulte [AWSfaturas discriminadas por tags](#).

Note

Todos os índices secundários locais (LSI) e índices secundários globais (GSI) relacionados a tabelas marcadas são rotulados com as mesmas tags automaticamente. Atualmente, o uso de Streams pelo DynamoDB não pode ser marcado.

A marcação é suportada pelo AWSComo o Amazon EC2, o Amazon S3, o DynamoDB e muito mais. Uma marcação eficiente é capaz de fornecer insights de custos, permitindo criar relatórios entre serviços que possuem uma tag específica.

Para começar a usar tags, faça o seguinte:

1. Compreender [Restrições de marcação no DynamoDB \(p. 389\)](#).
2. Criar tags usando [Marcação de recursos no DynamoDB \(p. 389\)](#).
3. Usar o [Relatórios de alocação de custos \(p. 392\)](#) Para acompanhar suas AWS Custos por tag ativa.

Por fim, é recomendável seguir estratégias de marcação ideais. Para obter mais informações, consulte [AWS Estratégias de marcação](#).

Restrições de marcação no DynamoDB

Cada tag consiste em uma chave e um valor, ambos definidos por você. As seguintes restrições são aplicáveis:

- Cada tabela do DynamoDB só pode ter uma tag com a mesma chave. Se você tentar adicionar uma tag existente (mesma chave), o valor da tag existente será atualizado para o novo valor.
- As chaves e os valores de tags diferenciam maiúsculas de minúsculas.
- O comprimento máximo da chave é 128 caracteres Unicode.
- O comprimento máximo do valor é 256 caracteres Unicode.
- Os caracteres permitidos são letras, espaço em branco e números, além dos seguintes caracteres especiais: + - = . _ : /
- O número máximo de tags por recurso é 50.
- Os nomes e valores de tags atribuídos pela AWS recebem automaticamente o `aws:Prefixo`, que não pode ser atribuído. Nomes de tags atribuídos não estão considerados no limite de 50. Nomes de tags atribuídos pelo usuário têm o prefixo `user:` no relatório de alocação de custos.
- Não é possível colocar uma data retroativa na aplicação de uma tag.

Marcação de recursos no DynamoDB

É possível usar o console do Amazon DynamoDB ou o AWS Command Line Interface(AWS CLI) para adicionar, listar, editar ou excluir tags. Em seguida, você pode ativar essas tags definidas pelo usuário para que elas apareçam no console do AWS Billing and Cost Management para o controle da alocação de custos. Para mais informações, consulte [Relatórios de alocação de custos \(p. 392\)](#).

Para edição em massa, também é possível usar o Tag Editor no AWS Management Console. Para obter mais informações, consulte [Trabalhar com o Tag Editor](#).

Para usar a API do DynamoDB, consulte as operações a seguir no [Referência de API do Amazon DynamoDB](#):

- [TagResource](#)
- [UntagResource](#)
- [ListTagsOfResource](#)

Tópicos

- [Definindo permissões para filtrar por tags \(p. 390\)](#)
- [Adicionando tags a tabelas novas ou existentes \(AWS Management Console\) \(p. 390\)](#)
- [Adicionando tags a tabelas novas ou existentes \(AWS CLI\) \(p. 391\)](#)

Definindo permissões para filtrar por tags

Para usar tags para filtrar sua lista de tabelas no console do DynamoDB, certifique-se de que as políticas de usuário do IAM incluem acesso às seguintes operações:

- `tag:GetTagKeys`
- `tag:GetTagValues`

Você pode acessar essas operações anexando uma nova política do IAM ao usuário do IAM seguindo as etapas abaixo.

1. Acesse a[Console do IAM](#)Com um usuário administrador.
2. Selecione “Políticas” no menu de navegação esquerdo do.
3. Selecione “Criar política”.
4. Cole a política a seguir no editor JSON.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "tag:GetTagKeys",  
                "tag:GetTagValues"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

5. Conclua o assistente e atribua um nome à política (por exemplo,`TagKeysAndValuesReadAccess`).
6. Selecione “Usuários” no menu de navegação esquerdo.
7. Na lista, selecione o usuário que você normalmente usa para acessar o console do DynamoDB.
8. Selecione “Adicionar permissões”.
9. Selecione “Associe políticas existentes diretamente”.
10. Na lista, selecione a política que você criou anteriormente.
11. Assista todo o assistente.

Adicionando tags a tabelas novas ou existentes (AWS Management Console)

Você pode usar o console do DynamoDB para adicionar tags a novas tabelas ao criá-las, ou para adicionar, editar ou excluir tags de tabelas existentes.

Para marcar recursos na criação (console)

1. Faça login no AWS Management Console e abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, selecione Tables (Tabelas) e Create table (Criar tabela).
3. Na página Create DynamoDB table (Criar tabela DynamoDB), forneça um nome e uma chave primária. Selecione Adicionar tag e insira as tags que você deseja usar.

Para obter informações sobre a estrutura da tag, consulte [Restrições de marcação no DynamoDB \(p. 389\)](#).

Para obter mais informações sobre como criar tabelas, consulte [Operações básicas nas tabelas do DynamoDB \(p. 340\)](#).

Para marcar recursos existentes (console)

Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.

1. No painel de navegação, selecione Tables (Tabelas).
2. Selecione uma tabela na lista e selecione a guia Tags para adicionar, editar ou excluir suas tags.

Adicionando tags a tabelas novas ou existentes (AWS CLI)

Os exemplos a seguir mostram como usar a AWS CLI para especificar tags ao criar tabelas e índices e para marcar recursos existentes.

Para marcar recursos na criação (AWS CLI)

- O exemplo a seguir cria uma nova tabela `Movies` e adiciona a tag `Owner` com um valor de `blueTeam`:

```
aws dynamodb create-table \
--table-name Movies \
--attribute-definitions AttributeName=Title,AttributeType=S \
--key-schema AttributeName=Title,KeyType=HASH \
--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
--tags Key=Owner,Value=blueTeam
```

Para marcar recursos existentes (AWS CLI)

- O exemplo a seguir adiciona a tag `Owner` com um valor de `blueTeam` à tabela `Movies`:

```
aws dynamodb tag-resource \
--resource-arn arn:aws:dynamodb:us-east-1:123456789012:table/Movies \
--tags Key=Owner,Value=blueTeam
```

Para listar todas as tags de uma tabela (AWS CLI)

- O exemplo a seguir relaciona todas as tags associadas à tabela `Movies`.

```
aws dynamodb list-tags-of-resource \
--resource-arn arn:aws:dynamodb:us-east-1:123456789012:table/Movies
```

Relatórios de alocação de custos

AWS usa tags para organizar os custos de recursos no seu relatório de alocação de custos. AWS fornece dois tipos de tags alocação de custos:

- Uma AWS-tag gerada. AWS define, cria e aplica essa tag para você.
- Tags definidas pelo usuário. Você define, cria e aplica essas tags.

É necessário ativar os dois tipos de tags separadamente para que elas possam ser exibidas no Explorador de custos ou em um relatório de alocação de custos.

Para ativar o AWSTags geradas:

1. Faça login no AWS Management Consolee abra o console Billing and Cost Management em <https://console.aws.amazon.com/billing/home#/>.
2. No painel de navegação, escolha Cost Allocation Tags.
3. Under AWSTags de alocação de custos geradas pela, escolha Ativar.

Para ativar tags definidas pelo usuário:

1. Faça login no AWS Management Consolee abra o console Billing and Cost Management em <https://console.aws.amazon.com/billing/home#/>.
2. No painel de navegação, escolha Cost Allocation Tags.
3. Em Tags de alocação de custos definidos pelo usuário, escolha Ativar.

Depois de criar e ativar tags, AWS gera um relatório de alocação de custos com a utilização e os custos agrupados de acordo com as tags ativas. O relatório de alocação de custos inclui todas as suas AWS Custos para cada período de faturamento. O relatório inclui recursos com e sem tags, para que você possa organizar claramente as cobranças de cada um deles.

Note

Atualmente, os dados transferidos do DynamoDB não são categorizados por tags em relatórios de alocação de custos.

Para obter mais informações, consulte [Como usar tags de alocação de custos](#).

Como trabalhar com tabelas do DynamoDB em Java

Você pode usar o AWS SDK for Java Para criar, atualizar e excluir tabelas do Amazon DynamoDB, listar todas as tabelas em sua conta ou obter informações sobre uma tabela específica.

Tópicos

- [Criação de uma tabela \(p. 393\)](#)
- [Atualização de uma tabela \(p. 394\)](#)
- [Exclusão de uma tabela \(p. 394\)](#)
- [Listagem de tabelas \(p. 395\)](#)
- [Exemplo: Crie, atualize, exclua e liste tabelas usando o AWS SDK for Java API do Documento \(p. 395\)](#)

Criação de uma tabela

Para criar uma tabela, você deve fornecer o nome da tabela, a chave primária e os valores de throughput provisionado. O trecho de código a seguir cria um exemplo de tabela que usa um ID de atributo do tipo numérico como sua chave primária.

Para criar uma tabela usando a API do AWS SDK for Java

1. Crie uma instância da classe `DynamoDB`.
2. Criar uma instância de `CreateTableRequest` para fornecer as informações.

Você deve fornecer o nome da tabela, as definições de atributo, o esquema de chaves e os valores de throughput provisionado.

3. Execute o método `createTable` fornecendo o objeto de solicitação como um parâmetro.

O exemplo de código a seguir demonstra as etapas anteriores.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

List<AttributeDefinition> attributeDefinitions= new ArrayList<AttributeDefinition>();
attributeDefinitions.add(new
    AttributeDefinition().withAttributeName("Id").withAttributeType("N"));

List<KeySchemaElement> keySchema = new ArrayList<KeySchemaElement>();
keySchema.add(new KeySchemaElement().withAttributeName("Id").withKeyType(KeyType.HASH));

CreateTableRequest request = new CreateTableRequest()
    .withTableName(tableName)
    .withKeySchema(keySchema)
    .withAttributeDefinitions(attributeDefinitions)
    .withProvisionedThroughput(new ProvisionedThroughput()
        .withReadCapacityUnits(5L)
        .withWriteCapacityUnits(6L));

Table table = dynamoDB.createTable(request);

table.waitForActive();
```

A tabela não está pronta para uso até que o DynamoDB a crie e defina seu status como ATIVO. A solicitação `createTable` retorna um objeto `Table` que você pode usar para obter mais informações sobre a tabela.

Example

```
TableDescription tableDescription =
    dynamoDB.getTable(tableName).describe();

System.out.printf("%s: %s \t ReadCapacityUnits: %d \t WriteCapacityUnits: %d",
    tableDescription.getTableStatus(),
    tableDescription.getTableName(),
    tableDescription.getProvisionedThroughput().getReadCapacityUnits(),
    tableDescription.getProvisionedThroughput().getWriteCapacityUnits());
```

Você pode chamar o método `describe` do cliente para obter informações da tabela a qualquer momento.

Example

```
TableDescription tableDescription = dynamoDB.getTable(tableName).describe();
```

Atualização de uma tabela

Você pode atualizar apenas os valores de throughput provisionado de uma tabela existente. Dependendo das necessidades de seu aplicativo, talvez você precise atualizar esses valores.

Note

Para obter mais informações sobre aumentos e diminuições da taxa de transferência por dia, consulte [Cotas de serviço, conta e tabela no Amazon DynamoDB \(p. 1056\)](#).

Para atualizar uma tabela usando a API do AWS SDK for Java

1. Crie uma instância da classe `Table`.
2. Crie uma instância da classe `ProvisionedThroughput` para fornecer os novos valores de throughput provisionado.
3. Execute `updateTable` fornecendo o método `ProvisionedThroughputComo` um parâmetro.

O exemplo de código a seguir demonstra as etapas anteriores.

Example

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("ProductCatalog");

ProvisionedThroughput provisionedThroughput = new ProvisionedThroughput()
    .withReadCapacityUnits(15L)
    .withWriteCapacityUnits(12L);

table.updateTable(provisionedThroughput);

table.waitForActive();
```

Exclusão de uma tabela

Para excluir uma tabela usando a API do AWS SDK for Java

1. Crie uma instância da classe `Table`.
2. Crie uma instância da classe `DeleteTableRequest` e forneça o nome da tabela que deseja excluir.
3. Execute `deleteTable` fornecendo o método `TableComo` um parâmetro.

O exemplo de código a seguir demonstra as etapas anteriores.

Example

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("ProductCatalog");

table.delete();

table.waitForDelete();
```

Listagem de tabelas

Para listar tabelas em sua conta, crie uma instância de `DynamoDB` e execute o `listTables` método do `OListTables`. A operação não requer parâmetros.

Example

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

TableCollection<ListTablesResult> tables = dynamoDB.listTables();
Iterator<Table> iterator = tables.iterator();

while (iterator.hasNext()) {
    Table table = iterator.next();
    System.out.println(table.getTableName());
}
```

Exemplo: Crie, atualize, exclua e liste tabelas usando o AWS SDK for Java API do Documento

O exemplo de código a seguir usa o AWS SDK for Java API de documento para criar, atualizar e excluir uma tabela do Amazon DynamoDB (`ExampleTable`). Como parte da atualização da tabela, ele aumenta os valores de throughput provisionado. O exemplo também lista todas as tabelas em sua conta e obtém a descrição de uma tabela específica. Para obter instruções detalhadas sobre como executar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples.document;

import java.util.ArrayList;
import java.util.Iterator;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.document.TableCollection;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.ListTablesResult;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.services.dynamodbv2.model.TableDescription;
```

```
public class DocumentAPITableExample {

    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
    static DynamoDB dynamoDB = new DynamoDB(client);

    static String tableName = "ExampleTable";

    public static void main(String[] args) throws Exception {

        createExampleTable();
        listMyTables();
        getTableInformation();
        updateExampleTable();

        deleteExampleTable();
    }

    static void createExampleTable() {

        try {

            List<AttributeDefinition> attributeDefinitions = new
ArrayList<AttributeDefinition>();
            attributeDefinitions.add(new
AttributeDefinition().withAttributeName("Id").withAttributeType("N"));

            List<KeySchemaElement> keySchema = new ArrayList<KeySchemaElement>();
            keySchema.add(new
KeySchemaElement().withAttributeName("Id").withKeyType(KeyType.HASH)); // Partition

            // key

            CreateTableRequest request = new
CreateTableRequest().withTableName(tableName).withKeySchema(keySchema)
                .withAttributeDefinitions(attributeDefinitions).withProvisionedThroughput(
                    new
ProvisionedThroughput().withReadCapacityUnits(5L).withWriteCapacityUnits(6L));

            System.out.println("Issuing CreateTable request for " + tableName);
            Table table = dynamoDB.createTable(request);

            System.out.println("Waiting for " + tableName + " to be created...this may take
a while...");
            table.waitForActive();

            getTableInformation();

        }
        catch (Exception e) {
            System.err.println("CreateTable request failed for " + tableName);
            System.err.println(e.getMessage());
        }
    }

    static void listMyTables() {

        TableCollection<ListTablesResult> tables = dynamoDB.listTables();
        Iterator<Table> iterator = tables.iterator();

        System.out.println("Listing table names");

        while (iterator.hasNext()) {
            Table table = iterator.next();
            System.out.println(table.getTableName());
        }
    }
}
```

```
}

static void getTableInformation() {

    System.out.println("Describing " + tableName);

    TableDescription tableDescription = dynamoDB.getTable(tableName).describe();
    System.out.format(
        "Name: %s:\n" + "Status: %s \n" + "Provisioned Throughput (read capacity units/
sec): %d \n"
        + "Provisioned Throughput (write capacity units/sec): %d \n",
        tableDescription.getTableName(), tableDescription.getTableStatus(),
        tableDescription.getProvisionedThroughput().getReadCapacityUnits(),
        tableDescription.getProvisionedThroughput().getWriteCapacityUnits());
}

static void updateExampleTable() {

    Table table = dynamoDB.getTable(tableName);
    System.out.println("Modifying provisioned throughput for " + tableName);

    try {
        table.updateTable(new
ProvisionedThroughput().withReadCapacityUnits(6L).withWriteCapacityUnits(7L));

        table.waitForActive();
    }
    catch (Exception e) {
        System.err.println("UpdateTable request failed for " + tableName);
        System.err.println(e.getMessage());
    }
}

static void deleteExampleTable() {

    Table table = dynamoDB.getTable(tableName);
    try {
        System.out.println("Issuing DeleteTable request for " + tableName);
        table.delete();

        System.out.println("Waiting for " + tableName + " to be deleted...this may take
a while...");

        table.waitForDelete();
    }
    catch (Exception e) {
        System.err.println("DeleteTable request failed for " + tableName);
        System.err.println(e.getMessage());
    }
}
}
```

Como trabalhar com tabelas do DynamoDB no .NET

Você pode usar o AWS SDK for .NET para criar, atualizar e excluir tabelas, listar todas as tabelas em sua conta ou obter informações sobre uma tabela específica.

Veja a seguir as etapas comuns para operações de tabela do Amazon DynamoDB usando oAWS SDK for .NET.

1. Crie uma instância da classe `AmazonDynamoDBClient` (o cliente).

2. Forneça os parâmetros obrigatórios e opcionais para a operação, criando os objetos de solicitação correspondentes.

Por exemplo, crie um objeto `CreateTableRequest` para criar uma tabela e o objeto `UpdateTableRequest` para atualizar uma tabela existente.

3. Execute o método apropriado fornecido pelo cliente que você criou na etapa anterior.

Note

Os exemplos desta seção não funcionam com o .NET core, porque ele não oferece suporte a métodos síncronos. Para obter mais informações, consulte [AWS APIs assíncronas da para .NET](#).

Tópicos

- [Criação de uma tabela \(p. 398\)](#)
- [Atualização de uma tabela \(p. 399\)](#)
- [Exclusão de uma tabela \(p. 400\)](#)
- [Listagem de tabelas \(p. 400\)](#)
- [Exemplo: Crie, atualize, exclua e liste tabelas usando o AWS SDK for .NET API de baixo nível \(p. 401\)](#)

Criação de uma tabela

Para criar uma tabela, você deve fornecer o nome da tabela, a chave primária e os valores de throughput provisionado.

Para criar uma tabela usando a API de baixo nível do AWS SDK for .NET

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Crie uma instância da classe `CreateTableRequest` para fornecer as informações solicitadas. Você deve fornecer o nome da tabela, a chave primária e os valores de throughput provisionado.
3. Execute a `AmazonDynamoDBClient.CreateTable` Método do fornecendo o objeto de solicitação como um parâmetro.

O exemplo de C# a seguir demonstra as etapas anteriores. O exemplo cria uma tabela (`ProductCatalog`) que usa `Id` como a chave primária e o conjunto de valores de taxa de transferência provisionada. Dependendo das necessidades de seu aplicativo, você pode atualizar os valores de throughput provisionado usando a API `UpdateTable`.

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "ProductCatalog";

var request = new CreateTableRequest
{
    TableName = tableName,
    AttributeDefinitions = new List<AttributeDefinition>()
    {
        new AttributeDefinition
        {
            AttributeName = "Id",
            AttributeType = "N"
        }
    },
    KeySchema = new List<KeySchemaElement>()
    {
        new KeySchemaElement
        {
```

```
    AttributeName = "Id",
    KeyType = "HASH" //Partition key
}
},
ProvisionedThroughput = new ProvisionedThroughput
{
    ReadCapacityUnits = 10,
    WriteCapacityUnits = 5
}
};

var response = client.CreateTable(request);
```

Você deve aguardar até que o DynamoDB crie a tabela e defina seu status como ACTIVE. A resposta `CreateTable` inclui a propriedade `TableDescription` que fornece as informações necessárias da tabela.

Example

```
var result = response.CreateTableResult;
var tableDescription = result.TableDescription;
Console.WriteLine("{1}: {0} \t ReadCapacityUnits: {2} \t WriteCapacityUnits: {3}",
    tableDescription.TableStatus,
    tableDescription.TableName,
    tableDescription.ProvisionedThroughput.ReadCapacityUnits,
    tableDescription.ProvisionedThroughput.WriteCapacityUnits);

string status = tableDescription.TableStatus;
Console.WriteLine(tableName + " - " + status);
```

Você também pode chamar o método `DescribeTable` do cliente para obter informações da tabela a qualquer momento.

Example

```
var res = client.DescribeTable(new DescribeTableRequest{TableName = "ProductCatalog"});
```

Atualização de uma tabela

Você pode atualizar apenas os valores de throughput provisionado de uma tabela existente. Dependendo das necessidades de seu aplicativo, talvez você precise atualizar esses valores.

Note

Você pode aumentar a capacidade da taxa de transferência com a frequência necessária e diminuí-la dentro de determinadas restrições. Para obter mais informações sobre aumentos e diminuições da taxa de transferência por dia, consulte [Cotas de serviço, conta e tabela no Amazon DynamoDB \(p. 1056\)](#).

Para atualizar uma tabela usando a API de baixo nível do AWS SDK for .NET

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Crie uma instância da classe `UpdateTableRequest` para fornecer as informações solicitadas.
 - Você deve fornecer o nome da tabela e os novos valores de throughput provisionado.
3. Execute a `AmazonDynamoDBClient.UpdateTable` Método do fornecendo o objeto de solicitação como um parâmetro.

O exemplo de C# a seguir demonstra as etapas anteriores.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "ExampleTable";

var request = new UpdateTableRequest()
{
    TableName = tableName,
    ProvisionedThroughput = new ProvisionedThroughput()
    {
        // Provide new values.
        ReadCapacityUnits = 20,
        WriteCapacityUnits = 10
    }
};
var response = client.UpdateTable(request);
```

Exclusão de uma tabela

Siga estas etapas para excluir uma tabela usando a API de baixo nível do .NET.

Para excluir uma tabela usando a API de baixo nível do AWS SDK for .NET.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Crie uma instância da classe `DeleteTableRequest` e forneça o nome da tabela que deseja excluir.
3. Execute a `AmazonDynamoDBClient.DeleteTable` Método do fornecendo o objeto de solicitação como um parâmetro.

O exemplo de código C# a seguir demonstra as etapas anteriores.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "ExampleTable";

var request = new DeleteTableRequest{ TableName = tableName };
var response = client.DeleteTable(request);
```

Listagem de tabelas

Para listar tabelas na conta usando o AWS SDK for .NET API de baixo nível do, crie uma instância `AmazonDynamoDBClient` execute a `ListTables` Método do.

O `ListTables` A operação não requer parâmetros. No entanto, você pode especificar parâmetros opcionais. Por exemplo, você pode definir o parâmetro `Limit` se desejar usar paginação para limitar o número de nomes de tabela por página. Isso requer que você crie um objeto `ListTablesRequest` e forneça parâmetros opcionais, conforme mostrado no seguinte exemplo de C#. Junto com o tamanho da página, a solicitação define o parâmetro `ExclusiveStartTableName`. Inicialmente, `ExclusiveStartTableName` é nulo. No entanto, após a busca da primeira página de resultados, para recuperar a próxima página de resultados, você deve definir esse valor de parâmetro como a propriedade `LastEvaluatedTableName` do resultado atual.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
```

```
// Initial value for the first page of table names.  
string lastEvaluatedTableName = null;  
do  
{  
    // Create a request object to specify optional parameters.  
    var request = new ListTablesRequest  
    {  
        Limit = 10, // Page size.  
        ExclusiveStartTableName = lastEvaluatedTableName  
    };  
  
    var response = client.ListTables(request);  
    ListTablesResult result = response.ListTablesResult;  
    foreach (string name in result.TableNames)  
        Console.WriteLine(name);  
  
    lastEvaluatedTableName = result.LastEvaluatedTableName;  
} while (lastEvaluatedTableName != null);
```

Exemplo: Crie, atualize, exclua e liste tabelas usando oAWS SDK for .NETAPI de baixo nível

O exemplo de código C# a seguir cria, atualiza e exclui uma tabela (`ExampleTable`). Ele também lista todas as tabelas em sua conta e obtém a descrição de uma tabela específica. A atualização da tabela aumenta os valores de throughput provisionado. Para ver as instruções passo a passo para testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
using System;  
using System.Collections.Generic;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.Model;  
using Amazon.Runtime;  
  
namespace com.amazonaws.codesamples  
{  
    class LowLevelTableExample  
    {  
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();  
        private static string tableName = "ExampleTable";  
  
        static void Main(string[] args)  
        {  
            try  
            {  
                CreateExampleTable();  
                ListMyTables();  
                GetTableInformation();  
                UpdateExampleTable();  
            }  
        }  
    }  
}
```

```
        DeleteExampleTable();

        Console.WriteLine("To continue, press Enter");
        Console.ReadLine();
    }
    catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message); }
    catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
    catch (Exception e) { Console.WriteLine(e.Message); }
}

private static void CreateExampleTable()
{
    Console.WriteLine("\n*** Creating table ***");
    var request = new CreateTableRequest
    {
        AttributeDefinitions = new List<AttributeDefinition>()
    {
        new AttributeDefinition
        {
            AttributeName = "Id",
            AttributeType = "N"
        },
        new AttributeDefinition
        {
            AttributeName = "ReplyDateTime",
            AttributeType = "N"
        }
    },
    KeySchema = new List<KeySchemaElement>
    {
        new KeySchemaElement
        {
            AttributeName = "Id",
            KeyType = "HASH" //Partition key
        },
        new KeySchemaElement
        {
            AttributeName = "ReplyDateTime",
            KeyType = "RANGE" //Sort key
        }
    },
    ProvisionedThroughput = new ProvisionedThroughput
    {
        ReadCapacityUnits = 5,
        WriteCapacityUnits = 6
    },
    TableName = tableName
};

    var response = client.CreateTable(request);

    var tableDescription = response.TableDescription;
    Console.WriteLine("{1}: {0} \t ReadsPerSec: {2} \t WritesPerSec: {3}",
        tableDescription.TableStatus,
        tableDescription.TableName,
        tableDescription.ProvisionedThroughput.ReadCapacityUnits,
        tableDescription.ProvisionedThroughput.WriteCapacityUnits);

    string status = tableDescription.TableStatus;
    Console.WriteLine(tableName + " - " + status);

    WaitUntilTableReady(tableName);
}

private static void ListMyTables()
```

```
{  
    Console.WriteLine("\n*** listing tables ***");  
    string lastTableNameEvaluated = null;  
    do  
    {  
        var request = new ListTablesRequest  
        {  
            Limit = 2,  
            ExclusiveStartTableName = lastTableNameEvaluated  
        };  
  
        var response = client.ListTables(request);  
        foreach (string name in response.TableNames)  
            Console.WriteLine(name);  
  
        lastTableNameEvaluated = response.LastEvaluatedTableName;  
    } while (lastTableNameEvaluated != null);  
}  
  
private static void GetTableInformation()  
{  
    Console.WriteLine("\n*** Retrieving table information ***");  
    var request = new DescribeTableRequest  
    {  
        TableName = tableName  
    };  
  
    var response = client.DescribeTable(request);  
  
    TableDescription description = response.Table;  
    Console.WriteLine("Name: {0}", description.TableName);  
    Console.WriteLine("# of items: {0}", description.ItemCount);  
    Console.WriteLine("Provision Throughput (reads/sec): {0}",  
        description.ProvisionedThroughput.ReadCapacityUnits);  
    Console.WriteLine("Provision Throughput (writes/sec): {0}",  
        description.ProvisionedThroughput.WriteCapacityUnits);  
}  
  
private static void UpdateExampleTable()  
{  
    Console.WriteLine("\n*** Updating table ***");  
    var request = new UpdateTableRequest()  
    {  
        TableName = tableName,  
        ProvisionedThroughput = new ProvisionedThroughput()  
        {  
            ReadCapacityUnits = 6,  
            WriteCapacityUnits = 7  
        };  
    };  
  
    var response = client.UpdateTable(request);  
  
    WaitUntilTableReady(tableName);  
}  
  
private static void DeleteExampleTable()  
{  
    Console.WriteLine("\n*** Deleting table ***");  
    var request = new DeleteTableRequest  
    {  
        TableName = tableName  
    };  
  
    var response = client.DeleteTable(request);
```

```
        Console.WriteLine("Table is being deleted...");
    }

    private static void WaitUntilTableReady(string tableName)
    {
        string status = null;
        // Let us wait until table is created. Call DescribeTable.
        do
        {
            System.Threading.Thread.Sleep(5000); // Wait 5 seconds.
            try
            {
                var res = client.DescribeTable(new DescribeTableRequest
                {
                    TableName = tableName
                });

                Console.WriteLine("Table name: {0}, status: {1}",
                    res.Table.TableName,
                    res.Table.TableStatus);
                status = res.Table.TableStatus;
            }
            catch (ResourceNotFoundException)
            {
                // DescribeTable is eventually consistent. So you might
                // get resource not found. So we handle the potential exception.
            }
        } while (status != "ACTIVE");
    }
}
```

Trabalho com itens e atributos

No Amazon DynamoDB, um item é uma coleção de atributos. Cada atributo tem um nome e um valor. Um valor de atributo pode ser escalar, um conjunto ou um tipo de documento. Para mais informações, consulte [Amazon DynamoDB: Como ele funciona \(p. 2\)](#).

O DynamoDB oferece quatro operações para a funcionalidade básica create, read, update e delete (CRUD - criação, leitura, atualização e exclusão):

- **PutItem**— Cria um item.
- **GetItem**— Ler um item.
- **UpdateItem**— Atualiza um item.
- **DeleteItem**— Excluir um item.

Cada uma dessas operações exige que você especifique a chave primária do item com o qual deseja trabalhar. Por exemplo, para ler um item usando **GetItem**, você deve especificar a chave de partição e a chave de classificação (se aplicável) desse item.

Além das quatro operações CRUD básicas, o DynamoDB também fornece o seguinte:

- **BatchGetItem**— ler até 100 itens de uma ou mais tabelas.
- **BatchWriteItem**— criar ou excluir até 25 itens em uma ou mais tabelas.

Essas operações em lote combinam várias operações CRUD em uma única solicitação. Além disso, as operações em lote leem e gravam itens em paralelo para reduzir as latências de resposta.

Esta seção descreve como usar essas operações e inclui tópicos relacionados, como contadores atômicos e atualizações condicionais. Esta seção também inclui código de exemplo que usa o AWSSDKs.

Tópicos

- [Leitura de um item \(p. 405\)](#)
- [Gravação de um item \(p. 406\)](#)
- [Valores de retorno \(p. 407\)](#)
- [Operações em lote \(p. 408\)](#)
- [Contadores atômicos \(p. 410\)](#)
- [Gravações condicionais \(p. 410\)](#)
- [Uso de expressões no DynamoDB \(p. 414\)](#)
- [Itens expirando usando o Time to Live \(TL — Time to Live — uso do DynamoDB \(p. 439\)](#)
- [Trabalho com itens: Java \(p. 446\)](#)
- [Trabalho com itens: .NET \(p. 466\)](#)

Leitura de um item

Para ler um item em uma tabela do DynamoDB, use a função `GetItem` operação. Você deve fornecer o nome da tabela, juntamente com a chave primária do item desejado.

Example

O exemplo da AWS CLI a seguir mostra como ler um item da tabela `ProductCatalog`.

```
aws dynamodb get-item \  
  --table-name ProductCatalog \  
  --key '{"Id":{"N":"1"}}'
```

Note

Com `GetItem`, você deve especificar a chave primária inteira, não apenas parte dela. Por exemplo, se uma tabela tiver uma chave primária composta (chave de partição e chave de classificação), você deve fornecer um valor para a chave de partição e um valor para a chave de classificação.

A solicitação de `GetItem` executa uma leitura eventualmente consistente, por padrão. Você pode usar o parâmetro `ConsistentRead` para solicitar uma leitura fortemente consistente. (Isso consome unidades de capacidade de leitura adicionais, mas retorna a versão mais recente do item.)

`GetItem` retornará todos os atributos do item. Você pode usar uma expressão de projeção para retornar apenas alguns dos atributos. Para mais informações, consulte [Expressões de projeção \(p. 417\)](#).

Para retornar o número de unidades de capacidade de leitura consumidas por `GetItem`, defina o parâmetro `ReturnConsumedCapacity` como `TOTAL`.

Example

O exemplo da AWS Command Line Interface (AWS CLI) a seguir mostra alguns dos parâmetros opcionais de `GetItem`.

```
aws dynamodb get-item \  
  --table-name ProductCatalog \  
  --key '{"Id":{"N":"1"}}' \  
  --consistent-read \  
  --projection-expression "Description, Price, RelatedItems" \  
  --return-consumed-capacity TOTAL
```

Gravação de um item

Para criar, atualizar ou excluir um item em uma tabela do DynamoDB, use uma das seguintes operações:

- `PutItem`
- `UpdateItem`
- `DeleteItem`

Para cada uma dessas operações, você deve especificar a chave primária inteira, não apenas parte dela. Por exemplo, se uma tabela tiver uma chave primária composta (chave de partição e chave de classificação), você deve fornecer um valor para a chave de partição e um valor para a chave de classificação.

Para retornar o número de unidades de capacidade de gravação consumidas por qualquer uma dessas operações, defina o parâmetro `ReturnConsumedCapacity` de uma das seguintes formas:

- `TOTAL`— retorna o número total de unidades de capacidade de gravação consumidas.
- `INDEXES`— retorna o número total de unidades de capacidade de gravação consumidas, com subtotais para a tabela e para todos os índices secundários que foram afetados pela operação.
- `NONE`— nenhum detalhe da capacidade de gravação é retornado. (Esse é o padrão.)

PutItem

`PutItem` cria um novo item. Se um item com a mesma chave já existe na tabela, ele é substituído pelo novo item.

Example

Gravar um novo item na tabela `Thread`. A chave primária de `Thread` consiste em `ForumName` (chave de partição) e `Subject` (chave de classificação).

```
aws dynamodb put-item \
  --table-name Thread \
  --item file://item.json
```

Os argumentos de `--item` são armazenados no arquivo `item.json`.

```
{
    "ForumName": {"S": "Amazon DynamoDB"},
    "Subject": {"S": "New discussion thread"},
    "Message": {"S": "First post in this thread"},
    "LastPostedBy": {"S": "fred@example.com"},
    "LastPostDateTime": {"S": "201603190422"}
}
```

UpdateItem

Se um item com a chave especificada não existir, `UpdateItem` cria um novo item. Caso contrário, ele modifica os atributos de um item existente.

Você usa uma expressão de atualização para especificar os atributos que deseja modificar e seus novos valores. Para mais informações, consulte [Expressões de atualização \(p. 431\)](#).

Na expressão de atualização, use valores de atributos de expressão como espaços reservados para os valores reais. Para mais informações, consulte [Valores de atributo de expressão \(p. 421\)](#).

Example

Modificar vários atributos no item `Thread`. O parâmetro opcional `ReturnValues` mostra o item como ele aparece após a atualização. Para mais informações, consulte [Valores de retorno \(p. 407\)](#).

```
aws dynamodb update-item \  
    --table-name Thread \  
    --key file://key.json \  
    --update-expression "SET Answered = :zero, Replies = :zero, LastPostedBy  
= :lastpostedby" \  
    --expression-attribute-values file://expression-attribute-values.json \  
    --return-values ALL_NEW
```

Os argumentos de `--key` são armazenados no arquivo `key.json`.

```
{  
    "ForumName": {"S": "Amazon DynamoDB"},  
    "Subject": {"S": "New discussion thread"}  
}
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `expression-attribute-values.json`.

```
{  
    ":zero": {"N": "0"},  
    ":lastpostedby": {"S": "barney@example.com"}  
}
```

DeleteItem

`DeleteItem` exclui o item com a chave especificada.

Example

O exemplo da AWS CLI a seguir mostra como excluir o item `Thread`.

```
aws dynamodb delete-item \  
    --table-name Thread \  
    --key file://key.json
```

Valores de retorno

Em alguns casos, talvez você queira que o DynamoDB retorne determinados valores de atributos como eles aparecem antes ou depois de modificados. As operações `PutItem`, `UpdateItem` e `DeleteItem` têm um parâmetro `ReturnValues` que você pode usar para retornar os valores de atributo antes ou depois que eles são modificados.

O valor padrão para o `ReturnValues` é `NONE`, o que significa que o DynamoDB não retornará nenhuma informação sobre atributos que foram modificadas.

A seguir estão outras configurações válidas de `ReturnValues`, organizado pela operação da API do DynamoDB.

PutItem

- `ReturnValues: ALL_OLD`

- Se você substituir um item existente, `ALL_OLD` retornará o item inteiro, conforme ele aparecia antes da substituição.
- Se você gravar um item que não existe, `ALL_OLD` não terá efeito.

UpdateItem

O uso mais comum de `UpdateItem` é para atualizar um item existente. No entanto, `UpdateItem` realmente executa um upsert, o que significa que ele criará o item automaticamente se ele ainda não existir.

- `ReturnValues: ALL_OLD`
 - Se você atualizar um item existente, `ALL_OLD` retornará o item inteiro, conforme ele aparecia antes da atualização.
 - Se você atualizar um item que não existe (upsert), `ALL_OLD` não terá efeito.
- `ReturnValues: ALL_NEW`
 - Se você atualizar um item existente, `ALL_NEW` retornará o item inteiro, conforme ele aparecia depois da atualização.
 - Se você atualizar um item que não existe (upsert), `ALL_NEW` retornará o item inteiro.
- `ReturnValues: UPDATED_OLD`
 - Se você atualizar um item existente, `UPDATED_OLD` retornará apenas os atributos atualizados, como eles apareciam antes da atualização.
 - Se você atualizar um item que não existe (upsert), `UPDATED_OLD` não terá efeito.
- `ReturnValues: UPDATED_NEW`
 - Se você atualizar um item existente, `UPDATED_NEW` retornará apenas os atributos afetados, como eles apareciam depois da atualização.
 - Se você atualizar um item que não existe (upsert), `UPDATED_NEW` retornará apenas os atributos atualizados, conforme eles aparecem após a atualização.

DeleteItem

- `ReturnValues: ALL_OLD`
 - Se você excluir um item existente, `ALL_OLD` retornará o item inteiro, como ele aparecia antes de você excluí-lo.
 - Se você excluir um item que não existe, `ALL_OLD` não retornará nenhum dado.

Operações em lote

Para aplicativos que precisam ler ou gravar vários itens, o DynamoDB oferece `BatchGetItem` e `BatchWriteItem` operações. Usar essas operações pode reduzir o número de round trips da rede do seu aplicativo para o DynamoDB. Além disso, o DynamoDB executa as operações de leitura ou de gravação individuais em paralelo. Seus aplicativos se beneficiam desse paralelismo sem a necessidade de gerenciar a simultaneidade ou threading.

As operações em lote são essencialmente wrappers em torno de várias solicitações de leitura ou de gravação. Por exemplo, se um `BatchGetItem` contém cinco itens, o DynamoDB executa cinco `GetItem` operações em seu nome. Da mesma forma, se um `BatchWriteItem` A solicitação contém duas solicitações Put e quatro solicitações Delete, o DynamoDB realizará dois `PutItem` e quatro `DeleteItem` solicitações.

Em geral, uma operação em lote não falha, a menos que todas as solicitações no lote falhem. Por exemplo, suponha que você execute uma operação `BatchGetItem`, mas que uma das solicitações

`GetItem` individuais no lote falhe. Neste caso, `BatchGetItem` retorna as chaves e os dados da solicitação `GetItem` que falhou. As outras solicitações `GetItem` no lote não são afetadas.

BatchGetItem

Um único `BatchGetItem` A operação pode conter até 100 individuais. `GetItem` pode recuperar até 16 MB de dados. Além disso, uma operação `BatchGetItem` pode recuperar itens de várias tabelas.

Example

Recuperar dois itens da tabela `Thread` usando uma expressão de projeção para retornar apenas alguns dos atributos.

```
aws dynamodb batch-get-item \  
  --request-items file://request-items.json
```

Os argumentos de `--request-items` são armazenados no arquivo `request-items.json`.

```
{  
    "Thread": {  
        "Keys": [  
            {  
                "ForumName": {"S": "Amazon DynamoDB"},  
                "Subject": {"S": "DynamoDB Thread 1"}  
            },  
            {  
                "ForumName": {"S": "Amazon S3"},  
                "Subject": {"S": "S3 Thread 1"}  
            }  
        ],  
        "ProjectionExpression": "ForumName, Subject, LastPostedDateTime, Replies"  
    }  
}
```

BatchWriteItem

O `BatchWriteItem` A operação pode conter até 25 individuais. `PutItem` e `DeleteItem` pode gravar até 16 MB de dados. (O tamanho máximo de um item individual é de 400 KB.) Além disso, uma operação `BatchWriteItem` pode inserir ou excluir itens em várias tabelas.

Note

`BatchWriteItem` não é compatível com solicitações `UpdateItem`.

Example

Gravar dois itens na tabela `ProductCatalog`.

```
aws dynamodb batch-write-item \  
  --request-items file://request-items.json
```

Os argumentos de `--request-items` são armazenados no arquivo `request-items.json`.

```
{  
    "ProductCatalog": [  
        {  
            "PutRequest": {  
                "Item": {  
                    "Id": { "N": "601" },  
                    "Description": { "S": "Snowboard" },  
                    "Price": { "N": "199.99" },  
                    "Type": { "S": "Snowboard" }  
                }  
            }  
        }  
    ]  
}
```

```
        "QuantityOnHand": { "N": "5" },
        "Price": { "N": "100" }
    }
},
{
    "PutRequest": {
        "Item": {
            "Id": { "N": "602" },
            "Description": { "S": "Snow shovel" }
        }
    }
}
]
```

Contadores atômicos

Você pode usar o `UpdateItem` operação para implementar um contador atômico- um atributo numérico que é incrementado, incondicionalmente, sem interferir em outras solicitações de gravação. (Todas as solicitações de gravação são aplicadas na ordem em que foram recebidas.) Com um contador atômico, as atualizações não são imutáveis. Em outras palavras, o valor numérico é incrementado cada vez que você chama `UpdateItem`.

Você pode usar um contador atômico para rastrear o número de visitantes de um site. Neste caso, seu aplicativo poderia incrementar um valor numérico, independentemente do seu valor atual. Se uma operação `UpdateItem` falhar, o aplicativo poderá simplesmente tentar a operação novamente. Isso arriscaria atualizar o contador duas vezes, mas provavelmente você poderia tolerar uma pequena contagem a mais ou a menos de visitantes do site.

Um contador atômico não seria apropriado onde uma contagem a mais ou a menos não pudesse ser tolerada (por exemplo, em um aplicativo bancário). Nesse caso, é mais seguro usar uma atualização condicional em vez de um contador atômico.

Para mais informações, consulte [Incremento e redução de atributos numéricos \(p. 435\)](#).

Example

O exemplo da AWS CLI a seguir incrementa o `Price` de um produto em 5. (Como `UpdateItem` não é imutável, o `Price` aumenta cada vez que você executa este exemplo.)

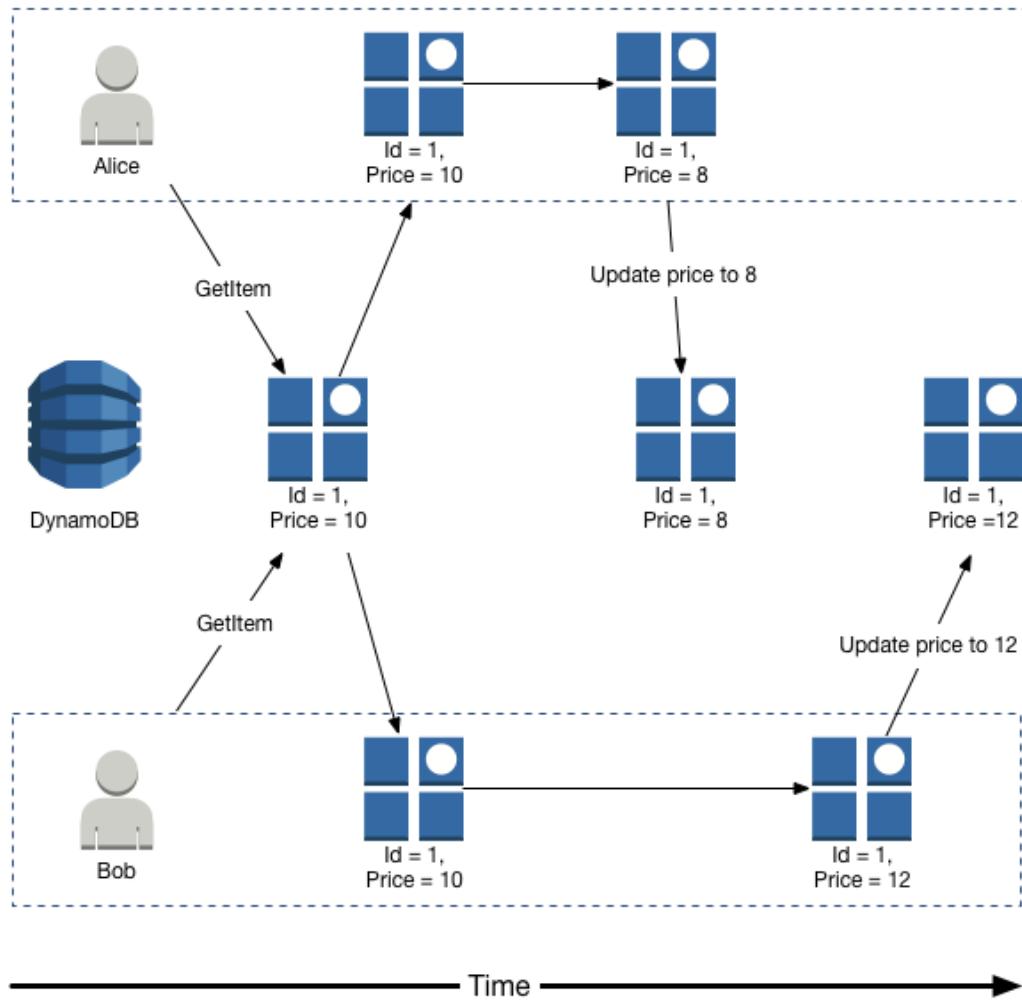
```
aws dynamodb update-item \
--table-name ProductCatalog \
--key '{"Id": { "N": "601" }}' \
--update-expression "SET Price = Price + :incr" \
--expression-attribute-values '{":incr":{"N":"5"}}' \
--return-values UPDATED_NEW
```

Gravações condicionais

Por padrão, as operações de gravação do DynamoDB (`PutItem`,`UpdateItem`,`DeleteItem`) são incondicionais: Cada operação substitui um item existente que possua a chave primária especificada.

Opcionalmente, o DynamoDB é compatível com gravações condicionais dessas operações. Uma gravação condicional é bem-sucedida somente se os atributos de item atenderem a uma ou mais condições esperadas. Caso contrário, um erro é retornado. Gravações condicionais são úteis em muitas situações. Por exemplo, talvez você queira que uma operação `PutItem` seja bem-sucedida somente se já não houver um item com a mesma chave primária. Ou você poderia impedir que uma operação `UpdateItem` modificasse um item, se um de seus atributos tivesse um determinado valor.

As gravações condicionais são úteis nos casos em que vários usuários tentam modificar o mesmo item. Considere o diagrama a seguir, no qual dois usuários (Alice e Bob) estão trabalhando com o mesmo item em uma tabela do DynamoDB.



Suponha que Alice usa a AWS CLI para atualizar o atributo **Price** para 8.

```
aws dynamodb update-item \
--table-name ProductCatalog \
--key '{"Id":{"N":"1"}}' \
--update-expression "SET Price = :newval" \
--expression-attribute-values file://expression-attribute-values.json
```

Os argumentos de **--expression-attribute-values** são armazenados no arquivo **expression-attribute-values.json**:

```
{
  ":newval":{"N":"8"}}
```

}

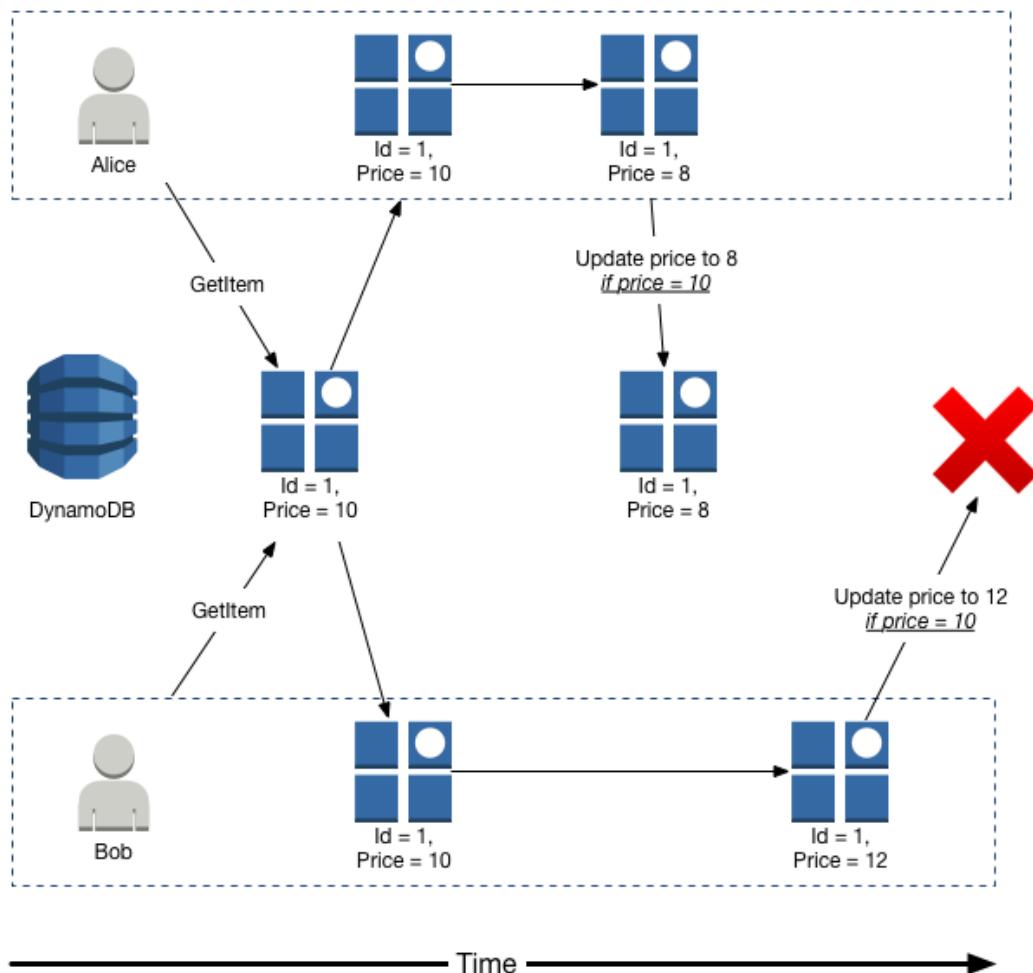
Agora, suponha que Bob emita uma solicitação `UpdateItem` semelhante mais tarde, mas altere o `Price` para 12. Para Bob, o parâmetro `--expression-attribute-values` tem a seguinte aparência.

```
{  
    ":newval": {"N": "12"}  
}
```

A solicitação de Bob é bem-sucedida, mas a atualização anterior de Alice é perdida.

Para solicitar um `PutItem`, `DeleteItem` ou `UpdateItem` condicional, você especifica uma expressão de condição. Uma expressão de condição é uma string que contém nomes de atributos, operadores condicionais e funções internas. A expressão inteira deve ser avaliada como verdadeira. Caso contrário, a operação falhará.

Agora, considere o seguinte diagrama que mostra como gravações condicionais impediriam que a atualização de Alice fosse substituída.



Alice primeiro tenta atualizar `Price` para 8, mas somente se o `Price` atual for 10.

```
aws dynamodb update-item \
    --table-name ProductCatalog \
    --key '{"Id":{"N":"1"}}' \
    --update-expression "SET Price = :newval" \
    --condition-expression "Price = :currval" \
    --expression-attribute-values file://expression-attribute-values.json
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `expression-attribute-values.json`.

```
{
    ":newval":{"N":"8"},
    ":currval":{"N":"10"}
}
```

A atualização de Alice é bem-sucedida porque a condição é avaliada como verdadeira.

Em seguida, Bob tenta atualizar o `Price` para 12, mas somente se o `Price` atual for 10. Para Bob, o parâmetro `--expression-attribute-values` tem a seguinte aparência.

```
{
    ":newval":{"N":"12"},
    ":currval":{"N":"10"}
}
```

Como Alice mudou o `Price` para 8 anteriormente, a expressão de condição é avaliada como falsa, e a atualização de Bob falha.

Para mais informações, consulte [Expressões de condição \(p. 422\)](#).

Imutabilidade da gravação condicional

As gravações condicionais podem ser imutáveis se a verificação condicional estiver no mesmo atributo que está sendo atualizado. Isso significa que o DynamoDB realiza uma determinada solicitação de gravação somente se certos valores de atributo no item corresponderem aos valores estimados no momento da solicitação.

Por exemplo, suponha que você emita uma solicitação de `UpdateItem` para aumentar o `Price` de um item em 3, mas somente se o `Price` atual for 20. Depois de enviar a solicitação, mas antes de obter os resultados de volta, ocorre um erro de rede e você não sabe se a solicitação foi bem-sucedida. Como essa gravação condicional é imutável, você pode tentar novamente o mesmo `UpdateItem`, e o DynamoDB atualizará o item somente se a `Price` é atualmente 20.

Unidades de capacidade consumidas por gravações condicionais

Em caso de `ConditionExpression` for avaliado como false durante uma gravação condicional, o DynamoDB ainda consumirá capacidade de gravação da tabela:

- Se o item não existir no momento na tabela, o DynamoDB consumirá uma unidade de capacidade de gravação.
- Se o item existir, o número de unidades de capacidade de gravação consumidas dependerá do tamanho do item. Por exemplo, uma gravação condicional falha de um item de 1 KB consumiria uma unidade de capacidade de gravação. Se os itens forem duas vezes esse tamanho, a gravação condicional falha consumiria duas unidades de capacidade de gravação.

Note

As operações de gravação consomem apenas unidades de capacidade de gravação. Elas nunca consomem unidades de capacidade de leitura.

Uma gravação condicional com falha retorna uma `ConditionalCheckFailedException`. Quando isso ocorrer, você não recebe nenhuma informação na resposta sobre a capacidade de gravação que foi consumida. No entanto, você pode ver o `ConsumedWriteCapacityUnits` para a tabela no Amazon CloudWatch. Para obter mais informações, consulte [Métricas do DynamoDB \(p. 930\)](#) em [Registro em log e monitoramento no DynamoDB \(p. 928\)](#).

Para retornar o número de unidades de capacidade de gravação consumidas durante uma gravação condicional, você deve usar o parâmetro `ReturnConsumedCapacity`:

- `TOTAL`— retorna o número total de unidades de capacidade de gravação consumidas.
- `INDEXES`— retorna o número total de unidades de capacidade de gravação consumidas, com subtotais para a tabela e para todos os índices secundários que foram afetados pela operação.
- `NONE`— nenhum detalhe da capacidade de gravação é retornado. (Esse é o padrão.)

Note

Ao contrário de um índice secundário global, um índice secundário local compartilha seu throughput provisionado com sua tabela. A atividade de leitura e gravação em um índice secundário local consome a capacidade de throughput provisionado da tabela.

Uso de expressões no DynamoDB

No Amazon DynamoDB, você usa expressões para indicar os atributos que você deseja ler a partir de um item. Você também usa expressões ao gravar um item para indicar as condições que devem ser atendidas (o que também é conhecido como atualização condicional) e para indicar como os atributos devem ser atualizados. Esta seção descreve a gramática de expressão básica e os tipos disponíveis de expressões.

Note

Para compatibilidade com versões anteriores, o DynamoDB também aceita parâmetros condicionais que não usam expressões. Para mais informações, consulte [Parâmetros condicionais herdados \(p. 1134\)](#).

Novos aplicativos devem usar expressões em vez de parâmetros herdados.

Tópicos

- [Especificar atributos de item ao usar expressões \(p. 414\)](#)
- [Expressões de projeção \(p. 417\)](#)
- [Nomes de atributo de expressão no DynamoDB \(p. 418\)](#)
- [Valores de atributo de expressão \(p. 421\)](#)
- [Expressões de condição \(p. 422\)](#)
- [Expressões de atualização \(p. 431\)](#)

Especificar atributos de item ao usar expressões

Esta seção descreve como consultar atributos de item em uma expressão no Amazon DynamoDB. Você pode trabalhar com qualquer atributo, mesmo se ele for altamente aninhado dentro de várias listas e mapas.

Tópicos

- [Atributos de nível superior \(p. 416\)](#)
- [Atributos aninhados \(p. 416\)](#)
- [Caminhos de documentos \(p. 417\)](#)

Um item de exemplo: ProductCatalog

Veja a seguir uma representação de um item na tabela `ProductCatalog`. (Esta tabela é descrita na seção [Tabelas e dados de exemplo \(p. 1070\)](#).)

```
{  
    "Id": 123,  
    "Title": "Bicycle 123",  
    "Description": "123 description",  
    "BicycleType": "Hybrid",  
    "Brand": "Brand-Company C",  
    "Price": 500,  
    "Color": ["Red", "Black"],  
    "ProductCategory": "Bicycle",  
    "InStock": true,  
    "QuantityOnHand": null,  
    "RelatedItems": [  
        341,  
        472,  
        649  
    ],  
    "Pictures": {  
        "FrontView": "http://example.com/products/123_front.jpg",  
        "RearView": "http://example.com/products/123_rear.jpg",  
        "SideView": "http://example.com/products/123_left_side.jpg"  
    },  
    "ProductReviews": {  
        "FiveStar": [  
            "Excellent! Can't recommend it highly enough! Buy it!",  
            "Do yourself a favor and buy this."  
        ],  
        "OneStar": [  
            "Terrible product! Do not buy this."  
        ]  
    },  
    "Comment": "This product sells out quickly during the summer",  
    "Safety.Warning": "Always wear a helmet"  
}
```

Observe o seguinte:

- O valor da chave de partição (`Id`) é 123. Não há chave de classificação.
- A maioria dos atributos têm tipos de dados escalares, como `String`, `Number`, `Boolean` e `Null`.
- Um atributo (`Color`) é um `String Set`.
- Os atributos a seguir são tipos de dados de documento:
 - Uma lista de `RelatedItems`. Cada elemento é um `Id` de um produto relacionado.
 - Uma mapa de `Pictures`. Cada elemento é uma breve descrição de uma imagem, juntamente com um URL para o arquivo de imagem correspondente.
 - Uma mapa de `ProductReviews`. Cada elemento representa uma classificação e uma lista de avaliações correspondentes a essa classificação. Inicialmente, este mapa é preenchido com avaliações de cinco estrelas e de uma estrela.

Atributos de nível superior

Um atributo será considerado como de nível superior se ele não estiver incorporado a outro atributo. Para o item do `ProductCatalog`, os atributos de nível superior são os seguintes:

- `Id`
- `Title`
- `Description`
- `BicycleType`
- `Brand`
- `Price`
- `Color`
- `ProductCategory`
- `InStock`
- `QuantityOnHand`
- `RelatedItems`
- `Pictures`
- `ProductReviews`
- `Comment`
- `Safety.Warning`

Todos esses atributos de nível superior são escalares, exceto `Color` (lista), `RelatedItems` (lista), `Pictures` (mapa) e `ProductReviews` (mapa).

Atributos aninhados

Um atributo é considerado aninhado se ele estiver incorporado a outro atributo. Para acessar um atributo aninhado, utiliza-se operadores de cancelamento de referência:

- `[n]`— para elementos de lista
- `.(ponto)` — para elementos de mapa

Acesso a elementos de lista

O operador de cancelamento de referência de um elemento de lista é `[n]`, onde `n` é o número do elemento. Os elementos de lista são baseados em zero, portanto, `[0]` representa o primeiro elemento na lista, `[1]` representa o segundo, e assim por diante. Aqui estão alguns exemplos:

- `MyList[0]`
- `AnotherList[12]`
- `ThisList[5][11]`

O próprio elemento `ThisList[5]` é uma lista aninhada. Portanto, `ThisList[5][11]` refere-se ao décimo segundo elemento na lista.

O número dentro de colchetes deve ser um inteiro não negativo. Portanto, as seguintes expressões são inválidas:

- `MyList[-1]`

- `MyList[0..4]`

Acesso a elementos de mapa

O operador de cancelamento de referência de um elemento de mapa é `.` (um ponto). Use um ponto como um separador entre os elementos em um mapa:

- `MyMap.nestedField`
- `MyMap.nestedField.deeplyNestedField`

Caminhos de documentos

Em uma expressão, você usa um Caminho de documentos para informar ao DynamoDB onde encontrar um atributo. Para um atributo de nível superior, o caminho do documento é simplesmente o nome do atributo. Para um atributo aninhado, você pode construir o caminho do documento usando operadores de cancelamento de referência.

Veja a seguir alguns exemplos de caminhos de documentos. (Consulte o item mostrado em [Especificando atributos de item ao usar expressões \(p. 414\)](#).)

- Um atributo escalar de nível superior.

`Description`

- Um atributo de lista de nível superior. (Isso retorna a lista inteira, não apenas alguns dos elementos.)

`RelatedItems`

- O terceiro elemento na lista `RelatedItems`. (Lembre-se de que os elementos de lista são baseadas em zero.)

`RelatedItems[2]`

- A imagem da visão frontal do produto.

`Pictures.FrontView`

- Todas as críticas de cinco estrelas.

`ProductReviews.FiveStar`

- A primeira das críticas de cinco estrelas.

`ProductReviews.FiveStar[0]`

Note

A profundidade máxima de um caminho de documento é 32. Portanto, o número de operadores de cancelamento de referência em um caminho não pode exceder esse limite.

Você pode usar qualquer nome de atributo em um caminho de documento, desde que o primeiro caractere seja `a-z` ou `A-Z` e o segundo caractere (se houver) seja `a-z`, `A-Z` ou `0-9`. Se um nome de atributo não atender a essa exigência, você deverá definir um nome de atributo de expressão como um espaço reservado. Para mais informações, consulte [Nomes de atributo de expressão no DynamoDB \(p. 418\)](#).

Expressões de projeção

Para ler dados de uma tabela, use operações como `GetItem`, `Query`, ou `Scan`. O Amazon DynamoDB retornará todos os atributos de item por padrão. Para obter apenas alguns, em vez de todos os atributos, use uma expressão de projeção.

Uma expressão de projeção é uma string que identifica os atributos que você deseja. Para recuperar um único atributo, especifique o seu nome. Para vários atributos, os nomes devem ser separados por vírgulas.

Veja a seguir alguns exemplos de expressões de projeção, com base no item do `ProductCatalog` em [Especificificar atributos de item ao usar expressões \(p. 414\)](#):

- Um único atributo de nível superior.

`Title`

- Três atributos de nível superior. O DynamoDB recupera todo o `Colorset`.

`Title, Price, Color`

- Quatro atributos de nível superior. O DynamoDB retorna todo o conteúdo `doRelatedItems` e `ProductReviews`.

`Title, Description, RelatedItems, ProductReviews`

Você pode usar qualquer nome de atributo em uma expressão de projeção, desde que o primeiro caractere seja a-z ou A-Z e o segundo caractere (se houver) seja a-z, A-Z ou 0-9. Se um nome de atributo não atender a essa exigência, você deverá definir um nome de atributo de expressão como um espaço reservado. Para mais informações, consulte [Nomes de atributo de expressão no DynamoDB \(p. 418\)](#).

O exemplo de AWS CLI a seguir mostra como usar uma expressão de projeção com uma operação `GetItem`. Essa expressão de projeção recupera um atributo escalar de nível superior (`Description`), o primeiro elemento em uma lista (`RelatedItems[0]`) e uma lista aninhada em um mapa (`ProductReviews.FiveStar`).

```
aws dynamodb get-item \
--table-name ProductCatalog \
--key file://key.json \
--projection-expression "Description, RelatedItems[0], ProductReviews.FiveStar"
```

Os argumentos de `--key` são armazenados no arquivo `key.json`.

```
{
  "Id": { "N": "123" }
}
```

Para ver exemplos de código específicos de linguagem de programação, consulte [Conceitos básicos do DynamoDB e do AWSSDKs da \(p. 81\)](#).

Nomes de atributo de expressão no DynamoDB

Um nome de atributo de expressão é um espaço reservado que você usa em uma expressão do Amazon DynamoDB como uma alternativa ao nome de atributo real. Um nome de atributo de expressão deve começar com um sinal de libra (#) seguido por um ou mais caracteres alfanuméricos.

Esta seção descreve várias situações em que você precisa usar nomes de atributos de expressão.

Note

Os exemplos desta seção usam a AWS Command Line Interface (AWS CLI). Para ver exemplos de código específicos de linguagem de programação, consulte [Conceitos básicos do DynamoDB e do AWSSDKs da \(p. 81\)](#).

Tópicos

- [Palavras reservadas \(p. 419\)](#)
- [Nomes de atributo que contêm pontos \(p. 419\)](#)
- [Atributos aninhados \(p. 420\)](#)
- [Nomes de atributo repetitivos \(p. 420\)](#)

Palavras reservadas

Algumas vezes, pode ser necessário escrever uma expressão contendo um nome de atributo que está em conflito com uma palavra reservada do DynamoDB. (Para obter uma lista completa de palavras reservadas, consulte [Palavras reservadas no DynamoDB \(p. 1125\)](#).)

Por exemplo, o seguinte exemplo da AWS CLI falharia porque `COMMENT` é uma palavra reservada.

```
aws dynamodb get-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"123"}}' \  
    --projection-expression "Comment"
```

Para resolver esse problema, você pode substituir `Comment` por um nome de atributo de expressão, como `#c`. A `#` (cerquilha) é obrigatória e indica que esse é um espaço reservado para um nome de atributo. O exemplo da AWS CLI agora deve ter a seguinte aparência.

```
aws dynamodb get-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"123"}}' \  
    --projection-expression "#c" \  
    --expression-attribute-names '{"#c":"Comment"}'
```

Note

Se um nome de atributo começar com um número ou contiver um espaço, um caractere especial, ou uma palavra reservada, você deverá usar um nome de atributo de expressão para substituir esse nome de atributo na expressão.

Nomes de atributo que contêm pontos

Em uma expressão, um ponto (`.`) é interpretado como um caractere de separação em um caminho de documento. No entanto, o DynamoDB também permite que você use um ponto como parte de um nome de atributo. Isso pode ser ambíguo em alguns casos. Para ilustrar, vamos supor que você queira recuperar o atributo `Safety.Warning` de um item do `ProductCatalog` (consulte [Especificando atributos de item ao usar expressões \(p. 414\)](#)).

Suponha que você queira acessar `Safety.Warning` usando uma expressão de projeção.

```
aws dynamodb get-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"123"}}' \  
    --projection-expression "Safety.Warning"
```

O DynamoDB retornaria um resultado vazio, em vez da string esperada ("Always wear a helmet"). Isso ocorre porque o DynamoDB interpreta um ponto em uma expressão como um separador de caminho de documento. Neste caso, defina um nome de atributo de expressão (por exemplo, `#sw`) como um substituto para `Safety.Warning`. Você poderia usar a seguinte expressão de projeção.

```
aws dynamodb get-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"123"}}' \  
    --projection-expression "#sw"
```

```
--table-name ProductCatalog \
--key '{"Id":{"N":"123"}}' \
--projection-expression "#sw" \
--expression-attribute-names '{"#sw":"Safety.Warning"}'
```

Em seguida, o DynamoDB deverá retornar o resultado correto.

Atributos aninhados

Suponha que você queira acessar o atributo aninhado `ProductReviews.OneStar` usando a seguinte expressão de projeção.

```
aws dynamodb get-item \
--table-name ProductCatalog \
--key '{"Id":{"N":"123"}}' \
--projection-expression "ProductReviews.OneStar"
```

O resultado deveria conter todas as avaliações de produto de uma estrela, que é o esperado.

Mas e se você decidiu usar um nome de atributo de expressão em vez disso? Por exemplo, o que aconteceria se você definisse `#pr1star` como um substituto para `ProductReviews.OneStar`?

```
aws dynamodb get-item \
--table-name ProductCatalog \
--key '{"Id":{"N":"123"}}' \
--projection-expression "#pr1star" \
--expression-attribute-names {"#pr1star":"ProductReviews.OneStar"}
```

O DynamoDB retornaria um resultado vazio, em vez do mapa esperado de revisões de uma estrela. Isso ocorre porque o DynamoDB interpreta um ponto no nome de atributo de expressão como um caractere no nome de um atributo. Quando o DynamoDB avalia o nome do atributo de expressão `#pr1star`, ele determina que `ProductReviews.OneStar` se refere a um atributo escalar, o que não era pretendido.

A abordagem correta seria definir um nome de atributo de expressão para cada elemento no caminho do documento:

- `#pr` – `ProductReviews`
- `#1star` – `OneStar`

Você poderia usar `#pr.#1star` para a expressão de projeção.

```
aws dynamodb get-item \
--table-name ProductCatalog \
--key '{"Id":{"N":"123"}}' \
--projection-expression "#pr.#1star" \
--expression-attribute-names {"#pr":"ProductReviews", "#1star": "OneStar"}
```

Em seguida, o DynamoDB deverá retornar o resultado correto.

Nomes de atributo repetitivos

Os nomes de atributo de expressão são úteis quando você precisa consultar o mesmo nome de atributo repetidamente. Por exemplo, considere a seguinte expressão para recuperar algumas das revisões de um item do `ProductCatalog`.

```
aws dynamodb get-item \
```

```
--table-name ProductCatalog \
--key '{"Id":{"N":"123"}}' \
--projection-expression "ProductReviews.FiveStar, ProductReviews.ThreeStar,
ProductReviews.OneStar"
```

Para tornar isso mais conciso, você pode substituir `ProductReviews` por um nome de atributo de expressão, como `#pr`. A expressão revisada agora teria a seguinte aparência.

- `#pr.FiveStar, #pr.ThreeStar, #pr.OneStar`

```
aws dynamodb get-item \
--table-name ProductCatalog \
--key '{"Id":{"N":"123"}}' \
--projection-expression "#pr.FiveStar, #pr.ThreeStar, #pr.OneStar" \
--expression-attribute-names '{"#pr":"ProductReviews"}'
```

Caso defina um nome de atributo de expressão, você deverá usá-lo de forma consistente na expressão inteira. Além disso, não é possível omitir o símbolo `#`.

Valores de atributo de expressão

Se você precisa comparar um atributo com um valor, defina um valor de atributo de expressão como um espaço reservado. Valores de atributo de expressão no Amazon DynamoDB são substitutos para os valores reais que você deseja comparar — valores que você pode não saber até o tempo de execução. Um valor de atributo de expressão deve começar com um sinal de dois pontos (`:`) seguido por um ou mais caracteres alfanuméricos.

Por exemplo, suponha que você quisesse retornar todos os itens de `ProductCatalog` que estão disponíveis em `Black` e custam 500 ou menos. Você poderia usar uma operação `Scan` com uma expressão de filtro, como neste exemplo da AWS Command Line Interface (AWS CLI).

```
aws dynamodb scan \
--table-name ProductCatalog \
--filter-expression "contains(Color, :c) and Price <= :p" \
--expression-attribute-values file://values.json
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `values.json`.

```
{
  ":c": { "S": "Black" },
  ":p": { "N": "500" }
}
```

Note

Uma operação `Scan` lê cada item de uma tabela. Portanto, você deve evitar o uso de `Scan` com tabelas grandes.

A expressão de filtro é aplicada aos resultados de `Scan` e os itens que não correspondem à expressão do filtro são descartados.

Caso defina um valor de atributo de expressão, você deverá usá-lo de forma consistente na expressão inteira. Além disso, não é possível omitir o símbolo `:`.

Valores de atributo de expressão são usados com expressões de condição chave, expressões de condição, expressões de atualização e expressões de filtro.

Note

Para ver exemplos de código específicos de linguagem de programação, consulte [Conceitos básicos do DynamoDB e do AWSSDKs da \(p. 81\)](#).

Expressões de condição

Para manipular dados em uma tabela do Amazon DynamoDB, use o `PutItem`, `UpdateItem`, `DeleteItem` operações. (Você também pode usar `BatchWriteItem` para executar várias operações `PutItem` ou `DeleteItem` em uma única chamada.)

Para essas operações de manipulação de dados, é possível especificar uma expressão de condição para determinar quais itens devem ser modificados. Se a expressão da condição for avaliada como true, a operação será bem-sucedida. Caso contrário, a operação falhará.

Veja a seguir alguns exemplos da AWS Command Line Interface (AWS CLI) para uso de expressões de condição. Estes exemplos se baseiam na tabela `ProductCatalog`, que foi apresentada em [Especificando atributos de item ao usar expressões \(p. 414\)](#). A chave de partição dessa tabela é `Id`. Não há uma chave de classificação. A seguinte operação `PutItem` cria um item `ProductCatalog` de amostra ao qual os exemplos se referem.

```
aws dynamodb put-item \
--table-name ProductCatalog \
--item file://item.json
```

Os argumentos de `--item` são armazenados no arquivo `item.json`. (Para simplificar, apenas alguns atributos de item são usados.)

```
{
  "Id": {"N": "456" },
  "ProductCategory": {"S": "Sporting Goods" },
  "Price": {"N": "650" }
}
```

Tópicos

- [Put condicional \(p. 422\)](#)
- [Exclusões condicionais \(p. 423\)](#)
- [Atualizações condicionais \(p. 424\)](#)
- [Exemplos de expressão condicional \(p. 424\)](#)
- [Referência a operadores de comparação e funções \(p. 426\)](#)

Put condicional

A operação `PutItem` substitui um item com a mesma chave (se ele existir). Se quiser evitar isso, use uma expressão de condição. Isso permite que a gravação continue apenas se o item em questão ainda não tiver a mesma chave.

```
aws dynamodb put-item \
--table-name ProductCatalog \
--item file://item.json \
--condition-expression "attribute_not_exists(Id)"
```

Se a expressão de condição for avaliada como falsa, o DynamoDB retornará uma mensagem de erro informando que: A solicitação condicional falhou.

Note

Para obter mais informações sobre `attribute_not_exists` e outras funções, consulte [Referência a operadores de comparação e funções \(p. 426\)](#).

Exclusões condicionais

Para realizar uma exclusão condicional, use uma operação `DeleteItem` com uma expressão de condição. A expressão de condição deve ser avaliada como "true" para que a operação tenha sucesso; caso contrário, a operação falhará.

Considere o item em [Expressões de condição \(p. 422\)](#)

```
{  
    "Id": {  
        "N": "456"  
    },  
    "Price": {  
        "N": "650"  
    },  
    "ProductCategory": {  
        "S": "Sporting Goods"  
    }  
}
```

Suponha que você queira excluir o item, mas somente nas seguintes condições:

- O valor de `ProductCategory` é "Sporting Goods" ou "Gardening Supplies".
- O valor de `Price` está entre 500 e 600.

O exemplo a seguir tenta excluir o item.

```
aws dynamodb delete-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"456"}}' \  
    --condition-expression "(ProductCategory IN (:cat1, :cat2)) AND (Price BETWEEN :lo  
    AND :hi)" \  
    --expression-attribute-values file://values.json
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `values.json`.

```
{  
    ":cat1": {"S": "Sporting Goods"},  
    ":cat2": {"S": "Gardening Supplies"},  
    ":lo": {"N": "500"},  
    ":hi": {"N": "600"}  
}
```

Note

Na expressão de condição, o `:`(caractere dois-pontos) indica um valor de atributo de expressão—um espaço reservado para um valor real. Para mais informações, consulte [Valores de atributo de expressão \(p. 421\)](#).

Para obter mais informações sobre `IN`, `AND` e outras palavras-chave, consulte [Referência a operadores de comparação e funções \(p. 426\)](#).

Neste exemplo, a comparação `ProductCategory` é avaliada como true, mas a comparação `Price` é avaliada como false. Isso faz com que a expressão de condição seja avaliada como falsa e a operação `DeleteItem` falhe.

Atualizações condicionais

Para realizar uma atualização condicional, use uma operação `UpdateItem` com uma expressão de condição. A expressão de condição deve ser avaliada como "true" para que a operação tenha sucesso; caso contrário, a operação falhará.

Note

`UpdateItem` também oferece suporte a expressões de atualização, nas quais você especifica as modificações que deseja fazer em um item. Para mais informações, consulte [Expressões de atualização \(p. 431\)](#).

Suponha que você tenha começado com o item mostrado em [Expressões de condição \(p. 422\)](#).

```
{  
    "Id": { "N": "456"},  
    "Price": { "N": "650"},  
    "ProductCategory": { "S": "Sporting Goods"}  
}
```

O exemplo a seguir realiza uma operação `UpdateItem`. Ele tenta reduzir o `Price` de um produto em 75, mas a expressão de condição impedirá a atualização se o `Price` for menor ou igual a 500.

```
aws dynamodb update-item \  
  --table-name ProductCatalog \  
  --key '{ "Id": { "N": "456"} }' \  
  --update-expression "SET Price = Price - :discount" \  
  --condition-expression "Price > :limit" \  
  --expression-attribute-values file://values.json
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `values.json`.

```
{  
    ":discount": { "N": "75"},  
    ":limit": { "N": "500"}  
}
```

Se o valor inicial de `Price` for 650, a operação `UpdateItem` reduzirá o `Price` para 575. Se você executar a operação `UpdateItem` novamente, o valor de `Price` será reduzido para 500. Se você executá-la uma terceira vez, a expressão de condição será avaliada como false, e a atualização falhará.

Note

Na expressão de condição, o `:`(caractere dois-pontos) indica um valor de atributo de expressão—um espaço reservado para um valor real. Para mais informações, consulte [Valores de atributo de expressão \(p. 421\)](#).

Para obter mais informações sobre "`>`" e outros operadores, consulte [Referência a operadores de comparação e funções \(p. 426\)](#).

Exemplos de expressão condicional

Para obter mais informações sobre as funções usadas nos exemplos a seguir, consulte [Referência a operadores de comparação e funções \(p. 426\)](#). Se você quiser saber mais sobre como especificar diferentes tipos de atributo em uma expressão, consulte [Especificar atributos de item ao usar expressões \(p. 414\)](#).

Como verificar atributos em um item

Você pode verificar a existência (ou inexistência) de qualquer atributo. Se a expressão da condição for avaliada como verdadeira, a operação será bem-sucedida. Caso contrário, falhará.

O exemplo a seguir usa `attribute_not_exists` para excluir um produto apenas se ele não tiver um atributo `Price`.

```
aws dynamodb delete-item \
--table-name ProductCatalog \
--key '{"Id": {"N": "456"}}' \
--condition-expression "attribute_not_exists(Price)"
```

O DynamoDB também fornece uma `attribute_exists` função. O exemplo a seguir excluirá um produto apenas se este tiver recebido revisões ruins.

```
aws dynamodb delete-item \
--table-name ProductCatalog \
--key '{"Id": {"N": "456"}}' \
--condition-expression "attribute_exists(ProductReviews.OneStar)"
```

Verificar o tipo de atributo

É possível verificar o tipo de dados de um valor de atributo usando a função `attribute_type`. Se a expressão da condição for avaliada como verdadeira, a operação será bem-sucedida. Caso contrário, falhará.

O exemplo a seguir usa `attribute_type` para excluir um produto somente se ele tiver um atributo `Color` do tipo Conjunto de strings.

```
aws dynamodb delete-item \
--table-name ProductCatalog \
--key '{"Id": {"N": "456"}}' \
--condition-expression "attribute_type(Color, :v_sub)" \
--expression-attribute-values file://expression-attribute-values.json
```

Os argumentos para `--expression-attribute-values` são armazenados no arquivo `expression-attribute-values.json`.

```
{
  ":v_sub": {"S": "SS"}
}
```

Verificar o valor de início da string

É possível verificar se um valor de atributo String começa com uma substring específica usando a função `begins_with`. Se a expressão da condição for avaliada como verdadeira, a operação será bem-sucedida. Caso contrário, falhará.

O exemplo a seguir usará `begins_with` para excluir um produto somente se o elemento `FrontView` do mapa `Pictures` começar com um valor específico.

```
aws dynamodb delete-item \
--table-name ProductCatalog \
--key '{"Id": {"N": "456"}}' \
--condition-expression "begins_with(Pictures.FrontView, :v_sub)" \
--expression-attribute-values file://expression-attribute-values.json
```

Os argumentos para `--expression-attribute-values` são armazenados no arquivo `expression-attribute-values.json`.

```
{
```

```
    ":v_sub":{ "S":"http://"}  
}
```

Verificar um elemento em um conjunto

É possível verificar se há um elemento em um conjunto ou procurar uma substring em uma string usando a função `contains`. Se a expressão da condição for avaliada como verdadeira, a operação será bem-sucedida. Caso contrário, falhará.

O exemplo a seguir usará `contains` para excluir um produto somente se o Conjunto de strings `Color` tiver um elemento com um valor específico.

```
aws dynamodb delete-item \  
  --table-name ProductCatalog \  
  --key '{"Id": {"N": "456"}}' \  
  --condition-expression "contains(Color, :v_sub)" \  
  --expression-attribute-values file://expression-attribute-values.json
```

Os argumentos para `--expression-attribute-values` são armazenados no arquivo `expression-attribute-values.json`.

```
{  
  ":v_sub":{ "S":"Red"}  
}
```

Verificar o tamanho do valor de um atributo

É possível verificar o tamanho do valor de um atributo usando a função `size`. Se a expressão da condição for avaliada como verdadeira, a operação será bem-sucedida. Caso contrário, falhará.

O exemplo a seguir usará `size` para excluir um produto somente se o tamanho do atributo binário `VideoClip` for maior que 64000 bytes.

```
aws dynamodb delete-item \  
  --table-name ProductCatalog \  
  --key '{"Id": {"N": "456"}}' \  
  --condition-expression "size(VideoClip) > :v_sub" \  
  --expression-attribute-values file://expression-attribute-values.json
```

Os argumentos para `--expression-attribute-values` são armazenados no arquivo `expression-attribute-values.json`.

```
{  
  ":v_sub":{ "N":"64000"}  
}
```

Referência a operadores de comparação e funções

Esta seção discute as funções internas e as palavras-chave para escrever expressões de condição no Amazon DynamoDB.

Tópicos

- [Sintaxe de expressões de condição \(p. 427\)](#)
- [Como fazer comparações \(p. 427\)](#)
- [Functions \(p. 428\)](#)

- Avaliações lógicas (p. 430)
- Parentheses (p. 431)
- Precedência em condições (p. 431)

Sintaxe de expressões de condição

No seguinte resumo de sintaxe, um *operando* pode ser o seguinte:

- Um nome de atributo de nível superior, como `Id`, `Title`, `Description` ou `ProductCategory`
- Um caminho de documento que faz referência a um atributo aninhado

```
condition-expression ::=  
    operand comparator operand  
    | operand BETWEEN operand AND operand  
    | operand IN ( operand (', ' operand (, ...) ))  
    | function  
    | condition AND condition  
    | condition OR condition  
    | NOT condition  
    | ( condition )  
  
comparator ::=  
    =  
    | <>  
    | <  
    | <=  
    | >  
    | >=  
  
function ::=  
    attribute_exists (path)  
    | attribute_not_exists (path)  
    | attribute_type (path, type)  
    | begins_with (path, substr)  
    | contains (path, operand)  
    | size (path)
```

Como fazer comparações

Use estes comparadores para comparar um operando com um intervalo de valores ou com uma lista enumerada de valores:

- $a = b$ — true se a for igual a b
- $a <> b$ — true se a não for igual a b
- $a < b$ — true se a for menor que b
- $a <= b$ — true se a for menor que ou igual a b
- $a > b$ — true se a for maior que b
- $a >= b$ — true se a for maior ou igual a b

Use as palavras-chave `BETWEEN` e `IN` para comparar um operando com um intervalo de valores ou com uma lista enumerada de valores:

- $a \text{ BETWEEN } b \text{ AND } c$ — verdadeiro se a for maior ou igual a b e menor ou igual a c .
- $a \text{ IN } (b, c, d)$ — true if a is equal to any value in the list — por exemplo, qualquer um dos valores na lista b, c, d . A lista pode conter até 100 valores, separados por vírgulas.

Functions

Use as funções a seguir para determinar se um atributo existe em um item ou para avaliar o valor de um atributo. Esses nomes de funções diferenciam maiúsculas de minúsculas. Para um atributo aninhado, você deve fornecer o caminho completo do documento.

Função	Descrição
<code>attribute_exists (path)</code>	<p>True se o item contiver o atributo especificado por path.</p> <p>Exemplo: Verifique se um item noProductA tabela tem uma imagem vista lateral.</p> <ul style="list-style-type: none"> • <code>attribute_exists (Pictures.SideView)</code>
<code>attribute_not_exists (path)</code>	<p>True se o atributo especificado por path não existir no item.</p> <p>Exemplo: Verifique se um item tem umManufacturerAttribute.</p> <ul style="list-style-type: none"> • <code>attribute_not_exists (Manufacturer)</code>
<code>attribute_type (path, type)</code>	<p>True se o atributo no caminho especificado for de um tipo de dados específico. O parâmetro type deve ser um dos seguintes:</p> <ul style="list-style-type: none"> • S — String • SS — Conjunto de strings • N — telefone • NS — Conjunto de números • B — Binary • BS — Conjunto binário • BOOL — Booleano • NULL — Nulo • L — Lista • M — Mapa <p>Você deve usar um valor de atributo de expressão para o parâmetro type.</p> <p>Exemplo: Verifique se oQuantityOnHandO atributo é do tipo Lista. Neste exemplo, :v_sub é um espaço reservado para a string L.</p> <ul style="list-style-type: none"> • <code>attribute_type (ProductReviews.FiveStar, :v_sub)</code> <p>Você deve usar um valor de atributo de expressão para o parâmetro type.</p>
<code>begins_with (path, substr)</code>	True se o atributo especificado por path começar com uma substring específica.

Função	Descrição
	<p>Exemplo: Verifique se os primeiros caracteres da URL da imagem de vista frontal são <code>http://</code>.</p> <ul style="list-style-type: none"> • <code>begins_with (Pictures.FrontView, :v_sub)</code> <p>O valor do atributo de expressão <code>:v_sub</code> é um espaço reservado para <code>http://</code>.</p>
<code>contains (path, operand)</code>	<p>Verdadeiro se o atributo especificado por <code>path</code> for um dos seguintes:</p> <ul style="list-style-type: none"> • Uma <code>String</code> que contém uma substring específica. • Um <code>Set</code> que contém um elemento específico dentro dele. <p>O operando deve ser um <code>String</code>. Se o atributo especificado por <code>path</code> é um <code>Set</code>, o operando deve ser o tipo de elemento do conjunto.</p> <p>O caminho e o operando devem ser distintos; ou seja, <code>contains (a, a)</code> retornará um erro.</p> <p>Exemplo: Verifique se o atributo <code>Brand</code> contém a substring <code>Company</code>.</p> <ul style="list-style-type: none"> • <code>contains (Brand, :v_sub)</code> <p>O valor do atributo de expressão <code>:v_sub</code> é um espaço reservado para <code>Company</code>.</p> <p>Exemplo: Verifique se o produto está disponível em vermelho.</p> <ul style="list-style-type: none"> • <code>contains (Color, :v_sub)</code> <p>O valor do atributo de expressão <code>:v_sub</code> é um espaço reservado para <code>Red</code>.</p>

Função	Descrição
<code>size (path)</code>	<p>Retorna um número que representa o tamanho de um atributo. Veja a seguir os tipos de dados válidos para uso com <code>size</code>.</p> <p>Se o atributo for do tipo <code>String</code>, <code>size</code> retornará o comprimento da string.</p> <p>Exemplo: Verifique se a string <code>Brand</code> for menor ou igual a 20 caracteres. O valor do atributo de expressão <code>:v_sub</code> é um espaço reservado para 20.</p> <ul style="list-style-type: none"> • <code>size (Brand) <= :v_sub</code> <p>Se o atributo for do tipo <code>Binary</code>, <code>size</code> retornará o número de bytes no valor do atributo.</p> <p>Exemplo: Suponha que o <code>ProductCatalog</code> tem um atributo binário chamado <code>VideoClip</code>, que contém um curto vídeo sobre o produto em uso. A seguinte expressão verifica se <code>VideoClip</code> excede 64.000 bytes. O valor do atributo de expressão <code>:v_sub</code> é um espaço reservado para 64000.</p> <ul style="list-style-type: none"> • <code>size(VideoClip) > :v_sub</code>
	<p>Se o atributo for de um tipo de dados de <code>Set</code>, <code>size</code> retornará o número de elementos no conjunto.</p> <p>Exemplo: Verifique se o produto está disponível em mais de uma cor. O valor do atributo de expressão <code>:v_sub</code> é um espaço reservado para 1.</p> <ul style="list-style-type: none"> • <code>size (Color) < :v_sub</code>
	<p>Se o atributo for do tipo <code>List</code> ou <code>Map</code>, <code>size</code> retornará o número de elementos filho.</p> <p>Exemplo: Verifique se o número de <code>OneStarComments</code> excedeu um determinado limite. O valor do atributo de expressão <code>:v_sub</code> é um espaço reservado para 3.</p> <ul style="list-style-type: none"> • <code>size(ProductReviews.OneStar) > :v_sub</code>

Avaliações lógicas

Use as palavras-chave `AND`, `OR` e `NOT` para executar avaliações lógicas. Na lista a seguir, `a` e `b` representam condições a serem avaliadas.

- $a \text{ AND } b$ — true se a e b forem ambos true.
- $a \text{ OR } b$ — true se a ou b (ou ambos) for true.
- $\text{NOT } a$ — true se a for false; false se a for true.

Parentheses

Use parênteses para alterar a precedência de uma avaliação lógica. Por exemplo, suponha que as condições a e b sejam verdadeiras e que a condição c seja falsa. As expressões a seguir são avaliadas como true:

- $a \text{ OR } b \text{ AND } c$

No entanto, se você colocar uma condição entre parênteses, ela será avaliada primeiro. Por exemplo, o seguinte é avaliado como false:

- $(a \text{ OR } b) \text{ AND } c$

Note

Você pode aninhar parênteses em uma expressão. Os componentes mais internos são avaliados primeiro.

Precedência em condições

O DynamoDB avalia as condições da esquerda para a direita, usando as seguintes regras de precedência:

- $= <> < <= > >=$
- IN
- BETWEEN
- attribute_exists attribute_not_exists begins_with contains
- Parênteses
- NOT
- AND
- OR

Expressões de atualização

Para atualizar um item existente em uma tabela do Amazon DynamoDB, use o `UpdateItem` operação. Você deve fornecer a chave do item que deseja atualizar. Você também deve fornecer uma expressão de atualização indicando os atributos que deseja modificar e os valores que deseja atribuir a eles.

Uma expressão de atualização especifica como `UpdateItem` modificará os atributos de um item — por exemplo, definindo um valor escalar ou removendo elementos de uma lista ou de um mapa.

Veja a seguir um resumo da sintaxe para expressões de atualização.

```
update-expression ::=  
  [ SET action [, action] ... ]  
  [ REMOVE action [, action] ... ]  
  [ ADD action [, action] ... ]  
  [ DELETE action [, action] ... ]
```

Uma expressão de atualização consiste em uma ou mais cláusulas. Cada cláusula começa com uma palavra-chave SET, REMOVE, ADD ou DELETE. Você pode incluir qualquer uma dessas cláusulas em uma expressão de atualização, em qualquer ordem. No entanto, cada palavra-chave de ação pode aparecer apenas uma vez.

Dentro de cada cláusula há uma ou mais ações, separadas por vírgulas. Cada ação representa uma modificação de dados.

Os exemplos desta seção se baseiam no item `ProductCatalog` mostrado em [Expressões de projeção \(p. 417\)](#).

Tópicos

- [SET — Modificação ou adição de atributos de item \(p. 432\)](#)
- [Remover — exclusão de atributos de um item \(p. 436\)](#)
- [ADD — Atualização de números e conjuntos \(p. 437\)](#)
- [DELETE — remoção de elementos de um conjunto \(p. 439\)](#)

SET — Modificação ou adição de atributos de item

Use a ação `SET` em uma expressão de atualização para adicionar um ou mais atributos a um item. Se qualquer um desses atributos já existir, eles serão substituídos pelos novos valores.

Você também pode usar `SET` para adicionar ou subtrair de um atributo do tipo `Number`. Para executar várias ações `SET`, separe-as com vírgulas.

No seguinte resumo de sintaxe:

- O elemento `path` é o caminho do documento para o item.
- Um elemento `operand` pode ser o caminho de um documento para um item ou uma função.

```
set-action ::=  
    path = value  
  
value ::=  
    operand  
    | operand '+' operand  
    | operand '-' operand  
  
operand ::=  
    path | function
```

A seguinte operação `PutItem` cria um item de exemplo ao qual os exemplos fazem referência.

```
aws dynamodb put-item \  
    --table-name ProductCatalog \  
    --item file://item.json
```

Os argumentos de `--item` são armazenados no arquivo `item.json`. (Para simplificar, apenas alguns atributos de item são usados.)

```
{  
    "Id": {"N": "789"},  
    "ProductCategory": {"S": "Home Improvement"},  
    "Price": {"N": "52"},  
    "InStock": {"BOOL": true},
```

```
    "Brand": { "S": "Acme" }  
}
```

Tópicos

- [Modificar atributos \(p. 433\)](#)
- [Adição de listas e mapas \(p. 433\)](#)
- [Adicionar elementos a uma lista \(p. 434\)](#)
- [Adição de atributos de mapa aninhados \(p. 434\)](#)
- [Incremento e redução de atributos numéricos \(p. 435\)](#)
- [Acrecentar elementos a uma lista \(p. 435\)](#)
- [Como impedir substituições de um atributo existente \(p. 436\)](#)

Modificar atributos

Example

Atualize os atributos `ProductCategory` e `Price`.

```
aws dynamodb update-item \  
  --table-name ProductCatalog \  
  --key '{ "Id": {"N": "789"} }' \  
  --update-expression "SET ProductCategory = :c, Price = :p" \  
  --expression-attribute-values file://values.json \  
  --return-values ALL_NEW
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `values.json`.

```
{  
  ":c": { "S": "Hardware" },  
  ":p": { "N": "60" }  
}
```

Note

No `UpdateItem` operação, `--return-values ALL_NEW` faz com que o DynamoDB retorne o item como ele aparece após a atualização.

Adição de listas e mapas

Example

Adicione uma nova lista e um novo mapa.

```
aws dynamodb update-item \  
  --table-name ProductCatalog \  
  --key '{ "Id": {"N": "789"} }' \  
  --update-expression "SET RelatedItems = :ri, ProductReviews = :pr" \  
  --expression-attribute-values file://values.json \  
  --return-values ALL_NEW
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `values.json`.

```
{
```

```
":ri": {  
    "L": [  
        { "S": "Hammer" }  
    ]  
},  
":pr": {  
    "M": {  
        "FiveStar": {  
            "L": [  
                { "S": "Best product ever!" }  
            ]  
        }  
    }  
}
```

Adicionar elementos a uma lista

Example

Adicione um novo atributo à lista `RelatedItems`. (Lembre-se de que elementos de lista são baseados em zero e, portanto, [0] representa o primeiro elemento da lista, [1] representa o segundo, e assim por diante.)

```
aws dynamodb update-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"789"}}' \  
    --update-expression "SET RelatedItems[1] = :ri" \  
    --expression-attribute-values file://values.json \  
    --return-values ALL_NEW
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `values.json`.

```
{  
    ":ri": { "S": "Nails" }  
}
```

Note

Quando você usa `SET` para atualizar um elemento de lista, o conteúdo desse elemento é substituído pelos novos dados especificados. Se o elemento ainda não existir, `SET` acrescentará o novo elemento ao final da lista.

Se você adicionar vários elementos em uma única operação `SET`, estes serão classificados por número de elemento.

Adição de atributos de mapa aninhados

Example

Adicione alguns atributos de mapa aninhados.

```
aws dynamodb update-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"789"}}' \  
    --update-expression "SET #pr.#5star[1] = :r5, #pr.#3star = :r3" \  
    --expression-attribute-names file://names.json \  
    --expression-attribute-values file://values.json \  
    --return-values ALL_NEW
```

Os argumentos de `--expression-attribute-names` são armazenados no arquivo `names.json`.

```
{  
    "#pr": "ProductReviews",  
    "#5star": "FiveStar",  
    "#3star": "ThreeStar"  
}
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `values.json`.

```
{  
    ":r5": { "S": "Very happy with my purchase" },  
    ":r3": {  
        "L": [  
            { "S": "Just OK - not that great" }  
        ]  
    }  
}
```

Incremento e redução de atributos numéricos

Você pode adicionar ou subtrair de um atributo numérico existente. Para fazer isso, use os operadores + (mais) e - (menos).

Example

Diminua o valor do `Price` de um item.

```
aws dynamodb update-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"789"}}' \  
    --update-expression "SET Price = Price - :p" \  
    --expression-attribute-values '{":p": {"N":"15"}}' \  
    --return-values ALL_NEW
```

Para aumentar o valor do `Price`, você usa o operador + na expressão de atualização.

Acrescentar elementos a uma lista

É possível adicionar elementos ao final de uma lista. Para fazer isso, use `SET` com a função `list_append`. (O nome da função diferencia maiúsculas de minúsculas.) A função `list_append` é específica à ação `SET` e só pode ser usada em uma expressão de atualização. A sintaxe é a seguinte.

- `list_append (list1, list2)`

A função utiliza duas listas como entrada e acrescenta todos os elementos de `list2` a `list1`.

Example

Em [Adicionar elementos a uma lista \(p. 434\)](#), você cria a lista `RelatedItems` e a preenche com dois elementos: `Hammer` e `Nails`. Na sequência, você acrescenta mais dois elementos ao final de `RelatedItems`.

```
aws dynamodb update-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"789"}}' \  
    --update-expression "SET #ri = list_append(#ri, :vals)" \  
    --expression-attribute-values '{":vals": ["Hammer", "Nails"]}'
```

```
--expression-attribute-names '{"#ri": "RelatedItems"}' \
--expression-attribute-values file://values.json \
--return-values ALL_NEW
```

Os argumentos de --expression-attribute-values são armazenados no arquivo values.json.

```
{
  ":vals": {
    "L": [
      { "S": "Screwdriver" },
      {"S": "Hacksaw" }
    ]
  }
}
```

Por fim, você acrescenta mais um elemento ao início de RelatedItems. Para fazer isso, alterne a ordem dos elementos de list_append, (Lembre-se de que list_append usa duas listas como entrada e acrescenta a segunda lista à primeira.)

```
aws dynamodb update-item \
  --table-name ProductCatalog \
  --key '{"Id":{"N":"789"}}' \
  --update-expression "SET #ri = list_append(:vals, #ri)" \
  --expression-attribute-names '{"#ri": "RelatedItems"}' \
  --expression-attribute-values '{":vals": {"L": [ { "S": "Chisel" } ]}}' \
  --return-values ALL_NEW
```

Agora, o atributo RelatedItems resultante contém cinco elementos, na seguinte ordem: Chisel, Hammer, Nails, Screwdriver, Hacksaw.

Como impedir substituições de um atributo existente

Se você deseja evitar a substituição de um atributo existente, pode usar SET com a função if_not_exists. (O nome da função diferencia maiúsculas de minúsculas.) A função if_not_exists é específica à ação SET e só pode ser usada em uma expressão de atualização. A sintaxe é a seguinte.

- `if_not_exists (path, value)`

Se o item não contiver um atributo no `path` especificado, `if_not_exists` avaliará como `value`, caso contrário avaliará como `path`.

Example

Defina o Price de um item, mas somente se o item ainda não tiver um atributo Price. (Se o atributo Price já existir, não acontecerá nada.)

```
aws dynamodb update-item \
  --table-name ProductCatalog \
  --key '{"Id":{"N":"789"}}' \
  --update-expression "SET Price = if_not_exists(Price, :p)" \
  --expression-attribute-values '{":p": {"N": "100"}}' \
  --return-values ALL_NEW
```

Remover — exclusão de atributos de um item

Usar a `REMOVE`Em uma expressão de atualização para remover um ou mais atributos de um item no Amazon DynamoDB. Para executar várias ações REMOVE, separe-as com vírgulas.

O seguinte é um resumo da sintaxe de `REMOVE` em uma expressão de atualização. O único operando é o caminho do documento do atributo que você deseja remover.

```
remove-action ::=  
    path
```

Example

Remova alguns atributos de um item. (Se os atributos não existirem, nada acontecerá.)

```
aws dynamodb update-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"789"}}' \  
    --update-expression "REMOVE Brand, InStock, QuantityOnHand" \  
    --return-values ALL_NEW
```

Remover elementos de uma lista

É possível usar `REMOVE` para excluir elementos individuais de uma lista.

Example

Em [Acrecentar elementos a uma lista \(p. 435\)](#), você modifica um atributo da lista (`RelatedItems`) de forma que ele contenha cinco elementos:

- [0]—Chisel
- [1]—Hammer
- [2]—Nails
- [3]—Screwdriver
- [4]—Hacksaw

O exemplo da AWS Command Line Interface (AWS CLI) a seguir exclui Hammer e Nails da lista.

```
aws dynamodb update-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"789"}}' \  
    --update-expression "REMOVE RelatedItems[1], RelatedItems[2]" \  
    --return-values ALL_NEW
```

Depois que Hammer e Nails forem removidos, os elementos restantes serão deslocados. Agora, a lista contém o seguinte:

- [0]—Chisel
- [1]—Screwdriver
- [2]—Hacksaw

ADD — Atualização de números e conjuntos

Note

Em geral, recomendamos o uso de `SET` em vez de `ADD`.

Use a ação `ADD` em uma expressão de atualização para adicionar um novo atributo e seus valores a um item.

Se o atributo já existir, o comportamento de ADD dependerá do tipo de dados do atributo:

- Se o atributo for um número, e o valor que você está adicionando também for um número, esse valor será matematicamente adicionado ao atributo existente. (Se o valor for um número negativo, ele será subtraído do atributo existente.)
- Se o atributo for um conjunto, e o valor que você está adicionando também for um conjunto, esse valor será acrescentado ao conjunto existente.

Note

A ação ADD oferece suporte apenas a tipos de dados de número e conjunto.

Para executar várias ações ADD, separe-as com vírgulas.

No seguinte resumo de sintaxe:

- O elemento **path** é o caminho do documento para um atributo. O atributo deve ser um `Number` ou um tipo de dados de conjunto.
- O elemento **value** é um número que você deseja adicionar ao atributo (para tipos de dados `Number`) ou um conjunto a ser acrescentado ao atributo (para tipos de conjunto).

```
add-action ::=  
    path value
```

Adição de um número

Suponha que o atributo `QuantityOnHand` não exista. O exemplo da AWS CLI a seguir define `QuantityOnHand` como 5.

```
aws dynamodb update-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"789"}}' \  
    --update-expression "ADD QuantityOnHand :q" \  
    --expression-attribute-values '{":q": {"N": "5"}}' \  
    --return-values ALL_NEW
```

Agora que `QuantityOnHand` existe, você pode executar novamente o exemplo para incrementar `QuantityOnHand` em 5 de cada vez.

Como adicionar elementos a um conjunto

Suponha que o atributo `Color` não exista. O exemplo da AWS CLI a seguir define `Color` como um conjunto de strings com dois elementos.

```
aws dynamodb update-item \  
    --table-name ProductCatalog \  
    --key '{"Id":{"N":"789"}}' \  
    --update-expression "ADD Color :c" \  
    --expression-attribute-values '{":c": {"SS":["Orange", "Purple"]}}' \  
    --return-values ALL_NEW
```

Agora que `Color` existe, você pode adicionar mais elementos a ele.

```
aws dynamodb update-item \  
    --table-name ProductCatalog \  
    --key ...
```

```
--key '{"Id":{"N":"789"}}' \
--update-expression "ADD Color :c" \
--expression-attribute-values '{":c": {"SS":["Yellow", "Green", "Blue"]}}' \
--return-values ALL_NEW
```

DELETE — remoção de elementos de um conjunto

Important

A ação **DELETE** oferece suporte apenas a tipos de dados Set.

Use a ação **DELETE** em uma expressão de atualização para remover um ou mais elementos de um conjunto. Para executar várias ações **DELETE**, separe-as com vírgulas.

No seguinte resumo de sintaxe:

- O elemento **path** é o caminho do documento para um atributo. Esse atributo deve ser um tipo de dados de conjunto.
- O elemento **subset** é um ou mais elementos que você deseja excluir de **path**. Você deve especificar o elemento **subset** como um tipo set.

```
delete-action ::=  
    path value
```

Example

Em [Como adicionar elementos a um conjunto \(p. 438\)](#), você cria o conjunto de tipo string **Color**. Este exemplo remove alguns dos elementos desse conjunto.

```
aws dynamodb update-item \
  --table-name ProductCatalog \
  --key '{"Id":{"N":"789"}}' \
  --update-expression "DELETE Color :p" \
  --expression-attribute-values '{":p": {"SS": ["Yellow", "Purple"]}}' \
  --return-values ALL_NEW
```

Itens expirando usando o Time to Live (TL — Time to Live — uso do DynamoDB

O tempo de vida útil do Amazon DynamoDB (TTL) permite definir um time stamp por item para determinar quando um item não é mais necessário. Pouco depois da data e hora do time stamp especificado, o DynamoDB exclui o item da tabela sem consumir nenhuma taxa de transferência de gravação. O TTL é fornecido sem custo adicional como um meio de reduzir os volumes de dados armazenados, mantendo apenas os itens que permanecem atualizados para as necessidades da sua carga de trabalho.

O TTL é útil se você armazena itens que perdem relevância após um tempo específico. Veja a seguir exemplos de casos de uso de TTL:

- Remova dados do usuário ou do sensor após um ano de inatividade em um aplicativo.
- Arquive itens expirados em um data lake do Amazon S3 por meio do Amazon DynamoDB Streams e do AWS Lambda.
- Mantenha dados confidenciais por um determinado período de acordo com obrigações contratuais ou regulamentares.

Para obter mais informações sobre TTL, consulte estes tópicos:

- [Usar a vida útil.](#)
- [Tempo de vida: Como ele funciona.](#)
- [Habilitar a vida útil.](#)

Como funciona: Time to Live (TTL — Tempo de vida) do DynamoDB

Quando o TTL é habilitado em uma tabela do DynamoDB, é necessário identificar um nome de atributo específico que o serviço procurará ao determinar se um item está qualificado para expiração. Depois de habilitar o TTL em uma tabela, um processo de varredura por partição, executado em segundo plano, avalia de forma automática e contínua o status de expiração dos itens na tabela.

O processo de varredura, executado em segundo plano, compara a hora atual no [formato de tempo do Unix epoch](#) em segundos com o valor armazenado no atributo definido pelo usuário de um item. Se o atributo for `umNumber`, o valor do atributo é um carimbo de data/hora em [Formato de tempo de época Unix](#) em segundos e o valor de time stamp for mais antigo do que o tempo atual, mas não cinco anos ou mais (para evitar uma possível exclusão acidental devido a um valor de TTL malformado), o item será definido como expirado. Para obter detalhes sobre como formatar atributos TTL, consulte [Formatar o atributo TTL de um item \(p. 442\)](#). Um segundo processo em segundo plano verifica se há itens expirados e os exclui. Os dois processos ocorrem automaticamente em segundo plano, não afetam o tráfego de leitura ou gravação na tabela e não têm um custo monetário.

À medida que os itens são excluídos da tabela, duas operações em segundo plano acontecem simultaneamente:

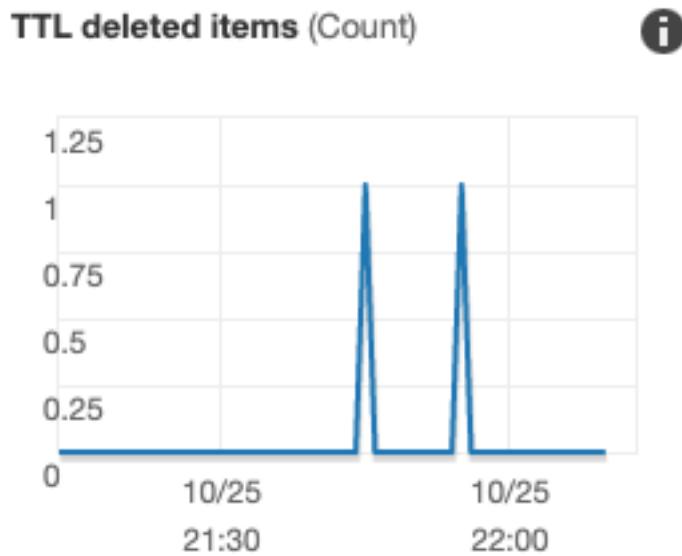
- Os itens são removidos de qualquer índice secundário local e índice secundário global da mesma forma que uma operação `DeleteItem`.
- Uma operação de exclusão para cada item entra no DynamoDB Stream, mas é marcada como uma exclusão do sistema e não como uma exclusão regular. Para obter mais informações sobre como usar essa exclusão do sistema, consulte [Streams e vida útil do DynamoDB](#).

Important

- Dependendo do tamanho e do nível de atividade de uma tabela, a operação de exclusão real de um item expirado pode variar. Como o TTL deve ser um processo em segundo plano, a natureza da capacidade usada para expirar e excluir itens via TTL é variável (mas gratuita). O TTL geralmente exclui os itens expirados em até 48 horas após a expiração.
- Itens que expiraram, mas ainda não foram excluídos pelo TTL, ainda aparecem em leituras, consultas e varreduras. Se você não quiser itens expirados no conjunto de resultados, deverá filtrá-los. Para isso, use uma expressão de filtro que retorne apenas itens nos quais o valor de expiração da vida útil seja maior que a hora atual no formato epoch. Para mais informações, consulte [Expressões de filtro de Scan \(p. 509\)](#).
- Os itens que passaram da expiração, mas ainda não foram excluídos poderão ser atualizados, e atualizações bem-sucedidas para alterar ou remover o atributo de expiração serão respeitadas.

É possível monitorar as taxas de TTL na guia de métricas do CloudWatch para uma tabela e ver quando e a taxa em que os itens são excluídos.

Time to Live (TTL)



Exemplo de vida útil

Por exemplo, considere uma tabela chamada SessionData que controla o histórico de sessão dos usuários. Cada item em SessionData é identificado pela chave de partição (UserName) e a chave de classificação (SessionId). Atributos adicionais, como UserName, SessionId, CreationTime e ExpirationTime controlam as informações das sessões. O atributo ExpirationTime é definido como o atributo TTL na tabela (nem todos os atributos em cada item são mostrados).

SessionData

UserName	SessionId	CreationTime	ExpirationTime (TTL)	SessionInfo	...
user1	74686572652773	1571820360	1571827560	{JSON Document}	...
user2	6e6f7468696e67	1571820180	1571827380	{JSON Document}	...
user3	746f2073656520	1571820923	1571828123	{JSON Document}	...
user4	68657265212121	1571820683	1571827883	{JSON Document}	...
user5	6e6572642e2e2e	1571820743	1571831543	{JSON Document}	...
...

Neste exemplo, cada item tem um valor de atributo `ExpirationTime` definido quando o item foi criado. Considere o item da tabela a seguir.

SessionData

UserName	SessionId	CreationTime	ExpirationTime (TTL)	SessionInfo	...
user1	74686572652773	1571820360	1571827560	{JSON Document}	...

Neste exemplo, o item `CreationTime` é definido como quarta-feira, 23 de outubro 08:46 UTC 2019, e o `ExpirationTime` é definido como 2 horas depois na quarta-feira, 23 de outubro 10:46 UTC 2019. O item expirará quando a hora atual, no formato epoch, for maior que a hora do atributo `ExpirationTime`. Neste caso, o item com a chave { `Username`: `user1`, `SessionId`: `74686572652773`} expira 10:46 (1571827560).

Usar a vida útil do DynamoDB (TTTL — vida útil)

Quando o TTL é usado, a maior parte do trabalho difícil é realizada nos bastidores pelo DynamoDB em seu nome. No entanto, lembre-se de algumas considerações para ajudar sua implementação a prosseguir sem problemas.

Tópicos

- [Formatar o atributo TTL de um item \(p. 442\)](#)
- [Observações sobre o uso \(p. 442\)](#)
- [Solução de problemas do TTL \(p. 443\)](#)

Formatar o atributo TTL de um item

Ao habilitar o TTL em uma tabela, o DynamoDB exige identificar um nome de atributo específico que o serviço procurará ao determinar se um item está qualificado para expiração. Além disso, outros requisitos garantem que os processos TTL em segundo plano usem o valor do atributo TTL. Se um item for elegível para expiração via TTL:

- O item deverá conter o atributo especificado quando o TTL foi habilitado na tabela. Por exemplo, se você especificar para uma tabela usar o nome do atributo `expdate` como o atributo TTL, mas um item não tiver um atributo com esse nome, o processo TTL ignorará o item.
- O valor do atributo TTL deve ser um tipo de dados `Number`. Por exemplo, se você especificar para uma tabela usar o nome do atributo `expdate` como o atributo TTL, mas o atributo em um item for um tipo de dados `String`, os processos TTL ignorarão o item.
- O valor do atributo TTL deve ser um time stamp no [formato de tempo do Unix epoch](#) em segundos. Se você usar qualquer outro formato, os processos TTL ignorarão o item. Por exemplo, se você definir o valor do atributo como 1645119622, ou seja, quinta-feira, 17 de fevereiro de 2022 17:40:22 (GMT), o item expirá após esse horário. Para obter um formulário Web visual para testar valores e ver exemplos de código de diferentes idiomas no formato de tempo de época, consulte [Epoch Converter](#).
- O valor do atributo TTL deve ser um time stamp com uma expiração não superior a cinco anos no passado. Por exemplo, se você definir o valor do atributo como 1171734022, será 17 de fevereiro de 2007 17:40:22 (GMT) e mais de cinco anos. Como resultado, os processos TTL não expirarão esse item.

Observações sobre o uso

Ao usar o TTL, considere o seguinte:

- As tarefas de habilitar, desabilitar ou alterar configurações de TTL em uma tabela podem levar aproximadamente uma hora para que as configurações se propaguem e permitam a execução de outras ações relacionadas a TTL.
- Não é possível reconfigurar o TTL para procurar um atributo diferente. É necessário desativar o TTL e, depois, reativar o TTL com o novo atributo daqui em diante.
- Quando você usa AWS CloudFormation, você pode habilitar o TTL ao criar uma tabela do DynamoDB. Para obter mais informações, consulte o [Guia do usuário do AWS CloudFormation](#).
- Você pode usar o AWS Identity and Access Management Policies do (IAM) para evitar atualizações não autorizadas no atributo TTL em um item ou na configuração do TTL. Se você permitir acesso apenas a ações especificadas nas suas políticas existentes do IAM, certifique-se de que suas políticas sejam atualizadas de forma a permitir dynamodb:UpdateTimeToLive para funções que precisam habilitar ou desabilitar o TTL em tabelas. Para obter mais informações, consulte [Usar políticas baseadas em identidade \(políticas do IAM\) para o Amazon DynamoDB](#).
- Considere se você precisa fazer qualquer pós-processamento de itens excluídos por meio do Amazon DynamoDB Streams, como arquivamento de itens em um data lake do Amazon S3. Os registros de streams de exclusões de TTL são marcados como exclusões do sistema e exclusões normais, e é possível filtrar as exclusões do sistema usando uma AWS Lambda função. Para obter mais informações sobre as adições aos registros de streams, consulte [Streams e vida útil do Amazon DynamoDB](#).
- Se a recuperação de dados for uma preocupação, recomendamos fazer backup da tabela.
 - Para backups de tabela totalmente gerenciados, use o [DynamoDB Backups sob demanda](#) ou [backups contínuos com a recuperação point-in-time](#).
 - Para uma janela de recuperação de 24 horas, é possível usar o Amazon DynamoDB Streams. Para obter mais informações, consulte [Streams e vida útil do Amazon DynamoDB](#).

Solução de problemas do TTL

Se o DynamoDB TTL não estiver funcionando, verifique o seguinte:

- Confirme se você habilitou o TTL na tabela e se o nome do atributo selecionado para TTL está definido como o que seu código está gravando nos itens. É possível confirmar essas informações na guia Visão geral de uma tabela no console do DynamoDB.
- Veja as métricas do Amazon CloudWatch na [Métricas](#) no console do DynamoDB para confirmar que o TTL está excluindo itens conforme o esperado.
- Confirme se o valor do atributo TTL está formatado corretamente. Para obter mais informações, consulte, [Formatar o atributo TTL de um item \(p. 442\)](#).

Habilitar a vida útil (TTL — vida útil)

É possível usar o console do Amazon DynamoDB ou o AWS Command Line Interface(AWS CLI) para ativar a vida útil. Para usar a API, consulte [Referência de API do Amazon DynamoDB](#).

Tópicos

- [Habilitar a vida útil \(Console\) \(p. 443\)](#)
- [Habilitar a vida útil \(AWS CLI\) \(p. 445\)](#)

Habilitar a vida útil (Console)

Siga estas etapas para habilitar o Time to Live usando o console do DynamoDB:

1. Faça login no AWS Management Console e abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.

2. Escolha Tables (Tabelas) e selecione a tabela que você deseja modificar.
3. Em Table details (Detalhes da tabela), ao lado de TTL attribute (Atributo TTL), escolha Manage TTL (Gerenciar TTL).

Table details

Table name	ttl-test
Primary partition key	name (String)
Primary sort key	-
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	April 20, 2016 at 2:23:18 PM UTC-7

4. Na caixa de diálogo Manage TTL (Gerenciar TTL), escolha Enable TTL (Habilitar TTL) e insira o nome do TTL attribute (Atributo TTL).

Enable TTL

TTL is a mechanism to set a specific timestamp for expiring items from your table. The timestamp should be expressed as an attribute on the items in the table. The attribute should be a Number data type containing time in epoch format. Once the timestamp expires, the corresponding item is deleted from the table in the background.

TTL attribute	ttl
Info: Enabling TTL can take up to 1 hour to apply across all partitions, and you will not be able to make further TTL changes until the action is complete. Please verify all information is correct to avoid loss of important data from your table.	
DynamoDB Streams <input type="checkbox"/> Enable with view type New and old images Streams are currently not enabled	
Info: Enabling Streams gives a 24-hour backup window for TTL deleted items. Additional charges and Maximum Write Capacity Limit may apply.	

Preview TTL

Before enabling TTL, it is recommended you run a preview to see samples of what items will be deleted once TTL is enabled on this table.

Run preview	preview items expiring by <input type="text" value="February 15, 2017"/>	<input style="border: 1px solid #ccc; border-radius: 5px; padding: 2px 5px;" type="button" value="09"/> : <input style="border: 1px solid #ccc; border-radius: 5px; padding: 2px 5px;" type="button" value="45"/> UTC-8
--------------------	--	---

[Cancel](#) [Continue](#)

Há três configurações em Manage TTL (Gerenciar TTL):

- Habilitar TL- Escolha esta opção para habilitar ou desabilitar o TTL na tabela. Pode levar até uma hora para que a alteração seja totalmente processada.
 - Attribute TL— O nome do atributo DynamoDB para armazenar o carimbo de data e hora do TL para os itens.
 - Streams de backup 24 horas— escolha essa configuração para habilitar os streams do Amazon DynamoDB na tabela. Para obter mais informações sobre como usar o DynamoDB Streams para backup, consulte [DynamoDB Streams e vida útil \(p. 655\)](#).
5. (Opcional) Para visualizar alguns dos itens que serão excluídos quando a TTL estiver habilitada, escolha Run preview (Executar visualização).

Warning

Isso fornece um exemplo de lista de itens para você. Isso não fornece uma lista completa de itens que serão excluídos pelo TTL.

6. Escolha Continue (Continuar) para salvar as configurações e habilitar a TTL.

Agora que o TL está habilitado, o atributo TL é marcado como TTL quando você visualiza itens no console do DynamoDB.

Você pode visualizar a data e a hora em que um item expira posicionando o mouse sobre o atributo.

	id	ttl (TTL)
	7	1460232057
	10	1459700753
	3	1459221815 
	2	1459221806
	21	1459703052

UTC: March 29, 2016 at 3:23:35 AM UTC
Local: March 28, 2016 at 8:23:35 PM UTC-7
Region (N. Virginia): March 28, 2016 at 11:23:35 PM UTC-4

Habilitar a vida útil (AWS CLI)

1. Habilite o TTL na tabela TTLExample.

```
aws dynamodb update-time-to-live --table-name TTLExample --time-to-live-specification "Enabled=true, AttributeName=ttl"
```

2. Descreva o TTL na tabela TTLExample.

```
aws dynamodb describe-time-to-live --table-name TTLExample
{
    "TimeToLiveDescription": {
        "AttributeName": "ttl",
        "TimeToLiveStatus": "ENABLED"
    }
}
```

3. Adicione um item à TTLExampleCom o conjunto de atributos Time to Live usando o shell BASH e a tabela AWS CLI.

```
EXP=`date -d '+5 days' +%s`
aws dynamodb put-item --table-name "TTLExample" --item '{"id": {"N": "1"}, "ttl": {"N": "'$EXP'"}}'
```

Este exemplo inicia na data atual e adiciona 5 dias a ela para criar uma data de expiração. Depois, ele converte a data de expiração em formato de hora epoch para finalmente adicionar um item à tabela "TTLExample".

Note

Uma forma de definir os valores de expiração para o Tempo de Vida útil é calcular o número de segundos para adicionar o tempo de expiração. Por exemplo, 5 dias é igual a 432.000 segundos. No entanto, muitas vezes é preferível começar com uma data e trabalhar a partir desse ponto.

É muito simples obter a hora atual no formato de hora epoch, como nos seguintes exemplos.

- Terminal Linux: `date +%`
- Python: `import time; int(time.time())`
- Java: `System.currentTimeMillis() / 1000L`
- JavaScript: `Math.floor(Date.now() / 1000)`

Trabalho com itens: Java

Você pode usar o AWS SDK for Java API de documento do para realizar operações create, read, update e delete (CRUD - criação, leitura, atualização e exclusão) típicas em itens do Amazon DynamoDB em uma tabela.

Note

O SDK for Java também oferece um modelo de persistência de objeto, permitindo que você mapeie suas classes do lado do cliente para tabelas do DynamoDB. Essa abordagem pode reduzir a quantidade de código que você precisa escrever. Para mais informações, consulte [Java: DynamoDBMapper \(p. 228\)](#).

Esta seção contém exemplos de Java para executar várias ações de itens da API de documento do Java e vários exemplos funcionais completos.

Tópicos

- [Inserção de um item \(p. 446\)](#)
- [Obter um item \(p. 449\)](#)
- [Gravação em Batch: Como inserir e excluir vários itens \(p. 451\)](#)
- [Obter Batch: Obter vários itens \(p. 452\)](#)
- [Atualizar um item \(p. 453\)](#)
- [Excluir um item \(p. 455\)](#)
- [Exemplo: Operações CRUD usando o AWS SDK for JavaDocumentar API \(p. 456\)](#)
- [Exemplo: Operações em Batch usando o AWS SDK for JavaDocumentar API \(p. 459\)](#)
- [Exemplo: Tratamento de atributos do tipo binário usando o AWS SDK for JavaDocumentar API \(p. 463\)](#)

Inserção de um item

O método `putItem` armazena um item em uma tabela. Se o item existe, ele substitui o item inteiro. Em vez de substituir o item inteiro, se você quiser atualizar apenas atributos específicos, use o método `updateItem`. Para mais informações, consulte [Atualizar um item \(p. 453\)](#).

Siga estas etapas:

1. Crie uma instância da classe `DynamoDB`.
2. Crie uma instância da classe `Table` para representar a tabela com a qual você deseja trabalhar.
3. Crie uma instância da classe `Item` para representar o novo item. Você deve especificar a chave primária do novo item e seus atributos.

4. Chame o método `putItem` do objeto `Table`, usando o `Item` que você criou na etapa anterior.

O exemplo de código Java a seguir demonstra as tarefas anteriores. O código grava um novo item na tabela `ProductCatalog`.

Example

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("ProductCatalog");

// Build a list of related items
List<Number> relatedItems = new ArrayList<Number>();
relatedItems.add(341);
relatedItems.add(472);
relatedItems.add(649);

//Build a map of product pictures
Map<String, String> pictures = new HashMap<String, String>();
pictures.put("FrontView", "http://example.com/products/123_front.jpg");
pictures.put("RearView", "http://example.com/products/123_rear.jpg");
pictures.put("SideView", "http://example.com/products/123_left_side.jpg");

//Build a map of product reviews
Map<String, List<String>> reviews = new HashMap<String, List<String>>();

List<String> fiveStarReviews = new ArrayList<String>();
fiveStarReviews.add("Excellent! Can't recommend it highly enough! Buy it!");
fiveStarReviews.add("Do yourself a favor and buy this");
reviews.put("FiveStar", fiveStarReviews);

List<String> oneStarReviews = new ArrayList<String>();
oneStarReviews.add("Terrible product! Do not buy this.");
reviews.put("OneStar", oneStarReviews);

// Build the item
Item item = new Item()
    .withPrimaryKey("Id", 123)
    .withString("Title", "Bicycle 123")
    .withString("Description", "123 description")
    .withString("BicycleType", "Hybrid")
    .withString("Brand", "Brand-Company C")
    .withNumber("Price", 500)
    .withStringSet("Color", new HashSet<String>(Arrays.asList("Red", "Black")))
    .withString("ProductCategory", "Bicycle")
    .withBoolean("InStock", true)
    .withNull("QuantityOnHand")
    .withList("RelatedItems", relatedItems)
    .withMap("Pictures", pictures)
    .withMap("Reviews", reviews);

// Write the item to the table
PutItemOutcome outcome = table.putItem(item);
```

No exemplo anterior, o item tem atributos que são escalares (`String`, `Number`, `Boolean`, `Null`), conjuntos (`String Set`) e tipos de documento (`List`, `Map`).

Especificação de parâmetros opcionais

Além dos parâmetros obrigatórios, você também pode especificar parâmetros opcionais para o método `putItem`. Por exemplo, o seguinte exemplo de código Java usa um parâmetro opcional para especificar

uma condição para fazer upload do item. Se a condição que você especificar não for atendida, o AWS SDK for Java lançará uma `ConditionalCheckFailedException`. O exemplo de código especifica os seguintes parâmetros opcionais no método `putItem`:

- Uma `ConditionExpression` que define as condições para a solicitação. O código define a condição de que o item existente com a mesma chave primária será substituído somente se tiver um atributo ISBN igual a um valor específico.
- Um mapa para `ExpressionAttributeValues` que é usado na condição. Nesse caso, há somente uma substituição necessária: O espaço reservado `:val` na expressão de condição é substituída no tempo de execução pelo valor ISBN real a ser verificado.

O exemplo a seguir adiciona um novo item de livro usando esses parâmetros opcionais.

Example

```
Item item = new Item()
    .withPrimaryKey("Id", 104)
    .withString("Title", "Book 104 Title")
    .withString("ISBN", "444-4444444444")
    .withNumber("Price", 20)
    .withStringSet("Authors",
        new HashSet<String>(Arrays.asList("Author1", "Author2")));

Map<String, Object> expressionAttributeValues = new HashMap<String, Object>();
expressionAttributeValues.put(":val", "444-4444444444");

PutItemOutcome outcome = table.putItem(
    item,
    "ISBN = :val", // ConditionExpression parameter
    null,           // ExpressionAttributeNames parameter - we're not using it for this
example
    expressionAttributeValues);
```

PutItem e documentos JSON

Você pode armazenar um documento JSON como um atributo em uma tabela do DynamoDB. Para fazer isso, use o método `withJSON` de `Item`. Esse método analisa o documento JSON e mapeia cada elemento para um tipo de dados nativo do DynamoDB.

Suponha que você quisesse armazenar o seguinte documento JSON que contém os fornecedores que podem atender a pedidos de um determinado produto.

Example

```
{
    "V01": {
        "Name": "Acme Books",
        "Offices": [ "Seattle" ]
    },
    "V02": {
        "Name": "New Publishers, Inc.",
        "Offices": [ "London", "New York" ]
    },
    "V03": {
        "Name": "Better Buy Books",
        "Offices": [ "Tokyo", "Los Angeles", "Sydney" ]
    }
}
```

```
}
```

Você pode usar o método `withJSON` para armazenar essas informações na tabela `ProductCatalog`, em um atributo Map chamado `VendorInfo`. O exemplo de código Java a seguir demonstra como fazer isso.

```
// Convert the document into a String. Must escape all double-quotes.
String vendorDocument = "{" +
    + "    \"V01\": {" +
    + "        \"Name\": \"Acme Books\","
    + "        \"Offices\": [ \"Seattle\"]"
    + "    },"
    + "    \"V02\": {" +
    + "        \"Name\": \"New Publishers, Inc.\","
    + "        \"Offices\": [ \"London\", \"New York\" ]"
    + "    },"
    + "    \"V03\": {" +
    + "        \"Name\": \"Better Buy Books\","
    + "        \"Offices\": [ \"Tokyo\", \"Los Angeles\", \"Sydney\" ]"
    + "    }"
    + "};"

Item item = new Item()
    .withPrimaryKey("Id", 210)
    .withString("Title", "Book 210 Title")
    .withString("ISBN", "210-2102102102")
    .withNumber("Price", 30)
    .withJSON("VendorInfo", vendorDocument);

PutItemOutcome outcome = table.putItem(item);
```

Obter um item

Para recuperar um único item, use o método `getItem` de um objeto `Table`. Siga estas etapas:

1. Crie uma instância da classe `DynamoDB`.
2. Crie uma instância da classe `Table` para representar a tabela com a qual você deseja trabalhar.
3. Chame o método `getItem` de instância de `Table`. Você deve especificar a chave primária do item que deseja recuperar.

O exemplo de código Java a seguir demonstra as etapas anteriores. O código obtém o item que tem a chave de partição especificada.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("ProductCatalog");

Item item = table.getItem("Id", 210);
```

Especificação de parâmetros opcionais

Além dos parâmetros obrigatórios, você também pode especificar parâmetros opcionais do método `getItem`. Por exemplo, o seguinte exemplo de código Java usa um método opcional para recuperar somente uma lista específica de atributos e para especificar leituras fortemente consistentes. (Para saber mais sobre a consistência de leitura, consulte [Consistência de leituras \(p. 17\)](#).)

Você pode usar uma `ProjectionExpression` para recuperar somente atributos ou elementos específicos, em vez de todo o item. A `ProjectionExpression` pode especificar atributos aninhados

ou de alto nível, usando caminhos de documentos. Para mais informações, consulte [Expressões de projeção \(p. 417\)](#).

Os parâmetros do método `getItem` não permitem que você especifique consistência de leitura. No entanto, você pode criar uma `GetItemSpec` que fornece acesso total a todas as entradas para a operação `GetItem` de baixo nível. O exemplo de código a seguir cria uma `GetItemSpec` e usa essa especificação como entrada para o método `getItem`.

Example

```
GetItemSpec spec = new GetItemSpec()
    .withPrimaryKey("Id", 206)
    .withProjectionExpression("Id, Title, RelatedItems[0], Reviews.FiveStar")
    .withConsistentRead(true);

Item item = table.getItem(spec);

System.out.println(item.toJSONString());
```

Para imprimir um `Item` em um formato legível, use o método `toJSONPretty`. A saída do exemplo anterior é semelhante à seguinte.

```
{
    "RelatedItems" : [ 341 ],
    "Reviews" : {
        "FiveStar" : [ "Excellent! Can't recommend it highly enough! Buy it!", "Do yourself a
favor and buy this" ]
    },
    "Id" : 123,
    "Title" : "20-Bicycle 123"
}
```

GetItem e documentos JSON

Na seção [PutItem e documentos JSON \(p. 448\)](#), você armazenou um documento JSON em um atributo `Map` chamado `VendorInfo`. Você pode usar o método `getItem` para recuperar o documento inteiro no formato JSON. Ou pode usar a notação do caminho do documento para recuperar apenas alguns dos elementos no documento. O exemplo de código Java a seguir demonstra essas técnicas.

```
GetItemSpec spec = new GetItemSpec()
    .withPrimaryKey("Id", 210);

System.out.println("All vendor info:");
spec.withProjectionExpression("VendorInfo");
System.out.println(table.getItem(spec).toJSON());

System.out.println("A single vendor:");
spec.withProjectionExpression("VendorInfo.V03");
System.out.println(table.getItem(spec).toJSON());

System.out.println("First office location for this vendor:");
spec.withProjectionExpression("VendorInfo.V03.Offices[0]");
System.out.println(table.getItem(spec).toJSON());
```

A saída do exemplo anterior é semelhante à seguinte.

```
All vendor info:
```

```
{"VendorInfo": {"V03": {"Name": "Better Buy Books", "Offices": ["Tokyo", "Los Angeles", "Sydney"]}, "V02": {"Name": "New Publishers, Inc.", "Offices": ["London", "New York"]}, "V01": {"Name": "Acme Books", "Offices": ["Seattle"]}}}  
A single vendor:  
{"VendorInfo": {"V03": {"Name": "Better Buy Books", "Offices": ["Tokyo", "Los Angeles", "Sydney"]}}}  
First office location for a single vendor:  
{"VendorInfo": {"V03": {"Offices": ["Tokyo"]}}}
```

Note

Você pode usar o método `toJSON` para converter qualquer item (ou seus atributos) em uma string formatada para JSON. O exemplo de código a seguir recupera vários atributos aninhados e de alto nível e imprime os resultados como JSON.

```
GetItemSpec spec = new GetItemSpec()  
    .withPrimaryKey("Id", 210)  
    .withProjectionExpression("VendorInfo.V01, Title, Price");  
  
Item item = table.getItem(spec);  
System.out.println(item.toJSON());
```

A saída é semelhante à seguinte.

```
{"VendorInfo": {"V01": {"Name": "Acme Books", "Offices": ["Seattle"]}}, "Price": 30, "Title": "Book 210 Title"}
```

Gravação em Batch: Como inserir e excluir vários itens

Gravação em lote se refere a inserir e excluir vários itens em um lote. O método `batchWriteItem` permite que você insira e exclua vários itens de uma ou mais tabelas em uma única chamada. Veja a seguir as etapas para inserir ou excluir vários itens usando a API de documento do AWS SDK for Java.

1. Crie uma instância da classe `DynamoDB`.
2. Crie uma instância da classe `TableWriteItems` que descreve todas as operações Put e Delete de uma tabela. Se você quiser gravar em várias tabelas com uma única operação de gravação em lote, crie uma instância de `TableWriteItems` por tabela.
3. Chame o método `batchWriteItem`, fornecendo os objetos `TableWriteItems` que você criou na etapa anterior.
4. Processe a resposta. Você deve verificar se há itens de solicitação não processados retornados na resposta. Isso poderá acontecer se você atingir a cota de taxa de transferência provisionada ou por algum outro erro temporário. Além disso, o DynamoDB limita o tamanho da solicitação e o número de operações que você pode especificar em uma solicitação. Se você exceder esses limites, o DynamoDB rejeitará a solicitação. Para mais informações, consulte [Cotas de serviço, conta e tabela no Amazon DynamoDB \(p. 1056\)](#).

O exemplo de código Java a seguir demonstra as etapas anteriores. O exemplo executa uma operação `batchWriteItem` em duas tabelas: `Forum` e `Thread`. Os objetos `TableWriteItems` correspondentes definem as seguintes ações:

- Inserir um item na tabela `Forum`.
- Inserir e excluir um item na tabela `Thread`.

O código chama `batchWriteItem` para executar a operação.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

TableWriteItems forumTableWriteItems = new TableWriteItems("Forum")
    .withItemsToPut(
        new Item()
            .withPrimaryKey("Name", "Amazon RDS")
            .withNumber("Threads", 0));

TableWriteItems threadTableWriteItems = new TableWriteItems("Thread")
    .withItemsToPut(
        new Item()
            .withPrimaryKey("ForumName", "Amazon RDS", "Subject", "Amazon RDS Thread 1")
            .withHashAndRangeKeysToDelete("ForumName", "Some partition key value", "Amazon S3",
                "Some sort key value"));

BatchWriteItemOutcome outcome = dynamoDB.batchWriteItem(forumTableWriteItems,
    threadTableWriteItems);

// Code for checking unprocessed items is omitted in this example
```

Para obter um exemplo de trabalho, consulte [Exemplo: Operação de gravação em Batch usando o AWS SDK for Java Documentar API \(p. 460\)](#).

Obter Batch: Obter vários itens

O método `batchGetItem` permite que você recupere vários itens de uma ou mais tabelas. Para recuperar um único item, você pode usar o método `getItem`.

Siga estas etapas:

1. Crie uma instância da classe `DynamoDB`.
2. Crie uma instância da classe `TableKeysAndAttributes` que descreve uma lista de valores de chave primária para recuperar de uma tabela. Se você desejar ler em várias tabelas em uma única operação de aquisição em lote, será necessário criar uma instância de `TableKeysAndAttributes` por tabela.
3. Chame o método `batchGetItem`, fornecendo os objetos `TableKeysAndAttributes` que você criou na etapa anterior.

O exemplo de código Java a seguir demonstra as etapas anteriores. O exemplo recupera dois itens da tabela `Forum` e três itens da tabela `Thread`.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

TableKeysAndAttributes forumTableKeysAndAttributes = new
TableKeysAndAttributes(forumTableName);
forumTableKeysAndAttributes.addHashOnlyPrimaryKeys("Name",
    "Amazon S3",
    "Amazon DynamoDB");

TableKeysAndAttributes threadTableKeysAndAttributes = new
TableKeysAndAttributes(threadTableName);
threadTableKeysAndAttributes.addHashAndRangePrimaryKeys("ForumName", "Subject",
    "Amazon DynamoDB", "DynamoDB Thread 1",
    "Amazon DynamoDB", "DynamoDB Thread 2",
    "Amazon S3", "S3 Thread 1");

BatchGetItemOutcome outcome = dynamoDB.batchGetItem(
    forumTableKeysAndAttributes, threadTableKeysAndAttributes);
```

```
for (String tableName : outcome.getTableItems().keySet()) {  
    System.out.println("Items in table " + tableName);  
    List<Item> items = outcome.getTableItems().get(tableName);  
    for (Item item : items) {  
        System.out.println(item);  
    }  
}
```

Especificação de parâmetros opcionais

Além dos parâmetros obrigatórios, você também pode especificar parâmetros opcionais ao usar `batchGetItem`. Por exemplo, você pode fornecer uma `ProjectionExpression` com cada `TableKeysAndAttributes` que você definir. Isso permite que você especifique os atributos que deseja recuperar da tabela.

O exemplo de código a seguir recupera dois itens da tabela `Forum`. O parâmetro `withProjectionExpression` especifica que apenas o atributo `Threads` deve ser recuperado.

Example

```
TableKeysAndAttributes forumTableKeysAndAttributes = new TableKeysAndAttributes("Forum")  
    .withProjectionExpression("Threads");  
  
forumTableKeysAndAttributes.addHashOnlyPrimaryKeys("Name",  
    "Amazon S3",  
    "Amazon DynamoDB");  
  
BatchGetItemOutcome outcome = dynamoDB.batchGetItem(forumTableKeysAndAttributes);
```

Atualizar um item

O método `updateItem` de um objeto `Table` pode atualizar valores de atributo existentes, adicionar novos atributos ou excluir atributos de um item existente.

O método `updateItem` se comporta da seguinte forma:

- Se não existir um item (não houver nenhum item na tabela com a chave primária especificada), `updateItem` adicionará um novo item à tabela
- Se existir um item, `updateItem` executará a atualização conforme especificado pelo parâmetro `UpdateExpression`.

Note

Também é possível "atualizar" um item usando `putItem`. Por exemplo, se você chamar `putItem` para adicionar um item à tabela, mas já existir um item com a chave primária especificada, `putItem` substituirá o item inteiro. Se houver atributos no item existente que não são especificados na entrada, `putItem` removerá esses atributos do item.

Em geral, recomendamos que você use `updateItem` sempre que desejar modificar quaisquer atributos de item. O método `updateItem` só modifica os atributos de item que você especifica na entrada, e os outros atributos no item permanecem inalterados.

Siga estas etapas:

1. Crie uma instância da classe `Table` para representar a tabela com a qual você deseja trabalhar.
2. Chame o método `updateTable` de instância de `Table`. Você deve especificar a chave primária do item que deseja recuperar, juntamente com uma `UpdateExpression` que descreve os atributos a serem modificados e como modificá-los.

O exemplo de código Java a seguir demonstra as tarefas anteriores. O código atualiza um item na tabela `ProductCatalog`. Ele adiciona um novo autor ao conjunto de `Authors` e exclui o atributo `ISBN` existente. Ele também reduz o preço por um.

Um mapa `ExpressionAttributeValues` é usado na `UpdateExpression`. Os espaços reservados `:val1` e `:val2` serão substituídos em tempo de execução pelos valores reais de `Authors` e `Price`.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("ProductCatalog");

Map<String, String> expressionAttributeNames = new HashMap<String, String>();
expressionAttributeNames.put("#A", "Authors");
expressionAttributeNames.put("#P", "Price");
expressionAttributeNames.put("#I", "ISBN");

Map<String, Object> expressionAttributeValues = new HashMap<String, Object>();
expressionAttributeValues.put(":val1",
    new HashSet<String>(Arrays.asList("Author YY","Author ZZ")));
expressionAttributeValues.put(":val2", 1); //Price

UpdateItemOutcome outcome = table.updateItem(
    "Id",           // key attribute name
    101,            // key attribute value
    "add #A :val1 set #P = #P - :val2 remove #I", // UpdateExpression
    expressionAttributeNames,
    expressionAttributeValues);
```

Especificação de parâmetros opcionais

Além dos parâmetros obrigatórios, você também pode especificar parâmetros opcionais para o método `updateItem`, incluindo uma condição que deve ser atendida para que a atualização ocorra. Se a condição que você especificar não for atendida, o AWS SDK for Java lançará uma `ConditionalCheckFailedException`. Por exemplo, o seguinte exemplo de código Java atualiza condicionalmente o preço de item de um livro para 25. Ele especifica uma `ConditionExpression` que declara que o preço deve ser atualizado somente se o preço existente for 20.

Example

```
Table table = dynamoDB.getTable("ProductCatalog");

Map<String, String> expressionAttributeNames = new HashMap<String, String>();
expressionAttributeNames.put("#P", "Price");

Map<String, Object> expressionAttributeValues = new HashMap<String, Object>();
expressionAttributeValues.put(":val1", 25); // update Price to 25...
expressionAttributeValues.put(":val2", 20); //...but only if existing Price is 20

UpdateItemOutcome outcome = table.updateItem(
    new PrimaryKey("Id",101),
    "set #P = :val1", // UpdateExpression
    "#P = :val2",    // ConditionExpression
    expressionAttributeNames,
    expressionAttributeValues);
```

Contador atômico

Você pode usar `updateItem` para implementar um contador atômico, onde pode aumentar ou reduzir o valor de um atributo existente sem interferir em outras solicitações de gravação. Para aumentar um

contador atômico, use uma `UpdateExpression` com uma ação `set` para adicionar um valor numérico a um atributo existente do tipo `Number`.

O exemplo de código a seguir demonstra isso incrementando o atributo `Quantity` em um. Ele também demonstra o uso do parâmetro `ExpressionAttributeNames` em uma `UpdateExpression`.

```
Table table = dynamoDB.getTable("ProductCatalog");

Map<String, String> expressionAttributeNames = new HashMap<String, String>();
expressionAttributeNames.put("#p", "PageCount");

Map<String, Object> expressionAttributeValues = new HashMap<String, Object>();
expressionAttributeValues.put(":val", 1);

UpdateItemOutcome outcome = table.updateItem(
    "Id", 121,
    "set #p = #p + :val",
    expressionAttributeNames,
    expressionAttributeValues);
```

Excluir um item

O método `deleteItem` exclui um item de uma tabela. É necessário fornecer a chave primária do item que você deseja excluir.

Siga estas etapas:

1. Crie uma instância do cliente DynamoDB.
2. Chame o método `deleteItem`, fornecendo a chave do item que você deseja excluir.

O exemplo de código Java a seguir demonstra essas tarefas.

Example

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("ProductCatalog");

DeleteItemOutcome outcome = table.deleteItem("Id", 101);
```

Especificação de parâmetros opcionais

Você pode especificar parâmetros opcionais para `deleteItem`. Por exemplo, o seguinte exemplo de código Java especifica uma `ConditionExpression` que declara que um item de livro em `ProductCatalog` só pode ser excluído se o livro não estiver mais em publicação (o atributo `InPublication` é falso).

Example

```
Map<String, Object> expressionAttributeValues = new HashMap<String, Object>();
expressionAttributeValues.put(":val", false);

DeleteItemOutcome outcome = table.deleteItem("Id", 103,
    "InPublication = :val",
    null, // ExpressionAttributeNames - not used in this example
    expressionAttributeValues);
```

Exemplo: Operações CRUD usando oAWS SDK for JavaDocumentar API

O exemplo de código a seguir ilustra operações CRUD em um item do Amazon DynamoDB. O exemplo cria um item, o recupera, executa várias atualizações e, por fim, o exclui.

Note

O SDK for Java também oferece um modelo de persistência de objeto, permitindo que você mapeie suas classes do lado do cliente para tabelas do DynamoDB. Essa abordagem pode reduzir a quantidade de código que você precisa escrever. Para mais informações, consulte [Java: DynamoDBMapper \(p. 228\)](#).

Note

Este exemplo de código pressupõe que você já carregou dados no DynamoDB para sua conta seguindo as instruções no[Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#)seção.

Para obter instruções detalhadas sobre como executar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples.document;

import java.io.IOException;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DeleteItemOutcome;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Item;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.document.UpdateItemOutcome;
import com.amazonaws.services.dynamodbv2.document.spec.DeleteItemSpec;
import com.amazonaws.services.dynamodbv2.document.spec.UpdateItemSpec;
import com.amazonaws.services.dynamodbv2.document.utils.NameMap;
import com.amazonaws.services.dynamodbv2.document.utils.ValueMap;
import com.amazonaws.services.dynamodbv2.model.ReturnValue;

public class DocumentAPIItemCRUDExample {

    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
    static DynamoDB dynamoDB = new DynamoDB(client);
```

```
static String tableName = "ProductCatalog";

public static void main(String[] args) throws IOException {
    createItems();
    retrieveItem();

    // Perform various updates.
    updateMultipleAttributes();
    updateAddNewAttribute();
    updateExistingAttributeConditionally();

    // Delete the item.
    deleteItem();
}

private static void createItems() {
    Table table = dynamoDB.getTable(tableName);
    try {

        Item item = new Item().withPrimaryKey("Id", 120).withString("Title", "Book 120
Title")
            .withString("ISBN", "120-1111111111")
            .withStringSet("Authors", new HashSet<String>(Arrays.asList("Author12",
"Author22")))
            .withNumber("Price", 20).withString("Dimensions",
"8.5x11.0x.75").withNumber("PageCount", 500)
            .withBoolean("InPublication", false).withString("ProductCategory", "Book");
        table.putItem(item);

        item = new Item().withPrimaryKey("Id", 121).withString("Title", "Book 121
Title")
            .withString("ISBN", "121-1111111111")
            .withStringSet("Authors", new HashSet<String>(Arrays.asList("Author21",
"Author 22")))
            .withNumber("Price", 20).withString("Dimensions",
"8.5x11.0x.75").withNumber("PageCount", 500)
            .withBoolean("InPublication", true).withString("ProductCategory", "Book");
        table.putItem(item);

    }
    catch (Exception e) {
        System.err.println("Create items failed.");
        System.err.println(e.getMessage());
    }
}

private static void retrieveItem() {
    Table table = dynamoDB.getTable(tableName);
    try {

        Item item = table.getItem("Id", 120, "Id, ISBN, Title, Authors", null);

        System.out.println("Printing item after retrieving it....");
        System.out.println(item.toJSONString());

    }
    catch (Exception e) {
        System.err.println("GetItem failed.");
        System.err.println(e.getMessage());
    }
}
```

```
}

private static void updateAddNewAttribute() {
    Table table = dynamoDB.getTable(tableName);

    try {

        UpdateItemSpec updateItemSpec = new UpdateItemSpec().withPrimaryKey("Id", 121)
            .withUpdateExpression("set #na = :val1").withNameMap(new
        NameMap().with("#na", "NewAttribute"))
            .WithValueMap(new ValueMap().withString(":val1", "Some
        value")).withReturnValues(ReturnValue.ALL_NEW);

        UpdateItemOutcome outcome = table.updateItem(updateItemSpec);

        // Check the response.
        System.out.println("Printing item after adding new attribute...");
        System.out.println(outcome.getItem().toJSONPretty());

    }
    catch (Exception e) {
        System.err.println("Failed to add new attribute in " + tableName);
        System.err.println(e.getMessage());
    }
}

private static void updateMultipleAttributes() {

    Table table = dynamoDB.getTable(tableName);

    try {

        UpdateItemSpec updateItemSpec = new UpdateItemSpec().withPrimaryKey("Id", 120)
            .withUpdateExpression("add #a :val1 set #na=:val2")
            .withNameMap(new NameMap().with("#a", "Authors").with("#na",
        "NewAttribute"))
            .WithValueMap(
                new ValueMap().withStringSet(":val1", "Author YY", "Author
        ZZ").withString(":val2", "someValue"))
            .withReturnValues(ReturnValue.ALL_NEW);

        UpdateItemOutcome outcome = table.updateItem(updateItemSpec);

        // Check the response.
        System.out.println("Printing item after multiple attribute update...");
        System.out.println(outcome.getItem().toJSONPretty());

    }
    catch (Exception e) {
        System.err.println("Failed to update multiple attributes in " + tableName);
        System.err.println(e.getMessage());
    }
}

private static void updateExistingAttributeConditionally() {

    Table table = dynamoDB.getTable(tableName);

    try {

        // Specify the desired price (25.00) and also the condition (price =
        // 20.00)

        UpdateItemSpec updateItemSpec = new UpdateItemSpec().withPrimaryKey("Id", 120)
```

```
        .withReturnValues(ReturnValue.ALL_NEW).withUpdateExpression("set #p
= :val1")
        .withConditionExpression("#p = :val2").withNameMap(new NameMap().with("#p",
"Price"))
        .WithValueMap(new ValueMap().withNumber(":val1", 25).withNumber(":val2",
20));

        UpdateItemOutcome outcome = table.updateItem(updateItemSpec);

        // Check the response.
        System.out.println("Printing item after conditional update to new
attribute...");
        System.out.println(outcome.getItem().toJSONPretty());

    }
    catch (Exception e) {
        System.err.println("Error updating item in " + tableName);
        System.err.println(e.getMessage());
    }
}

private static void deleteItem() {

    Table table = dynamoDB.getTable(tableName);

    try {

        DeleteItemSpec deleteItemSpec = new DeleteItemSpec().withPrimaryKey("Id", 120)
            .withConditionExpression("#ip = :val").withNameMap(new
NameMap().with("#ip", "InPublication"))
            .WithValueMap(new ValueMap().withBoolean(":val",
false)).withReturnValues(ReturnValue.ALL_OLD);

        DeleteItemOutcome outcome = table.deleteItem(deleteItemSpec);

        // Check the response.
        System.out.println("Printing item that was deleted...");
        System.out.println(outcome.getItem().toJSONPretty());

    }
    catch (Exception e) {
        System.err.println("Error deleting item in " + tableName);
        System.err.println(e.getMessage());
    }
}
}
```

Exemplo: Operações em Batch usando oAWS SDK for JavaDocumentar API

Esta seção fornece exemplos de operações de gravação e aquisição em lote no Amazon DynamoDB usando oAWS SDK for JavaDocumentar API.

Note

O SDK for Java também oferece um modelo de persistência de objeto, permitindo que você mapeie suas classes do lado do cliente para tabelas do DynamoDB. Essa abordagem pode reduzir a quantidade de código que você precisa escrever. Para mais informações, consulte [Java: DynamoDBMapper \(p. 228\)](#).

Tópicos

- [Exemplo: Operação de gravação em Batch usando oAWS SDK for JavaDocumentar API \(p. 460\)](#)

- Exemplo: Operação de aquisição em Batch usando oAWS SDK for JavaDocumentar API (p. 462)

Exemplo: Operação de gravação em Batch usando oAWS SDK for JavaDocumentar API

O exemplo de código Java a seguir usa o método `batchWriteItem` para executar as operações Put e Delete a seguir:

- Inserir um item na tabela `Forum`.
- Inserir e excluir um item da tabela `Thread`.

Você pode especificar quantas solicitações de inserção e exclusão desejar em uma ou mais tabelas ao criar sua solicitação de gravação em lote. No entanto, `batchWriteItem` limita o tamanho de uma solicitação de gravação em lote e o número de operações Put e Delete em uma única operação. Se a sua solicitação ultrapassar esses limites, ela será rejeitada. Se a sua tabela não tiver throughput provisionado suficiente para servir a essa solicitação, os itens de solicitação não processados serão retornados na resposta.

O exemplo a seguir verifica a resposta para conferir se existem itens de solicitação não processados. Se houver, ele retorna e reenvia a solicitação `batchWriteItem` com itens não processados na solicitação. Se você tiver seguido a seção [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#), já deverá ter criado as tabelas `Forum` e `Thread`. Você também pode criar essas tabelas e fazer upload dos dados de exemplo de forma programática. Para mais informações, consulte [Como criar tabelas de exemplo e carregar dados usando oAWS SDK for Java \(p. 1080\)](#).

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
package com.amazonaws.codesamples.document;  
  
import java.io.IOException;  
import java.util.Arrays;  
import java.util.HashSet;  
import java.util.List;  
import java.util.Map;  
  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.BatchWriteItemOutcome;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.TableWriteItems;  
import com.amazonaws.services.dynamodbv2.model.WriteRequest;
```

```
public class DocumentAPIBatchWrite {

    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
    static DynamoDB dynamoDB = new DynamoDB(client);

    static String forumTableName = "Forum";
    static String threadTableName = "Thread";

    public static void main(String[] args) throws IOException {
        writeMultipleItemsBatchWrite();
    }

    private static void writeMultipleItemsBatchWrite() {
        try {

            // Add a new item to Forum
            TableWriteItems forumTableWriteItems = new TableWriteItems(forumTableName) // Forum
                .withItemsToPut(new Item().withPrimaryKey("Name", "Amazon RDS").withNumber("Threads", 0));

            // Add a new item, and delete an existing item, from Thread
            // This table has a partition key and range key, so need to specify
            // both of them
            TableWriteItems threadTableWriteItems = new TableWriteItems(threadTableName)
                .withItemsToPut(new Item().withPrimaryKey("ForumName", "Amazon RDS", "Subject", "Amazon RDS Thread 1")
                    .withString("Message", "ElastiCache Thread 1 message")
                    .withStringSet("Tags", new HashSet<String>(Arrays.asList("cache", "in-memory"))))
                .withHashAndRangeKeysToDelete("ForumName", "Subject", "Amazon S3", "S3 Thread 100");

            System.out.println("Making the request.");
            BatchWriteItemOutcome outcome = dynamoDB.batchWriteItem(forumTableWriteItems,
            threadTableWriteItems);

            do {

                // Check for unprocessed keys which could happen if you exceed
                // provisioned throughput

                Map<String, List<WriteRequest>> unprocessedItems =
                outcome.getUnprocessedItems();

                if (outcome.getUnprocessedItems().size() == 0) {
                    System.out.println("No unprocessed items found");
                }
                else {
                    System.out.println("Retrieving the unprocessed items");
                    outcome = dynamoDB.batchWriteItemUnprocessed(unprocessedItems);
                }

            } while (outcome.getUnprocessedItems().size() > 0);

        }
        catch (Exception e) {
            System.err.println("Failed to retrieve items: ");
            e.printStackTrace(System.err);
        }
    }
}
```

```
}
```

Exemplo: Operação de aquisição em Batch usando oAWS SDK for JavaDocumentar API

O exemplo de código Java a seguir usa o método `batchGetItem` para recuperar vários itens das tabelas `Forum` e `Thread`. A `BatchGetItemRequest` especifica os nomes de tabela e uma lista de chaves de cada item a ser obtido. O exemplo processa a resposta, imprimindo os itens recuperados.

Note

Este exemplo de código pressupõe que você já carregou dados no DynamoDB para sua conta seguindo as instruções no[Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#)seção.

Para obter instruções detalhadas sobre como executar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
package com.amazonaws.codesamples.document;  
  
import java.io.IOException;  
import java.util.List;  
import java.util.Map;  
  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.BatchGetItemOutcome;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.TableKeysAndAttributes;  
import com.amazonaws.services.dynamodbv2.model.KeysAndAttributes;  
  
public class DocumentAPIBatchGet {  
    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();  
    static DynamoDB dynamoDB = new DynamoDB(client);  
  
    static String forumTableName = "Forum";  
    static String threadTableName = "Thread";  
  
    public static void main(String[] args) throws IOException {  
        retrieveMultipleItemsBatchGet();  
    }  
  
    private static void retrieveMultipleItemsBatchGet() {  
  
        try {
```

```
        TableKeysAndAttributes forumTableKeysAndAttributes = new
TableKeysAndAttributes(forumTableName);
        // Add a partition key
        forumTableKeysAndAttributes.addHashOnlyPrimaryKeys("Name", "Amazon S3", "Amazon
DynamoDB");

        TableKeysAndAttributes threadTableKeysAndAttributes = new
TableKeysAndAttributes(threadTableName);
        // Add a partition key and a sort key
        threadTableKeysAndAttributes.addHashAndRangePrimaryKeys("ForumName", "Subject",
"Amazon DynamoDB",
        "DynamoDB Thread 1", "Amazon DynamoDB", "DynamoDB Thread 2", "Amazon S3",
"S3 Thread 1");

        System.out.println("Making the request.");

        BatchGetItemOutcome outcome =
dynamoDB.batchGetItem(forumTableKeysAndAttributes,
        threadTableKeysAndAttributes);

        Map<String, KeysAndAttributes> unprocessed = null;

        do {
            for (String tableName : outcome.getTableItems().keySet()) {
                System.out.println("Items in table " + tableName);
                List<Item> items = outcome.getTableItems().get(tableName);
                for (Item item : items) {
                    System.out.println(item.toJSONPretty());
                }
            }

            // Check for unprocessed keys which could happen if you exceed
            // provisioned
            // throughput or reach the limit on response size.
            unprocessed = outcome.getUnprocessedKeys();

            if (unprocessed.isEmpty()) {
                System.out.println("No unprocessed keys found");
            }
            else {
                System.out.println("Retrieving the unprocessed keys");
                outcome = dynamoDB.batchGetItemUnprocessed(unprocessed);
            }
        } while (!unprocessed.isEmpty());

    }
    catch (Exception e) {
        System.err.println("Failed to retrieve items.");
        System.err.println(e.getMessage());
    }
}
```

Exemplo: Tratamento de atributos do tipo binário usando o AWS SDK for Java Documentar API

O exemplo de código Java a seguir ilustra o tratamento de atributos do tipo binário. O exemplo adiciona um item à tabela Reply. O item inclui um atributo do tipo binário (`ExtendedMessage`) que armazena

dados compactados. Em seguida, o exemplo recupera um item e imprime todos os valores do atributo. Para ilustração, o exemplo usa a classe `GZIPOutputStream` para compactar um stream de exemplo e atribuí-lo ao atributo `ExtendedMessage`. Quando o atributo binário é recuperado, ele é descompactado usando a classe `GZIPInputStream`.

Note

O SDK for Java também oferece um modelo de persistência de objeto, permitindo que você mapeie suas classes do lado do cliente para tabelas do DynamoDB. Essa abordagem pode reduzir a quantidade de código que você precisa escrever. Para mais informações, consulte [Java: DynamoDBMapper \(p. 228\)](#).

Se você tiver seguido a seção [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#), já deverá ter criado a tabela `Reply`. Você também pode criar essa tabela de forma programática. Para mais informações, consulte [Como criar tabelas de exemplo e carregar dados usando o AWS SDK for Java \(p. 1080\)](#).

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
package com.amazonaws.codesamples.document;  
  
import java.io.ByteArrayInputStream;  
import java.io.ByteArrayOutputStream;  
import java.io.IOException;  
import java.nio.ByteBuffer;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.TimeZone;  
import java.util.zip.GZIPInputStream;  
import java.util.zip.GZIPOutputStream;  
  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.Table;  
import com.amazonaws.services.dynamodbv2.document.spec.GetItemSpec;  
  
public class DocumentAPIItemBinaryExample {  
  
    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();  
    static DynamoDB dynamoDB = new DynamoDB(client);  
  
    static String tableName = "Reply";  
    static SimpleDateFormat dateFormatter = new SimpleDateFormat("yyyy-MM-  
dd'T'HH:mm:ss.SSS'Z'");  
}
```

```
public static void main(String[] args) throws IOException {
    try {

        // Format the primary key values
        String threadId = "Amazon DynamoDB#DynamoDB Thread 2";

        dateFormatter.setTimeZone(TimeZone.getTimeZone("UTC"));
        String replyDateTime = dateFormatter.format(new Date());

        // Add a new reply with a binary attribute type
        createItem(threadId, replyDateTime);

        // Retrieve the reply with a binary attribute type
        retrieveItem(threadId, replyDateTime);

        // clean up by deleting the item
        deleteItem(threadId, replyDateTime);
    }
    catch (Exception e) {
        System.err.println("Error running the binary attribute type example: " + e);
        e.printStackTrace(System.err);
    }
}

public static void createItem(String threadId, String replyDateTime) throws IOException
{

    Table table = dynamoDB.getTable(tableName);

    // Craft a long message
    String messageInput = "Long message to be compressed in a lengthy forum reply";

    // Compress the long message
    ByteBuffer compressedMessage = compressString(messageInput.toString());

    table.putItem(new Item().withPrimaryKey("Id", threadId).withString("ReplyDateTime",
    replyDateTime)
        .withString("Message", "Long message follows").withBinary("ExtendedMessage",
    compressedMessage)
        .withString("PostedBy", "User A"));
}

public static void retrieveItem(String threadId, String replyDateTime) throws
IOException {

    Table table = dynamoDB.getTable(tableName);

    GetItemSpec spec = new GetItemSpec().withPrimaryKey("Id", threadId,
    "ReplyDateTime", replyDateTime)
        .withConsistentRead(true);

    Item item = table.getItem(spec);

    // Uncompress the reply message and print
    String uncompressed =
uncompressString(ByteBuffer.wrap(item.getBinary("ExtendedMessage")));

    System.out.println("Reply message:\n" + " Id: " + item.getString("Id") + "\n" +
    "ReplyDateTime: "
        + item.getString("ReplyDateTime") + "\n" + " PostedBy: " +
    item.getString("PostedBy") + "\n" + " Message: "
        + item.getString("Message") + "\n" + " ExtendedMessage (uncompressed): " +
    uncompressed + "\n");
}
```

```
public static void deleteItem(String threadId, String replyDateTime) {  
  
    Table table = dynamoDB.getTable(tableName);  
    table.deleteItem("Id", threadId, "ReplyDateTime", replyDateTime);  
}  
  
private static ByteBuffer compressString(String input) throws IOException {  
    // Compress the UTF-8 encoded String into a byte[]  
    ByteArrayOutputStream baos = new ByteArrayOutputStream();  
    GZIPOutputStream os = new GZIPOutputStream(baos);  
    os.write(input.getBytes("UTF-8"));  
    os.close();  
    baos.close();  
    byte[] compressedBytes = baos.toByteArray();  
  
    // The following code writes the compressed bytes to a ByteBuffer.  
    // A simpler way to do this is by simply calling  
    // ByteBuffer.wrap(compressedBytes);  
    // However, the longer form below shows the importance of resetting the  
    // position of the buffer  
    // back to the beginning of the buffer if you are writing bytes directly  
    // to it, since the SDK  
    // will consider only the bytes after the current position when sending  
    // data to DynamoDB.  
    // Using the "wrap" method automatically resets the position to zero.  
    ByteBuffer buffer = ByteBuffer.allocate(compressedBytes.length);  
    buffer.put(compressedBytes, 0, compressedBytes.length);  
    buffer.position(0); // Important: reset the position of the ByteBuffer  
                      // to the beginning  
    return buffer;  
}  
  
private static String uncompressString(ByteBuffer input) throws IOException {  
    byte[] bytes = input.array();  
    ByteArrayInputStream bais = new ByteArrayInputStream(bytes);  
    ByteArrayOutputStream baos = new ByteArrayOutputStream();  
    GZIPInputStream is = new GZIPInputStream(bais);  
  
    int chunkSize = 1024;  
    byte[] buffer = new byte[chunkSize];  
    int length = 0;  
    while ((length = is.read(buffer, 0, chunkSize)) != -1) {  
        baos.write(buffer, 0, length);  
    }  
  
    String result = new String(baos.toByteArray(), "UTF-8");  
  
    is.close();  
    baos.close();  
    bais.close();  
  
    return result;  
}  
}
```

Trabalho com itens: .NET

Você pode usar a API de baixo nível do AWS SDK for .NET para executar operações típicas create, read, update, delete (CRUD - criação, leitura, atualização e exclusão) em um item de uma tabela. Veja a seguir as etapas comuns que você segue para executar operações CRUD em dados usando a API de baixo nível do .NET:

1. Crie uma instância da classe `AmazonDynamoDBClient` (o cliente).
2. Forneça os parâmetros necessários específicos à operação em um objeto de solicitação correspondente.

Por exemplo, use o objeto de solicitação `PutItemRequest` ao carregar um item e use o objeto de solicitação `GetItemRequest` ao recuperar um item existente.

Você pode usar o objeto de solicitação para fornecer tanto parâmetros necessários quanto opcionais.

3. Execute o método apropriado fornecido pelo cliente, transmitindo o objeto de solicitação que você criou na etapa anterior.

O cliente `AmazonDynamoDBClient` fornece os métodos `PutItem`, `GetItem`, `UpdateItem` e `DeleteItem` para as operações CRUD.

Tópicos

- [Inserção de um item \(p. 467\)](#)
- [Obter um item \(p. 469\)](#)
- [Atualizar um item \(p. 470\)](#)
- [Contador atômico \(p. 472\)](#)
- [Excluir um item \(p. 472\)](#)
- [Gravação em Batch: Como inserir e excluir vários itens \(p. 473\)](#)
- [Obter Batch: Obter vários itens \(p. 475\)](#)
- [Exemplo: Operações CRUD usando oAWS SDK for .NETAPI de baixo nível \(p. 477\)](#)
- [Exemplo: Operações em Batch usando oAWS SDK for .NETAPI de baixo nível \(p. 481\)](#)
- [Exemplo: Tratamento de atributos do tipo binário usando oAWS SDK for .NETAPI de baixo nível \(p. 487\)](#)

Inserção de um item

O método `PutItem` carrega um item em uma tabela. Se o item existe, ele substitui o item inteiro.

Note

Em vez de substituir o item inteiro, se você quiser atualizar apenas atributos específicos, use o método `UpdateItem`. Para mais informações, consulte [Atualizar um item \(p. 470\)](#).

Veja a seguir as etapas para carregar um item usando a API do SDK do .NET de baixo nível:

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Forneça os parâmetros necessários, criando uma instância da classe `PutItemRequest`.
Para inserir item, você deve fornecer o nome da tabela e o item.
3. Execute a `PutItem` fornecendo o método `PutItemRequest` que você criou na etapa anterior.

O exemplo de C# a seguir demonstra as etapas anteriores. O exemplo faz upload de um item para a tabela `ProductCatalog`.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "ProductCatalog";
```

```
var request = new PutItemRequest
{
    TableName = tableName,
    Item = new Dictionary<string, AttributeValue>()
    {
        { "Id", new AttributeValue { N = "201" } },
        { "Title", new AttributeValue { S = "Book 201 Title" } },
        { "ISBN", new AttributeValue { S = "11-11-11-11" } },
        { "Price", new AttributeValue { S = "20.00" } },
        {
            "Authors",
            new AttributeValue
            { SS = new List<string>{"Author1", "Author2"} }
        }
    };
    client.PutItem(request);
```

No exemplo anterior, você faz upload de um item de livro que tem os atributos `Id`, `Title`, `ISBN` e `Authors`. Observe que `Id` é um atributo de tipo numérico, e todos os outros atributos são do tipo string. `Autores` é um conjunto String.

Especificação de parâmetros opcionais

Você também pode fornecer parâmetros opcionais usando o objeto `PutItemRequest`, conforme mostrado no seguinte exemplo de C#. O exemplo especifica os seguintes parâmetros opcionais:

- `ExpressionAttributeNames`, `ExpressionAttributeValues` e `ConditionExpression` especificam que o item pode ser substituído somente se o item existente tiver o atributo ISBN com um valor específico.
- o parâmetro `ReturnValues` para solicitar o item antigo na resposta.

Example

```
var request = new PutItemRequest
{
    TableName = tableName,
    Item = new Dictionary<string, AttributeValue>()
    {
        { "Id", new AttributeValue { N = "104" } },
        { "Title", new AttributeValue { S = "Book 104 Title" } },
        { "ISBN", new AttributeValue { S = "444-4444444444" } },
        { "Authors",
            new AttributeValue { SS = new List<string>{"Author3"} }
        },
        // Optional parameters.
        ExpressionAttributeNames = new Dictionary<string,string>()
        {
            { "#I", "ISBN" }
        },
        ExpressionAttributeValues = new Dictionary<string, AttributeValue>()
        {
            { ":isbn", new AttributeValue { S = "444-4444444444" } }
        },
        ConditionExpression = "#I = :isbn"
    };
    var response = client.PutItem(request);
```

Para obter mais informações, consulte [PutItem](#).

Obter um item

O método `GetItem` recupera um item.

Note

Para recuperar vários itens, você pode usar o método `BatchGetItem`. Para mais informações, consulte [Obter Batch: Obter vários itens \(p. 475\)](#).

Veja a seguir as etapas para recuperar um item existente usando a API do AWS SDK for .NET de baixo nível.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Forneça os parâmetros necessários, criando uma instância da classe `GetItemRequest`.
Para obter um item, você deve fornecer o nome da tabela e a chave primária desse item.
3. Execute a `GetItem` fornecendo o método `GetItemRequest` que você criou na etapa anterior.

O exemplo de C# a seguir demonstra as etapas anteriores. O exemplo recupera um item da tabela `ProductCatalog`.

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "ProductCatalog";

var request = new GetItemRequest
{
    TableName = tableName,
    Key = new Dictionary<string,AttributeValue>() { { "Id", new AttributeValue { N =
    "202" } } },
};
var response = client.GetItem(request);

// Check the response.
var result = response.GetItemResult;
var attributeMap = result.Item; // Attribute list in the response.
```

Especificação de parâmetros opcionais

Você também pode fornecer parâmetros opcionais usando o objeto `GetItemRequest`, conforme mostrado no seguinte exemplo de C#. O exemplo especifica os parâmetros opcionais a seguir:

- o parâmetro `ProjectionExpression` para especificar os atributos a serem recuperados.
- o parâmetro `ConsistentRead` para realizar uma leitura fortemente consistente. Para saber mais sobre a consistência de leitura, consulte [Consistência de leituras \(p. 17\)](#).

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "ProductCatalog";

var request = new GetItemRequest
{
    TableName = tableName,
    Key = new Dictionary<string,AttributeValue>() { { "Id", new AttributeValue { N =
    "202" } } },
    // Optional parameters.
    ProjectionExpression = "Id, ISBN, Title, Authors",
    ConsistentRead = true
```

```
};

var response = client.GetItem(request);

// Check the response.
var result = response.GetItemResult;
var attributeMap = result.Item;
```

Para obter mais informações, consulte[GetItem](#).

Atualizar um item

O método `UpdateItem` atualiza um item existente, se estiver presente. É possível usar a operação `UpdateItem` para atualizar valores de atributos existentes, adicionar novos atributos ou excluir atributos da coleção existente. Se o item que tem a chave primária especificada não for encontrado, ela adicionará um novo item.

A operação `UpdateItem` usa as seguintes diretrizes:

- Se o item não existir, o `UpdateItem` adicionará um novo item usando a chave primária especificada na entrada.
- Se o item existir, o `UpdateItem` aplicará as atualizações da seguinte maneira:
 - Substitui os valores de atributos existentes pelos valores na atualização.
 - Se o atributo que você fornecer na entrada não existir, ele adicionará um novo atributo ao item.
 - Se o valor do atributo de entrada for nulo, ele excluirá o atributo, se estiver presente.
 - Se você usar `ADD` para a `Action`, poderá adicionar valores a um conjunto existente (conjunto de strings ou números) ou adicionar (usar um número positivo) ou subtrair (usar um número negativo) matematicamente com base no valor de atributo numérico existente.

Note

A operação `PutItem` também pode realizar uma atualização. Para mais informações, consulte [Inserção de um item \(p. 467\)](#). Por exemplo, se você chamar `PutItem` para fazer upload de um item, e a chave primária existir, a operação `PutItem` substituirá o item inteiro. Se houver atributos no item existente e esses atributos não forem especificados na entrada, a operação `PutItem` os excluirá. No entanto, `UpdateItem` só atualiza os atributos de entrada especificados. Outros atributos existentes desse item permanecerão inalterados.

Veja a seguir as etapas para atualizar um item existente usando a API do SDK do .NET de baixo nível:

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Forneça os parâmetros necessários, criando uma instância da classe `UpdateItemRequest`.
Este é o objeto de solicitação em que você descreve todas as atualizações, por exemplo, adicionar atributos, atualizar atributos existentes ou excluir atributos. Para excluir um atributo existente, especifique o nome desse atributo com um valor nulo.
3. Execute a `UpdateItem` fornecendo o método `UpdateItemRequest` que você criou na etapa anterior.

O exemplo de código C# a seguir demonstra as etapas anteriores. O exemplo de código atualiza um item na tabela `ProductCatalog`. Ele adiciona um novo autor à coleção `Authors` e exclui o atributo `ISBN` existente. Ele também reduz o preço por um.

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "ProductCatalog";
```

```
var request = new UpdateItemRequest
{
    TableName = tableName,
    Key = new Dictionary<string,AttributeValue>() { { "Id", new AttributeValue { N =
    "202" } } },
    ExpressionAttributeNames = new Dictionary<string,string>()
    {
        {"#A", "Authors"},
        {"#P", "Price"},
        {"#NA", "NewAttribute"},
        {"#I", "ISBN"}
    },
    ExpressionAttributeValues = new Dictionary<string,AttributeValue>()
    {
        {"::auth",new AttributeValue { SS = {"Author YY","Author ZZ"} }},
        {"::p",new AttributeValue { N = "1"}},
        {"::newattr",new AttributeValue { S = "someValue"}},
    },
    // This expression does the following:
    // 1) Adds two new authors to the list
    // 2) Reduces the price
    // 3) Adds a new attribute to the item
    // 4) Removes the ISBN attribute from the item
    UpdateExpression = "ADD #A :auth SET #P = :p, #NA = :newattr REMOVE #I"
};
var response = client.UpdateItem(request);
```

Especificação de parâmetros opcionais

Você também pode fornecer parâmetros opcionais usando o objeto `UpdateItemRequest`, conforme mostrado no seguinte exemplo de C#. Especifica os parâmetros opcionais a seguir:

- `ExpressionAttributeValues` e `ConditionExpression` para especificar que o preço apenas poderá ser atualizado se o preço existente for 20,00.
- o parâmetro `ReturnValues` para solicitar o item atualizado na resposta.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "ProductCatalog";

var request = new UpdateItemRequest
{
    Key = new Dictionary<string,AttributeValue>() { { "Id", new AttributeValue { N =
    "202" } } },

    // Update price only if the current price is 20.00.
    ExpressionAttributeNames = new Dictionary<string,string>()
    {
        {"#P", "Price"}
    },
    ExpressionAttributeValues = new Dictionary<string,AttributeValue>()
    {
        {"::newprice",new AttributeValue { N = "22"}},
        {"::currprice",new AttributeValue { N = "20"}}
    },
    UpdateExpression = "SET #P = :newprice",
    ConditionExpression = "#P = :currprice",
    TableName = tableName,
    ReturnValues = "ALL_NEW" // Return all the attributes of the updated item.
};
```

```
var response = client.UpdateItem(request);
```

Para obter mais informações, consulte [UpdateItem](#).

Contador atômico

Você pode usar `updateItem` para implementar um contador atômico, onde pode aumentar ou reduzir o valor de um atributo existente sem interferir em outras solicitações de gravação. Para atualizar um contador atômico, use `updateItem` com um atributo do tipo `Number` no parâmetro `UpdateExpression` e `ADD` como a `Action`.

O exemplo de código a seguir demonstra isso incrementando o atributo `Quantity` em um.

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "ProductCatalog";

var request = new UpdateItemRequest
{
    Key = new Dictionary<string, AttributeValue>() { { "Id", new AttributeValue { N =
    "121" } } },
    ExpressionAttributeNames = new Dictionary<string, string>()
    {
        {"#Q", "Quantity"}
    },
    ExpressionAttributeValues = new Dictionary<string, AttributeValue>()
    {
        {"::incr", new AttributeValue { N = "1" }}
    },
    UpdateExpression = "SET #Q = #Q + :incr",
    TableName = tableName
};

var response = client.UpdateItem(request);
```

Excluir um item

O método `DeleteItem` exclui um item de uma tabela.

Veja a seguir as etapas para excluir um item usando a API de SDK do .NET de baixo nível.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Forneça os parâmetros necessários, criando uma instância da classe `DeleteItemRequest`.
Para excluir um item, o nome da tabela e a chave primária desse item são necessários.
3. Execute a `DeleteItem` fornecendo o método `DeleteItemRequest` que você criou na etapa anterior.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "ProductCatalog";

var request = new DeleteItemRequest
{
    TableName = tableName,
    Key = new Dictionary<string,AttributeValue>() { { "Id", new AttributeValue { N =
    "201" } } },
};
```

```
var response = client.DeleteItem(request);
```

Especificação de parâmetros opcionais

Você também pode fornecer parâmetros opcionais usando o objeto `DeleteItemRequest`, conforme mostrado no seguinte exemplo de código C#. Especifica os parâmetros opcionais a seguir:

- `ExpressionAttributeValues` e `ConditionExpression` para especificar que o item de livro apenas poderá ser excluído se não estiver mais em publicação (se o valor do atributo `InPublication` for `false`).
- o parâmetro `ReturnValues` para solicitar o item excluído na resposta.

Example

```
var request = new DeleteItemRequest
{
    TableName = tableName,
    Key = new Dictionary<string,AttributeValue>() { { "Id", new AttributeValue { N =
    "201" } } },

    // Optional parameters.
    ReturnValues = "ALL_OLD",
    ExpressionAttributeNames = new Dictionary<string, string>()
    {
        {"#IP", "InPublication"}
    },
    ExpressionAttributeValues = new Dictionary<string, AttributeValue>()
    {
        {"::inpub",new AttributeValue {BOOL = false}}
    },
    ConditionExpression = "#IP = :inpub"
};

var response = client.DeleteItem(request);
```

Para obter mais informações, consulte[DeleteItem](#).

Gravação em Batch: Como inserir e excluir vários itens

Gravação em lote se refere a inserir e excluir vários itens em um lote. O método `BatchWriteItem` permite que você insira e exclua vários itens de uma ou mais tabelas em uma única chamada. Veja a seguir as etapas para recuperar vários itens usando a API de SDK do .NET de baixo nível.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Descreva todas as operações de inserção e exclusão, criando uma instância da classe `BatchWriteItemRequest`.
3. Execute a `BatchWriteItem` fornecendo o método `BatchWriteItemRequest` que você criou na etapa anterior.
4. Procresse a resposta. Você deve verificar se há itens de solicitação não processados retornados na resposta. Isso poderá acontecer se você atingir a cota de taxa de transferência provisionada ou por algum outro erro temporário. Além disso, o DynamoDB limita o tamanho da solicitação e o número de operações que você pode especificar em uma solicitação. Se você exceder esses limites, o DynamoDB rejeitará a solicitação. Para obter mais informações, consulte[BatchWriteItem](#).

O exemplo de código C# a seguir demonstra as etapas anteriores. O exemplo cria uma `BatchWriteItemRequest` para realizar as seguintes operações de gravação:

- Inserir um item na tabela Forum.
- Inserir e excluir um item da tabela Thread.

O código é executado no BatchWriteItem para realizar uma operação em lote.

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();

string table1Name = "Forum";
string table2Name = "Thread";

var request = new BatchWriteItemRequest
{
    RequestItems = new Dictionary<string, List<WriteRequest>>
    {
        {
            table1Name, new List<WriteRequest>
            {
                new WriteRequest
                {
                    PutRequest = new PutRequest
                    {
                        Item = new Dictionary<string,AttributeValue>
                        {
                            { "Name", new AttributeValue { S = "Amazon S3 forum" } },
                            { "Threads", new AttributeValue { N = "0" } }
                        }
                    }
                }
            }
        ,
        {
            table2Name, new List<WriteRequest>
            {
                new WriteRequest
                {
                    PutRequest = new PutRequest
                    {
                        Item = new Dictionary<string,AttributeValue>
                        {
                            { "ForumName", new AttributeValue { S = "Amazon S3 forum" } },
                            { "Subject", new AttributeValue { S = "My sample question" } },
                            { "Message", new AttributeValue { S = "Message Text." } },
                            { "KeywordTags", new AttributeValue { SS = new List<string> { "Amazon S3", "Bucket" } } }
                        }
                    }
                },
                new WriteRequest
                {
                    DeleteRequest = new DeleteRequest
                    {
                        Key = new Dictionary<string,AttributeValue>()
                        {
                            { "ForumName", new AttributeValue { S = "Some forum name" } },
                            { "Subject", new AttributeValue { S = "Some subject" } }
                        }
                    }
                }
            }
        }
    };
response = client.BatchWriteItem(request);
```

Para obter um exemplo de trabalho, consulte [Exemplo: Operações em Batch usando o AWS SDK for .NET API de baixo nível \(p. 481\)](#).

Obter Batch: Obter vários itens

O método `BatchGetItem` permite que você recupere vários itens de uma ou mais tabelas.

Note

Para recuperar um único item, você pode usar o método `GetItem`.

Veja a seguir as etapas para recuperar vários itens usando a API do AWS SDK for .NET de baixo nível.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Forneça os parâmetros necessários, criando uma instância da classe `BatchGetItemRequest`.

Para recuperar vários itens, o nome da tabela e uma lista de valores de chave primária são necessários.

3. Execute a `BatchGetItem` fornecendo o método `BatchGetItemRequest` que você criou na etapa anterior.
4. Processe a resposta. Você deve verificar se houve chaves não processadas, o que pode acontecer caso você tenha atingido a cota de taxa de transferência provisionada ou caso tenha ocorrido algum outro erro temporário.

O exemplo de código C# a seguir demonstra as etapas anteriores. O exemplo recupera itens de duas tabelas, `Forum` e `Thread`. A solicitação especifica dois itens na tabela `Forum` e três itens em `Thread`. A resposta inclui itens de ambas as tabelas. O código mostra como você pode processar a resposta.

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();

string table1Name = "Forum";
string table2Name = "Thread";

var request = new BatchGetItemRequest
{
    RequestItems = new Dictionary<string, KeysAndAttributes>()
    {
        { table1Name,
            new KeysAndAttributes
            {
                Keys = new List<Dictionary<string, AttributeValue>>()
                {
                    new Dictionary<string, AttributeValue>()
                    {
                        { "Name", new AttributeValue { S = "DynamoDB" } }
                    },
                    new Dictionary<string, AttributeValue>()
                    {
                        { "Name", new AttributeValue { S = "Amazon S3" } }
                    }
                }
            }
        },
        { table2Name,
            new KeysAndAttributes
            {
                Keys = new List<Dictionary<string, AttributeValue>>()
                {

```

```
new Dictionary<string, AttributeValue>()
{
    { "ForumName", new AttributeValue { S = "DynamoDB" } },
    { "Subject", new AttributeValue { S = "DynamoDB Thread 1" } }
},
new Dictionary<string, AttributeValue>()
{
    { "ForumName", new AttributeValue { S = "DynamoDB" } },
    { "Subject", new AttributeValue { S = "DynamoDB Thread 2" } }
},
new Dictionary<string, AttributeValue>()
{
    { "ForumName", new AttributeValue { S = "Amazon S3" } },
    { "Subject", new AttributeValue { S = "Amazon S3 Thread 1" } }
}
}
}
};

var response = client.BatchGetItem(request);

// Check the response.
var result = response.BatchGetItemResult;
var responses = result.Responses; // The attribute list in the response.

var table1Results = responses[table1Name];
Console.WriteLine("Items in table {0}" + table1Name);
foreach (var item1 in table1Results.Items)
{
    PrintItem(item1);
}

var table2Results = responses[table2Name];
Console.WriteLine("Items in table {1}" + table2Name);
foreach (var item2 in table2Results.Items)
{
    PrintItem(item2);
}
// Any unprocessed keys? could happen if you exceed ProvisionedThroughput or some other
error.
Dictionary<string, KeysAndAttributes> unprocessedKeys = result.UnprocessedKeys;
foreach (KeyValuePair<string, KeysAndAttributes> pair in unprocessedKeys)
{
    Console.WriteLine(pair.Key, pair.Value);
}
```

Especificação de parâmetros opcionais

Você também pode fornecer parâmetros opcionais usando o objeto `BatchGetItemRequest`, conforme mostrado no seguinte exemplo de código C#. O exemplo recupera dois itens da tabela `Forum`. Ele especifica o parâmetro opcional a seguir:

- o parâmetro `ProjectionExpression` para especificar os atributos a serem recuperados.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();

string table1Name = "Forum";
```

```
var request = new BatchGetItemRequest
{
    RequestItems = new Dictionary<string, KeysAndAttributes>()
    {
        { tableName,
            new KeysAndAttributes
            {
                Keys = new List<Dictionary<string, AttributeValue>>()
                {
                    new Dictionary<string, AttributeValue>()
                    {
                        { "Name", new AttributeValue { S = "DynamoDB" } }
                    },
                    new Dictionary<string, AttributeValue>()
                    {
                        { "Name", new AttributeValue { S = "Amazon S3" } }
                    }
                },
                // Optional - name of an attribute to retrieve.
                ProjectionExpression = "Title"
            }
        };
    };
};

var response = client.BatchGetItem(request);
```

Para obter mais informações, consulte[BatchGetItem](#).

Exemplo: Operações CRUD usando oAWS SDK for .NET API de baixo nível

O exemplo de código C# a seguir ilustra operações CRUD em um item do Amazon DynamoDB. O exemplo adiciona um item à tabela `ProductCatalog`, recupera-o, executa várias atualizações e, por fim, exclui o item. Se você tiver seguido as etapas em [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#), já terá a tabela `ProductCatalog` criada. Você também pode criar essas tabelas de amostra de forma programática. Para mais informações, consulte [Como criar tabelas de exemplo e carregar dados usando oAWS SDK for .NET \(p. 1086\)](#).

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

Example

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
*/
using System;
using System.Collections.Generic;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;
```

```
using Amazon.Runtime;
using Amazon.SecurityToken;

namespace com.amazonaws.codesamples
{
    class LowLevelItemCRUDExample
    {
        private static string tableName = "ProductCatalog";
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();

        static void Main(string[] args)
        {
            try
            {
                CreateItem();
                RetrieveItem();

                // Perform various updates.
                UpdateMultipleAttributes();
                UpdateExistingAttributeConditionally();

                // Delete item.
                DeleteItem();
                Console.WriteLine("To continue, press Enter");
                Console.ReadLine();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                Console.WriteLine("To continue, press Enter");
                Console.ReadLine();
            }
        }

        private static void CreateItem()
        {
            var request = new PutItemRequest
            {
                TableName = tableName,
                Item = new Dictionary<string, AttributeValue>()
            {
                { "Id", new AttributeValue {
                    N = "1000"
                }},
                { "Title", new AttributeValue {
                    S = "Book 201 Title"
                }},
                { "ISBN", new AttributeValue {
                    S = "11-11-11-11"
                }},
                { "Authors", new AttributeValue {
                    SS = new List<string>{"Author1", "Author2" }
                }},
                { "Price", new AttributeValue {
                    N = "20.00"
                }},
                { "Dimensions", new AttributeValue {
                    S = "8.5x11.0x.75"
                }},
                { "InPublication", new AttributeValue {
                    BOOL = false
                } }
            }
        };
        client.PutItem(request);
    }
}
```

```
private static void RetrieveItem()
{
    var request = new GetItemRequest
    {
        TableName = tableName,
        Key = new Dictionary<string, AttributeValue>()
    {
        { "Id", new AttributeValue {
            N = "1000"
        } }
    },
    ProjectionExpression = "Id, ISBN, Title, Authors",
    ConsistentRead = true
};
var response = client.GetItem(request);

// Check the response.
var attributeList = response.Item; // attribute list in the response.
Console.WriteLine("\nPrinting item after retrieving it .....");
PrintItem(attributeList);
}

private static void UpdateMultipleAttributes()
{
    var request = new UpdateItemRequest
    {
        Key = new Dictionary<string, AttributeValue>()
    {
        { "Id", new AttributeValue {
            N = "1000"
        } }
    },
    // Perform the following updates:
    // 1) Add two new authors to the list
    // 1) Set a new attribute
    // 2) Remove the ISBN attribute
    ExpressionAttributeNames = new Dictionary<string, string>()
    {
        {"#A","Authors"},
        {"#NA","NewAttribute"},
        {"#I","ISBN"}
    },
    ExpressionAttributeValues = new Dictionary<string, AttributeValue>()
    {
        {"":auth",new AttributeValue {
            SS = {"Author YY", "Author ZZ"}
        }},
        {"":new",new AttributeValue {
            S = "New Value"
        }}
    },
    UpdateExpression = "ADD #A :auth SET #NA = :new REMOVE #I",
    TableName = tableName,
    ReturnValues = "ALL_NEW" // Give me all attributes of the updated item.
};
var response = client.UpdateItem(request);

// Check the response.
var attributeList = response.Attributes; // attribute list in the response.
                                            // print attributeList.
Console.WriteLine("\nPrinting item after multiple attribute
update .....");
PrintItem(attributeList);
```

```
}

private static void UpdateExistingAttributeConditionally()
{
    var request = new UpdateItemRequest
    {
        Key = new Dictionary<string, AttributeValue>()
    {
        { "Id", new AttributeValue {
            N = "1000"
        } }
    },
    ExpressionAttributeNames = new Dictionary<string, string>()
    {
        {"#P", "Price"}
    },
    ExpressionAttributeValues = new Dictionary<string, AttributeValue>()
    {
        {"<:newprice",new AttributeValue {
            N = "22.00"
        }},
        {"<:currprice",new AttributeValue {
            N = "20.00"
        }}
    },
    // This updates price only if current price is 20.00.
    UpdateExpression = "SET #P = :newprice",
    ConditionExpression = "#P = :currprice",

    TableName = tableName,
    ReturnValues = "ALL_NEW" // Give me all attributes of the updated item.
};
var response = client.UpdateItem(request);

// Check the response.
var attributeList = response.Attributes; // attribute list in the response.
Console.WriteLine("\nPrinting item after updating price value
conditionally .....");
PrintItem(attributeList);
}

private static void DeleteItem()
{
    var request = new DeleteItemRequest
    {
        TableName = tableName,
        Key = new Dictionary<string, AttributeValue>()
    {
        { "Id", new AttributeValue {
            N = "1000"
        } }
    },

    // Return the entire item as it appeared before the update.
    ReturnValues = "ALL_OLD",
    ExpressionAttributeNames = new Dictionary<string, string>()
    {
        {"#IP", "InPublication"}
    },
    ExpressionAttributeValues = new Dictionary<string, AttributeValue>()
    {
        {"<:inpub",new AttributeValue {
            BOOL = false
        }}
    },
    ConditionExpression = "#IP = :inpub"
```

```
};

var response = client.DeleteItem(request);

// Check the response.
var attributeList = response.Attributes; // Attribute list in the response.
                                         // Print item.
Console.WriteLine("\nPrinting item that was just deleted .....");
PrintItem(attributeList);
}

private static void PrintItem(Dictionary<string, AttributeValue> attributeList)
{
    foreach (KeyValuePair<string, AttributeValue> kvp in attributeList)
    {
        string attributeName = kvp.Key;
        AttributeValue value = kvp.Value;

        Console.WriteLine(
            attributeName + " " +
            (value.S == null ? "" : "S=[\"" + value.S + "\"]") +
            (value.N == null ? "" : "N=[\"" + value.N + "\"]") +
            (value.SS == null ? "" : "SS=[\"" + string.Join(",", value.SS.ToArray()) +
            "\"]") +
            (value.NS == null ? "" : "NS=[\"" + string.Join(",", value.NS.ToArray()) +
            "\"]"));
    }
    Console.WriteLine("*****");
}
}
```

Exemplo: Operações em Batch usando oAWS SDK for .NETAPI de baixo nível

Tópicos

- [Exemplo: Operação de gravação em Batch usando oAWS SDK for .NETAPI de baixo nível \(p. 481\)](#)
- [Exemplo: Operação de aquisição em Batch usando oAWS SDK for .NETAPI de baixo nível \(p. 484\)](#)

Esta seção fornece exemplos de operações em lote, Gravação em lote e Obter lote, que o Amazon DynamoDB oferece suporte.

Exemplo: Operação de gravação em Batch usando oAWS SDK for .NETAPI de baixo nível

O exemplo de código C# a seguir usa o método `BatchWriteItem` para executar as operações de inserção e exclusão a seguir:

- Inserir um item na tabela `Forum`.
- Inserir e excluir um item da tabela `Thread`.

Você pode especificar quantas solicitações de inserção e exclusão desejar em uma ou mais tabelas ao criar sua solicitação de gravação em lote. No entanto, o `DynamoDBBatchWriteItem` limita o tamanho de uma solicitação de gravação em lote e o número de operações de inserção e exclusão em uma única operação de gravação em lote. Para obter mais informações, consulte [BatchWriteItem](#). Se a sua solicitação ultrapassar esses limites, ela será rejeitada. Se a sua tabela não tiver throughput provisionado suficiente para servir a essa solicitação, os itens de solicitação não processados serão retornados na resposta.

O exemplo a seguir verifica a resposta para conferir se existem itens de solicitação não processados. Se houver, ele retorna e reenvia a solicitação BatchWriteItem com itens não processados na solicitação. Se você tiver seguido as etapas em [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#), já terá as tabelas Forum e Thread criadas. Você também pode criar essas tabelas de exemplo e carregar dados de exemplo de forma programática. Para mais informações, consulte [Como criar tabelas de exemplo e carregar dados usando o AWS SDK for .NET \(p. 1086\)](#).

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
using System;  
using System.Collections.Generic;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.Model;  
using Amazon.Runtime;  
  
namespace com.amazonaws.codesamples  
{  
    class LowLevelBatchWrite  
    {  
        private static string table1Name = "Forum";  
        private static string table2Name = "Thread";  
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();  
  
        static void Main(string[] args)  
        {  
            try  
            {  
                TestBatchWrite();  
            }  
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }  
            catch (Exception e) { Console.WriteLine(e.Message); }  
  
            Console.WriteLine("To continue, press Enter");  
            Console.ReadLine();  
        }  
  
        private static void TestBatchWrite()  
        {  
            var request = new BatchWriteItemRequest  
            {  
                ReturnConsumedCapacity = "TOTAL",  
                RequestItems = new Dictionary<string, List<WriteRequest>>  
            };  
            {  
                table1Name, new List<WriteRequest>  
                {  
                    new WriteRequest
```

```
        {
            PutRequest = new PutRequest
            {
                Item = new Dictionary<string, AttributeValue>
                {
                    { "Name", new AttributeValue {
                        S = "S3 forum"
                    } },
                    { "Threads", new AttributeValue {
                        N = "0"
                    } }
                }
            }
        },
        {
            table2Name, new List<WriteRequest>
            {
                new WriteRequest
                {
                    PutRequest = new PutRequest
                    {
                        Item = new Dictionary<string, AttributeValue>
                        {
                            { "ForumName", new AttributeValue {
                                S = "S3 forum"
                            } },
                            { "Subject", new AttributeValue {
                                S = "My sample question"
                            } },
                            { "Message", new AttributeValue {
                                S = "Message Text."
                            } },
                            { "KeywordTags", new AttributeValue {
                                SS = new List<string> { "S3", "Bucket" }
                            } }
                        }
                    }
                },
                new WriteRequest
                {
                    // For the operation to delete an item, if you provide a
                    // primary key value
                    // that does not exist in the table, there is no error, it is
                    // just a no-op.
                    DeleteRequest = new DeleteRequest
                    {
                        Key = new Dictionary<string, AttributeValue>()
                        {
                            { "ForumName", new AttributeValue {
                                S = "Some partition key value"
                            } },
                            { "Subject", new AttributeValue {
                                S = "Some sort key value"
                            } }
                        }
                    }
                }
            }
        };
    }

    CallBatchWriteTillCompletion(request);
}
```

```
private static void CallBatchWriteTillCompletion(BatchWriteItemRequest request)
{
    BatchWriteItemResponse response;

    int callCount = 0;
    do
    {
        Console.WriteLine("Making request");
        response = client.BatchWriteItem(request);
        callCount++;

        // Check the response.

        var tableConsumedCapacities = response.ConsumedCapacity;
        var unprocessed = response.UnprocessedItems;

        Console.WriteLine("Per-table consumed capacity");
        foreach (var tableConsumedCapacity in tableConsumedCapacities)
        {
            Console.WriteLine("{0} - {1}", tableConsumedCapacity.TableName,
tableConsumedCapacity.CapacityUnits);
        }

        Console.WriteLine("Unprocessed");
        foreach (var unp in unprocessed)
        {
            Console.WriteLine("{0} - {1}", unp.Key, unp.Value.Count);
        }
        Console.WriteLine();

        // For the next iteration, the request will have unprocessed items.
        request.RequestItems = unprocessed;
    } while (response.UnprocessedItems.Count > 0);

    Console.WriteLine("Total # of batch write API calls made: {0}", callCount);
}
}
```

Exemplo: Operação de aquisição em Batch usando oAWS SDK for .NETAPI de baixo nível

O exemplo de código C# a seguir usa a propriedade `BatchGetItem` para recuperar vários itens do `Forum` e `Threads` no Amazon DynamoDB. A `BatchGetItemRequest` especifica os nomes de tabelas e uma lista de chaves primárias para cada tabela. O exemplo processa a resposta, imprimindo os itens recuperados.

Se você seguiu as etapas em [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#), já tem essas tabelas criadas com dados de exemplo. Você também pode criar essas tabelas de exemplo e carregar dados de exemplo de forma programática. Para mais informações, consulte [Como criar tabelas de exemplo e carregar dados usando oAWS SDK for .NET \(p. 1086\)](#).

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

Example

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
```

```
* This file is licensed under the Apache License, Version 2.0 (the "License").  
* You may not use this file except in compliance with the License. A copy of  
* the License is located at  
*  
* http://aws.amazon.com/apache2.0/  
*  
* This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
* CONDITIONS OF ANY KIND, either express or implied. See the License for the  
* specific language governing permissions and limitations under the License.  
*/  
using System;  
using System.Collections.Generic;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.Model;  
using Amazon.Runtime;  
  
namespace com.amazonaws.codesamples  
{  
    class LowLevelBatchGet  
    {  
        private static string table1Name = "Forum";  
        private static string table2Name = "Thread";  
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();  
  
        static void Main(string[] args)  
        {  
            try  
            {  
                RetrieveMultipleItemsBatchGet();  
  
                Console.WriteLine("To continue, press Enter");  
                Console.ReadLine();  
            }  
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }  
            catch (Exception e) { Console.WriteLine(e.Message); }  
        }  
  
        private static void RetrieveMultipleItemsBatchGet()  
        {  
            var request = new BatchGetItemRequest  
            {  
                RequestItems = new Dictionary<string, KeysAndAttributes>()  
            };  
            { table1Name,  
              new KeysAndAttributes  
              {  
                  Keys = new List<Dictionary<string, AttributeValue>>()  
                  {  
                      new Dictionary<string, AttributeValue>()  
                      {  
                          { "Name", new AttributeValue {  
                                S = "Amazon DynamoDB"  
                            } }  
                      },  
                      new Dictionary<string, AttributeValue>()  
                      {  
                          { "Name", new AttributeValue {  
                                S = "Amazon S3"  
                            } }  
                      }  
                  }  
              },  
              table2Name,  
              new KeysAndAttributes  
              {  
                  Keys = new List<Dictionary<string, AttributeValue>>()  
                  {  
                      new Dictionary<string, AttributeValue>()  
                      {  
                          { "Name", new AttributeValue {  
                                S = "Amazon S3"  
                            } }  
                      }  
                  }  
              }  
            };  
        }  
    }  
}
```

```
        Keys = new List<Dictionary<string, AttributeValue>>()
        {
            new Dictionary<string, AttributeValue>()
            {
                { "ForumName", new AttributeValue {
                    S = "Amazon DynamoDB"
                } },
                { "Subject", new AttributeValue {
                    S = "DynamoDB Thread 1"
                } }
            },
            new Dictionary<string, AttributeValue>()
            {
                { "ForumName", new AttributeValue {
                    S = "Amazon DynamoDB"
                } },
                { "Subject", new AttributeValue {
                    S = "DynamoDB Thread 2"
                } }
            },
            new Dictionary<string, AttributeValue>()
            {
                { "ForumName", new AttributeValue {
                    S = "Amazon S3"
                } },
                { "Subject", new AttributeValue {
                    S = "S3 Thread 1"
                } }
            }
        }
    }
};

BatchGetItemResponse response;
do
{
    Console.WriteLine("Making request");
    response = client.BatchGetItem(request);

    // Check the response.
    var responses = response.Responses; // Attribute list in the response.

    foreach (var tableResponse in responses)
    {
        var tableResults = tableResponse.Value;
        Console.WriteLine("Items retrieved from table {0}", tableResponse.Key);
        foreach (var item1 in tableResults)
        {
            PrintItem(item1);
        }
    }

    // Any unprocessed keys? Could happen if you exceed ProvisionedThroughput
    or some other error.
    Dictionary<string, KeysAndAttributes> unprocessedKeys =
    response.UnprocessedKeys;
    foreach (var unprocessedTableKeys in unprocessedKeys)
    {
        // Print table name.
        Console.WriteLine(unprocessedTableKeys.Key);
        // Print unprocessed primary keys.
        foreach (var key in unprocessedTableKeys.Value.Keys)
        {
            PrintItem(key);
        }
    }
}
```

```
        }

        request.RequestItems = unprocessedKeys;
    } while (response.UnprocessedKeys.Count > 0);
}

private static void PrintItem(Dictionary<string, AttributeValue> attributeList)
{
    foreach (KeyValuePair<string, AttributeValue> kvp in attributeList)
    {
        string attributeName = kvp.Key;
        AttributeValue value = kvp.Value;

        Console.WriteLine(
            attributeName + " " +
            (value.S == null ? "" : "S=[" + value.S + "]") +
            (value.N == null ? "" : "N=[" + value.N + "]") +
            (value.SS == null ? "" : "SS=[" + string.Join(",", value.SS.ToArray()) +
            "])" +
            (value.NS == null ? "" : "NS=[" + string.Join(",", value.NS.ToArray()) +
            "])");
    }
    Console.WriteLine("*****");
}
}
```

Exemplo: Tratamento de atributos do tipo binário usando oAWS SDK for .NET API de baixo nível

O exemplo de código C# a seguir ilustra a manipulação de atributos do tipo Binário. O exemplo adiciona um item à tabela Reply. O item inclui um atributo do tipo binário (ExtendedMessage) que armazena dados compactados. Em seguida, o exemplo recupera um item e imprime todos os valores do atributo. Para ilustração, o exemplo usa a classe GZipStream para compactar um stream de exemplo e designá-lo ao atributo ExtendedMessage. Em seguida, ele descompacta esse stream ao imprimir o valor do atributo.

Se você tiver seguido as etapas em [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#), já terá a tabela Reply criada. Você também pode criar essas tabelas de amostra de forma programática. Para mais informações, consulte [Como criar tabelas de exemplo e carregar dados usando oAWS SDK for .NET \(p. 1086\)](#).

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

Example

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
*/
```

```
/*
using System;
using System.Collections.Generic;
using System.IO;
using System.IO.Compression;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;
using Amazon.Runtime;

namespace com.amazonaws.codesamples
{
    class LowLevelItemBinaryExample
    {
        private static string tableName = "Reply";
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();

        static void Main(string[] args)
        {
            // Reply table primary key.
            string replyIdPartitionKey = "Amazon DynamoDB#DynamoDB Thread 1";
            string replyDateTimeSortKey = Convert.ToString(DateTime.UtcNow);

            try
            {
                CreateItem(replyIdPartitionKey, replyDateTimeSortKey);
                RetrieveItem(replyIdPartitionKey, replyDateTimeSortKey);
                // Delete item.
                DeleteItem(replyIdPartitionKey, replyDateTimeSortKey);
                Console.WriteLine("To continue, press Enter");
                Console.ReadLine();
            }
            catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message); }
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
            catch (Exception e) { Console.WriteLine(e.Message); }
        }

        private static void CreateItem(string partitionKey, string sortKey)
        {
            MemoryStream compressedMessage = ToGzipMemoryStream("Some long extended message
to compress.");
            var request = new PutItemRequest
            {
                TableName = tableName,
                Item = new Dictionary<string, AttributeValue>()
            };
            {
                {"Id", new AttributeValue {
                    S = partitionKey
                }},
                {"ReplyDateTime", new AttributeValue {
                    S = sortKey
                }},
                {"Subject", new AttributeValue {
                    S = "Binary type "
                }},
                {"Message", new AttributeValue {
                    S = "Some message about the binary type"
                }},
                {"ExtendedMessage", new AttributeValue {
                    B = compressedMessage
                }}
            };
            client.PutItem(request);
        }

        private static void RetrieveItem(string partitionKey, string sortKey)
```

```
{  
    var request = new GetItemRequest  
{  
        TableName = tableName,  
        Key = new Dictionary<string, AttributeValue>()  
{  
            { "Id", new AttributeValue {  
                S = partitionKey  
            } },  
            { "ReplyDateTime", new AttributeValue {  
                S = sortKey  
            } }  
        },  
        ConsistentRead = true  
    };  
    var response = client.GetItem(request);  
  
    // Check the response.  
    var attributeList = response.Item; // attribute list in the response.  
    Console.WriteLine("\nPrinting item after retrieving it .....");  
  
    PrintItem(attributeList);  
}  
  
private static void DeleteItem(string partitionKey, string sortKey)  
{  
    var request = new DeleteItemRequest  
{  
        TableName = tableName,  
        Key = new Dictionary<string, AttributeValue>()  
{  
            { "Id", new AttributeValue {  
                S = partitionKey  
            } },  
            { "ReplyDateTime", new AttributeValue {  
                S = sortKey  
            } }  
        }  
    };  
    var response = client.DeleteItem(request);  
}  
  
private static void PrintItem(Dictionary<string, AttributeValue> attributeList)  
{  
    foreach (KeyValuePair<string, AttributeValue> kvp in attributeList)  
    {  
        string attributeName = kvp.Key;  
        AttributeValue value = kvp.Value;  
  
        Console.WriteLine(  
            attributeName + " " +  
            (value.S == null ? "" : "S=[ " + value.S + " ]") +  
            (value.N == null ? "" : "N=[ " + value.N + " ]") +  
            (value.SS == null ? "" : "SS=[ " + string.Join(",", value.SS.ToArray())  
+ " ]") +  
            (value.NS == null ? "" : "NS=[ " + string.Join(",", value.NS.ToArray())  
+ " ]") +  
            (value.B == null ? "" : "B=[ " + FromGzipMemoryStream(value.B) + " ]");  
        )  
        Console.WriteLine("*****");  
    }  
  
    private static MemoryStream ToGzipMemoryStream(string value)  
    {  
        MemoryStream output = new MemoryStream();  
    }
```

```
        using (GZipStream zipStream = new GZipStream(output, CompressionMode.Compress,
true))
        using (StreamWriter writer = new StreamWriter(zipStream))
        {
            writer.Write(value);
        }
        return output;
    }

    private static string FromGzipMemoryStream(MemoryStream stream)
    {
        using (GZipStream zipStream = new GZipStream(stream,
CompressionMode.Decompress))
        using (StreamReader reader = new StreamReader(zipStream))
        {
            return reader.ReadToEnd();
        }
    }
}
```

Como trabalhar com consultas no DynamoDB

O `Query` no Amazon DynamoDB encontrará itens com base nos valores das chaves primárias.

Você deve fornecer o nome do atributo de chave de partição e um único valor para esse atributo. A operação `Query` retorna todos os itens com esse valor de chave de partição. Opcionalmente, você pode fornecer um atributo de chave de classificação e usar um operador de comparação para refinar os resultados da pesquisa.

Tópicos

- [Expressão de condição principal \(p. 490\)](#)
- [Expressões de filtro para consulta \(p. 492\)](#)
- [Limitação do número de itens no conjunto de resultados \(p. 493\)](#)
- [Paginar resultados de consulta de tabela \(p. 493\)](#)
- [Contagem dos itens nos resultados \(p. 495\)](#)
- [Unidades de capacidade consumidas por Query \(p. 495\)](#)
- [Consistência de leitura de query \(p. 496\)](#)
- [Consulta de tabelas e índices: Java \(p. 496\)](#)
- [Consulta de tabelas e índices: .NET \(p. 502\)](#)

Expressão de condição principal

Para especificar os critérios de pesquisa, você deve usar uma expressão de condição principal – uma string que determina os itens a serem lidos da tabela ou índice.

Você deve especificar o nome e o valor da chave de partição como uma condição de igualdade.

Opcionalmente, você pode fornecer uma segunda condição para a chave de classificação (se houver). A condição de chave de classificação deve usar um dos seguintes operadores de comparação:

- `a = b` — verdadeiro se o atributo `a` for igual ao valor `b`
- `a < b` — true se `a` for menor que `b`
- `a <= b` — true se `a` for menor que ou igual a `b`

- $a > b$ — true se a for maior que b
- $a \geq b$ — true se a for maior ou igual a b
- $a \text{ BETWEEN } b \text{ AND } c$ - verdadeiro se a for maior ou igual a b e menor ou igual a c .

A função a seguir também tem suporte:

- `begins_with (a, substr)`- verdadeiro se o valor do atributo a começa com uma determinada substring.

Os seguintes exemplos da AWS Command Line Interface (AWS CLI) demonstram o uso das expressões de condição de chave. Essas expressões usam espaços reservados (como `:name` e `:sub`), em vez de valores reais. Para mais informações, consulte [.](#)

Example

Consulte a tabela `Thread` para encontrar um `ForumName` específico (chave de partição). Todos os itens com esse valor de `ForumName` são lidos pela consulta, porque a chave de classificação (`Subject`) não está incluída na `KeyConditionExpression`.

```
aws dynamodb query \  
  --table-name Thread \  
  --key-condition-expression "ForumName = :name" \  
  --expression-attribute-values '{":name":{"S":"Amazon DynamoDB"}}'
```

Example

Consulte a tabela `Thread` para encontrar um `ForumName` específico (chave de partição), mas, desta vez, retornar apenas os itens com um determinado `Subject` (chave de classificação).

```
aws dynamodb query \  
  --table-name Thread \  
  --key-condition-expression "ForumName = :name and Subject = :sub" \  
  --expression-attribute-values file://values.json
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `values.json`.

```
{  
  ":name":{"S":"Amazon DynamoDB"},  
  ":sub":{"S":"DynamoDB Thread 1"}  
}
```

Example

Consulte a tabela `Reply` para encontrar um `Id` específico (chave de partição), mas retornar apenas os itens cuja `ReplyDateTime` (chave de classificação) começa com determinados caracteres.

```
aws dynamodb query \  
  --table-name Reply \  
  --key-condition-expression "Id = :id and begins_with(ReplyDateTime, :dt)" \  
  --expression-attribute-values file://values.json
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `values.json`.

```
{
```

```
":id": {"S": "Amazon DynamoDB#DynamoDB Thread 1"},  
":dt": {"S": "2015-09"}  
}
```

Você pode usar qualquer nome de atributo em uma expressão de condição principal, desde que o primeiro caractere seja a-z ou A-Z e o segundo caractere (se houver) seja a-z, A-Z ou 0-9. Além disso, o nome do atributo não deve ser uma palavra reservada do DynamoDB. (Para obter uma lista completa dessas palavras reservadas, consulte [Palavras reservadas no DynamoDB \(p. 1125\)](#).) Se um nome de atributo não atender a esses requisitos, defina um nome de atributo de expressão como um espaço reservado. Para mais informações, consulte [Nomes de atributo de expressão no DynamoDB \(p. 418\)](#).

Para itens com um determinado valor de chave de partição, o DynamoDB armazena esses itens juntos, na ordem classificada por valor de chave de classificação. Em um `Query`, o DynamoDB recupera os itens na ordem classificada e, em seguida, processa os itens usando o `KeyConditionExpression` `FilterExpression` que pode estar presente. Somente então os resultados de `Query` são enviados de volta para o cliente.

Uma operação `Query` sempre retorna um conjunto de resultados. Se não forem encontrados itens correspondentes, o conjunto de resultados estará vazio.

Os resultados de `Query` são sempre classificados pelo valor de chave de classificação. Se o tipo de dados da chave de classificação for `Number`, os resultados serão retornados em ordem numérica. Caso contrário, os resultados serão retornados na ordem de bytes UTF-8. Por padrão, a ordem de classificação é crescente. Para reverter a ordem, defina o parâmetro `ScanIndexForward` como `false`.

Um único `Query` operação pode recuperar um máximo de 1 MB de dados. Esse limite se aplica antes de qualquer `FilterExpression` ser aplicada aos resultados. Se `LastEvaluatedKey` estiver presente na resposta e não for nula, você deverá paginar o conjunto de resultados (consulte [Paginar resultados de consulta de tabela \(p. 493\)](#)).

Expressões de filtro para consulta

Se você precisar refinar ainda mais os resultados de `Query`, existe a opção de utilizar uma expressão de filtro. Uma expressão de filtro determina quais itens dos resultados de `Query` devem ser retornados para você. Todos os outros resultados serão descartados.

Uma expressão de filtro é aplicada depois que uma operação `Query` é concluída, mas antes que os resultados sejam retornados. Portanto, uma operação `Query` consome a mesma quantidade de capacidade de leitura, independentemente de uma expressão de filtro estar presente.

A `Query` operação pode recuperar um máximo de 1 MB de dados. Esse limite se aplica antes de a expressão de filtro ser avaliada.

Uma expressão de filtro não pode conter atributos de chave de partição ou de chave de classificação. É necessário especificar esses atributos na expressão de condição principal, não na de filtro.

A sintaxe de uma expressão de filtro é idêntica à de uma expressão de condição de chave. As expressões de filtro podem usar os mesmos comparadores, funções e operadores lógicos que uma expressão de condição de chave, com a adição do operador não-igual (`<>`). Para mais informações, consulte [Expressão de condição principal \(p. 490\)](#).

Example

O exemplo da AWS CLI a seguir consulta a tabela `Thread` para encontrar um `ForumName` (chave de partição) e um `Subject` (chave de classificação) específicos. Dos itens que são encontrados, somente os threads de discussão mais populares são retornados — em outras palavras, somente os threads com mais de um determinado número de `Views`.

```
aws dynamodb query \  
    --table-name Thread \  
    --key-condition-expression "ForumName = :fn and Subject = :sub" \  
    --filter-expression "#v >= :num" \  
    --expression-attribute-names '{"#v": "Views"}' \  
    --expression-attribute-values file://values.json
```

Os argumentos de `--expression-attribute-values` são armazenados no arquivo `values.json`.

```
{  
    ":fn": {"S": "Amazon DynamoDB"},  
    ":sub": {"S": "DynamoDB Thread 1"},  
    ":num": {"N": "3"}  
}
```

Observações que `Views` é uma palavra reservada no DynamoDB (consulte [Palavras reservadas no DynamoDB \(p. 1125\)](#)), então este exemplo usa `#v` como um espaço reservado. Para mais informações, consulte [Nomes de atributo de expressão no DynamoDB \(p. 418\)](#).

Note

Uma expressão de filtro remove itens do conjunto de resultados de `Query`. Se possível, evite usar `Query` onde você espera recuperar um grande número de itens, mas também precisa descartar a maioria desses itens.

Limitação do número de itens no conjunto de resultados

A operação `Query` permite que você limite o número de itens que ela lê. Para fazer isso, defina o parâmetro `Limit` com o número máximo de itens que você deseja.

Por exemplo, suponha que você execute a operação `Query` em uma tabela, com um valor de `Limit` igual a 6 e sem uma expressão de filtro. O resultado de `Query` contém os primeiros seis itens da tabela que correspondem à expressão de condição da chave da solicitação.

Agora suponha que você adicione uma expressão de filtro à operação `Query`. Nesse caso, o DynamoDB lê até seis itens e retorna somente aqueles que correspondem à expressão do filtro. A final `Query` contém seis itens ou menos, mesmo que mais itens tenham correspondido à expressão do filtro se o DynamoDB tivesse continuado lendo mais itens.

Paginar resultados de consulta de tabela

DynamoDB paginados. Os resultados de `Query` operações. Com a paginação, o `Query` resultados de são divididos em “páginas” de dados que têm 1 MB de tamanho (ou menos). Um aplicativo pode processar a primeira página de resultados e, em seguida, a segunda página, e assim por diante.

Um único `Query` retorna apenas um conjunto de resultados que estão dentro do limite de tamanho de 1 MB. Para determinar se há mais resultados e para recuperá-los em uma página por vez, os aplicativos devem fazer o seguinte:

1. Examine o resultado de `Query` de baixo nível:
 - Se o resultado contiver um `LastEvaluatedKey` elemento e não é nulo, prossiga para a etapa 2.
 - Se não houver um `LastEvaluatedKey` no resultado, não haverá mais itens a serem recuperados.

2. Construa a nova solicitação de `Query` com os mesmos parâmetros da anterior. No entanto, desta vez, use o valor de `LastEvaluatedKey` da etapa 1 como o parâmetro `ExclusiveStartKey` na nova solicitação de `Query`.
3. Execute a nova solicitação de `Query`.
4. Vá para a etapa 1.

Em outras palavras, o valor `LastEvaluatedKey` de uma resposta de `Query` deve ser usado como `ExclusiveStartKey` da próxima solicitação de `Query`. Se não houver um elemento `LastEvaluatedKey` em uma resposta de `Query`, então, você recuperou a página de resultados final. Se `LastEvaluatedKey` não está vazio, isso não necessariamente significa que há mais dados no conjunto de resultados. A única maneira de saber quando você atingiu o final do conjunto de resultados é quando `LastEvaluatedKey` estiver vazio.

Você pode usar a AWS CLI para visualizar esse comportamento. OAWS CLlenvia de baixo nível `doQuerysolicitações` ao DynamoDB repetidamente, até`LastEvaluatedKeyNão` está mais presente nos resultados. Considere o seguinte exemplo da AWS CLI que recupera títulos de filmes de um determinado ano.

```
aws dynamodb query --table-name Movies \
--projection-expression "title" \
--key-condition-expression "#y = :yyyy" \
--expression-attribute-names '{"#y": "year"}' \
--expression-attribute-values '{":yyyy":{"N": "1993"}}' \
--page-size 5 \
--debug
```

Normalmente, a AWS CLI lida com a paginação automaticamente. No entanto, neste exemplo, o parâmetro `--page-size` da AWS CLI limita o número de itens por página. O parâmetro de `--debug` imprime as informações de baixo nível sobre solicitações e respostas.

Se você executar o exemplo, a primeira resposta do DynamoDB será similar à seguinte.

```
2017-07-07 11:13:15,603 - MainThread - botocore.parsers - DEBUG - Response body:
b'{"Count":5,"Items":[{"title":{"S":"A Bronx Tale"}}, {"title":{"S":"A Perfect World"}}, {"title":{"S":"Addams Family Values"}}, {"title":{"S":"Alive"}}, {"title":{"S":"Benny & Joon"}}, "LastEvaluatedKey":{"year":{"N":"1993"}, "title":{"S":"Benny & Joon"}}, "ScannedCount":5}'
```

O `LastEvaluatedKey` na resposta indica que nem todos os itens foram recuperados. OAWS CLIEm seguida, emite outro`QuerySolicitação` ao DynamoDB. Essa solicitação e o padrão de resposta continuam, até a resposta final.

```
2017-07-07 11:13:16,291 - MainThread - botocore.parsers - DEBUG - Response body:
b'{"Count":1,"Items":[{"title":{"S":"What\ 's Eating Gilbert Grape"}]}, "ScannedCount":1}'
```

A ausência de `LastEvaluatedKey` indica que não há mais itens a serem recuperados.

Note

OAWSOs SDKs lidam com as respostas de API do DynamoDB de baixo nível (incluindo a presença ou a ausência de`LastEvaluatedKey`) e fornecer várias abstrações para `paginaçãoQueryResultados`. Por exemplo, o SDK para interface de documento Java fornece`java.util.IteratorPara` que você possa abordar um resultado de cada vez. Para obter exemplos de código em várias linguagens de programação, consulte o[Guia de conceitos básicos do Amazon DynamoDB](#) e a[AWS Documentação do SDK](#) para o seu idioma.

Para obter mais informações sobre como consultar com o DynamoDB, consulte[Como trabalhar com consultas no DynamoDB \(p. 490\)](#).

Contagem dos itens nos resultados

Além dos itens que correspondem aos seus critérios, a resposta de `Query` contém os elementos a seguir:

- `ScannedCount`— O número de itens que corresponderam à expressão de condição de chaveAntesUma expressão de filtro (se houver) foi aplicada.
- `Count`— O número de itens que permanecemDepoisUma expressão de filtro (se houver) foi aplicada.

Note

Se você não usar uma expressão de filtro, `ScannedCount` e `Count` terão o mesmo valor.

Se o tamanho do queryconjunto de resultados é maior que 1 MB, `ScannedCount` e `Count` representam apenas uma contagem parcial do total de itens. Você precisa executar várias operações `Query` para recuperar todos os resultados (consulte [Paginar resultados de consulta de tabela \(p. 493\)](#)).

Cada resposta de `Query` contém os valores de `ScannedCount` e de `Count` dos itens que foram processados pela solicitação de `Query` específica. Para obter os totais gerais de todas as solicitações de `Query`, você pode manter um total em execução de `ScannedCount` e `Count`.

Unidades de capacidade consumidas por Query

Você pode `query` Qualquer tabela ou índice secundário, desde que haja uma chave primária composta (chave de partição e chave de classificação). As operações consomem unidades de capacidade de leitura da seguinte forma.

Se você executar uma operação <code>Query</code> em...	O DynamoDB consumirá unidades de capacidade de leitura de...
Tabela	A capacidade de leitura provisionada da tabela.
Índice secundário global	A capacidade de leitura provisionada do índice.
Índice secundário local	A capacidade de leitura provisionada da tabela-base.

Por padrão, uma operação `Query` não retorna quaisquer dados sobre quanta capacidade de leitura ela consome. Entretanto, você pode especificar o parâmetro `ReturnConsumedCapacity` em uma solicitação de `Query` para obter essas informações. A seguir estão as configurações válidas de `ReturnConsumedCapacity`:

- `NONE`— nenhum dado de capacidade consumida é retornado. (Esse é o padrão.)
- `TOTAL`— a resposta inclui o número agregado de unidades de capacidade de leitura consumidas.
- `INDEXES`— a resposta mostra o número agregado de unidades de capacidade de leitura consumidas, junto com a capacidade consumida para cada tabela e índice acessados.

O DynamoDB calcula o número de unidades de capacidade de leitura consumidas com base no tamanho do item, não na quantidade de dados que é retornada para um aplicativo. Por esse motivo, o número de unidades de capacidade consumidas será o mesmo, independentemente de você solicitar todos os atributos (o comportamento padrão) ou apenas alguns deles (usando uma expressão de projeção). O número também é o mesmo quer você use uma expressão de filtro ou não.

Consistência de leitura de query

Uma operação `Query` executa leituras eventualmente consistentes, por padrão. Isso significa que os resultados de `Query` talvez não reflitam as alterações causadas pelas operações `PutItem` ou `UpdateItem` concluídas recentemente. Para mais informações, consulte [Consistência de leituras \(p. 17\)](#).

Se você precisar de leituras fortemente consistentes, defina o parâmetro `ConsistentRead` como `true` na solicitação de `Query`.

Consulta de tabelas e índices: Java

O `query` A operação permite consultar uma tabela ou um índice secundário no Amazon DynamoDB. Você deve fornecer um valor de chave de partição e uma condição de igualdade. Se a tabela ou o índice tiver uma chave de classificação, você poderá refinar os resultados fornecendo um valor de chave de classificação e uma condição.

Note

O AWS SDK for Java também oferece um modelo de persistência de objeto, permitindo que você mapeie suas classes do lado do cliente para tabelas do DynamoDB. Essa abordagem pode reduzir a quantidade de código que você precisa escrever. Para mais informações, consulte [Java: DynamoDBMapper \(p. 228\)](#).

Veja a seguir as etapas para recuperar um item usando a API de Documento AWS SDK for Java.

1. Crie uma instância da classe `DynamoDB`.
2. Crie uma instância da classe `Table` para representar a tabela com a qual você deseja trabalhar.
3. Chame o método `query` de instância de `Table`. Você deve especificar o valor da chave de partição dos itens que deseja recuperar, juntamente com todos os parâmetros de consulta opcionais.

A resposta inclui um objeto `ItemCollection`, que fornece todos os itens retornados pela consulta.

O exemplo de código Java a seguir demonstra as tarefas anteriores. O exemplo pressupõe que você tenha uma tabela `Reply` que armazena respostas para tópicos de fórum. Para mais informações, consulte [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#).

```
Reply ( Id, ReplyDateTime, ... )
```

Cada tópico de fórum tem um ID exclusivo e pode ter zero ou mais respostas. Portanto, o atributo `Id` da tabela `Reply` é composto pelo nome e pelo assunto do fórum. O `Id` (chave de partição) e a `ReplyDateTime` (chave de classificação) compõem a chave primária composta da tabela.

A consulta a seguir recupera todas as respostas de um assunto de conversa específico. A consulta requer o nome da tabela e o valor de `Subject`.

Example

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
    .withRegion(Regions.US_WEST_2).build();
DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("Reply");

QuerySpec spec = new QuerySpec()
    .withKeyConditionExpression("Id = :v_id")
    .WithValueMap(new ValueMap()
        .withString(":v_id", "Amazon DynamoDB#DynamoDB Thread 1"));
```

```
ItemCollection<QueryOutcome> items = table.query(spec);

Iterator<Item> iterator = items.iterator();
Item item = null;
while (iterator.hasNext()) {
    item = iterator.next();
    System.out.println(item.toJSONPretty());
}
```

Especificação de parâmetros opcionais

O método `query` aceita vários parâmetros opcionais. Por exemplo, você pode opcionalmente restringir os resultados na consulta anterior para retornar respostas nas últimas duas semanas, especificando uma condição. A condição é chamada de condição de chave de classificação, pois o DynamoDB avalia a condição de consulta que você especifica de acordo com a chave de classificação da chave primária. É possível especificar outros parâmetros opcionais para recuperar apenas uma lista específica de atributos dos itens no resultado da consulta.

O exemplo de código Java a seguir recupera respostas a tópicos de threads do fórum publicadas nos últimos 15 dias. O exemplo especifica os parâmetros opcionais usando o seguinte:

- Uma `KeyConditionExpression` para recuperar as respostas de um fórum de discussão específico (chave de partição) e, dentro desse conjunto de itens, as respostas que foram postadas nos últimos 15 dias (chave de classificação).
- Uma `FilterExpression` para retornar apenas as respostas de um usuário específico. O filtro é aplicado depois que a consulta é processada, mas antes que os resultados sejam retornados ao usuário.
- Um `ValueMap` para definir os valores reais dos espaços reservados de `KeyConditionExpression`.
- Uma configuração de `ConsistentRead` de `true`, para solicitar uma leitura fortemente consistente.

Este exemplo usa um objeto `QuerySpec` que dá acesso a todos os parâmetros de entrada de baixo nível de `Query`.

Example

```
Table table = dynamoDB.getTable("Reply");

long twoWeeksAgoMilli = (new Date()).getTime() - (15L*24L*60L*60L*1000L);
Date twoWeeksAgo = new Date();
twoWeeksAgo.setTime(twoWeeksAgoMilli);
SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");
String twoWeeksAgoStr = df.format(twoWeeksAgo);

QuerySpec spec = new QuerySpec()
    .withKeyConditionExpression("Id = :v_id and ReplyDateTime > :v_reply_dt_tm")
    .withFilterExpression("PostedBy = :v_posted_by")
    .WithValueMap(new ValueMap()
        .withString(":v_id", "Amazon DynamoDB#DynamoDB Thread 1")
        .withString(":v_reply_dt_tm", twoWeeksAgoStr)
        .withString(":v_posted_by", "User B"))
    .withConsistentRead(true);

ItemCollection<QueryOutcome> items = table.query(spec);

Iterator<Item> iterator = items.iterator();
while (iterator.hasNext()) {
    System.out.println(iterator.next().toJSONPretty());
}
```

Opcionalmente, você também pode limitar o número de itens por página usando o método `withMaxPageSize`. Quando você chama o método `query`, você recebe uma `ItemCollection` que contém os itens resultantes. Em seguida, você poderá percorrer os resultados, processamento uma página por vez, até haver mais páginas.

O exemplo de código Java a seguir modifica a especificação de consulta mostrada anteriormente. Dessa vez, a especificação de consulta usa o método `withMaxPageSize`. A classe `Page` fornece um iterador que permite que o código processe os itens em cada página.

Example

```
spec.withMaxPageSize(10);

ItemCollection<QueryOutcome> items = table.query(spec);

// Process each page of results
int pageNum = 0;
for (Page<Item, QueryOutcome> page : items.pages()) {

    System.out.println("\nPage: " + ++pageNum);

    // Process each item on the current page
    Iterator<Item> item = page.iterator();
    while (item.hasNext()) {
        System.out.println(item.next().toJSONPretty());
    }
}
```

Exemplo – consulta usando Java

As tabelas a seguir armazenam informações sobre um conjunto de fóruns. Para mais informações, consulte [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#).

Note

O SDK for Java também oferece um modelo de persistência de objeto, permitindo que você mapeie suas classes do lado do cliente para tabelas do DynamoDB. Essa abordagem pode reduzir a quantidade de código que você precisa escrever. Para mais informações, consulte [Java: DynamoDBMapper \(p. 228\)](#).

Example

```
Forum ( Name, ... )
Thread ( ForumName, Subject, Message, LastPostedBy, LastPostDateTime, ... )
Reply ( Id, ReplyDateTime, Message, PostedBy, ... )
```

Neste exemplo de código Java, você executa variações para encontrar respostas para um thread “Thread do DynamoDB 1 no fórum “DynamoDB”.

- Encontrar respostas para um tópico.
- Encontre respostas para um tópico, especificando um limite sobre o número de itens por página de resultados. Se o número de itens no conjunto de resultados exceder o tamanho da página, você receberá apenas a primeira página de resultados. Esse padrão de codificação garante que o código processe todas as páginas no resultado da consulta.
- Encontrar respostas nos últimos 15 dias.
- Encontrar respostas em um intervalo de datas específico.

As duas consultas anteriores mostram como você pode especificar condições de chave de classificação para restringir os resultados da consulta e usar outros parâmetros de consulta opcionais.

Note

Este exemplo de código pressupõe que você já carregou dados no DynamoDB para sua conta seguindo as instruções no [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#) seção.

Para obter instruções detalhadas sobre como executar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples.document;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Iterator;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Item;
import com.amazonaws.services.dynamodbv2.document.ItemCollection;
import com.amazonaws.services.dynamodbv2.document.Page;
import com.amazonaws.services.dynamodbv2.document.QueryOutcome;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.document.spec.QuerySpec;
import com.amazonaws.services.dynamodbv2.document.utils.ValueMap;

public class DocumentAPIQuery {

    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
    static DynamoDB dynamoDB = new DynamoDB(client);

    static String tableName = "Reply";

    public static void main(String[] args) throws Exception {

        String forumName = "Amazon DynamoDB";
        String threadSubject = "DynamoDB Thread 1";

        findRepliesForAThread(forumName, threadSubject);
        findRepliesForAThreadSpecifyOptionalLimit(forumName, threadSubject);
        findRepliesInLast15DaysWithConfig(forumName, threadSubject);
        findRepliesPostedWithinTimePeriod(forumName, threadSubject);
        findRepliesUsingAFilterExpression(forumName, threadSubject);
    }
}
```

```
private static void findRepliesForAThread(String forumName, String threadSubject) {  
  
    Table table = dynamoDB.getTable(tableName);  
  
    String replyId = forumName + "#" + threadSubject;  
  
    QuerySpec spec = new QuerySpec().withKeyConditionExpression("Id = :v_id")  
        .withValueMap(new ValueMap().withString(":v_id", replyId));  
  
    ItemCollection<QueryOutcome> items = table.query(spec);  
  
    System.out.println("\nfindRepliesForAThread results:");  
  
    Iterator<Item> iterator = items.iterator();  
    while (iterator.hasNext()) {  
        System.out.println(iterator.next().toJSONPretty());  
    }  
  
}  
  
private static void findRepliesForAThreadSpecifyOptionalLimit(String forumName, String  
threadSubject) {  
  
    Table table = dynamoDB.getTable(tableName);  
  
    String replyId = forumName + "#" + threadSubject;  
  
    QuerySpec spec = new QuerySpec().withKeyConditionExpression("Id = :v_id")  
        .withValueMap(new ValueMap().withString(":v_id", replyId)).withMaxPageSize(1);  
  
    ItemCollection<QueryOutcome> items = table.query(spec);  
  
    System.out.println("\nfindRepliesForAThreadSpecifyOptionalLimit results:");  
  
    // Process each page of results  
    int pageNum = 0;  
    for (Page<Item, QueryOutcome> page : items.pages()) {  
  
        System.out.println("\nPage: " + ++pageNum);  
  
        // Process each item on the current page  
        Iterator<Item> item = page.iterator();  
        while (item.hasNext()) {  
            System.out.println(item.next().toJSONPretty());  
        }  
    }  
}  
  
private static void findRepliesInLast15DaysWithConfig(String forumName, String  
threadSubject) {  
  
    Table table = dynamoDB.getTable(tableName);  
  
    long twoWeeksAgoMilli = (new Date()).getTime() - (15L * 24L * 60L * 60L * 1000L);  
    Date twoWeeksAgo = new Date();  
    twoWeeksAgo.setTime(twoWeeksAgoMilli);  
    SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");  
    String twoWeeksAgoStr = df.format(twoWeeksAgo);  
  
    String replyId = forumName + "#" + threadSubject;  
  
    QuerySpec spec = new QuerySpec().withProjectionExpression("Message, ReplyDateTime,  
PostedBy")  
        .withKeyConditionExpression("Id = :v_id and ReplyDateTime <= :v_reply_dt_tm")  
        .withValueMap(new ValueMap().withString(":v_id",  
replyId).withString(":v_reply_dt_tm", twoWeeksAgoStr));
```

```
    ItemCollection<QueryOutcome> items = table.query(spec);

    System.out.println("\nfindRepliesInLast15DaysWithConfig results:");
    Iterator<Item> iterator = items.iterator();
    while (iterator.hasNext()) {
        System.out.println(iterator.next().toJSONPretty());
    }

}

private static void findRepliesPostedWithinTimePeriod(String forumName, String
threadSubject) {

    Table table = dynamoDB.getTable(tableName);

    long startDateMilli = (new Date()).getTime() - (15L * 24L * 60L * 60L * 1000L);
    long endDateMilli = (new Date()).getTime() - (5L * 24L * 60L * 60L * 1000L);
    java.text.SimpleDateFormat df = new java.text.SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss.SSS'Z'");
    String startDate = df.format(startDateMilli);
    String endDate = df.format(endDateMilli);

    String replyId = forumName + "#" + threadSubject;

    QuerySpec spec = new QuerySpec().withProjectionExpression("Message, ReplyDateTime,
PostedBy")
        .withKeyConditionExpression("Id = :v_id and ReplyDateTime between :v_start_dt
and :v_end_dt")
        .withValueMap(new ValueMap().withString(":v_id",
replyId).withString(":v_start_dt", startDate)
            .withString(":v_end_dt", endDate));

    ItemCollection<QueryOutcome> items = table.query(spec);

    System.out.println("\nfindRepliesPostedWithinTimePeriod results:");
    Iterator<Item> iterator = items.iterator();
    while (iterator.hasNext()) {
        System.out.println(iterator.next().toJSONPretty());
    }
}

private static void findRepliesUsingAFilterExpression(String forumName, String
threadSubject) {

    Table table = dynamoDB.getTable(tableName);

    String replyId = forumName + "#" + threadSubject;

    QuerySpec spec = new QuerySpec().withProjectionExpression("Message, ReplyDateTime,
PostedBy")
        .withKeyConditionExpression("Id = :v_id").withFilterExpression("PostedBy
= :v_postedby")
        .withValueMap(new ValueMap().withString(":v_id",
replyId).withString(":v_postedby", "User B"));

    ItemCollection<QueryOutcome> items = table.query(spec);

    System.out.println("\nfindRepliesUsingAFilterExpression results:");
    Iterator<Item> iterator = items.iterator();
    while (iterator.hasNext()) {
        System.out.println(iterator.next().toJSONPretty());
    }
}
```

Consulta de tabelas e índices: .NET

A operação `Query` permite consultar uma tabela ou um índice secundário no Amazon DynamoDB. Você deve fornecer um valor de chave de partição e uma condição de igualdade. Se a tabela ou o índice tiver uma chave de classificação, você poderá refinar os resultados fornecendo um valor de chave de classificação e uma condição.

As seguintes são as etapas para consultar uma tabela usando a API de baixo nível do AWS SDK for .NET.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Crie uma instância da classe `QueryRequest` e forneça parâmetros de operação de consulta.
3. Execute a query fornecendo o método `QueryRequest` que você criou na etapa anterior.

A resposta inclui o objeto `QueryResult`, que fornece todos os itens retornados pela consulta.

O exemplo de código C# a seguir demonstra as tarefas anteriores. O código pressupõe que você tem uma tabela `Reply` que armazena respostas para threads de fórum. Para mais informações, consulte [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#).

Example

```
Reply Id, ReplyDateTime, ... )
```

Cada tópico de fórum tem um ID exclusivo e pode ter zero ou mais respostas. Portanto, a chave primária é composta de `Id` (chave de partição) e `ReplyDateTime` (chave de classificação).

A consulta a seguir recupera todas as respostas de um assunto de conversa específico. A consulta requer o nome da tabela e o valor de `Subject`.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();

var request = new QueryRequest
{
    TableName = "Reply",
    KeyConditionExpression = "Id = :v_Id",
    ExpressionAttributeValues = new Dictionary<string, AttributeValue> {
        {"v_Id", new AttributeValue { S = "Amazon DynamoDB#DynamoDB Thread 1" }}}
};

var response = client.Query(request);

foreach (Dictionary<string, AttributeValue> item in response.Items)
{
    // Process the result.
    PrintItem(item);
}
```

Especificação de parâmetros opcionais

O método `Query` aceita vários parâmetros opcionais. Por exemplo, você pode opcionalmente refinar o resultado da consulta na consulta anterior para retornar respostas nas últimas duas semanas, especificando uma condição. A condição é chamada de condição da chave de classificação, pois o DynamoDB avalia a condição de consulta que você especifica de acordo com a chave de classificação.

da chave primária. É possível especificar outros parâmetros opcionais para recuperar apenas uma lista específica de atributos dos itens no resultado da consulta. Para obter mais informações, consulte[Consulte](#).

O exemplo de código C# a seguir recupera respostas a threads de fórum postadas nos últimos 15 dias. O exemplo especifica os seguintes parâmetros opcionais:

- Uma `KeyConditionExpression` para recuperar somente as respostas nos últimos 15 dias.
- Um parâmetro `ProjectionExpression` para especificar uma lista de atributos a ser recuperada para itens nos resultados da consulta.
- Um parâmetro `ConsistentRead` para realizar uma leitura fortemente consistente.

Example

```
DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
string twoWeeksAgoString = twoWeeksAgoDate.ToString(AWSSDKUtils.ISO8601DateFormat);

var request = new QueryRequest
{
    TableName = "Reply",
    KeyConditionExpression = "Id = :v_Id and ReplyDateTime > :v_twoWeeksAgo",
    ExpressionAttributeValues = new Dictionary<string, AttributeValue> {
        {"":v_Id", new AttributeValue { S = "Amazon DynamoDB#DynamoDB Thread 2" }},
        {"":v_twoWeeksAgo", new AttributeValue { S = twoWeeksAgoString }}
    },
    ProjectionExpression = "Subject, ReplyDateTime, PostedBy",
    ConsistentRead = true
};

var response = client.Query(request);

foreach (Dictionary<string, AttributeValue> item in response.Items)
{
    // Process the result.
    PrintItem(item);
}
```

Opcionalmente, você também pode limitar o tamanho da página, ou o número de itens por página, usando o parâmetro opcional `Limit`. Cada vez que você executar o `Query`, você obtém uma página de resultados que possui o número especificado de itens. Para buscar a próxima página, execute o `Query` novamente, fornecendo o valor de chave primária do último item na página anterior, para que o método possa recuperar o próximo conjunto de itens. Você fornece essas informações na solicitação, definindo a propriedade `ExclusiveStartKey`. Inicialmente, essa propriedade pode ser nula. Para recuperar páginas subsequentes, você deve atualizar esse valor de propriedade para a chave primária do último item da página anterior.

O seguinte exemplo de código C# consulta a tabela `Reply`. Na solicitação, ele especifica os parâmetros opcionais `Limit` e `ExclusiveStartKey`. O loop do/while continua a verificar uma página por vez até que `LastEvaluatedKey` retorne um valor nulo.

Example

```
Dictionary<string,AttributeValue> lastKeyEvaluated = null;

do
{
    var request = new QueryRequest
    {
        TableName = "Reply",
        KeyConditionExpression = "Id = :v_Id",
        Limit = 2
    };

    var response = client.Query(request);

    foreach (Dictionary<string, AttributeValue> item in response.Items)
    {
        // Process the result.
        PrintItem(item);
    }

    if (response.LastEvaluatedKey != null)
    {
        lastKeyEvaluated = response.LastEvaluatedKey;
        request.ExclusiveStartKey = lastKeyEvaluated;
    }
}
```

```
ExpressionAttributeValues = new Dictionary<string, AttributeValue> {
    {":v_Id", new AttributeValue { S = "Amazon DynamoDB#DynamoDB Thread 2" }}
},
// Optional parameters.
Limit = 1,
ExclusiveStartKey = lastKeyEvaluated
};

var response = client.Query(request);

// Process the query result.
foreach (Dictionary<string, AttributeValue> item in response.Items)
{
    PrintItem(item);
}

lastKeyEvaluated = response.LastEvaluatedKey;
} while (lastKeyEvaluated != null && lastKeyEvaluated.Count != 0);
```

Exemplo - Consulta usando oAWS SDK for .NET

As tabelas a seguir armazenam informações sobre um conjunto de fóruns. Para mais informações, consulte [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#).

Example

```
Forum ( Name, ... )
Thread ( ForumName, Subject, Message, LastPostedBy, LastPostDateTime, ... )
Reply ( Id, ReplyDateTime, Message, PostedBy, ... )
```

Neste exemplo, você executa variações de “Encontrar respostas para um thread “Thread do DynamoDB 1 do” no fórum “DynamoDB”.

- Encontrar respostas para um tópico.
- Encontrar respostas para um tópico. Especifique o parâmetro de consulta `Limit` para definir o tamanho da página.

Esta função ilustra o uso da paginação para processar resultado de várias páginas. O DynamoDB tem um limite de tamanho de página e, se o seu resultado exceder esse limite, você receberá apenas a primeira página de resultados. Esse padrão de codificação assegura que o seu código processe todas as páginas no resultado da consulta.

- Encontrar respostas nos últimos 15 dias.
- Encontrar respostas em um intervalo de datas específico.

As duas consultas anteriores mostram como você pode especificar condições de chave de classificação para restringir os resultados da consulta e usar outros parâmetros de consulta opcionais.

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
```

```
* the License is located at
*
* http://aws.amazon.com/apache2.0/
*
* This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
* CONDITIONS OF ANY KIND, either express or implied. See the License for the
* specific language governing permissions and limitations under the License.
*/
using System;
using System.Collections.Generic;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;
using Amazon.Runtime;
using Amazon.Util;

namespace com.amazonaws.codesamples
{
    class LowLevelQuery
    {
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();

        static void Main(string[] args)
        {
            try
            {
                // Query a specific forum and thread.
                string forumName = "Amazon DynamoDB";
                string threadSubject = "DynamoDB Thread 1";

                FindRepliesForAThread(forumName, threadSubject);
                FindRepliesForAThreadSpecifyOptionalLimit(forumName, threadSubject);
                FindRepliesInLast15DaysWithConfig(forumName, threadSubject);
                FindRepliesPostedWithinTimePeriod(forumName, threadSubject);

                Console.WriteLine("Example complete. To continue, press Enter");
                Console.ReadLine();
            }
            catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message);
Console.ReadLine(); }
            catch (AmazonServiceException e) { Console.WriteLine(e.Message);
Console.ReadLine(); }
            catch (Exception e) { Console.WriteLine(e.Message); Console.ReadLine(); }
        }

        private static void FindRepliesPostedWithinTimePeriod(string forumName, string
threadSubject)
        {
            Console.WriteLine("**** Executing FindRepliesPostedWithinTimePeriod() ***");
            string replyId = forumName + "#" + threadSubject;
            // You must provide date value based on your test data.
            DateTime startDate = DateTime.UtcNow - TimeSpan.FromDays(21);
            string start = startDate.ToString(AWSSDKUtils.ISO8601DateFormat);

            // You provide date value based on your test data.
            DateTime endDate = DateTime.UtcNow - TimeSpan.FromDays(5);
            string end = endDate.ToString(AWSSDKUtils.ISO8601DateFormat);

            var request = new QueryRequest
            {
                TableName = "Reply",
                ReturnConsumedCapacity = "TOTAL",
                KeyConditionExpression = "Id = :v_replyId and ReplyDateTime
between :v_start and :v_end",
                ExpressionAttributeValues = new Dictionary<string, AttributeValue> {
                    {":v_replyId", new AttributeValue {
                        S = replyId
                }}}
            };
        }
    }
}
```

```
        }},
        {"<v_start", new AttributeValue {
            S = start
        }},
        {"<v_end", new AttributeValue {
            S = end
        }}
    }
};

var response = client.Query(request);

Console.WriteLine("\nNo. of reads used (by query in
FindRepliesPostedWithinTimePeriod) {0}",
    response.ConsumedCapacity.CapacityUnits);
foreach (Dictionary<string, AttributeValue> item
    in response.Items)
{
    PrintItem(item);
}
Console.WriteLine("To continue, press Enter");
Console.ReadLine();
}

private static void FindRepliesInLast15DaysWithConfig(string forumName, string
threadSubject)
{
    Console.WriteLine("**** Executing FindRepliesInLast15DaysWithConfig() ****");
    string replyId = forumName + "#" + threadSubject;

    DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
    string twoWeeksAgoString =
        twoWeeksAgoDate.ToString(AWSSDKUtils.ISO8601DateFormat);

    var request = new QueryRequest
    {
        TableName = "Reply",
        ReturnConsumedCapacity = "TOTAL",
        KeyConditionExpression = "Id = :v_replyId and ReplyDateTime > :v_interval",
        ExpressionAttributeValues = new Dictionary<string, AttributeValue> {
            {"<v_replyId", new AttributeValue {
                S = replyId
            }},
            {"<v_interval", new AttributeValue {
                S = twoWeeksAgoString
            }}
        },
        // Optional parameter.
        ProjectionExpression = "Id, ReplyDateTime, PostedBy",
        // Optional parameter.
        ConsistentRead = true
    };

    var response = client.Query(request);

    Console.WriteLine("No. of reads used (by query in
FindRepliesInLast15DaysWithConfig) {0}",
        response.ConsumedCapacity.CapacityUnits);
    foreach (Dictionary<string, AttributeValue> item
        in response.Items)
    {
        PrintItem(item);
    }
    Console.WriteLine("To continue, press Enter");
    Console.ReadLine();
}
```

```
        }

        private static void FindRepliesForAThreadSpecifyOptionalLimit(string forumName,
string threadSubject)
{
    Console.WriteLine("**** Executing FindRepliesForAThreadSpecifyOptionalLimit()"
***");
    string replyId = forumName + "#" + threadSubject;

    Dictionary<string, AttributeValue> lastKeyEvaluated = null;
    do
    {
        var request = new QueryRequest
        {
            TableName = "Reply",
            ReturnConsumedCapacity = "TOTAL",
            KeyConditionExpression = "Id = :v_replyId",
            ExpressionAttributeValues = new Dictionary<string, AttributeValue> {
                {":v_replyId", new AttributeValue {
                    S = replyId
                }}
            },
            Limit = 2, // The Reply table has only a few sample items. So the page
size is smaller.
            ExclusiveStartKey = lastKeyEvaluated
        };

        var response = client.Query(request);

        Console.WriteLine("No. of reads used (by query in
FindRepliesForAThreadSpecifyLimit) {0}\n",
                           response.ConsumedCapacity.CapacityUnits);
        foreach (Dictionary<string, AttributeValue> item
                 in response.Items)
        {
            PrintItem(item);
        }
        lastKeyEvaluated = response.LastEvaluatedKey;
    } while (lastKeyEvaluated != null && lastKeyEvaluated.Count != 0);

    Console.WriteLine("To continue, press Enter");

    Console.ReadLine();
}

private static void FindRepliesForAThread(string forumName, string threadSubject)
{
    Console.WriteLine("**** Executing FindRepliesForAThread() ***");
    string replyId = forumName + "#" + threadSubject;

    var request = new QueryRequest
    {
        TableName = "Reply",
        ReturnConsumedCapacity = "TOTAL",
        KeyConditionExpression = "Id = :v_replyId",
        ExpressionAttributeValues = new Dictionary<string, AttributeValue> {
            {":v_replyId", new AttributeValue {
                S = replyId
            }}
        },
    };

    var response = client.Query(request);
    Console.WriteLine("No. of reads used (by query in FindRepliesForAThread)
{0}\n",
```

```
        response.ConsumedCapacity.CapacityUnits);
    foreach (Dictionary<string, AttributeValue> item in response.Items)
    {
        PrintItem(item);
    }
    Console.WriteLine("To continue, press Enter");
    Console.ReadLine();
}

private static void PrintItem(
    Dictionary<string, AttributeValue> attributeList)
{
    foreach (KeyValuePair<string, AttributeValue> kvp in attributeList)
    {
        string attributeName = kvp.Key;
        AttributeValue value = kvp.Value;

        Console.WriteLine(
            attributeName + " " +
            (value.S == null ? "" : "S=[ " + value.S + " ]") +
            (value.N == null ? "" : "N=[ " + value.N + " ]") +
            (value.SS == null ? "" : "SS=[ " + string.Join(",", value.SS.ToArray()) +
            " ]" ) +
            (value.NS == null ? "" : "NS=[ " + string.Join(",", value.NS.ToArray()) +
            " ]"));
    }
    Console.WriteLine("*****");
}
}
```

Como trabalhar com verificações no DynamoDB

A `Scan` no Amazon DynamoDB lê cada item de uma tabela ou de um índice secundário. Por padrão, uma operação `Scan` retorna todos os atributos de dados de cada item na tabela ou índice. Você pode usar o parâmetro `ProjectionExpression` para que a operação `Scan` retorne apenas alguns dos atributos, em vez de todos eles.

`Scan` sempre retorna um conjunto de resultados. Se não forem encontrados itens correspondentes, o conjunto de resultados estará vazio.

Uma única `Scan` solicitação de pode recuperar um máximo de 1 MB de dados. Como opção, o DynamoDB pode aplicar uma expressão de filtro a esses dados, o que refina os resultados antes que eles sejam retornados para o usuário.

Tópicos

- [Expressões de filtro de Scan \(p. 509\)](#)
- [Limitação do número de itens no conjunto de resultados \(p. 509\)](#)
- [Paginação de resultados \(p. 509\)](#)
- [Contagem dos itens nos resultados \(p. 511\)](#)
- [Unidades de capacidade consumidas por Scan \(p. 511\)](#)
- [Consistência de leitura de Scan \(p. 512\)](#)
- [Scan em paralelo \(p. 512\)](#)
- [Verificação de tabelas e índices: Java \(p. 514\)](#)
- [Verificação de tabelas e índices: .NET \(p. 521\)](#)

Expressões de filtro de Scan

Se você precisar refinar ainda mais os resultados de Scan, existe a opção de utilizar uma expressão de filtro. Uma expressão de filtro determina quais itens dos resultados de Scan devem ser retornados para você. Todos os outros resultados serão descartados.

Uma expressão de filtro é aplicada depois que uma operação Scan é concluída, mas antes que os resultados sejam retornados. Portanto, uma operação Scan consome a mesma quantidade de capacidade de leitura, independentemente de uma expressão de filtro estar presente.

A ScanA operação de pode recuperar um máximo de 1 MB de dados. Esse limite se aplica antes de a expressão de filtro ser avaliada.

Com o Scan, você pode especificar quaisquer atributos em uma expressão de filtro, incluindo atributos de chave de partição ou de chave de classificação.

A sintaxe de uma expressão de filtro é idêntica à de uma expressão de condição. As expressões de filtro podem usar os mesmos comparadores, funções e operadores lógicos que uma expressão de condição. Para obter mais informações, [Expressões de condição \(p. 422\)](#).

Example

O exemplo da AWS Command Line Interface (AWS CLI) a seguir verifica a tabela Thread e retorna apenas os itens que foram publicados pela última vez por um usuário específico.

```
aws dynamodb scan \  
    --table-name Thread \  
    --filter-expression "LastPostedBy = :name" \  
    --expression-attribute-values '{":name": {"S": "User A"}}'
```

Limitação do número de itens no conjunto de resultados

A operação Scan permite que você limite o número de itens retornados no resultado. Para fazer isso, defina o parâmetro Limit como o número máximo de itens que você deseja que a operação Scan retorne, antes da avaliação da expressão de filtro.

Por exemplo, suponha que você execute a operação Scan em uma tabela, com um valor de Limit igual a 6 e sem uma expressão de filtro. O resultado de ScanO resultado de contém os primeiros seis itens da tabela.

Agora suponha que você adicione uma expressão de filtro à operação Scan. Nesse caso, o DynamoDB aplica a expressão de filtro aos seis itens que foram retornados e descarta os que não correspondem. O resultado final de Scan contém 6 itens ou menos, dependendo do número de itens que foram filtrados.

Paginação de resultados

DynamoDBpaginadosOs resultados doScanOperações Com a paginação, oScanOs resultados são divididos em “páginas” de dados que têm 1 MB de tamanho (ou menos). Um aplicativo pode processar a primeira página de resultados e, em seguida, a segunda página, e assim por diante.

Uma únicaScanO retorna apenas um conjunto de resultados que estão dentro do limite de tamanho de 1 MB. Para determinar se há mais resultados e para recuperá-los em uma página por vez, os aplicativos devem fazer o seguinte:

1. Examine o resultado de Scan de baixo nível:

- Se o resultado contiver um elemento LastEvaluatedKey, prossiga para a etapa 2.

- Se não houver um `LastEvaluatedKey` no resultado, não haverá mais itens a serem recuperados.
2. Construa a nova solicitação de `Scan` com os mesmos parâmetros da anterior. No entanto, desta vez, use o valor de `LastEvaluatedKey` da etapa 1 como o parâmetro `ExclusiveStartKey` na nova solicitação de `Scan`.
 3. Execute a nova solicitação de `Scan`.
 4. Vá para a etapa 1.

Em outras palavras, o valor `LastEvaluatedKey` de uma resposta de `Scan` deve ser usado como `ExclusiveStartKey` da próxima solicitação de `Scan`. Se não houver um elemento `LastEvaluatedKey` em uma resposta de `Scan`, você terá recuperado a página de resultados final. (A ausência de `LastEvaluatedKey` é a única forma de você saber que chegou ao fim do conjunto de resultados.)

Você pode usar a AWS CLI para visualizar esse comportamento. O AWS CLI envia baixo nível `doScan` solicitações ao DynamoDB, repetidamente, até `LastEvaluatedKey` não está mais presente nos resultados. Considere o seguinte exemplo da AWS CLI que verifica toda a tabela `Movies`, mas retorna apenas os filmes de um determinado gênero.

```
aws dynamodb scan \  
  --table-name Movies \  
  --projection-expression "title" \  
  --filter-expression 'contains(info.genres,:gen)' \  
  --expression-attribute-values '{":gen":{"S":"Sci-Fi"}}' \  
  --page-size 100 \  
  --debug
```

Normalmente, a AWS CLI lida com a paginação automaticamente. No entanto, neste exemplo, o parâmetro `--page-size` da AWS CLI limita o número de itens por página. O parâmetro de `--debug` imprime as informações de baixo nível sobre solicitações e respostas.

Se você executar o exemplo, a primeira resposta do DynamoDB será similar à seguinte.

```
2017-07-07 12:19:14,389 - MainThread - botocore.parsers - DEBUG - Response body:  
b'{"Count":7,"Items":[{"title":{"S":"Monster on the Campus"}}, {"title":{"S":"+1"}},  
 {"title":{"S":"100 Degrees Below Zero"}}, {"title":{"S":"About Time"}}, {"title":{"S":"After  
Earth"}},  
 {"title":{"S":"Age of Dinosaurs"}}, {"title":{"S":"Cloudy with a Chance of Meatballs 2"}},  
 {"LastEvaluatedKey":{"year":{"N":"2013"}, "title":{"S":"Curse of  
Chucky"}}, "ScannedCount":100}'
```

O `LastEvaluatedKey` na resposta indica que nem todos os itens foram recuperados. O AWS CLI em seguida, emite outro `Scan` ao DynamoDB. Essa solicitação e o padrão de resposta continuam, até a resposta final.

```
2017-07-07 12:19:17,830 - MainThread - botocore.parsers - DEBUG - Response body:  
b'{"Count":1,"Items":[{"title":{"S":"WarGames"}}], "ScannedCount":6}'
```

A ausência de `LastEvaluatedKey` indica que não há mais itens a serem recuperados.

Note

OAWSOs SDKs lidam com as respostas de API do DynamoDB de baixo nível (incluindo a presença ou a ausência de `LastEvaluatedKey`) e fornecer várias abstrações para paginação `ScanResultados`. Por exemplo, o SDK para interface de documento Java fornece `java.util.Iterator` suporte a, para que você possa abordar um resultado de cada vez.

Para obter exemplos de código em várias linguagens de programação, consulte o [Guia de conceitos básicos do Amazon DynamoDB](#): O e a AWS Documentação do SDK para o seu idioma.

Contagem dos itens nos resultados

Além dos itens que correspondem aos seus critérios, a resposta de Scan contém os elementos a seguir:

- **ScannedCount**— O número de itens avaliados, antes de qualquerScanFilteré aplicada. Um valor alto de ScannedCount com poucos ou nenhum resultado de Count indica uma operação Scan ineficiente. Se você não tiver usado um filtro na solicitação, ScannedCount será o mesmo que Count.
- **Count**— O número de itens que permanecem,DepoisUma expressão de filtro (se houver) foi aplicada.

Note

Se você não usar uma expressão de filtro, ScannedCount e Count terão o mesmo valor.

Se o tamanho doScanconjunto de resultados é maior que 1 MB,ScannedCounteCountrepresentam apenas uma contagem parcial do total de itens. Você precisa executar várias operações Scan para recuperar todos os resultados (consulte [Paginação de resultados \(p. 509\)](#)).

Cada resposta de Scan contém os valores de ScannedCount e de Count dos itens que foram processados pela solicitação de Scan específica. Para obter os totais gerais de todas as solicitações de Scan, você pode manter um total em execução de ScannedCount e de Count.

Unidades de capacidade consumidas por Scan

Você podeScanqualquer tabela ou índice secundário. ScanAs operações consomem unidades de capacidade de leitura da seguinte forma.

Se você executar uma operação Scan em...	O DynamoDB consumirá unidades de capacidade de leitura de...
Tabela	A capacidade de leitura provisionada da tabela.
Índice secundário global	A capacidade de leitura provisionada do índice.
Índice secundário local	A capacidade de leitura provisionada da tabela-base.

Por padrão, uma operação Scan não retorna quaisquer dados sobre quanta capacidade de leitura ela consome. Entretanto, você pode especificar o parâmetro `ReturnConsumedCapacity` em uma solicitação de Scan para obter essas informações. A seguir estão as configurações válidas de `ReturnConsumedCapacity`:

- **NONE**— nenhum dado de capacidade consumida é retornado. (Esse é o padrão.)
- **TOTAL**— A resposta inclui o número agregado de unidades de capacidade de leitura consumidas.
- **INDEXES**— a resposta mostra o número agregado de unidades de capacidade de leitura consumidas, junto com a capacidade consumida para cada tabela e índice acessados.

O DynamoDB calcula o número de unidades de capacidade de leitura consumidas com base no tamanho do item, não na quantidade de dados que é retornada para um aplicativo. Por esse motivo, o número de unidades de capacidade consumidas será o mesmo, independentemente de você solicitar todos os atributos (o comportamento padrão) ou apenas alguns deles (usando uma expressão de projeção). O número também é o mesmo quer você use uma expressão de filtro ou não.

Consistência de leitura de Scan

Uma operação Scan executa leituras eventualmente consistentes, por padrão. Isso significa que os resultados de Scan talvez não refletem as alterações causadas pelas operações PutItem ou UpdateItem concluídas recentemente. Para mais informações, consulte [Consistência de leituras \(p. 17\)](#).

Se você precisar de leituras fortemente consistentes, como a hora em que a operação Scan começa, defina o parâmetro ConsistentRead como true na solicitação de Scan. Isso garante que todas as operações de gravação que foram concluídas antes de Scan ter começado sejam incluídas na resposta de Scan.

Configurar ConsistentRead para true pode ser útil em cenários de backup ou replicação da tabela, juntamente com o [DynamoDB Streams](#). Primeiro, você usa Scan com ConsistentRead definida como true para obter uma cópia consistente dos dados na tabela. Durante a Scan, o DynamoDB Streams registra qualquer atividade de gravação adicional que ocorra na tabela. Após a conclusão de Scan, você pode aplicar a atividade de gravação do fluxo à tabela.

Note

Uma operação Scan com ConsistentRead definida como true consome duas vezes mais unidades de capacidade de leitura, em comparação com deixar ConsistentRead com seu valor padrão (false).

Scan em paralelo

Por padrão, o ScanA operação processa os dados sequencialmente. O Amazon DynamoDB retorna os dados do para o aplicativo em incrementos de 1 MB e um aplicativo realiza operações adicionais Scan para recuperar os próximos 1 MB de dados.

Quanto maior a tabela ou índice que está sendo verificado, mais tempo o ScanA leva para completar. Além disso, um ScanA sequencial pode nem sempre ser capaz de usar totalmente a capacidade de taxa de transferência de leitura provisionada: Mesmo que o DynamoDB distribua os dados de uma tabela grande em várias partições físicas, um ScanA operação de pode ler apenas uma partição por vez. Por esse motivo, o throughput de uma operação Scan é restrito pelo throughput máximo de uma única partição.

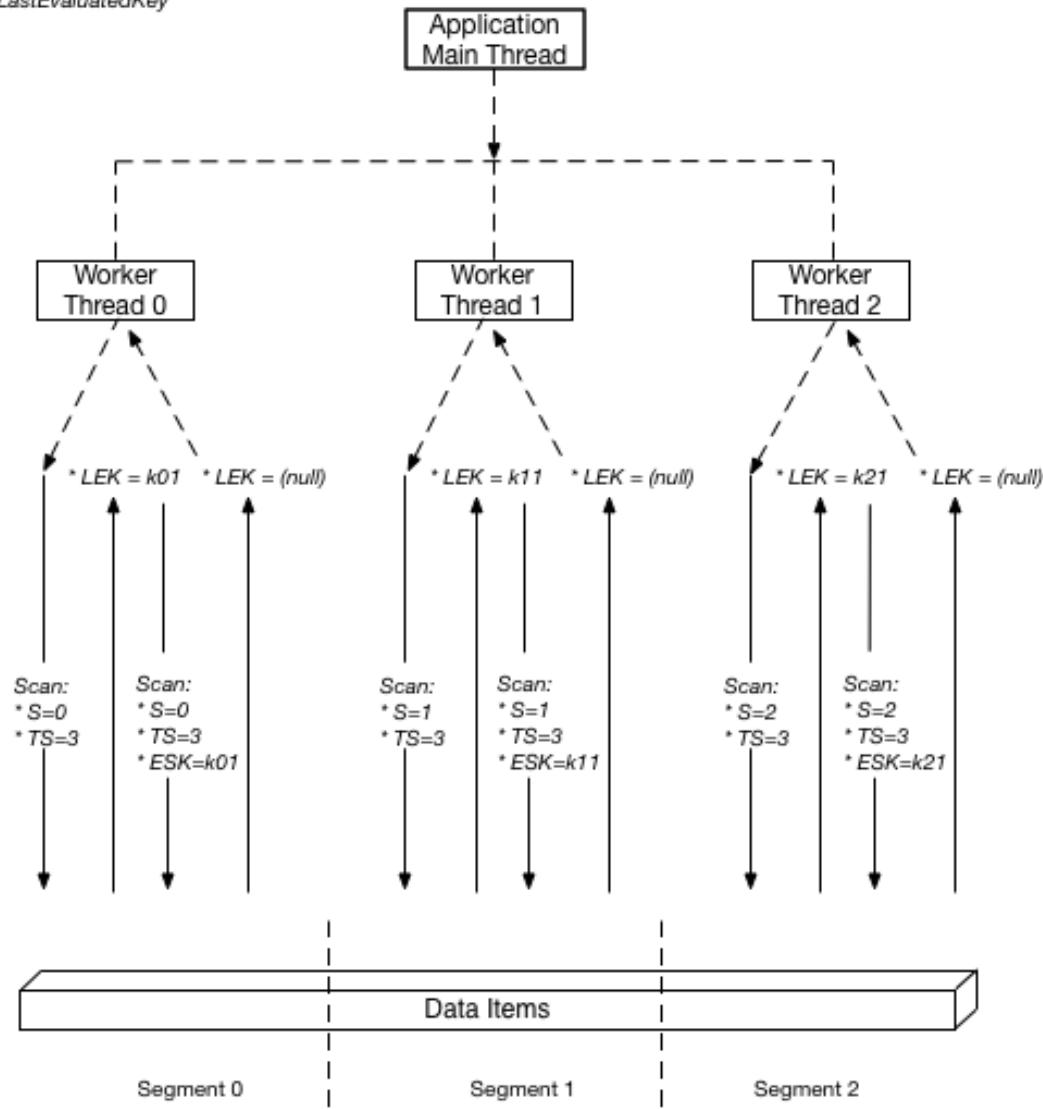
Para resolver esses problemas, o ScanA operação do pode dividir logicamente uma tabela ou índice secundário em vários Segmentos do, com vários funcionários de aplicativos verificando os segmentos em paralelo. Cada operador pode ser um thread (em linguagens de programação que aceitam multithreading) ou um processo do sistema operacional. Para executar uma verificação em paralelo, cada operador emite sua própria solicitação de Scan com os seguintes parâmetros:

- **Segment**— Um segmento a ser verificado por um determinado operador. Cada operador deve usar um valor diferente para Segment.
- **TotalSegments**— O número total de segmentos para a verificação em paralelo. Esse valor deve ser o mesmo que o número de operadores que o seu aplicativo usará.

O diagrama a seguir mostra como um aplicativo multithread executa uma operação do Scan com três graus de paralelismo.

S: Segment
TS: TotalSegments

ESK: ExclusiveStartKey
LEK: LastEvaluatedKey



Neste diagrama, o aplicativo gera três threads e atribui um número a cada thread. (Segmentos são baseados em zero, portanto, o primeiro número é sempre 0.) Cada thread emite umScan solicitação, configuraçãoSegment para o seu número designado e definiçãoTotalSegmentsPara 3. Cada thread verifica seu segmento designado, recuperando dados de 1 MB por vez e retorna os dados para o thread principal do aplicativo.

Os valores paraSegmentTotalSegmentsAplicar ao indivíduoScan, e você pode usar valores diferentes a qualquer momento. Talvez você precise experimentar com esses valores, e o número de operadores que usa, até que seu aplicativo atinja o melhor desempenho.

Note

Uma operação Scan em paralelo com um grande número de operadores pode facilmente consumir todo o throughput provisionado da tabela ou índice que está sendo verificado. É melhor

evitar tais verificações, se a tabela ou índice também incorrer em uso intensivo de atividades de leitura ou de gravação de outros aplicativos.

Para controlar a quantidade de dados retornados por solicitação, use o `Limit` parâmetro. Isso pode ajudar a evitar situações em que um operador consome todo o throughput provisionado, às custas de todos os outros operadores.

Verificação de tabelas e índices: Java

O `Scan` lê todos os itens em uma tabela ou índice no Amazon DynamoDB.

Veja a seguir as etapas para realizar uma verificação de uma tabela usando a API de documento do AWS SDK for Java.

1. Crie uma instância da classe `AmazonDynamoDB`.
2. Crie uma instância da classe `ScanRequest` e forneça o parâmetro de verificação.
O único parâmetro obrigatório é o nome da tabela.
3. Execute a `scan` forneça o método `ScanRequest` que você criou na etapa anterior.

A seguinte tabela `Reply` armazena respostas para threads de fóruns.

Example

```
Reply ( Id, ReplyDateTime, Message, PostedBy )
```

A tabela mantém todas as respostas para vários threads de fóruns. Portanto, a chave primária é composta de `Id` (chave de partição) e `ReplyDateTime` (chave de classificação). O exemplo de código Java a seguir verifica a tabela inteira. A instância de `ScanRequest` especifica o nome da tabela a ser verificada.

Example

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();

ScanRequest scanRequest = new ScanRequest()
    .withTableName("Reply");

ScanResult result = client.scan(scanRequest);
for (Map<String, AttributeValue> item : result.getItems()){
    printItem(item);
}
```

Especificação de parâmetros opcionais

O método `scan` aceita vários parâmetros opcionais. Por exemplo, opcionalmente, você pode usar uma expressão de filtro para filtrar o resultado da verificação. Em uma expressão de filtro, você pode especificar uma condição e nomes e valores de atributos nos quais você deseja avaliar a condição. Para obter mais informações, consulte [Scan](#).

O exemplo de Java a seguir verifica a tabela `ProductCatalog` para localizar itens com preço menor que 0. O exemplo especifica os seguintes parâmetros opcionais:

- Uma expressão de filtro para recuperar apenas os itens com preço inferior a 0 (condição de erro).
- Uma lista de atributos para recuperar itens nos resultados da consulta.

Example

```
Map<String, AttributeValue> expressionAttributeValues =
    new HashMap<String, AttributeValue>();
expressionAttributeValues.put(":val", new AttributeValue().withN("0"));

ScanRequest scanRequest = new ScanRequest()
    .withTableName("ProductCatalog")
    .withFilterExpression("Price < :val")
    .withProjectionExpression("Id")
    .withExpressionAttributeValues(expressionAttributeValues);

ScanResult result = client.scan(scanRequest);
for (Map<String, AttributeValue> item : result.getItems()) {
    printItem(item);
}
```

Você também pode, opcionalmente, limitar o tamanho da página, ou o número de itens por página, usando o método `withLimit` da solicitação de verificação. Cada vez que você executar o comando `scan`, você obtém uma página de resultados que possui o número de itens especificado. Para obter a próxima página, execute `scan` novamente, fornecendo o valor de chave primária do último item da página anterior, para que o método `scan` possa retornar o próximo conjunto de itens. Você fornece essas informações na solicitação, usando o método `withExclusiveStartKey`. Inicialmente, o parâmetro desse método pode ser nulo. Para recuperar páginas subsequentes, você deve atualizar esse valor de propriedade para a chave primária do último item da página anterior.

O exemplo de código Java a seguir verifica a tabela `ProductCatalog` inteira. Na solicitação, os métodos `withLimit` e `withExclusiveStartKey` são usados. O loop `do/while` continua a verificar uma página por vez, até que o método `getLastEvaluatedKey` do resultado retorne um valor null.

Example

```
Map<String, AttributeValue> lastKeyEvaluated = null;
do {
    ScanRequest scanRequest = new ScanRequest()
        .withTableName("ProductCatalog")
        .withLimit(10)
        .withExclusiveStartKey(lastKeyEvaluated);

    ScanResult result = client.scan(scanRequest);
    for (Map<String, AttributeValue> item : result.getItems()){
        printItem(item);
    }
    lastKeyEvaluated = result.getLastEvaluatedKey();
} while (lastKeyEvaluated != null);
```

Exemplo – verificação usando Java

O exemplo de código Java a seguir fornece um exemplo de trabalho que verifica a tabela `ProductCatalog` para encontrar itens com preço menor que 100.

Note

O SDK for Java também oferece um modelo de persistência de objeto, permitindo que você mapeie suas classes do lado do cliente para tabelas do DynamoDB. Essa abordagem pode reduzir a quantidade de código que você precisa escrever. Para mais informações, consulte [Java: DynamoDBMapper \(p. 228\)](#).

Note

Este exemplo de código pressupõe que você já carregou dados no DynamoDB para sua conta seguindo as instruções no[Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#)seção.

Para obter instruções detalhadas sobre como executar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
package com.amazonaws.codesamples.document;  
  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.Map;  
  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.ItemCollection;  
import com.amazonaws.services.dynamodbv2.document.ScanOutcome;  
import com.amazonaws.services.dynamodbv2.document.Table;  
  
public class DocumentAPIScan {  
  
    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();  
    static DynamoDB dynamoDB = new DynamoDB(client);  
    static String tableName = "ProductCatalog";  
  
    public static void main(String[] args) throws Exception {  
  
        findProductsForPriceLessThanOneHundred();  
    }  
  
    private static void findProductsForPriceLessThanOneHundred() {  
  
        Table table = dynamoDB.getTable(tableName);  
  
        Map<String, Object> expressionAttributeValues = new HashMap<String, Object>();  
        expressionAttributeValues.put(":pr", 100);  
  
        ItemCollection<ScanOutcome> items = table.scan("Price < :pr", // FilterExpression  
            "Id, Title, ProductCategory, Price", // ProjectionExpression  
            null, // ExpressionAttributeNames - not used in this example  
            expressionAttributeValues);  
  
        System.out.println("Scan of " + tableName + " for items with a price less than  
100.");  
        Iterator<Item> iterator = items.iterator();  
        while (iterator.hasNext()) {
```

```
        System.out.println(iterator.next().toJSONPretty());
    }
}
```

Exemplo – verificação paralela usando Java

O exemplo de código Java a seguir demonstra uma verificação paralela. O programa exclui e recria uma tabela chamada `ParallelScanTest` e carrega os dados na tabela. Quando a carga de dados for concluída, o programa criará vários threads e emitirá solicitações `Scan` paralelas. O programa imprime estatísticas de tempo de execução para cada solicitação paralela.

Note

O SDK for Java também oferece um modelo de persistência de objeto, permitindo que você mapeie suas classes do lado do cliente para tabelas do DynamoDB. Essa abordagem pode reduzir a quantidade de código que você precisa escrever. Para mais informações, consulte [Java: DynamoDBMapper \(p. 228\)](#).

Note

Este exemplo de código pressupõe que você já carregou dados no DynamoDB para sua conta seguindo as instruções no [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#) seção.

Para obter instruções detalhadas sobre como executar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples.document;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Item;
import com.amazonaws.services.dynamodbv2.document.ItemCollection;
```

```
import com.amazonaws.services.dynamodbv2.document.ScanOutcome;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.document.spec.ScanSpec;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;

public class DocumentAPIParallelScan {

    // total number of sample items
    static int scanItemCount = 300;

    // number of items each scan request should return
    static int scanItemLimit = 10;

    // number of logical segments for parallel scan
    static int parallelScanThreads = 16;

    // table that will be used for scanning
    static String parallelScanTestTableName = "ParallelScanTest";

    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
    static DynamoDB dynamoDB = new DynamoDB(client);

    public static void main(String[] args) throws Exception {
        try {

            // Clean up the table
            deleteTable(parallelScanTestTableName);
            createTable(parallelScanTestTableName, 10L, 5L, "Id", "N");

            // Upload sample data for scan
            uploadSampleProducts(parallelScanTestTableName, scanItemCount);

            // Scan the table using multiple threads
            parallelScan(parallelScanTestTableName, scanItemLimit, parallelScanThreads);
        }
        catch (AmazonServiceException ase) {
            System.err.println(ase.getMessage());
        }
    }

    private static void parallelScan(String tableName, int itemLimit, int numberOfThreads)
    {
        System.out.println(
            "Scanning " + tableName + " using " + numberOfThreads + " threads " + itemLimit
+ " items at a time");
        ExecutorService executor = Executors.newFixedThreadPool(numberOfThreads);

        // Divide DynamoDB table into logical segments
        // Create one task for scanning each segment
        // Each thread will be scanning one segment
        int totalSegments = numberOfThreads;
        for (int segment = 0; segment < totalSegments; segment++) {
            // Runnable task that will only scan one segment
            ScanSegmentTask task = new ScanSegmentTask(tableName, itemLimit, totalSegments,
segment);

            // Execute the task
            executor.execute(task);
        }

        shutDownExecutorService(executor);
    }
}
```

```
// Runnable task for scanning a single segment of a DynamoDB table
private static class ScanSegmentTask implements Runnable {

    // DynamoDB table to scan
    private String tableName;

    // number of items each scan request should return
    private int itemLimit;

    // Total number of segments
    // Equals to total number of threads scanning the table in parallel
    private int totalSegments;

    // Segment that will be scanned with by this task
    private int segment;

    public ScanSegmentTask(String tableName, int itemLimit, int totalSegments, int
segment) {
        this.tableName = tableName;
        this.itemLimit = itemLimit;
        this.totalSegments = totalSegments;
        this.segment = segment;
    }

    @Override
    public void run() {
        System.out.println("Scanning " + tableName + " segment " + segment + " out of "
+ totalSegments
        + " segments " + itemLimit + " items at a time...");
        int totalScannedItemCount = 0;

        Table table = dynamoDB.getTable(tableName);

        try {
            ScanSpec spec = new
ScanSpec().withMaxResultSize(itemLimit).withTotalSegments(totalSegments)
                .withSegment(segment);

            ItemCollection<ScanOutcome> items = table.scan(spec);
            Iterator<Item> iterator = items.iterator();

            Item currentItem = null;
            while (iterator.hasNext()) {
                totalScannedItemCount++;
                currentItem = iterator.next();
                System.out.println(currentItem.toString());
            }
        }
        catch (Exception e) {
            System.err.println(e.getMessage());
        }
        finally {
            System.out.println("Scanned " + totalScannedItemCount + " items from
segment " + segment + " out of "
            + totalSegments + " of " + tableName);
        }
    }
}

private static void uploadSampleProducts(String tableName, int itemCount) {
    System.out.println("Adding " + itemCount + " sample items to " + tableName);
    for (int productIndex = 0; productIndex < itemCount; productIndex++) {
        uploadProduct(tableName, productIndex);
    }
}
```

```
private static void uploadProduct(String tableName, int productIndex) {  
  
    Table table = dynamoDB.getTable(tableName);  
  
    try {  
        System.out.println("Processing record #" + productIndex);  
  
        Item item = new Item().withPrimaryKey("Id", productIndex)  
            .withString("Title", "Book " + productIndex + " Title").withString("ISBN",  
"111-1111111111")  
            .withStringSet("Authors", new  
HashSet<String>(Arrays.asList("Author1"))).withNumber("Price", 2)  
            .withString("Dimensions", "8.5 x 11.0 x 0.5").withNumber("PageCount", 500)  
            .withBoolean("InPublication", true).withString("ProductCategory", "Book");  
        table.putItem(item);  
  
    }  
    catch (Exception e) {  
        System.err.println("Failed to create item " + productIndex + " in " +  
tableName);  
        System.err.println(e.getMessage());  
    }  
}  
  
private static void deleteTable(String tableName) {  
    try {  
  
        Table table = dynamoDB.getTable(tableName);  
        table.delete();  
        System.out.println("Waiting for " + tableName + " to be deleted...this may take  
a while...");  
        table.waitForDelete();  
  
    }  
    catch (Exception e) {  
        System.err.println("Failed to delete table " + tableName);  
        e.printStackTrace(System.err);  
    }  
}  
  
private static void createTable(String tableName, long readCapacityUnits, long  
writeCapacityUnits,  
        String partitionKeyName, String partitionKeyType) {  
  
    createTable(tableName, readCapacityUnits, writeCapacityUnits, partitionKeyName,  
partitionKeyType, null, null);  
}  
  
private static void createTable(String tableName, long readCapacityUnits, long  
writeCapacityUnits,  
        String partitionKeyName, String partitionKeyType, String sortKeyName, String  
sortKeyType) {  
  
    try {  
        System.out.println("Creating table " + tableName);  
  
        List<KeySchemaElement> keySchema = new ArrayList<KeySchemaElement>();  
        keySchema.add(new  
KeySchemaElement().withAttributeName(partitionKeyName).withKeyType(KeyType.HASH)); //  
Partition  
  
        // key  
  
        List<AttributeDefinition> attributeDefinitions = new  
ArrayList<AttributeDefinition>();  
}
```

```
        attributeDefinitions
            .add(new
AttributeDefinition().withAttributeName(partitionKeyName).withAttributeType(partitionKeyType));

        if (sortKeyName != null) {
            keySchema.add(new
KeySchemaElement().withAttributeName(sortKeyName).withKeyType(KeyType.RANGE)); // Sort

            // key
            attributeDefinitions
                .add(new
AttributeDefinition().withAttributeName(sortKeyName).withAttributeType(sortKeyType));
        }

        Table table = dynamoDB.createTable(tableName, keySchema, attributeDefinitions,
new ProvisionedThroughput()

.withReadCapacityUnits(readCapacityUnits).withWriteCapacityUnits(writeCapacityUnits));
        System.out.println("Waiting for " + tableName + " to be created...this may take
a while..."); 
        table.waitForActive();

    }
    catch (Exception e) {
        System.err.println("Failed to create table " + tableName);
        e.printStackTrace(System.err);
    }
}

private static void shutDownExecutorService(ExecutorService executor) {
    executor.shutdown();
    try {
        if (!executor.awaitTermination(10, TimeUnit.SECONDS)) {
            executor.shutdownNow();
        }
    }
    catch (InterruptedException e) {
        executor.shutdownNow();

        // Preserve interrupt status
        Thread.currentThread().interrupt();
    }
}
}
```

Verificação de tabelas e índices: .NET

O `Scan` lê todos os itens em uma tabela ou índice no Amazon DynamoDB.

Veja a seguir as etapas para realizar uma verificação de uma tabela usando a AWS SDK for .NET API de baixo nível:

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Crie uma instância da classe `ScanRequest` e forneça parâmetros de operação de verificação.
O único parâmetro obrigatório é o nome da tabela.
3. Execute a `Scan` fornecendo o método `QueryRequest` que você criou na etapa anterior.

A seguinte tabela `Reply` armazena respostas para threads de fóruns.

Example

```
>Reply ( <emphasis role="underline">Id</emphasis>, <emphasis role="underline">ReplyDateTime</emphasis>, Message, PostedBy )
```

A tabela mantém todas as respostas para vários threads de fóruns. Portanto, a chave primária é composta de `Id` (chave de partição) e `ReplyDateTime` (chave de classificação). O exemplo de código C# a seguir verifica toda a tabela. A instância de `ScanRequest` especifica o nome da tabela a ser verificada.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();

var request = new ScanRequest
{
    TableName = "Reply",
};

var response = client.Scan(request);
var result = response.ScanResult;

foreach (Dictionary<string, AttributeValue> item in response.ScanResult.Items)
{
    // Process the result.
    PrintItem(item);
}
```

Especificação de parâmetros opcionais

O método `Scan` aceita vários parâmetros opcionais. Por exemplo, você pode opcionalmente usar um filtro de verificação para filtrar o resultado da verificação. Em um filtro de verificação, você pode especificar uma condição e um nome de atributo no qual deseja que essa condição seja avaliada. Para obter mais informações, consulte [Scan](#).

O seguinte código C# verifica o `ProductCatalog` para localizar itens cujo preço é menor que 0. O exemplo especifica os parâmetros opcionais a seguir:

- Um parâmetro `FilterExpression` para recuperar apenas os itens com preço inferior a 0 (condição de erro).
- Um parâmetro `ProjectionExpression` para especificar os atributos para recuperar itens nos resultados da consulta.

O exemplo de código C# a seguir verifica o `ProductCatalog` para localizar todos os itens cujo preço é menor que 0.

Example

```
var forumScanRequest = new ScanRequest
{
    TableName = "ProductCatalog",
    // Optional parameters.
    ExpressionAttributeValues = new Dictionary<string,AttributeValue> {
        {":val", new AttributeValue { N = "0" }}
    },
    FilterExpression = "Price < :val",
    ProjectionExpression = "Id"
};
```

Como opção, você também pode limitar o tamanho da página ou o número de itens por página, usando o opcional `Limit` parâmetro. Cada vez que você executar o comando `Scan`, você obtém uma página de resultados que possui o número de itens especificado. Para obter a próxima página, execute `Scan` novamente, fornecendo o valor de chave primária do último item da página anterior, para que o `Scan` método pode retornar o próximo conjunto de itens. Você fornece essas informações na solicitação, definindo a propriedade `ExclusiveStartKey`. Inicialmente, essa propriedade pode ser nula. Para recuperar páginas subsequentes, você deve atualizar esse valor de propriedade para a chave primária do último item da página anterior.

O exemplo de código C# a seguir verifica o `ProductCatalog` Tabela do. Na solicitação, ele especifica os parâmetros opcionais `Limit` e `ExclusiveStartKey`. O loop do/while continua a verificar uma página por vez até que `LastEvaluatedKey` retorne um valor nulo.

Example

```
Dictionary<string, AttributeValue> lastKeyEvaluated = null;
do
{
    var request = new ScanRequest
    {
        TableName = "ProductCatalog",
        Limit = 10,
        ExclusiveStartKey = lastKeyEvaluated
    };

    var response = client.Scan(request);

    foreach (Dictionary<string, AttributeValue> item
        in response.Items)
    {
        PrintItem(item);
    }
    lastKeyEvaluated = response.LastEvaluatedKey;
} while (lastKeyEvaluated != null && lastKeyEvaluated.Count != 0);
```

Exemplo – verificação usando o .NET

O código C # a seguir fornece um exemplo de trabalho que verifica o `ProductCatalog` Para localizar itens cujo preço é menor que 0.

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
using System;
using System.Collections.Generic;
using Amazon.DynamoDBv2;
```

```
using Amazon.DynamoDBv2.Model;
using Amazon.Runtime;

namespace com.amazonaws.codesamples
{
    class LowLevelScan
    {
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();

        static void Main(string[] args)
        {
            try
            {
                FindProductsForPriceLessThanZero();

                Console.WriteLine("Example complete. To continue, press Enter");
                Console.ReadLine();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                Console.WriteLine("To continue, press Enter");
                Console.ReadLine();
            }
        }

        private static void FindProductsForPriceLessThanZero()
        {
            Dictionary<string, AttributeValue> lastKeyEvaluated = null;
            do
            {
                var request = new ScanRequest
                {
                    TableName = "ProductCatalog",
                    Limit = 2,
                    ExclusiveStartKey = lastKeyEvaluated,
                    ExpressionAttributeValues = new Dictionary<string, AttributeValue> {
                        {"<:val", new AttributeValue {
                            N = "0"
                        }}
                    },
                    FilterExpression = "Price < :val",
                    ProjectionExpression = "Id, Title, Price"
                };

                var response = client.Scan(request);

                foreach (Dictionary<string, AttributeValue> item
                         in response.Items)
                {
                    Console.WriteLine("\nScanThreadTableUsePaging - printing.....");
                    PrintItem(item);
                }
                lastKeyEvaluated = response.LastEvaluatedKey;
            } while (lastKeyEvaluated != null && lastKeyEvaluated.Count != 0);

            Console.WriteLine("To continue, press Enter");
            Console.ReadLine();
        }

        private static void PrintItem(
            Dictionary<string, AttributeValue> attributeList)
        {
            foreach (KeyValuePair<string, AttributeValue> kvp in attributeList)
            {
```

```
        string attributeName = kvp.Key;
        AttributeValue value = kvp.Value;

        Console.WriteLine(
            attributeName + " " +
            (value.S == null ? "" : "S=[" + value.S + "]") +
            (value.N == null ? "" : "N=[" + value.N + "]") +
            (value.SS == null ? "" : "SS=[" + string.Join(",", value.SS.ToArray()) +
            "])" +
            (value.NS == null ? "" : "NS=[" + string.Join(",", value.NS.ToArray()) +
            "])");
    }
    Console.WriteLine("*****");
}
}
```

Exemplo – verificação paralela usando o .NET

O exemplo de código C# a seguir demonstra uma verificação paralela. O programa exclui e depois recria oProductCataloge, em seguida, carrega a tabela com dados. Quando a carga de dados for concluída, o programa criará vários threads e emitirá solicitações Scan paralelas. Por fim, o programa imprime um resumo das estatísticas de tempo de execução.

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
using System;
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;
using Amazon.Runtime;

namespace com.amazonaws.codesamples
{
    class LowLevelParallelScan
    {
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();
        private static string tableName = "ProductCatalog";
        private static int exampleItemCount = 100;
        private static int scanItemLimit = 10;
        private static int totalSegments = 5;

        static void Main(string[] args)
        {
```

```
try
{
    DeleteExampleTable();
    CreateExampleTable();
    UploadExampleData();
    ParallelScanExampleTable();
}
catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message); }
catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
catch (Exception e) { Console.WriteLine(e.Message); }

Console.WriteLine("To continue, press Enter");
Console.ReadLine();
}

private static void ParallelScanExampleTable()
{
    Console.WriteLine("\n*** Creating {0} Parallel Scan Tasks to scan {1}",
totalSegments, tableName);
    Task[] tasks = new Task[totalSegments];
    for (int segment = 0; segment < totalSegments; segment++)
    {
        int tmpSegment = segment;
        Task task = Task.Factory.StartNew(() =>
        {
            ScanSegment(totalSegments, tmpSegment);
        });

        tasks[segment] = task;
    }

    Console.WriteLine("All scan tasks are created, waiting for them to complete.");
    Task.WaitAll(tasks);

    Console.WriteLine("All scan tasks are completed.");
}

private static void ScanSegment(int totalSegments, int segment)
{
    Console.WriteLine("**** Starting to Scan Segment {0} of {1} out of {2} total
segments ***", segment, tableName, totalSegments);
    Dictionary<string, AttributeValue> lastEvaluatedKey = null;
    int totalScannedItemCount = 0;
    int totalScanRequestCount = 0;
    do
    {
        var request = new ScanRequest
        {
            TableName = tableName,
            Limit = scanItemLimit,
            ExclusiveStartKey = lastEvaluatedKey,
            Segment = segment,
            TotalSegments = totalSegments
        };

        var response = client.Scan(request);
        lastEvaluatedKey = response.LastEvaluatedKey;
        totalScanRequestCount++;
        totalScannedItemCount += response.ScannedCount;
        foreach (var item in response.Items)
        {
            Console.WriteLine("Segment: {0}, Scanned Item with Title: {1}",
segment, item["Title"].S);
        }
    } while (lastEvaluatedKey.Count != 0);
}
```

```
Console.WriteLine("**** Completed Scan Segment {0} of {1}.\nTotalScanRequestCount: {2}, TotalScannedItemCount: {3} ***", segment, tableName,
totalScanRequestCount, totalScannedItemCount);
}

private static void UploadExampleData()
{
    Console.WriteLine("\n*** Uploading {0} Example Items to {1} Table***",
exampleItemCount, tableName);
    Console.Write("Uploading Items: ");
    for (int itemIndex = 0; itemIndex < exampleItemCount; itemIndex++)
    {
        Console.Write("{0}, ", itemIndex);
        CreateItem(itemIndex.ToString());
    }
    Console.WriteLine();
}

private static void CreateItem(string itemIndex)
{
    var request = new PutItemRequest
    {
        TableName = tableName,
        Item = new Dictionary<string, AttributeValue>()
    };
    {
        "Id", new AttributeValue {
            N = itemIndex
        },
        "Title", new AttributeValue {
            S = "Book " + itemIndex + " Title"
        },
        "ISBN", new AttributeValue {
            S = "11-11-11-11"
        },
        "Authors", new AttributeValue {
            SS = new List<string>{"Author1", "Author2" }
        },
        "Price", new AttributeValue {
            N = "20.00"
        },
        "Dimensions", new AttributeValue {
            S = "8.5x11.0x.75"
        },
        "InPublication", new AttributeValue {
            BOOL = false
        }
    };
    client.PutItem(request);
}

private static void CreateExampleTable()
{
    Console.WriteLine("\n*** Creating {0} Table ***", tableName);
    var request = new CreateTableRequest
    {
        AttributeDefinitions = new List<AttributeDefinition>()
    };
    new AttributeDefinition
    {
        AttributeName = "Id",
        AttributeType = "N"
    },
    KeySchema = new List<KeySchemaElement>
    {
```

```
        new KeySchemaElement
    {
        AttributeName = "Id",
        KeyType = "HASH" //Partition key
    }
},
ProvisionedThroughput = new ProvisionedThroughput
{
    ReadCapacityUnits = 5,
    WriteCapacityUnits = 6
},
TableName = tableName
};

var response = client.CreateTable(request);

var result = response;
var tableDescription = result.TableDescription;
Console.WriteLine("{1}: {0} \t ReadsPerSec: {2} \t WritesPerSec: {3}",
    tableDescription.TableStatus,
    tableDescription.TableName,
    tableDescription.ProvisionedThroughput.ReadCapacityUnits,
    tableDescription.ProvisionedThroughput.WriteCapacityUnits);

string status = tableDescription.TableStatus;
Console.WriteLine(tableName + " - " + status);

WaitUntilTableReady(tableName);
}

private static void DeleteExampleTable()
{
    try
    {
        Console.WriteLine("\n*** Deleting {0} Table ***", tableName);
        var request = new DeleteTableRequest
        {
            TableName = tableName
        };

        var response = client.DeleteTable(request);
        var result = response;
        Console.WriteLine("{0} is being deleted...", tableName);
        WaitUntilTableDeleted(tableName);
    }
    catch (ResourceNotFoundException)
    {
        Console.WriteLine("{0} Table delete failed: Table does not exist",
tableName);
    }
}

private static void WaitUntilTableReady(string tableName)
{
    string status = null;
    // Let us wait until table is created. Call DescribeTable.
    do
    {
        System.Threading.Thread.Sleep(5000); // Wait 5 seconds.
        try
        {
            var res = client.DescribeTable(new DescribeTableRequest
            {
                TableName = tableName
            });
        }
    }
}
```

```
        Console.WriteLine("Table name: {0}, status: {1}",
                           res.Table.TableName,
                           res.Table.TableStatus);
        status = res.Table.TableStatus;
    }
    catch (ResourceNotFoundException)
    {
        // DescribeTable is eventually consistent. So you might
        // get resource not found. So we handle the potential exception.
    }
} while (status != "ACTIVE");
}

private static void WaitUntilTableDeleted(string tableName)
{
    string status = null;
    // Let us wait until table is deleted. Call DescribeTable.
    do
    {
        System.Threading.Thread.Sleep(5000); // Wait 5 seconds.
        try
        {
            var res = client.DescribeTable(new DescribeTableRequest
            {
                TableName = tableName
            });

            Console.WriteLine("Table name: {0}, status: {1}",
                               res.Table.TableName,
                               res.Table.TableStatus);
            status = res.Table.TableStatus;
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine("Table name: {0} is not found. It is deleted",
tableName);
            return;
        }
    } while (status == "DELETING");
}
}
```

PartiQL - Uma linguagem de consulta compatível com SQL para o Amazon DynamoDB

O Amazon DynamoDB é compatível com [PartiQL](#), um idioma de consulta compatível com SQL, para selecionar, inserir, atualizar e excluir dados no Amazon DynamoDB. Usando o PartiQL, você pode interagir facilmente com tabelas do DynamoDB e executar consultas ad hoc usando o AWS Management Console, NoSQL Workbench, AWS Command Line Interfacee as APIs do DynamoDB para o PartiQL.

As operações do PartiQL fornecem a mesma disponibilidade, latência e desempenho que as outras operações de plano de dados do DynamoDB.

As seções a seguir descrevem a implementação do PartiQL do DynamoDB.

Tópicos

- [O que é o PartiQL? \(p. 530\)](#)
- [PartiQL no Amazon DynamoDB \(p. 530\)](#)

- Conceitos básicos do PartiQL para DynamoDB (p. 530)
- Tipos de dados do PartiQL para DynamoDB (p. 536)
- Instruções do PartiQL para DynamoDB (p. 538)
- Usar Funções do PartiQL com o Amazon DynamoDB (p. 545)
- Operadores aritméticos, comparativos e lógicos do PartiQL para DynamoDB (p. 549)
- Executando transações com o PartiQL para DynamoDB (p. 550)
- Executando operações em Batch com o PartiQL para DynamoDB (p. 553)
- Políticas de segurança do IAM com o PartiQL para DynamoDB (p. 556)

O que é o PartiQL?

PartiQL fornece acesso de consulta compatível com SQL em vários armazenamentos de dados contendo dados estruturados, dados semiestruturados e dados aninhados. Ele é amplamente utilizado dentro da Amazon e agora está disponível como parte de muitos AWS, incluindo o DynamoDB.

Para obter a especificação do PartiQL e um tutorial sobre a linguagem de consulta principal, consulte [o Documentação PartiQL](#).

Note

- O Amazon DynamoDB é compatível com uma subconjunto do PartiQL idioma de consulta.
- O Amazon DynamoDB não é compatível com o [Amazon Ion](#) formato de dados ou literais Amazon Ion.

PartiQL no Amazon DynamoDB

Para executar consultas do PartiQL no DynamoDB, você pode usar:

- O console do DynamoDB.
- O NoSQL Workbench
- A AWS Command Line Interface (AWS CLI).
- As APIs do DynamoDB.

Para obter informações sobre como usar esses métodos para acessar o DynamoDB, consulte [Acessando o DynamoDB](#).

Conceitos básicos do PartiQL para DynamoDB

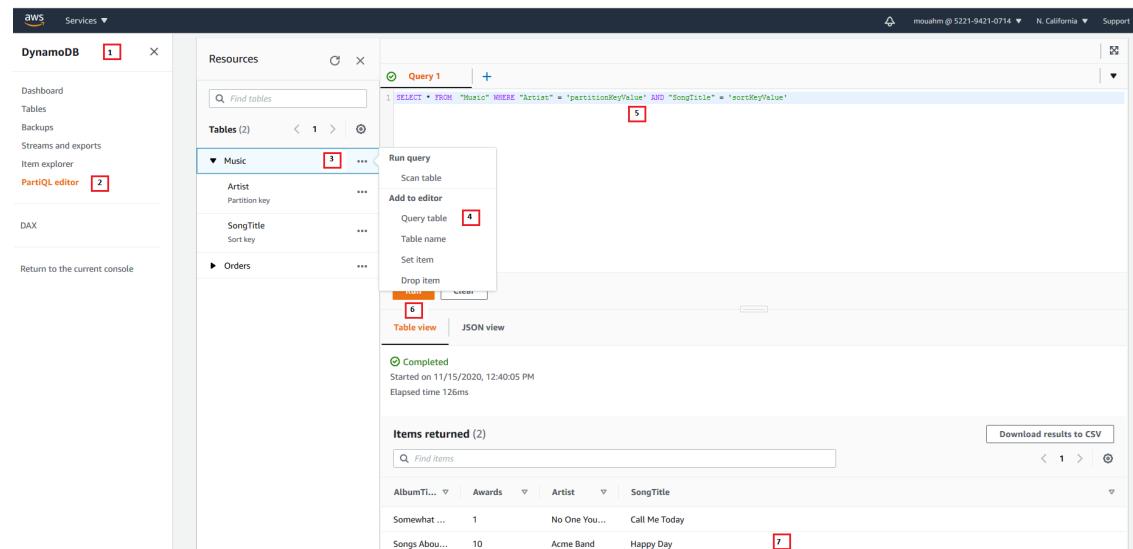
Esta seção descreve como usar o PartiQL para DynamoDB no console do Amazon DynamoDB, o AWS Command Line Interface (AWS CLI) e as APIs do DynamoDB.

Nos exemplos a seguir, a tabela do DynamoDB que é definida no [Conceitos básicos do DynamoDB](#) tutorial é um pré-requisito.

Para obter informações sobre como usar o console do DynamoDB, AWS Command Line Interface ou APIs do DynamoDB para acessar o DynamoDB, consulte [Acessando o DynamoDB](#).

Para [download](#) e usar a [NoSQL Workbench](#) Para criar [PartiQL para DynamoDB](#) Instruções de escolha de operações PartiQL No canto superior direito do NoSQL Workbench for DynamoDB [Criador de operações](#).

Console



Note

O PartiQL para DynamoDB só está disponível no novo console do DynamoDB. Para usar o novo console do DynamoDB, escolhaExperimente a visualização do novo consoleNo painel de navegação, no lado esquerdo do console.

1. Faça login noAWS Management Consolee abra o console do DynamoDB em<https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecioneEditor de PartiQL.
3. Selecione oMúsicatabela do.
4. SelecioneTabela de consulta. Essa ação gera uma consulta que não resultará em uma varredura de tabela completa.
5. SubstituirpartitionKeyValuecom o valor de stringAcme Band. SubstituirsortKeyValuecom o valor de stringHappy Day.
6. Selecione o botão Run (Executar).
7. Você pode visualizar os resultados da consulta escolhendo oExibição de tabelaou oVisualização JSON.

NoSQL Workbench

The screenshot shows the NoSQL Workbench interface with the following elements highlighted:

- 1.** Radio button for "PartiQL statement".
- 2.** The query text area containing the PartiQL SELECT statement: "1 SELECT * 2 FROM Music 3 WHERE Artist=? and SongTitle=?".
- 3.a.** The "Optional request parameters" section.
- 3.b.** The "+ Add new parameter" button.
- 3.c.** The first parameter entry: Attribute type "String" and Attribute value "Acme Band".
- 4.** The "Run" button.
- 5.** The "Generate code" button.
- 6.** The "Clear form" button.
- 7.** The "Hide operation" button.
- 8.** The "Results" tab (selected) and "Generated code" tab.

1. Selecione a instrução PartiQL.
2. Insira o seguinte PartiQL Instrução SELECT

```
SELECT *
FROM Music
WHERE Artist=? and SongTitle=?
```

3. Para especificar um valor para o parâmetro Artist e SongTitle Parâmetros:
 - a. Selecione Parâmetros de solicitação op.
 - b. Selecione Adicionar novos parâmetros.
 - c. Escolha o tipo de atributo String e value Acme Band.
 - d. Repita as etapas b e c e escolha o tipo String e value PartiQL Rocks.
4. Se desejar gerar código, escolha Generate code (Gerar código).

Selecione a linguagem desejada nas guias exibidas. Agora você pode copiar esse código e usá-lo no seu aplicativo.

5. Se desejar que a operação seja executada imediatamente, escolha o Execução do.

AWS CLI

1. Crie um item no Music usando a instrução INSERT PartiQL.

```
aws dynamodb execute-statement --statement "INSERT INTO Music \
    VALUE \
    {'Artist':'Acme Band','SongTitle':'PartiQL Rocks'}"
```

2. Recupere um item da tabela Música usando a instrução SELECT PartiQL.

```
aws dynamodb execute-statement --statement "SELECT * FROM Music \\\\"
```

```
WHERE Artist='Acme Band' AND  
SongTitle='PartiQL Rocks'"
```

3. Atualize um item noMusicusando a instrução UPDATE PartiQL.

```
aws dynamodb execute-statement --statement "UPDATE Music \  
SET AwardsWon=1 \  
SET AwardDetail={'Grammys':[2020,  
2018]} \  
WHERE Artist='Acme Band' AND  
SongTitle='PartiQL Rocks'"
```

Adicione um valor de lista para um item noMusictabela do.

```
aws dynamodb execute-statement --statement "UPDATE Music \  
SET AwardDetail.Grammys  
=list_append(AwardDetail.Grammys,[2016]) \  
WHERE Artist='Acme Band' AND  
SongTitle='PartiQL Rocks'"
```

Remova um valor de lista para um item na caixaMusictabela do.

```
aws dynamodb execute-statement --statement "UPDATE Music \  
REMOVE AwardDetail.Grammys[2] \  
WHERE Artist='Acme Band' AND  
SongTitle='PartiQL Rocks'"
```

Adicione um novo membro do mapa para um item noMusictabela do.

```
aws dynamodb execute-statement --statement "UPDATE Music \  
SET AwardDetail.BillBoard=[2020] \  
WHERE Artist='Acme Band' AND  
SongTitle='PartiQL Rocks'"
```

Adicione um novo atributo de conjunto de strings para um item noMusictabela do.

```
aws dynamodb execute-statement --statement "UPDATE Music \  
SET BandMembers =<<'member1',  
'member2'>> \  
WHERE Artist='Acme Band' AND  
SongTitle='PartiQL Rocks'"
```

Atualize um atributo de conjunto de strings para um item noMusictabela do.

```
aws dynamodb execute-statement --statement "UPDATE Music \  
SET BandMembers =set_add(BandMembers,  
<<'newmember'>>) \  
WHERE Artist='Acme Band' AND  
SongTitle='PartiQL Rocks'"
```

4. Excluir um item da caixa de diálogoMusicusando a instrução DELETE PartiQL.

```
aws dynamodb execute-statement --statement "DELETE FROM Music \  
WHERE Artist='Acme Band' AND SongTitle='PartiQL Rocks'"
```

Java

```
import java.util.ArrayList;
import java.util.List;

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.ConditionalCheckFailedException;
import com.amazonaws.services.dynamodbv2.model.ExecuteStatementRequest;
import com.amazonaws.services.dynamodbv2.model.ExecuteStatementResult;
import com.amazonaws.services.dynamodbv2.model.InternalServerErrorException;
import
com.amazonaws.services.dynamodbv2.model.ItemCollectionSizeLimitExceededException;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughputExceededException;
import com.amazonaws.services.dynamodbv2.model.RequestLimitExceededException;
import com.amazonaws.services.dynamodbv2.model.ResourceNotFoundException;
import com.amazonaws.services.dynamodbv2.model.TransactionConflictException;

public class DynamoDBPartiQLGettingStarted {

    public static void main(String[] args) {
        // Create the DynamoDB Client with the region you want
        AmazonDynamoDB dynamoDB = createDynamoDbClient("us-west-1");

        try {
            // Create ExecuteStatementRequest
            ExecuteStatementRequest executeStatementRequest = new
ExecuteStatementRequest();
            List<AttributeValue> parameters= getPartiQLParameters();

            //Create an item in the Music table using the INSERT PartiQL statement
            processResults(executeStatementRequest(dynamoDB, "INSERT INTO Music value
{'Artist':?, 'SongTitle':?}");
            parameters);

            //Retrieve an item from the Music table using the SELECT PartiQL statement.
            processResults(executeStatementRequest(dynamoDB, "SELECT * FROM Music
where Artist=? and SongTitle=?"));
            parameters);

            //Update an item in the Music table using the UPDATE PartiQL statement.
            processResults(executeStatementRequest(dynamoDB, "UPDATE Music SET
AwardsWon=1 SET AwardDetail={'Grammys':[2020, 2018]} where Artist=? and SongTitle=?",
parameters));

            //Add a list value for an item in the Music table.
            processResults(executeStatementRequest(dynamoDB, "UPDATE Music SET
AwardDetail.Grammys =list_append(AwardDetail.Grammys,[2016]) where Artist=? and
SongTitle=?"));
            parameters);

            //Remove a list value for an item in the Music table.
            processResults(executeStatementRequest(dynamoDB, "UPDATE Music REMOVE
AwardDetail.Grammys[2] where Artist=? and SongTitle=?"));
            parameters);

            //Add a new map member for an item in the Music table.
            processResults(executeStatementRequest(dynamoDB, "UPDATE Music set
AwardDetail.BillBoard=[2020] where Artist=? and SongTitle=?"));
            parameters);

            //Add a new string set attribute for an item in the Music table.
            processResults(executeStatementRequest(dynamoDB, "UPDATE Music SET
BandMembers =<<'member1', 'member2'>> where Artist=? and SongTitle=?"));
            parameters);

            //update a string set attribute for an item in the Music table.
        }
    }
}
```

```
    processResults(executeStatementRequest(dynamoDB, "UPDATE Music SET
BandMembers =set_add(BandMembers, <<'newmember'>>) where Artist=? and SongTitle=?",
parameters));

    //Retrieve an item from the Music table using the SELECT PartiQL statement.
    processResults(executeStatementRequest(dynamoDB, "SELECT * FROM Music
where Artist=? and SongTitle=?", parameters));

    //delete an item from the Music Table
    processResults(executeStatementRequest(dynamoDB, "DELETE FROM Music where
Artist=? and SongTitle=?", parameters));
} catch (Exception e) {
    handleExecuteStatementErrors(e);
}
}

private static AmazonDynamoDB createDynamoDbClient(String region) {
    return AmazonDynamoDBClientBuilder.standard().withRegion(region).build();
}

private static List<AttributeValue> getPartiQLParameters() {
    List<AttributeValue> parameters = new ArrayList<AttributeValue>();
    parameters.add(new AttributeValue("Acme Band"));
    parameters.add(new AttributeValue("PartiQL Rocks"));
    return parameters;
}

private static ExecuteStatementResult executeStatementRequest(AmazonDynamoDB
client, String statement, List<AttributeValue> parameters ) {
    ExecuteStatementRequest request = new ExecuteStatementRequest();
    request.setStatement(statement);
    request.setParameters(parameters);
    return client.executeStatement(request);
}

private static void processResults(ExecuteStatementResult executeStatementResult) {
    System.out.println("ExecuteStatement successful: "+
executeStatementResult.toString());
}

// Handles errors during ExecuteStatement execution. Use recommendations in error
messages below to add error handling specific to
// your application use-case.
private static void handleExecuteStatementErrors(Exception exception) {
    try {
        throw exception;
    } catch (ConditionalCheckFailedException ccfe) {
        System.out.println("Condition check specified in the operation failed,
review and update the condition " +
                           "check before retrying. Error: " +
ccfe.getErrorMessage());
    } catch (TransactionConflictException tce) {
        System.out.println("Operation was rejected because there is an ongoing
transaction for the item, generally " +
                           "safe to retry with exponential back-off. Error:
" + tce.getErrorMessage());
    } catch (ItemCollectionSizeLimitExceededException icslee) {
        System.out.println("An item collection is too large, you're using Local
Secondary Index and exceeded " +
                           "size limit of items per partition key. Consider
using Global Secondary Index instead. Error: " + icslee.getErrorMessage());
    } catch (Exception e) {
        handleCommonErrors(e);
    }
}
```

```

private static void handleCommonErrors(Exception exception) {
    try {
        throw exception;
    } catch (InternalServerErrorException isee) {
        System.out.println("Internal Server Error, generally safe to retry with
exponential back-off. Error: " + isee.getErrorMessage());
    } catch (RequestLimitExceededException rlee) {
        System.out.println("Throughput exceeds the current throughput limit for
your account, increase account level throughput before " +
                           "retrying. Error: " + rlee.getErrorMessage());
    } catch (ProvisionedThroughputExceededException ptee) {
        System.out.println("Request rate is too high. If you're using a custom
retry strategy make sure to retry with exponential back-off. " +
                           "Otherwise consider reducing frequency of
requests or increasing provisioned capacity for your table or secondary index. Error:
" +
                           ptee.getErrorMessage());
    } catch (ResourceNotFoundException rnfe) {
        System.out.println("One of the tables was not found, verify table exists
before retrying. Error: " + rnfe.getErrorMessage());
    } catch (AmazonServiceException ase) {
        System.out.println("An AmazonServiceException occurred, indicates that the
request was correctly transmitted to the DynamoDB " +
                           "service, but for some reason, the service was
not able to process it, and returned an error response instead. Investigate and " +
                           "configure retry strategy. Error type: " +
                           ase.getErrorType() + ". Error message: " + ase.getErrorMessage());
    } catch (AmazonClientException ace) {
        System.out.println("An AmazonClientException occurred, indicates that the
client was unable to get a response from DynamoDB " +
                           "service, or the client was unable to parse the
response from the service. Investigate and configure retry strategy. " +
                           "Error: " + ace.getMessage());
    } catch (Exception e) {
        System.out.println("An exception occurred, investigate and configure retry
strategy. Error: " + e.getMessage());
    }
}
}

```

Tipos de dados do PartiQL para DynamoDB

A tabela a seguir lista os tipos de dados que você pode usar com o PartiQL para DynamoDB.

Tipo de dados do DynamoDB	Representação PartiQL	Observações
Boolean	TRUE FALSE	Não diferencia letras maiúsculas
Binary	N/D	Somente suportado via código.
List	[value1, value2,...]	Não há restrições quanto aos tipos de dados que podem ser armazenados em um tipo de Lista, e os elementos de uma Lista não precisam ser do mesmo tipo.
Map	{'name': value}	Não há restrições quanto aos tipos de dados que podem ser

Tipo de dados do DynamoDB	Representação PartiQL	Observações
		armazenados em um tipo de Mapa, e os elementos de um Mapa não precisam ser do mesmo tipo.
Null	NULL	Não diferencia letras maiúsculas
Number	1, 1.0, 1e0	Números podem ser positivos, negativo ou zero. Os números podem ter uma precisão de até 38 dígitos.
Number Set	<number1, number2><>	Os elementos em um conjunto de números devem ser do tipo Number.
String Set	<<<'string1', 'string2'>>	Os elementos em um conjunto de strings devem ser do tipo String.
String	'Valor da string '	Aspas simples devem ser usadas para especificar valores de String.

Examples

A instrução a seguir demonstra como inserir os seguintes tipos de dados: String, Number, Map, List, Number Set e String Set.

```
INSERT INTO TypesTable value {'primarykey':'1',
'NumberType':1,
'MapType' : {'entryname1': 'value', 'entryname2': 4},
'ListType': [1,'stringval'],
'NumberSetType':<<1,34,32,4.5>>,
'StringSetType':<<'stringval','stringval2'>>
}'
```

A instrução a seguir demonstra como inserir novos elementos no Map, List, Number Set e altere o valor de um Number type.

```
UPDATE TypesTable
SET NumberType=NumberType + 100
SET MapType.NewMapEntry=[2020, 'stringvalue', 2.4]
SET ListType = LIST_APPEND(ListType, [4, <<'string1', 'string2'>>])
SET NumberSetType= SET_ADD(NumberSetType, <<345, 48.4>>)
SET StringSetType = SET_ADD(StringSetType, <<'stringsetvalue1', 'stringsetvalue2'>>)
WHERE primarykey='1'
```

A instrução a seguir demonstra como remover elementos do Map, List, Number Set e String Set e altere o valor de um Number type.

```
UPDATE TypesTable
SET NumberType=NumberType - 1
REMOVE ListType[1]
REMOVE MapType.NewMapEntry
SET NumberSetType = SET_DELETE( NumberSetType, <<345>>)
SET StringSetType = SET_DELETE( StringSetType, <<'stringsetvalue1'>>)
```

```
WHERE primarykey='1'
```

Para obter mais informações, consulte[Tipos de dados do DynamoDB](#).

Instruções do PartiQL para DynamoDB

O Amazon DynamoDB é compatível com as instruções do PartiQL a seguir.

Note

O DynamoDB não oferece suporte a todas as instruções do PartiQL.

Esta referência fornece sintaxe básica e exemplos de uso de instruções do PartiQL que você executa manualmente usando o AWS CLI ou APIs.

Linguagem de manipulação de dados(DML) é o conjunto de instruções do PartiQL que você usa para gerenciar dados em tabelas do DynamoDB. Use instruções DML para adicionar, modificar ou excluir dados em uma tabela.

As seguintes instruções de linguagem DML e consulta são suportadas:

- [Instruções do PartiQL Select para o DynamoDB \(p. 538\)](#)
- [Instruções de atualização do PartiQL para o DynamoDB \(p. 541\)](#)
- [Instruções de inserção PartiQL para o DynamoDB \(p. 544\)](#)
- [Instruções de exclusão do PartiQL para DynamoDB \(p. 543\)](#)

[Executando transações com o PartiQL para DynamoDB \(p. 550\)](#) e [Executando operações em Batch com o PartiQL para DynamoDB \(p. 553\)](#) também são compatíveis com o PartiQL para DynamoDB.

Instruções do PartiQL Select para o DynamoDB

Usar a `SELECT` para recuperar os dados de uma tabela no Amazon DynamoDB.

Usar o `SELECT` pode resultar em uma verificação completa da tabela se uma condição de igualdade com uma chave de partição não for fornecida na cláusula `WHERE`. Uma operação de varredura examina todos os itens quanto aos valores solicitados e pode usar o throughput provisionado para uma tabela ou índice grande em uma única operação.

Se você quiser evitar a verificação completa da tabela no PartiQL, você pode:

- Autor do seu `SELECT` para não resultar em verificações completas de tabela, certificando-se de que seu [Condição da cláusula W](#) está configurado adequadamente.
- Desative verificações completas de tabela usando a política do IAM especificada em [Exemplo: Permitir instruções Select e negar instruções de verificação de tabela completa no PartiQL for DynamoDB \(p. 559\)](#), no guia do desenvolvedor do DynamoDB.

Para obter mais informações, consulte[Melhores práticas para consulta e verificação de dados](#), no guia do desenvolvedor do DynamoDB.

Tópicos

- [Syntax \(p. 539\)](#)
- [Parameters \(p. 539\)](#)
- [Examples \(p. 540\)](#)

Syntax

```
SELECT expression [, ...]
FROM table[.index]
[ WHERE condition ] [ [ORDER BY key [DESC|ASC] , ...]
```

Parameters

expressão

(Obrigatório) Uma projeção formada a partir do *curinga ou uma lista de projeção de um ou mais nomes de atributo ou caminhos de documento do conjunto de resultados. Uma expressão pode consistir em chamadas para [Usar Funções do PartiQL com o Amazon DynamoDB \(p. 545\)](#) ou campos que são modificados por [Operadores aritméticos, comparativos e lógicos do PartiQL para DynamoDB \(p. 549\)](#).

tabela

(Obrigatório) O nome da tabela a ser consultado.

índice

(Opcional) O nome do índice a ser consultado.

condição

(Opcional) Os critérios de seleção para a consulta.

Important

Para garantir que um SELECT não resultar em uma verificação completa da tabela, a instrução WHERE A condição de cláusula deve especificar uma chave de partição. Use o operador igualdade ou IN.

Por exemplo, se tiver um Order tabela com um OrderID e outros atributos não-chave, incluindo um Address, as instruções a seguir não resultariam em uma varredura completa da tabela:

```
SELECT *
FROM Orders
WHERE OrderID = 100

SELECT *
FROM Orders
WHERE OrderID = 100 and Address='some address'

SELECT *
FROM Orders
WHERE OrderID = 100 or pk = 200

SELECT *
FROM Orders
WHERE OrderID IN [100, 300, 234]
```

Os seguintes exemplos de SELECT, no entanto, resultarão em uma varredura completa da tabela:

```
SELECT *
FROM Orders
WHERE OrderID > 1

SELECT *
```

```
FROM Orders
WHERE Address='some address'

SELECT *
FROM Orders
WHERE OrderID = 100 OR Address='some address'
```

chave

(Opcional) Uma chave de hash ou uma chave de classificação a ser usada para ordenar resultados retornados. A ordem padrão é crescente (ASC) especificar DESC se quiser que os resultados sejam reajustados em ordem decrescente.

Note

Se você omitir o WHERE cláusula, todos os itens da tabela serão recuperados.

Examples

A consulta a seguir retorna um item, se existir um, do `Orders` especificando a chave de partição, `OrderID`, e usando o operador de igualdade.

```
SELECT OrderID, Total
FROM Orders
WHERE OrderID = 1
```

A consulta a seguir retorna todos os itens do `Orders` Tabela que têm uma chave de partição específica, `OrderID`, valores usando o operador OR.

```
SELECT OrderID, Total
FROM Orders
WHERE OrderID = 1 OR OrderID = 2
```

A consulta a seguir retorna todos os itens do `Orders` Tabela que têm uma chave de partição específica, `OrderID`, valores usando o operador IN. Os resultados retornados estão em ordem decrescente, com base no `OrderID` attribute-chave.

```
SELECT OrderID, Total
FROM Orders
WHERE OrderID IN [1, 2, 3] ORDER BY OrderID DESC
```

A consulta a seguir mostra uma varredura de tabela completa que retorna todos os itens do `Orders` que têm uma tabela `Total` maior que 500, onde `Total` é um atributo não-chave.

```
SELECT OrderID, Total
FROM Orders
WHERE Total > 500
```

A consulta a seguir mostra uma varredura de tabela completa que retorna todos os itens do `Orders` Tabela em uma tabela específica do `Total` intervalo de ordem, usando o operador IN e um atributo não-chave `Total`.

```
SELECT OrderID, Total
FROM Orders
WHERE Total IN [500, 600]
```

A consulta a seguir mostra uma varredura de tabela completa que retorna todos os itens da tabela `Orders` em uma tabela específica do `Total` intervalo de ordem, usando o operador `BETWEEN` e um atributo não-chave `Total`.

```
SELECT OrderID, Total
FROM Orders
WHERE Total BETWEEN 500 AND 600
```

A consulta a seguir retorna a primeira data em que um dispositivo de detecção de fogo foi usado para observar especificando a chave de partição `CustomerID`, a chave de classificação `MovieID` na condição da cláusula WHERE e usando caminhos de documento na cláusula SELECT.

```
SELECT Devices.FireStick.DateWatched[0]
FROM WatchList
WHERE CustomerID= 'C1' AND MovieID= 'M1'
```

A consulta a seguir mostra uma varredura de tabela completa que retorna a lista de itens em que um dispositivo de detecção de fogo foi usado pela primeira vez após 12/12/19 usando caminhos de documento na condição cláusula WHERE.

```
SELECT Devices
FROM WatchList
WHERE Devices.FireStick.DateWatched[0] >= '12/12/19'
```

Instruções de atualização do PartiQL para o DynamoDB

Usar a `UPDATE` para modificar o valor de um ou mais atributos em um item em uma tabela do Amazon DynamoDB.

Note

Você só pode atualizar um item de cada vez; não é possível emitir uma única instrução PartiQL do DynamoDB que atualize vários itens. Para obter informações sobre como atualizar vários itens, consulte [Executando transações com o PartiQL para DynamoDB \(p. 550\)](#) ou [Executando operações em Batch com o PartiQL para DynamoDB \(p. 553\)](#).

Tópicos

- [Syntax \(p. 541\)](#)
- [Parameters \(p. 541\)](#)
- [Valor de retorno \(p. 542\)](#)
- [Examples \(p. 542\)](#)

Syntax

```
UPDATE table
[SET | REMOVE] path [= data] [...]
WHERE condition [RETURNING returnvalues]
<i></i><returnvalues> ::= [ALL OLD | MODIFIED OLD | ALL NEW | MODIFIED NEW] *
```

Parameters

tabela

(Obrigatório) A tabela que contém os dados a serem modificados.

path

(Obrigatório) Um nome de atributo ou caminho de documento a ser criado ou modificado.

dados

(Obrigatório) Um valor de atributo ou o resultado de uma operação.

As operações suportadas a serem usadas com SET:

- LIST_APPEND: adiciona um valor a um tipo de lista.
- SET_ADD: adiciona um valor a um conjunto de números ou strings.
- SET_DELETE: remove um valor de um número ou conjunto de strings.

condição

(Obrigatório) Os critérios de seleção para o item a ser modificado. Essa condição deve ser resolvida para um único valor de chave primária.

valores de retorno

(Opcional) Usar `returnvalues` se você quiser obter os atributos de item como eles aparecem antes ou depois de serem atualizados. Os valores válidos são:

- ALL OLD * - Retorna todos os atributos do item, como eles apareciam antes da operação de atualização.
- MODIFIED OLD * - Retorna apenas os atributos atualizados, como eles apareciam antes da operação de atualização.
- ALL NEW * - Retorna todos os atributos do item, conforme aparecem após a operação de atualização.
- MODIFIED NEW * - Retorna apenas os atributos atualizados, como eles aparecem após a operação de atualização.

Valor de retorno

Esta instrução não retorna um valor a menos que `returnvalues` parâmetro é especificado.

Note

Se a cláusula WHERE da instrução UPDATE não for avaliada como verdadeira para nenhum item da tabela do DynamoDB, `ConditionalCheckFailedException` é retornado.

Examples

Atualize um valor de atributo em um item existente. Se o atributo não existir, ele será criado.

A consulta a seguir atualiza um item na seção Music adicionando um atributo do tipo número (`AwardsWon`) e um atributo do tipo map (`AwardDetail`).

```
UPDATE Music
SET AwardsWon=1
SET AwardDetail={'Grammys':[2020, 2018]}
WHERE Artist='Acme Band' AND SongTitle='PartiQL Rocks'
```

A consulta a seguir atualiza um item na seção Music removendo de uma lista `AwardDetail.Grammys`.

```
UPDATE Music
SET AwardDetail.Grammys =list_append(AwardDetail.Grammys,[2016])
WHERE Artist='Acme Band' AND SongTitle='PartiQL Rocks'
```

A consulta a seguir atualiza um item na seção Music removendo de uma lista `AwardDetail.Grammys`.

```
UPDATE Music
REMOVE AwardDetail.Grammys[2]
WHERE Artist='Acme Band' AND SongTitle='PartiQL Rocks'
```

A consulta a seguir atualiza um item na seção `Music` tabela adicionando `BillBoard` para o mapa `AwardDetail`.

```
UPDATE Music
SET AwardDetail.BillBoard=[2020]
WHERE Artist='Acme Band' AND SongTitle='PartiQL Rocks'
```

A consulta a seguir atualiza um item na seção `Music` adicionando o atributo de conjunto de strings `BandMembers`.

```
UPDATE Music
SET BandMembers =<<'member1', 'member2'>>
WHERE Artist='Acme Band' AND SongTitle='PartiQL Rocks'
```

A consulta a seguir atualiza um item na seção `Music` tabela adicionando novo `bandmember` para o conjunto de strings `BandMembers`.

```
UPDATE Music
SET BandMembers =set_add(BandMembers, <<'newbandmember'>>)
WHERE Artist='Acme Band' AND SongTitle='PartiQL Rocks'
```

Instruções de exclusão do PartiQL para DynamoDB

Usar `DELETE` para excluir um item existente da tabela do Amazon DynamoDB.

Note

Você pode excluir apenas um item por vez. Você não pode emitir uma única instrução PartiQL do DynamoDB que exclua vários itens. Para obter informações sobre como excluir vários itens, consulte [Executando transações com o PartiQL para DynamoDB \(p. 550\)](#) ou [Executando operações em Batch com o PartiQL para DynamoDB \(p. 553\)](#).

Tópicos

- [Syntax \(p. 543\)](#)
- [Parameters \(p. 543\)](#)
- [Valor de retorno \(p. 544\)](#)
- [Examples \(p. 544\)](#)

Syntax

```
DELETE FROM table
WHERE condition [RETURNING returnvalues]
<returnvalues> ::= ALL OLD *
```

Parameters

table

(Obrigatório) A tabela do DynamoDB que contém o item a ser excluído.

condição

(Obrigatório) Os critérios de seleção para o item a ser excluído; essa condição deve ser resolvida para um único valor de chave primária.

valores de retorno

(Opcional) Usar `returnvalues`Se desejar obter os atributos do item como eles apareciam antes que eles fossem excluídos. Os valores válidos são:

- ALL OLD *- O conteúdo do item antigo é retornado.

Valor de retorno

Esta instrução não retorna um valor a menos que `returnvalues`parâmetro é especificado.

Note

Se a tabela do DynamoDB não tiver nenhum item com a mesma chave primária que a do item para o qual a DELETE foi emitida, o SUCCESS será retornado com 0 itens excluídos. Se a tabela tiver um item com a mesma chave primária, mas a condição na cláusula WHERE da instrução DELETE for avaliada como false, `ConditionalCheckFailedException` é retornado.

Examples

```
DELETE FROM "Music" WHERE "Artist" = 'Acme Band' AND "SongTitle" = 'PartiQL Rocks'
```

Instruções de inserção PartiQL para o DynamoDB

Usar a `INSERT`Para adicionar um item a uma tabela no Amazon DynamoDB.

Note

Você só pode inserir um item de cada vez; não é possível emitir uma única instrução PartiQL do DynamoDB que insira vários itens. Para obter informações sobre como inserir vários itens, consulte [Executando transações com o PartiQL para DynamoDB \(p. 550\)](#)ou [Executando operações em Batch com o PartiQL para DynamoDB \(p. 553\)](#).

Tópicos

- [Syntax \(p. 544\)](#)
- [Parameters \(p. 544\)](#)
- [Valor de retorno \(p. 545\)](#)
- [Examples \(p. 545\)](#)

Syntax

Insira um único item.

```
INSERT INTO table VALUE item
```

Parameters

tabela

(Obrigatório) A tabela na qual você deseja inserir os dados. A tabela já deve existir.

item

(Obrigatório) Um item válido do DynamoDB representado como um[Tupla PartiQL](#). Você só deve especificar o nome de cada atributo no item. A diferença entre maiúsculas e minúsculas é ignorada.

Os valores de string também são denotados comÚnicoAspas (' . . . ') no PartiQL.

Valor de retorno

Esta instrução não retorna nenhum valor.

Note

Se a tabela do DynamoDB já tiver um item com a mesma chave primária que a chave primária do item que está sendo inserido, `DuplicateItemException` é retornado.

Examples

```
INSERT INTO
Music value {'Artist' : 'Acme Band', 'SongTitle' : 'PartiQL Rocks'}
```

Usar Funções do PartiQL com o Amazon DynamoDB

O PartiQL no Amazon DynamoDB é compatível com as seguintes variantes incorporadas de funções padrão SQL.

Note

Todas as funções SQL que não estiverem nessa lista não são suportadas no momento no DynamoDB.

Funções agregadas

- Uso da função SIZE com o PartiQL para Amazon DynamoDB (p. 548)

Funções condicionais

- Uso da função EXISTS com o PartiQL para DynamoDB (p. 545)
- Usando a função ATTRIBUTE_TYPE com o PartiQL para DynamoDB (p. 547)
- Usando a função BEGINS_WITH com o PartiQL para DynamoDB (p. 546)
- Usando a função CONTAINS com o PartiQL para DynamoDB (p. 547)
- Usando a função MISSING com o PartiQL para DynamoDB (p. 546)

Uso da função EXISTS com o PartiQL para DynamoDB

Você pode usar EXISTS para executar a mesma função que `ConditionCheck` faz no `TransactWriteItems` API DA. A função EXISTS só pode ser usada em transações.

Dado um valor, retorna `TRUE` se o valor for uma coleção não-vazia. Caso contrário, gera `FALSE`.

Note

Esta função só pode ser usada em operações transacionais.

Syntax

```
EXISTS ( statement )
```

Arguments

instrução

(Obrigatório) A instrução SELECT que a função avalia.

Note

A instrução SELECT deve especificar uma chave primária completa e uma outra condição.

Tipo de retorno

bool

Examples

```
EXISTS(  
    SELECT * FROM "Music"  
    WHERE "Artist" = 'Acme Band' AND "SongTitle" = 'PartiQL Rocks')
```

Usando a função BEGINS_WITH com o PartiQL para DynamoDB

Retorno TRUESe o atributo especificado começar com uma determinada substring.

Syntax

```
begins_with(path, value )
```

Arguments

path

(Obrigatório) O nome do atributo ou o caminho do documento a ser usado.

value

(Obrigatório) A string a ser pesquisada.

Tipo de retorno

bool

Examples

```
SELECT * FROM "Orders" WHERE "OrderID"=1 AND begins_with("Address", '7834 24th')
```

Usando a função MISSING com o PartiQL para DynamoDB

Retorno TRUESe o item não contiver o atributo especificado. Somente operadores de igualdade e desigualdade podem ser usados com esta função.

Syntax

```
attributename IS | IS NOT MISSING
```

Arguments

attributeName

(Obrigatório) O nome do atributo a ser procurado.

Tipo de retorno

bool

Examples

```
SELECT * FROM Music WHERE "Awards" is MISSING
```

Usando a função ATTRIBUTE_TYPE com o PartiQL para DynamoDB

Retorno no TRUESe o atributo no caminho especificado for de um tipo de dados específico.

Syntax

```
attribute_type( attributename, type )
```

Arguments

attributeName

(Obrigatório) O nome do atributo a ser usado.

type

(Obrigatório) O tipo de atributo a ser verificado. Para obter uma lista de valores válidos, consulte DynamoDBattribute_type.

Tipo de retorno

bool

Examples

```
SELECT * FROM "Music" WHERE attribute_type("Artist", 'S')
```

Usando a função CONTAINS com o PartiQL para DynamoDB

Retorno no TRUESe o atributo especificado pelo caminho for um dos seguintes:

- Uma String que contém uma substring específica.

- Um Conjunto que contém um elemento específico dentro dele.

Para obter mais informações, consulte o DynamoDB[contains](#) função.

Syntax

```
contains( path, substring )
```

Arguments

path

(Obrigatório) O nome do atributo ou o caminho do documento a ser usado.

substring

(Obrigatório) A substring de atributo ou o membro do conjunto a ser verificado. Para obter mais informações, consulte o DynamoDB[contains](#) função.

Tipo de retorno

bool

Examples

```
SELECT * FROM "Orders" WHERE "OrderID"=1 AND contains("Address", 'Kirkland')
```

Uso da função SIZE com o PartiQL para Amazon DynamoDB

Retorna um número que representa o tamanho de um atributo em bytes. Veja a seguir os tipos de dados válidos para uso com o tamanho. Para obter mais informações, consulte o DynamoDB[size](#) função.

Syntax

```
size( path )
```

Arguments

path

(Obrigatório) O nome do atributo ou o caminho do documento.

Para obter os tipos compatíveis, consulte DynamoDB[size](#) função.

Tipo de retorno

int

Examples

```
SELECT * FROM "Orders" WHERE "OrderID"=1 AND size("Image") >300
```

Operadores aritméticos, comparativos e lógicos do PartiQL para DynamoDB

O PartiQL no Amazon DynamoDB é compatível com o seguinte[Operadores padrão do SQL](#).

Note

Todos os operadores SQL que não estiverem nessa lista não são compatíveis com o DynamoDB.

Operadores aritméticos

Operador	Descrição
+	Adicionar
-	Subtract (Subtrair)

Operadores de comparação

Operador	Descrição
=	igual a
<>	Não igual a
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

Operadores lógicos

Operador	Descrição
AND	TRUE se todas as condições separadas por ANDSão TRUE
BETWEEN	TRUE se o operando estiver dentro do intervalo de comparações
IN	TRUE se o operando for igual a uma de uma lista de expressões
IS	TRUE se o operando for um determinado tipo de dados PartiQL, incluindo NULL ou MISSING
NOT	Reverte o valor de uma determinada expressão booleana

Operador	Descrição
OR	TRUE se qualquer uma das condições separadas por ORSão TRUE

Executando transações com o PartiQL para DynamoDB

Esta seção descreve como usar transações com o PartiQL para DynamoDB.

Para obter mais informações sobre transações do DynamoDB, consulte[Gerenciar fluxos de trabalho complexos com transações do DynamoDB](#).

Note

A transação inteira deve consistir em instruções de leitura ou instruções de gravação; você não pode misturar ambos em uma transação. A função EXISTS é uma exceção e pode ser usada para verificar a condição de atributos específicos do item de uma maneira semelhante aConditionCheckno[TransactWriteItemsAPI DA](#).

Tópicos

- [Syntax \(p. 550\)](#)
- [Parameters \(p. 550\)](#)
- [Valores de retorno \(p. 551\)](#)
- [Examples \(p. 551\)](#)

Syntax

```
[  
  {  
    "Statement": " statement ",  
    "Parameters": [  
      {  
        " parameter type " : " parameter value "  
      }, ...]  
    } , ...  
]
```

Parameters

instrução

(Obrigatório) Uma instrução compatível com o PartiQL for DynamoDB.

Note

A transação inteira deve consistir em instruções de leitura ou instruções de gravação; você não pode misturar ambos em uma transação.

tipo de parâmetro

(Opcional) Um tipo do DynamoDB, se os parâmetros foram usados ao especificar a instrução do PartiQL.

parameter value

(Opcional) Um valor de parâmetro se os parâmetros foram usados ao especificar a instrução PartiQL.

Valores de retorno

Esta instrução não retorna nenhum valor.

Note

Se qualquer uma das operações singleton INSERT, UPDATE ou DELETE retornar um erro, as transações são canceladas com a função `TransactionCanceledException`, e o código de motivo de cancelamento inclui os erros das operações singleton individuais.

Examples

AWS CLI

1. Salve o seguinte json em um arquivo chamado `partiql.json`

```
[  
  {  
    "Statement": "EXISTS(SELECT * FROM Music where Artist='No One You Know' and SongTitle='Call Me Today' and Awards is MISSING)"  
  },  
  {  
    "Statement": "INSERT INTO Music value {'Artist':'?', 'SongTitle':'?'}",  
    "Parameters": [{"S": "Acme Band"}, {"S": "Best Song"}]  
  },  
  {  
    "Statement": "UPDATE Music SET AwardsWon=1 SET AwardDetail={'Grammys': [2020, 2018]} where Artist='Acme Band' and SongTitle='PartiQL Rocks'"  
  }  
]
```

2. Execute o comando a seguir em um prompt de comando.

```
aws dynamodb execute-transaction --transact-statements file://partiql.json
```

Java

```
public class DynamoDBPartiqlTransaction {  
  
    public static void main(String[] args) {  
        // Create the DynamoDB Client with the region you want  
        AmazonDynamoDB dynamoDB = createDynamoDbClient("us-west-2");  
  
        try {  
            // Create ExecuteTransactionRequest  
            ExecuteTransactionRequest executeTransactionRequest =  
createExecuteTransactionRequest();  
            ExecuteTransactionResult executeTransactionResult =  
dynamoDB.executeTransaction(executeTransactionRequest);  
            System.out.println("ExecuteTransaction successful.");  
            // Handle executeTransactionResult  
  
        } catch (Exception e) {  
            handleExecuteTransactionErrors(e);  
        }  
    }  
  
    private static AmazonDynamoDB createDynamoDbClient(String region) {  
        return AmazonDynamoDBClientBuilder.standard().withRegion(region).build();  
    }  
}
```

```
private static ExecuteTransactionRequest createExecuteTransactionRequest() {
    ExecuteTransactionRequest request = new ExecuteTransactionRequest();

    // Create statements
    List<ParameterizedStatement> statements = getPartiQLTransactionStatements();

    request.setTransactStatements(statements);
    return request;
}

private static List<ParameterizedStatement> getPartiQLTransactionStatements() {
    List<ParameterizedStatement> statements = new
ArrayList<ParameterizedStatement>();

    statements.add(new ParameterizedStatement()
        .withStatement("EXISTS(SELECT * FROM Music where
Artist='No One You Know' and SongTitle='Call Me Today' and Awards is MISSING"));

    statements.add(new ParameterizedStatement()
        .withStatement("INSERT INTO Music value
{'Artist':'?', 'SongTitle':'?'}'")
        .withParameters(new AttributeValue("Acme Band"), new
AttributeValue("Best Song")));

    statements.add(new ParameterizedStatement()
        .withStatement("UPDATE Music SET AwardsWon=1 SET
AwardDetail={'Grammys':[2020, 2018]} where Artist='Acme Band' and SongTitle='PartiQL
Rocks'"));

    return statements;
}

// Handles errors during ExecuteTransaction execution. Use recommendations in error
messages below to add error handling specific to
// your application use-case.
private static void handleExecuteTransactionErrors(Exception exception) {
    try {
        throw exception;
    } catch (TransactionCanceledException tce) {
        System.out.println("Transaction Cancelled, implies a client issue, fix
before retrying. Error: " + tce.getErrorMessage());
    } catch (TransactionInProgressException tipe) {
        System.out.println("The transaction with the given request token is already
in progress, consider changing " +
        "retry strategy for this type of error. Error: " +
        tipe.getErrorMessage());
    } catch (IdempotentParameterMismatchException ipme) {
        System.out.println("Request rejected because it was retried with a
different payload but with a request token that was already used, " +
        "change request token for this payload to be accepted. Error: " +
        ipme.getErrorMessage());
    } catch (Exception e) {
        handleCommonErrors(e);
    }
}

private static void handleCommonErrors(Exception exception) {
    try {
        throw exception;
    } catch (InternalServerErrorException isee) {
        System.out.println("Internal Server Error, generally safe to retry with
exponential back-off. Error: " + isee.getErrorMessage());
    } catch (RequestLimitExceededException rlee) {
        System.out.println("Throughput exceeds the current throughput limit for
your account, increase account level throughput before " +
        rlee.getErrorMessage());
    }
}
```

```
        "retrying. Error: " + rlee.getErrorMessage());
    } catch (ProvisionedThroughputExceededException ptee) {
        System.out.println("Request rate is too high. If you're using a custom
retry strategy make sure to retry with exponential back-off. " +
                           "Otherwise consider reducing frequency of requests or increasing
provisioned capacity for your table or secondary index. Error: " +
                           ptee.getErrorMessage());
    } catch (ResourceNotFoundException rnfe) {
        System.out.println("One of the tables was not found, verify table exists
before retrying. Error: " + rnfe.getErrorMessage());
    } catch (AmazonServiceException ase) {
        System.out.println("An AmazonServiceException occurred, indicates that the
request was correctly transmitted to the DynamoDB " +
                           "service, but for some reason, the service was not able to process it,
and returned an error response instead. Investigate and " +
                           "configure retry strategy. Error type: " + ase.getErrorType() + ".
Error message: " + ase.getErrorMessage());
    } catch (AmazonClientException ace) {
        System.out.println("An AmazonClientException occurred, indicates that the
client was unable to get a response from DynamoDB " +
                           "service, or the client was unable to parse the response from the
service. Investigate and configure retry strategy. " +
                           "Error: " + ace.getMessage());
    } catch (Exception e) {
        System.out.println("An exception occurred, investigate and configure retry
strategy. Error: " + e.getMessage());
    }
}
```

Executando operações em Batch com o PartiQL para DynamoDB

Esta seção descreve como usar instruções em lote com o PartiQL para DynamoDB.

Note

O lote inteiro deve consistir em instruções de leitura ou instruções de gravação; você não pode misturar ambos em um lote.

Tópicos

- [Syntax \(p. 553\)](#)
- [Parameters \(p. 554\)](#)
- [Examples \(p. 554\)](#)

Syntax

```
[{
  "Statement": " statement ",
  "Parameters": [
    {
      " parametername " : " parametervalue "
    }, ...
  ], ...
}]
```

Parameters

instrução

(Obrigatório) Uma instrução compatível com o PartiQL for DynamoDB.

Note

O lote inteiro deve consistir em instruções de leitura ou instruções de gravação; você não pode misturar ambos em um lote.

tipo de parâmetro

(Opcional) Um tipo do DynamoDB, se os parâmetros foram usados ao especificar a instrução do PartiQL.

Parametervalue

(Opcional) Um valor de parâmetro se os parâmetros foram usados ao especificar a instrução PartiQL.

Examples

AWS CLI

1. Salve o seguinte json em um arquivo chamado partiql.json

```
[  
  {  
    "Statement": "INSERT INTO Music value {'Artist':'?', 'SongTitle':'?'},  
    "Parameters": [{"S": "Acme Band"}, {"S": "Best Song"}]  
  },  
  {  
    "Statement": "UPDATE Music SET AwardsWon=1 SET AwardDetail={'Grammys':  
    [2020, 2018]} where Artist='Acme Band' and SongTitle='PartiQL Rocks'"  
  }  
]
```

2. Execute o comando a seguir em um prompt de comando.

```
aws dynamodb batch-execute-statement --statements file://partiql.json
```

Java

```
public class DynamoDBPartiqlBatch {  
  
    public static void main(String[] args) {  
        // Create the DynamoDB Client with the region you want  
        AmazonDynamoDB dynamoDB = createDynamoDbClient("us-west-2");  
  
        try {  
            // Create BatchExecuteStatementRequest  
            BatchExecuteStatementRequest batchExecuteStatementRequest =  
            createBatchExecuteStatementRequest();  
            BatchExecuteStatementResult batchExecuteStatementResult =  
            dynamoDB.batchExecuteStatement(batchExecuteStatementRequest);  
            System.out.println("BatchExecuteStatement successful.");  
            // Handle batchExecuteStatementResult  
  
        } catch (Exception e) {  
            handleBatchExecuteStatementErrors(e);  
        }  
    }  
}
```

```
        }

    }

    private static AmazonDynamoDB createDynamoDbClient(String region) {
        return AmazonDynamoDBClientBuilder.standard().withRegion(region).build();
    }

    private static BatchExecuteStatementRequest createBatchExecuteStatementRequest() {
        BatchExecuteStatementRequest request = new BatchExecuteStatementRequest();

        // Create statements
        List<BatchStatementRequest> statements = getPartiQLBatchStatements();

        request.setStatements(statements);
        return request;
    }

    private static List<BatchStatementRequest> getPartiQLBatchStatements() {
        List<BatchStatementRequest> statements = new
        ArrayList<BatchStatementRequest>();

        statements.add(new BatchStatementRequest()
            .withStatement("INSERT INTO Music value {'Artist':'Acme
Band','SongTitle':'PartiQL Rocks'}"));

        statements.add(new BatchStatementRequest()
            .withStatement("UPDATE Music set
AwardDetail.BillBoard=[2020] where Artist='Acme Band' and SongTitle='PartiQL
Rocks'"));

        return statements;
    }

    // Handles errors during BatchExecuteStatement execution. Use recommendations in
    error messages below to add error handling specific to
    // your application use-case.
    private static void handleBatchExecuteStatementErrors(Exception exception) {
        try {
            throw exception;
        } catch (Exception e) {
            // There are no API specific errors to handle for BatchExecuteStatement,
            common DynamoDB API errors are handled below
            handleCommonErrors(e);
        }
    }

    private static void handleCommonErrors(Exception exception) {
        try {
            throw exception;
        } catch (InternalServerErrorException isee) {
            System.out.println("Internal Server Error, generally safe to retry with
            exponential back-off. Error: " + isee.getErrorMessage());
        } catch (RequestLimitExceededException rlee) {
            System.out.println("Throughput exceeds the current throughput limit for
            your account, increase account level throughput before " +
            "retrying. Error: " + rlee.getErrorMessage());
        } catch (ProvisionedThroughputExceededException ptee) {
            System.out.println("Request rate is too high. If you're using a custom
            retry strategy make sure to retry with exponential back-off. " +
            "Otherwise consider reducing frequency of requests or increasing
            provisioned capacity for your table or secondary index. Error: " +
            ptee.getErrorMessage());
        } catch (ResourceNotFoundException rnfe) {
            System.out.println("One of the tables was not found, verify table exists
            before retrying. Error: " + rnfe.getErrorMessage());
        }
    }
}
```

```
        } catch (AmazonServiceException ase) {
            System.out.println("An AmazonServiceException occurred, indicates that the
request was correctly transmitted to the DynamoDB " +
                "service, but for some reason, the service was not able to process it,
and returned an error response instead. Investigate and " +
                "configure retry strategy. Error type: " + ase.getErrorType() + ".
Error message: " + ase.getMessage());
        } catch (AmazonClientException ace) {
            System.out.println("An AmazonClientException occurred, indicates that the
client was unable to get a response from DynamoDB " +
                "service, or the client was unable to parse the response from the
service. Investigate and configure retry strategy. " +
                "Error: " + ace.getMessage());
        } catch (Exception e) {
            System.out.println("An exception occurred, investigate and configure retry
strategy. Error: " + e.getMessage());
        }
    }
}
```

Políticas de segurança do IAM com o PartiQL para DynamoDB

As seguintes permissões são necessárias:

- Para ler itens usando o PartiQL para DynamoDB, você deve ter `dynamodb:PartiQLSelect` na tabela ou índice.
- Para inserir itens usando o PartiQL para DynamoDB, você deve ter `dynamodb:PartiQLInsert` na tabela ou índice.
- Para atualizar itens usando o PartiQL para DynamoDB, você deve ter `dynamodb:PartiQLUpdate` na tabela ou índice.
- Para excluir itens usando o PartiQL para DynamoDB, você deve ter `dynamodb:PartiQLDelete` na tabela ou índice.

Exemplo: Permitir todas as instruções PartiQL para DynamoDB (Selecionar/Inserir/Atualizar/Delete) em uma tabela

A política do IAM a seguir concede permissões para executar todas as instruções do PartiQL para DynamoDB em uma tabela.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:PartiQLInsert",
                "dynamodb:PartiQLUpdate",
                "dynamodb:PartiQLDelete",
                "dynamodb:PartiQLSelect"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/Music"
            ]
        }
    ]
}
```

```
    ]  
}
```

Exemplo: Permitir instruções de seleção do PartiQL para DynamoDB em uma tabela

A política do IAM a seguir concede permissões para executar o `select` em uma tabela específica.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:PartiQLSelect"  
            ],  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:123456789012:table/Music"  
            ]  
        }  
    ]  
}
```

Exemplo: Permitir que o PartiQL para DynamoDB insira instruções em um índice

A política do IAM a seguir concede permissões para executar o `insert` em um índice específico.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:PartiQLInsert"  
            ],  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:123456789012:table/Music/index/index1"  
            ]  
        }  
    ]  
}
```

Exemplo: Permitir instruções transacionais do PartiQL para DynamoDB somente em uma tabela

A política do IAM a seguir concede permissões para executar apenas instruções transacionais em uma tabela específica.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:PartiQLInsert",  
                "dynamodb:PartiQLUpdate",  
                "dynamodb:PartiQLDelete"  
            ]  
        }  
    ]  
}
```

```
        "dynamodb:PartiQLDelete",
        "dynamodb:PartiQLSelect"
    ],
    "Resource": [
        "arn:aws:dynamodb:us-west-2:123456789012:table/Music"
    ],
    "Condition": {
        "StringEquals": {
            "dynamodb:EnclosingOperation": [
                "ExecuteTransaction"
            ]
        }
    }
}
]
```

Exemplo: Permitir leituras e gravações não transacionais do PartiQL para DynamoDB e bloquear leituras transacionais e gravações do PartiQL em uma tabela.

A seguinte política do IAM concede permissões para executar o PartiQL para leituras e gravações não transacionais do DynamoDB enquanto bloqueia leituras e gravações transacionais do PartiQL para DynamoDB.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "dynamodb:PartiQLInsert",
                "dynamodb:PartiQLUpdate",
                "dynamodb:PartiQLDelete",
                "dynamodb:PartiQLSelect"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/Music"
            ],
            "Condition": {
                "StringEquals": {
                    "dynamodb:EnclosingOperation": [
                        "ExecuteTransaction"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:PartiQLInsert",
                "dynamodb:PartiQLUpdate",
                "dynamodb:PartiQLDelete",
                "dynamodb:PartiQLSelect"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/Music"
            ]
        }
    ]
}
```

Exemplo: Permitir instruções Select e negar instruções de verificação de tabela completa no PartiQL for DynamoDB

A política do IAM a seguir concede permissões para executar o selectem uma tabela específica ao bloquear selectInstruções que resultam em uma verificação de tabela completa.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": [  
                "dynamodb:PartiQLSelect"  
            ],  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:123456789012:table/WatchList"  
            ],  
            "Condition": {  
                "Bool": {  
                    "dynamodb:FullTableScan": [  
                        "true"  
                    ]  
                }  
            }  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:PartiQLSelect"  
            ],  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:123456789012:table/WatchList"  
            ]  
        }  
    ]  
}
```

Como melhorar o acesso a dados com índices secundários

Tópicos

- [Como usar índices secundários globais no DynamoDB \(p. 562\)](#)
- [Índices secundários locais \(p. 604\)](#)

O Amazon DynamoDB fornece acesso rápido a itens em uma tabela, especificando valores de chave primária. No entanto, muitos aplicativos podem se beneficiar por terem uma ou mais chaves secundárias (ou alternativas) disponíveis, para permitir acesso eficiente aos dados com atributos que não sejam a chave primária. Para resolver isso, você pode criar um ou mais índices secundários em uma tabela e emitir solicitações de Query ou de Scan nesses índices.

A índice secundário é uma estrutura de dados que contém um subconjunto de atributos de uma tabela, juntamente com uma chave alternativa para dar suporte a query operações. Você pode recuperar dados do índice usando uma Query, de maneira muito semelhante ao uso de uma Query com uma tabela. Uma tabela pode ter vários índices secundários, o que permite que seus aplicativos tenham acesso a muitos padrões de consulta diferentes.

Note

Você também pode `Scan` um índice, da mesma forma como poderia `Scan` uma tabela.

Cada índice secundário está associado a exatamente uma tabela, da qual ele obtém seus dados. Ela é chamada de tabela base para o índice. Ao criar um índice, você define uma chave alternativa para ele (chave de partição e chave de classificação). Você também define os atributos a serem projetados, ou copiados, da tabela base para o índice. O DynamoDB copia esses atributos para o índice, juntamente com os atributos de chave primária da tabela base. Você pode consultar ou verificar o índice como faria com qualquer tabela.

Cada índice secundário é automaticamente mantido pelo DynamoDB. Quando você adiciona, modifica ou exclui itens na tabela base, todos os índices dessa tabela também são atualizados para refletir essas alterações.

O DynamoDB é compatível com dois tipos de índices secundários:

- **índice secundário global**—um índice com uma chave de partição e uma chave de classificação que podem ser diferentes das contidas na tabela base. Um índice secundário global é considerado “global” porque as consultas no índice podem abranger todos os dados em uma tabela base, além de todas as partições. Um índice secundário global é armazenado em seu próprio espaço de partição longe da tabela base e é dimensionado separadamente da tabela base.
- **índice secundário local**—um índice que possui a mesma chave de partição da tabela base, mas uma chave de classificação diferente. Um índice secundário local é “local” no sentido de que cada partição de um índice secundário local tem a determinação de escopo para uma partição de tabela base com o mesmo valor de chave de partição.

Você deve considerar os requisitos do seu aplicativo ao determinar qual tipo de índice usar. A tabela a seguir mostra as principais diferenças entre um índice secundário global e um índice secundário local.

Característica	Índice secundário global	Índice secundário local
Esquema de chaves	A chave primária de um índice secundário global pode ser simples (chave de partição) ou composta (chave de partição e chave de classificação).	A chave primária de um índice secundário local deve ser composta (chave de partição e chave de classificação).
Atributos de chaves	A chave de partição e a chave de classificação (se presente) do índice podem ser qualquer atributo da tabela base do tipo String, Número ou Binário.	A chave de partição do índice é o mesmo atributo que a chave de partição da tabela base. A chave de classificação pode ser qualquer atributo da tabela base do tipo String, Número ou Binário.
Restrições de tamanho por valor de chave de partição	Não há restrições de tamanho para índices secundários globais.	Para cada valor de chave de partição, o tamanho total de todos os itens indexados deve ser de 10 GB ou menos.
Operações de índice online	Os índices secundários globais podem ser criados ao mesmo tempo em que você cria uma tabela. Você também pode adicionar um novo índice	Índices secundários locais são criados ao mesmo tempo que você cria uma tabela. Não é possível adicionar um índice secundário local a uma tabela

Característica	Índice secundário global	Índice secundário local
	secundário global a uma tabela existente ou excluir um índice secundário global existente. Para mais informações, consulte Atualizar índices secundários globais (p. 571) .	existente nem excluir índices secundários locais existentes.
Consultas e partições	Um índice secundário global permite que você consulte a tabela inteira, em todas as partições.	Um índice secundário local permite consultar uma única partição, conforme especificado pelo valor da chave de partição na consulta.
Consistência de leituras	Consultas em índices secundários globais somente oferecem suporte à consistência eventual.	Ao consultar um índice secundário local, você pode escolher a consistência final ou a coerência forte.
Consumo de throughput provisionado	Cada índice secundário global tem suas próprias configurações de throughput provisionado para atividades de leitura e gravação. Consultas ou verificações em um índice secundário global consomem unidades de capacidade do índice, e não da tabela-base. O mesmo vale para atualizações de índice secundário global devido a gravações de tabela.	Consultas ou verificações em um índice secundário local consomem unidades de capacidade de leitura da tabela-base. Quando você grava em uma tabela, seus índices secundários locais também são atualizados. Essas atualizações consomem unidades de capacidade de gravação da tabela base.
Atributos projetados	Com consultas de índice secundário global ou verificações de, você pode solicitar somente os atributos que estão projetados no índice. O DynamoDB não buscará atributos da tabela.	Se você consultar ou verificar um índice secundário local, poderá solicitar atributos que não estão projetados no índice. O DynamoDB buscará automaticamente os atributos da tabela.

Se quiser criar mais de uma tabela com índices secundários, você deverá fazer isso sequencialmente. Por exemplo, você poderia criar a primeira tabela e esperar até que ela entrasse no estado `ACTIVE`. Em seguida, você criaria a seguinte tabela e esperaria até que ela entrasse no estado `ACTIVE` e assim por diante. Se você tentar criar mais de uma tabela ao mesmo tempo com um índice secundário, o DynamoDB retornará uma `LimitExceededException`.

Para cada índice secundário, é necessário especificar o seguinte:

- O tipo de índice a ser criado — um índice secundário global ou um índice secundário local.
- Um nome para o índice. As regras de nomenclatura de índices são iguais às de tabelas, conforme listado em [Cotas de serviço, conta e tabela no Amazon DynamoDB \(p. 1056\)](#). O nome deve ser exclusivo para a tabela base à qual ele está associado, mas você pode usar o mesmo nome para índices que estão associados a diferentes tabelas base.
- O esquema de chaves do índice. Cada atributo no esquema de chaves do índice deve ser um atributo de nível superior do tipo `String`, `Number` ou `Binary`. Outros tipos de dados, incluindo documentos e conjuntos, não são permitidos. Outros requisitos para o esquema de chaves dependem do tipo de índice:

- Para um índice secundário global, a chave de partição pode ser qualquer atributo escalar da tabela base. Uma chave de classificação é opcional e também pode ser qualquer atributo escalar da tabela base.
- Para um índice secundário local, a chave de partição deve igual à chave de partição da tabela base, e a chave de classificação deve ser um atributo da tabela-base não relacionado a chaves.
- Atributos adicionais, se houver, a serem projetados da tabela base no índice. Esses atributos são adicionais aos atributos de chave da tabela, que são automaticamente projetados em cada índice. Você pode projetar atributos de qualquer tipo de dados, incluindo escalares, documentos e conjuntos.
- As configurações de throughput provisionado para o índice, se necessário:
 - Para um índice secundário global, você deve especificar configurações de unidades de capacidade de gravação. Essas configurações de throughput provisionado são independentes das configurações da tabela base.
 - Para um índice secundário local, não é necessário especificar configurações de unidades de capacidade de gravação. Qualquer operação de leitura e gravação em um índice secundário local extrai das configurações de throughput provisionado de sua tabela base.

Para obter a flexibilidade máxima de consultas, é possível criar até 20 índices secundários globais (cota padrão) e até 5 índices secundários globais por tabela.

A cota de índices secundários globais por tabela é cinco para o seguinte AWS Regiões:

- AWS GovCloud (Leste dos EUA)
- AWS GovCloud (Oeste dos EUA)
- Europe (Stockholm)

Para obter uma listagem detalhada de índices secundários em uma tabela, use o `DescribeTable` operação. `DescribeTable` retorna o nome, o tamanho de armazenamento e as contagens de itens de cada índice secundário na tabela. Esses valores não são atualizados em tempo real, mas aproximadamente a cada seis horas.

Você pode acessar os dados em um índice secundário usando a `query` ou `scan` operação. Você deve especificar o nome da tabela base e o nome do índice que deseja usar, os atributos a serem retornados nos resultados e qualquer expressão de condição a ser aplicada. O DynamoDB pode retornar os resultados em ordem crescente ou decrescente.

Quando uma tabela é excluída, todos os índices associados a ela também são excluídos.

Para ver as melhores práticas, consulte [Práticas recomendadas para usar índices secundários no DynamoDB \(p. 993\)](#).

Como usar índices secundários globais no DynamoDB

Alguns aplicativos talvez precisem executar muitos tipos de consultas, usando uma variedade de atributos diferentes como critérios de consulta. Para oferecer suporte a esses requisitos, você pode criar um ou mais índices secundários globais e o `query` em relação a esses índices no Amazon DynamoDB.

Tópicos

- [CENÁRIO Uso de um índice secundário global \(p. 563\)](#)
- [Projeções de atributo \(p. 566\)](#)
- [Lendo dados de um índice secundário global \(p. 568\)](#)
- [Sincronização de dados entre tabelas e índices secundários globais \(p. 568\)](#)
- [Considerações sobre a taxa de transferência provisionada para índices secundários globais \(p. 569\)](#)

- Considerações sobre armazenamento de índices secundários globais (p. 570)
- Atualizar índices secundários globais (p. 571)
- Como trabalhar com índices secundários globais: Java (p. 584)
- Como trabalhar com índices secundários globais: .NET (p. 591)
- Como trabalhar com índices secundários globais: AWS CLI (p. 602)

CENÁRIO Uso de um índice secundário global

Para ilustrar, considere uma tabela chamada `GameScores` que controla os usuários e os placares de um aplicativo de jogo móvel. Cada item em `GameScores` é identificado por uma chave de partição (`UserId`) e por uma chave de classificação (`GameTitle`). O diagrama a seguir mostra como os itens da tabela seriam organizados. (Nem todos os atributos são mostrados.)

GameScores

<code>UserId</code>	<code>GameTitle</code>	<code>TopScore</code>	<code>TopScoreDateTime</code>	<code>Wins</code>	<code>Losses</code>	...
"101"	"Galaxy Invaders"	5842	"2015-09-15:17:24:31"	21	72	...
"101"	"Meteor Blasters"	1000	"2015-10-22:23:18:01"	12	3	...
"101"	"Starship X"	24	"2015-08-31:13:14:21"	4	9	...
"102"	"Alien Adventure"	192	"2015-07-12:11:07:56"	32	192	...
"102"	"Galaxy Invaders"	0	"2015-09-18:07:33:42"	0	5	...
"103"	"Attack Ships"	3	"2015-10-19:01:13:24"	1	8	...
"103"	"Galaxy Invaders"	2317	"2015-09-11:06:53:00"	40	3	...
"103"	"Meteor Blasters"	723	"2015-10-19:01:13:24"	22	12	...
"103"	"Starship X"	42	"2015-07-11:06:53:00"	4	19	...
...

Agora, vamos supor que você quisesse gravar um aplicativo de placar para exibir as pontuações máximas de cada jogo. Uma consulta que especificasse os atributos chave (`UserId` e `GameTitle`) seria muito eficaz. No entanto, se o aplicativo precisasse recuperar dados de `GameScores` com base no `GameTitle` apenas, ele precisaria usar uma operação `Scan`. À medida que mais itens são adicionados à tabela, as verificações de todos os dados se tornam lentas e ineficientes. Isso dificulta responder a questões como as seguintes:

- Qual é a pontuação máxima já registrada no jogo Meteor Blasters?
- Qual usuário tinha a maior pontuação no Galaxy Invaders?
- Qual era a maior proporção de vitórias versus derrotas?

Para acelerar as consultas em atributos que não são de chave, você pode criar um índice secundário global. Um índice secundário global contém uma seleção de atributos da tabela-base, mas eles são organizados por uma chave primária que é diferente daquela da região da tabela. A chave do índice não precisa ter nenhum dos atributos de chaves da tabela. Ela não precisa nem ter o mesmo esquema de chaves que a tabela.

Por exemplo, você poderia criar um índice secundário global chamado `GameTitleIndex`, com uma chave de partição `GameTitle` e uma chave de tipo `TopScore`. Os atributos de chave primária da tabela base são sempre projetados em um índice, portanto, o atributo `UserId` também está presente. O diagrama a seguir mostra qual é a aparência do índice `GameTitleIndex`.

`GameTitleIndex`

<code>GameTitle</code>	<code>TopScore</code>	<code>UserId</code>
“Alien Adventure”	192	“102”
“Attack Ships”	3	“103”
“Galaxy Invaders”	0	“102”
“Galaxy Invaders”	2317	“103”
“Galaxy Invaders”	5842	“101”
“Meteor Blasters”	723	“103”
“Meteor Blasters”	1000	“101”
“Starship X”	24	“101”
“Starship X”	42	“103”
***	***	***

Agora você pode consultar `GameTitleIndex` e obter as pontuações do jogo `Meteor Blasters` com facilidade. Os resultados são ordenados por valores de chave de classificação, `TopScore`. Se você definir o parâmetro `ScanIndexForward` como falso, os resultados serão retornados em ordem decrescente, de modo que a maior pontuação é retornada primeiro.

Cada índice secundário global deve ter uma chave de partição e pode ter uma chave de classificação opcional. O esquema da chave de índice pode ser diferente do esquema da tabela base. Você poderia ter uma tabela com uma chave primária simples (chave de partição) e criar um índice secundário global com uma chave primária composta (chave de partição e chave de classificação), ou vice-versa. Os atributos de chaves do índice podem consistir em quaisquer atributos de nível superior, `String`, `Number` ou `Binary` da tabela base. Outros tipos escalares, tipos de documento e tipos de conjunto não são permitidos.

Você pode projetar outros atributos da tabela-base para o índice, se quisesse. Quando você consulta o índice, o DynamoDB pode recuperar esses atributos projetados com eficiência. No entanto, as consultas de índice secundário global não podem buscar atributos da tabela-base. Por exemplo, se você consultar `GameTitleIndex` conforme mostrado no diagrama anterior, a consulta poderá não acessar nenhum atributo que não seja de chave além de `TopScore` (embora os atributos de chave `GameTitle` e `UserId` sejam projetados automaticamente).

Em uma tabela do DynamoDB, cada valor de chave deve ser exclusivo. No entanto, os valores de chave em um índice secundário global não precisam ser exclusivos. Para ilustrar, suponhamos que um jogo

chamado Comet Quest seja muito difícil. Há vários novos usuários tentando obter uma pontuação acima de zero, mas não conseguem. Veja a seguir alguns dos dados que podem representar isso.

UserId	GameTitle	TopScore
123	Comet Quest	0
201	Comet Quest	0
301	Comet Quest	0

Quando esses dados são adicionados ao `GameScores`, o DynamoDB a propaga para `GameTitleIndex`. Se consultarmos o índice usando Comet Quest para `GameTitle` e 0 para `TopScore`, os seguintes dados serão retornados.

GameTitle	TopScore	User Id
“Comet Quest”	0	“123”
“Comet Quest”	0	“201”
“Comet Quest”	0	“301”

Apenas os itens com os valores de chaves especificados aparecem na resposta. Nesse conjunto de dados, os itens não estão em nenhuma ordem específica.

Um índice secundário global controla apenas os itens de dados em que seus atributos de chave realmente existem. Por exemplo, suponha que você tenha adicionado um novo item à tabela `GameScores`, mas forneceu somente os atributos de chave primária necessários.

User Id	GameTitle
400	Comet Quest

Porque você não especificou o `TopScore` o DynamoDB não propagará esse item para `GameTitleIndex`. Portanto, se você tivesse consultado `GameScores` para obter todos os itens do Comet Quest, obteria os quatro itens a seguir.

User Id	GameTitle	TopScore
“123”	“Comet Quest”	0
“201”	“Comet Quest”	0
“301”	“Comet Quest”	0
“400”	“Comet Quest”	

Uma consulta semelhante em `GameTitleIndex` ainda retornaria três itens, em vez de quatro. Isso acontece porque o item com `TopScore` inexistente não é propagado para o índice.

<i>GameTitle</i>	<i>TopScore</i>	<i>UserId</i>
“Comet Quest”	0	“123”
“Comet Quest”	0	“201”
“Comet Quest”	0	“301”

Projeções de atributo

A Projeção é o conjunto de atributos que é copiado de uma tabela para um índice secundário. A chave de partição e a chave de classificação da tabela são sempre projetadas no índice; você pode projetar outros atributos para suportar os requisitos de consulta do aplicativo. Quando você consulta um índice, o Amazon DynamoDB pode acessar qualquer atributo na projeção como se esses atributos estivessem em uma tabela própria.

Quando você cria um índice secundário, você precisa especificar os atributos que serão projetados no índice. O DynamoDB fornece três opções diferentes para isso:

- **KEYS_ONLY**— Cada item do índice consiste apenas nos valores de chaves de partição e das chaves de classificação da tabela, além dos valores de chaves do índice. O **KEYS_ONLY** resulta no menor índice secundário possível.
- **INCLUDE**— Além dos atributos descritos em **KEYS_ONLY**, o índice secundário incluirá outros atributos não chave que você especificar.
- **TUDO**— o índice secundário inclui todos os atributos da tabela de origem. Como todos os dados da tabela são duplicados no índice, o **TUDO** resulta no maior índice secundário possível.

No diagrama anterior, `GameTitleIndex` tem apenas um atributo projetado: `UserId`. Assim, enquanto um aplicativo pode determinar eficientemente os `UserId`s dos melhores marcadores para cada jogo usando `GameTitle` e `TopScore` em consultas, ele não pode determinar eficientemente a maior proporção de vitórias versus derrotas dos maiores marcadores. Para fazer isso, ele teria que realizar uma consulta adicional na tabela base para buscar os ganhos e perdas de cada um dos maiores artilheiros. A maneira mais eficiente de oferecer suporte a consultas nesses dados seria projetar esses atributos da tabela-base para o índice secundário global, como mostrado neste diagrama.

GameTitleIndex

<i>GameTitle</i>	<i>TopScore</i>	<i>UserId</i>	<i>Wins</i>	<i>Losses</i>
“Alien Adventure”	192	“102”	32	192
“Attack Ships”	3	“103”	1	8
“Galaxy Invaders”	0	“102”	0	5
“Galaxy Invaders”	2317	“103”	40	3
“Galaxy Invaders”	5842	“101”	21	72
“Meteor Blasters”	723	“103”	22	12
“Meteor Blasters”	1000	“101”	12	3
“Starship X”	24	“101”	4	9
“Starship X”	42	“103”	4	19
...

Como os atributos não são de chave *Wins* e *Losses* são projetados no índice, um aplicativo pode determinar a proporção de vitórias versus derrotas de qualquer jogo ou de qualquer combinação de jogo e ID do usuário.

Ao escolher os atributos para projetar em um índice secundário global, você deve considerar a desvantagem entre os custos de throughput provisionado e os custos de armazenamento:

- Se precisar acessar apenas alguns atributos com a latência mais baixa possível, considere projetar apenas os atributos em um índice secundário global. Quanto menor o índice, menores serão os custos de armazenamento e de gravação.
- Se o aplicativo acessar frequentemente alguns atributos que não são chave, considere projetar esses atributos em um índice secundário global. Os custos de armazenamento adicionais do índice secundário global compensarão o custo de realizar verificações de tabela frequentes.
- Se precisar acessar a maioria dos atributos que não são chave com frequência, você poderá projetar esses atributos - inclusive a tabela-base inteira - em um índice secundário global. Isso dá flexibilidade máxima a você. No entanto, o custo do armazenamento aumentará ou até dobrará.
- Se o aplicativo precisar consultar uma tabela com pouca frequência, mas precisar executar muitas gravações ou atualizações nos dados da tabela, considere projetar **KEYS_ONLY**. O índice secundário global seria de tamanho mínimo, mas ainda estaria disponível quando necessário para a atividade de consulta.

Lendo dados de um índice secundário global

Você pode recuperar itens de um índice secundário global usando a `Query` e `Scan` operações. `GetItem` e `GetBatchItemAs` operações não podem ser usadas em um índice secundário global.

Consultar um índice secundário global

Você pode usar o `Query` para acessar um ou mais itens em um índice secundário global. A consulta deve especificar o nome da tabela-base e o nome do índice que você deseja usar, os atributos a serem retornados nos resultados de consulta e quaisquer condições que você desejar aplicar. O DynamoDB pode retornar os resultados em ordem crescente ou decrescente.

Considere os seguintes dados retornados de uma `Query` que solicita dados de jogos para um aplicativo de placar.

```
{  
    "TableName": "GameScores",  
    "IndexName": "GameTitleIndex",  
    "KeyConditionExpression": "GameTitle = :v_title",  
    "ExpressionAttributeValues": {  
        ":v_title": {"S": "Meteor Blasters"}  
    },  
    "ProjectionExpression": "UserId, TopScore",  
    "ScanIndexForward": false  
}
```

Nesta consulta:

- Acessos ao DynamoDB `GameTitleIndex`, usando o `GameTitle` para localizar os itens de índice do Meteor Blasters. Todos os itens do índice com essa chave são armazenados lado a lado para rápida recuperação.
- Nesse jogo, o DynamoDB usa o índice para acessar todos os IDs de usuário e as pontuações máximas.
- Os resultados são retornados, classificados em ordem decrescente, pois o parâmetro `ScanIndexForward` está definido como falso.

Verificar um índice secundário global

Você pode usar o `Scan` para recuperar todos os dados de um índice secundário global. Você deve fornecer o nome da tabela-base e o nome de índice na solicitação. Com um `Scan` o DynamoDB lê todos os dados do índice e os retorna para o aplicativo. Você também pode solicitar que apenas alguns dos dados sejam retornados, e que os dados restantes sejam descartados. Para fazer isso, use o parâmetro `FilterExpression` da operação `Scan`. Para mais informações, consulte [Expressões de filtro de Scan \(p. 509\)](#).

Sincronização de dados entre tabelas e índices secundários globais

O DynamoDB sincroniza automaticamente cada índice secundário global com sua tabela-base. Quando um aplicativo grava ou exclui itens em uma tabela, quaisquer índices secundários globais na tabela são atualizados de forma assíncrona, usando um modelo eventualmente consistente. Os aplicativos nunca gravam diretamente em um índice. No entanto, é importante compreender as implicações de como o DynamoDB mantém esses índices.

Índices secundários globais herdam o modo de capacidade leitura/gravação da tabela base. Para mais informações, consulte [Considerações ao mudar o modo de capacidade de leitura/gravação \(p. 344\)](#).

Ao criar um índice secundário global, você especifica um ou mais atributos de chave de índice e seus tipos de dados. Isso significa que sempre que você grava um item na tabela-base, os tipos de dados desses atributos devem corresponder aos tipos de dados do esquema de chaves do índice. No caso de `GameTitleIndex`, a chave de partição `GameTitle` no índice é definida como um tipo de dados `String`. A chave de classificação `TopScore` no índice é do tipo `Number`. Se tentar adicionar um item à `GameScore` e especifique um tipo de dados diferente para `GameTitle` ou `TopScore`, o DynamoDB retorna um `ValidationException` devido à incompatibilidade do tipo de dados.

Quando você insere ou exclui itens em uma tabela, os índices secundários globais dessa tabela são atualizados de uma forma eventualmente consistente. As alterações na tabela são propagadas para os índices secundários globais em uma fração de segundo, sob condições normais. No entanto, em alguns cenários improváveis de falha, podem ocorrer atrasos de propagação mais longos. Consequentemente, os aplicativos precisam prever e lidar com situações em que uma consulta em um índice secundário global retorna resultados que não estão atualizados.

Se você gravar um item em uma tabela, não será necessário especificar os atributos para qualquer chave de classificação de índice secundário global. Usando `GameTitleIndex` como um exemplo, você não precisa especificar um valor para o atributo `TopScore` para gravar um novo item na tabela `GameScores`. Neste caso, o DynamoDB não grava todos os dados no índice deste item específico.

Os custos das atividades de gravação em uma tabela com muitos índices secundários serão mais altos do que em uma tabela com um número menor de índices. Para mais informações, consulte [Considerações sobre a taxa de transferência provisionada para índices secundários globais \(p. 569\)](#).

Considerações sobre a taxa de transferência provisionada para índices secundários globais

Ao criar um índice secundário global em uma tabela de modo provisionado, você deve especificar unidades de capacidade de leitura e gravação para a carga de trabalho esperada no índice. As configurações de throughput provisionado de um índice secundário global são separadas daquelas de sua tabela-base. A query em um índice secundário global consome unidades de capacidade de leitura do índice, não da tabela-base. Quando você insere ou exclui itens em uma tabela, os índices secundários globais dessa tabela também são atualizados. Essas atualizações de índice consomem unidades de capacidade do índice, não da tabela base.

Por exemplo, se você query um índice secundário global e exceder sua capacidade de leitura provisionada, sua solicitação será limitada. Se você executar atividades de gravação pesadas na tabela, mas um índice secundário global na tabela tiver capacidade de gravação insuficiente, a atividade de gravação na tabela será limitada.

Note

Para evitar o controle de utilização potencial, a capacidade de gravação provisionada para um índice secundário global deve ser igual ou maior que a capacidade de gravação da tabela base, porque as novas atualizações gravarão na tabela base e no índice secundário global.

Para visualizar as configurações da taxa de transferência provisionada de um índice secundário global, use a operação `DescribeTable`. Informações detalhadas sobre todos os índices secundários globais da tabela são retornados.

unidades de capacidade de leitura

Os índices secundários globais oferecem suporte a leituras eventualmente consistentes, cada uma delas consome metade de uma unidade de capacidade de leitura. Isso significa que uma única consulta de índice secundário global pode recuperar até $2 \times 4\text{ KB} = 8\text{ KB}$ por unidade de capacidade de leitura.

Para consultas de índice secundário global, o DynamoDB calcula atividade de leitura provisionada da mesma forma que para consultas em tabelas. A única diferença é que o cálculo é baseado no tamanho das entradas de índice, em vez do tamanho do item na tabela-base. O número de unidades de capacidade

de leitura é a soma de todos os tamanhos de atributos projetados em todos os itens retornados. O resultado é arredondado para o próximo limite de 4 KB. Para obter mais informações sobre como o DynamoDB calcula a utilização da taxa de transferência provisionada, consulte [Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB \(p. 345\)](#).

Tamanho máximo dos resultados retornados por um `Query` operação é de 1 MB. Isso inclui os tamanhos de todos os nomes e valores de atributos de todos os itens retornados.

Por exemplo, considere um índice secundário global em que cada item contém 2.000 bytes de dados. Agora suponha que você `Query` este índice e que o `KeyConditionExpression` corresponde a oito itens. O tamanho total dos itens correspondentes é 2.000 bytes × 8 itens = 16.000 bytes. O resultado é arredondado para o próximo limite de 4 KB. Como as consultas de índice secundário global são eventualmente consistentes, o custo total é $0,5 \times (16\text{ KB}/4\text{ KB})$, ou 2 unidades de capacidade de leitura.

Unidades de capacidade de gravação

Quando um item for adicionado, atualizado ou excluído de uma tabela, e um índice secundário global for afetado por isso, o índice secundário global consome unidades de capacidade de gravação provisionadas para a operação. O custo total da taxa de transferência provisionada para uma gravação consiste na soma das unidades de capacidade de gravação consumidas pela gravação na tabela base e pela atualização dos índices secundários globais. Se uma gravação em uma tabela não exigir uma atualização de índice secundário global, nenhuma capacidade de gravação será consumida do índice.

Para que uma gravação de tabela seja bem-sucedida, as configurações da taxa de transferência provisionada e todos os seus índices secundários globais devem ter capacidade de gravação suficiente para acomodar a gravação. Caso contrário, a gravação na tabela será limitada.

O custo de gravar um item em um índice secundário global depende de vários fatores:

- Se você gravar um novo item na tabela que define um atributo indexado, ou atualizar um item existente para definir um atributo indexado indefinido anteriormente, uma operação de gravação é necessária para inserir o item no índice.
- Se uma atualização na tabela alterar o valor de um atributo de chave indexado (de A para B), duas gravações serão necessárias, uma para excluir o item anterior do índice e outra gravação para inserir o novo item no índice.
- Se um item estava presente no índice, mas uma gravação na tabela fez com que o atributo indexado fosse excluído, uma gravação é necessária para excluir a projeção do item antigo do índice.
- Se um item não estiver presente no índice antes ou depois que o item é atualizado, não haverá custo de gravação adicionais para o índice.
- Se uma atualização na tabela alterar somente o valor dos atributos projetados no esquema de chaves do índice, mas não alterar os valores de qualquer atributo de chave indexado, uma gravação será necessária para atualizar os valores dos atributos projetados no índice.

Todos esses fatores supõem que o tamanho de cada item no índice seja menor ou igual ao tamanho do item 1 KB para calcular unidades de capacidade de gravação. Entradas de índice maiores exigirão unidades adicionais de capacidade de gravação. Você pode minimizar os custos de gravação, considerando quais atributos suas consultas precisarão retornar e projetar apenas esses atributos no índice.

Considerações sobre armazenamento de índices secundários globais

Quando um aplicativo grava um item em uma tabela, o DynamoDB copia automaticamente o subconjunto correto de atributos em todos os índices secundários globais nos quais esses atributos devem aparecer. Suas AWSA conta é cobrada pelo armazenamento do item na tabela-base e de atributos em qualquer índice secundário global dessa tabela.

A quantidade de espaço usada por um item do índice é a soma do seguinte:

- O tamanho em bytes da chave primária da tabela-base (chave de partição e chave de classificação)
- O tamanho em bytes do atributo de chave do índice
- O tamanho em bytes dos atributos projetados (se houver)
- 100 bytes de sobrecarga por item de índice

Para estimar os requisitos de armazenamento de um índice secundário global, você pode estimar o tamanho médio de um item no índice e multiplicar pelo número de itens da tabela-base que têm os atributos de chave do índice secundário global.

Se uma tabela contém um item em que um determinado atributo não é definido, mas é definido como uma chave de partição ou chave de classificação do índice, o DynamoDB não grava quaisquer dados desse item no índice.

Atualizar índices secundários globais

Esta seção descreve como criar, modificar e excluir índices secundários globais no Amazon DynamoDB.

Tópicos

- [Criar uma tabela com índices secundários globais \(p. 571\)](#)
- [Descrever os índices secundários globais em uma tabela \(p. 572\)](#)
- [Adicionando um índice secundário global a uma tabela existente \(p. 572\)](#)
- [Excluindo um índice secundário global \(p. 574\)](#)
- [Modificando um índice secundário global durante a criação \(p. 575\)](#)
- [Detecção e correção de violações de chaves de índice \(p. 575\)](#)

Criar uma tabela com índices secundários globais

Para criar uma tabela com um ou mais índices secundários globais, use a operação `CreateTable` com o parâmetro `GlobalSecondaryIndexes`. Para obter a flexibilidade máxima de consultas, é possível criar até 20 índices secundários globais (cota padrão) por tabela.

Você deve especificar um atributo que não atue como chave de partição do índice. Como opção, você pode especificar outro atributo para a chave de classificação do índice. Não é necessário que nenhum desses atributos de chave seja o mesmo que um atributo de chave na tabela. Por exemplo, na tabela `GameScores` (consulte [Como usar índices secundários globais no DynamoDB \(p. 562\)](#)), nem `TopScore` nem `TopScoreDateTime` são atributos chave. Você poderia criar um índice secundário global com uma chave de partição do tipo `TopScore` e uma chave de tipo `TopScoreDateTime`. Um índice desse tipo pode ser usado para determinar se há uma correlação entre pontuações altas e a hora do dia em que um jogo é jogado.

Cada atributo de chave de índice deve ser um escalar do tipo `String`, `Number` ou `Binary`. (Ele não pode ser um documento ou um conjunto.) Você pode projetar atributos de qualquer tipo de dados em um índice secundário global. Isso inclui escalares, documentos e conjuntos. Para obter uma lista completa de tipos de dados, consulte [Tipos de dados \(p. 14\)](#).

Se estiver usando o modo provisionado, você deve fornecer configurações de `ProvisionedThroughput` para o índice, formadas por `ReadCapacityUnits` e `WriteCapacityUnits`. Essas configurações de throughput provisionados são distintas daquelas na tabela, mas se comportam de forma semelhante. Para mais informações, consulte [Considerações sobre a taxa de transferência provisionada para índices secundários globais \(p. 569\)](#).

Índices secundários globais herdam o modo de capacidade leitura/gravação da tabela base. Para mais informações, consulte [Considerações ao mudar o modo de capacidade de leitura/gravação \(p. 344\)](#).

Descrever os índices secundários globais em uma tabela

Para ver o status de todos os índices secundários globais em uma tabela, use a operação `DescribeTable`. A parte `GlobalSecondaryIndexes` da resposta mostra todos os índices na tabela, juntamente com o status atual de cada (`IndexStatus`).

O `IndexStatus` para um índice secundário global será um dos seguintes:

- `CREATING`— o índice está sendo criado e ainda não está disponível para uso.
- `ACTIVE`— o índice está pronto para uso, e os aplicativos podem executar `Query` operações no índice.
- `UPDATING`— as configurações de taxa de transferência provisionada do índice estão sendo alteradas.
- `DELETING`— o índice está sendo excluído e não pode mais ser usado.

Quando o DynamoDB tiver terminado de construir um índice secundário global, o status do índice mudará de `CREATING` para `ACTIVE`.

Adicionando um índice secundário global a uma tabela existente

Para adicionar um índice secundário global a uma tabela existente, use o comando `UpdateTable` operação com o comando `GlobalSecondaryIndexUpdates` parâmetro. Você deve fornecer o seguinte:

- Um nome de índice. O nome deve ser exclusivo entre todos os índices na tabela.
- O esquema de chave do índice. É necessário especificar um atributo para a chave da partição de índice, mas existe a opção de especificar outro atributo para a chave de classificação de índice. Não é necessário que nenhum desses atributos de chave seja o mesmo que um atributo de chave na tabela. Os tipos de dados de cada atributo de esquema devem ser escalares: `String`, `Number` ou `Binary`.
- Os atributos a serem projetados da tabela para o índice:
 - `KEYS_ONLY`— Cada item do índice consiste apenas nos valores de chaves de partição e das chaves de classificação da tabela, além dos valores de chaves do índice.
 - `INCLUDE`— Além dos atributos descritos em `KEYS_ONLY`, o índice secundário inclui outros atributos não chave que você especificar.
 - `ALL`— o índice inclui todos os atributos da tabela de origem.
- As configurações do throughput provisionado para o índice, formadas por `ReadCapacityUnits` e `WriteCapacityUnits`. Essas configurações de throughput provisionados são distintas daquelas na tabela.

Você só pode criar um índice secundário global por `UpdateTable` operação.

Fases da criação de um índice

Quando você adiciona um novo índice secundário global a uma tabela existente, esta continua a estar disponível enquanto o índice está sendo construído. No entanto, o novo índice apenas estará disponível para operações de consulta quando seu status mudar de `CREATING` para `ACTIVE`.

Nos bastidores, o DynamoDB constrói o índice em duas fases:

Alocação de recursos

O DynamoDB aloca os recursos de computação e armazenamento que são necessários para a construção do índice.

Durante a fase de alocação de recursos, o atributo `IndexStatus` é `CREATING`, enquanto o atributo `Backfilling` é `false`. Use a operação `DescribeTable` para recuperar o status de uma tabela e todos os índices secundários dela.

Enquanto o índice estiver na fase de alocação de recurso, você poderá excluí-lo ou apagar a tabela pai dele. Você também não pode modificar a taxa de transferência provisionada do índice ou da tabela. Não é possível adicionar ou excluir outros índices na tabela. No entanto, você pode modificar a taxa de transferência provisionada desses outros índices.

Aterrramento

Para cada item na tabela, o DynamoDB determina qual conjunto de atributos deve ser gravado no índice com base em sua projeção (`KEYS_ONLY`, `INCLUDE`, ou `ALL`). Em seguida, ele grava esses atributos no índice. Durante a fase de aterrramento, o DynamoDB controla os itens que estão sendo adicionados, excluídos ou atualizados na tabela. Os atributos desses itens também são adicionados, excluídos ou atualizados no índice conforme apropriado.

Durante a fase de preenchimento, o atributo `IndexStatus` é definido como `CREATING`, e o atributo `Backfilling` é verdadeiro. Use a operação `DescribeTable` para recuperar o status de uma tabela e todos os índices secundários dela.

Enquanto o índice está em processo de aterrramento, não é possível excluir a tabela principal. No entanto, ainda é possível excluir o índice ou modificar a taxa de transferência provisionada da tabela e de qualquer um dos seus índices secundários globais.

Note

Durante a fase de aterrramento, algumas gravações de itens infratores podem ter sucesso, enquanto outras serão rejeitadas. Após o aterrramento, todas as gravações de itens que violarem o esquema de chaves do novo índice serão rejeitadas. Recomendamos que você execute a ferramenta Detector de violação após a conclusão da fase de aterrramento, para detectar e resolver qualquer violação de chave que possa ter ocorrido. Para mais informações, consulte [Detecção e correção de violações de chaves de índice \(p. 575\)](#).

Enquanto as fases de alocação de recurso e aterrramento estão em andamento, o índice permanece no estado `CREATING`. Nesse período de tempo, o DynamoDB executa operações de leitura na tabela. Você não é cobrado pelas operações de leitura da tabela base para preencher o índice secundário global. No entanto, você será cobrado pelas operações de gravação para preencher o índice secundário global recém-criado.

Quando a construção do índice estiver concluída, seu status mudará para `ACTIVE`. Você não pode executar `Query` ou `Scan` no índice até que ele esteja no modo `ACTIVE`.

Note

Em alguns casos, o DynamoDB não pode gravar dados da tabela no índice devido a violações de chave de índice. Isso poderá ocorrer se:

- O tipo de dados de um valor de atributo não corresponder ao tipo de dados de um esquema de chaves de índice.
- O tamanho de um atributo excede o tamanho máximo de um atributo de chave de índice.
- Um atributo de chave de índice tem um valor binários ou de string vazio.

Violações de chaves de índice não interferem na criação de índice secundário global. No entanto, quando o índice ficar `ACTIVE`, as chaves infratoras não estarão presentes no índice. O DynamoDB fornece uma ferramenta autônoma para localizar e resolver esses problemas. Para mais informações, consulte [Detecção e correção de violações de chaves de índice \(p. 575\)](#).

Como adicionar um índice secundário global a uma tabela grande

O tempo necessário para a construção de um índice secundário global depende de vários fatores, como os abaixo:

- O tamanho da tabela

- O número de itens na tabela que se qualificam para inclusão no índice
- O número de atributos projetados no índice
- A capacidade de gravação provisionada do índice
- A atividade de gravação na tabela principal durante a criação dos índices

Se você estiver adicionando um índice secundário global a uma tabela muito grande, a conclusão do processo de criação será demorada. Para monitorar o progresso e determinar se o índice tem capacidade de gravação suficiente, consulte as seguintes métricas do Amazon CloudWatch:

- `OnlineIndexPercentageProgress`
- `OnlineIndexConsumedWriteCapacity`
- `OnlineIndexThrottleEvents`

Note

Para obter mais informações sobre métricas do CloudWatch relacionadas ao DynamoDB, consulte [Métricas do DynamoDB \(p. 930\)](#).

Se a configuração do throughput provisionado de gravação no índice for muito baixa, a compilação do índice levará mais tempo para ser concluída. Para reduzir o tempo necessário para construir um novo índice secundário global, você pode aumentar sua capacidade de gravação provisionada temporariamente.

Note

Como regra geral, recomendamos definir a capacidade de gravação provisionada do índice como 1,5 vezes maior que a capacidade de gravação da tabela. Essa é uma boa configuração para muitos casos de uso. No entanto, seus requisitos reais podem ser maiores ou menores.

Enquanto um índice está sendo aterrado, o DynamoDB usa a capacidade do sistema interno para fazer leituras na tabela. Isso é feito para minimizar o impacto da criação do índice e garantir que sua tabela não fique sem capacidade de leitura.

No entanto, é possível que o volume de atividades de gravação recebidas exceda a capacidade de gravação provisionada do índice. Este é um cenário de afunilamento, no qual a criação do índice é mais demorada porque a atividade de gravação no índice é limitada. Durante a criação do índice, é recomendável monitorar as métricas do Amazon CloudWatch para o índice, a fim de determinar se a capacidade de gravação consumida está excedendo sua capacidade provisionada. Em um cenário de afunilamento, é necessário aumentar a capacidade de gravação provisionada no índice para evitar o controle de utilização de gravações durante a fase de aterrramento.

Depois que o índice tiver sido criado, você deverá definir sua capacidade de gravação provisionada para refletir o uso normal do seu aplicativo.

Excluindo um índice secundário global

Se você não precisa mais de um índice secundário global, pode excluí-lo usando a operação `UpdateTable`.

Você pode excluir apenas um índice secundário global por operação `UpdateTable`.

Enquanto o índice secundário global está sendo excluído, não há efeitos sobre atividades de leitura ou de gravação na tabela principal. Enquanto a exclusão está em andamento, você ainda pode modificar a taxa de transferência provisionada em outros índices.

Note

Quando você exclui uma tabela usando a ação `DeleteTable`, todos os índices secundários globais nessa tabela também são excluídos.

Modificando um índice secundário global durante a criação

Enquanto um índice está sendo construído, é possível usar a operação `DescribeTable` para determinar em qual fase ele se encontra. A descrição do índice inclui um atributo booleano, `BackfillingPara` indicar se o DynamoDB está carregando o índice com itens da tabela no momento. Se `Backfilling` for verdadeiro, a fase de alocação de recursos estará concluída, e o índice agora estará sendo preenchido.

Durante o aterrramento, é possível atualizar os parâmetros de throughput provisionado do índice. Você pode optar por fazer isso para acelerar a construção do índice: Você pode aumentar a capacidade de gravação do índice enquanto ele está sendo construído e, em seguida, reduzi-la. Para modificar as configurações de throughput provisionadas do índice, use a operação `UpdateTable`. O status do índice muda para `UPDATING`, e `Backfilling` é true até o índice estar pronto para uso.

Durante a fase de aterrramento, é possível excluir o índice que está sendo criado. Durante essa fase, não é possível adicionar ou excluir outros índices na tabela.

Note

Para índices que foram criados como parte de uma operação `CreateTable`, o atributo `Backfilling` não aparece na saída de `DescribeTable`. Para mais informações, consulte [Fases da criação de um índice \(p. 572\)](#).

Detecção e correção de violações de chaves de índice

Durante a fase de aterrramento da criação de um índice secundário global, o Amazon DynamoDB examina cada item na tabela para determinar se ele está qualificado para inclusão no índice. Alguns itens podem não ser qualificados, pois causariam violações de chave de índice. Nesses casos, os itens permanecem na tabela, mas o índice não tem uma entrada correspondente para eles.

Uma violação de chave de índice ocorre nas seguintes situações:

- Houver uma incompatibilidade de tipo de dados entre um valor de atributo e o tipo de dados do esquema de chaves de índice. Por exemplo, digamos que um dos itens no `GameScore` tinha um `TopScore` do tipo `String`. Se você adicionou um índice secundário global com uma chave de partição `TopScore`, do tipo `Number`, o item da tabela violaria a chave de índice.
- Um valor de atributo da tabela exceder o comprimento máximo para um atributo de chave de índice. O comprimento máximo de uma chave de partição é 2048 bytes e o comprimento máximo de uma chave de classificação é 1024 bytes. Se qualquer um dos valores de atributo correspondente na tabela exceder esses limites, o item da tabela violará a chave de índice.

Note

Se um valor de atributo binário ou de string for definido para um atributo usado como uma chave de índice, o valor do atributo deverá ter um tamanho maior que zero. Caso contrário, o item da tabela violaria a chave de índice.

Esta ferramenta não sinaliza esta violação de chave de índice neste momento.

Se ocorrer uma violação de chave de índice, a fase de aterrramento continuará sem interrupção. No entanto, os itens infratores não estão incluídos no índice. Concluída a fase de aterrramento, todas as gravações em itens que violam o esquema de chaves do novo índice serão rejeitadas.

Para identificar e corrigir os valores de atributos em uma tabela que violam uma chave de índice, use a ferramenta Detector de Violações. Para executar o Detector de violações, crie um arquivo de configuração que especifique o nome de uma tabela a ser verificada, os nomes e os tipos de dados da chave de partição do índice secundário global e a chave de classificação, bem como quais ações deverão ser realizadas se violações de chave de índice forem encontradas. O Detector de violações pode ser executado em um destes dois modos diferentes:

- Modo de detecção— Detecta violações de chave de índice. Use o modo de detecção para informar os itens na tabela que causariam violações de chaves em um índice secundário global. (Você tem a

opção de solicitar que esses itens de tabela infratores sejam excluídos imediatamente quando forem encontrados.) A saída do modo de detecção é gravada em um arquivo, que você pode usar para análise posterior.

- Modo de correção— Corrige violações de chave de índice. No modo de correção, o Detector de Violações lê um arquivo de entrada com o mesmo formato que o arquivo de saída do modo de detecção. O modo de correção lê os registros do arquivo de entrada e, para cada registro, ele exclui ou atualiza os itens correspondentes na tabela. (Observe que, se você optar por atualizar os itens, deverá editar o arquivo de entrada e definir os valores apropriados para essas atualizações.)

Download e execução do detector de violação

O Detector de violações está disponível como um Arquivo Java executável (.jar) e é executado em computadores Windows, macOS ou Linux. Violation Detector requer o Java 1.7 (ou posterior) e o Apache Maven.

- [Baixe o Detector de Violações do GitHub](#)

Siga as instruções na `README.md` para baixar e instalar o Detector de violações usando o Maven.

Para iniciar o Detector de Violações, acesse o diretório no qual você construiu `ViolationDetector.java` e insira o comando a seguir.

```
java -jar ViolationDetector.jar [options]
```

A linha de comando do Detector de Violações da aceita as seguintes opções:

- `-h | --help`— imprime um resumo de uso e opções para o Detector de violações.
- `-p | --configFilePath value`— o nome totalmente qualificado de um arquivo de configuração do Detector de Violações. Para mais informações, consulte [O arquivo de configuração do detector de violação \(p. 576\)](#).
- `-t | --detect value`— detecta violações de chave de índice na tabela e as grava no arquivo de saída do detector de violações. Se o valor desse parâmetro for definido como `keep`, itens com violações de chave não são modificados. Se o valor for definido como `delete`, os itens com violações de chave são excluídos da tabela.
- `-c | --correct value`— lê violações de chave de índice de um arquivo de entrada e toma ações corretivas nos itens da tabela. Se o valor desse parâmetro for definido como `update`, os itens com violações de chave são atualizados com valores novos não infratores. Se o valor for definido como `delete`, os itens com violações de chave são excluídos da tabela.

O arquivo de configuração do detector de violação

Em tempo de execução, a ferramenta Detector de Violações requer um arquivo de configuração da. Os parâmetros nesse arquivo determinam quais recursos do DynamoDB o Detector de violação pode acessar e quanto throughput provisionado ela pode consumir. A tabela a seguir descreve esses parâmetros.

Nome do parâmetro	Descrição	Obrigatório?
<code>awsCredentialsFile</code>	O nome totalmente qualificado de um arquivo que contém suas credenciais da AWS. O arquivo de credenciais deve estar no seguinte formato:	Sim

Nome do parâmetro	Descrição	Obrigatório?
	<pre>accessKey = <i>access_key_id_goes_here</i> secretKey = <i>secret_key_goes_here</i></pre>	
dynamoDBRegion	O AWSRegião em que a tabela reside. Por exemplo: us-west-2.	Sim
tableName	O nome da tabela do DynamoDB a ser verificada.	Sim
gsiHashKeyName	O nome da chave de partição do índice.	Sim
gsiHashKeyType	O tipo de dados da chave de partição do índice —String,Number, ouBinary: S N B	Sim
gsiRangeKeyName	O nome da chave de classificação do índice. Não especifique esse parâmetro se o índice tiver apenas uma chave primária simples (chave de partição).	Não
gsiRangeKeyType	O tipo de dados da chave de classificação do índice —String,Number, ouBinary: S N B Não especifique esse parâmetro se o índice tiver apenas uma chave primária simples (chave de partição).	Não
recordDetails	Se você deseja gravar os detalhes completos das violações de chaves de índice no arquivo de saída. Se definido como true (o padrão), todas as informações sobre os itens infratores serão relatadas. Se definido como false, apenas o número de violações será relatado.	Não

Nome do parâmetro	Descrição	Obrigatório?
<code>recordGsiValueInViolationReport</code>	<p>Se você deseja gravar os valores das chaves de índice infratoras no arquivo de saída. Se definido como <code>true</code>(padrão), os valores das chaves serão relatados. Se definido como <code>false</code>, os valores das chaves não serão relatados.</p>	Não
<code>detectionOutputPath</code>	<p>O caminho completo do arquivo de saída do Detector de violações. Esse parâmetro oferece suporte à gravação em um diretório local ou no Amazon Simple Storage Service (Amazon S3). Veja os exemplos a seguir:</p> <p><code>detectionOutputPath = //local/path/ filename.csv</code></p> <p><code>detectionOutputPath = s3://bucket/filename.csv</code></p> <p>As informações no arquivo de saída aparecem no formato CSV (valores separados por vírgulas). Se você não definir <code>detectionOutputPath</code>, o arquivo de saída é chamado <code>violation_detection.csv</code> e é gravado no seu diretório de trabalho atual.</p>	Não

Nome do parâmetro	Descrição	Obrigatório?
<code>numOfSegments</code>	<p>O número de segmentos de verificação paralela a serem usados quando o Detector de Violações verificar a tabela. O valor padrão é 1, significando que a tabela é verificada de maneira sequencial. Se o valor for 2 ou superior, o Detector de Violações dividirá a tabela no número correspondente de segmentos lógicos e em um número igual de threads de verificação.</p> <p>A configuração máxima para <code>numOfSegments</code> é 4096.</p> <p>Para tabelas maiores, uma verificação paralela é geralmente mais rápida do que uma verificação sequencial. Além disso, se a tabela for grande o suficiente para abranger várias partições, uma verificação paralela distribuirá sua atividade de leitura uniformemente entre essas várias partições.</p> <p>Para obter mais informações sobre varreduras paralelas no DynamoDB, consulte Scan em paralelo (p. 512).</p>	Não
<code>numOfViolations</code>	O limite superior de violações de chaves de índice para gravação no arquivo de saída. Se definido como -1 (o padrão), a tabela inteira será verificada. Se definido como um número inteiro positivo, o Detector de Violações será interrompido quando encontrar esse número de violações.	Não
<code>numOfRecords</code>	O número de itens na tabela a ser verificada. Se definido como -1 (o padrão), a tabela inteira é verificada. Se definido como um número inteiro positivo, o detector de violações será interrompido depois que verificar o mesmo número de itens na tabela.	Não

Nome do parâmetro	Descrição	Obrigatório?
<code>readWriteIOPSPercent</code>	Regula a porcentagem de unidades de capacidade de leitura provisionada que são consumidas durante a verificação de tabela. Os valores válidos variam de 1 para 100. O valor padrão (25) significa que o Detector de Violações não consumirá mais de 25% do throughput de leitura provisionado da tabela.	Não
<code>correctionInputPath</code>	O caminho completo do arquivo de entrada de correção da Violation Detector. Se você executar o Detector de violação no modo de correção, o conteúdo desse arquivo será usado para modificar ou excluir os itens de dados na tabela que violam o índice secundário global. O formato do arquivocorrectionInputPathO arquivo é o mesmo que o dodetectionOutputPathfile. Isso permite que você processe a saída do modo de detecção como uma entrada no modo de correção.	Não

Nome do parâmetro	Descrição	Obrigatório?
<code>correctionOutputPath</code>	<p>O caminho completo do arquivo de saída de correção da Violation Detector. Esse arquivo será criado somente se houver erros de atualização.</p> <p>Esse parâmetro oferece suporte à gravação em um diretório local ou no Amazon S3. Veja os exemplos a seguir:</p> <pre>correctionOutputPath = //local/path/ filename.csv</pre> <pre>correctionOutputPath = s3://bucket/filename.csv</pre> <p>As informações no arquivo de saída aparecem no formato CSV. Se você não definir <code>correctionOutputPath</code>, o arquivo de saída é chamado <code>violation_update_errors.csv</code> e é gravado no seu diretório de trabalho atual.</p>	Não

Detection

Para detectar violações de chave de índice, use o Detector de violação com `--detectA` opção de linha de comando. Para mostrar como essa opção funciona, considere a `ProductCatalog` tabela mostrada em [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#). Veja a seguir uma lista de itens na tabela. Somente a chave primária (`Id`) e o `Price` atributo são exibidos.

ID (Chave primária)	Preço
101	5
102	20
103	200
201	100
202	200
203	300
204	400
205	500

Todos os valores para `Price` são do tipo `Number`. No entanto, como o DynamoDB não tem esquema, é possível adicionar um item com um não numérico `Price`. Por exemplo, suponha que você adicione outro item à `ProductCatalog` tabela do.

ID (Chave primária)	Preço
999	"Hello"

Agora, a tabela tem um total de nove itens.

Agora você adiciona um novo índice secundário global à tabela: `PriceIndex`. A chave primária desse índice é uma chave de partição `Price`, que é do tipo `Number`. Depois que o índice tiver sido construído, ele conterá oito itens, mas o `ProductCatalog` tem nove itens. A razão para esta discrepância é que o valor "Hello" é do tipo `String`, mas `PriceIndex` tem uma chave primária do tipo `Number`. O tipo `String` viola a chave de índice secundário global e, por isso, não está presente no índice.

Para usar o detector de violações nesse cenário, você primeiro cria um arquivo de configuração, como o seguinte.

```
# Properties file for violation detection tool configuration.
# Parameters that are not specified will use default values.

awsCredentialsFile = /home/alice/credentials.txt
dynamoDBRegion = us-west-2
tableName = ProductCatalog
gsiHashKeyName = Price
gsiHashKeyType = N
recordDetails = true
recordGsiValueInViolationRecord = true
detectionOutputPath = ./gsi_violation_check.csv
correctionInputPath = ./gsi_violation_check.csv
numOfSegments = 1
readWriteIOPSPercent = 40
```

Em seguida, execute o Detector de violações como no exemplo a seguir.

```
$ java -jar ViolationDetector.jar --configFilePath config.txt --detect keep

Violation detection started: sequential scan, Table name: ProductCatalog, GSI name:
PriceIndex
Progress: Items scanned in total: 9,    Items scanned by this thread: 9,    Violations
found by this thread: 1, Violations deleted by this thread: 0
Violation detection finished: Records scanned: 9, Violations found: 1, Violations deleted:
0, see results at: ./gsi_violation_check.csv
```

Se o `recordDetails` parâmetro config é definido como `true`, o Detector de Violações grava detalhes de cada violação no arquivo de saída, como no exemplo a seguir.

```
Table Hash Key,GSI Hash Key Value,GSI Hash Key Violation Type,GSI Hash Key Violation
Description,GSI Hash Key Update Value(FOR USER),Delete Blank Attributes When Updating?(Y/
N)

999,"""S""":""Hello"""},Type Violation,Expected: N Found: S,,
```

O arquivo de saída está no formato CSV. A primeira linha do arquivo é um cabeçalho, seguida de um registro por item que viola a chave de índice. Os campos desses registros infratores são os seguintes:

- Chave de hash da tabela—o valor da chave de partição do item na tabela.
- Chave de intervalo de—o valor da chave de classificação do item na tabela.
- Valor da chave de hash do GSI—o valor de chave de partição do índice secundário global.
- GSI Hash Key Violation Type—ou `Type Violation` ou `Size Violation`.

- Descrição da violação da chave de hash do GSI— a causa da violação.
- Valor de atualização de chave de hash GSI (PARA USUÁRIO)— no modo de correção, um novo valor fornecido pelo usuário para o atributo.
- Valor chave do intervalo GSI— o valor da chave de classificação do índice secundário global.
- GSI Range Key Violation Type— ouType ViolationouSize Violation.
- Descrição da violação de chave do intervalo GSI— a causa da violação.
- Valor de atualização de chave de intervalo GSI (PARA USUÁRIO)— no modo de correção, um novo valor fornecido pelo usuário para o atributo.
- Excluir atributo em branco ao atualizar (S/N)— No modo de correção, o determina se o item infrator deve ser excluído (Y) ou mantido (N) na tabela, mas apenas se qualquer um dos seguintes campos estiver em branco:
 - GSI Hash Key Update Value(FOR USER)
 - GSI Range Key Update Value(FOR USER)

Se qualquer um desses campos não estiver em branco, Delete Blank Attribute When Updating(Y/N) não terá efeito.

Note

O formato de saída pode variar, dependendo do arquivo de configuração e das opções da linha de comando. Por exemplo, se a tabela tiver uma chave primária simples (sem uma chave de classificação), nenhum campo de chave de classificação estará presente na saída.

Os registros de violação no arquivo podem não estar na ordem classificada.

Correction

Para corrigir violações de chave de índice, use o Detector de violação com a--correctA opção de linha de comando. No modo de correção, o Detector de Violações lê o arquivo de entrada especificado pelo correctionInputPathparâmetro . Esse arquivo tem o mesmo formato que o arquivo detectionOutputPath e, portanto, você pode usar a saída da detecção como a entrada para a correção.

O Detector de Violações fornece duas maneiras diferentes para corrigir violações de chave de índice:

- Violações de exclusão— Excluir os itens da tabela que possuem valores de atributo infratores.
- Violações de atualização— Atualiza os itens da tabela, substituindo os atributos infratores por valores não infratores.

Em ambos os casos, você pode usar o arquivo de saída do modo de detecção como uma entrada para o modo de correção.

Continuando com oProductCatalogExemplo de, suponha que você queira excluir o item infrator da tabela. Para fazer isso, execute a seguinte linha de comando da.

```
$ java -jar ViolationDetector.jar --configFilePath config.txt --correct delete
```

Neste ponto, você será solicitado a confirmar se deseja excluir os itens infratores.

```
Are you sure to delete all violations on the table?y/n
y
Confirmed, will delete violations on the table...
Violation correction from file started: Reading records from file: ../
gsi_violation_check.csv, will delete these records from table.
Violation correction from file finished: Violations delete: 1, Violations Update: 0
```

Agora, tanto `ProductCatalog` quanto `PriceIndex` têm o mesmo número de itens.

Como trabalhar com índices secundários globais: Java

Você pode usar o AWS SDK for Java API de documento para criar uma tabela do Amazon DynamoDB com um ou mais índices secundários globais na tabela e executar consultas usando os índices.

Veja a seguir as etapas comuns para as operações de tabela.

1. Crie uma instância da classe `DynamoDB`.
2. Forneça os parâmetros obrigatórios e opcionais para a operação, criando os objetos de solicitação correspondentes.
3. Chame o método apropriado fornecido pelo cliente que você criou na etapa anterior.

Tópicos

- [Criar uma tabela com um índice secundário global \(p. 584\)](#)
- [Descrever uma tabela com um índice secundário global \(p. 585\)](#)
- [Consultar um índice secundário global \(p. 586\)](#)
- [Exemplo: Índices secundários globais que usam o AWS SDK for Java API do Documento \(p. 587\)](#)

Criar uma tabela com um índice secundário global

Você pode criar índices secundários globais ao mesmo tempo em que cria uma tabela. Para fazer isso, use `CreateTable` e forneça suas especificações para um ou mais índices secundários globais. O exemplo de código Java a seguir cria uma tabela para armazenar informações sobre dados climáticos. A chave de partição é `Location` e a chave de classificação é `Date`. Um índice secundário global chamado `PrecipIndex` permite acesso rápido aos dados de precipitação de vários locais.

Veja a seguir as etapas para criar uma tabela com um índice secundário global, usando a API de documento do DynamoDB.

1. Crie uma instância da classe `DynamoDB`.
2. Crie uma instância da classe `CreateTableRequest` para fornecer as informações solicitadas.

Você deve fornecer o nome da tabela, sua chave primária e os valores de throughput provisionado. Para o índice secundário global, você deve fornecer o nome do índice, suas configurações de throughput provisionado, as definições de atributo da chave de classificação do índice, o esquema de chaves do índice e a projeção do atributo.

3. Chame o método `createTable`, fornecendo o objeto de solicitação como um parâmetro.

O exemplo de código Java a seguir demonstra as etapas anteriores. O código cria uma tabela (`WeatherData`) com um índice secundário global (`PrecipIndex`). A chave de partição do índice é `Date` e a chave de classificação é `Precipitation`. Todos os atributos da tabela estão projetados no índice. Os usuários podem consultar esse índice para obter dados climáticos de uma data específica, opcionalmente, classificar os dados por quantidade de precipitação.

Como `Precipitation` não é um atributo de chave para a tabela, ele não é necessário. No entanto, os itens de `WeatherData` sem `Precipitation` não aparecem no `PrecipIndex`.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

// Attribute definitions
ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<AttributeDefinition>();
```

```
attributeDefinitions.add(new AttributeDefinition()
    .withAttributeName("Location")
    .withAttributeType("S"));
attributeDefinitions.add(new AttributeDefinition()
    .withAttributeName("Date")
    .withAttributeType("S"));
attributeDefinitions.add(new AttributeDefinition()
    .withAttributeName("Precipitation")
    .withAttributeType("N"));

// Table key schema
ArrayList<KeySchemaElement> tableKeySchema = new ArrayList<KeySchemaElement>();
tableKeySchema.add(new KeySchemaElement()
    .withAttributeName("Location")
    .withKeyType(KeyType.HASH)); //Partition key
tableKeySchema.add(new KeySchemaElement()
    .withAttributeName("Date")
    .withKeyType(KeyType.RANGE)); //Sort key

// PrecipIndex
GlobalSecondaryIndex precipIndex = new GlobalSecondaryIndex()
    .withIndexName("PrecipIndex")
    .withProvisionedThroughput(new ProvisionedThroughput()
        .withReadCapacityUnits((long) 10)
        .withWriteCapacityUnits((long) 1))
    .withProjection(new Projection().withProjectionType(ProjectionType.ALL));

ArrayList<KeySchemaElement> indexKeySchema = new ArrayList<KeySchemaElement>();

indexKeySchema.add(new KeySchemaElement()
    .withAttributeName("Date")
    .withKeyType(KeyType.HASH)); //Partition key
indexKeySchema.add(new KeySchemaElement()
    .withAttributeName("Precipitation")
    .withKeyType(KeyType.RANGE)); //Sort key

precipIndex.setKeySchema(indexKeySchema);

CreateTableRequest createTableRequest = new CreateTableRequest()
    .withTableName("WeatherData")
    .withProvisionedThroughput(new ProvisionedThroughput()
        .withReadCapacityUnits((long) 5)
        .withWriteCapacityUnits((long) 1))
    .withAttributeDefinitions(attributeDefinitions)
    .withKeySchema(tableKeySchema)
    .withGlobalSecondaryIndexes(precipIndex);

Table table = dynamoDB.createTable(createTableRequest);
System.out.println(table.getDescription());
```

Você deve aguardar até que o DynamoDB crie a tabela e defina o status da tabela para `ACTIVE`. Depois disso, você pode começar a inserir itens de dados na tabela.

Descreve uma tabela com um índice secundário global

Para obter mais informações sobre índices secundários globais em uma tabela, use `DescribeTable`. Para cada índice, você pode acessar seu nome, esquema de chaves e atributos projetados.

A seguir estão as etapas para acessar informações de índice secundário global uma tabela.

1. Crie uma instância da classe `DynamoDB`.
2. Crie uma instância da classe `Table` para representar o índice com o qual você deseja trabalhar.
3. Chame o método `describe` no objeto `Table`.

O exemplo de código Java a seguir demonstra as etapas anteriores.

Example

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("WeatherData");
TableDescription tableDesc = table.describe();

Iterator<GlobalSecondaryIndexDescription> gsiIter =
    tableDesc.getGlobalSecondaryIndexes().iterator();
while (gsiIter.hasNext()) {
    GlobalSecondaryIndexDescription gsiDesc = gsiIter.next();
    System.out.println("Info for index "
        + gsiDesc.getIndexName() + ":");

    Iterator<KeySchemaElement> kseIter = gsiDesc.getKeySchema().iterator();
    while (kseIter.hasNext()) {
        KeySchemaElement kse = kseIter.next();
        System.out.printf("\t%s: %s\n", kse.getAttributeName(), kse.getKeyType());
    }
    Projection projection = gsiDesc.getProjection();
    System.out.println("\tThe projection type is: "
        + projection.getProjectionType());
    if (projection.getProjectionType().toString().equals("INCLUDE")) {
        System.out.println("\t\tThe non-key projected attributes are: "
            + projection.getNonKeyAttributes());
    }
}
```

Consultar um índice secundário global

Você pode usar `queryEm` um índice secundário global, da mesma forma como você `queryUma tabela do`. Você precisa especificar o nome do índice, os critérios de consulta da chave de partição e da chave de classificação (se houver) do índice, e os atributos que você deseja retornar. Neste exemplo, o índice é `PrecipIndex`, que tem uma chave de partição `Date` e uma chave de classificação `Precipitation`. A consulta de índice retorna todos os dados climáticos de uma data específica, na qual a precipitação é maior que zero.

A seguir estão as etapas para consultar um índice secundário global usando o AWS SDK for Java Documentar API.

1. Crie uma instância da classe `DynamoDB`.
2. Crie uma instância da classe `Table` para representar o índice com o qual você deseja trabalhar.
3. Crie uma instância da classe `Index` para o índice que deseja consultar.
4. Chame o método `query` no objeto `Index`.

O nome do atributo `Date` é uma palavra reservada do DynamoDB. Portanto, use um nome de atributo de expressão como um espaço reservado na `KeyConditionExpression`.

O exemplo de código Java a seguir demonstra as etapas anteriores.

Example

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);
```

```
Table table = dynamoDB.getTable("WeatherData");
Index index = table.getIndex("PrecipIndex");

QuerySpec spec = new QuerySpec()
    .withKeyConditionExpression("#d = :v_date and Precipitation = :v_precip")
    .withNameMap(new NameMap()
        .with("#d", "Date"))
    .withValueMap(new ValueMap()
        .withString(":v_date", "2013-08-10")
        .withNumber(":v_precip", 0));

ItemCollection<QueryOutcome> items = index.query(spec);
Iterator<Item> iter = items.iterator();
while (iter.hasNext()) {
    System.out.println(iter.next().toJSONPretty());
}
```

Exemplo: Índices secundários globais que usam oAWS SDK for Java API do Documento

O código Java de exemplo a seguir mostra como trabalhar com índices secundários globais. O exemplo cria uma tabela chamada `Issues`, que pode ser usada em um sistema de controle de bugs simples para desenvolvimento de software. A chave de partição é `IssueId` e a chave de classificação é `Title`. Há três índices secundários globais nessa tabela:

- `CreateDateIndex`— A chave de partição é `CreateDate` e a chave de classificação é `IssueId`. Além das chaves da tabela, os atributos `Description` e `Status` são projetados no índice.
- `TitleIndex`— A chave de partição é `Title` e a chave de classificação é `IssueId`. Nenhum outro atributo além das chaves da tabela são projetados no índice.
- `DueDateIndex`— A chave de partição é `DueDate` e não há chave de classificação. Todos os atributos da tabela estão projetados no índice.

Depois que a tabela `Issues` é criada, o programa carrega a tabela com os dados que representam relatórios de bugs do software. Ele consulta os dados usando os índices secundários globais. Por fim, o programa exclui a tabela `Issues`.

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

Example

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples.document;

import java.util.ArrayList;
```

```
import java.util.Iterator;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Index;
import com.amazonaws.services.dynamodbv2.document.Item;
import com.amazonaws.services.dynamodbv2.document.ItemCollection;
import com.amazonaws.services.dynamodbv2.document.QueryOutcome;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.document.spec.QuerySpec;
import com.amazonaws.services.dynamodbv2.document.utils.ValueMap;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;
import com.amazonaws.services.dynamodbv2.model.GlobalSecondaryIndex;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.Projection;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;

public class DocumentAPIGlobalSecondaryIndexExample {

    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
    static DynamoDB dynamoDB = new DynamoDB(client);

    public static String tableName = "Issues";

    public static void main(String[] args) throws Exception {

        createTable();
        loadData();

        queryIndex("CreateDateIndex");
        queryIndex("TitleIndex");
        queryIndex("DueDateIndex");

        deleteTable(tableName);
    }

    public static void createTable() {

        // Attribute definitions
        ArrayList<AttributeDefinition> attributeDefinitions = new
        ArrayList<AttributeDefinition>();

        attributeDefinitions.add(new
        AttributeDefinition().withAttributeName("IssueId").withAttributeType("S"));
        attributeDefinitions.add(new
        AttributeDefinition().withAttributeName("Title").withAttributeType("S"));
        attributeDefinitions.add(new
        AttributeDefinition().withAttributeName("CreateDate").withAttributeType("S"));
        attributeDefinitions.add(new
        AttributeDefinition().withAttributeName("DueDate").withAttributeType("S"));

        // Key schema for table
        ArrayList<KeySchemaElement> tableKeySchema = new ArrayList<KeySchemaElement>();
        tableKeySchema.add(new
        KeySchemaElement().withAttributeName("IssueId").withKeyType(KeyType.HASH)); // Partition

        // key
        tableKeySchema.add(new
        KeySchemaElement().withAttributeName("Title").withKeyType(KeyType.RANGE)); // Sort

        // key
    }
}
```

```
// Initial provisioned throughput settings for the indexes
ProvisionedThroughput ptIndex = new
ProvisionedThroughput().withReadCapacityUnits(1L)
    .withWriteCapacityUnits(1L);

// CreateDateIndex
GlobalSecondaryIndex createDateIndex = new
GlobalSecondaryIndex().withIndexName("CreateDateIndex")
    .withProvisionedThroughput(ptIndex)
    .withKeySchema(new
KeySchemaElement().withAttributeName("CreateDate").withKeyType(KeyType.HASH), // Partition

        // key
        new
KeySchemaElement().withAttributeName("IssueId").withKeyType(KeyType.RANGE)) // Sort

// key
    .withProjection(
        new
Projection().withProjectionType("INCLUDE").withNonKeyAttributes("Description", "Status"));

// TitleIndex
GlobalSecondaryIndex titleIndex = new
GlobalSecondaryIndex().withIndexName("TitleIndex")
    .withProvisionedThroughput(ptIndex)
    .withKeySchema(new
KeySchemaElement().withAttributeName("Title").withKeyType(KeyType.HASH), // Partition

        // key
        new
KeySchemaElement().withAttributeName("IssueId").withKeyType(KeyType.RANGE)) // Sort

// key
    .withProjection(new Projection().withProjectionType("KEYS_ONLY"));

// DueDateIndex
GlobalSecondaryIndex dueDateIndex = new
GlobalSecondaryIndex().withIndexName("DueDateIndex")
    .withProvisionedThroughput(ptIndex)
    .withKeySchema(new
KeySchemaElement().withAttributeName("DueDate").withKeyType(KeyType.HASH)) // Partition

        // key
        .withProjection(new Projection().withProjectionType("ALL"));

CreateTableRequest createTableRequest = new
CreateTableRequest().withTableName(tableName)
    .withProvisionedThroughput(
        new ProvisionedThroughput().withReadCapacityUnits((long)
1).withWriteCapacityUnits((long) 1))
    .withAttributeDefinitions(attributeDefinitions).withKeySchema(tableKeySchema)
    .withGlobalSecondaryIndexes(createDateIndex, titleIndex, dueDateIndex);

System.out.println("Creating table " + tableName + "...");
dynamoDB.createTable(createTableRequest);

// Wait for table to become active
System.out.println("Waiting for " + tableName + " to become ACTIVE...");
try {
    Table table = dynamoDB.getTable(tableName);
    table.waitForActive();
}
catch (InterruptedException e) {
    e.printStackTrace();
}
}
```

```
public static void queryIndex(String indexName) {  
  
    Table table = dynamoDB.getTable(tableName);  
  
    System.out.println("\n*****\n");  
    System.out.print("Querying index " + indexName + "...");  
  
    Index index = table.getIndex(indexName);  
  
    ItemCollection<QueryOutcome> items = null;  
  
    QuerySpec querySpec = new QuerySpec();  
  
    if (indexName == "CreateDateIndex") {  
        System.out.println("Issues filed on 2013-11-01");  
        querySpec.withKeyConditionExpression("CreateDate = :v_date and  
begins_with(IssueId, :v_issue)")  
            .withValueMap(new ValueMap().withString(":v_date",  
"2013-11-01").withString(":v_issue", "A-"));  
        items = index.query(querySpec);  
    }  
    else if (indexName == "TitleIndex") {  
        System.out.println("Compilation errors");  
        querySpec.withKeyConditionExpression("Title = :v_title and  
begins_with(IssueId, :v_issue)")  
            .withValueMap(new ValueMap().withString(":v_title", "Compilation  
error").withString(":v_issue", "A-"));  
        items = index.query(querySpec);  
    }  
    else if (indexName == "DueDateIndex") {  
        System.out.println("Items that are due on 2013-11-30");  
        querySpec.withKeyConditionExpression("DueDate = :v_date")  
            .withValueMap(new ValueMap().withString(":v_date", "2013-11-30"));  
        items = index.query(querySpec);  
    }  
    else {  
        System.out.println("\nNo valid index name provided");  
        return;  
    }  
  
    Iterator<Item> iterator = items.iterator();  
  
    System.out.println("Query: printing results...");  
  
    while (iterator.hasNext()) {  
        System.out.println(iterator.next().toJSONPretty());  
    }  
}  
  
public static void deleteTable(String tableName) {  
  
    System.out.println("Deleting table " + tableName + "...");  
  
    Table table = dynamoDB.getTable(tableName);  
    table.delete();  
  
    // Wait for table to be deleted  
    System.out.println("Waiting for " + tableName + " to be deleted...");  
    try {  
        table.waitForDelete();  
    }  
    catch (InterruptedException e) {  
        e.printStackTrace();  
    }  
}
```

```
        }

    public static void loadData() {

        System.out.println("Loading data into table " + tableName + "...");

        // IssueId, Title,
        // Description,
        // CreateDate, LastUpdateDate, DueDate,
        // Priority, Status

        putItem("A-101", "Compilation error", "Can't compile Project X - bad version
number. What does this mean?",
                "2013-11-01", "2013-11-02", "2013-11-10", 1, "Assigned");

        putItem("A-102", "Can't read data file", "The main data file is missing, or the
permissions are incorrect",
                "2013-11-01", "2013-11-04", "2013-11-30", 2, "In progress");

        putItem("A-103", "Test failure", "Functional test of Project X produces errors",
"2013-11-01", "2013-11-02",
                "2013-11-10", 1, "In progress");

        putItem("A-104", "Compilation error", "Variable 'messageCount' was not
initialized.", "2013-11-15",
                "2013-11-16", "2013-11-30", 3, "Assigned");

        putItem("A-105", "Network issue", "Can't ping IP address 127.0.0.1. Please fix
this.", "2013-11-15",
                "2013-11-16", "2013-11-19", 5, "Assigned");

    }

    public static void putItem(

        String issueId, String title, String description, String createDate, String
lastUpdateDate, String dueDate,
        Integer priority, String status) {

        Table table = dynamoDB.getTable(tableName);

        Item item = new Item().withPrimaryKey("IssueId", issueId).withString("Title",
title)
                .withString("Description", description).withString("CreateDate", createDate)
                .withString("LastUpdateDate", lastUpdateDate).withString("DueDate", dueDate)
                .withNumber("Priority", priority).withString("Status", status);

        table.putItem(item);
    }
}
```

Como trabalhar com índices secundários globais: .NET

Você pode usar o AWS SDK for .NET API de baixo nível para criar uma tabela do Amazon DynamoDB com um ou mais índices secundários globais na tabela e executar consultas usando os índices. Essas operações são mapeadas para as operações do DynamoDB correspondentes. Para obter mais informações, consulte o [Referência de API do Amazon DynamoDB](#).

Veja a seguir as etapas comuns para operações de tabela usando a API de baixo nível do .NET.

1. Crie uma instância da classe `AmazonDynamoDBClient`.

2. Forneça os parâmetros obrigatórios e opcionais para a operação, criando os objetos de solicitação correspondentes.

Por exemplo, crie um objeto `CreateTableRequest` para criar uma tabela e um objeto `QueryRequest` para consultar uma tabela ou um índice.

3. Execute o método apropriado fornecido pelo cliente que você criou na etapa anterior.

Tópicos

- [Criar uma tabela com um índice secundário global \(p. 592\)](#)
- [Descreve uma tabela com um índice secundário global \(p. 593\)](#)
- [Consultar um índice secundário global \(p. 594\)](#)
- [Exemplo: Índices secundários globais que usam o AWS SDK for .NET API de baixo nível \(p. 595\)](#)

Criar uma tabela com um índice secundário global

Você pode criar índices secundários globais ao mesmo tempo em que cria uma tabela. Para fazer isso, use `CreateTable` e forneça suas especificações para um ou mais índices secundários globais. O código de exemplo C# a seguir cria uma tabela para armazenar informações sobre os dados climáticos. A chave de partição é `Location` e a chave de classificação é `Date`. Um índice secundário global chamado `PrecipIndex` permite acesso rápido aos dados de precipitação de vários locais.

Veja a seguir as etapas para criar uma tabela com um índice secundário global usando a API de baixo nível do .NET.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Crie uma instância da classe `CreateTableRequest` para fornecer as informações solicitadas.

Você deve fornecer o nome da tabela, sua chave primária e os valores de throughput provisionado. Para o índice secundário global, você deve fornecer o nome do índice, suas configurações de throughput provisionado, as definições de atributo da chave de classificação do índice, o esquema de chaves do índice e a projeção do atributo.

3. Execute o `CreateTable` método fornecendo o objeto de solicitação como um parâmetro.

O exemplo de código C# a seguir demonstra as etapas anteriores. O código cria uma tabela (`WeatherData`) com um índice secundário global (`PrecipIndex`). A chave de partição do índice é `Date` e a chave de classificação é `Precipitation`. Todos os atributos da tabela estão projetados no índice. Os usuários podem consultar esse índice para obter dados climáticos de uma data específica, opcionalmente, classificar os dados por quantidade de precipitação.

Como `Precipitation` não é um atributo de chave para a tabela, ele não é necessário. No entanto, os itens de `WeatherData` sem `Precipitation` não aparecem no `PrecipIndex`.

```
client = new AmazonDynamoDBClient();
string tableName = "WeatherData";

// Attribute definitions
var attributeDefinitions = new List<AttributeDefinition>()
{
    {new AttributeDefinition{
        AttributeName = "Location",
        AttributeType = "S"}},
    {new AttributeDefinition{
        AttributeName = "Date",
        AttributeType = "S"}},
```

```

        {new AttributeDefinition(){
            AttributeName = "Precipitation",
            AttributeType = "N"
        }
    };

    // Table key schema
    var tableKeySchema = new List<KeySchemaElement>()
    {
        {new KeySchemaElement {
            AttributeName = "Location",
            KeyType = "HASH"}}, //Partition key
        {new KeySchemaElement {
            AttributeName = "Date",
            KeyType = "RANGE"} //Sort key
    };
};

// PrecipIndex
var precipIndex = new GlobalSecondaryIndex
{
    IndexName = "PrecipIndex",
    ProvisionedThroughput = new ProvisionedThroughput
    {
        ReadCapacityUnits = (long)10,
        WriteCapacityUnits = (long)1
    },
    Projection = new Projection { ProjectionType = "ALL" }
};

var indexKeySchema = new List<KeySchemaElement> {
    {new KeySchemaElement { AttributeName = "Date", KeyType = "HASH"}}, //Partition key
    {new KeySchemaElement{AttributeName = "Precipitation",KeyType = "RANGE"}} //Sort key
};

precipIndex.KeySchema = indexKeySchema;

CreateTableRequest createTableRequest = new CreateTableRequest
{
    TableName = tableName,
    ProvisionedThroughput = new ProvisionedThroughput
    {
        ReadCapacityUnits = (long)5,
        WriteCapacityUnits = (long)1
    },
    AttributeDefinitions = attributeDefinitions,
    KeySchema = tableKeySchema,
    GlobalSecondaryIndexes = { precipIndex }
};

CreateTableResponse response = client.CreateTable(createTableRequest);
Console.WriteLine(response.CreateTableResult.TableDescription.TableName);
Console.WriteLine(response.CreateTableResult.TableDescription.TableStatus);

```

Você deve aguardar até que o DynamoDB crie a tabela e defina o status da tabela para **ACTIVE**. Depois disso, você pode começar a inserir itens de dados na tabela.

Descreve uma tabela com um índice secundário global

Para obter mais informações sobre índices secundários globais em uma tabela, use **DescribeTable**. Para cada índice, você pode acessar seu nome, esquema de chaves e atributos projetados.

Veja a seguir as etapas para acessar informações de índice secundário global de uma tabela usando a API de baixo nível .NET.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Execute o método `DescribeTable` fornecendo o objeto de solicitação como um parâmetro.
Crie uma instância da classe `DescribeTableRequest` para fornecer as informações solicitadas. Você deve fornecer o nome da tabela.
- 3.

O exemplo de código C# a seguir demonstra as etapas anteriores.

Example

```
client = new AmazonDynamoDBClient();
string tableName = "WeatherData";

DescribeTableResponse response = client.DescribeTable(new DescribeTableRequest { TableName
= tableName });

List<GlobalSecondaryIndexDescription> globalSecondaryIndexes =
response.DescribeTableResult.Table.GlobalSecondaryIndexes;

// This code snippet will work for multiple indexes, even though
// there is only one index in this example.

foreach (GlobalSecondaryIndexDescription gsiDescription in globalSecondaryIndexes) {
    Console.WriteLine("Info for index " + gsiDescription.IndexName + ":");

    foreach (KeySchemaElement kse in gsiDescription.KeySchema) {
        Console.WriteLine("\t" + kse.AttributeName + ": key type is " + kse.KeyType);
    }

    Projection projection = gsiDescription.Projection;
    Console.WriteLine("\tThe projection type is: " + projection.ProjectionType);

    if (projection.ProjectionType.ToString().Equals("INCLUDE")) {
        Console.WriteLine("\t\tThe non-key projected attributes are: "
+ projection.NonKeyAttributes);
    }
}
```

Consultar um índice secundário global

Você pode usar `Query` em um índice secundário global, da mesma forma como você `Query` uma tabela do. Você precisa especificar o nome do índice, os critérios de consulta da chave de partição e da chave de classificação (se houver) do índice, e os atributos que você deseja retornar. Neste exemplo, o índice é `PrecipIndex`, que tem uma chave de partição `Date` e uma chave de classificação `Precipitation`. A consulta de índice retorna todos os dados climáticos de uma data específica, na qual a precipitação é maior que zero.

Veja a seguir as etapas para consultar um índice secundário global usando a API de baixo nível do .NET.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Crie uma instância da classe `QueryRequest` para fornecer as informações solicitadas.
3. Execute o método `Query` fornecendo o objeto de solicitação como um parâmetro.

O nome do atributo `Date` é uma palavra reservada do DynamoDB. Portanto, use um nome de atributo de expressão como um espaço reservado na `KeyConditionExpression`.

O exemplo de código C# a seguir demonstra as etapas anteriores.

Example

```
client = new AmazonDynamoDBClient();

QueryRequest queryRequest = new QueryRequest
{
    TableName = "WeatherData",
    IndexName = "PrecipIndex",
    KeyConditionExpression = "#dt = :v_date and Precipitation > :v_precip",
    ExpressionAttributeNames = new Dictionary<String, String> {
        {"#dt", "Date"}
    },
    ExpressionAttributeValues = new Dictionary<string, AttributeValue> {
        {"<:v_date", new AttributeValue { S = "2013-08-01" }},
        {"<:v_precip", new AttributeValue { N = "0" }}
    },
    ScanIndexForward = true
};

var result = client.Query(queryRequest);

var items = result.Items;
foreach (var currentItem in items)
{
    foreach (string attr in currentItem.Keys)
    {
        Console.Write(attr + "---> ");
        if (attr == "Precipitation")
        {
            Console.WriteLine(currentItem[attr].N);
        }
        else
        {
            Console.WriteLine(currentItem[attr].S);
        }
    }
    Console.WriteLine();
}
```

Exemplo: Índices secundários globais que usam oAWS SDK for .NET API de baixo nível

O código de exemplo C# a seguir mostra como trabalhar com índices secundários globais. O exemplo cria uma tabela chamada `Issues`, que pode ser usada em um sistema de controle de bugs simples para desenvolvimento de software. A chave de partição é `IssueId` e a chave de classificação é `Title`. Há três índices secundários globais nessa tabela:

- `CreateDateIndex`— A chave de partição é `CreateDate` e a chave de classificação é `IssueId`. Além das chaves da tabela, os atributos `Description` e `Status` são projetados no índice.
- `TitleIndex`— A chave de partição é `Title` e a chave de classificação é `IssueId`. Nenhum outro atributo além das chaves da tabela são projetados no índice.
- `DueDateIndex`— A chave de partição é `DueDate` e não há chave de classificação. Todos os atributos da tabela estão projetados no índice.

Depois que a tabela `Issues` é criada, o programa carrega a tabela com os dados que representam relatórios de bugs do software. Ele consulta os dados usando os índices secundários globais. Por fim, o programa exclui a tabela `Issues`.

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.DataModel;  
using Amazon.DynamoDBv2.DocumentModel;  
using Amazon.DynamoDBv2.Model;  
using Amazon.Runtime;  
using Amazon.SecurityToken;  
  
namespace com.amazonaws.codesamples  
{  
    class LowLevelGlobalSecondaryIndexExample  
    {  
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();  
        public static String tableName = "Issues";  
  
        public static void Main(string[] args)  
        {  
            CreateTable();  
            LoadData();  
  
            QueryIndex("CreateDateIndex");  
            QueryIndex("TitleIndex");  
            QueryIndex("DueDateIndex");  
  
            DeleteTable(tableName);  
  
            Console.WriteLine("To continue, press enter");  
            Console.Read();  
        }  
  
        private static void CreateTable()  
        {  
            // Attribute definitions  
            var attributeDefinitions = new List<AttributeDefinition>()  
            {  
                {new AttributeDefinition {  
                    AttributeName = "IssueId", AttributeType = "S"  
                }},  
                {new AttributeDefinition {  
                    AttributeName = "Title", AttributeType = "S"  
                }},  
                {new AttributeDefinition {  
                    AttributeName = "CreateDate", AttributeType = "S"  
                }},  
            };  
        }  
    }  
}
```

```
{new AttributeDefinition {
    AttributeName = "DueDate", AttributeType = "S"
}}
};

// Key schema for table
var tableKeySchema = new List<KeySchemaElement>() {
{
    new KeySchemaElement {
        AttributeName= "IssueId",
        KeyType = "HASH" //Partition key
    }
},
{
    new KeySchemaElement {
        AttributeName = "Title",
        KeyType = "RANGE" //Sort key
    }
}
};

// Initial provisioned throughput settings for the indexes
var ptIndex = new ProvisionedThroughput
{
    ReadCapacityUnits = 1L,
    WriteCapacityUnits = 1L
};

// CreateDateIndex
var createDateIndex = new GlobalSecondaryIndex()
{
    IndexName = "CreateDateIndex",
    ProvisionedThroughput = ptIndex,
    KeySchema = {
        new KeySchemaElement {
            AttributeName = "CreateDate", KeyType = "HASH" //Partition key
        },
        new KeySchemaElement {
            AttributeName = "IssueId", KeyType = "RANGE" //Sort key
        }
    },
    Projection = new Projection
    {
        ProjectionType = "INCLUDE",
        NonKeyAttributes = {
            "Description", "Status"
        }
    }
};

// TitleIndex
var titleIndex = new GlobalSecondaryIndex()
{
    IndexName = "TitleIndex",
    ProvisionedThroughput = ptIndex,
    KeySchema = {
        new KeySchemaElement {
            AttributeName = "Title", KeyType = "HASH" //Partition key
        },
        new KeySchemaElement {
            AttributeName = "IssueId", KeyType = "RANGE" //Sort key
        }
    },
    Projection = new Projection
    {
        ProjectionType = "KEYS_ONLY"
    }
};
```

```
        }

};

// DueDateIndex
var dueDateIndex = new GlobalSecondaryIndex()
{
    IndexName = "DueDateIndex",
    ProvisionedThroughput = ptIndex,
    KeySchema = {
        new KeySchemaElement {
            AttributeName = "DueDate",
            KeyType = "HASH" //Partition key
        }
    },
    Projection = new Projection
    {
        ProjectionType = "ALL"
    }
};

var createTableRequest = new CreateTableRequest
{
    TableName = tableName,
    ProvisionedThroughput = new ProvisionedThroughput
    {
        ReadCapacityUnits = (long)1,
        WriteCapacityUnits = (long)1
    },
    AttributeDefinitions = attributeDefinitions,
    KeySchema = tableKeySchema,
    GlobalSecondaryIndexes = {
        createDateIndex, titleIndex, dueDateIndex
    }
};

Console.WriteLine("Creating table " + tableName + "...");
client.CreateTable(createTableRequest);

WaitUntilTableReady(tableName);
}

private static void LoadData()
{
    Console.WriteLine("Loading data into table " + tableName + "...");

    // IssueId, Title,
    // Description,
    // CreateDate, LastUpdateDate, DueDate,
    // Priority, Status

    putItem("A-101", "Compilation error",
        "Can't compile Project X - bad version number. What does this mean?",
        "2013-11-01", "2013-11-02", "2013-11-10",
        1, "Assigned");

    putItem("A-102", "Can't read data file",
        "The main data file is missing, or the permissions are incorrect",
        "2013-11-01", "2013-11-04", "2013-11-30",
        2, "In progress");

    putItem("A-103", "Test failure",
        "Functional test of Project X produces errors",
        "2013-11-01", "2013-11-02", "2013-11-10",
        1, "In progress");
```

```
putItem("A-104", "Compilation error",
        "Variable 'messageCount' was not initialized.",
        "2013-11-15", "2013-11-16", "2013-11-30",
        3, "Assigned");

putItem("A-105", "Network issue",
        "Can't ping IP address 127.0.0.1. Please fix this.",
        "2013-11-15", "2013-11-16", "2013-11-19",
        5, "Assigned");
}

private static void putItem(
    String issueId, String title,
    String description,
    String createDate, String lastUpdateDate, String dueDate,
    Int32 priority, String status)
{
    Dictionary<String, AttributeValue> item = new Dictionary<string,
    AttributeValue>();

    item.Add("IssueId", new AttributeValue
    {
        S = issueId
    });
    item.Add("Title", new AttributeValue
    {
        S = title
    });
    item.Add("Description", new AttributeValue
    {
        S = description
    });
    item.Add("CreateDate", new AttributeValue
    {
        S = createDate
    });
    item.Add("LastUpdateDate", new AttributeValue
    {
        S = lastUpdateDate
    });
    item.Add("DueDate", new AttributeValue
    {
        S = dueDate
    });
    item.Add("Priority", new AttributeValue
    {
        N = priority.ToString()
    });
    item.Add("Status", new AttributeValue
    {
        S = status
    });

    try
    {
        client.PutItem(new PutItemRequest
        {
            TableName = tableName,
            Item = item
        });
    }
    catch (Exception e)
    {
        Console.WriteLine(e.ToString());
    }
}
```



```
        else
    {
        Console.WriteLine("\nNo valid index name provided");
        return;
    }

queryRequest.KeyConditionExpression = keyConditionExpression;
queryRequest.ExpressionAttributeValues = expressionAttributeValues;

var result = client.Query(queryRequest);
var items = result.Items;
foreach (var currentItem in items)
{
    foreach (string attr in currentItem.Keys)
    {
        if (attr == "Priority")
        {
            Console.WriteLine(attr + "----> " + currentItem[attr].N);
        }
        else
        {
            Console.WriteLine(attr + "----> " + currentItem[attr].S);
        }
    }
    Console.WriteLine();
}
}

private static void DeleteTable(string tableName)
{
    Console.WriteLine("Deleting table " + tableName + "...");
    client.DeleteTable(new DeleteTableRequest
    {
        TableName = tableName
    });
    WaitForTableToDelete(tableName);
}

private static void WaitUntilTableReady(string tableName)
{
    string status = null;
    // Let us wait until table is created. Call DescribeTable.
    do
    {
        System.Threading.Thread.Sleep(5000); // Wait 5 seconds.
        try
        {
            var res = client.DescribeTable(new DescribeTableRequest
            {
                TableName = tableName
            });

            Console.WriteLine("Table name: {0}, status: {1}",
                res.Table.TableName,
                res.Table.TableStatus);
            status = res.Table.TableStatus;
        }
        catch (ResourceNotFoundException)
        {
            // DescribeTable is eventually consistent. So you might
            // get resource not found. So we handle the potential exception.
        }
    } while (status != "ACTIVE");
}

private static void WaitForTableToDelete(string tableName)
```

```
{  
    bool tablePresent = true;  
  
    while (tablePresent)  
    {  
        System.Threading.Thread.Sleep(5000); // Wait 5 seconds.  
        try  
        {  
            var res = client.DescribeTable(new DescribeTableRequest  
            {  
                TableName = tableName  
            });  
  
            Console.WriteLine("Table name: {0}, status: {1}",  
                res.Table.TableName,  
                res.Table.TableStatus);  
        }  
        catch (ResourceNotFoundException)  
        {  
            tablePresent = false;  
        }  
    }  
}  
}
```

Como trabalhar com índices secundários globais:AWS CLI

Você pode usar o AWS CLI para criar uma tabela do Amazon DynamoDB com um ou mais índices secundários globais na tabela e executar consultas usando os índices.

Tópicos

- [Criar uma tabela de com um índice secundário global \(p. 602\)](#)
- [Adicionar um índice secundário global a uma tabela existente \(p. 603\)](#)
- [Descreve uma tabela de com um índice secundário global \(p. 603\)](#)
- [Consultar um índice secundário global \(p. 603\)](#)

Criar uma tabela de com um índice secundário global

Índices secundários globais podem ser criados ao mesmo tempo em que uma tabela é criada. Para isso, use o comando `create-table` e forneça suas especificações para um ou mais índices secundários globais. O exemplo a seguir cria um perfil chamado `GameScores` com um índice secundário global chamado `GameTitleIndex`. A tabela base tem uma chave de partição `deUserId` e uma chave de tipo `GameTitle`, permitindo que você encontre a melhor pontuação de um usuário individual para um jogo específico de forma eficiente, enquanto o GSI tem uma chave de partição `deGameTitle` e uma chave de tipo `TopScore`, permitindo que você encontre rapidamente a pontuação mais alta geral para um jogo específico.

```
aws dynamodb create-table \  
    --table-name GameScores \  
    --attribute-definitions AttributeName=UserId,AttributeType=S \  
                            AttributeName=GameTitle,AttributeType=S \  
                            AttributeName=TopScore,AttributeType=N \  
    --key-schema AttributeName=UserId,KeyType=HASH \  
                            AttributeName=GameTitle,KeyType=RANGE \  
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \  
    --global-secondary-indexes \  
        "[
```

```
{  
    "IndexName": "GameTitleIndex",  
    "KeySchema": [{  
        "AttributeName": "GameTitle",  
        "KeyType": "HASH"  
    },  
    {  
        "AttributeName": "TopScore",  
        "KeyType": "RANGE"  
    }],  
    "Projection": {  
        "ProjectionType": "INCLUDE",  
        "NonKeyAttributes": ["UserId"]  
    }  
}  
]"
```

Você deve aguardar até que o DynamoDB crie a tabela e defina o status da tabela para `ACTIVE`. Depois disso, você pode começar a inserir itens de dados na tabela. Você pode usar [describe-table](#) para determinar o status da criação da tabela.

Adicionar um índice secundário global a uma tabela existente

Índices secundários globais também podem ser adicionados ou modificados após a criação da tabela. Para isso, use o comando `update-table` e forneça suas especificações para um ou mais índices secundários globais. O exemplo a seguir usa o mesmo esquema do exemplo anterior, mas pressupõe que a tabela já foi criada e que estamos adicionando o GSI mais tarde.

```
aws dynamodb update-table \  
    --table-name GameScores \  
    --attribute-definitions AttributeName=TopScore,AttributeType=N \  
    --global-secondary-index-updates \  
    "[  
        {  
            "Create": {  
                "IndexName": "GameTitleIndex",  
                "KeySchema": [{  
                    "AttributeName": "GameTitle",  
                    "KeyType": "HASH"  
                },  
                {  
                    "AttributeName": "TopScore",  
                    "KeyType": "RANGE"  
                }],  
                "Projection": {  
                    "ProjectionType": "INCLUDE",  
                    "NonKeyAttributes": ["UserId"]  
                }  
            }  
        }  
    ]"
```

Descreve uma tabela com um índice secundário global

Para acessar informações sobre índices secundários locais em uma tabela, use o `describe-table` parâmetro. Para cada índice, você pode acessar seu nome, esquema de chaves e atributos projetados.

```
aws dynamodb describe-table --table-name GameScores
```

Consultar um índice secundário global

Você pode usar o `query` Operação em um índice secundário global da mesma forma que você `query` uma tabela do. Você deve especificar o nome do índice, os critérios de consulta da chave de classificação do índice e os atributos que deseja retornar. Neste exemplo, o índice é `GameTitleIndex` e a chave de classificação do índice é `GameTitle`.

Os únicos atributos retornados são aqueles que foram projetados no índice. É possível modificar essa consulta para selecionar atributos não chave também, mas isso exigiria atividades de busca de tabela que são relativamente caras. Para obter mais informações sobre buscas de tabela, consulte [Projeções de atributo \(p. 566\)](#).

```
aws dynamodb query \  
    --index-name GameTitleIndex \  
    --key-condition-expression "GameTitle = :v_game" \  
    --expression-attribute-values '{"":v_game":{"S":"Alien Adventure"} }'
```

Índices secundários locais

Alguns aplicativos só precisam consultar dados usando a chave primária da tabela base. No entanto, podem haver situações em que uma chave de classificação alternativa seria útil. Para dar ao seu aplicativo uma opção de chaves de classificação, você pode criar um ou mais índices secundários locais em uma tabela do Amazon DynamoDB e emitir um `query` contra esses índices.

Tópicos

- [CENÁRIO Uso de um índice secundário local \(p. 604\)](#)
- [Projeções de atributo \(p. 607\)](#)
- [Criar um índice secundário local \(p. 608\)](#)
- [Lendo dados de um índice secundário local \(p. 609\)](#)
- [Gravações de itens e índices secundários locais \(p. 610\)](#)
- [Considerações sobre taxa de transferência provisionada para índices secundários locais \(p. 610\)](#)
- [Considerações sobre armazenamento de índices secundários locais \(p. 612\)](#)
- [Coleções de itens \(p. 612\)](#)
- [Como trabalhar com índices secundários locais: Java \(p. 615\)](#)
- [Como trabalhar com índices secundários locais: .NET \(p. 625\)](#)
- [Como trabalhar com índices secundários locais: AWS CLI \(p. 640\)](#)

CENÁRIO Uso de um índice secundário local

Por exemplo, considere a tabela `Thread` que é definida em [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#). Esta tabela é útil para um aplicativo, como o [AWS Fóruns de discussão da](#). O diagrama a seguir mostra como os itens da tabela seriam organizados. (Nem todos os atributos são mostrados.)

Thread				
ForumName	Subject	LastPostDateTime	Replies	
"S3"	"aaa"	"2015-03-15:17:24:31"	12	...
"S3"	"bbb"	"2015-01-22:23:18:01"	3	...
"S3"	"ccc"	"2015-02-31:13:14:21"	4	...
"S3"	"ddd"	"2015-01-03:09:21:11"	9	...
"EC2"	"yyy"	"2015-02-12:11:07:56"	18	...
"EC2"	"zzz"	"2015-01-18:07:33:42"	0	...
"RDS"	"rrr"	"2015-01-19:01:13:24"	3	...
"RDS"	"sss"	"2015-03-11:06:53:00"	11	...
"RDS"	"ttt"	"2015-10-22:12:19:44"	5	...
...

O DynamoDB armazena todos os itens com o mesmo valor de chave de partição contiguamente. Neste exemplo, dado um determinado `ForumName`, uma operação `Query` poderia localizar imediatamente todos os threads desse fórum. Em um grupo de itens com o mesmo valor de chave de partição, os itens são classificados pelo valor de chave de classificação. Se a chave de classificação (`Subject`) também for fornecida na consulta, o DynamoDB poderá reduzir os resultados retornados - por exemplo, retornar todos os threads do fórum "S3" em que `Subject` comece com a letra "a".

Algumas solicitações podem exigir padrões mais complexos de acesso aos dados. Por exemplo:

- Quais threads do fórum recebem visualizações e respostas?
- Qual thread em um determinado fórum tem o maior número de mensagens?
- Quantos threads foram publicados em um fórum específico em um determinado período?

A ação `Query` não seria suficiente para responder a essas perguntas. Em vez disso, você teria de `Scan` toda a tabela. Para uma tabela com milhões de itens, isso poderia consumir uma grande quantidade de throughput provisionado de leitura e levar muito tempo para ser concluído.

No entanto, é possível especificar um ou mais índices secundários em atributos que não são chaves, como `Replies` ou `LastPostDateTime`.

O índice secundário local mantém uma chave de classificação alternativa para um determinado valor de chave de partição. Um índice secundário local também contém uma cópia de alguns ou de todos os atributos de sua tabela base. Você especifica quais atributos são projetados no índice secundário local ao criar a tabela. Os dados em um índice secundário local são organizados com a mesma chave de partição que a tabela-base, mas com outra chave de classificação. Isso permite que você acesse os itens de dados de forma eficiente nessa dimensão diferente. Para uma maior flexibilidade de consulta ou verificação, você pode criar até cinco índices secundários locais por tabela.

Suponha que um aplicativo precise encontrar todos os threads que foram publicados nos últimos três meses em um fórum específico. Sem um índice secundário local, o aplicativo teria que `Scan` a totalidade do `Thread` e descartar quaisquer publicações que não estivessem dentro do período de tempo especificado. Com um índice secundário local, uma operação `Query` poderia usar `LastPostDateTime` como uma chave de classificação e encontre os dados rapidamente.

O diagrama a seguir mostra um índice secundário local chamado `LastPostIndex`. Observe que a chave de partição é a mesma que a da tabela `Thread`, mas a chave de classificação é `LastPostDateTime`.

LastPostIndex

<i>ForumName</i>	<i>LastPostDateTime</i>	<i>Subject</i>
“S3”	“2015-01-03:09:21:11”	“ddd”
“S3”	“2015-01-22:23:18:01”	“bbb”
“S3”	“2015-02-31:13:14:21”	“ccc”
“S3”	“2015-03-15:17:24:31”	“aaa”
“EC2”	“2015-01-18:07:33:42”	“zzz”
“EC2”	“2015-02-12:11:07:56”	“yyy”
“RDS”	“2015-01-19:01:13:24”	“rrr”
“RDS”	“2015-02-22:12:19:44”	“ttt”
“RDS”	“2015-03-11:06:53:00”	“sss”
...

Cada índice secundário local deve atender às seguintes condições:

- A chave de partição é a mesma que a de sua tabela-base.
- A chave de classificação consiste em exatamente um atributo escalar.
- A chave de classificação da tabela-base é projetada no índice, onde ela atua como um atributo que não é chave.

Neste exemplo, a chave de partição é `ForumName` e a chave de classificação do índice secundário local é `LastPostDateTime`. Além disso, o valor da chave de classificação da tabela base (neste exemplo, `Subject`) é projetado no índice, mas não faz parte da chave do índice. Se um aplicativo precisar de uma lista baseada em `ForumName` e `LastPostDateTime`, ele poderá emitir uma solicitação de `Query` com relação a `LastPostIndex`. Os resultados da consulta são classificados por `LastPostDateTime` e podem ser retornados em ordem crescente ou decrescente. A consulta também pode aplicar condições de chave, como retornar apenas os itens que têm uma `LastPostDateTime` dentro de um período específico.

Cada índice secundário local contém automaticamente as chaves de partição e de classificação de sua tabela-base, mas você pode opcionalmente projetar atributos de chave no índice. Quando você consulta o índice, o DynamoDB pode recuperar esses atributos projetados com eficiência. Quando você consulta um índice secundário local, a consulta também pode recuperar atributos que não são projetados no índice. O DynamoDB busca automaticamente esses atributos na tabela-base, mas com uma latência maior e com custos de throughput provisionado mais altos.

Para qualquer índice secundário local, você pode armazenar até 10 GB de dados por valor de chave de partição distinta. Esta imagem inclui todos os itens na tabela-base, além de todos os itens nos índices, que têm o mesmo valor de chave de partição. Para mais informações, consulte [Coleções de itens \(p. 612\)](#).

Projeções de atributo

Com `LastPostIndex`, um aplicativo poderia usar `ForumName` e `LastPostDateTime` como critérios de consulta. No entanto, para recuperar qualquer atributo adicional, o DynamoDB deve executar operações de leitura adicionais no `ThreadTabela`. Essas leituras extras são conhecidas como buscas e podem aumentar a quantidade total de throughput provisionado necessário para uma consulta.

Suponha que você quisesse preencher uma página da Web com uma lista de todos os threads de "S3" e o número de respostas para cada thread, classificados pela última data/hora de resposta, começando pela resposta mais recente. Para preencher essa lista, você precisaria dos seguintes atributos:

- `Subject`
- `Replies`
- `LastPostDateTime`

A maneira mais eficiente de consultar esses dados e evitar operações de busca, seria projetar o `Replies` da tabela no índice secundário local, como mostrado neste diagrama.

LastPostIndex

<i>ForumName</i>	<i>LastPostDateTime</i>	<i>Subject</i>	<i>Replies</i>
"S3"	"2015-01-03:09:21:11"	"ddd"	9
"S3"	"2015-01-22:23:18:01"	"bbb"	3
"S3"	"2015-02-31:13:14:21"	"ccc"	4
"S3"	"2015-03-15:17:24:31"	"aaa"	12
"EC2"	"2015-01-18:07:33:42"	"zzz"	0
"EC2"	"2015-02-12:11:07:56"	"yyy"	18
"RDS"	"2015-01-19:01:13:24"	"rrr"	3
"RDS"	"2015-02-22:12:19:44"	"ttt"	5
"RDS"	"2015-03-11:06:53:00"	"sss"	11
...

A Projeção é o conjunto de atributos que é copiado de uma tabela para um índice secundário. A chave de partição e a chave de classificação da tabela são sempre projetadas no índice; você pode projetar outros atributos para suportar os requisitos de consulta do aplicativo. Quando você consulta um índice, o Amazon DynamoDB pode acessar qualquer atributo na projeção como se esses atributos estivessem em uma tabela própria.

Quando você cria um índice secundário, você precisa especificar os atributos que serão projetados no índice. O DynamoDB fornece três opções diferentes para isso:

- **KEYS_ONLY**— Cada item do índice consiste apenas nos valores de chaves de partição e das chaves de classificação da tabela, além dos valores de chaves do índice. **KEYS_ONLY** resulta no menor índice secundário possível.
- **INCLUDE**— Além dos atributos descritos em **KEYS_ONLY**, o índice secundário incluirá outros atributos não chave que você especificar.
- **TUDO**— o índice secundário inclui todos os atributos da tabela de origem. Como todos os dados da tabela são duplicados no índice, **TUDO** resulta no maior índice secundário possível.

No diagrama anterior, o atributo `Replies` que não é chave é projetado em `LastPostIndex`. Um aplicativo pode consultar `LastPostIndex` em vez da tabela `Thread` completa para preencher uma página da Web com `Subject`, `Replies` e `LastPostDateTime`. Se quaisquer outros atributos que não são chave fossem solicitados, o DynamoDB precisaria buscar esses atributos na `ThreadTabela` do.

Do ponto de vista de um aplicativo, buscar atributos adicionais da tabela-base é automático e transparente, portanto, não há necessidade de reescrever qualquer lógica de aplicativo. No entanto, essa busca pode reduzir significativamente a vantagem de desempenho do uso de um índice secundário local.

Ao escolher os atributos para projetar em um índice secundário local, você deve considerar a desvantagem entre os custos de throughput provisionado e os custos de armazenamento:

- Se precisar acessar apenas alguns atributos com a latência mais baixa possível, considere projetar apenas os atributos em um índice secundário local. Quanto menor o índice, menores serão os custos de armazenamento e de gravação. Se houver atributos que você precisa buscar ocasionalmente, o custo de throughput provisionado pode ultrapassar o custo por um prazo mais longo do armazenamento desses atributos.
- Se o aplicativo acessar frequentemente alguns atributos que não são chave, considere projetar esses atributos em um índice secundário local. Os custos de armazenamento adicionais do índice secundário local compensarão o custo de realizar verificações de tabela frequentes.
- Se precisar acessar a maioria dos atributos que não são chave com frequência, você poderá projetar esses atributos - inclusive a tabela-base inteira - em um índice secundário local. Isso fornece flexibilidade máxima e menor consumo de taxa de transferência provisionada, porque nenhuma busca será necessária. No entanto, o custo do armazenamento deve aumentar ou até dobrar se você estiver projetando todos os atributos.
- Se o seu aplicativo precisa consultar uma tabela com pouca frequência, mas deve realizar muitas gravações ou atualizações nos dados na tabela, pense em projetar **KEYS_ONLY**. O índice secundário local seria de tamanho mínimo, mas ainda estaria disponível quando necessário para a atividade de consulta.

Criar um índice secundário local

Para criar um ou mais índices secundários locais em uma tabela, use o `LocalSecondaryIndexes` parâmetro da `CreateTable` operação. Os índices secundários locais em uma tabela são criados quando a tabela é criada. Quando você exclui uma tabela, todos os índices secundários locais da tabela também são excluídos.

Você deve especificar um atributo que não seja chave para atuar como a chave de classificação do índice secundário local. O atributo escolhido deve ser de um tipo escalar como `String`, `Number` ou `Binary`. Outros tipos escalares, tipos de documento e tipos de conjunto não são permitidos. Para obter uma lista completa de tipos de dados, consulte [Tipos de dados \(p. 14\)](#).

Important

Para tabelas com índices secundários locais, há um limite de tamanho de 10 GB por valor de chave de partição. Uma tabela com índices secundários locais pode armazenar qualquer número de itens, desde que o tamanho total de qualquer valor de chave de partição não exceda 10 GB. Para mais informações, consulte [Limite de tamanho da coleção de itens \(p. 614\)](#).

Você pode projetar atributos de qualquer tipo de dados em um índice secundário local. Isso inclui escalares, documentos e conjuntos. Para obter uma lista completa de tipos de dados, consulte [Tipos de dados \(p. 14\)](#).

Lendo dados de um índice secundário local

Você pode recuperar itens de um índice secundário local usando o comando `Query` e `Scan` operações. `GetItem`, `BatchGetItem` e `GetItem` operações não podem ser usadas em um índice secundário local.

Consultar um índice secundário local

Em uma tabela do DynamoDB, o valor da chave de partição combinada e o valor da chave de classificação de cada item deve ser exclusivo. No entanto, em um índice secundário local, o valor da chave de classificação não precisa ser exclusivo para um determinado valor de chave de partição. Se houver vários itens no índice secundário local que têm o mesmo valor de chave de classificação, uma `Query` retorna todos os itens que têm o mesmo valor de chave de partição. Na resposta, os itens correspondentes não são retornados em uma ordem específica.

Você pode consultar um índice secundário local usando leituras fortemente consistentes. Para especificar qual tipo de consistência você deseja, use o parâmetro `ConsistentRead` da operação `Query`. Uma leitura fortemente consistente de um índice secundário local sempre retorna os valores atualizados mais recentes. Se a consulta precisar buscar atributos adicionais na tabela base, esses atributos serão consistentes com relação ao índice.

Example

Considere os seguintes dados retornados de uma `Query` que solicita dados dos threads de discussão em um determinado fórum.

```
{  
    "TableName": "Thread",  
    "IndexName": "LastPostIndex",  
    "ConsistentRead": false,  
    "ProjectionExpression": "Subject, LastPostDateTime, Replies, Tags",  
    "KeyConditionExpression":  
        "ForumName = :v_forum and LastPostDateTime between :v_start and :v_end",  
    "ExpressionAttributeValues": {  
        ":v_start": {"S": "2015-08-31T00:00:00.000Z"},  
        ":v_end": {"S": "2015-11-31T00:00:00.000Z"},  
        ":v_forum": {"S": "EC2"}  
    }  
}
```

Nesta consulta:

- Acessos ao DynamoDB `LastPostIndex`, usando o `ForumName` para localizar os itens do índice de "EC2". Todos os itens do índice com essa chave são armazenados lado a lado para rápida recuperação.
- Neste fórum, o DynamoDB usa o índice para pesquisar as chaves que correspondem ao `LastPostDateTime` Condição.
- Como o `Replies` atributo é projetado no índice, o DynamoDB pode recuperar esse atributo sem consumir throughput provisionado adicional.

- O `Tags` atributo não é projetado no índice, portanto, o DynamoDB deve acessar o atributo `Threads` para buscar este atributo.
- Os resultados são retornados, classificados por `LastPostDateTime`. As entradas de índice são classificadas por valor de chave de partição e, em seguida, pelo valor de chave de classificação, e `Query` retorna-as na ordem em que são armazenadas. (Você pode usar o parâmetro `ScanIndexForward` para retornar os resultados em ordem decrescente.)

Como o `Tags` atributo não é projetado no índice secundário local, o DynamoDB deve consumir unidades de capacidade de leitura adicionais para buscar esse atributo na tabela-base. Se for necessário executar essa consulta com frequência, projete `Tags` no `LastPostIndex` para evitar a busca na tabela base. No entanto, se for necessário acessar `Tags` apenas ocasionalmente, o custo do armazenamento adicional para a projeção de `Tags` no índice pode não valer a pena.

Verificar um índice secundário local

Você pode usar `Scan` para recuperar todos os dados de um índice secundário local. Você deve fornecer o nome da tabela-base e o nome de índice na solicitação. Com `Scan`, o DynamoDB lê todos os dados do índice e os retorna para o aplicativo. Você também pode solicitar que apenas alguns dos dados sejam retornados, e que os dados restantes sejam descartados. Para fazer isso, use o parâmetro `FilterExpression` da API `Scan`. Para mais informações, consulte [Expressões de filtro de Scan \(p. 509\)](#).

Gravações de itens e índices secundários locais

O DynamoDB mantém automaticamente todos os índices secundários locais sincronizados com suas respectivas tabelas base. Os aplicativos nunca gravam diretamente em um índice. No entanto, é importante compreender as implicações de como o DynamoDB mantém esses índices.

Ao criar um índice secundário local, você especifica um atributo para servir como a chave de classificação do índice. Você também especifica um tipo de dados desse atributo. Isso significa que sempre que você grava um item na tabela-base, se o item define um atributo de chave do índice, seu tipo deve corresponder ao tipo de dados do esquema de chaves do índice. No caso de `LastPostIndex`, a chave de classificação de `LastPostDateTime` no índice é definida como um tipo de dados `String`. Se tentar adicionar um item à `Threads` e especificar um tipo de dados diferente para `LastPostDateTime` (`comoNumber`), o DynamoDB retorna uma `ValidationException` devido à incompatibilidade do tipo de dados.

Não há necessidade de um relacionamento de um para um entre os itens em uma tabela-base e os itens em um índice secundário local. Na verdade, esse comportamento pode ser vantajoso para muitos aplicativos.

Os custos das atividades de gravação em uma tabela com muitos índices secundários serão mais altos do que em uma tabela com um número menor de índices. Para mais informações, consulte [Considerações sobre taxa de transferência provisionada para índices secundários locais \(p. 610\)](#).

Important

Para tabelas com índices secundários locais, há um limite de tamanho de 10 GB por valor de chave de partição. Uma tabela com índices secundários locais pode armazenar qualquer número de itens, desde que o tamanho total de qualquer valor de chave de partição não exceda 10 GB. Para mais informações, consulte [Limite de tamanho da coleção de itens \(p. 614\)](#).

Considerações sobre taxa de transferência provisionada para índices secundários locais

Ao criar uma tabela no DynamoDB, você provisiona unidades de capacidade de leitura e gravação para a carga de trabalho esperada na tabela. Essa carga de trabalho inclui a atividade de leitura e gravação nos índices secundários locais da tabela.

Para visualizar as taxas atuais da capacidade de throughput provisionado, consulte[Definição de preços do Amazon DynamoDB](#).

unidades de capacidade de leitura

Quando você consulta um índice secundário local, o número de unidades de capacidade de leitura consumidas depende de como os dados são acessados.

Assim como ocorre com as consultas de tabela, uma consulta de índice pode usar leituras fortemente consistentes ou eventualmente consistentes, dependendo do valor de `ConsistentRead`. Uma leitura fortemente consistente consome uma unidade de capacidade de leitura, uma leitura eventualmente consistente consome apenas metade disso. Assim, escolhendo leituras eventualmente consistentes, você pode reduzir seus encargos de unidade de capacidade de leitura.

Para consultas do índice que solicitam apenas chaves de índice e atributos projetados, o DynamoDB calcula a atividade de leitura provisionada da mesma forma que para consultas em tabelas. A única diferença é que o cálculo é baseado no tamanho das entradas de índice, em vez do tamanho do item na tabela-base. O número de unidades de capacidade de leitura é a soma de todos os tamanhos de atributos projetados em todos os itens retornados; o resultado é, então, arredondado para o próximo limite de 4 KB. Para obter mais informações sobre como o DynamoDB calcula a utilização da taxa de transferência provisionada, consulte[Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB \(p. 345\)](#).

Para consultas do índice que leem atributos que não estão projetados no índice secundário local, o DynamoDB precisa buscar esses atributos da tabela-base, além de ler os atributos projetados no índice. Essas buscas ocorrem quando você inclui quaisquer atributos não projetados nos parâmetros `Select` ou `ProjectionExpression` da operação `Query`. A busca causa latência adicional nas respostas da consulta, e também incorre em um custo mais alto de throughput provisionado mais alto: Além das leituras do índice secundário local descritas anteriormente, você é cobrado pelas unidades de capacidade de leitura de cada item da tabela-base buscado. Essa cobrança é para ler cada item inteiro da tabela, não apenas os atributos solicitados.

Tamanho máximo dos resultados retornados por um `Query` operação é de 1 MB. Isso inclui os tamanhos de todos os nomes e valores de atributos de todos os itens retornados. No entanto, se uma consulta em um índice secundário local fizer o DynamoDB buscar atributos de item da tabela-base, o tamanho máximo dos dados no resultado poderá ser menor. Neste caso, o tamanho do resultado é a soma de:

- O tamanho dos itens correspondentes no índice, arredondado para os próximos 4 KB.
- O tamanho de cada item correspondente na tabela-base, com cada item individualmente arredondado para os próximos 4 KB.

Usando esta fórmula, o tamanho máximo dos resultados retornados por uma operação `Query` ainda é 1 MB.

Por exemplo, considere uma tabela na qual o tamanho de cada item é 300 bytes. Há um índice secundário local nessa tabela, mas apenas 200 bytes de cada item é projetado no índice. Agora suponha que você `Query` esse índice, que a consulta requer buscas de tabela para cada item, e que a consulta retorne 4 itens. O DynamoDB soma o seguinte:

- O tamanho dos itens correspondentes no índice: $200 \text{ bytes} \times 4 \text{ itens} = 800 \text{ bytes}$; isso é, então, arredondado para 4 KB.
- O tamanho de cada item correspondente na tabela-base: $(300 \text{ bytes, arredondados para 4 KB}) \times 4 \text{ itens} = 16 \text{ KB}$.

O tamanho total dos dados no resultado é, portanto, 20 KB.

Unidades de capacidade de gravação

Quando um item é adicionado, atualizado ou excluído de uma tabela, a atualização dos índices secundários locais consome unidades de capacidade de gravação provisionadas para a tabela. O custo total da taxa de transferência provisionada para uma gravação é a soma das unidades de capacidade de gravação consumidas pela gravação na tabela e aquelas consumidas pela atualização dos índices secundários locais.

O custo de gravar um item em um índice secundário local depende de vários fatores:

- Se você gravar um novo item na tabela que define um atributo indexado, ou atualizar um item existente para definir um atributo indexado indefinido anteriormente, uma operação de gravação é necessária para inserir o item no índice.
- Se uma atualização na tabela alterar o valor de um atributo de chave indexado (de A para B), duas gravações serão necessárias, uma para excluir o item anterior do índice e outra gravação para inserir o novo item no índice.
- Se um item estava presente no índice, mas uma gravação na tabela fez com que o atributo indexado fosse excluído, uma gravação é necessária para excluir a projeção do item antigo do índice.
- Se um item não estiver presente no índice antes ou depois que o item é atualizado, não haverá custo de gravação adicionais para o índice.

Todos esses fatores supõem que o tamanho de cada item no índice seja menor ou igual ao tamanho do item 1 KB para calcular unidades de capacidade de gravação. Entradas de índice maiores exigirão unidades adicionais de capacidade de gravação. Você pode minimizar os custos de gravação considerando de quais atributos suas consultas precisam para retornar e projetar apenas esses atributos no índice.

Considerações sobre armazenamento de índices secundários locais

Quando um aplicativo grava um item em uma tabela, o DynamoDB copia automaticamente o subconjunto correto de atributos em todos os índices secundários locais nos quais esses atributos devem aparecer. SuasAWSA conta é cobrada pelo armazenamento do item na tabela-base e de atributos em qualquer índice secundário local dessa tabela.

A quantidade de espaço usada por um item do índice é a soma do seguinte:

- O tamanho em bytes da chave primária da tabela-base (chave de partição e chave de classificação)
- O tamanho em bytes do atributo de chave do índice
- O tamanho em bytes dos atributos projetados (se houver)
- 100 bytes de sobrecarga por item de índice

Para estimar os requisitos de armazenamento de um índice secundário local, você pode estimar o tamanho médio de um item no índice e multiplicar pelo número de itens da tabela-base.

Se uma tabela contém um item em que um determinado atributo não é definido, mas é definido como uma chave de classificação do índice, o DynamoDB não grava quaisquer dados desse item no índice.

Coleções de itens

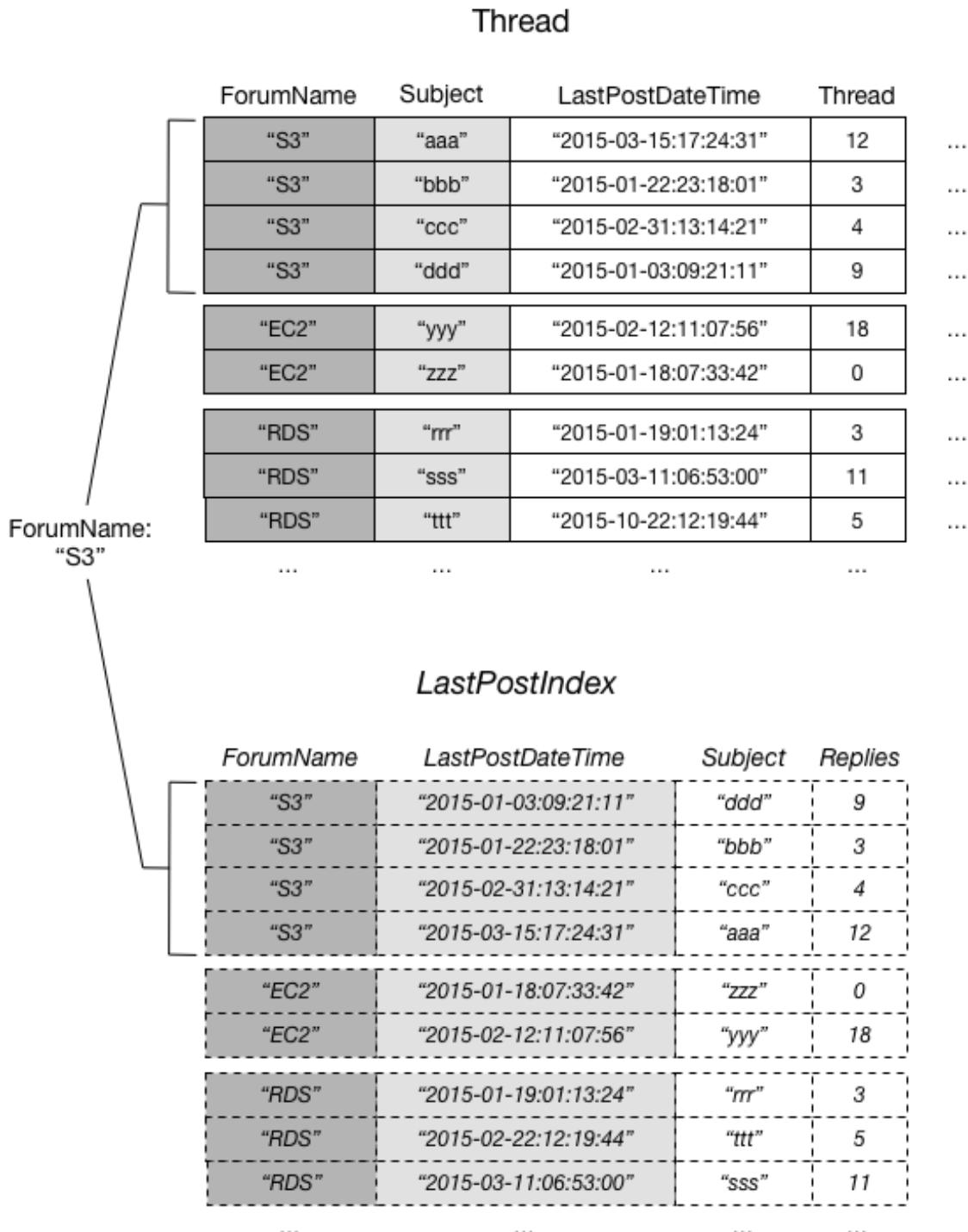
Note

A seção refere-se apenas a tabelas que têm índices secundários locais.

No DynamoDB, umColeta de itensO é qualquer grupo de itens que têm o mesmo valor de chave de partição em uma tabela e todos os seus índices secundários locais. Nos exemplos usados nesta seção, a

chave de partição da tabela `Thread` é `ForumName`, e a chave de partição de `LastPostIndex` também é `ForumName`. Todos os itens da tabela e do índice com o mesmo `ForumName` fazem parte da mesma coleção de itens. Por exemplo, na caixa de `Thread` a tabela do `LastPostIndex` índice secundário local, há uma coleção de itens para o fórum `EC2` e uma coleção de itens diferentes para o fórum `RDS`.

O seguinte diagrama mostra a coleção de itens do fórum do S3.



Neste diagrama, a coleção de itens consiste em todos os itens em `Thread` e `LastPostIndex` em que o valor da chave da partição `ForumName` é "S3". Se houver outros índices secundários locais na tabela, todos os itens nesses índices com `ForumName` igual a "S3" também farão parte da coleção de itens.

Você pode usar qualquer uma das operações a seguir no DynamoDB para retornar informações sobre coleções de itens:

- `BatchWriteItem`
- `DeleteItem`
- `PutItem`
- `UpdateItem`

Cada uma dessas operações é compatível com o parâmetro `ReturnItemCollectionMetrics`. Ao definir esse parâmetro como `SIZE`, você pode exibir informações sobre o tamanho de cada coleção de itens no índice.

Example

Este é um exemplo da saída de uma operação `UpdateItem` na tabela `Thread`, com `ReturnItemCollectionMetrics` definido como `SIZE`. O item que foi atualizado tinha um valor `ForumName` de "EC2", portanto, a saída inclui informações sobre essa coleção de itens.

```
{  
    ItemCollectionMetrics: {  
        ItemCollectionKey: {  
            ForumName: "EC2"  
        },  
        SizeEstimateRangeGB: [0.0, 1.0]  
    }  
}
```

O `SizeEstimateRangeGB` objeto mostra que o tamanho dessa coleção de itens está entre 0 e 1 GB. O DynamoDB atualiza periodicamente essa estimativa de tamanho, portanto, os números podem ser diferentes na próxima vez em que o item é modificado.

Limite de tamanho da coleção de itens

O tamanho máximo de uma coleção de itens é de 10 GB. Esse limite não se aplica a tabelas sem índices secundários globais. Apenas as tabelas que têm um ou mais índices secundários locais são afetadas.

Se uma coleção de itens exceder o limite de 10 GB, o DynamoDB retornará `ItemCollectionSizeLimitExceeded`. Não será possível adicionar mais itens à coleção de itens ou aumentar os tamanhos dos itens que estão na coleção de itens. (As operações de leitura e gravação que diminuem o tamanho da coleção de itens ainda são permitidas.) Você ainda pode adicionar itens a outras coleções de itens.

Para reduzir o tamanho de uma coleção de itens, você pode executar uma das seguintes ações:

- Excluir todos os itens desnecessários com o valor da chave de partição em questão. Quando você exclui esses itens da tabela-base, o DynamoDB também remove quaisquer entradas de índice que têm o mesmo valor de chave de partição.
- Atualize os itens, removendo atributos ou reduzindo o tamanho dos atributos. Se esses atributos forem projetados em qualquer índice secundário local, o DynamoDB também reduzirá o tamanho das entradas do índice correspondentes.
- Crie uma nova tabela com a mesma chave de partição e chave de classificação e, em seguida, mova os itens da tabela antiga para a nova tabela. Essa pode ser uma boa abordagem, se uma tabela tiver dados históricos que são acessados com pouca frequência. Você também pode considerar o arquivamento desses dados históricos no Amazon Simple Storage Service (Amazon S3).

Quando o tamanho total da coleção de itens cair abaixo de 10 GB, você pode adicionar itens novamente com o mesmo valor de chave de partição.

Recomendamos como uma melhor prática que você instrumente seu aplicativo para monitorar os tamanhos de suas coleções de itens. Uma maneira de fazer isso é definir o parâmetro `ReturnItemCollectionMetrics` como `SIZE` sempre que você usar `BatchWriteItem`, `DeleteItem`, `PutItem` ou `UpdateItem`. Seu aplicativo deve examinar o objeto `ReturnItemCollectionMetrics` na saída e gerar uma mensagem de erro sempre que uma coleção de itens exceder um limite definido pelo usuário (por exemplo, 8 GB). Configurar um limite menor que 10 GB oferece um sistema de aviso antecipado para que você saiba que uma coleção de itens está se aproximando do limite a tempo de resolver o problema.

Coleções de itens e partições

A tabela e os dados de índice de cada coleção de itens são armazenados em uma única partição. Tomando como referência o exemplo da tabela `Thread`, todos os itens da tabela base e do índice com o mesmo atributo `ForumName` seriam armazenados na mesma partição. A coleção de itens "S3" seria armazenada em uma partição, "EC2" em outra partição e "RDS" em uma terceira partição.

Você deve criar seus aplicativos para que os dados da tabela sejam distribuídos uniformemente entre diferentes valores de chave de partição. Para tabelas com índices secundários locais, seus aplicativos não devem criar pontos de atividade de leitura e gravação em uma única coleção de itens em uma única partição.

Como trabalhar com índices secundários locais: Java

Você pode usar o comando AWS SDK for Java API de documento do para criar uma tabela do Amazon DynamoDB com um ou mais índices secundários locais na tabela e executar consultas usando os índices.

Veja a seguir as etapas comuns para operações de tabela usando a API de documento do AWS SDK for Java.

1. Crie uma instância da classe `DynamoDB`.
2. Forneça os parâmetros obrigatórios e opcionais para a operação, criando os objetos de solicitação correspondentes.
3. Chame o método apropriado fornecido pelo cliente que você criou na etapa anterior.

Tópicos

- [Criar uma tabela de com um índice secundário local \(p. 615\)](#)
- [Descreve uma tabela de com um índice secundário local \(p. 617\)](#)
- [Consultar um índice secundário local \(p. 618\)](#)
- [Exemplo: Índices secundários locais que usam a API de documento Java \(p. 618\)](#)

Criar uma tabela de com um índice secundário local

Índices secundários locais devem ser criados ao mesmo tempo em que uma tabela é criada. Para isso, use o comando `createTable` e forneça suas especificações para um ou mais índices secundários locais. O código Java de exemplo a seguir cria uma tabela para armazenar informações sobre músicas em uma coleção de músicas. A chave de partição é `Artist` e a chave de classificação é `SongTitle`. Índice secundário, `AlbumTitleIndex`, facilita consultas por título de álbum.

Veja a seguir as etapas para criar uma tabela com um índice secundário local usando a API de documento do DynamoDB.

1. Crie uma instância da classe `DynamoDB`.
2. Crie uma instância da classe `CreateTableRequest` para fornecer as informações solicitadas.

Você deve fornecer o nome da tabela, sua chave primária e os valores de throughput provisionado. Para o índice secundário local, você deve fornecer o nome do índice, o nome e o tipo de dados da chave de classificação do índice, o esquema de chave do índice e a projeção do atributo.

3. Chame o método `createTable`, fornecendo o objeto de solicitação como um parâmetro.

O exemplo de código Java a seguir demonstra as etapas anteriores. O código cria uma tabela (`Music`) com um índice secundário no `AlbumTitleAttribute`. A chave de partição de tabela e a chave de classificação, bem como a chave de classificação de índice, são os únicos atributos projetados para o índice.

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

String tableName = "Music";

CreateTableRequest createTableRequest = new CreateTableRequest().withTableName(tableName);

//ProvisionedThroughput
createTableRequest.setProvisionedThroughput(new
    ProvisionedThroughput().withReadCapacityUnits((long)5).withWriteCapacityUnits((long)5));

//AttributeDefinitions
ArrayList<AttributeDefinition> attributeDefinitions= new ArrayList<AttributeDefinition>();
attributeDefinitions.add(new
    AttributeDefinition().withAttributeName("Artist").withAttributeType("S"));
attributeDefinitions.add(new
    AttributeDefinition().withAttributeName("SongTitle").withAttributeType("S"));
attributeDefinitions.add(new
    AttributeDefinition().withAttributeName("AlbumTitle").withAttributeType("S"));

createTableRequest.setAttributeDefinitions(attributeDefinitions);

//KeySchema
ArrayList<KeySchemaElement> tableKeySchema = new ArrayList<KeySchemaElement>();
tableKeySchema.add(new
    KeySchemaElement().withAttributeName("Artist").withKeyType(KeyType.HASH)); //Partition
key
tableKeySchema.add(new
    KeySchemaElement().withAttributeName("SongTitle").withKeyType(KeyType.RANGE)); //Sort key

createTableRequest.setKeySchema(tableKeySchema);

ArrayList<KeySchemaElement> indexKeySchema = new ArrayList<KeySchemaElement>();
indexKeySchema.add(new
    KeySchemaElement().withAttributeName("Artist").withKeyType(KeyType.HASH)); //Partition
key
indexKeySchema.add(new
    KeySchemaElement().withAttributeName("AlbumTitle").withKeyType(KeyType.RANGE)); //Sort
key

Projection projection = new Projection().withProjectionType(ProjectionType.INCLUDE);
ArrayList<String> nonKeyAttributes = new ArrayList<String>();
nonKeyAttributes.add("Genre");
nonKeyAttributes.add("Year");
projection.setNonKeyAttributes(nonKeyAttributes);

LocalSecondaryIndex localSecondaryIndex = new LocalSecondaryIndex()

    .withIndexName("AlbumTitleIndex").withKeySchema(indexKeySchema).withProjection(projection);
```

```
ArrayList<LocalSecondaryIndex> localSecondaryIndexes = new
    ArrayList<LocalSecondaryIndex>();
localSecondaryIndexes.add(localSecondaryIndex);
createTableRequest.setLocalSecondaryIndexes(localSecondaryIndexes);

Table table = dynamoDB.createTable(createTableRequest);
System.out.println(table.getDescription());
```

Você deve aguardar até que o DynamoDB crie a tabela e defina o status da tabela para ACTIVE. Depois disso, você pode começar a inserir itens de dados na tabela.

Descreve uma tabela com um índice secundário local

Para acessar informações sobre índices secundários locais em uma tabela, use o `describeTable` Método do. Para cada índice, você pode acessar seu nome, esquema de chaves e atributos projetados.

Veja a seguir as etapas para acessar informações de índice secundário local em uma tabela usando o AWS SDK for Java Documentar API.

1. Crie uma instância da classe `DynamoDB`.
2. Crie uma instância da classe `Table`. Você deve fornecer o nome da tabela.
3. Chame o método `describeTable` no objeto `Table`.

O exemplo de código Java a seguir demonstra as etapas anteriores.

Example

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

String tableName = "Music";

Table table = dynamoDB.getTable(tableName);

TableDescription tableDescription = table.describe();

List<LocalSecondaryIndexDescription> localSecondaryIndexes
    = tableDescription.getLocalSecondaryIndexes();

// This code snippet will work for multiple indexes, even though
// there is only one index in this example.

Iterator<LocalSecondaryIndexDescription> lsiIter = localSecondaryIndexes.iterator();
while (lsiIter.hasNext()) {

    LocalSecondaryIndexDescription lsiDescription = lsiIter.next();
    System.out.println("Info for index " + lsiDescription.getIndexName() + ":");
    Iterator<KeySchemaElement> kseIter = lsiDescription.getKeySchema().iterator();
    while (kseIter.hasNext()) {
        KeySchemaElement kse = kseIter.next();
        System.out.printf("\t%s: %s\n", kse.getAttributeName(), kse.getKeyType());
    }
    Projection projection = lsiDescription.getProjection();
    System.out.println("\tThe projection type is: " + projection.getProjectionType());
    if (projection.getProjectionType().toString().equals("INCLUDE")) {
        System.out.println("\t\tThe non-key projected attributes are: " +
projection.getNonKeyAttributes());
    }
}
```

Consultar um índice secundário local

Você pode usar o comando `Query` operação em um índice secundário local da mesma forma que você `Query` uma tabela do. Você deve especificar o nome do índice, os critérios de consulta da chave de classificação do índice e os atributos que deseja retornar. Neste exemplo, o índice `AlbumTitleIndex` é a chave de classificação do índice `AlbumTitle`.

Os únicos atributos retornados são aqueles que foram projetados no índice. É possível modificar essa consulta para selecionar atributos não chave também, mas isso exigiria atividades de busca de tabela que são relativamente caras. Para obter mais informações sobre buscas de tabela, consulte [Projeções de atributo \(p. 607\)](#).

A seguir estão as etapas para consultar um índice secundário local usando o comando AWS SDK for Java Documentar API.

1. Crie uma instância da classe `DynamoDB`.
2. Crie uma instância da classe `Table`. Você deve fornecer o nome da tabela.
3. Crie uma instância da classe `Index`. Você deve fornecer o nome do índice.
4. Chame o método `query` da classe `Index`.

O exemplo de código Java a seguir demonstra as etapas anteriores.

Example

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
DynamoDB dynamoDB = new DynamoDB(client);

String tableName = "Music";

Table table = dynamoDB.getTable(tableName);
Index index = table.getIndex("AlbumTitleIndex");

QuerySpec spec = new QuerySpec()
    .withKeyConditionExpression("Artist = :v_artist and AlbumTitle = :v_title")
    .WithValueMap(new ValueMap()
        .withString(":v_artist", "Acme Band")
        .withString(":v_title", "Songs About Life"));

ItemCollection<QueryOutcome> items = index.query(spec);

Iterator<Item> itemsIter = items.iterator();

while (itemsIter.hasNext()) {
    Item item = itemsIter.next();
    System.out.println(item.toJSONPretty());
}
```

Exemplo: Índices secundários locais que usam a API de documento Java

O código Java de exemplo a seguir mostra como trabalhar com índices secundários locais no Amazon DynamoDB. O exemplo cria uma tabela do denominada `CustomerOrders` com uma chave de partição `deCustomerId` uma chave de tipo `orderId`. Há dois índices secundários locais nessa tabela:

- `OrderCreationDateIndex`— A chave de classificação é `OrderCreationDate`, e os seguintes atributos são projetados no índice:
 - `ProductCategory`
 - `ProductName`
 - `OrderStatus`

- `ShipmentTrackingId`
- `IsOpenIndex`— A chave de classificação é `IsOpen`, e todos os atributos da tabela estão projetados no índice.

Após o `CustomerOrders` tabela é criada, o programa carrega a tabela com dados que representam pedidos de clientes. Ele consulta os dados usando os índices secundários locais. Por fim, o programa exclui a tabela `CustomerOrders`.

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código Java \(p. 334\)](#).

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
package com.amazonaws.codesamples.document;  
  
import java.util.ArrayList;  
import java.util.Iterator;  
  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Index;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.ItemCollection;  
import com.amazonaws.services.dynamodbv2.document.PutItemOutcome;  
import com.amazonaws.services.dynamodbv2.document.QueryOutcome;  
import com.amazonaws.services.dynamodbv2.document.Table;  
import com.amazonaws.services.dynamodbv2.document.spec.QuerySpec;  
import com.amazonaws.services.dynamodbv2.document.utils.ValueMap;  
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;  
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;  
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;  
import com.amazonaws.services.dynamodbv2.model.KeyType;  
import com.amazonaws.services.dynamodbv2.model.LocalSecondaryIndex;  
import com.amazonaws.services.dynamodbv2.model.Projection;  
import com.amazonaws.services.dynamodbv2.model.ProjectionType;  
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;  
import com.amazonaws.services.dynamodbv2.model.ReturnConsumedCapacity;  
import com.amazonaws.services.dynamodbv2.model.Select;  
  
public class DocumentAPILocalSecondaryIndexExample {  
  
    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();  
    static DynamoDB dynamoDB = new DynamoDB(client);  
  
    public static String tableName = "CustomerOrders";
```

```
public static void main(String[] args) throws Exception {

    createTable();
    loadData();

    query(null);
    query("IsOpenIndex");
    query("OrderCreationDateIndex");

    deleteTable(tableName);

}

public static void createTable() {

    CreateTableRequest createTableRequest = new
CreateTableRequest().withTableName(tableName)
    .withProvisionedThroughput(
        new ProvisionedThroughput().withReadCapacityUnits((long)
1).withWriteCapacityUnits((long) 1));

    // Attribute definitions for table partition and sort keys
    ArrayList<AttributeDefinition> attributeDefinitions = new
ArrayList<AttributeDefinition>();
    attributeDefinitions.add(new
AttributeDefinition().withAttributeName("CustomerId").withAttributeType("S"));
    attributeDefinitions.add(new
AttributeDefinition().withAttributeName("OrderId").withAttributeType("N"));

    // Attribute definition for index primary key attributes
    attributeDefinitions
        .add(new
AttributeDefinition().withAttributeName("OrderCreationDate").withAttributeType("N"));
    attributeDefinitions.add(new
AttributeDefinition().withAttributeName("IsOpen").withAttributeType("N"));

    createTableRequest.setAttributeDefinitions(attributeDefinitions);

    // Key schema for table
    ArrayList<KeySchemaElement> tableKeySchema = new ArrayList<KeySchemaElement>();
    tableKeySchema.add(new
KeySchemaElement().withAttributeName("CustomerId").withKeyType(KeyType.HASH)); // Partition

    // key
    tableKeySchema.add(new
KeySchemaElement().withAttributeName("OrderId").withKeyType(KeyType.RANGE)); // Sort

    // key
    createTableRequest.setKeySchema(tableKeySchema);

    ArrayList<LocalSecondaryIndex> localSecondaryIndexes = new
ArrayList<LocalSecondaryIndex>();

    // OrderCreationDateIndex
    LocalSecondaryIndex orderCreationDateIndex = new
LocalSecondaryIndex().withIndexName("OrderCreationDateIndex");

    // Key schema for OrderCreationDateIndex
    ArrayList<KeySchemaElement> indexKeySchema = new ArrayList<KeySchemaElement>();
    indexKeySchema.add(new
KeySchemaElement().withAttributeName("CustomerId").withKeyType(KeyType.HASH)); // Partition

    // key
}
```

```
    indexKeySchema.add(new
KeySchemaElement().withAttributeName("OrderCreationDate").withKeyType(KeyType.RANGE)); // Sort

        // key

    orderCreationDateIndex.setKeySchema(indexKeySchema);

    // Projection (with list of projected attributes) for
    // OrderCreationDateIndex
    Projection projection = new
Projection().withProjectionType(ProjectionType.INCLUDE);
    ArrayList<String> nonKeyAttributes = new ArrayList<String>();
    nonKeyAttributes.add("ProductCategory");
    nonKeyAttributes.add("ProductName");
    projection.setNonKeyAttributes(nonKeyAttributes);

    orderCreationDateIndex.setProjection(projection);

    localSecondaryIndexes.add(orderCreationDateIndex);

    // IsOpenIndex
    LocalSecondaryIndex isOpenIndex = new
LocalSecondaryIndex().withIndexName("IsOpenIndex");

    // Key schema for IsOpenIndex
    indexKeySchema = new ArrayList<KeySchemaElement>();
    indexKeySchema.add(new
KeySchemaElement().withAttributeName("CustomerId").withKeyType(KeyType.HASH)); // Partition

        // key
    indexKeySchema.add(new
KeySchemaElement().withAttributeName("IsOpen").withKeyType(KeyType.RANGE)); // Sort

        // key

    // Projection (all attributes) for IsOpenIndex
    projection = new Projection().withProjectionType(ProjectionType.ALL);

    isOpenIndex.setKeySchema(indexKeySchema);
    isOpenIndex.setProjection(projection);

    localSecondaryIndexes.add(isOpenIndex);

    // Add index definitions to CreateTable request
    createTableRequest.setLocalSecondaryIndexes(localSecondaryIndexes);

    System.out.println("Creating table " + tableName + "...");
    System.out.println(dynamoDB.createTable(createTableRequest));

    // Wait for table to become active
    System.out.println("Waiting for " + tableName + " to become ACTIVE...");
    try {
        Table table = dynamoDB.getTable(tableName);
        table.waitForActive();
    }
    catch (InterruptedException e) {
        e.printStackTrace();
    }
}

public static void query(String indexName) {

    Table table = dynamoDB.getTable(tableName);
```

```
\n");
        System.out.println("\n*****");
        System.out.println("Querying table " + tableName + "...");

        QuerySpec querySpec = new
QuerySpec().withConsistentRead(true).withScanIndexForward(true)
            .withReturnConsumedCapacity(ReturnConsumedCapacity.TOTAL);

        if (indexName == "IsOpenIndex") {

            System.out.println("\nUsing index: '" + indexName + "' Bob's orders that are
open.");
            System.out.println("Only a user-specified list of attributes are returned\n");
            Index index = table.getIndex(indexName);

            querySpec.withKeyConditionExpression("CustomerId = :v_custid and IsOpen
= :v_isopen")
                .WithValueMap(new ValueMap().withString(":v_custid",
"bob@example.com").withNumber(":v_isopen", 1));

            querySpec.withProjectionExpression("OrderCreationDate, ProductCategory,
ProductName, OrderStatus");

            ItemCollection<QueryOutcome> items = index.query(querySpec);
            Iterator<Item> iterator = items.iterator();

            System.out.println("Query: printing results...");

            while (iterator.hasNext()) {
                System.out.println(iterator.next().toJSONPretty());
            }

        }
        else if (indexName == "OrderCreationDateIndex") {
            System.out.println("\nUsing index: '" + indexName + "' Bob's orders that were
placed after 01/31/2015.");
            System.out.println("Only the projected attributes are returned\n");
            Index index = table.getIndex(indexName);

            querySpec.withKeyConditionExpression("CustomerId = :v_custid and
OrderCreationDate >= :v_orddate")
                .WithValueMap(
                    new ValueMap().withString(":v_custid",
"bob@example.com").withNumber(":v_orddate", 20150131));

            querySpec.withSelect(Select.ALL_PROJECTED_ATTRIBUTES);

            ItemCollection<QueryOutcome> items = index.query(querySpec);
            Iterator<Item> iterator = items.iterator();

            System.out.println("Query: printing results...");

            while (iterator.hasNext()) {
                System.out.println(iterator.next().toJSONPretty());
            }

        }
        else {
            System.out.println("\nNo index: All of Bob's orders, by OrderId:\n");

            querySpec.withKeyConditionExpression("CustomerId = :v_custid")
                .WithValueMap(new ValueMap().withString(":v_custid", "bob@example.com"));

            ItemCollection<QueryOutcome> items = table.query(querySpec);
            Iterator<Item> iterator = items.iterator();
```

```
        System.out.println("Query: printing results...");

        while (iterator.hasNext()) {
            System.out.println(iterator.next().toJSONPretty());
        }

    }

}

public static void deleteTable(String tableName) {

    Table table = dynamoDB.getTable(tableName);
    System.out.println("Deleting table " + tableName + "...");
    table.delete();

    // Wait for table to be deleted
    System.out.println("Waiting for " + tableName + " to be deleted...");
    try {
        table.waitForDelete();
    }
    catch (InterruptedException e) {
        e.printStackTrace();
    }
}

public static void loadData() {

    Table table = dynamoDB.getTable(tableName);

    System.out.println("Loading data into table " + tableName + "...");

    Item item = new Item().withPrimaryKey("CustomerId",
"alice@example.com").withNumber("OrderId", 1)
    .withNumber("IsOpen", 1).withNumber("OrderCreationDate",
20150101).withString("ProductCategory", "Book")
    .withString("ProductName", "The Great Outdoors").withString("OrderStatus",
"PACKING ITEMS");
    // no ShipmentTrackingId attribute

    PutItemOutcome putItemOutcome = table.putItem(item);

    item = new Item().withPrimaryKey("CustomerId",
"alice@example.com").withNumber("OrderId", 2)
    .withNumber("IsOpen", 1).withNumber("OrderCreationDate",
20150221).withString("ProductCategory", "Bike")
    .withString("ProductName", "Super Mountain").withString("OrderStatus", "ORDER
RECEIVED");
    // no ShipmentTrackingId attribute

    putItemOutcome = table.putItem(item);

    item = new Item().withPrimaryKey("CustomerId",
"alice@example.com").withNumber("OrderId", 3)
    // no IsOpen attribute
    .withNumber("OrderCreationDate", 20150304).withString("ProductCategory",
"Music")
    .withString("ProductName", "A Quiet Interlude").withString("OrderStatus", "IN
TRANSIT")
    .withString("ShipmentTrackingId", "176493");

    putItemOutcome = table.putItem(item);

    item = new Item().withPrimaryKey("CustomerId",
"bob@example.com").withNumber("OrderId", 1)
    // no IsOpen attribute
```

```
        .withNumber("OrderCreationDate", 20150111).withString("ProductCategory",
"Movie")
        .withString("ProductName", "Calm Before The Storm").withString("OrderStatus",
"SHIPPING DELAY")
        .withString("ShipmentTrackingId", "859323");

    putItemOutcome = table.putItem(item);

    item = new Item().withPrimaryKey("CustomerId",
"bob@example.com").withNumber("OrderId", 2)
        // no IsOpen attribute
        .withNumber("OrderCreationDate", 20150124).withString("ProductCategory",
"Music")
        .withString("ProductName", "E-Z Listening").withString("OrderStatus",
"DELIVERED")
        .withString("ShipmentTrackingId", "756943");

    putItemOutcome = table.putItem(item);

    item = new Item().withPrimaryKey("CustomerId",
"bob@example.com").withNumber("OrderId", 3)
        // no IsOpen attribute
        .withNumber("OrderCreationDate", 20150221).withString("ProductCategory",
"Music")
        .withString("ProductName", "Symphony 9").withString("OrderStatus", "DELIVERED")
        .withString("ShipmentTrackingId", "645193");

    putItemOutcome = table.putItem(item);

    item = new Item().withPrimaryKey("CustomerId",
"bob@example.com").withNumber("OrderId", 4)
        .withNumber("IsOpen", 1).withNumber("OrderCreationDate",
20150222).withString("ProductCategory", "Hardware")
        .withString("ProductName", "Extra Heavy Hammer").withString("OrderStatus",
"PACKING ITEMS");
        // no ShipmentTrackingId attribute

    putItemOutcome = table.putItem(item);

    item = new Item().withPrimaryKey("CustomerId",
"bob@example.com").withNumber("OrderId", 5)
        /* no IsOpen attribute */
        .withNumber("OrderCreationDate", 20150309).withString("ProductCategory",
"Book")
        .withString("ProductName", "How To Cook").withString("OrderStatus", "IN
TRANSIT")
        .withString("ShipmentTrackingId", "440185");

    putItemOutcome = table.putItem(item);

    item = new Item().withPrimaryKey("CustomerId",
"bob@example.com").withNumber("OrderId", 6)
        // no IsOpen attribute
        .withNumber("OrderCreationDate", 20150318).withString("ProductCategory",
"Luggage")
        .withString("ProductName", "Really Big Suitcase").withString("OrderStatus",
"DELIVERED")
        .withString("ShipmentTrackingId", "893927");

    putItemOutcome = table.putItem(item);

    item = new Item().withPrimaryKey("CustomerId",
"bob@example.com").withNumber("OrderId", 7)
        /* no IsOpen attribute */
        .withNumber("OrderCreationDate", 20150324).withString("ProductCategory",
"Golf")
```

```
        .withString("ProductName", "PGA Pro II").withString("OrderStatus", "OUT FOR
DELIVERY")
        .withString("ShipmentTrackingId", "383283");

    putItemOutcome = table.putItem(item);
    assert putItemOutcome != null;
}

}
```

Como trabalhar com índices secundários locais: .NET

Tópicos

- [Criar uma tabela de com um índice secundário local \(p. 625\)](#)
- [Descreve uma tabela de com um índice secundário local \(p. 627\)](#)
- [Consultar um índice secundário local \(p. 628\)](#)
- [Exemplo: Índices secundários locais que usam oAWS SDK for .NETAPI de baixo nível \(p. 629\)](#)

Você pode usar o comandoAWS SDK for .NETAPI de baixo nível para criar uma tabela do Amazon DynamoDB com um ou mais índices secundários locais na tabela e executar consultas usando os índices. Essas operações são mapeadas para as ações de API do DynamoDB de baixo nível do correspondentes. Para mais informações, consulte [Exemplos de código .NET \(p. 336\)](#).

Veja a seguir as etapas comuns para operações de tabela usando a API de baixo nível do .NET.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Forneça os parâmetros obrigatórios e opcionais para a operação, criando os objetos de solicitação correspondentes.

Por exemplo, crie um objeto `CreateTableRequest` para criar uma tabela e crie um objeto `QueryRequest` para consultar uma tabela ou um índice.

3. Execute o método apropriado fornecido pelo cliente que você criou na etapa anterior.

Criar uma tabela de com um índice secundário local

Os índices secundários locais devem ser criados ao mesmo tempo em que você cria uma tabela. Para isso, use o`CreateTable` forneca suas especificações para um ou mais índices secundários locais. O código de exemplo C# a seguir cria uma tabela para armazenar informações sobre músicas em uma coleção de músicas. A chave de partição é `Artist` e a chave de classificação é `SongTitle`. Índice secundário, `AlbumTitleIndex`, facilita consultas por título de álbum.

Veja a seguir as etapas para criar uma tabela com um índice secundário local usando a API de baixo nível do .NET.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Crie uma instância da classe `CreateTableRequest` para fornecer as informações solicitadas.

Você deve fornecer o nome da tabela, sua chave primária e os valores de throughput provisionado. Para o índice secundário local, você deve fornecer o nome do índice, o nome e o tipo de dados da chave de classificação do índice, o esquema de chave do índice e a projeção do atributo.

3. Execute a`CreateTable`Método fornecendo o objeto de solicitação como um parâmetro.

O exemplo de código C# a seguir demonstra as etapas anteriores. O código cria uma tabela (`Music`) com um índice secundário `noAlbumTitleAttribute`. A chave de partição de tabela e a chave de classificação, bem como a chave de classificação de índice, são os únicos atributos projetados para o índice.

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "Music";

CreateTableRequest createTableRequest = new CreateTableRequest()
{
    TableName = tableName
};

//ProvisionedThroughput
createTableRequest.ProvisionedThroughput = new ProvisionedThroughput()
{
    ReadCapacityUnits = (long)5,
    WriteCapacityUnits = (long)5
};

//AttributeDefinitions
List<AttributeDefinition> attributeDefinitions = new List<AttributeDefinition>();

attributeDefinitions.Add(new AttributeDefinition()
{
   AttributeName = "Artist",
   AttributeType = "S"
});

attributeDefinitions.Add(new AttributeDefinition()
{
   AttributeName = "SongTitle",
   AttributeType = "S"
});

attributeDefinitions.Add(new AttributeDefinition()
{
   AttributeName = "AlbumTitle",
   AttributeType = "S"
});

createTableRequest.AttributeDefinitions = attributeDefinitions;

//KeySchema
List<KeySchemaElement> tableKeySchema = new List<KeySchemaElement>();

tableKeySchema.Add(new KeySchemaElement() { AttributeName = "Artist", KeyType = "HASH" });
//Partition key
tableKeySchema.Add(new KeySchemaElement() { AttributeName = "SongTitle", KeyType =
"RANGE" }); //Sort key

createTableRequest.KeySchema = tableKeySchema;

List<KeySchemaElement> indexKeySchema = new List<KeySchemaElement>();
indexKeySchema.Add(new KeySchemaElement() { AttributeName = "Artist", KeyType = "HASH" });
//Partition key
indexKeySchema.Add(new KeySchemaElement() { AttributeName = "AlbumTitle", KeyType =
"RANGE" }); //Sort key

Projection projection = new Projection() { ProjectionType = "INCLUDE" };

List<string> nonKeyAttributes = new List<string>();
nonKeyAttributes.Add("Genre");
nonKeyAttributes.Add("Year");
projection.NonKeyAttributes = nonKeyAttributes;
```

```
LocalSecondaryIndex localSecondaryIndex = new LocalSecondaryIndex()
{
    IndexName = "AlbumTitleIndex",
    KeySchema = indexKeySchema,
    Projection = projection
};

List<LocalSecondaryIndex> localSecondaryIndexes = new List<LocalSecondaryIndex>();
localSecondaryIndexes.Add(localSecondaryIndex);
createTableRequest.LocalSecondaryIndexes = localSecondaryIndexes;

CreateTableResponse result = client.CreateTable(createTableRequest);
Console.WriteLine(result.CreateTableResult.TableDescription.TableName);
Console.WriteLine(result.CreateTableResult.TableDescription.TableStatus);
```

Você deve aguardar até que o DynamoDB crie a tabela e defina o status da tabela para **ACTIVE**. Depois disso, você pode começar a inserir itens de dados na tabela.

Descreve uma tabela com um índice secundário local

Para acessar informações sobre índices secundários locais em uma tabela, use o `DescribeTableAPI`. Para cada índice, você pode acessar seu nome, esquema de chaves e atributos projetados.

Veja a seguir as etapas para acessar informações de índice secundário local em uma tabela usando a API de baixo nível .NET.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Crie uma instância da classe `DescribeTableRequest` para fornecer as informações solicitadas. Você deve fornecer o nome da tabela.
3. Execute a `DescribeTable` Método fornecendo o objeto de solicitação como um parâmetro.
- 4.

O exemplo de código C# a seguir demonstra as etapas anteriores.

Example

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "Music";

DescribeTableResponse response = client.DescribeTable(new DescribeTableRequest()
    { TableName = tableName });
List<LocalSecondaryIndexDescription> localSecondaryIndexes =
    response.DescribeTableResult.Table.LocalSecondaryIndexes;

// This code snippet will work for multiple indexes, even though
// there is only one index in this example.
foreach (LocalSecondaryIndexDescription lsiDescription in localSecondaryIndexes)
{
    Console.WriteLine("Info for index " + lsiDescription.IndexName + ":");

    foreach (KeySchemaElement kse in lsiDescription.KeySchema)
    {
        Console.WriteLine("\t" + kse.AttributeName + ": key type is " + kse.KeyType);
    }

    Projection projection = lsiDescription.Projection;

    Console.WriteLine("\tThe projection type is: " + projection.ProjectionType);
```

```
if (projection.ProjectionType.ToString().Equals("INCLUDE"))
{
    Console.WriteLine("\t\tThe non-key projected attributes are:");

    foreach (String s in projection.NonKeyAttributes)
    {
        Console.WriteLine("\t\t" + s);
    }
}
```

Consultar um índice secundário local

Você pode usar `QueryEm` um índice secundário local da forma semelhante ao `Query` uma tabela do. Você deve especificar o nome do índice, os critérios de consulta da chave de classificação do índice e os atributos que deseja retornar. Neste exemplo, o índice é `AlbumTitleIndex` a chave de classificação do índice é `AlbumTitle`.

Os únicos atributos retornados são aqueles que foram projetados no índice. É possível modificar essa consulta para selecionar atributos não chave também, mas isso exigiria atividades de busca de tabela que são relativamente caras. Para obter mais informações sobre buscas de tabela, consulte [Projeções de atributo \(p. 607\)](#)

Veja a seguir as etapas para consultar um índice secundário local usando a API de baixo nível do .NET.

1. Crie uma instância da classe `AmazonDynamoDBClient`.
2. Crie uma instância da classe `QueryRequest` para fornecer as informações solicitadas.
3. Execute a `query` método fornecendo o objeto de solicitação como um parâmetro.

O exemplo de código C# a seguir demonstra as etapas anteriores.

Example

```
QueryRequest queryRequest = new QueryRequest
{
    TableName = "Music",
    IndexName = "AlbumTitleIndex",
    Select = "ALL_ATTRIBUTES",
    ScanIndexForward = true,
    KeyConditionExpression = "Artist = :v_artist and AlbumTitle = :v_title",
    ExpressionAttributeValues = new Dictionary<string, AttributeValue>()
    {
        {"":v_artist",new AttributeValue {S = "Acme Band"}},
        {"":v_title",new AttributeValue {S = "Songs About Life"}}
    },
};

QueryResponse response = client.Query(queryRequest);

foreach (var attribs in response.Items)
{
    foreach (var attrib in attribs)
    {
        Console.WriteLine(attrib.Key + " ---> " + attrib.Value.S);
    }
    Console.WriteLine();
}
```

Exemplo: Índices secundários locais que usam o AWS SDK for .NET API de baixo nível

O código de exemplo C# a seguir mostra como trabalhar com índices secundários locais no Amazon DynamoDB. O exemplo cria uma tabela do denominada `CustomerOrders` com uma chave de partição `CustomerID` e uma chave de tipo `OrderID`. Há dois índices secundários locais nessa tabela:

- `OrderCreationDateIndex`— A chave de classificação é `OrderCreationDate`, e os seguintes atributos são projetados no índice:
 - `ProductCategory`
 - `ProductName`
 - `OrderStatus`
 - `ShipmentTrackingId`
- `IsOpenIndex`— A chave de classificação é `IsOpen`, e todos os atributos da tabela estão projetados no índice.

Após o `CustomerOrders` tabela é criada, o programa carrega a tabela com dados que representam pedidos de clientes. Ele consulta os dados usando os índices secundários locais. Por fim, o programa exclui a tabela `CustomerOrders`.

Para obter instruções passo a passo sobre como testar o exemplo a seguir, consulte [Exemplos de código .NET \(p. 336\)](#).

Example

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.DataModel;  
using Amazon.DynamoDBv2.DocumentModel;  
using Amazon.DynamoDBv2.Model;  
using Amazon.Runtime;  
using Amazon.SecurityToken;  
  
namespace com.amazonaws.codesamples  
{  
    class LowLevelLocalSecondaryIndexExample  
    {  
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();  
        private static string tableName = "CustomerOrders";  
  
        static void Main(string[] args)  
        {  
            try
```

```
{  
    CreateTable();  
    LoadData();  
  
    Query(null);  
    Query("IsOpenIndex");  
    Query("OrderCreationDateIndex");  
  
    DeleteTable(tableName);  
  
    Console.WriteLine("To continue, press Enter");  
    Console.ReadLine();  
}  
catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message); }  
catch (AmazonServiceException e) { Console.WriteLine(e.Message); }  
catch (Exception e) { Console.WriteLine(e.Message); }  
}  
  
private static void CreateTable()  
{  
    var createTableRequest =  
        new CreateTableRequest()  
    {  
        TableName = tableName,  
        ProvisionedThroughput =  
            new ProvisionedThroughput()  
        {  
            ReadCapacityUnits = (long)1,  
            WriteCapacityUnits = (long)1  
        }  
    };  
  
    var attributeDefinitions = new List<AttributeDefinition>()  
    {  
        // Attribute definitions for table primary key  
        { new AttributeDefinition() {  
            AttributeName = "CustomerId", AttributeType = "S"  
        } },  
        { new AttributeDefinition() {  
            AttributeName = "OrderId", AttributeType = "N"  
        } },  
        // Attribute definitions for index primary key  
        { new AttributeDefinition() {  
            AttributeName = "OrderCreationDate", AttributeType = "N"  
        } },  
        { new AttributeDefinition() {  
            AttributeName = "IsOpen", AttributeType = "N"  
        } }  
    };  
  
    createTableRequest.AttributeDefinitions = attributeDefinitions;  
  
    // Key schema for table  
    var tableKeySchema = new List<KeySchemaElement>()  
    {  
        { new KeySchemaElement() {  
            AttributeName = "CustomerId", KeyType = "HASH"  
        } },  
        { new KeySchemaElement() {  
            AttributeName = "OrderId", KeyType = "RANGE"  
        } }  
    };  
  
    createTableRequest.KeySchema = tableKeySchema;  
  
    var localSecondaryIndexes = new List<LocalSecondaryIndex>();  
}
```

```
// OrderCreationDateIndex
LocalSecondaryIndex orderCreationDateIndex = new LocalSecondaryIndex()
{
    IndexName = "OrderCreationDateIndex"
};

// Key schema for OrderCreationDateIndex
var indexKeySchema = new List<KeySchemaElement>()
{
    { new KeySchemaElement() {
        AttributeName = "CustomerId", KeyType = "HASH"
    } },
    { new KeySchemaElement() {
        AttributeName = "OrderCreationDate", KeyType = "RANGE"
    } } //Partition key
};

orderCreationDateIndex.KeySchema = indexKeySchema;

// Projection (with list of projected attributes) for
// OrderCreationDateIndex
var projection = new Projection()
{
    ProjectionType = "INCLUDE"
};

var nonKeyAttributes = new List<string>()
{
    "ProductCategory",
    "ProductName"
};
projection.NonKeyAttributes = nonKeyAttributes;

orderCreationDateIndex.Projection = projection;

localSecondaryIndexes.Add(orderCreationDateIndex);

// IsOpenIndex
LocalSecondaryIndex isOpenIndex
    = new LocalSecondaryIndex()
    {
        IndexName = "IsOpenIndex"
    };

// Key schema for IsOpenIndex
indexKeySchema = new List<KeySchemaElement>()
{
    { new KeySchemaElement() {
        AttributeName = "CustomerId", KeyType = "HASH"
    } },
    { new KeySchemaElement() {
        AttributeName = "IsOpen", KeyType = "RANGE"
    } } //Partition key
};

// Projection (all attributes) for IsOpenIndex
projection = new Projection()
{
    ProjectionType = "ALL"
};

isOpenIndex.KeySchema = indexKeySchema;
isOpenIndex.Projection = projection;

localSecondaryIndexes.Add(isOpenIndex);
```

```
// Add index definitions to CreateTable request
createTableRequest.LocalSecondaryIndexes = localSecondaryIndexes;

Console.WriteLine("Creating table " + tableName + "...");
client.CreateTable(createTableRequest);
WaitUntilTableReady(tableName);
}

public static void Query(string indexName)
{

Console.WriteLine("\n*****\n");
Console.WriteLine("Querying table " + tableName + "...");

QueryRequest queryRequest = new QueryRequest()
{
    TableName = tableName,
    ConsistentRead = true,
    ScanIndexForward = true,
    ReturnConsumedCapacity = "TOTAL"
};

String keyConditionExpression = "CustomerId = :v_customerId";
Dictionary<string, AttributeValue> expressionAttributeValues = new
Dictionary<string, AttributeValue> {
    {":v_customerId", new AttributeValue {
        S = "bob@example.com"
    }}
};

if (indexName == "IsOpenIndex")
{
    Console.WriteLine("\nUsing index: '" + indexName
        + "' Bob's orders that are open.");
    Console.WriteLine("Only a user-specified list of attributes are returned
\n");
    queryRequest.IndexName = indexName;

    keyConditionExpression += " and IsOpen = :v_isOpen";
    expressionAttributeValues.Add(":v_isOpen", new AttributeValue
    {
        N = "1"
    });

    // ProjectionExpression
    queryRequest.ProjectionExpression = "OrderCreationDate, ProductCategory,
ProductName, OrderStatus";
}
else if (indexName == "OrderCreationDateIndex")
{
    Console.WriteLine("\nUsing index: '" + indexName
        + "' Bob's orders that were placed after 01/31/2013.");
    Console.WriteLine("Only the projected attributes are returned\n");
    queryRequest.IndexName = indexName;

    keyConditionExpression += " and OrderCreationDate > :v_Date";
    expressionAttributeValues.Add(":v_Date", new AttributeValue
    {
        N = "20130131"
    });

    // Select
    queryRequest.Select = "ALL_PROJECTED_ATTRIBUTES";
}
```

```
        }
    else
    {
        Console.WriteLine("\nNo index: All of Bob's orders, by OrderId:\n");
    }
queryRequest.KeyConditionExpression = keyConditionExpression;
queryRequest.ExpressionAttributeValues = expressionAttributeValues;

var result = client.Query(queryRequest);
var items = result.Items;
foreach (var currentItem in items)
{
    foreach (string attr in currentItem.Keys)
    {
        if (attr == "OrderId" || attr == "IsOpen"
            || attr == "OrderCreationDate")
        {
            Console.WriteLine(attr + "---> " + currentItem[attr].N);
        }
        else
        {
            Console.WriteLine(attr + "---> " + currentItem[attr].S);
        }
    }
    Console.WriteLine();
}
Console.WriteLine("\nConsumed capacity: " +
result.ConsumedCapacity.CapacityUnits + "\n");
}

private static void DeleteTable(string tableName)
{
    Console.WriteLine("Deleting table " + tableName + "...");
    client.DeleteTable(new DeleteTableRequest()
    {
        TableName = tableName
    });
    WaitForTableToDelete(tableName);
}

public static void LoadData()
{
    Console.WriteLine("Loading data into table " + tableName + "...");

    Dictionary<string, AttributeValue> item = new Dictionary<string,
    AttributeValue>();

    item["CustomerId"] = new AttributeValue
    {
        S = "alice@example.com"
    };
    item["OrderId"] = new AttributeValue
    {
        N = "1"
    };
    item["IsOpen"] = new AttributeValue
    {
        N = "1"
    };
    item["OrderCreationDate"] = new AttributeValue
    {
        N = "20130101"
    };
    item["ProductCategory"] = new AttributeValue
    {
        S = "Book"
    };
}
```

```
};

item["ProductName"] = new AttributeValue
{
    S = "The Great Outdoors"
};
item["OrderStatus"] = new AttributeValue
{
    S = "PACKING ITEMS"
};
/* no ShipmentTrackingId attribute */
PutItemRequest putItemRequest = new PutItemRequest
{
    TableName = tableName,
    Item = item,
    ReturnItemCollectionMetrics = "SIZE"
};
client.PutItem(putItemRequest);

item = new Dictionary<string, AttributeValue>();
item["CustomerId"] = new AttributeValue
{
    S = "alice@example.com"
};
item["OrderId"] = new AttributeValue
{
    N = "2"
};
item["IsOpen"] = new AttributeValue
{
    N = "1"
};
item["OrderCreationDate"] = new AttributeValue
{
    N = "20130221"
};
item["ProductCategory"] = new AttributeValue
{
    S = "Bike"
};
item["ProductName"] = new AttributeValue
{
    S = "Super Mountain"
};
item["OrderStatus"] = new AttributeValue
{
    S = "ORDER RECEIVED"
};
/* no ShipmentTrackingId attribute */
putItemRequest = new PutItemRequest
{
    TableName = tableName,
    Item = item,
    ReturnItemCollectionMetrics = "SIZE"
};
client.PutItem(putItemRequest);

item = new Dictionary<string, AttributeValue>();
item["CustomerId"] = new AttributeValue
{
    S = "alice@example.com"
};
item["OrderId"] = new AttributeValue
{
    N = "3"
};
/* no IsOpen attribute */
```

```
item["OrderCreationDate"] = new AttributeValue
{
    N = "20130304"
};
item["ProductCategory"] = new AttributeValue
{
    S = "Music"
};
item["ProductName"] = new AttributeValue
{
    S = "A Quiet Interlude"
};
item["OrderStatus"] = new AttributeValue
{
    S = "IN TRANSIT"
};
item["ShipmentTrackingId"] = new AttributeValue
{
    S = "176493"
};
putItemRequest = new PutItemRequest
{
    TableName = tableName,
    Item = item,
    ReturnItemCollectionMetrics = "SIZE"
};
client.PutItem(putItemRequest);

item = new Dictionary<string, AttributeValue>();
item["CustomerId"] = new AttributeValue
{
    S = "bob@example.com"
};
item["OrderId"] = new AttributeValue
{
    N = "1"
};
/* no IsOpen attribute */
item["OrderCreationDate"] = new AttributeValue
{
    N = "20130111"
};
item["ProductCategory"] = new AttributeValue
{
    S = "Movie"
};
item["ProductName"] = new AttributeValue
{
    S = "Calm Before The Storm"
};
item["OrderStatus"] = new AttributeValue
{
    S = "SHIPPING DELAY"
};
item["ShipmentTrackingId"] = new AttributeValue
{
    S = "859323"
};
putItemRequest = new PutItemRequest
{
    TableName = tableName,
    Item = item,
    ReturnItemCollectionMetrics = "SIZE"
};
client.PutItem(putItemRequest);
```

```
item = new Dictionary<string, AttributeValue>();
item["CustomerId"] = new AttributeValue
{
    S = "bob@example.com"
};
item["OrderId"] = new AttributeValue
{
    N = "2"
};
/* no IsOpen attribute */
item["OrderCreationDate"] = new AttributeValue
{
    N = "20130124"
};
item["ProductCategory"] = new AttributeValue
{
    S = "Music"
};
item["ProductName"] = new AttributeValue
{
    S = "E-Z Listening"
};
item["OrderStatus"] = new AttributeValue
{
    S = "DELIVERED"
};
item["ShipmentTrackingId"] = new AttributeValue
{
    S = "756943"
};
putItemRequest = new PutItemRequest
{
    TableName = tableName,
    Item = item,
    ReturnItemCollectionMetrics = "SIZE"
};
client.PutItem(putItemRequest);

item = new Dictionary<string, AttributeValue>();
item["CustomerId"] = new AttributeValue
{
    S = "bob@example.com"
};
item["OrderId"] = new AttributeValue
{
    N = "3"
};
/* no IsOpen attribute */
item["OrderCreationDate"] = new AttributeValue
{
    N = "20130221"
};
item["ProductCategory"] = new AttributeValue
{
    S = "Music"
};
item["ProductName"] = new AttributeValue
{
    S = "Symphony 9"
};
item["OrderStatus"] = new AttributeValue
{
    S = "DELIVERED"
};
item["ShipmentTrackingId"] = new AttributeValue
{
```

```
        S = "645193"
    };
putItemRequest = new PutItemRequest
{
    TableName = tableName,
    Item = item,
    ReturnItemCollectionMetrics = "SIZE"
};
client.PutItem(putItemRequest);

item = new Dictionary<string, AttributeValue>();
item["CustomerId"] = new AttributeValue
{
    S = "bob@example.com"
};
item["OrderId"] = new AttributeValue
{
    N = "4"
};
item["IsOpen"] = new AttributeValue
{
    N = "1"
};
item["OrderCreationDate"] = new AttributeValue
{
    N = "20130222"
};
item["ProductCategory"] = new AttributeValue
{
    S = "Hardware"
};
item["ProductName"] = new AttributeValue
{
    S = "Extra Heavy Hammer"
};
item["OrderStatus"] = new AttributeValue
{
    S = "PACKING ITEMS"
};
/* no ShipmentTrackingId attribute */
putItemRequest = new PutItemRequest
{
    TableName = tableName,
    Item = item,
    ReturnItemCollectionMetrics = "SIZE"
};
client.PutItem(putItemRequest);

item = new Dictionary<string, AttributeValue>();
item["CustomerId"] = new AttributeValue
{
    S = "bob@example.com"
};
item["OrderId"] = new AttributeValue
{
    N = "5"
};
/* no IsOpen attribute */
item["OrderCreationDate"] = new AttributeValue
{
    N = "20130309"
};
item["ProductCategory"] = new AttributeValue
{
    S = "Book"
};
```

```
item["ProductName"] = new AttributeValue
{
    S = "How To Cook"
};
item["OrderStatus"] = new AttributeValue
{
    S = "IN TRANSIT"
};
item["ShipmentTrackingId"] = new AttributeValue
{
    S = "440185"
};
putItemRequest = new PutItemRequest
{
    TableName = tableName,
    Item = item,
    ReturnItemCollectionMetrics = "SIZE"
};
client.PutItem(putItemRequest);

item = new Dictionary<string, AttributeValue>();
item["CustomerId"] = new AttributeValue
{
    S = "bob@example.com"
};
item["OrderId"] = new AttributeValue
{
    N = "6"
};
/* no IsOpen attribute */
item["OrderCreationDate"] = new AttributeValue
{
    N = "20130318"
};
item["ProductCategory"] = new AttributeValue
{
    S = "Luggage"
};
item["ProductName"] = new AttributeValue
{
    S = "Really Big Suitcase"
};
item["OrderStatus"] = new AttributeValue
{
    S = "DELIVERED"
};
item["ShipmentTrackingId"] = new AttributeValue
{
    S = "893927"
};
putItemRequest = new PutItemRequest
{
    TableName = tableName,
    Item = item,
    ReturnItemCollectionMetrics = "SIZE"
};
client.PutItem(putItemRequest);

item = new Dictionary<string, AttributeValue>();
item["CustomerId"] = new AttributeValue
{
    S = "bob@example.com"
};
item["OrderId"] = new AttributeValue
{
    N = "7"
};
```

```
};

/* no IsOpen attribute */
item["OrderCreationDate"] = new AttributeValue
{
    N = "20130324"
};
item["ProductCategory"] = new AttributeValue
{
    S = "Golf"
};
item["ProductName"] = new AttributeValue
{
    S = "PGA Pro II"
};
item["OrderStatus"] = new AttributeValue
{
    S = "OUT FOR DELIVERY"
};
item["ShipmentTrackingId"] = new AttributeValue
{
    S = "383283"
};
putItemRequest = new PutItemRequest
{
    TableName = tableName,
    Item = item,
    ReturnItemCollectionMetrics = "SIZE"
};
client.PutItem(putItemRequest);
}

private static void WaitUntilTableReady(string tableName)
{
    string status = null;
    // Let us wait until table is created. Call DescribeTable.
    do
    {
        System.Threading.Thread.Sleep(5000); // Wait 5 seconds.
        try
        {
            var res = client.DescribeTable(new DescribeTableRequest
            {
                TableName = tableName
            });

            Console.WriteLine("Table name: {0}, status: {1}",
                res.Table.TableName,
                res.Table.TableStatus);
            status = res.Table.TableStatus;
        }
        catch (ResourceNotFoundException)
        {
            // DescribeTable is eventually consistent. So you might
            // get resource not found. So we handle the potential exception.
        }
    } while (status != "ACTIVE");
}

private static void WaitForTableToDelete(string tableName)
{
    bool tablePresent = true;

    while (tablePresent)
    {
        System.Threading.Thread.Sleep(5000); // Wait 5 seconds.
        try
```

```
{  
    var res = client.DescribeTable(new DescribeTableRequest  
    {  
        TableName = tableName  
    });  
  
    Console.WriteLine("Table name: {0}, status: {1}",  
        res.Table.TableName,  
        res.Table.TableStatus);  
}  
catch (ResourceNotFoundException)  
{  
    tablePresent = false;  
}  
}  
}  
}  
}
```

Como trabalhar com índices secundários locais:AWS CLI

Você pode usar o comando AWS CLI Para criar uma tabela do Amazon DynamoDB com um ou mais índices secundários locais na tabela e executar consultas usando os índices.

Tópicos

- [Criar uma tabela de com um índice secundário local \(p. 640\)](#)
- [Descreve uma tabela de com um índice secundário local \(p. 641\)](#)
- [Consultar um índice secundário local \(p. 641\)](#)

Criar uma tabela de com um índice secundário local

Índices secundários locais devem ser criados ao mesmo tempo em que uma tabela é criada. Para isso, use o comando `create-table` e forneça suas especificações para um ou mais índices secundários locais. O exemplo a seguir cria uma tabela (`Music`) para armazenar informações sobre músicas em uma coleção de músicas. A chave de partição é `Artist` e a chave de classificação é `SongTitle`. Índice secundário, `AlbumTitleIndex` no `AlbumTitle` O atributo facilita consultas por título de álbum.

```
aws dynamodb create-table \  
    --table-name Music \  
    --attribute-definitions AttributeName=Artist,AttributeType=S  
    AttributeName=SongTitle,AttributeType=S \  
    AttributeName=AlbumTitle,AttributeType=S \  
    --key-schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE \  
    --provisioned-throughput \  
        ReadCapacityUnits=10,WriteCapacityUnits=5 \  
    --local-secondary-indexes \  
        "[{"IndexName": "AlbumTitleIndex",  
        "KeySchema": [{"AttributeName": "Artist", "KeyType": "HASH"},  
                    {"AttributeName": "AlbumTitle", "KeyType": "RANGE"}],  
        "Projection": {"ProjectionType": "INCLUDE", "NonKeyAttributes": ["Genre",  
                           "Year"]}]}"
```

Você deve aguardar até que o DynamoDB crie a tabela e defina o status da tabela para `ACTIVE`. Depois disso, você pode começar a inserir itens de dados na tabela. Você pode usar `describe-table` Para determinar o status da criação da tabela.

Descreve uma tabela de com um índice secundário local

Para acessar informações sobre índices secundários locais em uma tabela, use o `describe-table` parâmetro. Para cada índice, você pode acessar seu nome, esquema de chaves e atributos projetados.

```
aws dynamodb describe-table --table-name Music
```

Consultar um índice secundário local

Você pode usar o comando `query` operação em um índice secundário local da mesma forma que você `query` uma tabela do. Você deve especificar o nome do índice, os critérios de consulta da chave de classificação do índice e os atributos que deseja retornar. Neste exemplo, o índice é `AlbumTitleIndex` e a chave de classificação do índice é `AlbumTitle`.

Os únicos atributos retornados são aqueles que foram projetados no índice. É possível modificar essa consulta para selecionar atributos não chave também, mas isso exigiria atividades de busca de tabela que são relativamente caras. Para obter mais informações sobre buscas de tabela, consulte [Projeções de atributo \(p. 607\)](#).

```
aws dynamodb query \  
    --table-name Music \  
    --index-name AlbumTitleIndex \  
    --key-condition-expression "Artist = :v_artist and AlbumTitle = :v_title" \  
    --expression-attribute-values '{":v_artist":{"S":"Acme Band"},":v_title":{"S":"Songs About Life"} }'
```

Captura de dados de alteração com o Amazon DynamoDB

Muitos aplicativos se beneficiam da captura de alterações nos itens armazenados em uma tabela do DynamoDB no momento em que elas ocorrem. Veja a seguir alguns exemplos de casos de uso:

- Um aplicativo móvel popular modifica os dados em uma tabela do DynamoDB, a uma taxa de milhares de atualizações por segundo. Outro aplicativo captura e armazena dados sobre essas atualizações, fornecendo métricas de uso quase em tempo real para o aplicativo móvel.
- Um aplicativo financeiro modifica dados do mercado de ações em uma tabela do DynamoDB. Diferentes aplicativos executados em paralelo acompanham essas mudanças em tempo real, calculam o valor em risco e reequilibram automaticamente as carteiras com base nos movimentos dos preços das ações.
- Sensores em veículos de transporte e equipamentos industriais enviam dados para uma tabela do DynamoDB. Diferentes aplicativos monitoram o desempenho e enviam alertas de mensagens quando um problema é detectado, prevêem possíveis defeitos aplicando algoritmos de aprendizado de máquina e compactam e arquivam dados no Amazon Simple Storage Service (Amazon S3).
- Um aplicativo envia notificações automaticamente aos dispositivos móveis de todos os jogadores de um grupo assim que um deles carrega uma nova imagem.
- Um novo cliente adiciona dados a uma tabela do DynamoDB. Esse evento invoca outro aplicativo que envia um e-mail de boas-vindas ao novo cliente.

O DynamoDB oferece suporte a streaming de registros de captura de dados de alteração em nível de item em tempo quase real. Você pode criar aplicativos que consomem esses streams e executam ações com base no conteúdo.

Tópicos

- [Opções de streaming para a captura de dados de alteração \(p. 642\)](#)
- [Alterar captura de dados para Kinesis Data Streams com o DynamoDB \(p. 643\)](#)
- [Captura de dados de alteração para DynamoDB Streams \(p. 651\)](#)

Opções de streaming para a captura de dados de alteração

O DynamoDB oferece dois modelos de streaming para captura de dados de alteração: Kinesis Data Streams para DynamoDB e DynamoDB Streams.

Para ajudá-lo a escolher a solução certa para seu aplicativo, a tabela a seguir resume os recursos de cada modelo de streaming.

Properties	Kinesis Data Streams para DynamoDB	DynamoDB Streams
Retenção de dados	Até 1 ano .	24 horas.
Suporte à Kinesis Client Library (KCL)	Suporte Versões KCL 1.X e 2.X .	Suporte KCL versão 1.X .
Número de consumidores	Até 5 simultâneos consumidores por fragmento, ou até 20 consumidores simultâneos por fragmento com fan-out aprimorado .	Até 2 simultâneos Consumidores por estilhaço.
Cotas do throughput	Ilimitado.	Sujeito à taxa de transferência Cotas dopor tabela do DynamoDB e AWSRegião :
Registro de modelo de entrega	Modelo Pull por HTTP usando o GetRecordse com fan-out aprimorado , o Kinesis Data Streams envia os registros por HTTP/2 usando SubscribeToShard .	Modelo Pull por HTTP usando GetRecords .
Ordenação dos registros	O atributo timestamp em cada registro de fluxo pode ser usado para identificar a ordem real na qual as alterações ocorreram na tabela do DynamoDB.	Para cada item modificado em uma tabela do DynamoDB, os registros de stream aparecem na mesma sequência que as modificações reais no item.
Registros duplicados	Os registros duplicados podem aparecer ocasionalmente no fluxo.	Nenhum registro duplicado aparece no fluxo.
Opções de processamento de stream	Processar registros de fluxo usando AWS Lambda , Kinesis Data Analytics , Kinesis Data Firehose , ou AWS GlueETL de streaming.	Processar registros de fluxo usando AWS Lambda ou DynamoDB Streams Kinesis Adapter .

Você pode habilitar ambos os modelos de streaming na mesma tabela do DynamoDB.

Alterar captura de dados para Kinesis Data Streams com o DynamoDB

O Kinesis Data Streams for DynamoDB captura modificações em nível de item em qualquer tabela do DynamoDB e as replica em um[Stream de dados do Kinesis](#)de sua escolha. Seus aplicativos podem acessar o fluxo de dados do Kinesis e visualizar as alterações no nível do item quase em tempo real.

O streaming de dados do DynamoDB para um stream de dados do Kinesis permite que você capture e armazene continuamente terabytes de dados por hora. O Kinesis Data Streams permite que você aproveite o tempo de retenção de dados mais longo, o recurso aprimorado de fan-out para mais de dois aplicativos de consumidor simultâneos e a transparência adicional de auditoria e segurança. O Kinesis Data Streams também oferece acesso a outros serviços do Kinesis, como[Amazon Kinesis Data Firehose](#)[Amazon Kinesis Data Analytics](#). Isso permite que você crie aplicativos para alimentar painéis em tempo real, gerar alertas, implementar definições de preço e de publicidade dinâmicas e realizar análises de dados sofisticadas, como a aplicação de algoritmos de aprendizado de máquina.

Note

O uso de Kinesis Data Streams para DynamoDB está sujeito a[Definição de preços do Kinesis Data Streams](#)para o fluxo de dados e[Definição de preços do DynamoDB](#)para a tabela de origem.

Como a captura de dados de alteração funciona para Kinesis Data Streams

O Amazon Kinesis Data Streams for Amazon DynamoDB opera de forma assíncrona e, portanto, não haverá impacto sobre o desempenho de uma tabela se um stream estiver habilitado. Sempre que os itens são criados, atualizados ou excluídos na tabela, o DynamoDB envia um registro de dados para o Kinesis. O registro contém informações sobre uma modificação de dados em um único item de uma tabela do DynamoDB. Especificamente, um registro de dados contém o atributo de chave primária do item modificado, junto com as imagens “antes” e “depois” do item modificado.

Os registros de dados são publicados quase em tempo real no seu fluxo de dados do Kinesis. Depois que eles são gravados no fluxo de dados do Kinesis, você pode ler os registros gravados como faria com qualquer outro registro de qualquer fluxo de dados do Kinesis. Por exemplo, você pode usar a Biblioteca de cliente do Kinesis, AWS Lambda ou chame a API do Kinesis Data Streams. Para obter mais informações, consulte[Lendo dados dos fluxos de dados do Amazon Kinesis](#)no Guia do desenvolvedor de Amazon Kinesis Data Streams.

Os registros de fluxo armazenados no fluxo de dados do Kinesis são criptografados em repouso. Para obter mais informações, consulte[Proteção de dados no Amazon Kinesis Data Streams](#).

Para cada item modificado em uma tabela do DynamoDB, os registros de stream de dados do Kinesis podem aparecer em uma sequência diferente da modificação real no item. Os registros de fluxo podem aparecer mais de uma vez no fluxo de dados do Kinesis. Você pode usar o atributo `timestamp``ApproximateCreationTime` em cada registro para identificar a ordem real na qual ocorreram modificações de item e para identificar registros duplicados no fluxo.

O DynamoDB cobra pela captura de dados de alteração para Kinesis Data Streams em unidades de captura de dados de alteração. O DynamoDB cobra uma unidade de captura de dados de alteração para cada gravação de item (até 1 KB). Gravações maiores que 1 KB exigem unidades de captura de dados de alteração adicionais. O DynamoDB calcula as unidades de captura de dados de alteração com base na maior das imagens “antes” e “depois” que constituem um registro de alteração, usando a mesma lógica que[consumo de unidade de capacidade para operações de gravação](#). Você não precisa provisionar

o throughput de capacidade para unidades de captura de dados de alteração, semelhante a como o DynamoDB [Sob demanda](#) funciona.

As seções a seguir descrevem os principais conceitos e o comportamento dos fluxos de dados do Amazon Kinesis para Amazon DynamoDB.

Tópicos

- [Ativação de Amazon Kinesis Data Streams para o Amazon DynamoDB \(p. 644\)](#)
- [Considerações de gerenciamento de estilhaços para Amazon Kinesis Data Streams \(p. 644\)](#)
- [Monitoramento de Amazon Kinesis Data Streams para o Amazon DynamoDB \(p. 645\)](#)

Ativação de Amazon Kinesis Data Streams para o Amazon DynamoDB

Você pode habilitar ou desabilitar o streaming para o Kinesis em uma tabela existente do DynamoDB usando o AWS Management Console, o AWSSDK ou o AWS Command Line Interface(AWS CLI).

- Você só pode ativar Amazon Kinesis Data Streams para streaming do Amazon DynamoDB para um stream de dados do Kinesis a partir do mesmo AWS Conta da e AWS Região como sua tabela.
- Você só pode ativar Amazon Kinesis Data Streams para streaming do Amazon DynamoDB de uma tabela do DynamoDB para um stream de dados do Kinesis.

Considerações de gerenciamento de estilhaços para Amazon Kinesis Data Streams

A [estilhaço](#) é a unidade de taxa de transferência básica de um stream de dados do Amazon Kinesis. Você precisa provisionar um número apropriado de fragmentos no fluxo de dados do Kinesis para acomodar os registros de captura de dados de alteração do DynamoDB.

Os valores de entrada a seguir ajudam a determinar o número de fragmentos necessários para o fluxo de dados do Kinesis suportar sua tabela do DynamoDB:

- O tamanho médio do registro da tabela do DynamoDB em bytes (`average_record_size_in_bytes`).
- O número máximo de operações de gravação executadas na tabela do DynamoDB por segundo. Isso inclui operações de criação, exclusão e atualização de dados executadas por seus aplicativos, bem como operações geradas automaticamente, como exclusões geradas por Tempo de Vida (`write_throughput`).
- A porcentagem de operações de atualização e substituição executadas na tabela em comparação com operações de criação ou exclusão (`percentage_of_updates`). As operações de atualização e substituição replicam as imagens antigas e novas do item modificado para o fluxo, gerando o dobro do tamanho do item do DynamoDB.

Você pode aproximar o número de estilhaços (`number_of_shards`) necessários para sua tabela do DynamoDB usando os valores de entrada na seguinte fórmula:

```
number_of_shards = ceiling( ((write_throughput * (1+percentage_of_updates) * average_record_size_in_bytes) /1024 /1024), 1)
```

Essa fórmula reflete apenas os fragmentos necessários para acomodar o throughput de streaming do DynamoDB. Ele não representa o número total de fragmentos necessários no fluxo de dados do Kinesis, como fragmentos necessários para dar suporte aos consumidores de fluxo de dados do Kinesis. Para saber mais sobre a determinação do tamanho de um stream de dados do Kinesis, consulte [Como determinar o tamanho inicial de um stream de dados do Kinesis](#).

Por exemplo, digamos que você tenha uma taxa de transferência máxima de 40 operações de gravação/segundo (`write_throughput`) com um tamanho médio de registro de 1285 bytes (`average_record_size_in_bytes`). Se 25% dessas operações de gravação forem operações de atualização (`percentage_of_updates`), então você precisará de 2 fragmentos (`number_of_shards`) para acomodar o throughput de streaming do DynamoDB: teto $((40 * (1+25) * 1285)/1024/1024)$, 1).

Monitoramento de Amazon Kinesis Data Streams para o Amazon DynamoDB

O DynamoDB fornece várias métricas do Amazon CloudWatch para ajudá-lo a monitorar a replicação da captura de dados de alteração para o Kinesis. Para obter uma lista completa das métricas do CloudWatch, consulte [Métricas e dimensões do DynamoDB \(p. 930\)](#).

Recomendamos que você monitore os seguintes itens durante a ativação do fluxo e na produção para determinar se o stream tem capacidade suficiente:

- `ThrottledPutRecordCount`: o número de registros que falharam ao replicar para o fluxo de dados do Kinesis devido à capacidade insuficiente de fluxo de dados do Kinesis. O `ThrottledPutRecordCount` deve permanecer o mais baixo possível, embora você possa experimentar algum controle durante picos excepcionais de uso. O DynamoDB tenta novamente colocar registros limitados no fluxo de dados do Kinesis, mas isso pode resultar em maior latência de replicação. Se você tiver uma limitação excessiva e regular, talvez seja necessário aumentar o número de fragmentos de stream do Kinesis proporcionalmente à taxa de transferência de gravação observada da tabela. Para saber mais sobre a determinação do tamanho de um stream de dados do Kinesis, consulte [Como determinar o tamanho inicial de um stream de dados do Kinesis](#).
- `AgeOfOldestUnreplicatedRecord`: o tempo decorrido desde a alteração de nível de item mais antiga ainda para replicar para o stream de dados do Kinesis apareceu na tabela do DynamoDB. Em operação normal, `AgeOfOldestUnreplicatedRecord` deve estar na ordem dos milissegundos. Esse número aumenta com base em tentativas de replicação malsucedidas. O DynamoDB tenta continuamente tentativas de replicação sem êxito quando há falha de replicação. Talvez seja necessário ajustar a capacidade do fluxo de dados do Kinesis para manter essa métrica o mais baixa possível.

Você pode criar alarmes do Amazon CloudWatch que enviam uma mensagem do Amazon Simple Notification Service (Amazon SNS) para notificação quando qualquer uma das métricas anteriores exceder um limite específico. Para mais informações, consulte [Criação de alarmes do CloudWatch para monitorar o DynamoDB \(p. 952\)](#).

Introdução ao Amazon Kinesis Data Streams para o Amazon Kinesis Data Streams para o Amazon DynamoDB

Esta seção descreve como usar Amazon Kinesis Data Streams para o Amazon DynamoDB com o console do Amazon DynamoDB, o AWS Command Line Interface(AWS CLI) e as APIs.

Todos esses exemplos usam o método `MusicTabela` do DynamoDB criada como parte da tabela [Conceitos básicos do DynamoDB](#) para "Hello, World!".

Console

1. Faça login no AWS Management Console e abra o console do Kinesis em <https://console.aws.amazon.com/kinesis/>.
2. Selecione Criar fluxo de dados e siga as instruções para criar um stream chamado `samplestream`.
3. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
4. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
5. Selecione o `MúsicaTabela` do.
6. Selecione o `Visão geralGuia` do.

7. Under Detalhes do stream de dados do Kinesis, escolha a opção Gerenciar streaming para o Kinesis.

Music Close

Overview Items Metrics Alarms Capacity Indexes Global Tables Backups

Recent alerts

No CloudWatch alarms have been triggered for this table.

Kinesis data stream details

Use Amazon Kinesis Data Streams for DynamoDB to capture item-level changes in your table as a Kinesis data stream. [Learn more](#)

Stream enabled No

Stream name -

Manage streaming to Kinesis

8. Selecione samplestream na lista suspensa.
9. Selecione o Habilitar Botão.

AWS CLI

1. Criar um stream do Kinesis chamado samplestream usando o comando `create-stream`.

```
aws kinesis create-stream --stream-name samplestream --shard-count 3
```

Consulte [Considerações de gerenciamento de estilhaços para Amazon Kinesis Data Streams \(p. 644\)](#). Antes de definir o número de fragmentos para o stream de dados do Kinesis.

2. Verifique se o stream do Kinesis está ativo e pronto para uso usando o comando `describe-stream`.

```
aws kinesis describe-stream --stream-name samplestream
```

3. Habilite o streaming do Kinesis na tabela do DynamoDB usando o `enable-kinesis-streaming-destination` Comando da. Substitua o valor de `stream-arn` pelo que foi retornado por `describe-stream` na etapa anterior.

```
aws dynamodb enable-kinesis-streaming-destination \
--table-name Music \
--stream-arn arn:aws:kinesis:us-west-2:123456789012:stream/samplestream
```

4. Verifique se o streaming do Kinesis está ativo na tabela usando o `DynamoDB describe-kinesis-streaming-destination` Comando da.

```
aws dynamodb describe-kinesis-streaming-destination --table-name Music
```

5. Grave dados na tabela do DynamoDB usando o método `put-item`, conforme descrito na seção [Guia do desenvolvedor do DynamoDB](#).

```
aws dynamodb put-item \
--table-name Music \
```

```
--item \
'{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Call Me Today"}, "AlbumTitle": {"S": "Somewhat Famous"}, "Awards": {"N": "1"}}

aws dynamodb put-item \
--table-name Music \
--item \
'{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}, "AlbumTitle": {"S": "Songs About Life"}, "Awards": {"N": "10"} }'
```

6. Usar o Kinesis[get-records](#)Comando da CLI para recuperar o conteúdo do stream do Kinesis. Em seguida, use o seguinte trecho de código para desserializar o conteúdo do fluxo.

```
/** 
 * Takes as input a Record fetched from Kinesis and does arbitrary processing as an example.
 */
public void processRecord(Record kinesisRecord) throws IOException {
    ByteBuffer kdsRecordByteBuffer = kinesisRecord.getData();
    JsonNode rootNode = OBJECT_MAPPER.readTree(kdsRecordByteBuffer.array());
    JsonNode dynamoDBRecord = rootNode.get("dynamodb");
    JsonNode oldItemImage = dynamoDBRecord.get("OldImage");
    JsonNode newItemImage = dynamoDBRecord.get("NewImage");
    Instant recordTimestamp = fetchTimestamp(dynamoDBRecord);

    /**
     * Say for example our record contains a String attribute named "stringName" and we want to fetch the value
     * of this attribute from the new item image. The following code fetches this value.
     */
    JsonNode attributeNode = newItemImage.get("stringName");
    JsonNode attributeValueNode = attributeNode.get("S"); // Using DynamoDB "S" type attribute
    String attributeValue = attributeValueNode.textValue();
    System.out.println(attributeValue);
}

private Instant fetchTimestamp(JsonNode dynamoDBRecord) {
    JsonNode timestampJson = dynamoDBRecord.get("ApproximateCreationDateTime");
    return Instant.ofEpochMilli(timestampJson.longValue());
}
```

Java

1. Siga as instruções no guia do desenvolvedor do Kinesis Data Streams para [create](#)um stream de dados do Kinesis chamados [sample](#)stream usando Java.

Consulte[Considerações de gerenciamento de estilhaços para Amazon Kinesis Data Streams \(p. 644\)](#)Antes de definir o número de fragmentos para o stream de dados do Kinesis.

2. Use o snippet de código abaixo habilite o Amazon Kinesis Data Streams para Amazon DynamoDB na tabela

```
EnableKinesisStreamingDestinationRequest enableKdsRequest =
    EnableKinesisStreamingDestinationRequest.builder()
        .tableName(tableName)
        .streamArn(kdsArn)
        .build();

EnableKinesisStreamingDestinationResponse enableKdsResponse =
    ddbClient.enableKinesisStreamingDestination(enableKdsRequest);
```

3. Siga as instruções no guia do desenvolvedor do Kinesis Data Streams para [readDo stream de dados criado](#).
4. Use o trecho de código abaixo para cancelar a serialização do conteúdo do fluxo

```
/**  
 * Takes as input a Record fetched from Kinesis and does arbitrary processing as an  
example.  
 */  
public void processRecord(Record kinesisRecord) throws IOException {  
    ByteBuffer kdsRecordByteBuffer = kinesisRecord.getData();  
    JsonNode rootNode = OBJECT_MAPPER.readTree(kdsRecordByteBuffer.array());  
    JsonNode dynamoDBRecord = rootNode.get("dynamodb");  
    JsonNode oldItemImage = dynamoDBRecord.get("OldImage");  
    JsonNode newItemImage = dynamoDBRecord.get("NewImage");  
    Instant recordTimestamp = fetchTimestamp(dynamoDBRecord);  
  
    /**  
     * Say for example our record contains a String attribute named "stringName"  
and we wanted to fetch the value  
     * of this attribute from the new item image, the below code would fetch this.  
     */  
    JsonNode attributeNode = newItemImage.get("stringName");  
    JsonNode attributeValueNode = attributeNode.get("S"); // Using DynamoDB "S"  
    type attribute  
    String attributeValue = attributeValueNode.textValue();  
    System.out.println(attributeValue);  
}  
  
private Instant fetchTimestamp(JsonNode dynamoDBRecord) {  
    JsonNode timestampJson = dynamoDBRecord.get("ApproximateCreationDateTime");  
    return Instant.ofEpochMilli(timestampJson.longValue());  
}
```

Para saber mais sobre como criar consumidores e conectar seu fluxo de dados do Kinesis a outros AWS serviços, consulte [Lendo dados dos fluxos de dados do Amazon Kinesis](#), no Guia do desenvolvedor de Amazon Kinesis Data Streams.

Políticas personalizadas do IAM para Amazon Kinesis Data Streams para Amazon DynamoDB

Na primeira vez que você habilita o Amazon Kinesis Data Streams para o Amazon DynamoDB, o DynamoDB cria automaticamente uma AWS Identity and Access Management Função vinculada ao serviço (IAM) para você. Esta função, `AWSServiceRoleForDynamoDBKinesisDataStreamsReplication`, permite que o DynamoDB gerencie a replicação de alterações no nível de item nos Kinesis Data Streams em seu nome. Não exclua essa função vinculada ao serviço.

Para obter mais informações sobre funções vinculadas a serviços, consulte [Usar funções vinculadas a serviços](#) no Guia do usuário do IAM.

Os exemplos a seguir mostram como usar políticas do IAM para conceder permissões para Amazon Kinesis Data Streams para o Amazon DynamoDB.

- Para habilitar Amazon Kinesis Data Streams para o Amazon DynamoDB, você deve ter `dynamodb:EnableKinesisStreamingDestination` na mesa.
- Para desativar Amazon Kinesis Data Streams para o Amazon DynamoDB, você deve ter `dynamodb:DisableKinesisStreamingDestination` na mesa.

- Para descrever Amazon Kinesis Data Streams para o Amazon DynamoDB para uma determinada tabela do DynamoDB, você deve ter `dynamodb:DescribeKinesisStreamingDestination` permissão na mesa.

Exemplo: Habilitar Amazon Kinesis Data Streams para o Amazon DynamoDB

A política do IAM a seguir concede permissões para ativar Amazon Kinesis Data Streams para o Amazon DynamoDB para o `Music` e não concede permissões para desativar ou descrever Kinesis Data Streams para o DynamoDB para a `Music` Tabela do.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "iam:CreateServiceLinkedRole",  
            "Resource": "arn:aws:iam::*:role/aws-service-role/  
kinesisreplication.dynamodb.amazonaws.com/  
AWSServiceRoleForDynamoDBKinesisDataStreamsReplication",  
            "Condition": {"StringLike": {"iam:AWSServiceName":  
"kinesisreplication.dynamodb.amazonaws.com"}}  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:EnableKinesisStreamingDestination"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Music"  
        }  
    ]  
}
```

Exemplo: Desativar Amazon Kinesis Data Streams para o Amazon DynamoDB

A política do IAM a seguir concede permissões para desativar o Amazon Kinesis Data Streams para o Amazon DynamoDB para o `Music` e não concede permissões para habilitar ou descrever Amazon Kinesis Data Streams para o Amazon DynamoDB para a `Music` Tabela do.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:DisableKinesisStreamingDestination"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Music"  
        }  
    ]  
}
```

Exemplo: Aplicar seletivamente permissões para Amazon Kinesis Data Streams para Amazon DynamoDB com base no recurso

A política do IAM a seguir concede permissões para ativar e descrever Amazon Kinesis Data Streams para o Amazon DynamoDB para o `Music` e nega permissões para desativar os Amazon Kinesis Data Streams para o Amazon DynamoDB para a `Orders` Tabela do.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:EnableKinesisStreamingDestination",  
                "dynamodb:DescribeKinesisStreamingDestination"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Music"  
        },  
        {  
            "Effect": "Deny",  
            "Action": [  
                "dynamodb:DisableKinesisStreamingDestination"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Orders"  
        }  
    ]  
}
```

Uso de funções vinculadas ao serviço do Kinesis Data Streams para DynamoDB

Amazon Kinesis Data Streams para Amazon DynamoDB usa AWS Identity and Access Management(IAM) [Funções vinculadas ao serviço](#). Uma função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente aos Kinesis Data Streams para DynamoDB. As funções vinculadas a serviços são predefinidas pelo Kinesis Data Streams for DynamoDB e incluem todas as permissões exigidas pelo serviço para chamar outras AWS Serviços do em seu nome.

Uma função vinculada a serviço facilita a configuração de Kinesis Data Streams para o DynamoDB porque você não precisa adicionar as permissões necessárias manualmente. O Kinesis Data Streams for DynamoDB define as permissões das funções vinculadas a serviços e, exceto se definido de outra forma, somente Kinesis Data Streams para DynamoDB podem assumir suas funções. As permissões definidas incluem as políticas de confiança e de permissões, e essa política de permissões não pode ser anexada a nenhuma outra entidade do IAM.

Para obter informações sobre outros serviços que oferecem suporte às funções vinculadas a serviço, consulte [Serviços da AWS compatíveis com o IAM](#) e procure os serviços que apresentam Sim na coluna Função vinculada a serviços. Escolha um Sim com um link para exibir a documentação da função vinculada a serviço desse serviço.

Permissões de função vinculada ao serviço Kinesis Data Streams para o DynamoDB

Kinesis Data Streams para DynamoDB usa a função vinculada ao serviço denominada `AWSServiceRoleForDynamoDBKinesisDataStreamsReplication`. O objetivo da função vinculada ao serviço é permitir que o Amazon DynamoDB gerencie a replicação de alterações no nível de item nos Kinesis Data Streams, em seu nome.

A função vinculada ao serviço `AWSServiceRoleForDynamoDBKinesisDataStreamsReplication` confia nos seguintes serviços para assumir a função:

- `kinesisreplication.dynamodb.amazonaws.com`

A política de permissões da função permite que Kinesis Data Streams para o DynamoDB concluam as seguintes ações nos recursos especificados:

- Ações: `Put records and describe no Kinesis stream`

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Para obter mais informações, consulte [Permissões de função vinculada ao serviço](#) no Guia do usuário do IAM.

[Criação de uma função vinculada ao serviço Kinesis Data Streams para o DynamoDB](#)

Você não precisa criar manualmente uma função vinculada ao serviço. Quando você habilita Kinesis Data Streams para o DynamoDB no AWS Management Console, o AWS CLI, ou o AWS Kinesis Data Streams for DynamoDB cria uma função vinculada ao serviço para você.

Se você excluir essa função vinculada ao serviço e precisar criá-la novamente, poderá usar esse mesmo processo para recriar a função em sua conta. Quando você habilita Kinesis Data Streams para o DynamoDB, Kinesis Data Streams para DynamoDB criará novamente a função vinculada ao serviço.

[Edição de uma função vinculada ao serviço Kinesis Data Streams para o DynamoDB](#)

O Kinesis Data Streams para DynamoDB não permite editar `oAWSServiceRoleForDynamoDBKinesisDataStreamsReplicationFunction` vinculada ao serviço do. Depois que criar uma função vinculada ao serviço, você não poderá alterar o nome da função, pois várias entidades podem fazer referência a ela. No entanto, você poderá editar a descrição da função usando o IAM. Para obter mais informações, consulte [Editar uma função vinculada ao serviço](#) no Guia do usuário do IAM.

[Exclusão de uma função vinculada ao serviço Kinesis Data Streams para o DynamoDB](#)

Você também pode usar o console do IAM, o AWS CLI ou o AWS API para excluir manualmente a função vinculada ao serviço. Para isso, primeiro você deve limpar manualmente os recursos de sua função vinculada a serviço e depois excluí-la manualmente.

Note

Se o serviço Kinesis Data Streams for DynamoDB estiver usando a função quando tenta excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Como excluir manualmente a função vinculada ao serviço usando o IAM

Use o console do IAM, o AWS CLI, ou o AWS API para excluir `oAWSServiceRoleForDynamoDBKinesisDataStreamsReplicationFunction` vinculada ao serviço do. Para obter mais informações, consulte [Excluir uma função vinculada ao serviço](#) no Guia do usuário do IAM.

Captura de dados de alteração para DynamoDB Streams

O DynamoDB Streams captura uma sequência em ordem temporal de modificações em nível de item qualquer tabela do DynamoDB e armazena essas informações em um log por até 24 horas. Os aplicativos podem acessar esse log e visualizar os itens de dados à medida que eles aparecem antes e depois que foram modificados, em tempo quase real.

A criptografia em repouso criptografa os dados nos fluxos do DynamoDB. Para mais informações, consulte [Criptografia do DynamoDB em repouso](#) (p. 874).

ASream do DynamoDB é um fluxo ordenado de informações sobre alterações em itens em uma tabela do DynamoDB. Quando você ativa um stream em uma tabela, o DynamoDB captura informações sobre todas as modificações em itens de dados na tabela.

Sempre que um aplicativo cria, atualiza ou exclui itens nessa tabela, o DynamoDB Streams grava um registro de stream com os atributos de chave primária dos itens que foram modificados. ARegistro de streamO contém informações sobre uma modificação de dados em um único item em uma tabela do

DynamoDB. É possível configurar o stream de modo que os registros de stream capturem informações adicionais, como as imagens "antes" e "depois" de itens modificados.

O DynamoDB Streams ajuda a garantir o seguinte:

- Cada registro de stream aparece exatamente uma vez no stream.
- Para cada item modificado em uma tabela do DynamoDB, os registros de stream aparecem na mesma sequência que as modificações reais no item.

O DynamoDB Streams grava registros de stream em tempo quase real, de modo que você possa criar aplicativos que consomem esses streams e realizam ações com base no conteúdo.

Tópicos

- [Endpoints para DynamoDB Streams \(p. 652\)](#)
- [Habilitação de um stream \(p. 652\)](#)
- [Leitura e processamento de um stream \(p. 653\)](#)
- [DynamoDB Streams e vida útil \(p. 655\)](#)
- [Uso do DynamoDB Streams Kinesis para processar registros de stream \(p. 656\)](#)
- [API de baixo nível do DynamoDB Streams: Exemplo de Java \(p. 668\)](#)
- [DynamoDB Streams e AWS LambdaGatilhos \(p. 672\)](#)

Endpoints para DynamoDB Streams

AWS mantém endpoints separados para o DynamoDB e DynamoDB Streams. Para trabalhar com índices e tabelas de banco de dados, seu aplicativo deve acessar um endpoint do DynamoDB. Para ler e processar registros do DynamoDB Streams, seu aplicativo precisa acessar um endpoint do DynamoDB Streams na mesma região.

A convenção de nomenclatura de endpoints do DynamoDB Streams é `streams.dynamodb.<region>.amazonaws.com`. Por exemplo, se você usar o `endpointdynamodb.us-west-2.amazonaws.com` para acessar o DynamoDB, você pode usar o `endpointstreams.dynamodb.us-west-2.amazonaws.com` para acessar DynamoDB Streams.

Note

Para obter uma lista completa de endpoints e regiões do Streams do DynamoDB e do DynamoDB, consulte [Regiões e endpoints do AWS Referência geral](#).

OAWSOs SDKs fornecem clientes separados para o DynamoDB e o DynamoDB Streams. Dependendo das suas necessidades, seu aplicativo pode acessar um endpoint do DynamoDB, um endpoint do DynamoDB Streams ou ambos ao mesmo tempo. Para se conectar a ambos os endpoints, seu aplicativo precisa instanciar dois clientes: um para o DynamoDB e outro para o DynamoDB Streams.

Habilitação de um stream

Você pode habilitar um stream em uma nova tabela ao criá-la usando oAWS CLIou uma das receitasAWSSDKs do. Também pode habilitar ou desabilitar um stream em uma tabela existente ou alterar as configurações de um stream. O DynamoDB Streams opera de forma assíncrona e, portanto, não haverá impacto sobre o desempenho de uma tabela se você habilitar um stream.

A maneira mais fácil de gerenciar o DynamoDB Streams é usando oAWS Management Console.

1. Faça login noAWS Management Consolesee abra o console do DynamoDB em<https://console.aws.amazon.com/dynamodb/>.
2. No painel do console do DynamoDB, escolhaTabelase selecione uma tabela existente.

3. Na guia Overview (Visão geral), escolha Manage Stream (Gerenciar stream).
4. Na janela Manage Stream (Gerenciar stream), escolha as informações que serão gravadas no stream sempre que os dados da tabela forem modificados:
 - Somente chaves— somente os atributos de chaves do item modificado.
 - Imagem nova— o item inteiro, como aparece depois de ser modificado.
 - Imagem antiga— o item inteiro como aparecia antes de ser modificado.
 - Imagens novas e antigas— as imagens nova e antiga do item.
- Quando estiver satisfeito com as configurações, escolha Enable (Habilitar).
5. (Opcional) Para desabilitar um stream existente, escolha Manage Stream (Gerenciar stream) e, em seguida, Disable (Desabilitar).

Você também pode usar as operações da API `CreateTable` ou `UpdateTable` para habilitar ou modificar um stream. O parâmetro `StreamSpecification` determina como o stream é configurado:

- `StreamEnabled`— Especifica se um stream está habilitado (`true`) ou desativado (`false`) para a tabela.
- `StreamViewType`— especifica as informações que serão gravadas no stream sempre que os dados na tabela forem modificados:
 - `KEYS_ONLY`— somente os atributos de chaves do item modificado.
 - `NEW_IMAGE`— o item inteiro como aparece depois de ser modificado.
 - `OLD_IMAGE`— o item inteiro como aparecia antes de ser modificado.
 - `NEW_AND_OLD_IMAGES`— as imagens nova e antiga do item.

É possível habilitar ou desabilitar um stream a qualquer momento. No entanto, você receberá uma `ResourceInUseException` se tentar habilitar um stream em uma tabela que já tem um stream. Você receberá uma `ValidationException` se tentar habilitar um stream em uma tabela que já tem um stream.

Quando você definir `streamEnabled` para `true`, o DynamoDB cria um novo stream com um descritor de stream exclusivo atribuído a ele. Se você desabilitar e reabilitar um streaming na tabela, será criado um novo streaming com um descritor diferente.

Cada stream é identificado exclusivamente por um Nome de recurso da Amazon (ARN). Veja a seguir um ARN de exemplo para um stream em uma tabela do DynamoDB chamada `TestTable`.

```
arn:aws:dynamodb:us-west-2:111122223333:table/TestTable/stream/2015-05-11T21:21:33.291
```

Para determinar o descritor de stream mais recente de uma tabela, emita um `DynamoDBDescribeTable` e procure o `LatestStreamArn` elemento na resposta.

Leitura e processamento de um stream

Para ler e processar um stream, seu aplicativo precisa se conectar a um endpoint do DynamoDB Streams e emitir solicitações de API.

Um stream consiste em registros de stream. Cada registro de stream representa uma única modificação de dados na tabela do DynamoDB à qual o stream pertence. Cada registro de stream recebe um número de sequência, refletindo a ordem em que ele foi publicado no stream.

Os registros de streaming estão organizados em grupos ou estilhaços. Cada estilhaço atua como um contêiner para vários registros de stream e contém informações necessárias para acessar esses registros

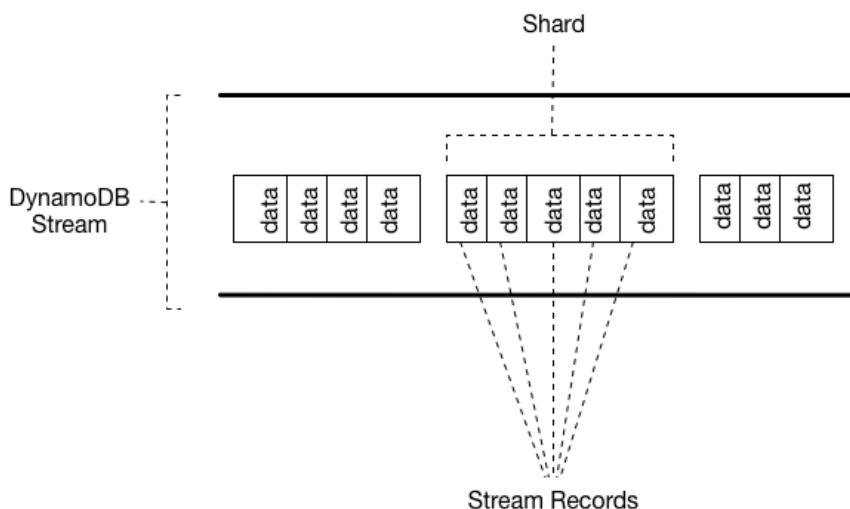
e fazer iterações neles. Os registros de stream em um estilhaço são removidos automaticamente depois de 24 horas.

Os fragmentos são efêmeros: Eles são criados e excluídos automaticamente, conforme necessário. Qualquer estilhaço também pode ser dividido em vários novos estilhaços, o que também ocorre automaticamente. (Também é possível que um estilhaço pai tenha apenas um estilhaço filho.) Um estilhaço pode ser dividido em resposta a altos níveis de atividades de gravação em sua tabela principal e, portanto, os aplicativos podem processar registros de vários estilhaços em paralelo.

Se você desabilitar um stream, todos os estilhaços que estiverem abertos serão fechados. Os dados no stream continuarão legíveis por 24 horas.

Como estilhaços têm uma linhagem (pai e filhos), um aplicativo sempre deverá processar um estilhaço pai antes de um estilhaço filho. Isso ajuda a garantir que os registros de streaming também sejam processados na ordem correta. (Se você usar o DynamoDB Streams Kinesis Adapter, isso será processado para você. Seu aplicativo processa os estilhaços e registros de streaming na ordem correta. Ele processa automaticamente os estilhaços novos ou expirados, além dos estilhaços que são divididos enquanto o aplicativo está em execução. Para obter mais informações, consulte [Uso do DynamoDB Streams Kinesis para processar registros de stream \(p. 656\)](#).)

O diagrama a seguir mostra a relação entre um stream, os estilhaços nesse stream e os registros de stream nesses estilhaços.



Note

Se você executar um `PutItem` ou `UpdateItem` que não altera nenhum dado em um item, o DynamoDB Streams não gravará um registro de fluxo para essa operação.

Para acessar um stream e processar os registros de stream dentro dele, você deve fazer o seguinte:

- Determinar o ARN exclusivo do stream que você deseja acessar.
- Determinar quais estilhaços no stream contêm os registros de stream desejados.
- Acessar os estilhaços e recuperar os registros de stream desejados.

Note

No máximo dois processos devem estar lendo do mesmo estilhaço de streams ao mesmo tempo. Ter mais de dois leitores por estilhaço pode resultar em controle de utilização.

A API do DynamoDB Streams fornece as seguintes ações para uso por programas aplicativos:

- [ListStreams](#)— retorna uma lista de descritores de stream para a conta e o endpoint atuais. Opcionalmente, você pode solicitar apenas os descritores de stream de um nome de tabela específico.
- [DescribeStream](#)— retorna informações detalhadas sobre um determinado stream. A saída inclui uma lista de estilhaços associados ao stream, incluindo os IDs desses estilhaços.
- [GetShardIterator](#)— Retorna um iterador de estilhaço, que descreve um local dentro de um estilhaço. Você pode solicitar que o iterador forneça acesso ao ponto mais antigo, o ponto mais novo ou um ponto específico no stream.
- [GetRecords](#)— Retorna os registros do stream de um determinado estilhaço. Você deve fornecer o iterador de estilhaço retornado de uma solicitação GetShardIterator.

Para obter descrições completas dessas operações da API, incluindo exemplos de solicitações e respostas, acesse o[Referência de API do Amazon DynamoDB Streams](#).

Límite de retenção de dados para DynamoDB Streams

Todos os dados no DynamoDB Streams estão sujeitos a um tempo de vida de 24 horas. É possível recuperar e analisar as atividades das últimas 24 horas de qualquer tabela. No entanto, os dados mais antigos que 24 horas são suscetíveis a remoção a qualquer momento.

Se você desabilitar um stream em uma tabela, os dados nesse stream continuarão legíveis por 24 horas. Depois desse tempo, os dados expirarão, e os registros de stream serão excluídos automaticamente. Não há nenhum mecanismo para excluir manualmente um stream existente. Aguarde até que o limite de retenção expire (24 horas), e todos os registros do stream serão excluídos.

DynamoDB Streams e vida útil

Você pode fazer backup dos itens excluídos pelo, ou de outra forma processá-los,[Tempo de vida \(p. 439\)](#)(TTL) habilitando o Amazon DynamoDB Streams na tabela e processando os registros de fluxos dos itens expirados.

O registro de streams contém um campo de identidade do usuário Records[[`<index>`](#)].[userIdentity](#).

Os itens excluídos pelo processo de tempo para uso após a expiração possuem os campos a seguir:

- `Records[<index>].userIdentity.type`
"Service"
- `Records[<index>].userIdentity.principalId`
"dynamodb.amazonaws.com"

O JSON a seguir mostra a parte relevante de um único registro de streams.

```
"Records": [
    {
        ...
        "userIdentity": {
            "type": "Service",
            "principalId": "dynamodb.amazonaws.com"
        }
        ...
    }
]
```

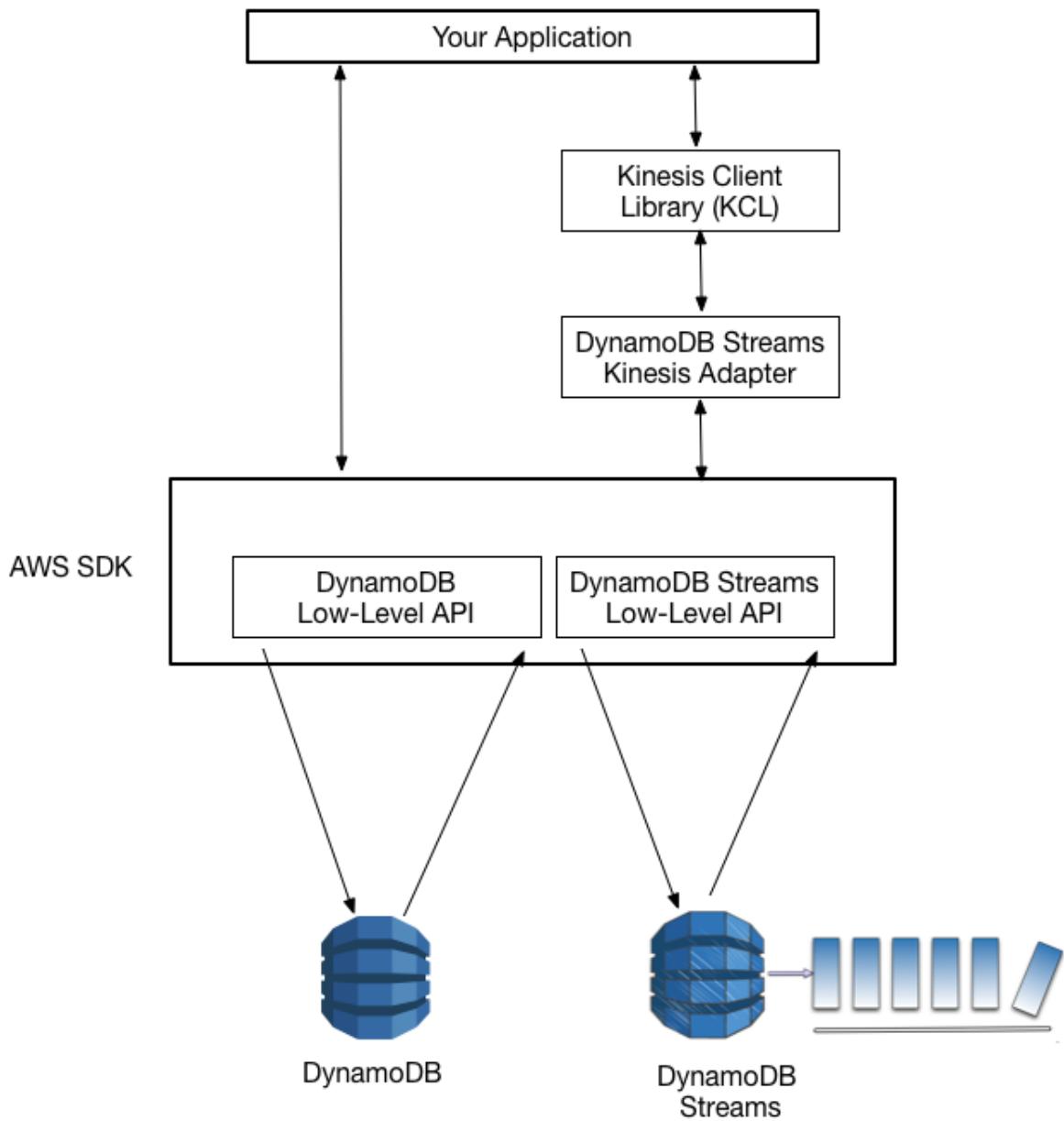
Uso do DynamoDB Streams Kinesis para processar registros de stream

Usar o Amazon Kinesis Adapter é a forma recomendada para consumir streams do Amazon DynamoDB. A API do DynamoDB Streams é intencionalmente semelhante à do Kinesis Data Streams, um serviço para processamento em tempo real de dados em streaming em altíssima escala. Em ambos os serviços, os streams de dados são compostos de estilhaços, que são contêineres de registros de stream. Ambas as APIs de serviços contêm `ListStreams`, `DescribeStream`, `GetShards`, e `GetShardIterator`. Operações (Embora essas ações do DynamoDB Streams sejam semelhantes às suas equivalentes no Kinesis Data Streams, elas não são 100% idênticas.)

Você pode escrever aplicativos para Kinesis Data Streams usando a Kinesis Client Library (KCL). A KCL simplifica a codificação, fornecendo abstrações úteis acima da API de baixo nível do Kinesis Data Streams. Para obter mais informações sobre a KCL, consulte a [Desenvolver consumidores usando a biblioteca do cliente Kinesis](#) no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Como um usuário do DynamoDB Streams, você pode usar os padrões de design encontrados no KCL para processar estilhaços do DynamoDB Streams e registros de stream. Para isso, você pode usar o DynamoDB Streams Kinesis Adapter. O Kinesis Adapter implementa a interface de Kinesis Data Streams para que a KCL possa ser usada para consumir e processar registros do DynamoDB Streams.

O diagrama a seguir mostra como essas bibliotecas interagem entre si.



Com o DynamoDB Streams Kinesis Adapter implementado, você pode começar a desenvolver a interface da KCL com as chamadas de API perfeitamente direcionadas no endpoint do DynamoDB Streams.

Quando o aplicativo é iniciado, ele chama a KCL para instanciar um operador. Você deve fornecer o operador com informações de configuração para o aplicativo, tais como o descritor do stream e o AWS SDK o nome de uma classe de processador de registro que você fornecer. Como ele executa o código no processador de registro, o operador executa as seguintes tarefas:

- Conecta-se ao stream.
- Enumera os estilhaços no stream.
- Coordena as associações do estilhaço com outros operadores (se houver).
- Cria uma instância de um processador de registro para cada estilhaço que gerencia.

- Extrai registros do stream.
- Envia os registros para o processador de registros correspondente.
- Registros processados pelos pontos de verificação.
- Equilibra as associações de estilhaço-operador quando a contagem de instância de operadores muda.
- Equilibra as associações de estilhaço-operador quando os estilhaços se dividem.

Note

Para obter uma descrição dos conceitos KCL listados aqui, consulte [Desenvolver consumidores usando a biblioteca do cliente Kinesis no Guia do desenvolvedor do Amazon Kinesis Data Streams](#).

Passo a passo: DynamoDB Streams Kinesis Adapter

Esta seção é um passo-a-passo de um aplicativo Java que usa a Amazon Kinesis Client Library (KCL) e o Amazon DynamoDB Streams Kinesis Adapter. O aplicativo mostra um exemplo de replicação de dados, no qual as atividades de gravação de uma tabela são aplicadas a uma segunda tabela, com o conteúdo de ambas mantido em sincronia. Para o código-fonte, consulte [Programa completo: DynamoDB Streams Kinesis Adapter \(p. 661\)](#).

O programa faz o seguinte:

1. Cria duas tabelas do DynamoDB denominadas `KCL-Demo-srce` e `KCL-Demo-dst`. Cada uma dessas tabelas tem um stream habilitado.
2. Gera atividades de atualização na tabela de origem, adicionando, atualizando e excluindo itens. Isso faz com que os dados sejam gravados no stream da tabela.
3. Lê os registros do stream, faz a reconstrução desses registros como solicitações do DynamoDB e aplica essas solicitações à tabela de destino.
4. Verifique as tabelas de origem e destino para garantir que o conteúdo seja idêntico.
5. Realiza uma limpeza excluindo as tabelas.

Essas etapas estão descritas nas seções a seguir, e a aplicação completa é mostrada no final do passo-a-passo.

Tópicos

- [Etapa 1: Criar tabelas do DynamoDB \(p. 658\)](#)
- [Etapa 2: Gerar atividade de atualização na tabela de origem \(p. 659\)](#)
- [Etapa 3: Processe o stream \(p. 659\)](#)
- [Etapa 4: Certifique-se de que ambas as tabelas tenham conteúdo idêntico \(p. 660\)](#)
- [Etapa 5: Limpar \(p. 660\)](#)
- [Programa completo: DynamoDB Streams Kinesis Adapter \(p. 661\)](#)

Etapa 1: Criar tabelas do DynamoDB

O primeiro passo é criar duas tabelas do DynamoDB: uma de origem e outra de destino. O `StreamViewType` no stream da tabela de origem é `NEW_IMAGE`. Isso significa que sempre que um item é modificado nessa tabela, o item "depois" da imagem é gravado no stream. Dessa forma, o stream mantém o controle de todas as atividades de gravação na tabela.

O exemplo a seguir mostra o código usado para a criação das duas tabelas.

```
java.util.List<AttributeDefinition> attributeDefinitions = new
    ArrayList<AttributeDefinition>();
attributeDefinitions.add(new
    AttributeDefinition().withAttributeName("Id").withAttributeType("N"));

java.util.List<KeySchemaElement> keySchema = new ArrayList<KeySchemaElement>();
keySchema.add(new KeySchemaElement().withAttributeName("Id").withKeyType(KeyType.HASH)); // Partition
// key

ProvisionedThroughput provisionedThroughput = new
    ProvisionedThroughput().withReadCapacityUnits(2L)
        .withWriteCapacityUnits(2L);

StreamSpecification streamSpecification = new StreamSpecification();
streamSpecification.setStreamEnabled(true);
streamSpecification.setStreamViewType(StreamViewType.NEW_IMAGE);
CreateTableRequest createTableRequest = new CreateTableRequest().withTableName(tableName)
    .withAttributeDefinitions(attributeDefinitions).withKeySchema(keySchema)

    .withProvisionedThroughput(provisionedThroughput).withStreamSpecification(streamSpecification);
```

Etapa 2: Gerar atividade de atualização na tabela de origem

O próximo passo é gerar algumas atividades de gravação na tabela de origem. Enquanto essas atividades estão ocorrendo, o stream da tabela de origem também é atualizado em tempo quase real.

O aplicativo define uma classe auxiliar com métodos que chamam as operações da API PutItem, UpdateItem e DeleteItem para gravar os dados. O exemplo de código a seguir mostra como esses métodos são usados.

```
StreamsAdapterDemoHelper.putItem(dynamoDBClient, tableName, "101", "test1");
StreamsAdapterDemoHelper.updateItem(dynamoDBClient, tableName, "101", "test2");
StreamsAdapterDemoHelper.deleteItem(dynamoDBClient, tableName, "101");
StreamsAdapterDemoHelper.putItem(dynamoDBClient, tableName, "102", "demo3");
StreamsAdapterDemoHelper.updateItem(dynamoDBClient, tableName, "102", "demo4");
StreamsAdapterDemoHelper.deleteItem(dynamoDBClient, tableName, "102");
```

Etapa 3: Processe o stream

Agora, o programa inicia o processamento do stream. O DynamoDB Streams Kinesis Adapter atua como uma camada transparente entre a KCL e o endpoint do DynamoDB Streams, para que o código possa usar totalmente a KCL em vez de precisar fazer chamadas do DynamoDB Streams de baixo nível. O programa realiza as seguintes tarefas:

- Ele define uma classe de processador de registro, StreamsRecordProcessor, com métodos que estão em conformidade com a definição de interface da KCL: initialize, processRecords e shutdown. O método processRecords contém a lógica necessária para leituras do stream da tabela de origem e para gravações na tabela de destino.
- Ele define uma fábrica de classes para a classe de processador de registro (StreamsRecordProcessorFactory). Isso é necessário para programas Java que usam a KCL.
- Ele instancia um novo Worker da KCL, que está associado à fábrica de classes.
- Ele desliga Worker quando o processamento do registro é concluído.

Para saber mais sobre a definição de interface KCL, consulte [Desenvolver consumidores usando a biblioteca do cliente Kinesis](#) no Guia do desenvolvedor do Amazon Kinesis Data Streams.

O exemplo de código a seguir mostra o loop principal em `StreamsRecordProcessor`. A instrução `case` determina a ação a ser executada, com base no `OperationType` que aparece no registro de stream.

```
for (Record record : records) {
    String data = new String(record.getData().array(), Charset.forName("UTF-8"));
    System.out.println(data);
    if (record instanceof RecordAdapter) {
        com.amazonaws.services.dynamodbv2.model.Record streamRecord =
        ((RecordAdapter) record)
            .getInternalObject();

        switch (streamRecord.getEventName()) {
            case "INSERT":
            case "MODIFY":
                StreamsAdapterDemoHelper.putItem(dynamoDBClient, tableName,
                    streamRecord.getDynamodb().getNewImage());
                break;
            case "REMOVE":
                StreamsAdapterDemoHelper.deleteItem(dynamoDBClient, tableName,
                    streamRecord.getDynamodb().getKeys().get("Id").getN());
        }
    }
    checkpointCounter += 1;
    if (checkpointCounter % 10 == 0) {
        try {
            checkpointer.checkpoint();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Etapa 4: Certifique-se de que ambas as tabelas tenham conteúdo idêntico

Neste ponto, o conteúdo das tabelas de origem e destino está sincronizado. O aplicativo emite solicitações `Scan` em ambas as tabelas para verificar se o conteúdo delas é realmente idêntico.

A classe `DemoHelper` contém um método `ScanTable` que chama a API de `Scan` de baixo nível. O exemplo a seguir mostra como fazer isso.

```
if (StreamsAdapterDemoHelper.scanTable(dynamoDBClient, srcTable).getItems()
    .equals(StreamsAdapterDemoHelper.scanTable(dynamoDBClient, destTable).getItems())) {
    System.out.println("Scan result is equal.");
}
else {
    System.out.println("Tables are different!");
}
```

Etapa 5: Limpar

A demonstração está concluída e, portanto, o aplicativo exclui as tabelas de origem e destino. Consulte o seguinte exemplo de código. Mesmo depois que as tabelas são excluídas, seus streams permanecem disponíveis por até 24 horas, após o que eles são automaticamente excluídos.

```
dynamoDBClient.deleteTable(new DeleteTableRequest().withTableName(srcTable));
dynamoDBClient.deleteTable(new DeleteTableRequest().withTableName(destTable));
```

Programa completo: DynamoDB Streams Kinesis Adapter

Veja a seguir o programa Java completo que realiza as tarefas descritas em [Passo a passo: DynamoDB Streams Kinesis Adapter \(p. 658\)](#). Quando executá-lo, você verá uma saída semelhante à seguinte:

```
Creating table KCL-Demo-src
Creating table KCL-Demo-dest
Table is active.
Creating worker for stream: arn:aws:dynamodb:us-west-2:111122223333:table/KCL-Demo-src/
stream/2015-05-19T22:48:56.601
Starting worker...
Scan result is equal.
Done.
```

Important

Para executar esse programa, verifique se o aplicativo cliente tem acesso ao DynamoDB e ao Amazon CloudWatch usando políticas. Para mais informações, consulte [Uso de políticas baseadas em identidade \(políticas do IAM\) com o Amazon DynamoDB \(p. 888\)](#).

O código-fonte consiste em quatro arquivos .java:

- `StreamsAdapterDemo.java`
- `StreamsRecordProcessor.java`
- `StreamsRecordProcessorFactory.java`
- `StreamsAdapterDemoHelper.java`

StreamsAdapterDemo.java

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples;

import com.amazonaws.auth.AWS CredentialsProvider;
import com.amazonaws.auth.DefaultAWS CredentialsProviderChain;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreams;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClientBuilder;
import com.amazonaws.services.dynamodbv2.model.DeleteTableRequest;
import com.amazonaws.services.dynamodbv2.model.DescribeTableResult;
import com.amazonaws.services.dynamodbv2.streamsadapter.AmazonDynamoDBStreamsAdapterClient;
import com.amazonaws.services.dynamodbv2.streamsadapter.StreamsWorkerFactory;
import com.amazonaws.services.kinesis.clientlibrary.interfaces.v2.IRecordProcessorFactory;
```

```
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.InitialPositionInStream;
import
com.amazonaws.services.kinesis.clientlibrary.lib.worker.KinesisClientLibConfiguration;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker;

public class StreamsAdapterDemo {
    private static Worker worker;
    private static KinesisClientLibConfiguration workerConfig;
    private static IRecordProcessorFactory recordProcessorFactory;

    private static AmazonDynamoDB dynamoDBClient;
    private static AmazonCloudWatch cloudWatchClient;
    private static AmazonDynamoDBStreams dynamoDBStreamsClient;
    private static AmazonDynamoDBStreamsAdapterClient adapterClient;

    private static String tablePrefix = "KCL-Demo";
    private static String streamArn;

    private static Regions awsRegion = Regions.US_EAST_2;

    private static AWSCredentialsProvider awsCredentialsProvider =
DefaultAWSCredentialsProviderChain.getInstance();

    /**
     * @param args
     */
    public static void main(String[] args) throws Exception {
        System.out.println("Starting demo...");

        dynamoDBClient = AmazonDynamoDBClientBuilder.standard()
            .withRegion(awsRegion)
            .build();
        cloudWatchClient = AmazonCloudWatchClientBuilder.standard()
            .withRegion(awsRegion)
            .build();
        dynamoDBStreamsClient = AmazonDynamoDBStreamsClientBuilder.standard()
            .withRegion(awsRegion)
            .build();
        adapterClient = new AmazonDynamoDBStreamsAdapterClient(dynamoDBStreamsClient);
        String srcTable = tablePrefix + "-src";
        String destTable = tablePrefix + "-dest";
        recordProcessorFactory = new StreamsRecordProcessorFactory(dynamoDBClient,
destTable);

        setUpTables();

        workerConfig = new KinesisClientLibConfiguration("streams-adapter-demo",
streamArn,
awsCredentialsProvider,
"streams-demo-worker")
            .withMaxRecords(1000)
            .withIdleTimeBetweenReadsInMillis(500)
            .withInitialPositionInStream(InitialPositionInStream.TRIM_HORIZON);

        System.out.println("Creating worker for stream: " + streamArn);
        worker = StreamsWorkerFactory.createDynamoDbStreamsWorker(recordProcessorFactory,
workerConfig, adapterClient, dynamoDBClient, cloudWatchClient);
        System.out.println("Starting worker...");
        Thread t = new Thread(worker);
        t.start();

        Thread.sleep(25000);
        worker.shutdown();
        t.join();

        if (StreamsAdapterDemoHelper.scanTable(dynamoDBClient, srcTable).getItems()
```

```
.equals(StreamsAdapterDemoHelper.scanTable(dynamoDBClient, destTable).getItems())) {
    System.out.println("Scan result is equal.");
}
else {
    System.out.println("Tables are different!");
}

System.out.println("Done.");
cleanupAndExit(0);
}

private static void setUpTables() {
    String srcTable = tablePrefix + "-src";
    String destTable = tablePrefix + "-dest";
    streamArn = StreamsAdapterDemoHelper.createTable(dynamoDBClient, srcTable);
    StreamsAdapterDemoHelper.createTable(dynamoDBClient, destTable);

    awaitTableCreation(srcTable);

    performOps(srcTable);
}

private static void awaitTableCreation(String tableName) {
    Integer retries = 0;
    Boolean created = false;
    while (!created && retries < 100) {
        DescribeTableResult result =
StreamsAdapterDemoHelper.describeTable(dynamoDBClient, tableName);
        created = result.getTable().getTableStatus().equals("ACTIVE");
        if (created) {
            System.out.println("Table is active.");
            return;
        }
        else {
            retries++;
            try {
                Thread.sleep(1000);
            }
            catch (InterruptedException e) {
                // do nothing
            }
        }
    }
    System.out.println("Timeout after table creation. Exiting...");
    cleanupAndExit(1);
}

private static void performOps(String tableName) {
    StreamsAdapterDemoHelper.putItem(dynamoDBClient, tableName, "101", "test1");
    StreamsAdapterDemoHelper.updateItem(dynamoDBClient, tableName, "101", "test2");
    StreamsAdapterDemoHelper.deleteItem(dynamoDBClient, tableName, "101");
    StreamsAdapterDemoHelper.putItem(dynamoDBClient, tableName, "102", "demo3");
    StreamsAdapterDemoHelper.updateItem(dynamoDBClient, tableName, "102", "demo4");
    StreamsAdapterDemoHelper.deleteItem(dynamoDBClient, tableName, "102");
}

private static void cleanupAndExit(Integer returnValue) {
    String srcTable = tablePrefix + "-src";
    String destTable = tablePrefix + "-dest";
    dynamoDBClient.deleteTable(new DeleteTableRequest().withTableName(srcTable));
    dynamoDBClient.deleteTable(new DeleteTableRequest().withTableName(destTable));
    System.exit(returnValue);
}
```

StreamsRecordProcessor.java

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
  
package com.amazonaws.codesamples;  
  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.streamsadapter.model.RecordAdapter;  
import com.amazonaws.services.kinesis.clientlibrary.interfaces.v2.IRecordProcessor;  
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.ShutdownReason;  
import com.amazonaws.services.kinesis.clientlibrary.types.InitializationInput;  
import com.amazonaws.services.kinesis.clientlibrary.types.ProcessRecordsInput;  
import com.amazonaws.services.kinesis.clientlibrary.types.ShutdownInput;  
import com.amazonaws.services.kinesis.model.Record;  
  
import java.nio.charset.Charset;  
  
public class StreamsRecordProcessor implements IRecordProcessor {  
    private Integer checkpointCounter;  
  
    private final AmazonDynamoDB dynamoDBClient;  
    private final String tableName;  
  
    public StreamsRecordProcessor(AmazonDynamoDB dynamoDBClient2, String tableName) {  
        this.dynamoDBClient = dynamoDBClient2;  
        this.tableName = tableName;  
    }  
  
    @Override  
    public void initialize(InitializationInput initializationInput) {  
        checkpointCounter = 0;  
    }  
  
    @Override  
    public void processRecords(ProcessRecordsInput processRecordsInput) {  
        for (Record record : processRecordsInput.getRecords()) {  
            String data = new String(record.getData().array(), Charset.forName("UTF-8"));  
            System.out.println(data);  
            if (record instanceof RecordAdapter) {  
                com.amazonaws.services.dynamodbv2.model.Record streamRecord =  
                ((RecordAdapter) record)  
                    .getInternalObject();  
  
                switch (streamRecord.getEventName()) {  
                    case "INSERT":  
                    case "MODIFY":  
                        StreamsAdapterDemoHelper.putItem(dynamoDBClient, tableName,  
                            streamRecord.getDynamodb().getNewImage());  
                }  
            }  
        }  
    }  
}
```

```
        break;
    case "REMOVE":
        StreamsAdapterDemoHelper.deleteItem(dynamoDBClient, tableName,
streamRecord.getDynamodb().getKeys().get("Id").getN());
    }
}
checkpointCounter += 1;
if (checkpointCounter % 10 == 0) {
    try {
        processRecordsInput.getCheckpointer().checkpoint();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
}
}

@Override
public void shutdown(ShutdownInput shutdownInput) {
    if (shutdownInput.getShutdownReason() == ShutdownReason.TERMINATE) {
        try {
            shutdownInput.getCheckpointer().checkpoint();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}
```

[StreamsRecordProcessorFactory.java](#)

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.kinesis.clientlibrary.interfaces.v2.IRecordProcessor;
import com.amazonaws.services.kinesis.clientlibrary.interfaces.v2.IRecordProcessorFactory;

public class StreamsRecordProcessorFactory implements IRecordProcessorFactory {
    private final String tableName;
    private final AmazonDynamoDB dynamoDBClient;

    public StreamsRecordProcessorFactory(AmazonDynamoDB dynamoDBClient, String tableName) {
        this.tableName = tableName;
    }
}
```

```
        this.dynamoDBClient = dynamoDBClient;
    }

    @Override
    public IRecordProcessor createProcessor() {
        return new StreamsRecordProcessor(dynamoDBClient, tableName);
    }
}
```

StreamsAdapterDemoHelper.java

```
/** 
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.model.AttributeAction;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.AttributeValueUpdate;
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;
import com.amazonaws.services.dynamodbv2.model.CreateTableResult;
import com.amazonaws.services.dynamodbv2.model.DeleteItemRequest;
import com.amazonaws.services.dynamodbv2.model.DescribeTableRequest;
import com.amazonaws.services.dynamodbv2.model.DescribeTableResult;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.services.dynamodbv2.model.PutItemRequest;
import com.amazonaws.services.dynamodbv2.model.ResourceInUseException;
import com.amazonaws.services.dynamodbv2.model.ScanRequest;
import com.amazonaws.services.dynamodbv2.model.ScanResult;
import com.amazonaws.services.dynamodbv2.model.StreamSpecification;
import com.amazonaws.services.dynamodbv2.model.StreamViewType;
import com.amazonaws.services.dynamodbv2.model.UpdateItemRequest;

public class StreamsAdapterDemoHelper {

    /**
     * @return StreamArn
     */
    public static String createTable(AmazonDynamoDB client, String tableName) {
        java.util.List<AttributeDefinition> attributeDefinitions = new
ArrayList<AttributeDefinition>();
        attributeDefinitions.add(new
AttributeDefinition().withAttributeName("Id").withAttributeType("N"));
    }
}
```

```
java.util.List<KeySchemaElement> keySchema = new ArrayList<KeySchemaElement>();
keySchema.add(new
KeySchemaElement().withAttributeName("Id").withKeyType(KeyType.HASH)); // Partition

// key

ProvisionedThroughput provisionedThroughput = new
ProvisionedThroughput().withReadCapacityUnits(2L)
    .withWriteCapacityUnits(2L);

StreamSpecification streamSpecification = new StreamSpecification();
streamSpecification.setStreamEnabled(true);
streamSpecification.setStreamViewType(StreamViewType.NEW_IMAGE);
CreateTableRequest createTableRequest = new
CreateTableRequest().withTableName(tableName)
    .withAttributeDefinitions(attributeDefinitions).withKeySchema(keySchema)

.withProvisionedThroughput(provisionedThroughput).withStreamSpecification(streamSpecification);

try {
    System.out.println("Creating table " + tableName);
    CreateTableResult result = client.createTable(createTableRequest);
    return result.getTableDescription().getLatestStreamArn();
}
catch (ResourceInUseException e) {
    System.out.println("Table already exists.");
    return describeTable(client, tableName).getTable().getLatestStreamArn();
}
}

public static DescribeTableResult describeTable(AmazonDynamoDB client, String
tableName) {
    return client.describeTable(new DescribeTableRequest().withTableName(tableName));
}

public static ScanResult scanTable(AmazonDynamoDB dynamoDBClient, String tableName) {
    return dynamoDBClient.scan(new ScanRequest().withTableName(tableName));
}

public static void putItem(AmazonDynamoDB dynamoDBClient, String tableName, String id,
String val) {
    java.util.Map<String, AttributeValue> item = new HashMap<String, AttributeValue>();
    item.put("Id", new AttributeValue().withN(id));
    item.put("attribute-1", new AttributeValue().withS(val));

    PutItemRequest putItemRequest = new
PutItemRequest().withTableName(tableName).withItem(item);
    dynamoDBClient.putItem(putItemRequest);
}

public static void putItem(AmazonDynamoDB dynamoDBClient, String tableName,
    java.util.Map<String, AttributeValue> items) {
    PutItemRequest putItemRequest = new
PutItemRequest().withTableName(tableName).withItem(items);
    dynamoDBClient.putItem(putItemRequest);
}

public static void updateItem(AmazonDynamoDB dynamoDBClient, String tableName, String
id, String val) {
    java.util.Map<String, AttributeValue> key = new HashMap<String, AttributeValue>();
    key.put("Id", new AttributeValue().withN(id));

    Map<String, AttributeValueUpdate> attributeUpdates = new HashMap<String,
AttributeValueUpdate>();
    AttributeValueUpdate update = new
AttributeValueUpdate().withAction(AttributeAction.PUT)
```

```
        .withValue(new AttributeValue().withS(val));
    attributeUpdates.put("attribute-2", update);

    UpdateItemRequest updateItemRequest = new
UpdateItemRequest().withTableName(tableName).withKey(key)
        .withAttributeUpdates(attributeUpdates);
    dynamoDBClient.updateItem(updateItemRequest);
}

public static void deleteItem(AmazonDynamoDB dynamoDBClient, String tableName, String
id) {
    java.util.Map<String, AttributeValue> key = new HashMap<String, AttributeValue>();
    key.put("Id", new AttributeValue().withN(id));

    DeleteItemRequest deleteItemRequest = new
DeleteItemRequest().withTableName(tableName).withKey(key);
    dynamoDBClient.deleteItem(deleteItemRequest);
}

}
```

API de baixo nível do DynamoDB Streams: Exemplo de Java

Note

O código nesta página não é completo e não trata todos os cenários de consumo do Amazon DynamoDB Streams. A maneira recomendada de consumir registros de stream do DynamoDB é com o Amazon Kinesis Adapter usando a Kinesis Client Library (KCL), conforme descrito em [Uso do DynamoDB Streams Kinesis para processar registros de stream \(p. 656\)](#).

Esta seção contém um programa Java que mostra o DynamoDB Streams em ação. O programa faz o seguinte:

1. Cria uma tabela do DynamoDB com um stream habilitado.
2. Descreve as configurações de fluxo dessa tabela.
3. Modifica os dados na tabela.
4. Descreve os estilhaços no stream.
5. Lê os registros de stream dos estilhaços.
6. Limpa.

Quando você executar o programa, verá um resultado semelhante ao seguinte.

```
Issuing CreateTable request for TestTableForStreams
Waiting for TestTableForStreams to be created...
Current stream ARN for TestTableForStreams: arn:aws:dynamodb:us-east-2:123456789012:table/
TestTableForStreams/stream/2018-03-20T16:49:55.208
Stream enabled: true
Update view type: NEW_AND_OLD_IMAGES

Performing write activities on TestTableForStreams
Processing item 1 of 100
Processing item 2 of 100
Processing item 3 of 100
...
Processing item 100 of 100

Shard: {ShardId: shardId-1234567890-..., SequenceNumberRange: {StartingSequenceNumber:
01234567890..., }, }
```

```
Shard iterator: EjYFEkX2a26eVTWe...
    ApproximateCreationDateTime: Tue Mar 20 09:50:00 PDT 2018,Keys:
    {Id={N: 1,}},NewImage: {Message={S: New item!,}, Id={N: 1,}},SequenceNumber:
    100000000003218256368,SizeBytes: 24,StreamViewType: NEW_AND_OLD_IMAGES}
        {ApproximateCreationDateTime: Tue Mar 20 09:50:00 PDT 2018,Keys: {Id={N:
    1,}},NewImage: {Message={S: This item has changed,}, Id={N: 1,}},OldImage:
    {Message={S: New item!,}, Id={N: 1,}},SequenceNumber: 200000000003218256412,SizeBytes:
    56,StreamViewType: NEW_AND_OLD_IMAGES}
            {ApproximateCreationDateTime: Tue Mar 20 09:50:00 PDT 2018,Keys: {Id={N:
    1,}},OldImage: {Message={S: This item has changed,}, Id={N: 1,}},SequenceNumber:
    300000000003218256413,SizeBytes: 36,StreamViewType: NEW_AND_OLD_IMAGES}
...
Deleting the table...
Demo complete
```

Example

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreams;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeAction;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.AttributeValueUpdate;
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;
import com.amazonaws.services.dynamodbv2.model.DescribeStreamRequest;
import com.amazonaws.services.dynamodbv2.model.DescribeStreamResult;
import com.amazonaws.services.dynamodbv2.model.DescribeTableResult;
import com.amazonaws.services.dynamodbv2.model.GetRecordsRequest;
import com.amazonaws.services.dynamodbv2.model.GetRecordsResult;
import com.amazonaws.services.dynamodbv2.model.GetShardIteratorRequest;
import com.amazonaws.services.dynamodbv2.model.GetShardIteratorResult;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.services.dynamodbv2.model.Record;
import com.amazonaws.services.dynamodbv2.model.Shard;
```

```
import com.amazonaws.services.dynamodbv2.model.ShardIteratorType;
import com.amazonaws.services.dynamodbv2.model.StreamSpecification;
import com.amazonaws.services.dynamodbv2.model.StreamViewType;
import com.amazonaws.services.dynamodbv2.util.TableUtils;

public class StreamsLowLevelDemo {

    public static void main(String args[]) throws InterruptedException {

        AmazonDynamoDB dynamoDBClient = AmazonDynamoDBClientBuilder
            .standard()
            .withRegion(Regions.US_EAST_2)
            .withCredentials(new DefaultAWSCredentialsProviderChain())
            .build();

        AmazonDynamoDBStreams streamsClient =
            AmazonDynamoDBStreamsClientBuilder
                .standard()
                .withRegion(Regions.US_EAST_2)
                .withCredentials(new DefaultAWSCredentialsProviderChain())
                .build();

        // Create a table, with a stream enabled
        String tableName = "TestTableForStreams";

        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>(
            Arrays.asList(new AttributeDefinition()
                .withAttributeName("Id")
                .withAttributeType("N")));

        ArrayList<KeySchemaElement> keySchema = new ArrayList<>(
            Arrays.asList(new KeySchemaElement()
                .withAttributeName("Id")
                .withKeyType(KeyType.HASH))); // Partition key

        StreamSpecification streamSpecification = new StreamSpecification()
            .withStreamEnabled(true)
            .withStreamViewType(StreamViewType.NEW_AND_OLD_IMAGES);

        CreateTableRequest createTableRequest = new
        CreateTableRequest().withTableName(tableName)
            .withKeySchema(keySchema).withAttributeDefinitions(attributeDefinitions)
            .withProvisionedThroughput(new ProvisionedThroughput()
                .withReadCapacityUnits(10L)
                .withWriteCapacityUnits(10L))
            .withStreamSpecification(streamSpecification);

        System.out.println("Issuing CreateTable request for " + tableName);
        dynamoDBClient.createTable(createTableRequest);
        System.out.println("Waiting for " + tableName + " to be created...");

        try {
            TableUtils.waitUntilActive(dynamoDBClient, tableName);
        } catch (AmazonClientException e) {
            e.printStackTrace();
        }

        // Print the stream settings for the table
        DescribeTableResult describeTableResult = dynamoDBClient.describeTable(tableName);
        String streamArn = describeTableResult.getTable().getLatestStreamArn();
        System.out.println("Current stream ARN for " + tableName + ": " +
            describeTableResult.getTable().getLatestStreamArn());
        StreamSpecification streamSpec =
        describeTableResult.getTable().getStreamSpecification();
        System.out.println("Stream enabled: " + streamSpec.getStreamEnabled());
        System.out.println("Update view type: " + streamSpec.getStreamViewType());
    }
}
```

```
System.out.println();

// Generate write activity in the table

System.out.println("Performing write activities on " + tableName);
int maxItemCount = 100;
for (Integer i = 1; i <= maxItemCount; i++) {
    System.out.println("Processing item " + i + " of " + maxItemCount);

    // Write a new item
    Map<String,AttributeValue> item = new HashMap<>();
    item.put("Id", new AttributeValue().withN(i.toString()));
    item.put("Message", new AttributeValue().withS("New item!"));
    dynamoDBClient.putItem(tableName, item);

    // Update the item
    Map<String,AttributeValue> key = new HashMap<>();
    key.put("Id", new AttributeValue().withN(i.toString()));
    Map<String,AttributeValueUpdate> attributeUpdates = new HashMap<>();
    attributeUpdates.put("Message", new AttributeValueUpdate()
        .withAction(AttributeAction.PUT)
        .WithValue(new AttributeValue()
            .withS("This item has changed")));
    dynamoDBClient.updateItem(tableName, key, attributeUpdates);

    // Delete the item
    dynamoDBClient.deleteItem(tableName, key);
}

// Get all the shard IDs from the stream. Note that DescribeStream returns
// the shard IDs one page at a time.
String lastEvaluatedShardId = null;

do {
    DescribeStreamResult describeStreamResult = streamsClient.describeStream(
        new DescribeStreamRequest()
            .withStreamArn(streamArn)
            .withExclusiveStartShardId(lastEvaluatedShardId));
    List<Shard> shards = describeStreamResult.getStreamDescription().getShards();

    // Process each shard on this page

    for (Shard shard : shards) {
        String shardId = shard.getShardId();
        System.out.println("Shard: " + shard);

        // Get an iterator for the current shard

        GetShardIteratorRequest getShardIteratorRequest = new
GetShardIteratorRequest()
            .withStreamArn(streamArn)
            .withShardId(shardId)
            .withShardIteratorType(ShardIteratorType.TRIM_HORIZON);
        GetShardIteratorResult getShardIteratorResult =
            streamsClient.getShardIterator(getShardIteratorRequest);
        String currentShardIter = getShardIteratorResult.getShardIterator();

        // Shard iterator is not null until the Shard is sealed (marked as
READ_ONLY).
        // To prevent running the loop until the Shard is sealed, which will be on
average
        // 4 hours, we process only the items that were written into DynamoDB and
then exit.
        int processedRecordCount = 0;
        while (currentShardIter != null && processedRecordCount < maxItemCount) {
```

```
        System.out.println("    Shard iterator: " +
currentShardIter.substring(380));

        // Use the shard iterator to read the stream records

        GetRecordsResult getRecordsResult = streamsClient.getRecords(new
GetRecordsRequest()
            .withShardIterator(currentShardIter));
        List<Record> records = getRecordsResult.getRecords();
        for (Record record : records) {
            System.out.println("        " + record.getDynamodb());
        }
        processedRecordCount += records.size();
        currentShardIter = getRecordsResult.getNextShardIterator();
    }
}

// If LastEvaluatedShardId is set, then there is
// at least one more page of shard IDs to retrieve
lastEvaluatedShardId =
describeStreamResult.getStreamDescription().getLastEvaluatedShardId();

} while (lastEvaluatedShardId != null);

// Delete the table
System.out.println("Deleting the table...");
dynamoDBClient.deleteTable(tableName);

System.out.println("Demo complete");

}
}
```

DynamoDB Streams eAWS LambdaGatilhos

Tópicos

- [Tutorial: Processe novos itens com o DynamoDB Streams e o Lambda \(p. 673\)](#)
- [Melhores práticas do Lambda \(p. 679\)](#)

O Amazon DynamoDB é integrado ao AWS Lambda para que você possa criar Gatilhos do— pedaços de código que respondem automaticamente a eventos no DynamoDB Streams. Com triggers, você pode criar aplicativos que reagem às modificações de dados em tabelas do DynamoDB.

Se você habilitar o DynamoDB Streams em uma tabela, poderá associar o Nome de recurso da Amazon (ARN) do stream a um AWS Lambda que você escreve. Imediatamente após um item da tabela ser modificado, um novo registro aparece no stream da tabela. O AWS Lambda consulta o stream e invoca sua função Lambda de forma síncrona ao detectar novos registros de stream.

A função do Lambda pode realizar qualquer ação que você especificar, como enviar uma notificação ou iniciar um fluxo de trabalho. Por exemplo, você pode gravar uma função do Lambda para simplificar a cópia de cada registro de stream para o armazenamento persistente, como o Amazon Simple Storage Service (Amazon S3), para criar uma trilha de auditoria permanente de atividades de gravação em sua tabela. Ou suponhamos que você tenha um aplicativo de jogos móveis que grava em uma tabela GameScores. Sempre que o atributo TopScore da tabela GameScores é atualizado, um registro de stream correspondente é gravado no stream da tabela. Este evento poderia, em seguida, acionar uma função do Lambda que posta uma mensagem de felicitações em uma rede de mídia social. (A função simplesmente ignoraria quaisquer registros de stream que não são atualizações para GameScores ou que não modificam o atributo TopScore.)

Para obter mais informações sobre o AWS Lambda, consulte o [Guia do desenvolvedor do AWS Lambda](#).

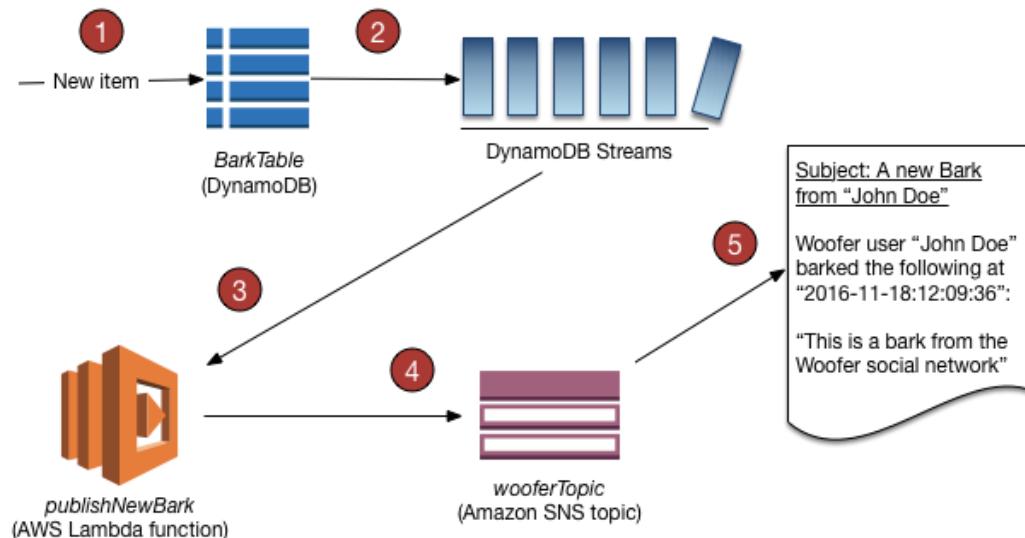
Tutorial: Processe novos itens com o DynamoDB Streams e o Lambda

Tópicos

- [Etapa 1: Crie uma tabela do DynamoDB com um stream habilitado \(p. 674\)](#)
- [Etapa 2: Criar uma função de execução do Lambda \(p. 674\)](#)
- [Etapa 3: Criar um tópico do Amazon SNS \(p. 675\)](#)
- [Etapa 4: Criar e testar uma função do Lambda \(p. 676\)](#)
- [Etapa 5: Criar e testar um trigger \(p. 678\)](#)

Neste tutorial, você cria um AWS Lambda para processar um stream de uma tabela do DynamoDB.

O cenário deste tutorial é o Woofer, uma rede social simples. Os usuários do Woofer se comunicam usando barks (mensagens de texto curtas) que são enviados a outros usuários do Woofer. O diagrama a seguir mostra os componentes e o fluxo de trabalho desse aplicativo.



1. Um usuário grava um item em uma tabela do DynamoDB (`BarkTable`). Cada item na tabela representa um bark.
2. Um novo registro de stream é gravado para refletir que um novo item foi adicionado à `BarkTable`.
3. O novo registro de stream aciona uma função do AWS Lambda (`publishNewBark`).
4. Se o registro de fluxo indicar que um novo item foi adicionado ao `BarkTable`, a função Lambda lê os dados do registro de stream e publica uma mensagem em um tópico no Amazon Simple Notification Service (Amazon SNS).
5. A mensagem é recebida pelos assinantes do tópico do Amazon SNS. (Neste tutorial, o único assinante é um endereço de e-mail.)

Antes de começar

Este tutorial usa a AWS Command Line Interface AWS CLI. Se você ainda não tiver feito isso, siga as instruções do [AWS Command Line Interface Guia do usuário](#) Para instalar e configurar o AWS CLI.

Etapa 1: Crie uma tabela do DynamoDB com um stream habilitado

Nesta etapa, você cria uma tabela do DynamoDB (`BarkTable`) para armazenar todos os barks dos usuários do Woofer. A chave primária é composta de `Username` (chave de partição) e de `Timestamp` (chave de classificação). Ambos os atributos são do tipo string.

`BarkTable` tem um stream habilitado. Mais adiante neste tutorial, você criará um trigger associando uma função do AWS Lambda ao stream.

1. Use o seguinte comando para criar a tabela.

```
aws dynamodb create-table \
    --table-name BarkTable \
    --attribute-definitions AttributeName=Username,AttributeType=S
    AttributeName=Timestamp,AttributeType=S \
    --key-schema AttributeName=Username,KeyType=HASH
    AttributeName=Timestamp,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

2. Na saída, procure o `LatestStreamArn`.

```
...
"LatestStreamArn": "arn:aws:dynamodb:region:accountID:table/BarkTable/stream/timestamp
..."
```

Anote a `region` e o `accountID`, pois eles serão necessários para as outras etapas deste tutorial.

Etapa 2: Criar uma função de execução do Lambda

Nesta etapa, você cria um AWS Identity and Access Management Função do (IAM) (`WooferLambdaRole`) e atribua permissões a ele. Essa função é usada pela função Lambda que você cria em [Etapa 4: Criar e testar uma função do Lambda \(p. 676\)](#).

Você também cria uma política para a função. A política contém todas as permissões de que a função do Lambda precisa em tempo de execução.

1. Crie um arquivo denominado `trust-relationship.json` com o seguinte conteúdo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. Insira o seguinte comando para criar a `WooferLambdaRole`.

```
aws iam create-role --role-name WooferLambdaRole \
    --path "/service-role/" \
    --assume-role-policy-document file://trust-relationship.json
```

3. Crie um arquivo denominado `role-policy.json` com o seguinte conteúdo. (Substituir `region` e `accountID` com suas respectivas AWS Região e ID da conta.)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "lambda:InvokeFunction",  
            "Resource": "arn:aws:lambda:region:accountID:function:publishNewBark"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "logs>CreateLogGroup",  
                "logs>CreateLogStream",  
                "logs:PutLogEvents"  
            ],  
            "Resource": "arn:aws:logs:region:accountID:*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:DescribeStream",  
                "dynamodb:GetRecords",  
                "dynamodb:GetShardIterator",  
                "dynamodb>ListStreams"  
            ],  
            "Resource": "arn:aws:dynamodb:region:accountID:table/BarkTable/stream/*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "sns:Publish"  
            ],  
            "Resource": [  
                "*"  
            ]  
        }  
    ]  
}
```

A política tem quatro declarações que fornecem permissões à `WooferLambdaRole` para fazer o seguinte:

- Execute uma função do Lambda (`publishNewBark`). Você cria a função mais adiante neste tutorial.
 - Acesse o Amazon CloudWatch Logs. A função Lambda grava diagnósticos no CloudWatch Logs em tempo de execução.
 - Leia os dados do stream do DynamoDB para o `BarkTable`.
 - Publique mensagens no Amazon SNS.
4. Execute o seguinte comando para anexar a política à função `WooferLambdaRole`.

```
aws iam put-role-policy --role-name WooferLambdaRole \  
    --policy-name WooferLambdaRolePolicy \  
    --policy-document file://role-policy.json
```

Etapa 3: Criar um tópico do Amazon SNS

Nesta etapa, você cria um tópico do Amazon SNS (`wooferTopic`) e inscreva um endereço de e-mail nele. Sua função do Lambda usa esse tópico para publicar novos barks de usuários do Woofer.

1. Insira o comando a seguir para criar um novo tópico do Amazon SNS.

```
aws sns create-topic --name wooferTopic
```

2. Digite o seguinte comando para inscrever um endereço de e-mail no `wooferTopic`.
(Substituir `region` e `accountID`Com suas receitasAWSID da região e da conta e substituir `example@example.com`Com um endereço de e-mail válido.)

```
aws sns subscribe \  
    --topic-arn arn:aws:sns:region:accountID:wooferTopic \  
    --protocol email \  
    --notification-endpoint example@example.com
```

3. O Amazon SNS envia uma mensagem de confirmação para o seu endereço de e-mail. Selecione o link Confirm subscription (Confirmar assinatura) na mensagem para concluir o processo de assinatura.

Etapa 4: Criar e testar uma função do Lambda

Nesta etapa, você cria uma função do AWS Lambda (`publishNewBark`) para processar registros de stream da `BarkTable`.

A função `publishNewBark` processa apenas os eventos de stream que correspondem a novos itens na `BarkTable`. A função lê dados de um evento como esse e, em seguida, invoca o Amazon SNS para publicá-lo.

1. Crie um arquivo denominado `publishNewBark.js` com o seguinte conteúdo.
Substituir `region` e `accountID`Com suas receitasAWSRegião e ID da conta.

```
'use strict';  
var AWS = require("aws-sdk");  
var sns = new AWS.SNS();  
  
exports.handler = (event, context, callback) => {  
  
    event.Records.forEach((record) => {  
        console.log('Stream record: ', JSON.stringify(record, null, 2));  
  
        if (record.eventName == 'INSERT') {  
            var who = JSON.stringify(record.dynamodb.NewImage.Username.S);  
            var when = JSON.stringify(record.dynamodb.NewImage.Timestamp.S);  
            var what = JSON.stringify(record.dynamodb.NewImage.Message.S);  
            var params = {  
                Subject: 'A new bark from ' + who,  
                Message: 'Woofer user ' + who + ' barked the following at ' + when + ':  
\n\n' + what,  
                TopicArn: 'arn:aws:sns:region:accountID:wooferTopic'  
            };  
            sns.publish(params, function(err, data) {  
                if (err) {  
                    console.error("Unable to send message. Error JSON:",  
JSON.stringify(err, null, 2));  
                } else {  
                    console.log("Results from sending message: ", JSON.stringify(data,  
null, 2));  
                }  
            });  
        }  
    });  
    callback(null, `Successfully processed ${event.Records.length} records.`);  
};
```

2. Crie um arquivo zip para conter `publishNewBark.js`. Se você tiver o utilitário de linha de comando `zip`, poderá digitar o seguinte comando para fazer isso.

```
zip publishNewBark.zip publishNewBark.js
```

3. Ao criar a função do Lambda, você especifica o nome de recurso da Amazon (ARN) para `WoofLambdaRole`, que você criou no [Etapa 2: Criar uma função de execução do Lambda \(p. 674\)](#). Digite o seguinte comando para recuperar o ARN.

```
aws iam get-role --role-name WoofLambdaRole
```

Na saída, procure o ARN da `WoofLambdaRole`.

```
...
"Arn": "arn:aws:iam::region:role/service-role/WoofLambdaRole"
...
```

Digite o comando a seguir para criar a função Lambda. Substitua `roleARN` pelo ARN da `WoofLambdaRole`.

```
aws lambda create-function \
--region region \
--function-name publishNewBark \
--zip-file file:///publishNewBark.zip \
--role roleARN \
--handler publishNewBark.handler \
--timeout 5 \
--runtime nodejs10.x
```

4. Agora teste o `publishNewBark` para verificar se ele funciona. Para fazer isso, você fornece informações que se parecem com um registro real do DynamoDB Streams.

Crie um arquivo denominado `payload.json` com o seguinte conteúdo.

```
{
    "Records": [
        {
            "eventID": "7de3041dd709b024af6f29e4fa13d34c",
            "eventName": "INSERT",
            "eventVersion": "1.1",
            "eventSource": "aws:dynamodb",
            "awsRegion": "region",
            "dynamodb": {
                "ApproximateCreationDateTime": 1479499740,
                "Keys": {
                    "Timestamp": {
                        "S": "2016-11-18:12:09:36"
                    },
                    "Username": {
                        "S": "John Doe"
                    }
                },
                "NewImage": {
                    "Timestamp": {
                        "S": "2016-11-18:12:09:36"
                    },
                    "Message": {
                        "S": "This is a bark from the Woof social network"
                    },
...
```

```
        "Username": {  
            "S": "John Doe"  
        }  
    },  
    "SequenceNumber": "1302160000000001596893679",  
    "SizeBytes": 112,  
    "StreamViewType": "NEW_IMAGE"  
},  
"eventSourceARN": "arn:aws:dynamodb:region:123456789012:table/BarkTable/  
stream/2016-11-16T20:42:48.104"  
}  
]  
}
```

Use o seguinte comando para testar a função `publishNewBark`.

```
aws lambda invoke --function-name publishNewBark --payload file://payload.json  
output.txt
```

Se o teste for bem-sucedido, você verá a seguinte saída.

```
{  
    "StatusCode": 200  
}
```

Além disso, o arquivo `output.txt` conterá o seguinte texto.

```
"Successfully processed 1 records."
```

Você também receberá uma nova mensagem de e-mail dentro de alguns minutos.

Note

AWS Lambda grava informações de diagnóstico nos Amazon CloudWatch Logs. Se você encontrar erros em sua função Lambda, poderá usar essas informações de diagnósticos para fins de solução de problemas:

1. Abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de logs a seguir: /aws/lambda/publishNewBark
4. Escolha o stream de logs mais recente para visualizar a saída (e os erros) da função.

Etapa 5: Criar e testar um trigger

Dentro [Etapa 4: Criar e testar uma função do Lambda \(p. 676\)](#), você testou a função Lambda para garantir que ela foi executada corretamente. Nesta etapa, você cria um trigger associando a função do Lambda (`publishNewBark`) com uma fonte de evento (o `BarkTable` stream).

1. Ao criar o trigger, você precisa especificar o ARN do stream de `BarkTable`. Digite o seguinte comando para recuperar o ARN.

```
aws dynamodb describe-table --table-name BarkTable
```

Na saída, procure o `LatestStreamArn`.

```
...
```

```
"LatestStreamArn": "arn:aws:dynamodb:region:accountID:table/BarkTable/stream/timestamp  
..."
```

2. Insira o seguinte comando para criar o trigger. Substitua **streamARN** pelo ARN do stream atual.

```
aws lambda create-event-source-mapping \  
  --region region \  
  --function-name publishNewBark \  
  --event-source streamARN \  
  --batch-size 1 \  
  --starting-position TRIM_HORIZON
```

3. Teste o trigger. Insira o seguinte comando para adicionar um item a BarkTable.

```
aws dynamodb put-item \  
  --table-name BarkTable \  
  --item Username={"S":"Jane  
Doe"},Timestamp={"S":"2016-11-18:14:32:17"},Message={"S":"Testing...1...2...3"}
```

Você deve receber uma nova mensagem de e-mail dentro de alguns minutos.

4. Abra o console do DynamoDB e adicione mais alguns itens a BarkTable. Você deve especificar valores para os atributos Username e Timestamp. (Você também deve especificar um valor para Message, embora isso não seja obrigatório.) Você deve receber uma nova mensagem de e-mail para cada item que adicionar a BarkTable.

A função Lambda processa apenas novos itens que você adiciona a BarkTable. Se você atualizar ou excluir um item na tabela, a função não faz nada.

Note

AWS Lambda grava informações de diagnóstico nos Amazon CloudWatch Logs. Se você encontrar erros em sua função Lambda, poderá usar essas informações de diagnósticos para fins de solução de problemas.

1. Abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de logs a seguir: /aws/lambda/publishNewBark
4. Escolha o stream de logs mais recente para visualizar a saída (e os erros) da função.

Melhores práticas do Lambda

Uma função AWS Lambda é executada em um contêiner - um ambiente de execução isolado de outras funções. Quando você executa uma função pela primeira vez, o AWS Lambda cria um novo contêiner e começa a executar o código da função.

Uma função Lambda tem um handler que é executado uma vez por invocação. O identificador contém a lógica de negócios principal da função. Por exemplo, a função do Lambda mostrada em [Etapa 4: Criar e testar uma função do Lambda \(p. 676\)](#) tem um identificador que pode processar registros em um stream do DynamoDB.

Você também pode fornecer o código de inicialização que é executado apenas uma vez - depois que o contêiner é criado, mas antes que o AWS Lambda execute o manipulador pela primeira vez. A função Lambda mostrada no [Etapa 4: Criar e testar uma função do Lambda \(p. 676\)](#) possui um código de inicialização que importa o SDK for JavaScript in Node.js e cria um cliente para o Amazon SNS. Node.js Esse objeto devem ser definidos somente uma vez, fora do identificador.

Depois que a função é executada, o AWS Lambda pode optar por reutilizar o contêiner para chamadas subsequentes da função. Neste caso, o identificador da função pode reutilizar os recursos que você definiu no seu código de inicialização. (Você não pode controlar por quanto tempo o AWS Lambda reterá o contêiner, ou se o contêiner será reutilizado.)

Para gatilhos do DynamoDB usando AWS Lambda, recomendamos o seguinte:

- AWSOs clientes de serviço devem ser instanciados no código de inicialização, não no manipulador. Isso permite que o AWS Lambda reutilize conexões existentes, durante o ciclo de vida do contêiner.
- Em geral, você não precisa gerenciar explicitamente as conexões ou implementar o pool de conexões porque o AWS Lambda gerencia isso para você.

Para obter mais informações, consulte [Melhores práticas para trabalhar com o AWS Lambda](#) [Funções](#) no AWS Lambda Guia do desenvolvedor.

Gerenciar fluxos de trabalho complexos com transações do DynamoDB

Transações do Amazon DynamoDB simplificam a experiência do desenvolvedor ao fazer alterações de tudo ou nada em vários itens dentro e entre tabelas. Transações fornecem atomicidade, consistência, isolamento e durabilidade (ACID) no DynamoDB, ajudando você a manter a exatidão dos dados em seus aplicativos.

Você pode usar as DynamoDB de leitura e gravação transacionais de para gerenciar fluxos de trabalho empresariais complexos que precisão de adição, atualização ou exclusão de vários itens como uma única operação tudo ou nada. Por exemplo, um desenvolvedor de videogames pode garantir que os perfis dos jogadores sejam atualizados corretamente quando eles trocam itens em um jogo ou fazem compras dentro do jogo.

Com a API de gravação de transações, você pode agrupar várias ações `Put`, `Update`, `Delete` e `ConditionCheck`. Você pode enviar as ações como uma única operação `TransactWriteItems` que é bem-sucedida ou falha como uma unidade. O mesmo é verdade para várias ações `Get`, que você pode agrupar e enviar como uma única operação `TransactGetItems`.

Não há custo adicional para habilitar transações para suas tabelas do DynamoDB. Você paga apenas pelas leituras ou gravações que fazem parte de sua transação. O DynamoDB realiza duas leituras subjacentes ou gravações de cada item na transação: uma para preparar a transação e uma para enviar a transação. Essas duas operações de leitura/gravação subjacente são visíveis nas métricas do Amazon CloudWatch.

Para começar a usar as transações do DynamoDB AWS SDK ou o AWS Command Line Interface(AWS CLI). Em seguida, siga o [Exemplo de transações do DynamoDB \(p. 688\)](#).

As seções a seguir fornecem uma visão geral detalhada das APIs de transação e como você pode usá-las no DynamoDB.

Tópicos

- [Amazon DynamoDB Transactions: Como ele funciona \(p. 681\)](#)
- [Uso do IAM com Transações do DynamoDB \(p. 686\)](#)
- [Exemplo de transações do DynamoDB \(p. 688\)](#)

Amazon DynamoDB Transactions: Como ele funciona

Com transações do Amazon DynamoDB, você pode agrupar várias ações junto e enviá-las como um único tudo ou nadaTransactWriteItemsouTransactGetItemsoperação. As seções a seguir descrevem operações API, gerenciamento de capacidade, melhores práticas e outros detalhes sobre usar operações transacionais no DynamoDB.

Tópicos

- [API TransactWriteItems \(p. 681\)](#)
- [API TransactGetItems \(p. 682\)](#)
- [Níveis de isolamento para transações do DynamoDB \(p. 683\)](#)
- [Lidar com conflitos de transação no DynamoDB \(p. 684\)](#)
- [Uso de APIs transacionais no DynamoDB Accelerator \(DAX\) \(p. 685\)](#)
- [Gerenciamento de capacidade para transações \(p. 685\)](#)
- [Melhores práticas para transações \(p. 686\)](#)
- [Uso de APIs Transacionais com Tabelas Globais \(p. 686\)](#)
- [Transações do DynamoDB \(p. 686\)](#)

API TransactWriteItems

`TransactWriteItems` é uma operação de gravação síncrona e idempotente que agrupa até 25 ações de gravação em uma única operação tudo ou nada. Essas ações podem ter como destino até 25 itens distintos em uma ou mais tabelas do DynamoDB dentro do mesmoAWSe na mesma região. O tamanho agregado dos itens na transação não podem exceder 4 MB. As ações são concluídas de forma atômica para que todas sejam bem-sucedidas ou nenhuma tenha êxito.

Uma operação `TransactWriteItems` difere de uma operação `BatchWriteItem` em que todas as ações contidas devem ser concluídas com êxito, ou nenhuma alteração foi feita. Com uma operação `BatchWriteItem`, é possível que apenas as ações no lote sejam bem-sucedidas enquanto as outras não.

Você não pode visar o mesmo item com várias operações dentro da mesma transação. Por exemplo, você não pode executar uma ação `ConditionCheck` e também uma `Update` no mesmo item na mesma transação.

Você pode adicionar os tipos a seguir de ações a uma transação:

- `Put`— Inicia um`PutItem`Operação para criar um novo item ou substituir um item antigo por um novo item, de forma condicional ou sem especificar nenhuma condição.
- `Update`— Inicia um`UpdateItem`Operação para editar um atributo de item existente ou adicionar um novo item à tabela, se ele ainda não existir. Use essa ação para adicionar, excluir ou atualizar atributos ou um item existente de forma condicional ou sem uma condição.
- `Delete`— Inicia um`DeleteItem`Operação para excluir um único item em uma tabela identificada por essa chave principal.
- `ConditionCheck`— Verifica se um item existe ou verifica a condição de atributos específicos do item.

Quando uma transação é concluída, as alterações feitas com essa transação são propagadas para índices secundários globais (GSIs), streams e backups. Como a propagação não é imediata ou instantânea, se uma tabela for restaurada do backup de um ponto no meio da transação, ela poderá conter algumas, mas não todas as alterações feitas durante uma transação recente.

Idempotency

Você pode, opcionalmente, incluir um token de cliente quando fizer uma chamada `TransactWriteItems` para garantir que a solicitação é idempotente. Realizar transações idempotentes ajuda a prevenir erros de aplicação se a mesma operação for enviada diversas vezes por conta de uma desconexão ou outro problema de conexão.

Se a chamada `TransactWriteItems` original for bem-sucedida, as chamadas `TransactWriteItems` subsequentes com o mesmo token de cliente retornarão com êxito sem nenhuma alteração. Se o parâmetro `ReturnConsumedCapacity` estiver definido, a chamada `TransactWriteItems` inicial retornará o número de unidades de capacidade de gravação consumidas para realizar as alterações. Chamadas `TransactWriteItems` subsequentes com o mesmo token de cliente devolvem o número de unidades de capacidade de leitura consumidas na leitura do item.

Pontos importantes sobre potência igual

- Um token de cliente é válido por 10 minutos após a solicitação utilizada acabar. Depois de 10 minutos, quaisquer solicitações que usarem o mesmo token de cliente são tratadas como uma nova solicitação. Você não deve reutilizar o mesmo token de cliente para a mesma solicitação após 10 minutos.
- Se você repetir uma solicitação com o mesmo token de cliente dentro de 10 minutos na janela de idempotência, mas alterar algum parâmetro de solicitação, o DynamoDB retorna uma `IdempotentParameterMismatchException`.

Manipulação de erros para gravação

Transações de gravação não são bem-sucedidas nas circunstâncias a seguir:

- Quando uma condição em uma das expressões de condição não é alcançada.
- Quando uma validação de transação ocorrer por conta de mais de uma ação na mesma operação `TransactWriteItems` ter como destino o mesmo item.
- Quando uma solicitação `TransactWriteItems` entra em conflito com uma operação `TransactWriteItems` em andamento em um ou mais itens na operação `TransactWriteItems`. Nesse caso, a solicitação falha com um `TransactionCanceledException`.
- Quando há capacidade provisionada insuficiente para a transação ser concluída.
- Quando o tamanho de um item é muito maior (maior que 400 KB), ou um índice secundário local (LSI) se torna muito grande, ou um erro de validação semelhante ocorre por conta das alterações feitas pela transação.
- Quando houver um erro de usuário, como formato de dados inválidos.

Para obter mais informações sobre como os conflitos com operações `TransactWriteItems` são gerenciados, consulte [Lidar com conflitos de transação no DynamoDB \(p. 684\)](#).

API TransactGetItems

`TransactGetItems` é uma operação de leitura síncrona que agrupa até 25 ações Get. Essas ações podem ter como destino até 25 itens distintos em uma ou mais tabelas do DynamoDB dentro do mesmo AWS Conta e região. O tamanho agregado dos itens na transação não pode exceder 4 MB.

As ações Get são realizadas de forma atômica para que todas sejam bem-sucedidas ou nenhuma tenha êxito:

- Get— Inicia um `GetItem` para recuperar um conjunto de atributos para o item com a chave primária fornecida. Se não houver item correspondente, Get não retornará quaisquer dados.

Manipulação de erros para leitura

Transações de leitura não são bem-sucedidas nas circunstâncias a seguir:

- Quando uma solicitação `TransactGetItems` entra em conflito com uma operação `TransactWriteItems` em andamento em um ou mais itens na operação `TransactGetItems`. Nesse caso, a solicitação falha com um `TransactionCanceledException`.
- Quando há capacidade provisionada insuficiente para a transação ser concluída.
- Quando houver um erro de usuário, como formato de dados inválidos.

Para obter mais informações sobre como os conflitos com operações `TransactGetItems` são gerenciados, consulte [Lidar com conflitos de transação no DynamoDB](#) (p. 684).

Níveis de isolamento para transações do DynamoDB

Os níveis de isolamento de operações transacionais (`TransactWriteItems` ou `TransactGetItems`) e outras operações são os seguintes.

SERIALIZABLE

Isolamento Serializável garante que os resultados de várias operações concomitantes sejam os mesmos como se nenhuma operação começar com o anterior finalizado.

Há um isolamento serializável entre os tipos a seguir de operação:

- Entre qualquer operação transacional e qualquer padrão de operação de gravação (`PutItem`, `UpdateItem` ou `DeleteItem`).
- Entre qualquer operação transacional e qualquer padrão de operação de leitura (`GetItem`).
- Entre uma operação `TransactWriteItems` e `TransactGetItems`.

Apesar de haver isolamento serializável entre operações transacionais, e cada padrão individual de gravação em uma operação `BatchWriteItem`, não há isolamento serializável entre a transação e a operação `BatchWriteItem` como uma unidade.

De modo semelhante, o nível de isolamento entre uma operação transacional e `GetItems` individual em uma operação `BatchGetItem` é serializável. Porém, o nível de isolamento entre a transação e a operação `BatchGetItem` como uma unidade é confirmado para leitura.

Um único `GetItem` é serializável em relação a um `TransactWriteItems`. Solicite de duas formas, antes ou depois da `TransactWriteItem` solicitação. Vários `GetItem` solicitações, contra chaves em um `TransactWriteItem` solicitações podem ser executadas em qualquer ordem e, portanto, os resultados são confirmados para leitura.

Por exemplo, se `GetItem` solicitações para o item A e o item B são executadas simultaneamente com um `TransactWriteItems` que modifica o item A e o item B, existem quatro possibilidades:

- Ambos `GetItem`s são executadas antes da `TransactWriteItem` solicitação.
- Ambos `GetItem` solicitações são executadas após a `TransactWriteItem` solicitação.
- `GetItem` solicitação para o item A é executada antes da `TransactWriteItem` solicitação. Para o item B, o `GetItem` é executado após `TransactWriteItems`.
- `GetItem` solicitação para o item B é executada antes da `TransactWriteItem` solicitação. Para o item A, o `GetItem` é executado após `TransactWriteItems`.

Se o nível de isolamento serializável for preferido para vários `GetItem`s pedidos, em seguida, por favor use `TransactGetItems`.

CONFIRMADO PARA LEITURA

Isolamento confirmado para leitura garante que operações de leitura sempre retornem valores confirmados de um item. Isolamento confirmado para leitura não previne modificações do item imediatamente após operação de leitura.

O nível de isolamento é confirmado para leitura entre qualquer operação transacional e qualquer operação de leitura que envolva vários padrões de leitura (`BatchGetItem`, `Query` ou `Scan`). Se uma gravação transacional atualizar um item no meio de uma operação `BatchGetItem`, `Query` ou `Scan`, a operação de leitura retorna o novo valor confirmado.

Resumo da operação

Para resumir, a tabela a seguir mostra os níveis de isolamento entre uma operação transacional (`TransactWriteItems` ou `TransactGetItems`) e outras operações.

Operação	Nível de isolamento
<code>DeleteItem</code>	Serializável
<code>PutItem</code>	Serializável
<code>UpdateItem</code>	Serializável
<code>GetItem</code>	Serializável
<code>BatchGetItem</code>	Confirmado para leitura*
<code>BatchWriteItem</code>	NÃO serializável*
<code>Query</code>	Confirmado para leitura
<code>Scan</code>	Confirmado para leitura
Outra operação transacional	Serializável

Níveis marcados com asterisco (*) aplicam-se à operação como uma unidade. No entanto, ações individuais dentro dessas operações tem um nível de isolamento serializável.

Lidar com conflitos de transação no DynamoDB

Um conflito de transação pode correr em solicitações simultâneas no nível do item, em um item dentro de uma transação. Os conflitos de transação podem ocorrer nos seguintes casos:

- Uma solicitação `PutItem`, `UpdateItem` ou `DeleteItem` de um item conflita com uma solicitação `TransactWriteItems` em andamento que inclui o mesmo item.
- Um item dentro de uma solicitação `TransactWriteItems` é parte de outra solicitação `TransactWriteItems` em andamento.
- Um item dentro de uma solicitação `TransactGetItems` é parte de outra solicitação `TransactWriteItems`, `BatchWriteItem`, `PutItem`, `UpdateItem` ou `DeleteItem` em andamento.

Note

- Quando uma solicitação `PutItem`, `UpdateItem` ou `DeleteItem` é rejeitada, a solicitação falha com um `TransactionConflictException`,

- Se qualquer solicitação no nível do item dentro de `TransactWriteItems` ou `TransactGetItems` é rejeitada, a solicitação falha com um `TransactionCanceledException`. Se essa solicitação falhar, AWSSDKs não repetem a solicitação.

Se você estiver usando AWS SDK for Java, a exceção conterá a lista de `CancellationReasons`, em ordem de acordo com a lista de itens no parâmetro de solicitação `TransactItems`. Para outros idiomas, uma representação em string da lista é incluída na mensagem de erro de exceção.

- Se uma operação `TransactWriteItems` ou `TransactGetItems` em andamento entrar em conflito com uma solicitação `GetItem` simultânea, as duas podem ter êxito.

A métrica `TransactionConflict` do CloudWatch é incrementada em cada solicitação que falhou no nível do item

Uso de APIs transacionais no DynamoDB Accelerator (DAX)

`TransactWriteItem`s e `TransactGetItems`s são compatíveis com o DynamoDB Accelerator (DAX) com os mesmos níveis de isolamento do DynamoDB.

`TransactWriteItem`s gravam através do DAX. DAX transmite um `TransactWriteItem` Chamada para o DynamoDB e retorna a resposta. Para preencher o cache após a gravação, o DAX chama `TransactGetItem`s em segundo plano para cada item no `TransactWriteItem` Operação, que consome unidades de capacidade de leitura adicionais. (Para obter mais informações, consulte [Gerenciamento de capacidade para transações \(p. 685\)](#).) Essa funcionalidade permite manter a lógica do aplicativo simples e usar o DAX para operações transacionais e não transacionais.

`TransactGetItem`s Chamadas de são transmitidas pelo DAX sem que os itens sejam armazenados em cache localmente. Esse é o mesmo comportamento para APIs de leitura fortemente consistente no DAX.

Gerenciamento de capacidade para transações

Não há custo adicional para habilitar transações para suas tabelas do DynamoDB. Você paga apenas pelas leituras ou gravações que fazem parte de sua transação. O DynamoDB realiza duas leituras subjacentes ou gravações de cada item na transação: uma para preparar a transação e uma para enviar a transação. Essas duas operações de leitura/gravação subjacente são visíveis nas métricas do Amazon CloudWatch.

Planeje para leituras e gravações adicionais que são necessárias por APIs transacionais ao provisionar a capacidade de suas tabelas. Por exemplo, suponha que seu aplicativo execute uma transação por segundo, e cada transação grave itens de 500 bytes em sua tabela. Cada item requer duas capacidades de gravação (WCUs): uma para preparar a transação e uma para enviar a transação. Portanto, você precisa provisionar seis WCUs na tabela.

Se você usar o DynamoDB Accelerator (DAX) no exemplo anterior, use também as unidades de capacidade de leitura (RCUs) para cada item no `TransactWriteItem`s Chamado. Então, você precisa provisionar seis RCUs adicionais na tabela.

De forma semelhante, se seu aplicativo executa uma transação de leitura por segundo, e cada transação lê itens de 500 bytes na sua tabela, é preciso prover seis unidades de capacidade de leitura (RCUs) na tabela. Leitura de cada item requer dois RCUs: uma para preparar a transação e uma para enviar a transação.

Além disso, comportamento de SDK padrão é tentar transações novamente em caso de uma exceção `TransactionInProgressException`. Planeje para as unidades de capacidade de leitura adicional (RCUs) que essas tentativas repetitivas consumem. O mesmo é verdade se você estiver tentando novamente transações em seu código usando um `ClientRequestToken`.

Melhores práticas para transações

Considere as seguintes práticas recomendadas ao usar transações do DynamoDB.

- Habilite Auto Scaling nas suas tabelas, ou garantir que você tem capacidade de taxa de transferência suficiente para realizar as duas operações de leitura ou gravação em cada item na transação.
- Se não estiver usando um AWS SDK fornecido, inclua `umClientRequestToken` quando você cria um `atributoTransactWriteItems` para garantir que a solicitação é idempotente.
- Não agrupe operações em uma transação se não for necessário. Por exemplo, se uma transação única com 10 operações puder ser separada em várias transações sem comprometimento da exatidão do aplicativo, recomendamos separar a transação. Transações mais simples melhorar a taxa de rendimento e têm maior chance de êxito.
- Transações múltiplas atualizando os mesmos itens simultaneamente podem causar conflitos que cancelam as transações. Recomendamos as melhores práticas do DynamoDB para modelagem de dados para minimizar esses conflitos.
- Se um conjunto de atributos for frequentemente atualizado entre vários itens como parte de uma única transação, considere agrupar os atributos em um único item para reduzir o escopo da transação.
- Evite usar transações para adicionar dados no grupo. Para gravações em grupo, é melhor usar `BatchWriteItem`.

Uso de APIs Transacionais com Tabelas Globais

As operações transacionais fornecem garantia de atomicidade, consistência, isolamento e durabilidade (ACID) somente na região onde a gravação é realizada originalmente. As transações não são compatíveis entre as regiões em tabelas globais. Por exemplo, se você tiver uma tabela global com réplicas nas regiões Leste dos EUA (Ohio) e Oeste dos EUA (Oregon) e realizar uma operação `TransactWriteItems` na região Leste dos EUA (Norte da Virgínia), poderá observar transações parcialmente concluídas na região Oeste dos EUA (Oregon) à medida que as alterações forem replicadas. As alterações só serão replicadas para outras regiões quando forem confirmadas na região de origem.

Transações do DynamoDB

Transações do DynamoDB fornecem uma substituição mais custo-efetiva e de desempenho para a `oAWSLabs` Biblioteca de cliente de transações. Sugerimos atualizar os aplicativos para usar com as APIs de transação do lado do servidor nativas.

Uso do IAM com Transações do DynamoDB

Você pode usar AWS Identity and Access Management (IAM) para restringir as ações que as operações transacionais podem executar no Amazon DynamoDB. Para obter mais informações sobre como usar as políticas do IAM no DynamoDB, consulte [Uso de políticas baseadas em identidade \(políticas do IAM\) com o Amazon DynamoDB \(p. 888\)](#).

Permissões para `Put`, `Update`, `Delete`, e ações `Get` são governadas pelas permissões usadas para operações `PutItem`, `UpdateItem`, `DeleteItem` e `GetItem` subjacentes. Para o `oConditionCheck` Você pode usar a opção `dynamodb:ConditionCheck` nas políticas do IAM.

Veja a seguir exemplos de políticas do IAM que você pode usar para configurar transações do DynamoDB.

Exemplo 1: Permitir operações transacionais

```
{  
    "Version": "2012-10-17",
```

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "dynamodb:ConditionCheckItem",
            "dynamodb:PutItem",
            "dynamodb:UpdateItem",
            "dynamodb>DeleteItem",
            "dynamodb:GetItem"
        ],
        "Resource": [
            "arn:aws:dynamodb:*:*:table/table04"
        ]
    }
]
```

Exemplo 2: Permitir apenas operações transacionais

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:ConditionCheckItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb>DeleteItem",
                "dynamodb:GetItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:*:*:table/table04"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "dynamodb:EnclosingOperation": [
                        "TransactWriteItems",
                        "TransactGetItems"
                    ]
                }
            }
        }
    ]
}
```

Exemplo 3: Permitir leituras e gravações não transacionais e bloquear leituras e gravações transacionais

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "dynamodb:ConditionCheckItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb>DeleteItem",
                "dynamodb:GetItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:*:*:table/table04"
            ]
        }
    ]
}
```

```
"Resource": [
    "arn:aws:dynamodb:*::table/table04"
],
"Condition": {
    "ForAnyValue:StringEquals": {
        "dynamodb:EnclosingOperation": [
            "TransactWriteItems",
            "TransactGetItems"
        ]
    }
},
{
    "Effect": "Allow",
    "Action": [
        "dynamodb:PutItem",
        "dynamodb>DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:UpdateItem"
    ],
    "Resource": [
        "arn:aws:dynamodb:*::table/table04"
    ]
}
]
```

Exemplo 4: Impedir que informações sejam devolvidas em uma falha ConditionCheck

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:ConditionCheckItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb>DeleteItem",
                "dynamodb:GetItem"
            ],
            "Resource": "arn:aws:dynamodb:*::table/table01",
            "Condition": {
                "StringEqualsIfExists": {
                    "dynamodb:ReturnValues": "NONE"
                }
            }
        }
    ]
}
```

Exemplo de transações do DynamoDB

Como exemplo de uma situação em que as transações do Amazon DynamoDB podem ser úteis, considere este exemplo de aplicativo Java para um marketplace on-line.

O aplicativo tem três tabelas do DynamoDB no backend:

- **Customers**— Esta tabela armazena detalhes sobre os clientes do marketplace. Sua chave primária é `customerId` identificador exclusivo.

- **ProductCatalog**— Esta tabela armazena detalhes como preço e disponibilidade sobre os produtos para venda no mercado. Sua chave primária é um `ProductId` identificador exclusivo.
- **Orders**— Esta tabela armazena detalhes sobre pedidos da loja. Sua chave primária é um `OrderID` identificador exclusivo.

Fazer um pedido

Os trechos de código a seguir ilustram como usar transações do DynamoDB para coordenar as várias etapas necessárias para criar e processar um pedido. O uso de uma única operação de tudo ou nada garante que, se qualquer parte da transação falhar, nenhuma ação na transação será executada e nenhuma alteração será feita.

Neste exemplo, você configura uma ordem de um cliente cujo `customerId` é `09e8e9c8-ec48`. Em seguida, executá-lo como uma única transação usando o seguinte fluxo de trabalho simples de processamento de pedidos:

1. Determine se a ID de cliente é válida.
2. Verifique se o produto é `IN_STOCK` atualize o status do produto para `SOLD`.
3. Certifique-se de que a ordem ainda não existe e crie a ordem.

Validar o cliente

Primeiro, defina uma ação para verificar se um cliente com `customerId` igual a `a09e8e9c8-ec48` existe na tabela de clientes.

```
final String CUSTOMER_TABLE_NAME = "Customers";
final String CUSTOMER_PARTITION_KEY = "CustomerId";
final String customerId = "09e8e9c8-ec48";
final HashMap<String, AttributeValue> customerItemKey = new HashMap<>();
customerItemKey.put(CUSTOMER_PARTITION_KEY, new AttributeValue(customerId));

ConditionCheck checkCustomerValid = new ConditionCheck()
    .withTableName(CUSTOMER_TABLE_NAME)
    .withKey(customerItemKey)
    .withConditionExpression("attribute_exists(" + CUSTOMER_PARTITION_KEY + ")");
```

Atualizar o status do produto

Em seguida, defina uma ação para atualizar o status do produto para `SOLD` se a condição de que o status do produto está atualmente definido como `IN_STOCK` é `true`. Definindo a opção `ReturnValuesOnConditionCheckFailure` retorna o item se o atributo de status do produto do item não for igual a `IN_STOCK`.

```
final String PRODUCT_TABLE_NAME = "ProductCatalog";
final String PRODUCT_PARTITION_KEY = "ProductId";
HashMap<String, AttributeValue> productItemKey = new HashMap<>();
productItemKey.put(PRODUCT_PARTITION_KEY, new AttributeValue(productKey));

Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
expressionAttributeValues.put(":new_status", new AttributeValue("SOLD"));
expressionAttributeValues.put(":expected_status", new AttributeValue("IN_STOCK"));

Update markItemSold = new Update()
    .withTableName(PRODUCT_TABLE_NAME)
    .withKey(productItemKey)
    .withUpdateExpression("SET ProductStatus = :new_status")
    .withReturnValuesOnConditionCheckFailure(true);
```

```
.withExpressionAttributeValues(expressionAttributeValues)
.withConditionExpression("ProductStatus = :expected_status")
.withReturnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD);
```

Criar a Ordem

Por último, crie a ordem, desde que uma ordem com essa orderId ainda não existe.

```
final String ORDER_PARTITION_KEY = "OrderId";
final String ORDER_TABLE_NAME = "Orders";

HashMap<String, AttributeValue> orderItem = new HashMap<>();
orderItem.put(ORDER_PARTITION_KEY, new AttributeValue(orderId));
orderItem.put(PRODUCT_PARTITION_KEY, new AttributeValue(productKey));
orderItem.put(CUSTOMER_PARTITION_KEY, new AttributeValue(customerId));
orderItem.put("OrderStatus", new AttributeValue("CONFIRMED"));
orderItem.put("OrderTotal", new AttributeValue("100"));

Put createOrder = new Put()
    .withTableName(ORDER_TABLE_NAME)
    .withItem(orderItem)
    .withReturnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
    .withConditionExpression("attribute_not_exists(" + ORDER_PARTITION_KEY + ")");
```

Executar a transação

O exemplo a seguir ilustra como executar as ações definidas anteriormente como uma única operação tudo ou nada.

```
Collection<TransactWriteItem> actions = Arrays.asList(
    new TransactWriteItem().withConditionCheck(checkCustomerValid),
    new TransactWriteItem().withUpdate(markItemSold),
    new TransactWriteItem().withPut(createOrder));

TransactWriteItemsRequest placeOrderTransaction = new TransactWriteItemsRequest()
    .withTransactItems(actions)
    .withReturnConsumedCapacity(ReturnConsumedCapacity.TOTAL);

// Run the transaction and process the result.
try {
    client.transactWriteItems(placeOrderTransaction);
    System.out.println("Transaction Successful");

} catch (ResourceNotFoundException rnf) {
    System.err.println("One of the table involved in the transaction is not found" + rnf.getMessage());
} catch (InternalServerException ise) {
    System.err.println("Internal Server Error" + ise.getMessage());
} catch (TransactionCanceledException tce) {
    System.out.println("Transaction Canceled " + tce.getMessage());
}
```

Lendo os Detalhes do Pedido

O exemplo da a seguir mostra como ler a ordem concluída transacionalmente no OrderseProductCatalogTabelas.

```
HashMap<String, AttributeValue> productItemKey = new HashMap<>();
productItemKey.put(PRODUCT_PARTITION_KEY, new AttributeValue(productKey));
```

```
HashMap<String, AttributeValue> orderKey = new HashMap<>();
orderKey.put(ORDER_PARTITION_KEY, new AttributeValue(orderId));

Get readProductSold = new Get()
    .withTableName(PRODUCT_TABLE_NAME)
    .withKey(productItemKey);
Get readCreatedOrder = new Get()
    .withTableName(ORDER_TABLE_NAME)
    .withKey(orderKey);

Collection<TransactGetItem> getActions = Arrays.asList(
    new TransactGetItem().withGet(readProductSold),
    new TransactGetItem().withGet(readCreatedOrder));

TransactGetItemsRequest readCompletedOrder = new TransactGetItemsRequest()
    .withTransactItems(getActions)
    .withReturnConsumedCapacity(ReturnConsumedCapacity.TOTAL);

// Run the transaction and process the result.
try {
    TransactGetItemsResult result = client.transactGetItems(readCompletedOrder);
    System.out.println(result.getResponses());
} catch (ResourceNotFoundException rnf) {
    System.err.println("One of the table involved in the transaction is not found" +
        rnf.getMessage());
} catch (InternalServerException ise) {
    System.err.println("Internal Server Error" + ise.getMessage());
} catch (TransactionCanceledException tce) {
    System.err.println("Transaction Canceled" + tce.getMessage());
}
```

Exemplos adicionais

- [Usando transações do DynamoDBMapper](#)

Backups do DynamoDB

Tópicos

- [Backup e restauração sob demanda para o DynamoDB \(p. 691\)](#)
- [Recuperação point-in-time do DynamoDB \(p. 706\)](#)

Backup e restauração sob demanda para o DynamoDB

Você pode criar backups sob demanda de suas tabelas do Amazon DynamoDB ou habilitar backups contínuos com recuperação point-in-time. Para obter mais informações sobre backups contínuos com a recuperação point-in-time, consulte [Recuperação point-in-time do DynamoDB \(p. 706\)](#).

Você pode usar o recurso de backup sob demanda do DynamoDB para criar backups completos de suas tabelas para retenção e arquivamento em longo prazo de modo a atender às necessidades de conformidade regulamentar. Você pode fazer backup dos dados de tabelas e restaurá-los a qualquer momento com um único clique no AWS Management Console ou com uma única chamada de API. As ações de backup e restauração são executadas sem causar impactos no desempenho nem na disponibilidade da tabela.

O processo de backup e restauração sob demanda é dimensionado sem degradar o desempenho nem a disponibilidade dos seus aplicativos. Ele usa uma nova tecnologia distribuída exclusiva que permite a realização de backups em questão de segundos, independentemente do tamanho da tabela. Você pode criar backups consistentes dentro de alguns segundos em milhares de partições sem se preocupar com horários ou processos de backup de longa duração. Todos os backups sob demanda são catalogados, detectados e retidos até que sejam excluídos.

Além disso, as operações de backup e restauração sob demanda não afetam o desempenho nem as latências da API. Os backups são preservados, independentemente da exclusão da tabela. Para mais informações, consulte [Fazer o backup e a restauração de tabelas do DynamoDB: Como ele funciona \(p. 692\)](#).

Você pode criar backups de tabela usando o console, a AWS Command Line Interface(AWS CLI) ou a API do DynamoDB. Para mais informações, consulte [Realizar o backup de uma tabela do DynamoDB \(p. 694\)](#).

Para obter mais informações sobre a restauração de tabelas a partir de backups, consulte [Restauração de uma tabela do DynamoDB por meio de backup \(p. 696\)](#).

Os backups sob demanda do DynamoDB estão disponíveis sem nenhum custo adicional além do preço normal associado ao tamanho de armazenamento do backup. Para obter mais informações sobre AWS Disponibilidade e preços da região, consulte [Definição de preços do Amazon DynamoDB](#).

Tópicos

- [Fazer o backup e a restauração de tabelas do DynamoDB: Como ele funciona \(p. 692\)](#)
- [Realizar o backup de uma tabela do DynamoDB \(p. 694\)](#)
- [Restauração de uma tabela do DynamoDB por meio de backup \(p. 696\)](#)
- [Excluir um backup da tabela do DynamoDB \(p. 701\)](#)
- [Uso do IAM com backup e restauração do DynamoDB \(p. 703\)](#)

Fazer o backup e a restauração de tabelas do DynamoDB: Como ele funciona

Você pode usar o recurso de backup sob demanda para criar backups completos das tabelas do Amazon DynamoDB. Esta seção oferece uma visão geral do que ocorre durante o processo de backup e restauração.

Backups

Ao criar um backup sob demanda, um marcador de tempo é catalogado para a solicitação. O backup é criado de forma assíncrona aplicando todas as alterações desde o momento da solicitação até o último snapshot da tabela completa. As solicitações de backup são processadas instantaneamente e a restauração é disponibilizada em minutos.

Note

Toda vez que você criar um backup sob demanda, será feito backup de todos os dados da tabela. Não há limite para o número de backups sob demanda que podem ser realizados.

Todos os backups no DynamoDB funcionam sem consumir throughput provisionado na tabela.

No DynamoDB, os backups não garantem consistência causal entre os itens; no entanto, a distorção entre as atualizações em um backup geralmente fica bem abaixo de um segundo.

Enquanto um backup estiver em andamento, você não poderá fazer o seguinte:

- Pausar ou cancelar a operação de backup.
- Excluir a tabela de origem do backup.
- Desativar backups em uma tabela se houver um backup em andamento para essa tabela.

Se não quiser criar scripts de agendamento e trabalhos de limpeza, use o AWS Backup para criar planos de backup com programações e políticas de retenção para suas tabelas do DynamoDB. AWS Backup executa os backups e os exclui quando expiram. Para obter mais informações, consulte o [Guia do desenvolvedor do AWS Backup](#).

Você pode programar backups periódicos ou futuros usando funções do AWS Lambda. Para obter mais informações, veja a postagem do blog [A serverless solution to schedule your Amazon DynamoDB On-Demand Backup](#).

Se estiver usando o console, todo backup criado usando o AWS Backup será listado na guia Backups com o Backup type (Tipo de backup) definido como AWS.

Note

Você não pode excluir backups marcados com um Tipo de backup de AWS Usando o console do DynamoDB. Para gerenciar esses backups, use o console do AWS Backup.

Para saber como realizar um backup, consulte [Realizar o backup de uma tabela do DynamoDB \(p. 694\)](#).

Restores

Você restaura uma tabela sem consumir a taxa de transferência provisionada na tabela. Você pode fazer uma restauração completa da tabela usando o backup do DynamoDB ou pode definir as configurações da tabela de destino. Ao fazer uma restauração, você pode alterar as seguintes configurações de tabela:

- Índices secundários globais (GSIs)
- Índices secundários locais (LSIs)
- Modo de faturamento
- Capacidade de leitura e gravação provisionada
- Configurações de criptografia

Important

Quando você faz uma restauração completa da tabela, a tabela de destino é definida com as mesmas unidades de capacidade de leitura e unidades de capacidade de gravação provisionadas da tabela de origem, conforme gravado no momento em que o backup foi solicitado. O processo de restauração também restaura os índices secundários locais e os índices secundários globais.

Você também pode restaurar os dados da tabela do DynamoDB em AWS Regiões de modo que a tabela restaurada seja criada em uma região diferente daquela na qual o backup reside. Você pode fazer restaurações entre regiões entre AWS Regiões comerciais, AWS Regiões da China, e AWS Regiões do GovCloud (US). Você paga somente pelos dados transferidos para fora da região de origem e pela restauração para uma nova tabela na região de destino.

As restaurações poderão ser mais rápidas e econômicas se você optar por excluir a criação de alguns ou todos os índices secundários na nova tabela restaurada.

Você deve configurar manualmente os itens a seguir na tabela restaurada:

- Políticas de Auto Scaling

- Políticas do AWS Identity and Access Management (IAM)
- Métricas e alarmes do Amazon CloudWatch
- Tags
- Configurações de stream
- Time to Live (TL - Tempo de vida)

Só é possível restaurar os dados completos da tabela para uma nova tabela por meio de backup. Você pode gravar na tabela restaurada somente depois que ela fica ativa.

Note

Você não pode substituir uma tabela existente durante uma operação de restauração.

As métricas de serviço mostram que 95% das restaurações da tabela dos clientes são concluídas em menos de uma hora. No entanto, os tempos de restauração estão diretamente relacionados à configuração das tabelas (como o tamanho das tabelas e o número de partições subjacentes) e outras variáveis relacionadas. Uma prática recomendada ao planejar a recuperação de desastres é documentar regularmente os tempos médios de conclusão da restauração e estabelecer como esses horários afetam seu objetivo geral de tempo de recuperação.

Para saber como realizar uma restauração, consulte [Restauração de uma tabela do DynamoDB por meio de backup \(p. 696\)](#).

Você pode usar políticas do IAM para controle de acesso. Para mais informações, consulte [Uso do IAM com backup e restauração do DynamoDB \(p. 703\)](#).

Todas as ações de backup e restauração de console e API são capturadas e registradas no AWS CloudTrail para registro, monitoramento contínuo e auditoria.

Realizar o backup de uma tabela do DynamoDB

Esta seção descreve como usar o console do Amazon DynamoDB ou o AWS Command Line Interface para fazer backup de uma tabela.

Tópicos

- [Criar o backup de uma tabela \(console\) \(p. 694\)](#)
- [Criar o backup de uma tabela \(AWS CLI\) \(p. 695\)](#)

Criar o backup de uma tabela (console)

Siga estas etapas para criar um backup chamado **MusicBackup** de uma tabela **Music** existente usando o AWS Management Console.

Para criar um backup de tabela

1. Faça login no AWS Management Console e abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. Para criar um backup, realize um dos seguintes procedimentos:
 - Na guia Backups da tabela **Music**, selecione Create backup (Criar backup).
 - No painel de navegação, no lado esquerdo do console, selecione Backups. Em seguida, escolha Create backup.
3. Verifique se **Music** é o nome da tabela e insira **MusicBackup** como o nome do backup. Em seguida, escolha Create para criar o backup.

Create table backup

Table name: Music

Backup name: MusicBackup

[Cancel](#) [Create](#)

Note

Se você criar backups usando a seção Backups no painel de navegação, a tabela não ficará pré-selecionada para você. Você precisará escolher manualmente o nome da tabela de origem para o backup.

Enquanto o backup estiver sendo criado, o status do backup será Creating. Depois que o backup for finalizado, o status dele mudará para Available (Disponível).

The screenshot shows the AWS DynamoDB Backup page. At the top, there are three buttons: 'Create backup' (highlighted in blue), 'Restore backup', and 'Delete backup'. Below these are filters for 'Filter by backup name' (with a search icon and clear button) and 'Time Range' (set to 'Last 30 Days'). A dropdown for 'Backup type' is set to 'All backups'. The main area displays a table with one backup entry:

	Backup name	Status	Creation time
<input checked="" type="radio"/>	MusicBackup	Available	February 12, 2020 at 3:04:59 PM UTC-8

Criar o backup de uma tabela (AWS CLI)

Siga estas etapas para criar um backup de uma tabela `Music` existente usando a AWS CLI.

Para criar um backup de tabela

- Crie um backup com o nome `MusicBackup` para a tabela `Music`.

```
aws dynamodb create-backup --table-name Music \
--backup-name MusicBackup
```

Durante a criação do backup, o status dele será CREATING.

```
{
    "BackupDetails": {
        "BackupName": "MusicBackup",
        "BackupArn": "arn:aws:dynamodb:us-east-1:123456789012:table/Music/
backup/01489602797149-73d8d5bc",
        "BackupStatus": "CREATING",
        "BackupCreationDateTime": 1489602797.149
    }
}
```

}

Quando o processo de backup for concluído, o `BackupStatus` deverá mudar para `AVAILABLE`. Para confirmar, use o comando `describe-backup`. Você pode obter o valor de entrada de `backup-arn` com a saída da etapa anterior ou usando o comando `list-backups`.

```
aws dynamodb describe-backup \
--backup-arn arn:aws:dynamodb:us-east-1:123456789012:table/Music/backup/01489173575360-
b308cd7d
```

Para acompanhar os backups, você pode usar o comando `list-backups`. Ele lista todos os backups em status `CREATING` ou `AVAILABLE`.

```
aws dynamodb list-backups
```

Os comandos `list-backups` e `describe-backup` são úteis para verificar informações sobre a tabela de origem do backup.

Restauração de uma tabela do DynamoDB por meio de backup

Esta seção descreve como restaurar uma tabela por meio de backup usando o console do Amazon DynamoDB ou a AWS Command Line Interface(AWS CLI).

Note

Se quiser usar a AWS CLI, você precisa configurá-la primeiro. Para mais informações, consulte [Acessar o DynamoDB \(p. 60\)](#).

Tópicos

- [Como restaurar uma tabela por meio de backup \(console\) \(p. 696\)](#)
- [Como restaurar uma tabela por meio de backup \(AWS CLI\) \(p. 699\)](#)

Como restaurar uma tabela por meio de backup (console)

O procedimento a seguir mostra como restaurar a tabela `Music` usando o arquivo `MusicBackup` que é criado no tutorial[Realizar o backup de uma tabela do DynamoDB \(p. 694\)](#).

Note

Este procedimento supõe que a tabela `Music` não existe mais antes de restaurá-la usando o arquivo `MusicBackup`.

Para restaurar uma tabela de um backup

1. Faça login no AWS Management Console e abra o console do DynamoDB em<https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Backups.
3. Na lista de backups, escolha `MusicBackup`.

The screenshot shows the 'Restore backup' tab selected. A search bar at the top has 'Filter by backup name' and a clear button. To its right are 'Time Range' set to 'Last 30 Days' and a refresh icon. Below this is a dropdown for 'Backup type' set to 'All backups'. The main area displays a table with columns: 'Backup name', 'Status', and 'Creation time'. One row is shown, labeled 'MusicBackup' with status 'Available' and creation time 'February 12, 2020 at 3:04:59 PM UTC-8'.

4. Escolha Restore backup (Restaurar backup).

This screenshot is identical to the one above, but the 'Restore backup' button in the top navigation bar is highlighted with a red oval.

5. Insira **Music** como o novo nome da tabela. Confirme o nome do backup e outros detalhes. Em seguida, escolha Restore table (Restaurar tabela) para iniciar o processo de restauração.

Note

Você pode restaurar a tabela para o mesmo AWS Região ou para uma região diferente daquela na qual o backup reside. Você também pode excluir a criação de índices secundários na nova tabela restaurada. Além disso, você pode especificar um modo de criptografia diferente.

Restore table from backup



New table name*

- Restore entire table data
 - Restored table will include all local secondary indexes and global secondary indexes.
- Restore without secondary indexes
 - Restored table will exclude the local secondary indexes and global secondary indexes. Note: Restores can be faster and more cost efficient if you choose to exclude secondary indexes from being created.

Cross region restore

- Same region
 - Restore the table to the same AWS Region.
- Cross region
 - Restore the table to a different AWS Region.

Select the encryption type

DEFAULT

The key is owned by Amazon DynamoDB. You are not charged any fee for using these CMKs.

KMS - Customer managed CMK

The key is stored in your account that you create, own, and manage. AWS Key Management Service (KMS) charges apply. [Learn more](#)

KMS - AWS managed CMK

The key is stored in your account and is managed by AWS Key Management Service (KMS). AWS KMS charges apply.

Backup table details

Original table name	Music
Backup name	MusicBackup
Backup ARN	arn:aws:dynamodb:eu-north-1:063317358631:table/Music/backup/01581548699157-81ef0887
Primary partition key	Artist
Sort key	SongTitle
Read/write capacity mode	Provisioned
Provisioned read capacity units	5
Provisioned write capacity units	5
Encryption Type	DEFAULT
Encryption Status	-
KMS Master Key ARN	Not Applicable
Auto Scaling	DISABLED
Stream enabled	No

Indexes

There are no global secondary indexes or local secondary indexes.

Restores can take several hours to complete.

* Required

Cancel

Restore table

A tabela que está sendo restaurada é mostrada com o status Creating (Em criação). Quando o processo de restauração for concluído, o status da tabela Music mudará para Active (Ativo).

Name	Status
Music	Active

Como restaurar uma tabela por meio de backup (AWS CLI)

Siga estas etapas para usar a AWS CLI para restaurar a tabela Music usando o MusicBackup que é criado no tutorial [Realizar o backup de uma tabela do DynamoDB \(p. 694\)](#).

Para restaurar uma tabela de um backup

1. Confirme o backup que você deseja restaurar usando o comando `list-backups`. Este exemplo usa `MusicBackup`.

```
aws dynamodb list-backups
```

Para obter outros detalhes sobre o backup, use o comando `describe-backup`. É possível obter a entrada `backup-arn` da etapa anterior:

```
aws dynamodb describe-backup \
--backup-arn arn:aws:dynamodb:us-east-1:123456789012:table/Music/backup/01489173575360-
b308cd7d
```

2. Restaure a tabela por meio do backup. Nesse caso, o `MusicBackup` restaura o `music` para a mesma tabela AWS Região :

```
aws dynamodb restore-table-from-backup \
--target-table-name Music \
--backup-arn arn:aws:dynamodb:us-east-1:123456789012:table/Music/backup/01489173575360-
b308cd7d
```

3. Restaure a tabela do backup com configurações de tabela personalizadas. Nesse caso, o `MusicBackup` restaura a tabela `Music` e especifica um modo de criptografia para a tabela restaurada.

Note

O parâmetro `sse-specification-override` assume os mesmos valores do parâmetro `sse-specification-override` usado no comando `CreateTable`. Para saber mais, consulte [Gerenciamento das tabelas criptografadas no DynamoDB \(p. 878\)](#).

```
aws dynamodb restore-table-from-backup \
--target-table-name Music \
```

```
--backup-arn arn:aws:dynamodb:us-east-1:123456789012:table/Music/backup/01581080476474-e177ebe2 \
--sse-specification-override Enabled=true,SSEType=KMS,KMSMasterKeyId=abcd1234-abcd-1234-a123-ab1234a1b234
```

Você pode restaurar a tabela para outro AWS Região de onde o backup reside.

Note

- O parâmetro `sse-specification-override` é obrigatório para restaurações entre regiões, mas opcional para restaurações na mesma região da tabela de origem.
- Ao realizar uma restauração entre regiões na linha de comando, você deve definir o padrão do AWS Região para a Região de destino desejada. Para saber mais, consulte [Opções de linha de comando no AWS Command Line Interface Guia do usuário](#).

```
aws dynamodb restore-table-from-backup \
--target-table-name Music \
--backup-arn arn:aws:dynamodb:us-east-1:123456789012:table/Music/backup/01581080476474-e177ebe2 \
--sse-specification-override Enabled=true,SSEType=KMS
```

Você pode substituir o modo de faturamento e a taxa de transferência provisionada para a tabela restaurada.

```
aws dynamodb restore-table-from-backup \
--target-table-name Music \
--backup-arn arn:aws:dynamodb:us-east-1:123456789012:table/Music/backup/01489173575360-b308cd7d \
--billing-mode-override PAY_PER_REQUEST
```

Você pode excluir a criação de alguns ou todos os índices secundários na tabela restaurada.

Note

As restaurações poderão ser mais rápidas e econômicas se você excluir a criação de alguns ou todos os índices secundários na tabela restaurada.

```
aws dynamodb restore-table-from-backup \
--target-table-name Music \
--backup-arn arn:aws:dynamodb:us-east-1:123456789012:table/Music/backup/01581081403719-db9c1f91 \
--global-secondary-index-override '[]' \
--sse-specification-override Enabled=true,SSEType=KMS
```

Note

Os índices secundários fornecidos devem corresponder a índices existentes correspondentes. Você não pode criar novos índices no momento da restauração.

Você pode usar uma combinação de diferentes substituições. Por exemplo, você pode usar um único índice secundário global e alterar a taxa de transferência provisionada ao mesmo tempo, da seguinte forma.

```
aws dynamodb restore-table-from-backup \
--target-table-name Music \
--backup-arn arn:aws:dynamodb:eu-west-1:123456789012:table/Music/
backup/01581082594992-303b6239 \
--billing-mode-override PROVISIONED_VERSÃO DA API 2012-08-10 \
--provisioned-throughput-override ReadCapacityUnits=100,WriteCapacityUnits=100 \
```

```
--global-secondary-index-override IndexName=singers-
index,KeySchema=[ "{AttributeName=SingerName,KeyType=HASH}" ],Projection="{ProjectionType=KEYS_ONLY}",IndexName=singers-
\
--sse-specification-override Enabled=true,SSEType=KMS
```

Para verificar a restauração, use o comando `describe-table` para descrever a tabela `Music`.

```
aws dynamodb describe-table --table-name Music
```

A tabela que está sendo restaurada do backup é mostrada com o status `Creating` (Em criação). Quando o processo de restauração for concluído, o status da tabela `Music` mudará para `Active` (Ativo).

Important

Enquanto houver uma restauração em andamento, não modifique nem exclua política de função do IAM; do contrário, poderá haver um comportamento inesperado. Por exemplo, suponha que você tenha removido as permissões de gravação para uma tabela enquanto essa tabela estava sendo restaurada. Nesse caso, a operação `RestoreTableFromBackup` subjacente não conseguiria gravar na tabela nenhum dos dados restaurados.

Assim que a operação de restauração estiver concluída, você poderá modificar ou excluir a política de função do IAM.

Políticas do IAM envolvendo restrições do IP de origem para acessar a tabela de restauração de destino deve ter a propriedade `aws:ViaAWSService` chave definida como `false` para garantir que as restrições se aplicam apenas a pedidos feitos diretamente por um responsável principal. Caso contrário, a restauração será cancelada.

Se o backup estiver criptografado com um AWS CMK gerenciado pelo ou um CMK gerenciado pelo cliente, não desabilite nem exclua a chave enquanto a restauração estiver em andamento ou ela apresentará falha.

Depois que a operação de restauração for concluída, você poderá alterar a chave de criptografia da tabela restaurada e desabilitar ou excluir a chave antiga.

Excluir um backup da tabela do DynamoDB

Esta seção descreve como usar o AWS Management Console ou o AWS Command Line Interface (AWS CLI) para excluir um backup de tabela do Amazon DynamoDB.

Note

Se desejar usar a AWS CLI, você precisará configurá-la primeiro. Para mais informações, consulte [Como usar AWS CLI \(p. 62\)](#).

Tópicos

- [Excluir o backup de uma tabela \(console\) \(p. 701\)](#)
- [Excluir um backup da tabela \(AWS CLI\) \(p. 702\)](#)

Excluir o backup de uma tabela (console)

O procedimento a seguir mostra como usar o console para excluir o `MusicBackup` que é criado no tutorial [Realizar o backup de uma tabela do DynamoDB \(p. 694\)](#).

Para excluir um backup

1. Faça login no AWS Management Console e abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.

2. No painel de navegação, no lado esquerdo do console, selecione Backups.
3. Na lista de backups, escolha MusicBackup.

The screenshot shows the 'Backups' section of the Amazon DynamoDB console. At the top, there are three buttons: 'Create backup' (blue), 'Restore backup', and 'Delete backup'. Below the buttons is a search bar labeled 'Filter by backup name' and a dropdown for 'Time Range' set to 'Last 30 Days'. A dropdown for 'Backup type' is set to 'All backups'. The main area displays a table with one row. The columns are 'Backup name', 'Status', and 'Creation time'. The row shows 'MusicBackup' as 'Available' and created on 'February 12, 2020 at 3:04:59 PM UTC-8'.

4. Escolha Delete backup (Excluir backup).

This screenshot is identical to the previous one, showing the list of backups. However, the 'Delete backup' button in the top navigation bar is circled in red to emphasize the step.

Escolha Delete (Excluir) para excluir o backup.



Are you sure you want to delete the backup "MusicBackup" of source table "Music"?

[Cancel](#) [Delete](#)

Excluir um backup da tabela (AWS CLI)

O exemplo a seguir exclui o backup de uma tabela existente `Music` usando a AWS CLI.

```
aws dynamodb delete-backup \
--backup-arn arn:aws:dynamodb:us-east-1:123456789012:table/Music/
backup/01489602797149-73d8d5bc
```

Uso do IAM com backup e restauração do DynamoDB

Você pode usar o AWS Identity and Access Management(IAM) para restringir as ações de backup e restauração do Amazon DynamoDB em alguns recursos. As APIs `CreateBackup` e `RestoreTableFromBackup` operam por tabela.

Para obter mais informações sobre como usar as políticas do IAM no DynamoDB, consulte [Uso de políticas baseadas em identidade \(políticas do IAM\) com o Amazon DynamoDB \(p. 888\)](#).

Veja a seguir exemplos de políticas do IAM que você pode usar para configurar funcionalidades específicas de backup e restauração no DynamoDB.

Exemplo 1: Permitir ações `CreateBackup` e `RestoreTableFromBackup`

A política do IAM a seguir concede permissões para o `CreateBackup` e `RestoreTableFromBackup` Ações do DynamoDB em todas as tabelas:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:CreateBackup",  
                "dynamodb:RestoreTableFromBackup",  
                "dynamodb:PutItem",  
                "dynamodb:UpdateItem",  
                "dynamodb:DeleteItem",  
                "dynamodb:GetItem",  
                "dynamodb:Query",  
                "dynamodb:Scan",  
                "dynamodb:BatchWriteItem"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

Important

As permissões de gravação do DynamoDB são necessárias para a funcionalidade de restauração.

Exemplo 2: Permitir `CreateBackup` e negar `RestoreTableFromBackup`

Esta política do IAM concede permissões para a ação `CreateBackup` e recusa a ação `RestoreTableFromBackup`:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["dynamodb:CreateBackup"],  
            "Resource": "*"  
        },  
        {  
            "Effect": "Deny",  
            "Action": ["dynamodb:RestoreTableFromBackup"],  
            "Resource": "*"  
        }  
    ]  
}
```

```
}
```

Exemplo 3: Permitir ListBackups e recusar CreateBackup e RestoreTableFromBackup

Esta política do IAM concede permissões para a ação ListBackups e recusa as ações CreateBackup e RestoreTableFromBackup:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["dynamodb>ListBackups"],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "dynamodb>CreateBackup",
                "dynamodb:RestoreTableFromBackup"
            ],
            "Resource": "*"
        }
    ]
}
```

Exemplo 4: Permitir ListBackups e negar DeleteBackup

Esta política do IAM concede permissões para a ação ListBackups e recusa a ação DeleteBackup:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["dynamodb>ListBackups"],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": ["dynamodb>DeleteBackup"],
            "Resource": "*"
        }
    ]
}
```

Exemplo 5: Permitir RestoreTableFromBackup e DescribeBackup para todos os recursos e negar DeleteBackup para um backup específico

Esta política do IAM concede permissões para a RestoreTableFromBackup e DescribeBackup ações e nega o DeleteBackup ação para um recurso de backup específico:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:RestoreTableFromBackup",
                "dynamodb:DescribeBackup"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": ["dynamodb>DeleteBackup"],
            "Resource": "arn:aws:dynamodb:region:account-id:backu
```

```
"Action": [
    "dynamodb:DescribeBackup",
    "dynamodb:RestoreTableFromBackup",
    "dynamodb:PutItem",
    "dynamodb:UpdateItem",
    "dynamodb:DeleteItem",
    "dynamodb:GetItem",
    "dynamodb:Query",
    "dynamodb:Scan",
    "dynamodb:BatchWriteItem"
],
"Resource": "*"
},
{
"Effect": "Deny",
"Action": [
    "dynamodb>DeleteBackup"
],
"Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/Music/
backup/01489173575360-b308cd7d"
}
]
```

Important

As permissões de gravação do DynamoDB são necessárias para a funcionalidade de restauração.

Exemplo 6: Permitir CreateBackup para uma tabela específica

Esta política do IAM concede permissões para a `CreateBackup` Ação do `noMovies` Apenas tabela:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["dynamodb>CreateBackup"],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:123456789012:table/Movies"
            ]
        }
    ]
}
```

Exemplo 7: Permitir listBackups

Esta política do IAM concede permissões para a `ListBackups` Ação:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["dynamodb>ListBackups"],
            "Resource": "*"
        }
    ]
}
```

Important

Não é possível conceder permissões para a ação `ListBackups` em uma tabela específica.

Recuperação point-in-time do DynamoDB

Você pode criar backups sob demanda de suas tabelas do Amazon DynamoDB ou habilitar backups contínuos usando a recuperação point-in-time. Para obter mais informações sobre backups sob demanda, consulte [Backup e restauração sob demanda para o DynamoDB \(p. 691\)](#).

A recuperação point-in-time ajuda a proteger as tabelas do DynamoDB de operações acidentais de gravação ou exclusão. Com a recuperação point-in-time, você não precisa se preocupar com a criação, a manutenção ou a programação de backups sob demanda. Por exemplo, suponhamos que um script de teste seja gravado acidentalmente em uma tabela do DynamoDB de produção. Com a recuperação point-in-time, você pode recuperar a tabela para qualquer ponto durante os últimos 35 dias. O DynamoDB mantém backups incrementais da tabela.

Além disso, as operações point-in-time não afetam o desempenho ou as latências de API. Para mais informações, consulte [Recuperação point-in-time: Como ele funciona \(p. 706\)](#).

Você pode restaurar uma tabela do DynamoDB para um ponto no tempo usando o AWS Management Console, o AWS Command Line Interface(AWS CLI) ou a API do DynamoDB. O processo de recuperação point-in-time restaura uma nova tabela. Para mais informações, consulte [Restaurar uma tabela do DynamoDB para um point-in-time \(p. 708\)](#).

Para obter mais informações sobre AWS Disponibilidade e preços da região, consulte [Definição de preços do Amazon DynamoDB](#).

Tópicos

- [Recuperação point-in-time: Como ele funciona \(p. 706\)](#)
- [Antes de você começar a usar a recuperação point-in-time \(p. 707\)](#)
- [Restaurar uma tabela do DynamoDB para um point-in-time \(p. 708\)](#)

Recuperação point-in-time: Como ele funciona

A recuperação point-in-time (PITR) do Amazon DynamoDB fornece backups automáticos dos dados de tabela do DynamoDB. Esta seção apresenta uma visão geral de como o processo funciona no DynamoDB.

Você pode habilitar a recuperação point-in-time usando a AWS Management Console, AWS Command Line Interface(AWS CLI) ou a API do DynamoDB. Quando habilitada, a recuperação point-in-time oferece backups contínuos até que você a desative explicitamente. Para mais informações, consulte [Restaurar uma tabela do DynamoDB para um point-in-time \(p. 708\)](#).

Depois de habilitar a recuperação point-in-time, você pode restaurar para qualquer ponto no tempo entre `EarliestRestorableDateTime` e `LatestRestorableDateTime`. `LatestRestorableDateTime` normalmente é 5 minutos antes da hora atual.

Note

O processo de recuperação point-in-time sempre restaura uma nova tabela.

Para `EarliestRestorableDateTime`, você pode restaurar a tabela para qualquer ponto durante os últimos 35 dias. O período de retenção é de 35 dias fixos (cinco semanas no calendário civil) e não pode ser modificado. Qualquer número de usuários pode executar até quatro restaurações simultâneas (qualquer tipo de restauração) em uma determinada conta.

Important

Se desabilitar a recuperação point-in-time depois reabilitá-la em uma tabela, você redefinirá a hora de início para a qual pode recuperar essa tabela. Dessa forma, você só pode restaurar imediatamente essa tabela usando a tabela `LatestRestorableDateTime`.

Quando você restaura usando a recuperação point-in-time, o DynamoDB restaura os dados da tabela para o estado com base na data e hora selecionadas (`day:hour:minute:second`) para uma nova tabela.

Você restaura uma tabela sem consumir a taxa de transferência provisionada na tabela. Você pode fazer uma restauração completa da tabela usando a recuperação point-in-time ou pode definir as configurações da tabela de destino. Você pode alterar as seguintes configurações de tabela na tabela restaurada:

- Índices secundários globais (GSIs)
- Índices secundários locais (LSIs)
- Modo de faturamento
- Capacidade de leitura e gravação provisionada
- Configurações de criptografia

Important

Quando você faz uma restauração completa da tabela, todas as configurações da tabela restaurada são originadas das configurações atuais da tabela de origem no momento da restauração. Por exemplo, suponha que a taxa de transferência provisionada de uma tabela tenha sido reduzida recentemente para 50 unidades de capacidade de leitura e 50 unidades de capacidade de gravação. Você, então, restaura o estado da tabela para três semanas atrás, quando a taxa de transferência provisionada estava definida como 100 unidades de capacidade de leitura e 100 unidades de capacidade de gravação. Nesse caso, o DynamoDB restaura os dados da tabela para esse ponto no tempo, mas usa a taxa de transferência provisionada atual (50 unidades de capacidade de leitura e 50 unidades de capacidade de gravação).

Você também pode restaurar os dados da tabela do DynamoDB em AWS Regiões de modo que a tabela restaurada seja criada em uma região diferente daquela na qual a tabela de origem reside. Você pode fazer restaurações entre regiões entre AWS Regiões comerciais, AWS Regiões da China, e AWS Regiões do GovCloud (US). Você paga somente pelos dados transferidos para fora da região de origem e pela restauração para uma nova tabela na região de destino.

Note

A restauração entre regiões não funcionará se a região de origem ou destino for a Ásia-Pacífico (Hong Kong) ou o Oriente Médio (Bahrein).

As restaurações poderão ser mais rápidas e econômicas se você excluir a criação de alguns ou todos os índices na tabela restaurada.

Você deve configurar manualmente os itens a seguir na tabela restaurada:

- Políticas de Auto Scaling
- Políticas do AWS Identity and Access Management (IAM)
- Métricas e alarmes do Amazon CloudWatch
- Tags
- Configurações de stream
- Time to Live (TL - Tempo de vida)
- Configurações de recuperação point-in-time

O tempo necessário para restaurar uma tabela varia com base em vários fatores. Os tempos da restauração point-in-time nem sempre estão correlacionados diretamente com o tamanho da tabela. Para mais informações, consulte [Restores \(p. 693\)](#).

Antes de você começar a usar a recuperação point-in-time

Antes de habilitar a PITR em uma tabela do Amazon DynamoDB, considere o seguinte:

- Se desabilitar a recuperação point-in-time depois reabilitá-la em uma tabela, você redefinirá a hora de início para a qual pode recuperar essa tabela. Dessa forma, você só pode restaurar imediatamente essa tabela usando a tabela `LatestRestorableDateTime`.
- Se você excluir uma tabela com a recuperação point-in-time habilitada, um backup do sistema será criado automaticamente e será mantido por 35 dias (sem custo adicional). Os backups do sistema permitem restaurar a tabela excluída no estado em que estava imediatamente antes do ponto de exclusão. Todos os backups do sistema seguem uma convenção de nomenclatura padrão: `table-name$DeletedTableBackup`.
- Você pode habilitar a recuperação point-in-time em cada réplica local de uma tabela global. Quando você restaure a tabela, o backup restaura uma tabela independente que não faz parte da tabela global. Se estiver usando [Versão 2019.11.21 \(atual\) \(p. 367\)](#) das tabelas globais, você poderá criar uma tabela global usando a tabela restaurada. Para mais informações, consulte [Tabelas globais Como ele funciona \(p. 367\)](#).
- Você também pode restaurar os dados da tabela do DynamoDB em AWS Regiões de modo que a tabela restaurada seja criada em uma região diferente daquela na qual a tabela de origem reside. Você pode fazer restaurações entre regiões entre AWS Regiões comerciais, AWS Regiões da China, e AWS Regiões do GovCloud (US). Você paga somente pelos dados transferidos para fora da região de origem e pela restauração para uma nova tabela na região de destino.
- O AWS CloudTrail registra todas as ações de console e API da recuperação point-in-time para habilitar registro, monitoramento contínuo e auditoria. Para mais informações, consulte [Registro de operações do DynamoDB usando o AWS CloudTrail \(p. 954\)](#).

Restaurar uma tabela do DynamoDB para um point-in-time

A recuperação point-in-time (PITR) do Amazon DynamoDB fornece backups contínuos dos dados de tabela do DynamoDB. Você pode restaurar uma tabela para um ponto no tempo usando o console do DynamoDB ou o AWS Command Line Interface (AWS CLI). O processo de recuperação point-in-time restaura uma nova tabela.

Se quiser usar a AWS CLI, você precisa configurá-la primeiro. Para mais informações, consulte [Acessar o DynamoDB \(p. 60\)](#).

Tópicos

- [Restaurar uma tabela para um ponto \(console\) \(p. 708\)](#)
- [Restaurar uma tabela para um ponto \(AWS CLI\) \(p. 709\)](#)

Restaurar uma tabela para um ponto (console)

O exemplo a seguir demonstra como usar o console do DynamoDB para restaurar uma tabela existente chamada `Musica` um ponto no tempo.

Note

Esse procedimento supõe que você tenha habilitado a recuperação point-in-time. Para habilitá-la para a tabela `Musica`, na guia Overview (Visão geral), na seção Table details (Detalhes da tabela), selecione Enable (Habilitar) para Point-in-time recovery (Recuperação point-in-time).

Para restaurar uma tabela para um ponto no tempo

1. Faça login no AWS Management Console e abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Na lista de tabelas, escolha a tabela `Musica`.
4. Na guia Backups (Backups) da tabela `Musica`, na seção Point-in-time recovery (Recuperação point-in-time), selecione Restore to point-in-time (Restaurar para ponto no tempo).

5. Insira **MusicMinutesAgo** como o novo nome da tabela.

Note

Você pode restaurar a tabela para o mesmo AWS Região ou para uma região diferente daquela na qual a tabela de origem reside. Você também pode excluir a criação de índices secundários na tabela restaurada. Além disso, você pode especificar um modo de criptografia diferente.

6. Para confirmar a data restaurável, defina Restore date and time (Data e hora da restauração) como a Latest restore date (Data da última restauração). Em seguida, escolha Restore table (Restaurar tabela) para iniciar o processo de restauração.

Note

Você pode restaurar a qualquer momento no Primeira data de restauração e Data de restauração mais recente. O DynamoDB restaura os dados da tabela para o estado com base na data e hora selecionadas (day:hour:minute:second).

A tabela que está sendo restaurada é mostrada com o status Restoring (Em restauração). Quando o processo de restauração for concluído, o status da tabela **MusicMinutesAgo** mudará para Active (Ativo).

Restaurar uma tabela para um ponto (AWS CLI)

O seguinte procedimento mostra como usar a AWS CLI para restaurar uma tabela existente chamada **Music** para um ponto no tempo.

Note

Esse procedimento supõe que você tenha habilitado a recuperação point-in-time. Para ativar esse recurso para a tabela **Music**, execute o seguinte comando.

```
aws dynamodb update-continuous-backups \
    --table-name Music \
    --point-in-time-recovery-specification PointInTimeRecoveryEnabled=True
```

Para restaurar uma tabela para um ponto no tempo

1. Confirme se a recuperação point-in-time está habilitada para a tabela **Music** usando o comando `describe-continuous-backups`.

```
aws dynamodb describe-continuous-backups \
    --table-name Music
```

Os backups contínuos (habilitados automaticamente na criação da tabela) e a recuperação point-in-time estão habilitados.

```
{
    "ContinuousBackupsDescription": {
        "PointInTimeRecoveryDescription": {
            "PointInTimeRecoveryStatus": "ENABLED",
            "EarliestRestorableDateTime": 1519257118.0,
            "LatestRestorableDateTime": 1520018653.01
        },
        "ContinuousBackupsStatus": "ENABLED"
    }
}
```

}

2. Restaure a tabela para um ponto. Nesse caso, o Music é restaurada para a tabela LatestRestorableDateTime (~5 minutos atrás) para o mesmo AWS Região:

```
aws dynamodb restore-table-to-point-in-time \
--source-table-name Music \
--target-table-name MusicMinutesAgo \
--use-latest-restorable-time
```

Note

Você também pode restaurar para um ponto específico. Para isso, execute o comando usando o argumento `--restore-date-time` e especifique um timestamp. Você pode especificar qualquer ponto durante os últimos 35 dias. Por exemplo, o comando a seguir restaura a tabela para o EarliestRestorableDateTime.

```
aws dynamodb restore-table-to-point-in-time \
--source-table-name Music \
--target-table-name MusicEarliestRestorableDateTime \
--no-use-latest-restorable-time \
--restore-date-time 1519257118.0
```

Especificar o argumento `--no-use-latest-restorable-time` é opcional durante a restauração para um ponto específico.

3. Restaure a tabela para um momento determinado com configurações de tabela personalizadas. Nesse caso, a tabela Music é restaurada para o LatestRestorableDateTime (há aproximadamente cinco minutos).

Você pode especificar um modo de criptografia diferente para a tabela restaurada, conforme mostrado a seguir.

Note

O parâmetro `sse-specification-override` assume os mesmos valores do parâmetro `sse-specification-override` usado no comando `CreateTable`. Para saber mais, consulte [Gerenciamento das tabelas criptografadas no DynamoDB \(p. 878\)](#).

```
aws dynamodb restore-table-to-point-in-time \
--source-table-name Music \
--target-table-name MusicMinutesAgo \
--use-latest-restorable-time \
--sse-specification-override Enabled=true,SSEType=KMS,KMSMasterKeyId=abcd1234-
abcd-1234-a123-ab1234a1b234
```

Você pode restaurar a tabela para outro AWS Região da qual a tabela de origem reside.

Note

- O parâmetro `sse-specification-override` é obrigatório para restaurações entre regiões, mas opcional para restaurações para a mesma região da tabela de origem.
- O parâmetro `source-table-arn` deve ser fornecido para restaurações entre regiões.
- Ao realizar uma restauração entre regiões na linha de comando, você deve definir o padrão do AWS Região para a Região de destino desejada. Para saber mais, consulte [Opções de linha de comando no AWS Command Line Interface Guia do usuário](#).

```
aws dynamodb restore-table-to-point-in-time \
```

```
--source-table-arn arn:aws:dynamodb:us-east-1:123456789012:table/Music \
--target-table-name MusicMinutesAgo \
--use-latest-restorable-time \
--sse-specification-override Enabled=true,SSEType=KMS,KMSMasterKeyId=abcd1234-
abcd-1234-a123-ab1234a1b234
```

Você pode substituir o modo de faturamento e a taxa de transferência provisionada para a tabela restaurada.

```
aws dynamodb restore-table-to-point-in-time \
--source-table-name Music \
--target-table-name MusicMinutesAgo \
--use-latest-restorable-time \
--billing-mode-override PAY_PER_REQUEST
```

Você pode excluir a criação de alguns ou todos os índices secundários na tabela restaurada.

Note

As restaurações poderão ser mais rápidas e econômicas se você excluir a criação de alguns ou todos os índices secundários na nova tabela restaurada.

```
aws dynamodb restore-table-to-point-in-time \
--source-table-name Music \
--target-table-name MusicMinutesAgo \
--use-latest-restorable-time \
--global-secondary-index-override '[]'
```

Você pode usar uma combinação de diferentes substituições. Por exemplo, você pode usar um único índice secundário global e alterar a taxa de transferência provisionada ao mesmo tempo, da seguinte forma.

```
aws dynamodb restore-table-to-point-in-time \
--source-table-name Music \
--target-table-name MusicMinutesAgo \
--billing-mode-override PROVISIONED \
--provisioned-throughput-override ReadCapacityUnits=100,WriteCapacityUnits=100 \
--global-secondary-index-override IndexName=singers-
index,KeySchema=[{"AttributeName=SingerName,KeyType=HASH"}],Projection="{ProjectionType=KEYS_ONLY}" \
 \
--sse-specification-override Enabled=true,SSEType=KMS \
--use-latest-restorable-time
```

Para verificar a restauração, use o comando `describe-table` para descrever a tabela `MusicEarliestRestorableDateTime`.

```
aws dynamodb describe-table --table-name MusicEarliestRestorableDateTime
```

A tabela que está sendo restaurada é mostrada com o status `Creating` (Em criação) e restauração em andamento como `true` (verdadeiro). Quando o processo de restauração for concluído, o status da tabela `MusicEarliestRestorableDateTime` mudará para `Active` (Ativo).

Important

Enquanto a restauração estiver em andamento, não modifique nem exclua aAWS Identity and Access ManagementPolíticas do (IAM) que concedem à entidade do IAM (por exemplo, usuário, grupo ou função) permissão para realizar a restauração. Do contrário, pode haver um comportamento inesperado. Por exemplo, suponha que você tenha removido as permissões

de gravação de uma tabela enquanto essa tabela estava sendo restaurada. Neste caso, o `SubjacenteRestoreTableToPointInTime` operação não pode gravar nenhum dos dados restaurados para a tabela. As políticas do IAM envolvendo restrições de IP de origem para acessar a tabela de restauração de destino podem causar problemas de modo semelhante. Você pode modificar ou excluir permissões somente depois que a operação de restauração é concluída.

Aceleração na memória com o DynamoDB Accelerator (DAX)

O Amazon DynamoDB foi concebido para escala e desempenho. Na maioria dos casos, os tempos de resposta do DynamoDB podem ser medidos em milissegundos de um dígito. No entanto, existem certos casos de uso que exigem tempos de resposta em microssegundos. Para esses casos de uso, o DAX (DynamoDB Accelerator) oferece tempos de resposta rápidos para acessar dados eventualmente consistentes.

O DAX é um serviço de armazenamento em cache compatível com o DynamoDB no qual você pode se beneficiar com o rápido desempenho na memória para aplicativos exigentes. O DAX lida com três cenários principais:

1. Como um cache de memória, o DAX reduz os tempos de resposta de cargas de trabalho de leitura eventualmente consistentes por uma certa ordem de magnitude que varia de milissegundos de um único dígito até microssegundos.
2. O DAX reduz a complexidade operacional e do aplicativo fornecendo um serviço gerenciado que é compatível com a API do DynamoDB. Portanto, ele exige apenas alterações funcionais mínimas para uso com um aplicativo existente.
3. Para cargas de trabalho intermitentes ou pesadas para leitura, o DAX fornece maior throughput e mais potencial para economia de custos operacionais, reduzindo a necessidade de provisionar unidades de capacidade de leitura excessivamente. Isso é especialmente benéfico para aplicativos que exigem leituras repetidas para chaves individuais.

O DAX oferece suporte para criptografia do lado do servidor. Com a criptografia em repouso, os dados persistentes pelo DAX no disco serão criptografados. O DAX grava dados ao disco como parte das alterações de propagação do nó primário para as réplicas de leitura. Para obter mais informações, consulte [Criptografia DAX em repouso \(p. 815\)](#).

O DAX também oferece suporte à criptografia em trânsito, garantindo que todas as solicitações e respostas entre o aplicativo e o cluster sejam criptografadas por TLS (Transport Level Security, segurança de nível de transporte) e que as conexões com o cluster possam ser autenticadas pela verificação de um certificado x509 de cluster. Para obter mais informações, consulte [Criptografia do DAX em trânsito \(p. 817\)](#).

Tópicos

- [Casos de uso para o DAX \(p. 714\)](#)
- [Observações de uso do DAX \(p. 714\)](#)
- [DAX: Como ele funciona \(p. 715\)](#)
- [Componentes de cluster DAX \(p. 719\)](#)
- [Criar um cluster DAX \(p. 723\)](#)
- [Modelos de consistência DAX e DynamoDB \(p. 732\)](#)
- [Como desenvolver com o cliente do DynamoDB Accelerator \(DAX\) \(p. 738\)](#)
- [Gerenciando clusters DAX \(p. 782\)](#)
- [Monitoramento DAX \(p. 788\)](#)
- [Instâncias intermitentes DAX T3/T2 \(p. 803\)](#)
- [Controle de acesso DAX \(p. 805\)](#)

- Criptografia DAX em repouso (p. 815)
- Criptografia do DAX em trânsito (p. 817)
- Uso de funções do IAM vinculadas ao serviço para DAX (p. 817)
- Acessando DAX entre AWS Contas (p. 820)
- Guia de dimensionamento de cluster do DAX (p. 827)
- Referência de API do. (p. 829)

Casos de uso para o DAX

O DAX fornece acesso a dados eventualmente consistentes de tabelas do DynamoDB, com latência de microssegundos. Um cluster DAX multi-AZ pode servir milhões de solicitações por segundo.

O DAX é ideal para os seguintes tipos de aplicativo:

- Aplicativos que exigem o melhor tempo de resposta possível para leituras. Alguns exemplos incluem lances em tempo real, jogos sociais e aplicativos de negócios. O DAX oferece um desempenho de leitura rápido na memória para esses casos de uso.
- Aplicativos que fazem a leitura de um pequeno número de itens com mais frequência do que outros. Por exemplo, considere um sistema de comércio eletrônico que tem uma promoção de um produto popular válida por apenas um dia. Durante a promoção, a demanda por esse produto (e seus dados no DynamoDB) aumentaria drasticamente em comparação a todos os outros produtos. Para atenuar os impactos de uma chave de “aceleração” e uma distribuição de tráfego não uniforme, você pode descarregar as atividades de leitura para um cache do DAX até essa promoção acabar.
- Aplicativos que exigem leitura intensa, mas que também são sensíveis aos custos. Com o DynamoDB, você fornece o número de leituras por segundo que o seu aplicativo exige. Se as atividades de leitura aumentarem, você poderá aumentar o throughput de leitura provisionado das suas tabelas (a um custo adicional). Como alternativa, é possível descarregar as atividades do seu aplicativo em um cluster DAX e reduzir o número de unidades de capacidade de leitura que você precisa comprar.
- Aplicativos que exigem leituras repetidas em um grande conjunto de dados. Esses aplicativos poderiam desviar os recursos de banco de dados de outros aplicativos. Por exemplo, uma análise de longa execução de dados meteorológicos regionais pode consumir toda a capacidade de leitura em uma tabela do DynamoDB. Essa situação pode afetar negativamente outros aplicativos que precisam acessar os mesmos dados. Com o DAX, a análise meteorológica pode ser realizada em vez disso com base nos dados em cache.

O DAX é **ideal** para os seguintes tipos de aplicativo:

- Aplicativos que exigem leituras fortemente consistentes (ou que não toleram leituras eventualmente consistentes).
- Aplicativos que não precisam de tempos de resposta em microssegundos para leituras ou descarregar atividades de leitura repetidas de tabelas subjacentes.
- Aplicativos que exigem gravação intensa ou que não realizam muitas atividades de leitura.
- Aplicativos que já estão usando uma solução de armazenamento em cache diferente com o DynamoDB e estão usando sua própria lógica no lado do cliente para trabalhar com essa solução de armazenamento em cache.

Observações de uso do DAX

- Para uma lista de AWS Regiões onde o DAX está disponível, consulte [Definição de preços do Amazon DynamoDB](#).

- O DAX oferece suporte a aplicativos escritos em Go, Java, Node.js, Python e .NET, que usam oAWS-clientes fornecidos para essas linguagens de programação.
- O DAX só está disponível para a plataforma EC2-VPC. (Não há suporte para a plataforma EC2-Classic.)
- A política da função de serviço do cluster do DAX deve permitir `odynamodb:DescribeTable` a fim de manter os metadados sobre a tabela do DynamoDB.
- Os clusters do DAX mantêm metadados sobre os nomes de atributos de itens que armazenam. Esses metadados são mantidos indefinidamente (mesmo depois que o item expira ou é removido do cache). Aplicativos que usam um número não vinculado de nomes de atributos podem, com o tempo, provocar exaustão de memória no cluster do DAX. Essa limitação aplica-se somente aos nomes de atributo nível superior, não a nomes de atributo aninhados. Exemplos de nomes de atributo de nível superior problemáticos incluem time stamps, UUIDs e IDs de sessão.

Essa limitação se aplica a nomes de atributos, não a seus valores. Itens como esses não constituem um problema.

```
{  
    "Id": 123,  
    "Title": "Bicycle 123",  
    "CreationDate": "2017-10-24T01:02:03+00:00"  
}
```

Mas itens como esses serão um problema, se houver um número suficiente deles e cada um tiver um timestamp diferente.

```
{  
    "Id": 123,  
    "Title": "Bicycle 123",  
    "2017-10-24T01:02:03+00:00": "created"  
}
```

DAX: Como ele funciona

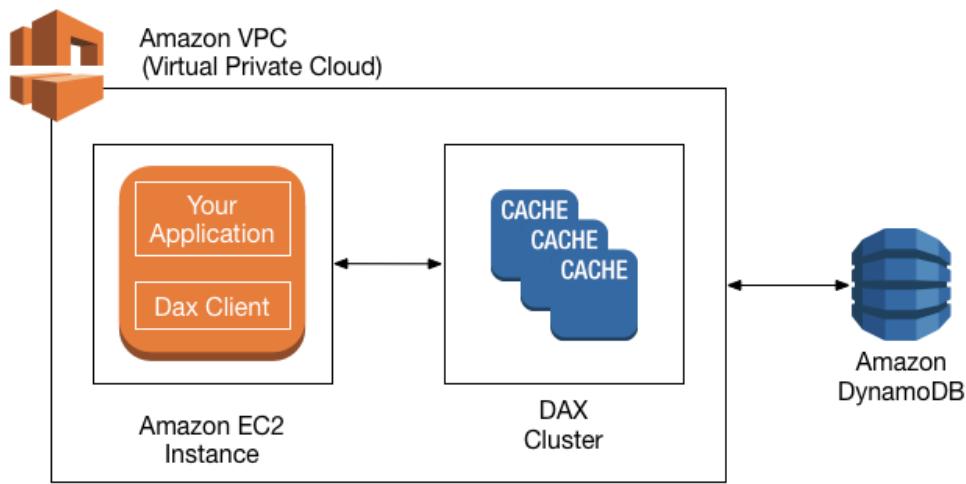
O Amazon DynamoDB Accelerator (DAX) foi projetado para ser executado em um ambiente Amazon Virtual Private Cloud (Amazon VPC). O serviço da Amazon VPC define uma rede virtual que lembra muito um datacenter tradicional. Com uma VPC, você tem controle sobre o intervalo de endereços IP, as sub-redes, as tabelas de roteamento, os gateways de rede e as configurações de segurança. Você pode iniciar um cluster DAX em sua rede virtual e controlar o acesso ao cluster, usando security groups da Amazon VPC.

Note

Se você criou seu AWSDepois de 4 de dezembro de 2013, você já tem uma VPC padrão em cada AWSRegião : A VPC está pronta para ser usada imediatamente, sem precisar executar nenhuma etapa de configuração adicional.

Para obter mais informações, consulte [Padrão VPC e sub-redes padrão](#) no Guia do usuário da Amazon VPC Manual.

O diagrama a seguir mostra uma visão geral de alto nível do DAX.



Para criar um cluster do DAX, você deve usar o método de AWS Management Console. A não ser que você especifique o contrário, o cluster do DAX será executado na sua VPC padrão. Para executar o aplicativo, você executa uma instância do Amazon EC2 na sua Amazon VPC. Em seguida, você implanta o aplicativo (com o cliente do DAX) na instância do EC2.

No tempo de execução, o cliente DAX direciona todas as solicitações de API do DynamoDB do seu aplicativo para o cluster DAX. Se o DAX puder, ele processará uma dessas solicitações de API diretamente. Caso contrário, ele passará a solicitação para o DynamoDB.

Por fim, o cluster do DAX retornará os resultados ao seu aplicativo.

Tópicos

- [Como o DAX processa solicitações \(p. 716\)](#)
- [Cache de itens \(p. 718\)](#)
- [Cache de consultas \(p. 718\)](#)

Como o DAX processa solicitações

Um cluster do DAX consiste em um ou mais nós. Cada nó executa sua própria instância do software de armazenamento em cache do DAX. Um dos nós serve como o nó primário do cluster. Os nós adicionais (se houver) servem como réplicas de leitura. Para mais informações, consulte [Nodes \(p. 719\)](#).

Seu aplicativo pode acessar o DAX, especificando o endpoint do cluster do DAX. O software cliente DAX do funciona com o endpoint do cluster para executar平衡amento de carga e roteamento inteligentes.

Operações de leitura

O DAX pode responder às chamadas de API a seguir:

- `GetItem`
- `BatchGetItem`
- `Query`

- Scan

Se a solicitação especificar leituras eventualmente consistentes (o comportamento padrão), ele tenta ler o item do DAX:

- Se o DAX tiver o item disponível (um acerto de cache), o DAX retornará o item ao aplicativo sem acessar o DynamoDB.
- Se o DAX não tiver o item disponível (um erro de cache), o DAX passará a solicitação para o DynamoDB. Quando recebe a resposta do DynamoDB, o DAX retornará os resultados ao aplicativo. Mas também grava os resultados no cache do nó primário.

Note

Se houver réplicas de leitura no cluster, o DAX automaticamente manterá as réplicas sincronizadas com o nó principal. Para mais informações, consulte [Clusters \(p. 719\)](#).

Se a solicitação especificar leituras fortemente consistentes, o DAX passará a solicitação para o DynamoDB. Os resultados do DynamoDB não são armazenados em cache no DAX. Em vez disso, eles são simplesmente retornados ao aplicativo.

Operações de gravação

As operações de API do DAX a seguir são consideradas “write-through”:

- `BatchWriteItem`
- `UpdateItem`
- `DeleteItem`
- `PutItem`

Com essas operações, os dados são primeiramente gravados na tabela do DynamoDB e, em seguida, no cluster do DAX. A operação é bem-sucedida somente se os dados são gravados com êxito no Ambosa tabela e para DAX.

Outras operações do

O DAX não reconhece as operações do DynamoDB para gerenciar tabelas (tais como `CreateTable`, `UpdateTable`, e assim por diante). Se o seu aplicativo precisar executar essas operações, ele precisará acessar o DynamoDB diretamente em vez de usar o DAX.

Para obter informações detalhadas sobre a consistência do DAX e do DynamoDB, consulte [Modelos de consistência DAX e DynamoDB \(p. 732\)](#).

Para obter informações sobre como as transações funcionam no DAX, consulte [Uso de APIs transacionais no DynamoDB Accelerator \(DAX\) \(p. 685\)](#).

Limitação de taxa de solicitações

Se o número de solicitações enviadas ao DAX exceder a capacidade de um nó, o DAX limitará a taxa na qual aceita solicitações adicionais retornando uma `ThrottlingException`. O DAX avalia continuamente a utilização da CPU para determinar o volume de solicitações que ela pode processar mantendo um estado de cluster íntegro.

Você pode monitorar a [Métrica do ThrottledRequestCount \(p. 791\)](#) que o DAX publica no Amazon CloudWatch. Se você vir essas exceções regularmente, considere [aumentar o cluster \(p. 784\)](#).

Cache de itens

O DAX mantém umaCache de itensPara armazenar os resultados doGetItemBatchGetItemoperações. Os itens no cache representam dados eventualmente consistentes do DynamoDB e são armazenados por seus valores de chave primária.

Quando um aplicativo envia umGetItemouBatchGetItem, o DAX tenta ler os itens diretamente do cache de itens usando os valores de chave especificados. Se os itens forem encontrados (acerto de cache), o DAX retorna-os para o aplicativo imediatamente. Se os itens não forem encontrados (erro de cache), o DAX enviará a solicitação para o DynamoDB. O DynamoDB processa as solicitações usando leituras eventualmente consistentes e retorna os itens ao DAX. O DAX armazena-os no cache de itens e, em seguida, retorna-os para o aplicativo.

Por padrão, o cache de itens tem uma configuração de tempo de vida (TTL) de 5 minutos. O DAX atribui um carimbo de data e hora para cada item que ele grava no cache de itens. Um item expira se ele permaneceu no cache por mais tempo que a configuração de TTL. Se você emitir umGetItemEm um item expirado, isso é considerado um erro de cache, e o DAX envia aGetItemSolicitação ao DynamoDB.

Note

Você pode especificar a configuração de TTL para o cache de itens ao criar um novo cluster do DAX. Para mais informações, consulte [Gerenciando clusters DAX \(p. 782\)](#).

O DAX também mantém uma lista de menos usados recentemente (LRU) para o cache de itens. A lista de LRU rastreia quando um item foi gravado pela primeira vez no cache, e quando o item foi lido pela última vez no cache. Se o cache de itens ficar cheio, o DAX excluirá os itens mais antigos (mesmo que eles ainda não tenham expirado) para dar espaço para novos itens. O algoritmo de LRU está sempre habilitado para o cache de itens, e não pode ser configurado pelo usuário.

Se você especificar zero comoCache de itensconfiguração TTL, os itens no cache do item só serão atualizados devido a uma execução LRU ou a um “[write-through](#)” operação.

Para obter informações detalhadas sobre a consistência do cache de itens no DAX, consulte [Comportamento do cache de itens do \(p. 732\)](#).

Cache de consultas

O DAX também mantém umaCache de consultasPara armazenar os resultados doqueryeScanoperações. Os itens nesse cache representam conjuntos de resultados de consultas e verificações nas tabelas do DynamoDB. Esses conjuntos de resultados são armazenados por seus valores de parâmetro.

Quando um aplicativo envia umqueryouScan, o DAX tenta ler um conjunto de resultados correspondente no cache de consultas usando os valores dos parâmetros especificados. Se o conjunto de resultados for encontrado (acerto de cache), o DAX o retornará para o aplicativo imediatamente. Se o conjunto de resultados não for encontrado (erro de cache), o DAX enviará a solicitação para o DynamoDB. O DynamoDB processa as solicitações usando leituras eventualmente consistentes e retorna o conjunto de resultados para o DAX. O DAX armazena-o no cache de consultas e, em seguida, retorna-o para o aplicativo.

Note

Você pode especificar a configuração de TTL para o cache de consultas ao criar um novo cluster do DAX. Para mais informações, consulte [Gerenciando clusters DAX \(p. 782\)](#).

O DAX também mantém uma lista de LRU para o cache de consultas. A lista rastreia quando um conjunto de resultados foi gravado pela primeira vez no cache, e quando o resultado foi lido pela última vez no cache. Se o cache de consultas ficar cheio, o DAX excluirá os conjuntos de resultados mais antigos

(mesmo que eles ainda não tenham expirado) para dar espaço para novos conjuntos de resultados. O algoritmo LRU está sempre ativado para o cache de consultas, e não pode ser configurado pelo usuário.

Se você especificar zero comoCache de consultas configuração TTL, a resposta da consulta não será armazenada em cache.

Para obter informações detalhadas sobre a consistência do cache de consultas no DAX, consulte., consulte [Comportamento do cache de consultas do \(p. 735\)](#).

Componentes de cluster DAX

Um cluster do Amazon DynamoDB Accelerator (DAX) consiste em AWS Componentes de infraestrutura. Esta seção descreve esses componentes e como eles funcionam em conjunto.

Tópicos

- [Nodes \(p. 719\)](#)
- [Clusters \(p. 719\)](#)
- [Regiões e zonas de disponibilidade \(p. 720\)](#)
- [Grupos de parâmetros \(p. 721\)](#)
- [Security Groups \(Grupos de segurança\) \(p. 721\)](#)
- [ARN do cluster \(p. 721\)](#)
- [Endpoint de cluster \(p. 721\)](#)
- [Endpoints de nó \(p. 722\)](#)
- [Grupos de sub-redes \(p. 722\)](#)
- [Events \(p. 722\)](#)
- [Janela de manutenção \(p. 722\)](#)

Nodes

Ano É o menor bloco de criação de um cluster DAX. Cada nó executa uma instância do software DAX e mantém uma única réplica dos dados em cache.

Você pode escalar seu cluster DAX de uma das seguintes formas:

- Adicionando mais nós ao cluster. Isso aumenta a taxa de transferência de leitura geral do cluster.
- Ao usar um tipo de nó maior. Tipos de nó maiores fornecem mais capacidade e podem aumentar o throughput. (Você deve criar um novo cluster com o novo tipo de nó.)

Cada nó em um cluster é do mesmo tipo de nó e executa o mesmo software de armazenamento em cache DAX. Para obter uma lista dos tipos de nó disponíveis, consulte [Definição de preço do Amazon DynamoDB](#).

Clusters

A cluster O é um agrupamento lógico de um ou mais nós que o DAX gerencia como uma unidade. Um dos nós do cluster é designado como o nó primário e os outros nós (se houver) são réplicas de leitura.

O nó primário é responsável pelo seguinte:

- Cumprir solicitações de aplicativo para dados em cache.

- Tratar de operações de gravação para DynamoDB.
- Remover dados do cache, de acordo com a política de remoção do cluster.

Quando são feitas alterações nos dados armazenados em cache no nó principal, o DAX propaga as alterações para todos os nós de réplica de leitura.

As réplicas de leitura são responsáveis pelo seguinte:

- Cumprir solicitações de aplicativo para dados em cache.
- Remover dados do cache, de acordo com a política de remoção do cluster.

No entanto, ao contrário do nó principal, as réplicas de leitura não gravam no DynamoDB.

As réplicas de leitura têm duas finalidades adicionais:

- Escalabilidade. Se você tiver um grande número de clientes de aplicativos que precisam acessar o DAX simultaneamente, será possível adicionar mais réplicas para escalabilidade de leitura. O DAX distribui a carga uniformemente entre todos os nós no cluster. (Outra forma de aumentar o throughput é usar maiores tipos de nó de cache.)
- Alta disponibilidade. Em caso de falha de um nó principal, o DAX recupera automaticamente uma réplica de leitura e a designa como o novo nó principal. Se um nó de réplica falhar, outros nós no cluster DAX ainda poderão atender às solicitações até que o nó com falha possa ser recuperado. Para obter a máxima tolerância a falhas, você deve implantar as réplicas de leitura em Zonas de disponibilidade separadas. Essa configuração garante que seu cluster DAX possa continuar a funcionar, mesmo se toda uma Zona de disponibilidade se tornar indisponível.

Um cluster DAX pode oferecer suporte para até 10 nós por cluster (o nó principal, mais um máximo de nove réplicas de leitura).

Important

Para uso em produção, é altamente recomendável usar o DAX com pelo menos três nós, e posicionar cada nó em diferentes zonas de disponibilidade. São necessários três nós para que um cluster DAX seja tolerante a falhas.

Um cluster DAX pode ser implantado com um ou dois nós para cargas de trabalho de desenvolvimento ou de teste. Os clusters de um e dois nós não são tolerantes a falhas, portanto, não convém usar menos de três nós na produção. Se o cluster de um ou dois nós encontrar erros de software ou de hardware, ele poderá se tornar indisponível ou perder os dados armazenados em cache.

Regiões e zonas de disponibilidade

Um cluster DAX em umAWSA região só pode interagir com tabelas do DynamoDB na mesma região. Por esse motivo, é necessário iniciar o cluster DAX na região correta. Se você tiver tabelas do DynamoDB em outras regiões, inicie os clusters DAX nessas regiões também.

Cada região é projetada para ser completamente isolada das outras regiões. Dentro de cada região há várias zonas de disponibilidade. Ao iniciar seus nós em diferentes zonas de disponibilidade, você é capaz de alcançar o máximo possível de tolerância a falhas.

Important

Não coloque todos os nós do cluster em uma única zona de disponibilidade. Nessa configuração, o cluster se tornará indisponível no caso de uma falha na zona de disponibilidade.

Para uso em produção, é altamente recomendável usar o DAX com pelo menos três nós, e posicionar cada nó em diferentes zonas de disponibilidade. São necessários três nós para que um cluster DAX seja tolerante a falhas.

Um cluster DAX pode ser implantado com um ou dois nós para cargas de trabalho de desenvolvimento ou de teste. Os clusters de um e dois nós não são tolerantes a falhas, portanto, não convém usar menos de três nós na produção. Se o cluster de um ou dois nós encontrar erros de software ou de hardware, ele poderá se tornar indisponível ou perder os dados armazenados em cache.

Grupos de parâmetros

Grupos de parâmetrosSão usados para gerenciar as configurações de tempo de execução para clusters DAX. O DAX tem vários parâmetros que você pode usar para otimizar o desempenho (como a definição de uma política de TL para dados armazenados em cache). Um grupo de parâmetros é um conjunto de parâmetros que você pode aplicar a um cluster. Dessa maneira, você garante que todos os nós desse cluster sejam configurados exatamente da mesma forma.

Security Groups (Grupos de segurança)

Um cluster DAX é executado em um ambiente Amazon Virtual Private Cloud (Amazon VPC). Esse ambiente é uma rede virtual dedicada ao seu AWSe é isolado de outras VPCs. Um grupo de segurança atua como um firewall virtual para a VPC, permitindo que você controle o tráfego de entrada e saída de rede.

Ao iniciar um cluster na VPC, você adiciona uma regra de entrada ao grupo de segurança para permitir tráfego de rede de entrada. A regra de entrada especifica o protocolo (TCP) e o número da porta (8111) para seu cluster. Depois que você adiciona essa regra ao seu security group, os aplicativos em execução na sua VPC podem acessar o cluster DAX.

ARN do cluster

A cada cluster DAX é atribuído umNome de recurso da Amazon(ARN). O formato do ARN é o seguinte.

```
arn:aws:dax:region:accountID:cache/clusterName
```

Você usa o ARN do cluster em uma política do IAM para definir permissões para operações de API do DAX. Para obter mais informações, consulte [Controle de acesso DAX \(p. 805\)](#).

Endpoint de cluster

Cada cluster DAX fornece umendpoint de clusterPara uso pelo seu aplicativo. Ao acessar o cluster usando o endpoint, o aplicativo não precisa saber os nomes de hosts e os números de portas de nós individuais no cluster. O aplicativo "conhece" automaticamente todos os nós no cluster, mesmo que você adicione ou remova réplicas de leitura.

Veja a seguir um exemplo de endpoint de cluster na região us-east-1 que não está configurada para usar criptografia em trânsito.

```
dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

Veja a seguir um exemplo de endpoint de cluster na mesma região que está configurada para usar criptografia em trânsito.

```
daxs://my-encrypted-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

Endpoints de nó

Cada um dos nós individuais em um cluster DAX tem seu próprio nome de host e número de porta. Veja a seguir um exemplo de endpoint de nó.

```
myDAXcluster-a.2cmrw1.clustercfg.dax.us-east-1.cache.amazonaws.com:8111
```

O aplicativo pode acessar um nó diretamente usando o endpoint desse nó. No entanto, recomendamos tratar o cluster DAX como uma unidade única e acessá-lo usando o endpoint do cluster. O endpoint do cluster faz com que o aplicativo não precise manter e atualizar uma lista de nós quando você adiciona ou remove nós do cluster.

Grupos de sub-redes

O acesso aos nós de cluster é restrito a aplicativos em execução em instâncias do Amazon EC2 em um ambiente da Amazon VPC. Você pode usar o Groups de sub-redePara conceder acesso ao cluster a partir de instâncias do Amazon EC2 em execução em sub-redes específicas. Um grupo de sub-redes é um conjunto de sub-redes (normalmente privadas) que você pode designar para seus clusters em execução em um ambiente da Amazon VPC.

Ao criar um cluster DAX, você deve especificar um grupo de sub-redes. O DAX usa esse grupo de sub-redes para selecionar uma sub-rede e endereços IP nessa sub-rede para associar aos seus nós.

Events

O DAX registra eventos significativos em seus clusters, tais como uma falha ao adicionar um nó, êxito ao adicionar um nó ou alterações nos security groups. Ao monitorar eventos importantes, você pode saber o estado atual dos seus clusters e, dependendo do evento, poderá executar a ação corretiva. Você pode acessar esses eventos usando o AWS Management Console ou o `DescribeEvents` API de gerenciamento DAX.

Você também pode solicitar que essas notificações sejam enviadas para um tópico específico do Amazon Simple Notification Service (Amazon SNS). Em seguida, você saberá imediatamente quando ocorrer um evento no cluster DAX.

Janela de manutenção

Cada cluster tem uma janela de manutenção semanal durante a qual todas as alterações do sistema são aplicadas. Se você não especificar uma janela de manutenção de sua preferência ao criar ou modificar um cluster de cache, o DAX atribuirá uma janela de manutenção de 60 minutos em um dia da semana selecionado aleatoriamente.

A janela de manutenção de 60 minutos é selecionada aleatoriamente de um bloco de tempo de 8 horas por AWSRegião : A seguinte tabela lista os blocos de tempo de cada região dos quais as janelas de manutenção padrão são atribuídas.

Código da região	Nome da região	Janela de manutenção
ap-northeast-1	Asia Pacific (Tokyo) Region	13:00 — 21:00 UTC
ap-southeast-1	Asia Pacific (Singapore) Region	14:00 - 22:00 UTC
ap-southeast-2	Asia Pacific (Sydney) Region	12:00 — 20:00 UTC
ap-south-1	Asia Pacific (Mumbai) Region	17:30 - 1:30 UTC

Código da região	Nome da região	Janela de manutenção
cn-northwest-1	Região China (Ningxia)	23:00 — 07:00 UTC
eu-central-1	Europe (Frankfurt) Region	23:00 — 07:00 UTC
eu-west-1	Europe (Ireland) Region	22:00 — 06:00 UTC
eu-west-2	Europe (London) Region	23:00 — 07:00 UTC
eu-west-3	Região Europa (Paris)	23:00 — 07:00 UTC
sa-east-1	South America (São Paulo) Region	01:00 — 09:00 UTC
us-east-1	US East (N. Virginia) Region	03:00 — 11:00 UTC
us-east-2	US East (Ohio) Region	23:00 — 07:00 UTC
us-west-1	US West (N. California) Region	06:00 — 14:00 UTC
us-west-2	US West (Oregon) Region	06:00 — 14:00 UTC

A janela de manutenção deve ser definida no horário de menor utilização e, portanto, talvez precise ser modificada de vez em quando. Você pode especificar um intervalo de tempo de até 24 horas de duração durante o qual todas as atividades de manutenção solicitadas devem ocorrer.

Criar um cluster DAX

Esta seção orienta você durante a primeira configuração e o uso do Amazon DynamoDB Accelerator (DAX) no ambiente da Amazon Virtual Private Cloud (Amazon VPC) padrão. Você pode criar seu primeiro cluster DAX usando aAWS Command Line Interface(AWS CLI) ou oAWS Management Console.

Depois de criar seu cluster DAX, você poderá acessá-lo de uma instância do Amazon EC2 em execução na mesma VPC. Você pode usar o cluster DAX com um programa aplicativo. Para obter mais informações, consulte [Como desenvolver com o cliente do DynamoDB Accelerator \(DAX\) \(p. 738\)](#).

Tópicos

- [Criar uma função de serviço do IAM para que o DAX acesse o DynamoDB \(p. 723\)](#)
- [Criar um cluster DAX usando oAWS CLI \(p. 725\)](#)
- [Criar um cluster DAX usando oAWS Management Console \(p. 729\)](#)

Criar uma função de serviço do IAM para que o DAX acesse o DynamoDB

Para que seu cluster DAX acesse as tabelas do DynamoDB em seu nome, será necessário criar umFunção de serviço do. Uma função de serviço é umAWS Identity and Access Management(IAM) que autoriza umAWSServiço da para agir em seu nome. A função de serviço permite que o DAX acesse as tabelas do DynamoDB, como se você mesmo estivesse acessando essas tabelas. Você deve criar a função de serviço antes de criar o cluster DAX.

Se você estiver usando o console, o fluxo de trabalho para a criação de um cluster verifica a presença de uma função de serviço DAX preexistente. Se nenhuma for encontrada, o console criará uma nova função

de serviço para você. Para obter mais informações, consulte [the section called “Etapa 2: Criar um cluster DAX” \(p. 730\)](#).

Se você estiver usando oAWS CLI, você deve especificar uma função de serviço DAX criada anteriormente. Caso contrário, é necessário criar uma nova função de serviço com antecedência. Para obter mais informações, consulte [Etapa 1: Crie uma função de serviço do IAM para que o DAX acesse o DynamoDB usando oAWS CLI \(p. 725\)](#).

Permissões necessárias para criar uma função de serviço

O gerenciado da AWSAdministratorAccessfornece todas as permissões necessárias para criar um cluster DAX e uma função de serviço. Se seu usuário do IAMAdministratorAccessanexada, nenhuma ação adicional é necessária.

Caso contrário, você deve adicionar as seguintes permissões à sua política do IAM para que seu usuário do IAM possa criar a função de serviço:

- `iam:CreateRole`
- `iam:CreatePolicy`
- `iam:AttachRolePolicy`
- `iam:PassRole`

Anexe essas permissões ao usuário que está tentando executar a ação.

Note

Oiam:CreateRole,iam:CreatePolicy,iam:AttachRolePolicy, eiam:PassRolePermissões do não são incluídas noAWSPolíticas gerenciadas do para o DynamoDB. Isso ocorre por design porque essas permissões fornecem a possibilidade de escalonamento de privilégios: Isto é, um usuário poderia usar essas permissões para criar uma nova política de administrador e anexá-la a uma função existente. Por esse motivo, você (o administrador do seu cluster DAX) deve adicionar explicitamente essas permissões à sua política.

Troubleshooting

Se sua política de usuário não tiver as permissões iam:CreateRole, iam:CreatePolicy e iam:AttachPolicy, você receberá mensagens de erro. A seguinte tabela lista essas mensagens e descreve como corrigir os problemas.

Se você vir essa mensagem de erro...	Faça o seguinte:
User: <code>arn:aws:iam::accountID:user/<i>userName</i></code> is not authorized to perform: <code>iam:CreateRole</code> on resource: <code>arn:aws:iam::accountID:role/service-role/<i>roleName</i></code>	Adicione <code>iam:CreateRole</code> à sua política de usuário.
User: <code>arn:aws:iam::accountID:user/<i>userName</i></code> is not authorized to perform: <code>iam:CreatePolicy</code> on resource: policy <code><i>policyName</i></code>	Adicione <code>iam:CreatePolicy</code> à sua política de usuário.
User: <code>arn:aws:iam::accountID:user/<i>userName</i></code>	Adicione <code>iam:AttachRolePolicy</code> à sua política de usuário.

Se você vir essa mensagem de erro...	Faça o seguinte:
<code>is not authorized to perform: iam:AttachRolePolicy on resource: role <i>daxServiceRole</i></code>	

Para obter mais informações sobre as políticas do IAM obrigatórias para a administração do cluster DAX, consulte [o Controle de acesso DAX \(p. 805\)](#).

Criar um cluster DAX usando oAWS CLI

Esta seção descreve como criar um cluster do Amazon DynamoDB Accelerator (DAX) usando oAWS Command Line Interface(AWS CLI). Caso ainda não tenha feito isso, você deve instalar e configurar a AWS CLI. Para fazer isso, consulte as seguintes instruções noAWS Command Line InterfaceGuia do usuário:

- [Instalar a AWS CLI](#)
- [Configuração do AWS CLI](#)

Important

Para gerenciar clusters DAX usando oAWS CLI, instale ou faça upgrade para a versão 1.11.110 ou superior.

Todos os exemplos da AWS CLI usam a região `us-west-2` e IDs de conta fictícios.

Tópicos

- [Etapa 1: Crie uma função de serviço do IAM para que o DAX acesse o DynamoDB usando oAWS CLI \(p. 725\)](#)
- [Etapa 2: Criar um grupo de sub-redes \(p. 727\)](#)
- [Etapa 3: Crie um cluster DAX usando oAWS CLI \(p. 727\)](#)
- [Etapa 4: Configure regras de entrada para o grupo de segurança usando oAWS CLI \(p. 728\)](#)

Etapa 1: Crie uma função de serviço do IAM para que o DAX acesse o DynamoDB usando oAWS CLI

Antes de criar um cluster do Amazon DynamoDB Accelerator (DAX), você deve criar uma função de serviço para ele. A função de serviço é um AWS Identity and Access Management(IAM) que autoriza um AWS Service da para agir em seu nome. A função de serviço permite que o DAX acesse as tabelas do DynamoDB como se você mesmo estivesse acessando essas tabelas.

Nesta etapa, você cria uma política do IAM e anexa essa política a uma função do IAM. Isso permite atribuir a função a um cluster do DAX para que ele possa realizar operações do DynamoDB em seu nome.

Como criar uma função de serviço do IAM para o DAX

1. Crie um arquivo denominado `service-trust-relationship.json` com o seguinte conteúdo.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {
```

```
        "Effect": "Allow",
        "Principal": {
            "Service": "dax.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }
}
```

- Crie a função de serviço.

```
aws iam create-role \
--role-name DAXServiceRoleForDynamoDBAccess \
--assume-role-policy-document file://service-trust-relationship.json
```

- Crie um arquivo denominado `service-role-policy.json` com o seguinte conteúdo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dynamodb:DescribeTable",
                "dynamodb:PutItem",
                "dynamodb:GetItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:ConditionCheckItem"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:dynamodb:us-west-2:accountID:*"
            ]
        }
    ]
}
```

Substituir **accountID** Com suas receitasAWSID da conta da. Para encontrar suas receitasAWSID da conta, no canto superior direito do console do, escolha seu ID de login. SuasAWSID da conta aparece no menu suspenso.

No nome de recurso da Amazon (ARN) do exemplo, **accountID** deve ser um número de 12 dígitos. Não use hifens ou qualquer outro sinal de pontuação.

- Crie uma política do IAM para a função de serviço.

```
aws iam create-policy \
--policy-name DAXServicePolicyForDynamoDBAccess \
--policy-document file://service-role-policy.json
```

Na saída, anote o ARN da política que você criou, como no exemplo abaixo.

arn:aws:iam::123456789012:policy/DAXServicePolicyForDynamoDBAccess

- Anexe a política à função de serviço. Substitua **arn** no seguinte código pelo ARN real da função da etapa anterior.

```
aws iam attach-role-policy \
```

```
--role-name DAXServiceRoleForDynamoDBAccess \
--policy-arn arn
```

Depois, especifique um grupo de sub-redes para a VPC padrão. Um grupo de sub-redes é uma coleção de uma ou mais sub-redes na sua VPC. Consulte [Etapa 2: Criar um grupo de sub-redes \(p. 727\)](#).

Etapa 2: Criar um grupo de sub-redes

Siga este procedimento para criar um grupo de sub-redes para o cluster do Amazon DynamoDB Accelerator (DAX) usando o AWS Command Line Interface(AWS CLI).

Note

Se você já criou um grupo de sub-redes para sua VPC padrão, pode ignorar esta etapa.

O DAX foi projetado para ser executado dentro de um ambiente do Amazon Virtual Private Cloud (Amazon VPC). Se você criou seu AWS Depois de 4 de dezembro de 2013, você já tem uma VPC padrão em cada AWS Região : Para obter mais informações, consulte [Padrão VPC e sub-redes padrão](#) no Guia do usuário da Amazon VPC.

Como criar um grupo de sub-redes

1. Para determinar o identificador da VPC padrão, insira o seguinte comando.

```
aws ec2 describe-vpcs
```

Na saída, anote o identificador da VPC padrão, como no seguinte exemplo.

vpc-12345678

2. Determine os IDs das sub-redes associadas à VPC padrão. Substituir *vpcID* Com sua ID de VPC real — por exemplo,vpc-12345678.

```
aws ec2 describe-subnets \
--filters "Name=vpc-id,Values=vpcID" \
--query "Subnets[*].SubnetId"
```

Na saída, anote os identificadores de sub-redes — por exemplo,subnet-11111111.

3. Crie o grupo de sub-redes. Certifique-se de especificar pelo menos um ID de sub-rede no parâmetro --subnet-ids.

```
aws dax create-subnet-group \
--subnet-group-name my-subnet-group \
--subnet-ids subnet-11111111 subnet-22222222 subnet-33333333 subnet-44444444
```

Para criar o cluster, consulte [Etapa 3: Crie um cluster DAX usando oAWS CLI \(p. 727\)](#).

Etapa 3: Crie um cluster DAX usando oAWS CLI

Siga este procedimento para usar oAWS Command Line Interface(AWS CLI) para criar um cluster do Amazon DynamoDB Accelerator (DAX) na Amazon VPC padrão.

Como criar um cluster do DAX

1. Obtenha o nome de recurso da Amazon (ARN) para a sua função de serviço.

```
aws iam get-role \
    --role-name DAXServiceRoleForDynamoDBAccess \
    --query "Role.Arn" --output text
```

Na saída, anote o ARN da função de serviço, como no seguinte exemplo.

```
arn:aws:iam::123456789012:role/DAXServiceRoleForDynamoDBAccess
```

- Crie o cluster do DAX. Substitua **roleARN** pelo ARN da etapa anterior.

```
aws dax create-cluster \
    --cluster-name mydaxcluster \
    --node-type dax.r4.large \
    --replication-factor 3 \
    --iam-role-arn roleARN \
    --subnet-group my-subnet-group \
    --sse-specification Enabled=true \
    --region us-west-2
```

Todos os nós do cluster são do tipo `dax.r4.large` (--node-type). Existem três nós (--replication-factor) — um nó primário e duas réplicas.

Note

Desesudoegrepão palavras-chave reservadas, você não pode criar um cluster DAX com essas palavras no nome do cluster. Por exemplo, `sudoesudo`clustersão nomes de cluster inválidos.

Para visualizar o status do cluster, insira o seguinte comando,

```
aws dax describe-clusters
```

O status é mostrado na saída — por exemplo, "Status": "creating".

Note

A criação do cluster demora vários minutos. Quando o cluster estiver pronto, seu status mudará para `available`. Enquanto isso, prossiga para [Etapa 4: Configure regras de entrada para o grupo de segurança usando oAWS CLI \(p. 728\)](#) e siga as instruções contidas ali.

Etapa 4: Configure regras de entrada para o grupo de segurança usando oAWS CLI

Os nós no cluster do Amazon DynamoDB Accelerator (DAX) usam o security group padrão para a Amazon VPC. Para o security group padrão, você deve autorizar o tráfego de entrada na porta TCP 8111 para clusters não criptografados ou a porta 9111 para clusters criptografados. Isso permite que instâncias do Amazon EC2 na Amazon VPC acessem o cluster DAX.

Note

Se você tiver iniciado o cluster DAX com um security group diferente (que não seja o (que não seja o**default**), você deve realizar esse procedimento para esse grupo em vez disso.

Como configurar regras de entrada para o grupo de segurança

- Para determinar o identificador do grupo de segurança padrão, digite o seguinte comando. Substitua **vpcID** pelo seu ID de VPC real (de [Etapa 2: Criar um grupo de sub-redes \(p. 727\)](#)).

```
aws ec2 describe-security-groups \
--filters Name=vpc-id,Values=vpcID,Name=group-name,Values=default \
--query "SecurityGroups[*].{GroupName:GroupName,GroupId:GroupId}"
```

Na saída, anote o identificador de security group — por exemplo, *sg-01234567*.

2. Insira o seguinte. Substitua *sgID* pelo seu identificador de grupo de segurança real. Usar porta *8111* para clusters não criptografados e *8111* para clusters criptografados.

```
aws ec2 authorize-security-group-ingress \
--group-id sgID --protocol tcp --port 8111
```

Criar um cluster DAX usando oAWS Management Console

Esta seção descreve como criar um cluster do Amazon DynamoDB Accelerator (DAX) usando oAWS Management Console.

Tópicos

- [Etapa 1: Crie um grupo de sub-redes usando oAWS Management Console \(p. 729\)](#)
- [Etapa 2: Crie um cluster DAX usando oAWS Management Console \(p. 730\)](#)
- [Etapa 3: Configure regras de entrada para o grupo de segurança usando oAWS Management Console \(p. 731\)](#)

Etapa 1: Crie um grupo de sub-redes usando oAWS Management Console

Siga este procedimento para criar um grupo de sub-redes para o cluster do Amazon DynamoDB Accelerator (DAX) usando oAWS Management Console.

Note

Se você já criou um grupo de sub-redes para sua VPC padrão, pode ignorar esta etapa.

O DAX foi projetado para ser executado dentro de um ambiente do Amazon Virtual Private Cloud (Amazon VPC). Se você criou seu AWS Depois de 4 de dezembro de 2013, você já tem uma VPC padrão em cada AWS Região : Para obter mais informações, consulte [Padrão VPC e sub-redes padrão](#) no Guia do usuário da Amazon VPC.

Como parte do processo de criação de um cluster DAX, você deve especificar um Grupo de sub-rede. Um grupo de sub-redes é uma coleção de uma ou mais sub-redes na VPC. Quando você cria o cluster DAX, os nós são implantados nas sub-redes dentro do grupo de sub-redes.

Como criar um grupo de sub-redes

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, escolha DAX.
3. Selecione Create subnet group (Criar grupo de sub-redes).
4. Na janela Create subnet group (Criar grupo de sub-redes), faça o seguinte:
 - a. Name (Nome)— insira um nome curto para o grupo de sub-redes.

- b. Descrição— insira uma descrição para o grupo de sub-redes.
- c. ID DA VPC— escolha o identificador do ambiente do Amazon VPC.
- d. Sub-redes— escolha uma ou mais sub-redes na lista.

Note

As sub-redes são distribuídas por várias Zonas de disponibilidade. Se você planeja criar um cluster do DAX de vários nós (um nó principal e uma ou mais réplicas de leitura), convém escolher vários IDs de sub-redes. Em seguida, o DAX pode implantar os nós do cluster em várias zonas de disponibilidade. Se uma zona de disponibilidade se tornar indisponível, o DAX fará o failover em uma zona de disponibilidade sobrevivente. O cluster DAX continuará a funcionar sem interrupção.

Quando estiver satisfeito com as configurações, escolha Create subnet group (Criar grupo de sub-rede).

Para criar o cluster, consulte [Etapa 2: Crie um cluster DAX usando oAWS Management Console \(p. 730\)](#).

Etapa 2: Crie um cluster DAX usando oAWS Management Console

Siga este procedimento para criar um cluster do Amazon DynamoDB Accelerator (DAX) na Amazon VPC padrão.

Como criar um cluster do DAX

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, em DAX, escolha Clusters.
3. Selecione Create cluster (Criar cluster).
4. Na janela Create cluster (Criar cluster), faça o seguinte:

- a. Nome do cluster— insira um nome curto para seu cluster DAX.

Note

Desdesudoegrepsão palavras-chave reservadas, você não pode criar um cluster DAX com essas palavras no nome do cluster. Por exemplo,sudoesudoclustersão nomes de cluster inválidos.

- b. Descrição do cluster— insira uma descrição para o cluster do.
- c. Tipo de nó— escolha o tipo de nó para todos os nós do cluster.
- d. Tamanho do cluster— escolha o número de nós no cluster. Um cluster consiste em um nó primário e em até nove réplicas de leitura.

Note

Para criar um cluster com um único nó, escolha 1. O cluster será composto por um único nó primário.

Para criar um cluster de vários nós, escolha um número entre 3 (um nó primário e duas réplicas de leitura) e 10 (um nó primário e nove réplicas de leitura).

Important

Para uso em produção, é altamente recomendável usar o DAX com pelo menos três nós, e posicionar cada nó em diferentes zonas de disponibilidade. São necessários três nós para que um cluster DAX seja tolerante a falhas.

Um cluster DAX pode ser implantado com um ou dois nós para cargas de trabalho de desenvolvimento ou de teste. Os clusters de um e dois nós não são tolerantes a falhas, portanto, não convém usar menos de três nós na produção. Se o cluster de um ou dois nós encontrar erros de software ou de hardware, ele poderá se tornar indisponível ou perder os dados armazenados em cache.

- e. Criptografia—ChooseEncryptionPara o cluster do DAX para ajudar a proteger os dados em repouso. Para obter mais informações, consulte [Criptografia DAX em repouso \(p. 815\)](#).
- f. Função de serviço do IAM para acesso ao DynamoDB—ChooseCriare insira as seguintes informações:
 - IAM role name (Nome da função do IAM)— insira um nome para uma função do IAM, por exemplo,`DAXServiceRole`. O console cria uma nova função do IAM, e o cluster DAX assume essa função em tempo de execução.
 - Política do IAM— insira um nome para uma política do IAM, por exemplo,`DAXServicePolicy`. O console cria uma nova política do IAM e anexa essa política à função do IAM.
 - Política de função do I—ChooseLeitura/gravação. Isso permite que o cluster DAX realize operações de leitura e gravação no DynamoDB.
 - Tabela do DynamoDB de destino—ChooseTodas as tabelas.
- g. Subnet group— escolha o grupo de sub-redes que você criou em[Etapa 1: Crie um grupo de sub-redes usando oAWS Management Console \(p. 729\)](#).
- h. Security groups do—Choosepadrão.

Uma função de serviço separada para o DAX acessar o Amazon EC2 também é necessária. O DAX cria automaticamente essa função de serviço para você. Para obter mais informações, consulte [Uso de funções vinculadas ao serviço para o DAX](#).

5. Quando estiver satisfeito com as configurações, escolha Launch cluster (Iniciar cluster).

NoClustersO cluster DAX será listado com um status deCriar.

Note

A criação do cluster demora vários minutos. Quando o cluster estiver pronto, seu status mudará para Available (Disponível).

Enquanto isso, prossiga para [Etapa 3: Configure regras de entrada para o grupo de segurança usando oAWS Management Console \(p. 731\)](#) e siga as instruções contidas ali.

Etapa 3: Configure regras de entrada para o grupo de segurança usando oAWS Management Console

Seu cluster do Amazon DynamoDB Accelerator (DAX) se comunica via porta TCP 8111 (para clusters não criptografados) ou 9111 (para clusters criptografados), portanto, você deve autorizar o tráfego de entrada nessa porta. Isso permite que instâncias do Amazon EC2 na Amazon VPC acessem o cluster DAX.

Note

Se você tiver iniciado o cluster DAX com um security group diferente (que não seja o (que não seja o`default`), você deve realizar esse procedimento para esse grupo em vez disso.

Como configurar regras de entrada para o grupo de segurança

1. Abra o console do Amazon EC2 em <https://console.aws.amazon.com/ec2/>.
2. No painel de navegação, selecione Grupos de segurança.
3. Escolha o grupo de segurança default (padrão). No menu Actions (Ações), escolha Edit inbound rules (Editar regras de entrada).

4. Escolha Add Rule (Adicionar regra) e insira as informações a seguir:

- Intervalo de Portas—Enter8111(se o cluster não estiver criptografado) ou9111(se o cluster estiver criptografado).

Origem—EnterpadrãoE escolha o identificador do security group padrão.

Quando estiver satisfeito com as configurações, clique em Salvar.

Modelos de consistência DAX e DynamoDB

O Amazon DynamoDB Accelerator (DAX) é um serviço de armazenamento em cache por gravação simultânea projetado para simplificar o processo de adição de um cache a tabelas do DynamoDB. Como o DAX opera separadamente do DynamoDB, é importante compreender os modelos de consistência do DAX e do DynamoDB para garantir que os aplicativos se comportem conforme esperado.

Em muitos casos de uso, a maneira como seu aplicativo usa o DAX afeta a consistência dos dados dentro no cluster do DAX e a consistência dos dados entre o DAX e o DynamoDB.

Tópicos

- [Consistência entre nós de cluster do DAX \(p. 732\)](#)
- [Comportamento do cache de itens do \(p. 732\)](#)
- [Comportamento do cache de consultas do \(p. 735\)](#)
- [Leituras fortemente consistentes e transacionais \(p. 735\)](#)
- [Armazenamento em cache negativo \(p. 736\)](#)
- [Estratégias para gravações \(p. 736\)](#)

Consistência entre nós de cluster do DAX

Para obter alta disponibilidade para o aplicativo, recomendamos provisionar o cluster DAX com pelo menos três nós. Além disso, coloque esses nós em várias zonas de disponibilidade dentro de uma região.

Quando o cluster do DAX estiver em execução, ele replicará os dados entre todos os nós do cluster (supondo que você tenha provisionado mais de um nó). Considere um aplicativo que realize uma operação bem-sucedidaUpdateItemusando DAX. Essa ação faz com que o cache de itens no nó primário seja modificado com o novo valor. Esse valor é, então, replicado em todos os outros nós no cluster. Essa replicação é eventualmente consistente e costuma demorar menos de um segundo para ser concluída.

Nesse cenário, é possível que dois clientes leiam a mesma chave do mesmo cluster DAX, mas recebam valores diferentes, dependendo do nó que cada cliente acessou. Os nós estarão todos consistentes quando a atualização tiver sido totalmente replicada por todos os nós do cluster. (Esse comportamento é semelhante à natureza eventualmente consistente do DynamoDB.)

Se você estiver criando um aplicativo que usa o DAX, esse aplicativo deverá ser projetado para tolerância a dados eventualmente consistentes.

Comportamento do cache de itens do

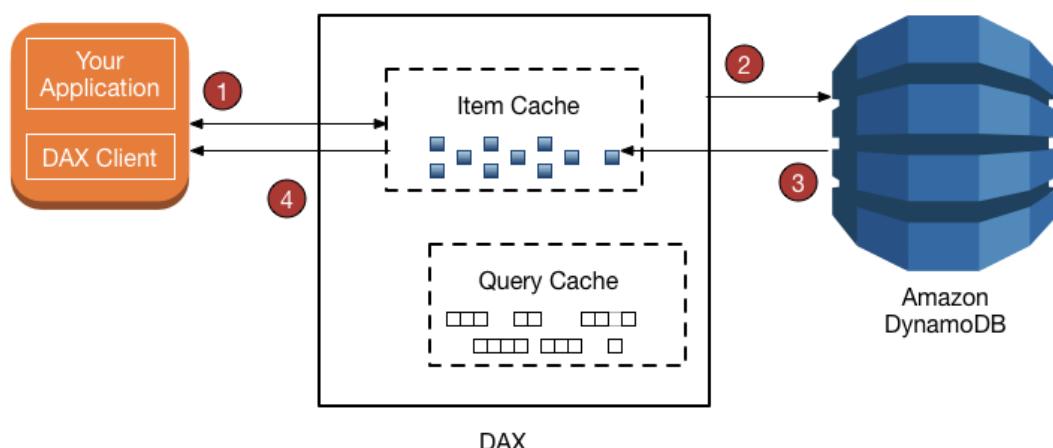
Cada cluster do DAX tem dois caches distintos: umitemCache e uma solicitaçãoquerycache. Para obter mais informações, consulte [DAX: Como ele funciona \(p. 715\)](#).

Esta seção aborda as implicações de consistência da leitura e da gravação no cache de itens do DAX.

Consistência das leituras

Com o DynamoDB, o `GetItem` operação executa uma leitura eventualmente consistente por padrão. Suponha que você use `UpdateItem` com o cliente DynamoDB. Se você tentar ler o mesmo item imediatamente, poderá ver os dados no estado em que estavam antes da atualização. Isso acontece devido ao atraso de propagação por todas as localizações de armazenamento do DynamoDB. A consistência normalmente é atingida em segundos. Portanto, se você repetir a leitura, provavelmente verá o item atualizado.

Quando você usa `GetItem` com o cliente DAX, a operação (neste caso, uma leitura eventualmente consistente) ocorre conforme mostrado abaixo.



1. O cliente do DAX emite uma solicitação `GetItem`. O DAX tenta ler o item solicitado do cache de itens. Se o item estiver no cache (acerto de cache), o DAX o retornará ao aplicativo.
2. Se o item não estiver disponível (falta de cache), o DAX executa uma `GetItem` operação no DynamoDB.
3. O DynamoDB retornará o item solicitado, e o DAX o armazenará no cache de itens.
4. O DAX retorna o item ao aplicativo.
5. (Não mostrado) Se o cluster do DAX contiver mais de um nó, o item será replicado a todos os outros nós do cluster.

O item permanece no cache de itens do DAX, sujeito à configuração de tempo de vida (TTL) e ao algoritmo de menos usado recentemente (LRU) do cache. Para obter mais informações, consulte [DAX: Como ele funciona \(p. 715\)](#).

No entanto, durante esse período, o DAX não lê novamente o item no DynamoDB. Se outra pessoa atualizar o item usando um cliente do DynamoDB, ignorando totalmente o DAX, um `GetItem` usando o cliente DAX produz resultados diferentes do mesmo `GetItem` usando o cliente DynamoDB. Nesse cenário, o DAX e o DynamoDB manterão valores inconsistentes para a mesma chave até que o TTL do item do DAX expire.

Se um aplicativo modificar dados em tabela do DynamoDB subjacente, ignorando o DAX, o aplicativo precisará antecipar e tolerar as inconsistências de dados que possam surgir.

Note

Além de `GetItem`, o cliente DAX também suporta `BatchGetItem` solicitações. `BatchGetItem` é essencialmente um wrapper ao redor de um ou mais `GetItem` As solicitações do DAX tratam cada um delas como uma `GetItem` operação.

Consistência das gravações

O DAX é um cache de gravação simultânea, o que simplifica o processo de manutenção do cache de itens do DAX consistente com as tabelas do DynamoDB subjacentes.

O cliente DAX oferece suporte às mesmas operações de API de gravação que o DynamoDB (`PutItem`, `UpdateItem`, `DeleteItem`, `BatchWriteItem`, e `TransactWriteItems`). Quando você usa essas operações com o cliente DAX, os itens são modificados tanto no DAX quanto no DynamoDB. O DAX atualiza os itens em seu cache de itens, independentemente do valor de TTL desses itens.

Por exemplo, suponha que você emita uma solicitação `GetItem` para ler um item do `ProductCatalogTabela` INTO. (A chave de partição é `Id` e não há uma chave de classificação). Você recupera o item cujo `Id` é 101. O `QuantityOnHand` valor para esse item é 42. O DAX armazena o item em seu cache de itens com um TTL específico. Para este exemplo, suponha que o TTL é de 10 minutos. Em seguida, 3 minutos depois, outro aplicativo usa o cliente DAX para atualizar o mesmo item de forma que o `QuantityOnHand` é agora 41. Supondo que o item não seja atualizado novamente, qualquer leitura subsequente do mesmo item durante os próximos dez minutos retornará o valor armazenado em cache a `QuantityOnHand` (41).

Como o DAX processa gravações

O DAX foi projetado para aplicativos que exigem leituras de alto desempenho. Como um cache de gravação, o DAX passa suas gravações para o DynamoDB de forma síncrona e, em seguida, replica automaticamente e assíncrona as atualizações resultantes para o cache de itens em todos os nós do cluster. Você não precisa gerenciar a lógica de invalidação de cache, pois o DAX lida com ela automaticamente.

O DAX oferece suporte às seguintes operações de gravação:
`PutItem`, `UpdateItem`, `DeleteItem`, `BatchWriteItem`, e `TransactWriteItems`.

Quando você envia um `PutItem`, `UpdateItem`, `DeleteItem`, ou `BatchWriteItem` ao DAX, ocorre o seguinte:

- O DAX envia a solicitação ao DynamoDB.
- O DynamoDB responde ao DAX, confirmado que a gravação foi bem-sucedida.
- O DAX grava o item em seu cache de itens.
- O DAX retorna uma resposta de êxito ao solicitante.

Quando você envia um `TransactWriteItem` ao DAX, ocorre o seguinte:

- O DAX envia a solicitação ao DynamoDB.
- O DynamoDB responde ao DAX, confirmado que a transação foi concluída.
- O DAX retorna uma resposta de êxito ao solicitante.
- Em segundo plano, o DAX faz um `TransactGetItem` para cada item `noTransactWriteItem` solicitação para armazenar o item no cache de itens. `TransactGetItem` é usado para garantir isolamento serializável (p. 683).

Se houver falha em uma gravação no DynamoDB por qualquer motivo, inclusive controle de utilização, o item não será armazenado em cache no DAX. A exceção da falha é retornada ao solicitante. Isso garante

que os dados apenas sejam gravados no cache do DAX se forem primeiramente gravados com êxito no DynamoDB.

Note

Toda gravação no DAX altera o estado do cache de itens. No entanto, as gravações no cache de itens não afeta o cache de consultas. (O cache de itens e o cache de consultas do DAX têm finalidades diferentes e operam de forma independente um do outro.)

Comportamento do cache de consultas do

O DAX armazena em cache os resultados de `Query` e `Scan` solicitações em seu cache de consultas. No entanto, esses resultados não afetam de forma alguma o cache de itens. Quando seu aplicativo emite um `Query` ou `Scan` no DAX, o conjunto de resultados é salvo no cache de consultas, e não no cache de itens. Você não pode "aquecer" o cache de itens executando uma operação `Scan` porque o cache de itens e o cache de consultas são entidades separadas.

Consistência da consulta de atualização de consultas

As atualizações no cache de itens, ou na tabela subjacente do DynamoDB, não invalidam nem modificam os resultados armazenados no cache de consultas.

Para ilustrar, considere o seguinte cenário. Um aplicativo está trabalhando com a tabela `DocumentRevisions`, que tem `DocId` como chave de partição e `RevisionNumber` como chave de classificação.

1. Um cliente do emite uma solicitação `Query` para `DocId` 101, para todos os itens com `RevisionNumber` maior ou igual a 5. O DAX armazena o conjunto de resultados no cache de consultas e retorna esse conjunto ao usuário.
2. O cliente do emite uma solicitação `PutItem` para a solicitação de `DocId` 101 com um `RevisionNumber` O valor de 20.
3. O cliente do emite o mesmo `Query` conforme descrito na etapa 1 (`DocId` 101 e `RevisionNumber >= 5`).

Neste cenário, o conjunto de resultados armazenado em cache para a `Query` emitida na etapa 3 será idêntico ao conjunto de resultados que foi armazenado em cache na etapa 1. A razão é que o DAX não invalida `Query` ou `Scan`. Conjuntos de resultados com base em atualizações de itens individuais. O `PutItem` operação da etapa 2 é refletida apenas no cache de consultas do DAX quando o TTL do `Query` expira.

Seu aplicativo deve considerar o valor do TTL do cache de consultas e por quanto tempo pode tolerar resultados inconsistentes entre o cache de consultas e o cache de itens.

Leituras fortemente consistentes e transacionais

Para realizar uma operação fortemente consistente `GetItem`, `BatchGetItem`, `Query`, ou `Scan`, você define a propriedade `ConsistentRead` como `true`. O DAX transfere solicitações de leitura fortemente consistente ao DynamoDB. Quando recebe uma resposta do DynamoDB, o DAX retorna os resultados ao cliente, mas não armazena os resultados em cache. O DAX não pode servir leituras fortemente consistentes por conta própria, porque ele não está firmemente acoplado ao DynamoDB. Por esse motivo, todas as leituras subsequentes do DAX precisariam ser leituras eventualmente consistentes. E todas as leituras fortemente consistentes subsequentes precisariam ser passadas para o DynamoDB.

Alças DAX `TransactGetItems` O solicita da mesma forma como trata leituras fortemente consistentes. DAX passa todos `TransactGetItems` ao DynamoDB. Quando recebe uma resposta do DynamoDB, o DAX retorna os resultados ao cliente, mas não armazena os resultados em cache.

Armazenamento em cache negativo

O DAX oferece suporte a entradas de cache negativas, tanto no cache de itens quanto no cache de consultas. A Entrada negativa do cache ocorre quando o DAX não consegue localizar os itens solicitados em uma tabela do DynamoDB subjacente. Em vez de gerar um erro, o DAX armazena em cache um resultado vazio e retorna esse resultado ao usuário.

Por exemplo, suponha que um aplicativo envie uma solicitação `GetItem` a um cluster DAX e que não haja itens correspondentes no cache de itens do DAX. Isso faz com que o DAX leia o item correspondente da tabela do DynamoDB subjacente. Se o item não existir no DynamoDB, o DAX armazenará um item vazio em seu cache de itens e retornará esse item vazio ao aplicativo. Agora suponha que o aplicativo envia outra solicitação para o mesmo item. DAX encontra o item vazio no cache de itens e o retorna imediatamente ao aplicativo. Ele não consulta o DynamoDB.

Uma entrada de cache negativa permanece no cache de itens do DAX até que o TTL do item expire, a LRU seja invocada ou o item seja modificado usando `PutItem`, `UpdateItem`, ou `DeleteItem`.

O cache de consultas DAX manipula resultados de cache negativos de maneira semelhante. Se um aplicativo executar um `Query` ou `Scan` e o cache de consultas DAX não contiver um resultado armazenado em cache, o DAX enviará a solicitação ao DynamoDB. Se não houver itens correspondentes no conjunto de resultados, o DAX armazenará um conjunto de resultados vazio no cache de consultas e retornará esse conjunto vazio ao aplicativo. Solicitações `Query` ou `Scan` subsequentes produzem o mesmo conjunto de resultados (vazio), até que o TTL desse conjunto de resultados tenha expirado.

Estratégias para gravações

O comportamento de gravação simultânea do DAX é apropriado para muitos padrões de aplicativo. No entanto, existem alguns padrões de aplicativo em que um modelo de gravação simultânea pode não ser apropriado.

Para aplicativos sensíveis à latência, a gravação por meio do DAX incorre em um salto de rede extra. Portanto, uma gravação no DAX é um pouco mais lenta que uma gravação diretamente no DynamoDB. Se o seu aplicativo for sensível à latência de gravação, você poderá reduzir essa latência gravando diretamente no DynamoDB em vez disso. Para obter mais informações, consulte [Write-Around \(gravação direta\) \(p. 737\)](#).

Para aplicativos que exigem gravação intensa (como aqueles que realizam o carregamento de dados em massa), talvez não seja conveniente gravar todos os dados por meio do DAX, pois apenas uma porcentagem pequena deles é lida pelo aplicativo. Quando você grava grandes quantidades de dados no DAX, ele deve chamar seu algoritmo LRU de forma a liberar espaço no cache para que os novos itens sejam lidos. Isso diminui a eficácia do DAX como um cache de leitura.

Quando você grava um item no DAX, o estado do cache de itens é alterado para acomodar o novo item. (Por exemplo, o DAX talvez precise remover dados antigos do cache de itens para liberar espaço para o novo item.) O novo item permanece no cache de itens, sujeito ao algoritmo LRU do cache e à configuração de TTL do cache. Enquanto o item persistir no cache de itens, o DAX não repetirá sua leitura no DynamoDB.

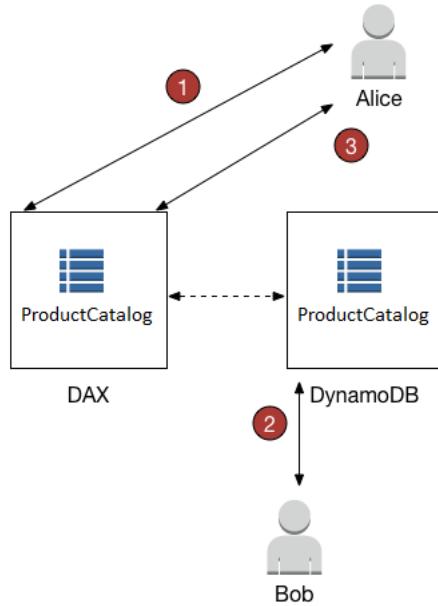
Gravação simultânea (write-through)

O cache de itens do DAX implementa uma política de gravação simultânea. Para obter mais informações, consulte [Como o DAX processa gravações \(p. 734\)](#).

Quando você grava um item, o DAX garante que o item em cache seja sincronizado com o item existente no DynamoDB. Isto é útil para aplicativos que precisam repetir a leitura de um item logo depois de gravá-

lo. No entanto, se outros aplicativos gravarem diretamente em uma tabela do DynamoDB, o item no cache de itens do DAX deixará de estar sincronizado com o DynamoDB.

Para ilustrar isso, considere dois usuários (Alice e Bob) que estão trabalhando com a tabela `ProductCatalog`. Alice acessa a tabela usando DAX, mas Bob ignora o DAX e acessa a tabela diretamente no DynamoDB.



1. Alice atualiza um item no `ProductCatalog` Tabela INTO. O DAX encaminha a solicitação ao DynamoDB e a atualização é bem-sucedida. Em seguida, o DAX grava o item em seu cache de itens e retorna uma resposta bem-sucedida a Alice. Desse ponto em diante, até que o item seja finalmente retirado do cache, qualquer usuário que o ler do DAX o verá com a atualização de Alice.
2. Pouco depois, Bob atualiza o mesmo item do `ProductCatalog` gravado por Alice. No entanto, Bob atualiza o item diretamente no DynamoDB. O DAX não atualiza automaticamente o cache de itens em resposta a atualizações via DynamoDB. Portanto, os usuários do DAX não veem a atualização de Bob.
3. Alice lê novamente o item do DAX. O item está no cache de itens e, portanto, o DAX o retorna à Alice sem acessar a tabela do DynamoDB.

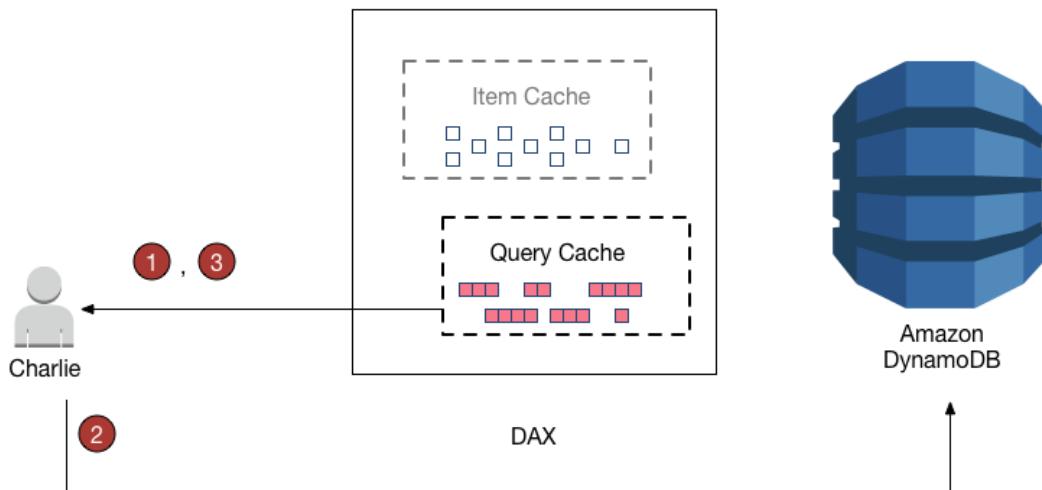
Nesse cenário, Alice e Bob veem representações diferentes do mesmo item do `ProductCatalog`. Esse será o caso até que o DAX remova o item do cache de itens ou até que outro usuário atualize o mesmo item novamente usando o DAX.

Write-Around (gravação direta)

Se o seu aplicativo precisa gravar grandes quantidades de dados (como um carregamento em massa de dados), talvez faça sentido ignorar o DAX e gravar os dados diretamente no DynamoDB. Essa estratégia de gravação direta (write-around) reduz a latência de gravação. No entanto, o cache de itens não permanece em sincronia com os dados no DynamoDB.

Se você optar por usar uma estratégia de gravação direta, lembre-se de que o DAX preenche o cache de itens sempre que os aplicativos usam o cliente DAX para ler os dados. Isso pode ser vantajoso em alguns casos porque garante que apenas os dados lidos com mais frequência sejam armazenados em cache (não os dados gravados com mais frequência).

Por exemplo, considere um usuário, Charlie, que deseja trabalhar com uma tabela diferente, oGameScores, usando DAX. A chave de partição de GameScores é UserId e, portanto, todas as pontuações de Charlie teriam o mesmo UserId.



1. Charlie deseja recuperar todos os seus escores e, para isso, envia uma `Query` ao DAX. Supondo que essa consulta não foi emitida antes, o DAX encaminha a consulta ao DynamoDB para processamento. Ele armazena os resultados no cache de consultas do DAX e retorna os resultados a Charlie. O conjunto de resultados permanece disponível no cache de consultas até ser removido.
2. Agora, suponha que Charlie jogue uma partida de Meteor Blasters e atinja um escore alto. Charlie envia uma solicitação `UpdateItem` ao DynamoDB, modificando um item na `GameScores` Tabela INTO.
3. Por fim, Charlie decide executar novamente sua `Query` anterior para recuperar todos os dados de `GameScores`. Charlie não vê sua alta pontuação do jogo Meteor Blasters nos resultados. Isso ocorre porque os resultados da consulta vêm do cache de consultas, e não do cache de itens. Os dois caches são independentes um do outro e, portanto, uma alteração em um cache não afeta o outro.

O DAX não atualiza conjuntos de resultados no cache de consultas com os dados mais atuais do DynamoDB. Cada conjunto de resultados no cache de consultas é atual no momento em que a operação `Query` ou `Scan` foi executada. Portanto, os resultados da `Query` de Charlie não refletem sua operação `PutItem`. Esse será o caso até que o DAX remova o conjunto de resultados do cache de consultas.

Como desenvolver com o cliente do DynamoDB Accelerator (DAX)

Para usar o DAX de um aplicativo, use o cliente DAX do para sua linguagem de programação. O cliente DAX foi projetado para o mínimo de interrupção nos seus aplicativos existentes do Amazon DynamoDB — com apenas algumas modificações simples necessárias de código.

Note

Clientes do DAX para várias linguagens de programação estão disponíveis no seguinte site:

- <http://dax-sdk.s3-website-us-west-2.amazonaws.com>

Esta seção demonstra como iniciar uma instância do Amazon EC2 na sua Amazon VPC padrão, conectar-se a essa instância e executar um aplicativo de exemplo. Ela também fornece informações sobre como modificar seu aplicativo existente para que ele possa usar seu cluster do DAX.

Tópicos

- [Tutorial: Como executar um aplicativo de exemplo usando o DynamoDB Accelerator \(DAX\) \(p. 739\)](#)
- [Como modificar um aplicativo existente para usar o DAX \(p. 776\)](#)
- [Consultar índices secundários globais \(p. 779\)](#)

Tutorial: Como executar um aplicativo de exemplo usando o DynamoDB Accelerator (DAX)

Este tutorial demonstra como executar uma instância do Amazon EC2 na sua nuvem privada virtual (VPC) padrão, conectar-se à instância e executar um aplicativo de amostra que use o Amazon DynamoDB Accelerator (DAX).

Note

Para completar este tutorial, você deve ter um cluster do DAX em execução na sua VPC padrão. Se você ainda não criou um cluster DAX, consulte [Criar um cluster DAX \(p. 723\)](#) Para obter instruções.

Tópicos

- [Etapa 1: Executar uma instância do Amazon EC2 \(p. 739\)](#)
- [Etapa 2: Criar um usuário do IAM e uma política \(p. 740\)](#)
- [Etapa 3: Configurar uma instância do Amazon EC2 \(p. 741\)](#)
- [Etapa 4: Executar um aplicativo de exemplo \(p. 742\)](#)

Etapa 1: Executar uma instância do Amazon EC2

Quando seu cluster do Amazon DynamoDB Accelerator (DAX) estiver disponível, você poderá iniciar uma instância do Amazon EC2 na sua Amazon VPC padrão. Assim, será possível instalar e executar o software cliente do DAX nessa instância.

Para iniciar uma instância do EC2

1. Faça login no AWS Management Console e abra o console do Amazon EC2 em <https://console.aws.amazon.com/ec2/>.
2. Selecione Launch Instance (Executar instância) e faça o seguinte:

Etapa 1: Escolha uma Imagem de máquina da Amazon (AMI)

1. Na lista de AMIs, localize Amazon Linux AMI (AMI do Amazon Linux) e escolha Select (Selecionar).

Etapa 2: Escolha um tipo de instância

1. Na lista de tipos de instância, escolha t2.micro.
2. Selecione Next (Próximo): Configurar os detalhes da instância.

Etapa 3: Configurar os detalhes da instância

1. Para Network (Rede), escolha a nuvem privada virtual (VPC) padrão.
2. Selecione Next (Próximo): Add Storage.

Etapa 4: Add Storage

1. Pule essa etapa escolhendo Próximo: Adicionar tags.

Etapa 5: Add Tags

1. Pule essa etapa escolhendo Próximo: Configurar o grupo de segurança.

Etapa 6: Configurar o grupo de segurança

1. Escolha Selecionar um security group existente.
2. Na lista de security groups, escolha padrão. Este é o security group padrão para sua VPC.
3. Selecione Next (Próximo): Revisar e executar.

Etapa 7: Revisar o lançamento da instância

1. Escolha Executar.
3. Na janela Selecionar um par de chaves existente ou criar um novo par de chaves, faça um dos seguintes:
 - Se você não tiver um key pair do Amazon EC2, escolha Criar um novo key pair e siga as instruções. É solicitado que você faça download de um arquivo de chave privada (arquivo .pem). Você precisará deste arquivo mais tarde quando fizer login na instância do Amazon EC2.
 - Se você já tiver um key pair existente do Amazon EC2, acesse Selecionar um par de chaves e escolha o seu key pair na lista. Você já deve ter o arquivo de chave privada (.pem) disponível para fazer login na instância do Amazon EC2.
4. Depois de configurar seu par de chaves, escolha Launch instances (Executar instâncias).
5. No painel de navegação do console, escolha EC2 Dashboard (Painel do EC2) e, em seguida, escolha a instância que você iniciou. No painel inferior, no Descrição, localize o DNS público para sua instância, por exemplo: ec2-11-22-33-44.us-west-2.compute.amazonaws.com. Anote esse nome DNS público, pois você precisará dele para [Etapa 3: Configurar uma instância do Amazon EC2 \(p. 741\)](#).

Note

Serão necessários alguns minutos para que a instância do Amazon EC2 se torne disponível. Enquanto isso, prossiga para [Etapa 2: Criar um usuário do IAM e uma política \(p. 740\)](#) e siga as instruções contidas ali.

Etapa 2: Criar um usuário do IAM e uma política

Nesta etapa, você cria um AWS Identity and Access Management Usuário do (IAM) com uma política que concede acesso ao seu cluster do Amazon DynamoDB Accelerator (DAX) e ao DynamoDB. Assim, você poderá executar aplicativos que interagem com seu cluster do DAX.

Como criar um usuário e uma política do IAM

1. Abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. No painel de navegação, escolha Usuários.
3. Selecione Add user.

4. Na página Details (Detalhes), insira as seguintes informações:
 - Nome de usuário— insira um nome exclusivo, por exemplo:`MyDAXUser`.
 - Tipo de acesso—EscolhaAcesso programático.

Selezione Next (Próximo): Permissions
5. Na página Set permissions (Definir permissões), escolha Attach existing policies directly (Anexar políticas existentes diretamente) e escolha Create policy (Criar política).
6. Na página Create policy (Criar política), escolha Create Your Own Policy (Criar sua própria política).
7. Na página Review policy (Revisar política), forneça as seguintes informações:
 - Nome da política— insira um nome exclusivo, por exemplo:`MyDAXUserPolicy`.
 - Descrição— insira uma breve descrição para a política.
 - Documento da política— Copie e cole o seguinte documento.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "dax:*"  
            ],  
            "Effect": "Allow",  
            "Resource": [  
                "*"  
            ]  
        },  
        {  
            "Action": [  
                "dynamodb:*"  
            ],  
            "Effect": "Allow",  
            "Resource": [  
                "*"  
            ]  
        }  
    ]  
}
```

8. Retorne para a página Permissions (Permissões). Na lista de políticas, escolha Refresh (Atualizar).
9. Para restringir a lista de políticas, escolha Filter (Filtro), Customer managed (Cliente gerenciado). Escolha a política do IAM criada na etapa anterior (por exemplo:`MyDAXUserPolicy`) e, depois, escolha Próximo: Análise.
10. Na página Review, selecione Create user.
11. Na página Complete (Concluir), vá para Secret access key (Chave de acesso secreta) e escolha Show (Mostrar). Após fazer isso, copie o Access key ID (ID da chave de acesso) e a Secret access key (Chave de acesso secreta). Você precisará de ambos os identificadores em [Etapa 3: Configurar uma instância do Amazon EC2 \(p. 741\)](#).

Etapa 3: Configurar uma instância do Amazon EC2

Quando sua instância do Amazon EC2 estiver disponível, você poderá fazer login na instância e prepará-la para uso.

Note

As etapas a seguir assumem que você está se conectando à sua instância do Amazon EC2 de um computador que executa o Linux. Para conhecer outras formas de se conectar, consulte [Conecte-se à sua instância do Linux](#) no Guia do usuário do Amazon EC2 para instâncias do Linux.

Como configurar a instância do EC2

1. Abra o console do Amazon EC2 em <https://console.aws.amazon.com/ec2/>.
2. Usar sshPara fazer login na sua instância do Amazon EC2, conforme mostrado no seguinte exemplo.

```
ssh -i my-keypair.pem ec2-user@public-dns-name
```

Você precisará especificar seu arquivo de chave privada (arquivo .pem) e o nome DNS público da instância. (Consulte [Etapa 1: Executar uma instância do Amazon EC2 \(p. 739\)](#).)

O ID de login é ec2-user. Nenhuma senha é necessária.

3. Depois de fazer login na instância do EC2, configure oAWSCredenciais, conforme mostrado a seguir. Insira o seuAWSAcesse a chave do e a chave secreta (de[Etapa 2: Criar um usuário do IAM e uma política \(p. 740\)](#)) e defina o nome da região padrão como a sua região atual. (No exemplo a seguir, o nome de região padrão é us-west-2.)

```
aws configure

AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]:
```

Após a execução e configuração da sua instância do Amazon EC2, você pode testar a funcionalidade do DAX usando um dos aplicativos de exemplo disponíveis. Para obter mais informações, consulte [Etapa 4: Executar um aplicativo de exemplo \(p. 742\)](#).

Etapa 4: Executar um aplicativo de exemplo

Para ajudar a testar a funcionalidade do Amazon DynamoDB Accelerator (DAX), você pode executar um dos aplicativos de exemplo disponíveis na instância do Amazon EC2.

Tópicos

- [SDK do DAX para Go \(p. 742\)](#)
- [Java e DAX \(p. 744\)](#)
- [.NET e DAX \(p. 753\)](#)
- [Node.js e DAX \(p. 762\)](#)
- [Python e DAX \(p. 770\)](#)

SDK do DAX para Go

Siga este procedimento para executar o aplicativo de exemplo do Amazon DynamoDB Accelerator (DAX) SDK for Go na instância do Amazon EC2.

Para executar a amostra do SDK for Go para DAX

- Configure o SDK for Go em sua instância do Amazon EC2:

- Instale a linguagem de programação Go (Golang).

```
sudo yum install -y golang
```

- Verifique se o Golang está instalado e funcionando corretamente.

```
go version
```

Uma mensagem como esta deve aparecer.

```
go version go1.15.5 linux/amd64
```

As instruções restantes dependem do suporte do módulo, que se tornou o padrão com Go versão 1.13.

- Instale o aplicativo de exemplo do Golang.

```
go get github.com/aws-samples/aws-dax-go-sample
```

- Execute os seguintes programas do Golang. O primeiro programa cria uma tabela do DynamoDB denominada TryDaxGoTable. O segundo programa grava dados na tabela.

```
go run ~/go/src/github.com/aws-samples/aws-dax-go-sample/try_dax.go -service dynamodb - command create-table
```

```
go run ~/go/src/github.com/aws-samples/aws-dax-go-sample/try_dax.go -service dynamodb - command put-item
```

- Execute os seguintes programas do Golang.

```
go run ~/go/src/github.com/aws-samples/aws-dax-go-sample/try_dax.go -service dynamodb - command get-item
```

```
go run ~/go/src/github.com/aws-samples/aws-dax-go-sample/try_dax.go -service dynamodb - command query
```

```
go run ~/go/src/github.com/aws-samples/aws-dax-go-sample/try_dax.go -service dynamodb - command scan
```

Anote as informações de tempo: o número de milissegundos necessários para os testes GetItem, Query e Scan.

- Na etapa anterior, você executou os programas no endpoint do DynamoDB. Agora, execute os programas novamente, mas, desta vez, o GetItem, Query, e Scan são processadas pelo cluster DAX.

Para determinar o endpoint do cluster do DAX, escolha uma das seguintes opções:

- Uso do console do DynamoDB— Escolha o cluster do DAX. O endpoint do cluster é mostrado no console, como no seguinte exemplo.

```
dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

- Usar o AWS CLI— Insira o comando a seguir.

```
aws dax describe-clusters --query "Clusters[*].ClusterDiscoveryEndpoint"
```

O endpoint do cluster é mostrado na saída, como no seguinte exemplo.

```
{  
    "Address": "my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com",  
    "Port": 8111,  
    "URL": "dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com"  
}
```

Agora, execute os programas novamente, mas, desta vez, especifique o endpoint do cluster como um parâmetro de linha de comando.

```
go run ~/go/src/github.com/aws-samples/aws-dax-go-sample/try_dax.go -service dax -  
command get-item -endpoint dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

```
go run ~/go/src/github.com/aws-samples/aws-dax-go-sample/try_dax.go -service dax -  
command query -endpoint dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

```
go run ~/go/src/github.com/aws-samples/aws-dax-go-sample/try_dax.go -service dax -  
command scan -endpoint dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

Observe o restante da saída e anote as informações de tempo. Os tempos decorridos para GetItem, Query, e ScanO deve ser significativamente mais baixo no DAX do que no DynamoDB.

6. Execute o seguinte programa Golang para excluir a TryDaxGoTable.

```
go run ~/go/src/github.com/aws-samples/aws-dax-go-sample/try_dax.go -service dynamodb -  
command delete-table
```

Java e DAX

Siga este procedimento para executar o Java de exemplo para o Amazon DynamoDB Accelerator (DAX) na instância do Amazon EC2.

Como executar o Java de exemplo para o DAX

1. Instale o Java Development Kit (JDK).

```
sudo yum install -y java-devel
```

2. Faça download do AWS SDK for Java (arquivo .zip) e extraia-o.

```
wget http://sdk-for-java.amazonwebservices.com/latest/aws-java-sdk.zip  
unzip aws-java-sdk.zip
```

3. Faça download da versão mais recente do cliente Java do DAX (.jar).

```
wget http://dax-sdk.s3-website-us-west-2.amazonaws.com/java/DaxJavaClient-latest.jar
```

Note

O cliente do SDK do DAX para Java está disponível no Apache Maven. Para obter mais informações, consulte [Uso do cliente como dependência do Apache Maven \(p. 747\)](#).

4. Defina a variável CLASSPATH. Neste exemplo, substitua `sdkVersion` com o número de versão atual do AWS SDK for Java (por exemplo, 1.11.112).

```
export SDKVERSION=sdkVersion

export CLASSPATH=$(pwd)/TryDax/java:$(pwd)/DaxJavaClient-latest.jar:$(pwd)/aws-java-
sdk-$SDKVERSION/lib/aws-java-sdk-$SDKVERSION.jar:$(pwd)/aws-java-sdk-$SDKVERSION/third-
party/lib/*
```

5. Faça download do código-fonte do programa de exemplo (arquivo .zip).

```
wget http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/TryDax.zip
```

Quando o download for concluído, extraia os arquivos de origem.

```
unzip TryDax.zip
```

6. Navegue até o diretório de código Java e compile o código da seguinte maneira.

```
cd TryDax/java/
javac TryDax*.java
```

7. Execute o programa.

```
java TryDax
```

Você deve ver saída semelhante a.

```
Creating a DynamoDB client

Attempting to create table; please wait...
Successfully created table.  Table status: ACTIVE
Writing data to the table...
Writing 10 items for partition key: 1
Writing 10 items for partition key: 2
Writing 10 items for partition key: 3
Writing 10 items for partition key: 4
Writing 10 items for partition key: 5
Writing 10 items for partition key: 6
Writing 10 items for partition key: 7
Writing 10 items for partition key: 8
Writing 10 items for partition key: 9
Writing 10 items for partition key: 10

Running GetItem, Scan, and Query tests...
First iteration of each test will result in cache misses
Next iterations are cache hits

GetItem test - partition key 1 and sort keys 1-10
Total time: 136.681 ms - Avg time: 13.668 ms
Total time: 122.632 ms - Avg time: 12.263 ms
Total time: 167.762 ms - Avg time: 16.776 ms
Total time: 108.130 ms - Avg time: 10.813 ms
Total time: 137.890 ms - Avg time: 13.789 ms
Query test - partition key 5 and sort keys between 2 and 9
```

```
Total time: 13.560 ms - Avg time: 2.712 ms
Total time: 11.339 ms - Avg time: 2.268 ms
Total time: 7.809 ms - Avg time: 1.562 ms
Total time: 10.736 ms - Avg time: 2.147 ms
Total time: 12.122 ms - Avg time: 2.424 ms
Scan test - all items in the table
Total time: 58.952 ms - Avg time: 11.790 ms
Total time: 25.507 ms - Avg time: 5.101 ms
Total time: 37.660 ms - Avg time: 7.532 ms
Total time: 26.781 ms - Avg time: 5.356 ms
Total time: 46.076 ms - Avg time: 9.215 ms

Attempting to delete table; please wait...
Successfully deleted table.
```

Anote as informações de tempo: o número de milissegundos necessários para os testes `GetItem`, `Query` e `Scan`.

8. Na etapa anterior, você executou o programa no endpoint do DynamoDB. Agora, execute o programa novamente, mas, desta vez, o `GetItem`, `Query`, e `Scan` processadas pelo cluster DAX.

Para determinar o endpoint do cluster do DAX, escolha uma das seguintes opções:

- Uso do console do DynamoDB— Escolha o cluster do DAX. O endpoint do cluster é mostrado no console, como no seguinte exemplo.

```
dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

- Usar o AWS CLI— Insira o comando a seguir.

```
aws dax describe-clusters --query "Clusters[*].ClusterDiscoveryEndpoint"
```

O endpoint do cluster é mostrado na saída, como no seguinte exemplo.

```
{
    "Address": "my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com",
    "Port": 8111,
    "URL": "dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com"
}
```

Agora execute o programa novamente, mas, desta vez, especifique o endpoint do cluster como um parâmetro de linha de comando.

```
java TryDax dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

Observe o restante da saída e anote as informações de tempo. Os tempos decorridos para `GetItem`, `Query`, e `Scan` deve ser significativamente mais baixo no DAX do que no DynamoDB.

Para obter mais informações sobre esse programa, consulte as seguintes seções:

- [TryDax.java \(p. 747\)](#)
- [TryDaxHelper.java \(p. 748\)](#)
- [TryDaxTests.java \(p. 751\)](#)

Uso do cliente como dependência do Apache Maven

Siga estas etapas para usar o cliente do SDK do DAX para Java em seu aplicativo como uma dependência.

Como usar o cliente como uma dependência do Maven

1. Faça download do Apache Maven e instale-o. Para obter mais informações, consulte [Download do Apache Maven e Instalação do Apache Maven](#).
2. Adicione a dependência do cliente Maven ao arquivo Project Object Model (POM) do aplicativo. Neste exemplo, substitua `x.x.x.x` pelo número da versão real do cliente (por exemplo, `1.0.200704.0`).

```
<!--Dependency:-->
<dependencies>
    <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>amazon-dax-client</artifactId>
        <version>x.x.x.x</version>
    </dependency>
</dependencies>
```

TryDax.java

O arquivo `TryDax.java` contém o método `main`. Se você executar o programa sem parâmetros de linha de comando, ele criará um cliente do Amazon DynamoDB e usará esse cliente para todas as operações de API. Se você especificar um endpoint de cluster do DynamoDB Accelerator (DAX) na linha de comando, o programa também criará um cliente DAX e o usará para `GetItem`, `Query`, `eScanOperações`.

É possível modificar o programa de várias maneiras:

- Use o cliente DAX do em vez do cliente do DynamoDB. Para obter mais informações, consulte [Java e DAX \(p. 744\)](#).
- Escolha um nome diferente para a tabela de teste.
- Modifique o número de itens gravados, alterando os parâmetros `helper.writeData`. O segundo parâmetro é o número de chaves de partição, e o terceiro parâmetro é o número de chaves de classificação. Por padrão, o programa usa 1—10 para valores de chaves de partição e 1—10 para valores de chaves de classificação, totalizando 100 itens gravados na tabela. Para obter mais informações, consulte [TryDaxHelper.java \(p. 748\)](#).
- Modifique o número de testes de `GetItem`, `Query` e `Scan` e modifique seus parâmetros.
- Assinale como comentários as linhas contendo `helper.createTable` e `helper.deleteTable` (se não quiser criar e excluir a tabela de cada vez que executar o programa).

Note

Para executar o programa, é possível configurar o Maven para usar o cliente do SDK do DAX para Java e o AWS SDK for Java como dependências do. Para obter mais informações, consulte [Uso do cliente como dependência do Apache Maven \(p. 747\)](#).

Como alternativa, você pode fazer download e incluir o cliente Java do DAX e o AWS SDK for Java no seu classpath. Consulte [Java e DAX \(p. 744\)](#) para obter um exemplo de configuração da variável `CLASSPATH`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 */
```

```
*  
* This file is licensed under the Apache License, Version 2.0 (the "License").  
* You may not use this file except in compliance with the License. A copy of  
* the License is located at  
*  
* http://aws.amazon.com/apache2.0/  
*  
* This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
* CONDITIONS OF ANY KIND, either express or implied. See the License for the  
* specific language governing permissions and limitations under the License.  
*/  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
  
public class TryDax {  
  
    public static void main(String[] args) throws Exception {  
  
        TryDaxHelper helper = new TryDaxHelper();  
        TryDaxTests tests = new TryDaxTests();  
  
        DynamoDB ddbClient = helper.getDynamoDBClient();  
        DynamoDB daxClient = null;  
        if (args.length >= 1) {  
            daxClient = helper.getDaxClient(args[0]);  
        }  
  
        String tableName = "TryDaxTable";  
  
        System.out.println("Creating table...");  
        helper.createTable(tableName, ddbClient);  
        System.out.println("Populating table...");  
        helper.writeData(tableName, ddbClient, 10, 10);  
  
        DynamoDB testClient = null;  
        if (daxClient != null) {  
            testClient = daxClient;  
        } else {  
            testClient = ddbClient;  
        }  
  
        System.out.println("Running GetItem, Scan, and Query tests...");  
        System.out.println("First iteration of each test will result in cache misses");  
        System.out.println("Next iterations are cache hits\n");  
  
        // GetItem  
        tests.getItemTest(tableName, testClient, 1, 10, 5);  
  
        // Query  
        tests.queryTest(tableName, testClient, 5, 2, 9, 5);  
  
        // Scan  
        tests.scanTest(tableName, testClient, 5);  
  
        helper.deleteTable(tableName, ddbClient);  
    }  
}
```

TryDaxHelper.java

O arquivo `TryDaxHelper.java` contém métodos utilitários.

`getDynamoDBClient()` e `getDaxClient()` fornecem clientes do Amazon DynamoDB Accelerator (DAX). Para operações de plano de controle (`CreateTable`, `DeleteTable`) e operações de gravação, o

programa usa o cliente do DynamoDB. Se você especificar um endpoint de cluster do DAX, o programa principal criará um cliente DAX para realizar operações de leitura (GetItem,Query,Scan).

O outro TryDaxHelperMétodos (`createTable`,`writeData`,`deleteTable`) são para configurar e destruir a tabela do DynamoDB e seus dados.

É possível modificar o programa de várias maneiras:

- Use configurações de throughput provisionado diferentes para a tabela.
- Modifique o tamanho de cada item gravado (consulte a variável `stringSize` no método `writeData`).
- Modifique o número de testes de `GetItem`, `Query` e `Scan` e seus parâmetros.
- Assinale como comentários as linhas contendo `helper.CreateTable` e `helper.DeleteTable` (se não quiser criar e excluir a tabela de cada vez que executar o programa).

Note

Para executar o programa, é possível configurar o Maven para usar o cliente do SDK do DAX para Java e o AWS SDK for JavaComo dependências do. Para obter mais informações, consulte [Uso do cliente como dependência do Apache Maven \(p. 747\)](#).

Ou, você pode fazer download e incluir o cliente Java do DAX e o AWS SDK for Java no seu classpath. Consulte [Java e DAX \(p. 744\)](#) para obter um exemplo de configuração da variável CLASSPATH.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
import java.util.Arrays;  
  
import com.amazonaws.dax.client.dynamodbv2.AmazonDaxClientBuilder;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.Table;  
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;  
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;  
import com.amazonaws.services.dynamodbv2.model.KeyType;  
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;  
import com.amazonaws.services.dynamodbv2.model.ScalarAttributeType;  
import com.amazonaws.util.EC2MetadataUtils;  
  
public class TryDaxHelper {  
  
    private static final String region = EC2MetadataUtils.getEC2InstanceRegion();  
  
    DynamoDB getDynamoDBClient() {  
        System.out.println("Creating a DynamoDB client");  
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()  
            .withRegion(region)  
    }  
}
```

```
        .build();
    return new DynamoDB(client);
}

DynamoDB getDaxClient(String daxEndpoint) {
    System.out.println("Creating a DAX client with cluster endpoint " + daxEndpoint);
    AmazonDaxClientBuilder daxClientBuilder = AmazonDaxClientBuilder.standard();
    daxClientBuilder.withRegion(region).withEndpointConfiguration(daxEndpoint);
    AmazonDynamoDB client = daxClientBuilder.build();
    return new DynamoDB(client);
}

void createTable(String tableName, DynamoDB client) {
    Table table = client.getTable(tableName);
    try {
        System.out.println("Attempting to create table; please wait...");

        table = client.createTable(tableName,
            Arrays.asList(
                new KeySchemaElement("pk", KeyType.HASH), // Partition key
                new KeySchemaElement("sk", KeyType.RANGE)), // Sort key
            Arrays.asList(
                new AttributeDefinition("pk", ScalarAttributeType.N),
                new AttributeDefinition("sk", ScalarAttributeType.N)),
            new ProvisionedThroughput(10L, 10L));
        table.waitForActive();
        System.out.println("Successfully created table. Table status: " +
            table.getDescription().getTableStatus());
    } catch (Exception e) {
        System.err.println("Unable to create table: ");
        e.printStackTrace();
    }
}

void writeData(String tableName, DynamoDB client, int pkmax, int skmax) {
    Table table = client.getTable(tableName);
    System.out.println("Writing data to the table...");

    int stringSize = 1000;
    StringBuilder sb = new StringBuilder(stringSize);
    for (int i = 0; i < stringSize; i++) {
        sb.append('X');
    }
    String someData = sb.toString();

    try {
        for (Integer ipk = 1; ipk <= pkmax; ipk++) {
            System.out.println("Writing " + skmax + " items for partition key: " +
ipk));
            for (Integer isk = 1; isk <= skmax; isk++) {
                table.putItem(new Item()
                    .withPrimaryKey("pk", ipk, "sk", isk)
                    .withString("someData", someData));
            }
        }
    } catch (Exception e) {
        System.err.println("Unable to write item:");
        e.printStackTrace();
    }
}

void deleteTable(String tableName, DynamoDB client) {
    Table table = client.getTable(tableName);
    try {
        System.out.println("\nAttempting to delete table; please wait...");
    }
```

```
        table.delete();
        table.waitForDelete();
        System.out.println("Successfully deleted table.");

    } catch (Exception e) {
        System.err.println("Unable to delete table: ");
        e.printStackTrace();
    }
}
```

TryDaxTests.java

O `TryDaxTests.java` contém métodos que realizam operações em uma tabela de teste no Amazon DynamoDB. Esses métodos não consideram como os dados serão acessados (usando o cliente do DynamoDB ou o cliente do DAX) e, portanto, não é necessário modificar a lógica do aplicativo.

É possível modificar o programa de várias maneiras:

- Modifique o método `queryTest` para que ele use um `KeyConditionExpression` diferente.
- Adicione um `ScanFilter` ao método `scanTest`, para que apenas alguns dos itens sejam retornados para você.

Note

Para executar o programa, é possível configurar o Maven para usar o cliente do SDK do DAX para Java e o AWS SDK for Java como dependências do. Para obter mais informações, consulte [Uso do cliente como dependência do Apache Maven \(p. 747\)](#).

Ou, você pode fazer download e incluir o cliente Java do DAX e o AWS SDK for Java no seu classpath. Consulte [Java e DAX \(p. 744\)](#) para obter um exemplo de configuração da variável CLASSPATH.

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
import java.util.HashMap;
import java.util.Iterator;

import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Item;
import com.amazonaws.services.dynamodbv2.document.ItemCollection;
import com.amazonaws.services.dynamodbv2.document.QueryOutcome;
import com.amazonaws.services.dynamodbv2.document.ScanOutcome;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.document.spec.QuerySpec;

public class TryDaxTests {
```

```
void getItemTest(String tableName, DynamoDB client, int pk, int sk, int iterations) {
    long startTime, endTime;
    System.out.println("GetItem test - partition key " + pk + " and sort keys 1-" +
sk);
    Table table = client.getTable(tableName);

    for (int i = 0; i < iterations; i++) {
        startTime = System.nanoTime();
        try {
            for (Integer ipk = 1; ipk <= pk; ipk++) {
                for (Integer isk = 1; isk <= sk; isk++) {
                    table.getItem("pk", ipk, "sk", isk);
                }
            }
        } catch (Exception e) {
            System.err.println("Unable to get item:");
            e.printStackTrace();
        }
        endTime = System.nanoTime();
        printTime(startTime, endTime, pk * sk);
    }
}

void queryTest(String tableName, DynamoDB client, int pk, int sk1, int sk2, int
iterations) {
    long startTime, endTime;
    System.out.println("Query test - partition key " + pk + " and sort keys between " +
sk1 + " and " + sk2);
    Table table = client.getTable(tableName);

    HashMap<String, Object> valueMap = new HashMap<String, Object>();
    valueMap.put(":pkval", pk);
    valueMap.put(":skval1", sk1);
    valueMap.put(":skval2", sk2);

    QuerySpec spec = new QuerySpec()
        .withKeyConditionExpression("pk = :pkval and sk between :skval1
and :skval2")
        .withValueMap(valueMap);

    for (int i = 0; i < iterations; i++) {
        startTime = System.nanoTime();
        ItemCollection<QueryOutcome> items = table.query(spec);

        try {
            Iterator<Item> iter = items.iterator();
            while (iter.hasNext()) {
                iter.next();
            }
        } catch (Exception e) {
            System.err.println("Unable to query table:");
            e.printStackTrace();
        }
        endTime = System.nanoTime();
        printTime(startTime, endTime, iterations);
    }
}

void scanTest(String tableName, DynamoDB client, int iterations) {
    long startTime, endTime;
    System.out.println("Scan test - all items in the table");
    Table table = client.getTable(tableName);

    for (int i = 0; i < iterations; i++) {
        startTime = System.nanoTime();
        ItemCollection<ScanOutcome> items = table.scan();
```

```
try {

    Iterator<Item> iter = items.iterator();
    while (iter.hasNext()) {
        iter.next();
    }
} catch (Exception e) {
    System.err.println("Unable to scan table:");
    e.printStackTrace();
}
endTime = System.nanoTime();
printTime(startTime, endTime, iterations);
}

public void printTime(long startTime, long endTime, int iterations) {
    System.out.format("\tTotal time: %.3f ms - ", (endTime - startTime) / (1000000.0));
    System.out.format("Avg time: %.3f ms\n", (endTime - startTime) / (iterations * 1000000.0));
}
```

.NET e DAX

Siga estas etapas para executar o exemplo .NET na instância do Amazon EC2.

Note

Este tutorial usa o .NET Core SDK. Mostra como você pode executar um programa na sua Amazon VPC padrão para acessar seu cluster do Amazon DynamoDB Accelerator (DAX). Se você preferir, pode usar o AWS Toolkit for Visual Studio para escrever um aplicativo .NET e implantá-lo em sua VPC.

Para obter mais informações, consulte [Criar e implantar aplicativos do Elastic Beanstalk no .NET usando AWSToolkit for Visual Studio](#) no AWS Elastic Beanstalk Guia do desenvolvedor.

Como executar o exemplo do .NET para o DAX

1. Acesse a [página Downloads da Microsoft](#) e faça download do SDK do .NET Core para Linux mais recente. O arquivo obtido por download é o `dotnet-sdk-N.N.N-linux-x64.tar.gz`.
2. Extraia os arquivos .NET Core.

```
mkdir dotnet
tar zxvf dotnet-sdk-N.N.N-linux-x64.tar.gz -C dotnet
```

Substitua `N.N.N` pelo número de versão real do SDK do .NET Core (por exemplo: `2.1.4`).

3. Verifique a instalação.

```
alias dotnet=$HOME/dotnet/dotnet
dotnet --version
```

Isso deve imprimir o número da versão do .NET Core SDK.

Note

Em vez de obter o número da versão, é possível que você receba o seguinte erro:
Erro: libunwind.so.8: não é possível abrir o arquivo de objeto compartilhado: Nenhum arquivo ou diretório
Para resolver o erro, instale o pacote `libunwind`.

```
sudo yum install -y libunwind
```

Depois disso, você deve conseguir executar o comando `dotnet --version` sem nenhum erro.

- Crie um novo projeto .NET.

```
dotnet new console -o myApp
```

Ele levará alguns minutos executando uma configuração isolada. Ao concluir, execute o projeto de exemplo.

```
dotnet run --project myApp
```

Você deverá receber a seguinte mensagem: `Hello World!`

- O arquivo `myApp/myApp.csproj` contém os metadados do projeto. Para usar o cliente DAX no aplicativo, modifique o arquivo para que seja semelhante ao seguinte.

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp2.0</TargetFramework>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="AWSSDK.DAX.Client" Version="*" />
  </ItemGroup>
</Project>
```

- Faça download do código-fonte do programa de exemplo (arquivo `.zip`).

```
wget http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/TryDax.zip
```

Quando o download for concluído, extraia os arquivos de origem.

```
unzip TryDax.zip
```

- Agora, execute os programas de exemplo, um por vez. Para cada programa, você copiará seu conteúdo em `myApp/Program.cs` e, seguida, executará o projeto `MyApp`.

Execute os seguintes programas .NET. O primeiro programa cria uma tabela do DynamoDB denominada `TryDaxTable`. O segundo programa grava dados na tabela.

```
cp 01-CREATETable.cs myApp/Program.cs
dotnet run --project myApp

cp 02-Write-Data.cs myApp/Program.cs
dotnet run --project myApp
```

- Depois, execute alguns programas para executar `GetItem`, `Query`, e `Scan` operações no cluster do DAX. Para determinar o endpoint do cluster do DAX, escolha uma das seguintes opções:

- Uso do console do DynamoDB— Escolha o cluster do DAX. O endpoint do cluster é mostrado no console, como no seguinte exemplo.

```
dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

- Usar o AWS CLI— Insira o comando a seguir.

```
aws dax describe-clusters --query "Clusters[*].ClusterDiscoveryEndpoint"
```

O endpoint do cluster é mostrado na saída, como no seguinte exemplo.

```
{  
    "Address": "my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com",  
    "Port": 8111,  
    "URL": "dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com"  
}
```

Agora execute os seguintes programas, especificando o endpoint do cluster como um parâmetro de linha de comando. (Substitua o endpoint de exemplo pelo endpoint do seu cluster do DAX real.)

```
cp 03-GetItem-Test.cs myApp/Program.cs  
dotnet run --project myApp dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com  
  
cp 04-Query-Test.cs myApp/Program.cs  
dotnet run --project myApp dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com  
  
cp 05-Scan-Test.cs myApp/Program.cs  
dotnet run --project myApp dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

Anote as informações de tempo: o número de milissegundos necessários para os testes GetItem, Query e Scan.

9. Execute o seguinte programa .NET para excluir a TryDaxTable.

```
cp 06>DeleteTable.cs myApp/Program.cs  
dotnet run --project myApp
```

Para obter mais informações sobre esses programas, consulte as seguintes seções:

- [01 CreateTable.cs \(p. 755\)](#)
- [02 Write Data.cs \(p. 756\)](#)
- [03 GetItem-Test.cs \(p. 757\)](#)
- [04 Query-Test.cs \(p. 759\)](#)
- [05 Scan-Test.cs \(p. 760\)](#)
- [06 DeleteTable.cs \(p. 761\)](#)

01-CreateTable.cs

O programa 01-CreateTable.cs cria uma tabela (TryDaxTable). Os demais programas .NET nesta seção dependem dessa tabela.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 */
```

```
* http://aws.amazon.com/apache2.0/
*
* This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
* CONDITIONS OF ANY KIND, either express or implied. See the License for the
* specific language governing permissions and limitations under the License.
*/
using Amazon.DynamoDBv2.Model;
using System.Collections.Generic;
using System;
using Amazon.DynamoDBv2;

namespace ClientTest
{
    class Program
    {
        static void Main(string[] args)
        {

            AmazonDynamoDBClient client = new AmazonDynamoDBClient();

            var tableName = "TryDaxTable";

            var request = new CreateTableRequest()
            {
                TableName = tableName,
                KeySchema = new List<KeySchemaElement>()
                {
                    new KeySchemaElement{ AttributeName = "pk",KeyType = "HASH" },
                    new KeySchemaElement{ AttributeName = "sk",KeyType = "RANGE" }
                },
                AttributeDefinitions = new List<AttributeDefinition>() {
                    new AttributeDefinition{ AttributeName = "pk",AttributeType = "N" },
                    new AttributeDefinition{ AttributeName = "sk",AttributeType = "N" }
                },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 10,
                    WriteCapacityUnits = 10
                }
            };

            var response = client.CreateTableAsync(request).Result;

            Console.WriteLine("Hit <enter> to continue...");
            Console.ReadLine();
        }
    }
}
```

02-Write-Data.cs

O programa 02-Write-Data.cs grava dados de teste em TryDaxTable.

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
```

```
* CONDITIONS OF ANY KIND, either express or implied. See the License for the
* specific language governing permissions and limitations under the License.
*/
using Amazon.DynamoDBv2.Model;
using System.Collections.Generic;
using System;
using Amazon.DynamoDBv2;

namespace ClientTest
{
    class Program
    {
        static void Main(string[] args)
        {

            AmazonDynamoDBClient client = new AmazonDynamoDBClient();

            var tableName = "TryDaxTable";

            string someData = new String('X', 1000);
            var pkmax = 10;
            var skmax = 10;

            for (var ipk = 1; ipk <= pkmax; ipk++)
            {
                Console.WriteLine("Writing " + skmax + " items for partition key: " + ipk);
                for (var isk = 1; isk <= skmax; isk++)
                {
                    var request = new PutItemRequest()
                    {
                        TableName = tableName,
                        Item = new Dictionary<string, AttributeValue>()
                        {
                            { "pk", new AttributeValue{N = ipk.ToString()} },
                            { "sk", new AttributeValue{N = isk.ToString()} },
                            { "someData", new AttributeValue{S = someData} }
                        };
                    };

                    var response = client.PutItemAsync(request).Result;

                }
            }

            Console.WriteLine("Hit <enter> to continue...");
            Console.ReadLine();
        }
    }
}
```

03-GetItem-Test.cs

O programa 03-GetItem-Test.cs executa operações GetItem em TryDaxTable.

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
```

```
* This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
* CONDITIONS OF ANY KIND, either express or implied. See the License for the
* specific language governing permissions and limitations under the License.
*/
using Amazon.Runtime;
using Amazon.DAX;
using Amazon.DynamoDBv2.Model;
using System.Collections.Generic;
using System;
using Amazon.DynamoDBv2;
using Amazon;

namespace ClientTest
{
    class Program
    {
        static void Main(string[] args)
        {

            String hostName = args[0].Split(':')[0];
            int port = Int32.Parse(args[0].Split(':')[1]);
            Console.WriteLine("Using DAX client - hostname=" + hostName + ", port=" +
port);

            var clientConfig = new DaxClientConfig(hostName, port)
            {
                AwsCredentials = FallbackCredentialsFactory.GetCredentials()

            };
            var client = new ClusterDaxClient(clientConfig);

            var tableName = "TryDaxTable";

            var pk = 1;
            var sk = 10;
            var iterations = 5;

            var startTime = DateTime.Now;

            for (var i = 0; i < iterations; i++)
            {

                for (var ipk = 1; ipk <= pk; ipk++)
                {
                    for (var isk = 1; isk <= sk; isk++)
                    {
                        var request = new GetItemRequest()
                        {
                            TableName = tableName,
                            Key = new Dictionary<string, AttributeValue>() {
                                {"pk", new AttributeValue {N = ipk.ToString()} },
                                {"sk", new AttributeValue {N = isk.ToString()} }
                            }
                        };
                        var response = client.GetItemAsync(request).Result;
                        Console.WriteLine("GetItem succeeded for pk: " + ipk + ", sk: " +
isk);
                    }
                }
            }

            var endTime = DateTime.Now;
            TimeSpan timeSpan = endTime - startTime;
            Console.WriteLine("Total time: " + (int)timeSpan.TotalMilliseconds + " milliseconds");
        }
    }
}
```

```
        Console.WriteLine("Hit <enter> to continue...");
        Console.ReadLine();
    }
}
```

04-Query-Test.cs

O programa 04-Query-Test.cs executa operações Query em TryDaxTable.

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
using Amazon.Runtime;
using Amazon.DAX;
using Amazon.DynamoDBv2.Model;
using System.Collections.Generic;
using System;
using Amazon.DynamoDBv2;
using Amazon;

namespace ClientTest
{
    class Program
    {
        static void Main(string[] args)
        {

            String hostName = args[0].Split(':')[0];
            int port = Int32.Parse(args[0].Split(':')[1]);
            Console.WriteLine("Using DAX client - hostname=" + hostName + ", port=" +
port);

            var clientConfig = new DaxClientConfig(hostName, port)
            {
                AwsCredentials = FallbackCredentialsFactory.GetCredentials()

            };
            var client = new ClusterDaxClient(clientConfig);

            var tableName = "TryDaxTable";

            var pk = 5;
            var sk1 = 2;
            var sk2 = 9;
            var iterations = 5;

            var startTime = DateTime.Now;
```

```
        for (var i = 0; i < iterations; i++)
    {
        var request = new QueryRequest()
        {
            TableName = tableName,
            KeyConditionExpression = "pk = :pkval and sk between :skval1
and :skval2",
            ExpressionAttributeValues = new Dictionary<string, AttributeValue>() {
                {":pkval", new AttributeValue {N = pk.ToString()} },
                {":skval1", new AttributeValue {N = sk1.ToString()} },
                {":skval2", new AttributeValue {N = sk2.ToString()} }
            }
        };
        var response = client.QueryAsync(request).Result;
        Console.WriteLine(i + ": Query succeeded");

    }

    var endTime = DateTime.Now;
    TimeSpan timeSpan = endTime - startTime;
    Console.WriteLine("Total time: " + (int)timeSpan.TotalMilliseconds + " milliseconds");

    Console.WriteLine("Hit <enter> to continue...");
    Console.ReadLine();
}
}
}
```

05-Scan-Test.cs

O programa 05-Scan-Test.cs executa operações Scan em TryDaxTable.

```
/**
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
using Amazon.Runtime;
using Amazon.DAX;
using Amazon.DynamoDBv2.Model;
using System.Collections.Generic;
using System;
using Amazon.DynamoDBv2;
using Amazon;

namespace ClientTest
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
String hostName = args[0].Split(':')[0];
int port = Int32.Parse(args[0].Split(':')[1]);
Console.WriteLine("Using DAX client - hostname=" + hostName + ", port=" +
port);

var clientConfig = new DaxClientConfig(hostName, port)
{
    AwsCredentials = FallbackCredentialsFactory.GetCredentials(
};

var client = new ClusterDaxClient(clientConfig);

var tableName = "TryDaxTable";

var iterations = 5;

var startTime = DateTime.Now;

for (var i = 0; i < iterations; i++)
{
    var request = new ScanRequest()
    {
        TableName = tableName
    };
    var response = client.ScanAsync(request).Result;
    Console.WriteLine(i + ": Scan succeeded");
}

var endTime = DateTime.Now;
TimeSpan timeSpan = endTime - startTime;
Console.WriteLine("Total time: " + (int)timeSpan.TotalMilliseconds + " milliseconds");

Console.WriteLine("Hit <enter> to continue...");
Console.ReadLine();
}
}
```

06-DeleteTable.cs

O programa 06-DeleteTable.cs exclui TryDaxTable. Execute esse programa depois de concluir o teste.

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
*/
using Amazon.DynamoDBv2.Model;
using System;
using Amazon.DynamoDBv2;
```

```
namespace ClientTest
{
    class Program
    {
        static void Main(string[] args)
        {

            AmazonDynamoDBClient client = new AmazonDynamoDBClient();

            var tableName = "TryDaxTable";

            var request = new DeleteTableRequest()
            {
                TableName = tableName
            };

            var response = client.DeleteTableAsync(request).Result;

            Console.WriteLine("Hit <enter> to continue...");
            Console.ReadLine();
        }
    }
}
```

Node.js e DAX

Siga estas etapas para executar o aplicativo de exemplo do Node.js na instância do Amazon EC2.

Como executar o exemplo do Node.js para DAX

1. Configure o Node.js na instância do Amazon EC2 da seguinte maneira:

- a. Instale o gerenciador de versão de nó (nvm).

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
```

- b. Use o nvm para instalar o Node.js.

```
nvm install 12.16.3
```

- c. Verifique se o Node.js está instalado e funcionando corretamente.

```
node -e "console.log('Running Node.js ' + process.version)"
```

Isso deve exibir a seguinte mensagem.

```
Running Node.js v12.16.3
```

2. Instale o cliente Node.js do usando o gerenciador de pacotes de nó (npm).

```
npm install amazon-dax-client
```

3. Faça download do código-fonte do programa de exemplo (arquivo .zip).

```
wget http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/TryDax.zip
```

Quando o download for concluído, extraia os arquivos de origem.

```
unzip TryDax.zip
```

4. Execute os seguintes programas Node.js. O primeiro programa cria uma tabela do Amazon DynamoDB denominada `TryDaxTable`. O segundo programa grava dados na tabela.

```
node 01-create-table.js  
node 02-write-data.js
```

5. Execute os seguintes programas Node.js.

```
node 03-getitem-test.js  
node 04-query-test.js  
node 05-scan-test.js
```

Anote as informações de tempo: o número de milissegundos necessários para os testes `GetItem`, `Query` e `Scan`.

6. Na etapa anterior, você executou os programas no endpoint do DynamoDB. Execute os programas novamente, mas, desta vez, o `GetItem`, `Query` e `Scan` são processadas pelo cluster DAX.

Para determinar o endpoint do cluster do DAX, escolha uma das seguintes opções.

- Uso do console do DynamoDB—Escolha o cluster do DAX. O endpoint do cluster é mostrado no console, como no seguinte exemplo.

```
dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

- Usar o AWS CLI—Insira o comando a seguir.

```
aws dax describe-clusters --query "Clusters[*].ClusterDiscoveryEndpoint"
```

O endpoint do cluster é mostrado na saída, como no seguinte exemplo.

```
{  
    "Address": "my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com",  
    "Port": 8111,  
    "URL": "dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com"  
}
```

Agora, execute os programas novamente, mas, desta vez, especifique o endpoint do cluster como um parâmetro de linha de comando.

```
node 03-getitem-test.js dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com  
node 04-query-test.js dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com  
node 05-scan-test.js dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

Observe o restante da saída e anote as informações de tempo. Os tempos decorridos para `GetItem`, `Query`, e `Scan` deve ser significativamente mais baixo no DAX do que no DynamoDB.

7. Execute o seguinte programa Node.js para excluir a `TryDaxTable`.

```
node 06-delete-table
```

Para obter mais informações sobre esses programas, consulte as seguintes seções:

- [01-create-table.js \(p. 764\)](#)
- [02-write-data.js \(p. 765\)](#)
- [03-getitem-test.js \(p. 766\)](#)
- [04-query-test.js \(p. 767\)](#)
- [05-scan-test.js \(p. 768\)](#)
- [06-delete-table.js \(p. 769\)](#)

[01-create-table.js](#)

O programa `01-create-table.js` cria uma tabela (`TryDaxTable`). Os programas Node.js restantes nesta seção dependem desta tabela.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
const AmazonDaxClient = require('amazon-dax-client');  
var AWS = require("aws-sdk");  
  
var region = "us-west-2";  
  
AWS.config.update({  
    region: region  
});  
  
var dynamodb = new AWS.DynamoDB() //low-level client  
  
var tableName = "TryDaxTable";  
  
var params = {  
    TableName : tableName,  
    KeySchema: [  
        { AttributeName: "pk", KeyType: "HASH"},   //Partition key  
        { AttributeName: "sk", KeyType: "RANGE" }  //Sort key  
    ],  
    AttributeDefinitions: [  
        { AttributeName: "pk", AttributeType: "N" },  
        { AttributeName: "sk", AttributeType: "N" }  
    ],  
    ProvisionedThroughput: {  
        ReadCapacityUnits: 10,  
        WriteCapacityUnits: 10  
    }  
};  
  
dynamodb.createTable(params, function(err, data) {  
    if (err) {  
        console.error("Unable to create table. Error JSON:", JSON.stringify(err, null, 2));  
    } else {  
        console.log("Created table. Table description JSON:", JSON.stringify(data, null, 2));  
    }  
});
```

```
}); }
```

02-write-data.js

O programa `02-write-data.js` grava dados de teste em `TryDaxTable`.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
const AmazonDaxClient = require('amazon-dax-client');  
var AWS = require("aws-sdk");  
  
var region = "us-west-2";  
  
AWS.config.update({  
    region: region  
});  
  
var ddbClient = new AWS.DynamoDB.DocumentClient()  
  
var tableName = "TryDaxTable";  
  
var someData = "X".repeat(1000);  
var pkmax = 10;  
var skmax = 10;  
  
for (var ipk = 1; ipk <= pkmax; ipk++) {  
  
    for (var isk = 1; isk <= skmax; isk++) {  
        var params = {  
            TableName: tableName,  
            Item: {  
                "pk": ipk,  
                "sk": isk,  
                "someData": someData  
            }  
        };  
  
        //  
        //put item  
  
        ddbClient.put(params, function(err, data) {  
            if (err) {  
                console.error("Unable to write data: ", JSON.stringify(err, null, 2));  
            } else {  
                console.log("PutItem succeeded");  
            }  
        });  
    }  
}
```

```
}
```

03-getitem-test.js

O programa 03-getitem-test.js executa operações GetItem em TryDaxTable.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
const AmazonDaxClient = require('amazon-dax-client');  
var AWS = require("aws-sdk");  
  
var region = "us-west-2";  
  
AWS.config.update({  
    region: region  
});  
  
var ddbClient = new AWS.DynamoDB.DocumentClient()  
var daxClient = null;  
  
if (process.argv.length > 2) {  
    var dax = new AmazonDaxClient({endpoints: [process.argv[2]], region: region})  
    daxClient = new AWS.DynamoDB.DocumentClient({service: dax});  
}  
  
var client = daxClient != null ? daxClient : ddbClient;  
var tableName = "TryDaxTable";  
  
var pk = 1;  
var sk = 10;  
var iterations = 5;  
  
for (var i = 0; i < iterations; i++) {  
  
    var startTime = new Date().getTime();  
  
    for (var ipk = 1; ipk <= pk; ipk++) {  
        for (var isk = 1; isk <= sk; isk++) {  
  
            var params = {  
                TableName: tableName,  
                Key:{  
                    "pk": ipk,  
                    "sk": isk  
                }  
            };  
  
            client.get(params, function(err, data) {  
                if (err) {  
                    console.error("Unable to read item. Error JSON:", JSON.stringify(err,  
null, 2));  
                }  
            });  
        }  
    }  
}
```

```
        } else {
            // GetItem succeeded
        });
    }
}

var endTime = new Date().getTime();
console.log("\tTotal time: ", (endTime - startTime) , "ms - Avg time: ", (endTime - startTime) / iterations, "ms");

}
```

04-query-test.js

O programa 04-query-test.js executa operações Query em TryDaxTable.

```
/***
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
const AmazonDaxClient = require('amazon-dax-client');
var AWS = require("aws-sdk");

var region = "us-west-2";

AWS.config.update({
    region: region
});

var ddbClient = new AWS.DynamoDB.DocumentClient()
var daxClient = null;

if (process.argv.length > 2) {
    var dax = new AmazonDaxClient({endpoints: [process.argv[2]], region: region})
    daxClient = new AWS.DynamoDB.DocumentClient({service: dax });
}

var client = daxClient != null ? daxClient : ddbClient;
var tableName = "TryDaxTable";

var pk = 5;
var sk1 = 2;
var sk2 = 9;
var iterations = 5;

var params = {
    TableName: tableName,
    KeyConditionExpression: "pk = :pkval and sk between :skval1 and :skval2",
    ExpressionAttributeValues: {
        ":pkval":pk,
        ":skval1":sk1,
        ":skval2":sk2
    }
}
```

```
};

for (var i = 0; i < iterations; i++) {
    var startTime = new Date().getTime();

    client.query(params, function(err, data) {
        if (err) {
            console.error("Unable to read item. Error JSON:", JSON.stringify(err, null,
2));
        } else {
            // Query succeeded
        }
    });
}

var endTime = new Date().getTime();
console.log("\tTotal time: ", (endTime - startTime) , "ms - Avg time: ", (endTime -
startTime) / iterations, "ms");

}
```

05-scan-test.js

O programa 05-scan-test.js executa operações Scan em TryDaxTable.

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */
const AmazonDaxClient = require('amazon-dax-client');
var AWS = require("aws-sdk");

var region = "us-west-2";

AWS.config.update({
    region: region
});

var ddbClient = new AWS.DynamoDB.DocumentClient()
var daxClient = null;

if (process.argv.length > 2) {
    var dax = new AmazonDaxClient({endpoints: [process.argv[2]], region: region})
    daxClient = new AWS.DynamoDB.DocumentClient({service: dax });
}

var client = daxClient != null ? daxClient : ddbClient;
var tableName = "TryDaxTable";

var iterations = 5;

var params = {
    TableName: tableName
};
var startTime = new Date().getTime();
```

```
for (var i = 0; i < iterations; i++) {  
  
    client.scan(params, function(err, data) {  
        if (err) {  
            console.error("Unable to read item. Error JSON:", JSON.stringify(err, null, 2));  
        } else {  
            // Scan succeeded  
        }  
    });  
  
}  
  
var endTime = new Date().getTime();  
console.log("\tTotal time: ", (endTime - startTime) , "ms - Avg time: ", (endTime -  
startTime) / iterations, "ms");
```

06-delete-table.js

O programa 06-delete-table.js exclui TryDaxTable. Execute esse programa depois de concluir o teste.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
const AmazonDaxClient = require('amazon-dax-client');  
var AWS = require("aws-sdk");  
  
var region = "us-west-2";  
  
AWS.config.update({  
    region: region  
});  
  
var dynamodb = new AWS.DynamoDB(); //low-level client  
  
var tableName = "TryDaxTable";  
  
var params = {  
    TableName : tableName  
};  
  
dynamodb.deleteTable(params, function(err, data) {  
    if (err) {  
        console.error("Unable to delete table. Error JSON:", JSON.stringify(err, null, 2));  
    } else {  
        console.log("Deleted table. Table description JSON:", JSON.stringify(data, null, 2));  
    }  
});
```

Python e DAX

Siga este procedimento para executar o aplicativo de exemplo do Python na instância do Amazon EC2.

Como executar o Python de exemplo para o DAX

1. Instale o cliente Python do DAX do usando o pip utilitário.

```
pip install amazon-dax-client
```

2. Faça download do código-fonte do programa de exemplo (arquivo .zip).

```
wget http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/TryDax.zip
```

Quando o download for concluído, extraia os arquivos de origem.

```
unzip TryDax.zip
```

3. Execute os seguintes programas do Python. O primeiro programa cria uma tabela do Amazon DynamoDB denominada TryDaxTable. O segundo programa grava dados na tabela.

```
python 01-create-table.py  
python 02-write-data.py
```

4. Execute os seguintes programas do Python.

```
python 03-getitem-test.py  
python 04-query-test.py  
python 05-scan-test.py
```

Anote as informações de tempo: o número de milissegundos necessários para os testes GetItem, Query e Scan.

5. Na etapa anterior, você executou os programas no endpoint do DynamoDB. Agora, execute os programas novamente, mas, desta vez, o GetItem, Query, e Scan são processadas pelo cluster DAX.

Para determinar o endpoint do cluster do DAX, escolha uma das seguintes opções:

- Uso do console do DynamoDB— Escolha o cluster do DAX. O endpoint do cluster é mostrado no console, como no seguinte exemplo.

```
dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

- Usar o AWS CLI— Insira o comando a seguir.

```
aws dax describe-clusters --query "Clusters[*].ClusterDiscoveryEndpoint"
```

O endpoint do cluster é mostrado na saída, como neste exemplo.

```
{  
    "Address": "my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com",  
    "Port": 8111,  
    "URL": "dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com"  
}
```

Execute os programas novamente, mas, desta vez, especifique o endpoint do cluster como um parâmetro de linha de comando.

```
python 03-getitem-test.py dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
python 04-query-test.py dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
python 05-scan-test.py dax://my-cluster.16fzcv.dax-clusters.us-east-1.amazonaws.com
```

Observe o restante da saída e anote as informações de tempo. Os tempos decorridos para `GetItem`, `Query`, e `Scan` deve ser significativamente mais baixo no DAX do que no DynamoDB.

6. Execute o seguinte programa do Python para excluir a `TryDaxTable`.

```
python 06-delete-table.py
```

Para obter mais informações sobre esses programas, consulte as seguintes seções:

- [01-create-table.py \(p. 771\)](#)
- [02-write-data.py \(p. 772\)](#)
- [03-getitem-test.py \(p. 772\)](#)
- [04-query-test.py \(p. 773\)](#)
- [05-scan-test.py \(p. 774\)](#)
- [06-delete-table.py \(p. 775\)](#)

01-create-table.py

O programa `01-create-table.py` cria uma tabela (`TryDaxTable`). Os demais programas Python nesta seção dependem dessa tabela.

```
import boto3

def create_dax_table(dyn_resource=None):
    """
    Creates a DynamoDB table.

    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The newly created table.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table_name = 'TryDaxTable'
    params = {
        'TableName': table_name,
        'KeySchema': [
            {'AttributeName': 'partition_key', 'KeyType': 'HASH'},
            {'AttributeName': 'sort_key', 'KeyType': 'RANGE'}
        ],
        'AttributeDefinitions': [
            {'AttributeName': 'partition_key', 'AttributeType': 'N'},
            {'AttributeName': 'sort_key', 'AttributeType': 'N'}
        ],
        'ProvisionedThroughput': {
            'ReadCapacityUnits': 10,
            'WriteCapacityUnits': 10
        }
    }
```

```
    }
    table = dyn_resource.create_table(**params)
    print(f"Creating {table_name}...")
    table.wait_until_exists()
    return table

if __name__ == '__main__':
    dax_table = create_dax_table()
    print(f"Created table.")
```

02-write-data.py

O programa 02-write-data.py grava dados de teste em TryDaxTable.

```
import boto3

def write_data_to_dax_table(key_count, item_size, dyn_resource=None):
    """
    Writes test data to the demonstration table.

    :param key_count: The number of partition and sort keys to use to populate the
                      table. The total number of items is key_count * key_count.
    :param item_size: The size of non-key data for each test item.
    :param dyn_resource: Either a Boto3 or DAX resource.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table = dyn_resource.Table('TryDaxTable')
    some_data = 'X' * item_size

    for partition_key in range(1, key_count + 1):
        for sort_key in range(1, key_count + 1):
            table.put_item(Item={
                'partition_key': partition_key,
                'sort_key': sort_key,
                'some_data': some_data
            })
            print(f"Put item ({partition_key}, {sort_key}) succeeded.")

    if __name__ == '__main__':
        write_key_count = 10
        write_item_size = 1000
        print(f"Writing {write_key_count}*{write_key_count} items to the table. "
              f"Each item is {write_item_size} characters.")
        write_data_to_dax_table(write_key_count, write_item_size)
```

03-getitem-test.py

O programa 03-getitem-test.py executa operações GetItem em TryDaxTable.

```
import argparse
import sys
import time
import amazonadax
import boto3

def get_item_test(key_count, iterations, dyn_resource=None):
    """
    Gets items from the table a specified number of times. The time before the
    
```

```
first iteration and the time after the last iteration are both captured
and reported.

:param key_count: The number of items to get from the table in each iteration.
:param iterations: The number of iterations to run.
:param dyn_resource: Either a Boto3 or DAX resource.
:return: The start and end times of the test.
"""

if dyn_resource is None:
    dyn_resource = boto3.resource('dynamodb')

table = dyn_resource.Table('TryDaxTable')
start = time.perf_counter()
for _ in range(iterations):
    for partition_key in range(1, key_count + 1):
        for sort_key in range(1, key_count + 1):
            table.get_item(Key={
                'partition_key': partition_key,
                'sort_key': sort_key
            })
            print('.', end='')
            sys.stdout.flush()
print()
end = time.perf_counter()
return start, end

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument(
        'endpoint_url', nargs='?',
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not used.")
    args = parser.parse_args()

    test_key_count = 10
    test_iterations = 50
    if args.endpoint_url:
        print(f"Getting each item from the table {test_iterations} times, "
              f"using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after completion.
        with amazonadax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as dax:
            test_start, test_end = get_item_test(
                test_key_count, test_iterations, dyn_resource=dax)
    else:
        print(f"Getting each item from the table {test_iterations} times, "
              f"using the Boto3 client.")
        test_start, test_end = get_item_test(
            test_key_count, test_iterations)
    print(f"Total time: {test_end - test_start:.4f} sec. Average time: "
          f"{{(test_end - test_start)/ test_iterations}}")
```

04-query-test.py

O programa 04-query-test.py executa operações Query em TryDaxTable.

```
import argparse
import time
import sys
import amazonadax
import boto3
from boto3.dynamodb.conditions import Key

def query_test(partition_key, sort_keys, iterations, dyn_resource=None):
    """
```

```
Queries the table a specified number of times. The time before the
first iteration and the time after the last iteration are both captured
and reported.

:param partition_key: The partition key value to use in the query. The query
                     returns items that have partition keys equal to this value.
:param sort_keys: The range of sort key values for the query. The query returns
                  items that have sort key values between these two values.
:param iterations: The number of iterations to run.
:param dyn_resource: Either a Boto3 or DAX resource.
:return: The start and end times of the test.
"""

if dyn_resource is None:
    dyn_resource = boto3.resource('dynamodb')

table = dyn_resource.Table('TryDaxTable')
key_condition_expression = \
    Key('partition_key').eq(partition_key) & \
    Key('sort_key').between(*sort_keys)

start = time.perf_counter()
for _ in range(iterations):
    table.query(KeyConditionExpression=key_condition_expression)
    print('.', end='')
    sys.stdout.flush()
print()
end = time.perf_counter()
return start, end

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument(
        'endpoint_url', nargs='?',
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not used.")
    args = parser.parse_args()

    test_partition_key = 5
    test_sort_keys = (2, 9)
    test_iterations = 100
    if args.endpoint_url:
        print(f"Querying the table {test_iterations} times, using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after completion.
        with amazonadax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as dax:
            test_start, test_end = query_test(
                test_partition_key, test_sort_keys, test_iterations, dyn_resource=dax)
    else:
        print(f"Querying the table {test_iterations} times, using the Boto3 client.")
        test_start, test_end = query_test(
            test_partition_key, test_sort_keys, test_iterations)

    print(f"Total time: {test_end - test_start:.4f} sec. Average time: "
          f"{{(test_end - test_start)/test_iterations}}")
```

05-scan-test.py

O programa 05-scan-test.py executa operações Scan em TryDaxTable.

```
import argparse
import time
import sys
import amazonadax
import boto3
```

```
def scan_test(iterations, dyn_resource=None):
    """
    Scans the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table = dyn_resource.Table('TryDaxTable')
    start = time.perf_counter()
    for _ in range(iterations):
        table.scan()
        print('.', end='')
        sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument(
        'endpoint_url', nargs='?',
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not used.")
    args = parser.parse_args()

    test_iterations = 100
    if args.endpoint_url:
        print(f"Scanning the table {test_iterations} times, using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after completion.
        with amazonadax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as dax:
            test_start, test_end = scan_test(test_iterations, dyn_resource=dax)
    else:
        print(f"Scanning the table {test_iterations} times, using the Boto3 client.")
        test_start, test_end = scan_test(test_iterations)
    print(f"Total time: {test_end - test_start:.4f} sec. Average time: "
          f"{{(test_end - test_start)/test_iterations}}")
```

06-delete-table.py

O programa 06-delete-table.py exclui TryDaxTable. Execute esse programa depois de concluir o teste da funcionalidade do Amazon DynamoDB Accelerator (DAX).

```
import boto3

def delete_dax_table(dyn_resource=None):
    """
    Deletes the demonstration table.

    :param dyn_resource: Either a Boto3 or DAX resource.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table = dyn_resource.Table('TryDaxTable')
    table.delete()

    print(f"Deleting {table.name}...")
```

```
    table.wait_until_not_exists()

if __name__ == '__main__':
    delete_dax_table()
    print("Table deleted!")
```

Como modificar um aplicativo existente para usar o DAX

Se você já tiver um aplicativo Java que use o Amazon DynamoDB, será necessário modificá-lo para que ele possa acessar seu cluster do DynamoDB Accelerator (DAX). Não é necessário reescrever todo o aplicativo porque o cliente Java do é muito semelhante ao cliente de baixo nível do DynamoDB incluído noAWS SDK for Java.

Suponha que você tenha uma tabela do DynamoDB denominadaMusic. A chave de partição dessa tabela éArtist, e sua chave de classificação éSongTitle. O seguinte programa lê um item diretamente da tabela Music.

```
import java.util.HashMap;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.GetItemRequest;
import com.amazonaws.services.dynamodbv2.model.GetItemResult;

public class GetMusicItem {

    public static void main(String[] args) throws Exception {

        // Create a DynamoDB client
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();

        HashMap<String, AttributeValue> key = new HashMap<String, AttributeValue>();
        key.put("Artist", new AttributeValue().withS("No One You Know"));
        key.put("SongTitle", new AttributeValue().withS("Scared of My Shadow"));

        GetItemRequest request = new GetItemRequest()
            .withTableName("Music").withKey(key);

        try {
            System.out.println("Attempting to read the item...");
            GetItemResult result = client.getItem(request);
            System.out.println("GetItem succeeded: " + result);

        } catch (Exception e) {
            System.err.println("Unable to read item");
            System.err.println(e.getMessage());
        }
    }
}
```

Para modificar o programa, substitua o cliente do DynamoDB por um cliente DAX.

```
import java.util.HashMap;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazon.dax.client.dynamodbv2.AmazonDaxClientBuilder;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.GetItemRequest;
```

```
import com.amazonaws.services.dynamodbv2.model.GetItemResult;

public class GetMusicItem {

    public static void main(String[] args) throws Exception {
        //Create a DAX client

        AmazonDaxClientBuilder daxClientBuilder = AmazonDaxClientBuilder.standard();
        daxClientBuilder.withRegion("us-
east-1").withEndpointConfiguration("mydaxcluster.2cmrw1.clustercfg.dax.usel.cache.amazonaws.com:8111");
        AmazonDynamoDB client = daxClientBuilder.build();

        /*
        ** ...
        ** Remaining code omitted (it is identical)
        ** ...
        */

    }
}
```

Uso da API de documento do DynamoDB

O AWS SDK for Java fornece uma interface de documento para o DynamoDB. A API de documento atua como um wrapper em torno do cliente do DynamoDB de baixo nível. Para obter mais informações, consulte [Interfaces de documento](#).

A interface de documento também pode ser usada com o cliente DAX de baixo nível, conforme mostrado no seguinte exemplo.

```
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazon.dax.client.dynamodbv2.AmazonDaxClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.GetItemOutcome;
import com.amazonaws.services.dynamodbv2.document.Table;

public class GetMusicItemWithDocumentApi {

    public static void main(String[] args) throws Exception {
        //Create a DAX client

        AmazonDaxClientBuilder daxClientBuilder = AmazonDaxClientBuilder.standard();
        daxClientBuilder.withRegion("us-
east-1").withEndpointConfiguration("mydaxcluster.2cmrw1.clustercfg.dax.usel.cache.amazonaws.com:8111");
        AmazonDynamoDB client = daxClientBuilder.build();

        // Document client wrapper
        DynamoDB docClient = new DynamoDB(client);

        Table table = docClient.getTable("Music");

        try {
            System.out.println("Attempting to read the item...");
            GetItemOutcome outcome = table.tgetItemOutcome(
                "Artist", "No One You Know",
                "SongTitle", "Scared of My Shadow");
            System.out.println(outcome.getItem());
            System.out.println("GetItem succeeded: " + outcome);
        } catch (Exception e) {
            System.err.println("Unable to read item");
        }
    }
}
```

```
        System.err.println(e.getMessage());
    }
}
```

Cliente assíncrono do DAX

O AmazonDaxClient é síncrono. Para uma operação de API DAX de longa execução, como umScanDe uma tabela grande, isso pode bloquear a execução do programa até que a operação seja concluída. Se o seu programa precisa realizar outros trabalhos enquanto uma operação de API do DAX está em andamento, useClusterDaxAsyncClientEm vez disso.

O programa a seguir mostra como usar ClusterDaxAsyncClient, junto com Java Future, para implementar uma solução sem bloqueio.

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
import java.util.HashMap;  
import java.util.concurrent.ExecutionException;  
import java.util.concurrent.Future;  
  
import com.amazonaws.dax.client.dynamodbv2.ClientConfig;  
import com.amazonaws.dax.client.dynamodbv2.ClusterDaxAsyncClient;  
import com.amazonaws.auth.profile.ProfileCredentialsProvider;  
import com.amazonaws.handlers.AsyncHandler;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBAsync;  
import com.amazonaws.services.dynamodbv2.model.AttributeValue;  
import com.amazonaws.services.dynamodbv2.model.GetItemRequest;  
import com.amazonaws.services.dynamodbv2.model.GetItemResult;  
  
public class DaxAsyncClientDemo {  
    public static void main(String[] args) throws Exception {  
  
        ClientConfig daxConfig = new ClientConfig().withCredentialsProvider(new  
        ProfileCredentialsProvider())  
            .withEndpoints("mydaxcluster.2cmrw1.clustercfg.dax.us-east-1.cache.amazonaws.com:8111");  
  
        AmazonDynamoDBAsync client = new ClusterDaxAsyncClient(daxConfig);  
  
        HashMap<String, AttributeValue> key = new HashMap<String, AttributeValue>();  
        key.put("Artist", new AttributeValue().withS("No One You Know"));  
        key.put("SongTitle", new AttributeValue().withS("Scared of My Shadow"));  
  
        GetItemRequest request = new GetItemRequest()  
            .withTableName("Music").withKey(key);  
  
        // Java Futures  
        Future<GetItemResult> call = client.getItemAsync(request);  
        while (!call.isDone()) {  
            // Do other processing while you're waiting for the response
```

```
        System.out.println("Doing something else for a few seconds...");  
        Thread.sleep(3000);  
    }  
    // The results should be ready by now  
  
    try {  
        call.get();  
  
    } catch (ExecutionException ee) {  
        // Futures always wrap errors as an ExecutionException.  
        // The *real* exception is stored as the cause of the  
        // ExecutionException  
        Throwable exception = ee.getCause();  
        System.out.println("Error getting item: " + exception.getMessage());  
    }  
  
    // Async callbacks  
    call = client.getItemAsync(request, new AsyncHandler<GetItemRequest, GetItemResult>() {  
  
        @Override  
        public void onSuccess(GetItemRequest request, GetItemResult getItemResult) {  
            System.out.println("Result: " + getItemResult);  
        }  
  
        @Override  
        public void onError(Exception e) {  
            System.out.println("Unable to read item");  
            System.err.println(e.getMessage());  
            // Callers can also test if exception is an instance of  
            // AmazonServiceException or AmazonClientException and cast  
            // it to get additional information  
        }  
  
    });  
    call.get();  
}  
}
```

Consultar índices secundários globais

Você pode usar o Amazon DynamoDB Accelerator (DAX) para consultar os índices secundários globais [do Uso do DynamoDB](#) [Interfaces programáticas](#).

O exemplo a seguir demonstra como usar o DAX para consultar o `CREATE_DATE` índice secundário global que é criado no [Exemplo: índices secundários globais que usam o AWS SDK for Java API de documento](#).

O `DAXClient` instancia os objetos de cliente que são necessários para interagir com as interfaces de programação do DynamoDB.

```
import com.amazon.dax.client.dynamodbv2.AmazonDaxClientBuilder;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.util.EC2MetadataUtils;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
  
public class DaxClient {  
  
    private static final String region = EC2MetadataUtils.getEC2InstanceRegion();  
  
    DynamoDB getDaxDocClient(String daxEndpoint) {
```

```
System.out.println("Creating a DAX client with cluster endpoint " + daxEndpoint);
AmazonDaxClientBuilder daxClientBuilder = AmazonDaxClientBuilder.standard();

daxClientBuilder.withRegion(region).withEndpointConfiguration(daxEndpoint);
AmazonDynamoDB client = daxClientBuilder.build();

return new DynamoDB(client);
}

DynamoDBMapper getDaxMapperClient(String daxEndpoint) {
    System.out.println("Creating a DAX client with cluster endpoint " + daxEndpoint);
    AmazonDaxClientBuilder daxClientBuilder = AmazonDaxClientBuilder.standard();

    daxClientBuilder.withRegion(region).withEndpointConfiguration(daxEndpoint);
    AmazonDynamoDB client = daxClientBuilder.build();

    return new DynamoDBMapper(client);
}
}
```

É possível consultar um índice secundário global das seguintes formas:

- Use o método `queryIndex` na classe `QueryIndexDax` definida no seguinte exemplo. O `QueryIndexDax` considera como um parâmetro o objeto de cliente que é retornado pelo método `getDaxDocClient` na classe `DaxClient`.
- Se você estiver usando a [interface de persistência de objetos](#), use o método `queryIndexMapper` na classe `QueryIndexDax` definida no seguinte exemplo. O `queryIndexMapper` considera como um parâmetro o objeto de cliente que é retornado pelo método `getDaxMapperClient` definido na classe `DaxClient`.

```
import java.util.Iterator;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;
import java.util.List;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBQueryExpression;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import java.util.HashMap;
import com.amazonaws.services.dynamodbv2.document.Item;
import com.amazonaws.services.dynamodbv2.document.utils.ValueMap;
import com.amazonaws.services.dynamodbv2.document.spec.QuerySpec;
import com.amazonaws.services.dynamodbv2.document.QueryOutcome;
import com.amazonaws.services.dynamodbv2.document.ItemCollection;
import com.amazonaws.services.dynamodbv2.document.Index;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;

public class QueryIndexDax {

    //This is used to query Index using the low-level interface.
    public static void queryIndex(DynamoDB client, String tableName, String indexName) {
        Table table = client.getTable(tableName);

        System.out.println("\n*****\n");
        System.out.print("Querying index " + indexName + "...");

        Index index = table.getIndex(indexName);

        ItemCollection<QueryOutcome> items = null;

        QuerySpec querySpec = new QuerySpec();

        if (indexName == "CreateDateIndex") {
            System.out.println("Issues filed on 2013-11-01");
        }
    }
}
```

```
querySpec.withKeyConditionExpression("CreateDate = :v_date and
begins_with(IssueId, :v_issue)")
    .withValueMap(new ValueMap().withString(":v_date",
"2013-11-01").withString(":v_issue", "A-"));
    items = index.query(querySpec);
} else {
    System.out.println("\nNo valid index name provided");
    return;
}

Iterator<Item> iterator = items.iterator();

System.out.println("Query: printing results...");

while (iterator.hasNext()) {
    System.out.println(iterator.next().toJSONPretty());
}

}

//This is used to query Index using the high-level mapper interface.
public static void queryIndexMapper(DynamoDBMapper mapper, String tableName, String
indexName) {
    HashMap<String,AttributeValue> eav = new HashMap<String,AttributeValue>();
    eav.put(":v_date", new AttributeValue().withS("2013-11-01"));
    eav.put(":v_issue", new AttributeValue().withS("A-"));
    DynamoDBQueryExpression<CreateDate> queryExpression = new
DynamoDBQueryExpression<CreateDate>()
        .withIndexName("CreateDateIndex").withConsistentRead(false)
        .withKeyConditionExpression("CreateDate = :v_date and begins_with(IssueId, :v_issue)")
        .withExpressionAttributeValues(eav);

    List<CreateDate> items = mapper.query(CreateDate.class, queryExpression);
    Iterator<CreateDate> iterator = items.iterator();

    System.out.println("Query: printing results...");

    while (iterator.hasNext()) {
        CreateDate iterObj = iterator.next();
        System.out.println(iterObj.getCreateDate());
        System.out.println(iterObj.getIssueId());
    }
}
}
```

A definição de classe abaixo representa a tabela de problemas e é usada no método `queryIndexMapper`.

```
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBIndexHashKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBIndexRangeKey;
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBHashKey;

@DynamoDBTable(tableName = "Issues")
public class CreateDate {
    private String createDate;
    @DynamoDBHashKey(attributeName = "IssueId")
    private String issueId;

    @DynamoDBIndexHashKey(globalSecondaryIndexName = "CreateDateIndex", attributeName =
"CreateDate")
    public String getCreateDate() {
        return createDate;
    }
}
```

```
public void setCreateDate(String createDate) {
    this.createDate = createDate;
}

@DynamoDBIndexRangeKey(globalSecondaryIndexName = "CreateDateIndex", attributeName =
"IssueId")
public String getIssueId() {
    return issueId;
}

public void setIssueId(String issueId) {
    this.issueId = issueId;
}
}
```

Gerenciando clusters DAX

Esta seção aborda algumas das tarefas comuns de gerenciamento de clusters do Amazon DynamoDB Accelerator (DAX).

Tópicos

- [Permissões do IAM para gerenciar um cluster DAX \(p. 782\)](#)
- [Escalabilidade de um cluster DAX \(p. 784\)](#)
- [Personalização das configurações do cluster DAX \(p. 785\)](#)
- [Configuração de definições de TTL \(p. 786\)](#)
- [Support à marcação para DAX \(p. 787\)](#)
- [Integração do AWS CloudTrail \(p. 787\)](#)
- [Excluindo um cluster DAX \(p. 788\)](#)

Permissões do IAM para gerenciar um cluster DAX

Quando você administra um cluster DAX usando o comando AWS Management Console ou o AWS Command Line Interface (AWS CLI), é altamente recomendável que restrinja o escopo de ações que os usuários podem executar. Ao fazer isso, você pode ajudar a reduzir os riscos, enquanto segue o princípio do privilégio mínimo.

A discussão a seguir aborda o controle de acesso das APIs de gerenciamento do DAX. Para obter mais informações, consulte [Amazon DynamoDB Accelerator](#) no Referência da API do Amazon DynamoDB.

Note

Para obter mais informações detalhadas sobre o gerenciamento do AWS Identity and Access Management Permissões do (IAM), consulte o seguinte:

- IAM e criação de clusters DAX:[Criar um cluster DAX \(p. 723\)](#).
- Operações de plano de dados do IAM e DAX:[Controle de acesso DAX \(p. 805\)](#).

Para as APIs de gerenciamento do DAX, você não pode projetar as ações da API para um recurso específico. O elemento Resource deve ser definido como "*". Isso é diferente das operações da API de plano de dados do DAX, como `GetItem`, `Query`, `Scan`. As operações do plano de dados são expostas por meio do cliente DAX e essas operações CANter escopo para recursos específicos.

Para ilustrar, considere o documento de política do IAM a seguir.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "dax:*"  
            ],  
            "Effect": "Allow",  
            "Resource": [  
                "arn:aws:dax:us-west-2:123456789012:cache/DAXCluster01"  
            ]  
        }  
    ]  
}
```

Suponha que a intenção dessa política seja permitir chamadas de API de gerenciamento DAX para o cluster DAXCluster01— e somente esse cluster.

Agora, suponha que um usuário emita o seguinte comando da AWS CLI.

```
aws dax describe-clusters
```

Esse comando falha com uma exceção de Não autorizado, pois a chamada à API `DescribeClusters` subjacente não pode ser colocada no escopo de um cluster específico. Embora a política seja sintaticamente válida, o comando falha porque o elemento `Resource` deve ser definido como `"*"`. No entanto, se o usuário executar um programa que envia chamadas de plano de dados DAX (tal como `GetItem` ou `Query`) para `DAXCluster01`, essas chamadas DDoF ter sucesso. Isso ocorre porque as APIs de plano de dados do DAX podem ser colocadas no escopo de recursos específicos (neste caso, `DAXCluster01`).

Se você deseja escrever uma única política de IAM abrangente para englobar as APIs de gerenciamento do DAX e as APIs de plano de dados do DAX, sugerimos que inclua duas instruções distintas no documento da política. Uma dessas instruções deve lidar com as APIs de plano de dados do DAX, enquanto a outra instrução lida com as APIs de gerenciamento do DAX.

A política de exemplo a seguir mostra essa abordagem. Observe como a declaração DAXDataAPIs é colocada no escopo do recurso DAXCluster01, mas o recurso para DAXManagementAPIs deve ser "*". As ações mostradas em cada declaração são apenas para ilustração. É possível personalizá-las conforme necessário para seu aplicativo.

```
"Action": [
    "dax>CreateParameterGroup",
    "dax>CreateSubnetGroup",
    "dax>DecreaseReplicationFactor",
    "dax>DeleteCluster",
    "dax>DeleteParameterGroup",
    "dax>DeleteSubnetGroup",
    "dax>DescribeClusters",
    "dax>DescribeDefaultParameters",
    "dax>DescribeEvents",
    "dax>DescribeParameterGroups",
    "dax>DescribeParameters",
    "dax>DescribeSubnetGroups",
    "dax>IncreaseReplicationFactor",
    "dax>ListTags",
    "dax>RebootNode",
    "dax>TagResource",
    "dax>UntagResource",
    "dax>UpdateCluster",
    "dax>UpdateParameterGroup",
    "dax>UpdateSubnetGroup"
],
"Effect": "Allow",
"Resource": [
    "*"
]
}
]
```

Escalabilidade de um cluster DAX

Há duas opções disponíveis para escalar um cluster DAX. A primeira opção é a escalabilidade horizontal, na qual você adiciona réplicas de leitura ao cluster. A segunda opção é a escalabilidade vertical, na qual você seleciona diferentes tipos de nó. Para obter conselhos sobre como abordar a escolha de um tamanho de cluster apropriado e um tipo de nó para seu aplicativo, consulte [Guia de dimensionamento de cluster do DAX \(p. 827\)](#).

Escalabilidade horizontal

Com a escalabilidade horizontal, você pode melhorar a taxa de transferência para operações de leitura adicionando mais réplicas de leitura ao cluster. Um único cluster DAX aceita até 10 réplicas de leitura, e você pode adicionar ou remover réplicas enquanto o cluster está em execução.

Os exemplos de AWS CLI a seguir mostram como aumentar ou diminuir o número de nós. O argumento `--new-replication-factor` especifica o número total de nós no cluster. Um dos nós é o nó primário, e os outros nós são réplicas de leitura.

```
aws dax increase-replication-factor \
--cluster-name MyNewCluster \
--new-replication-factor 5
```

```
aws dax decrease-replication-factor \
--cluster-name MyNewCluster \
--new-replication-factor 3
```

Note

O status do cluster muda para `modifying` quando você modifica o fator de replicação. O status muda para `available` quando a modificação é concluída.

Escalabilidade vertical

Se você tiver um grande conjunto de dados de trabalho, seu aplicativo poderá se beneficiar do uso de maiores tipos de nós. Os nós maiores podem permitir que o cluster armazene mais dados na memória, o que reduz os erros de cache e melhora o desempenho geral do aplicativo. (Todos os nós em um cluster DAX devem ser do mesmo tipo.)

Se o cluster DAX tiver uma alta taxa de operações de gravação ou falhas de cache, o aplicativo também poderá se beneficiar do uso de tipos maiores de nós. Operações de gravação e erros de cache consomem recursos no nó primário do cluster. Portanto, usar tipos maiores de nós pode aumentar o desempenho do nó primário e permitir uma taxa de transferência mais alta para esses tipos de operações.

Não é possível modificar os tipos de nós em um cluster DAX em execução. Em vez disso, você deve criar um novo cluster com o tipo de nó desejado. Para obter uma lista dos tipos de nó compatíveis, consulte [Nodes \(p. 719\)](#).

Você pode criar um novo cluster DAX usando a AWS Management Console, [AWS CloudFormation](#), o AWS CLI, ou o [AWSSDK](#). (Para a AWS CLI, use o parâmetro `--node-type` para especificar o tipo de nó.)

Personalização das configurações do cluster DAX

Quando você cria um cluster DAX, as configurações padrão a seguir são usadas:

- Remoção automática de cache habilitada com tempo de vida (TTL) de 5 minutos
- Nenhuma preferência para zonas de disponibilidade
- Nenhuma preferência para janelas de manutenção
- Notificações desabilitadas

Para novos clusters, você pode personalizar as configurações durante a criação. Para fazer isso no AWS Management Console, desmarque **Use default settings** (Usar configurações padrão) para modificar os seguintes ajustes:

- Redes e segurança- Permite que você execute nós individuais do cluster DAX em diferentes Zonas de disponibilidade dentro do AWS Região : Se você escolher **No Preference** (Sem preferência), os nós serão distribuídos entre as zonas de disponibilidade automaticamente.
- Parameter Group- Um conjunto nomeado de parâmetros que são aplicados a cada nó no cluster. Você pode usar um grupo de parâmetros para especificar o comportamento do TTL do cache. Você pode alterar o valor de qualquer parâmetro determinado em um grupo de parâmetros (exceto o grupo de parâmetros padrão `default.dax.1.0`) a qualquer momento.
- Maintenance Window- Um período de tempo semanal durante o qual são aplicadas atualizações e patches aos nós do cluster. Você pode escolher o dia e a hora de início e a duração da janela de manutenção. Se você escolher **No Preference** (Sem preferência), a janela de manutenção será selecionada aleatoriamente em um bloco de tempo de 8 horas por região. Para obter mais informações, consulte [Janela de manutenção \(p. 722\)](#).

Note

Parameter Group (Grupo de parâmetros) e Maintenance Window (Janela de manutenção) também podem ser modificados a qualquer momento em um cluster em execução.

Quando ocorre um evento de manutenção, o DAX pode notificá-lo usando o Amazon Simple Notification Service (Amazon SNS). Para configurar notificações, escolha uma opção no seletor Topic for SNS notification (Tópico de notificação do SNS). Você pode criar um novo tópico do Amazon SNS ou usar um tópico existente.

Para obter mais informações sobre como configurar e assinar um tópico do Amazon SNS, consulte[Conceitos básicos do Amazon SNS](#)noGuia do desenvolvedor do Amazon Simple Notification Service.

Configuração de definições de TTL

O DAX mantém dois caches para dados que ele lê no DynamoDB:

- Cache de itens—Para itens recuperados usando[GetItem](#)ou[BatchGetItem](#).
- Cache de consultas- Para conjuntos de resultados recuperados usando[Query](#)ou[Scan](#).

Para obter mais informações, consulte [Cache de itens \(p. 718\)](#) e [Cache de consultas \(p. 718\)](#).

O TTL padrão de cada um desses caches é 5 minutos. Se você deseja usar configurações diferentes de TTL, pode iniciar um cluster DAX usando um grupo de parâmetros personalizados. Para fazer isso no console, escolha DAX | Parameter groups (DAX | grupos de parâmetros) no painel de navegação.

Você também pode realizar essas tarefas usando a AWS CLI. O exemplo a seguir mostra como iniciar um novo cluster DAX usando um grupo de parâmetros personalizados. Neste exemplo, o TTL do cache de itens é definido como 10 minutos, e o TTL do cache de consultas é definido como 3 minutos.

1. Crie um novo parameter group.

```
aws dax create-parameter-group \
--parameter-group-name custom-ttl
```

2. Defina o TTL do cache de itens para 10 minutos (600000 milissegundos).

```
aws dax update-parameter-group \
--parameter-group-name custom-ttl \
--parameter-name-values "ParameterName=record-ttl-millis,ParameterValue=600000"
```

3. Defina o TTL do cache de consultas para 3 minutos (180000 milissegundos).

```
aws dax update-parameter-group \
--parameter-group-name custom-ttl \
--parameter-name-values "ParameterName=query-ttl-millis,ParameterValue=180000"
```

4. Verifique se os parâmetros foram configurados corretamente.

```
aws dax describe-parameters --parameter-group-name custom-ttl \
--query "Parameters[*].[ParameterName,Description,ParameterValue]"
```

Agora você pode iniciar um novo cluster DAX com esse grupo de parâmetros.

```
aws dax create-cluster \
--cluster-name MyNewCluster \
--node-type dax.r3.large \
--replication-factor 3 \
--iam-role-arn arn:aws:iam::123456789012:role/DAXServiceRole \
--parameter-group custom-ttl
```

Note

Não é possível modificar um grupo de parâmetros que está sendo usado por uma instância DAX em execução.

Support à marcação para DAX

Muitos AWS, incluindo o DynamoDB, suportam marcação - a habilidade de rotular recursos com nomes definidos pelo usuário. Você pode atribuir tags a clusters DAX, permitindo que você identifique rapidamente todos os seus AWS que têm a mesma tag, ou para categorizar o AWS faturas pelas etiquetas que você atribui.

Para obter mais informações, consulte [Adicionar tags e rótulos a recursos \(p. 388\)](#).

Usar a AWS Management Console

Para gerenciar tags de cluster DAX

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, em DAX, escolha Clusters.
3. Escolha o cluster com o qual você deseja trabalhar.
4. Escolha a guia Tags. Você pode adicionar, listar, editar ou excluir suas tags aqui.

Quando as configurações estiverem conforme você deseja, escolha Apply Changes (Aplicar alterações).

Usar a AWS CLI

Quando você usa o comando AWS CLI Para gerenciar tags de cluster DAX, você deve primeiro determinar o Nome de recurso da Amazon (ARN) do cluster. O exemplo a seguir mostra como determinar o ARN de um cluster chamado MyDAXCluster.

```
aws dax describe-clusters \
--cluster-name MyDAXCluster \
--query "Clusters[*].ClusterArn"
```

Na saída, o ARN será semelhante a este: arn:aws:dax:us-west-2:123456789012:cache/MyDAXCluster

O exemplo a seguir mostra como atribuir tags ao cluster.

```
aws dax tag-resource \
--resource-name arn:aws:dax:us-west-2:123456789012:cache/MyDAXCluster \
--tags="Key=ClusterUsage,Value=prod"
```

Para listar todas as tags de um cluster.

```
aws dax list-tags \
--resource-name arn:aws:dax:us-west-2:123456789012:cache/MyDAXCluster
```

Para remover uma tag, especifique a chave dela.

```
aws dax untag-resource \
--resource-name arn:aws:dax:us-west-2:123456789012:cache/MyDAXCluster \
--tag-keys ClusterUsage
```

Integração do AWS CloudTrail

O DAX está integrado ao AWS CloudTrail, o que permite auditar as atividades do cluster DAX. Você pode usar os logs do CloudTrail para visualizar todas as alterações que foram feitas no nível do cluster. Também

é possível ver as alterações em componentes do cluster, como nós, grupos de sub-redes e grupos de parâmetros. Para obter mais informações, consulte [Registro de operações do DynamoDB usando o AWS CloudTrail \(p. 954\)](#).

Excluindo um cluster DAX

Se você não estiver mais usando um cluster DAX, deverá excluí-lo para evitar ser cobrado por recursos não utilizados.

É possível excluir um cluster DAX usando o console ou a AWS CLI. Veja um exemplo a seguir.

```
aws dax delete-cluster --cluster-name mydaxcluster
```

Monitoramento DAX

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e o desempenho do Amazon DynamoDB Accelerator (DAX) e do AWS Solutions do. Você deve coletar dados de monitoramento de todas as partes do seu AWS Para que você possa depurar uma falha de vários pontos com facilidade, caso ocorra.

Antes de começar a monitorar o DAX, crie um plano de monitoramento que inclua respostas às seguintes perguntas:

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?
- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem realizará o monitoramento das tarefas?
- Quem deve ser notificado quando algo der errado?

A próxima etapa é estabelecer um parâmetro de desempenho normal do DAX no ambiente medindo o desempenho em vários momentos e em diferentes condições de carga. Ao monitorar o DAX, você deve pensar na possibilidade de armazenar os dados históricos de monitoramento. Esses dados armazenados fornecem a você uma linha de base com a qual comparar os dados de desempenho atuais, identificar padrões normais e anomalias de desempenho e criar métodos para solucionar problemas.

Para estabelecer uma linha de base, você deve, no mínimo, monitorar os seguintes itens durante o teste de carga e na produção:

- Utilização da CPU e solicitações limitadas, para que seja possível determinar se é necessário usar um tipo de nó maior no cluster. A utilização da CPU do seu cluster está disponível através da [CPUUtilization Métrica CloudWatch \(p. 791\)](#).
- A latência da operação (medida no lado do cliente) deve permanecer consistentemente dentro dos requisitos de latência do aplicativo.
- As taxas de erro devem permanecer baixas, como pode ser visto [noErrorRequestCount, FaultRequestCount, e FailedRequestCount Métricas do CloudWatch \(p. 791\)](#).

Além dos itens anteriores, você deve, no mínimo, monitorar os seguintes itens adicionais em produção:

- Tamanho estimado do banco de dados e tamanho da remoção, para que você possa determinar se o tipo de nó do cluster tem memória suficiente para manter seu conjunto de trabalho.

- Conexões de cliente, para que você possa monitorar todos os picos inexplicáveis em conexões com o cluster.

Tópicos

- [Ferramentas de monitoramento \(p. 789\)](#)
- [Monitoramento com Amazon CloudWatch \(p. 790\)](#)
- [Registrar operações do DAXAWS CloudTrail \(p. 803\)](#)

Ferramentas de monitoramento

AWS fornece ferramentas que você pode usar para monitorar o Amazon DynamoDB Accelerator (DAX). É possível configurar algumas dessas ferramentas para fazer o monitoramento para você, e algumas delas exigem intervenção manual. Recomendamos que as tarefas de monitoramento sejam automatizadas ao máximo possível.

Tópicos

- [Ferramentas de monitoramento automatizadas \(p. 789\)](#)
- [Ferramentas de monitoramento manual \(p. 789\)](#)

Ferramentas de monitoramento automatizadas

Use as seguintes ferramentas automatizadas de monitoramento para supervisionar o DAX e gerar relatórios quando algo estiver errado:

- Amazon CloudWatch Alarms (Alarms do Amazon CloudWatch): observe uma única métrica ao longo de um período que você especificar e realize uma ou mais ações com base no valor da métrica em relação a um determinado limite ao longo de vários períodos. A ação é uma notificação enviada para um tópico do Amazon Simple Notification Service (Amazon SNS) ou por uma política do Amazon EC2 Auto Scaling. Os alarmes do CloudWatch não invocam ações simplesmente porque estão em um estado específico. O estado deve ter sido alterado e mantido por um número específico de períodos. Para mais informações, consulte [Monitorar com o Amazon CloudWatch \(p. 930\)](#).
- Amazon CloudWatch LogsMonitor, armazene e accesse seus arquivos de log do AWS CloudTrailou outras fontes. Para obter mais informações, consulte [Monitorar arquivos de log](#) no Guia do usuário do Amazon CloudWatch.
- Amazon CloudWatch Events: faça correspondência de eventos e direcione-os a uma ou mais funções ou streams de destino para fazer alterações, capturar informações de estado e realizar ações corretivas. Para obter mais informações, consulte [O que são Amazon CloudWatch Events](#)noGuia do usuário do Amazon CloudWatch.
- AWS CloudTrailMonitorar logs do— compartilhe arquivos de log entre contas, monitore os arquivos de log do CloudTrail em tempo real enviando-os para o CloudWatch Logs, escreva aplicativos de processamento de logs em Java e confirme se os arquivos de log não foram alterados após a distribuição pelo CloudTrail. Para obter mais informações, consulte [Trabalho com arquivos de log do CloudTrail](#)noAWS CloudTrailGuia do usuário.

Ferramentas de monitoramento manual

Outra parte importante do monitoramento do DAX é o monitoramento manual dos itens que os alarmes do CloudWatch não abrangem. O DAX, CloudWatch, Trusted Advisor e outros AWS Management ConsoleOs painéis do fornecem uma visão rápida do estado do seu AWS Ambiente do. Recomendamos que você também verifique os arquivos de registro do DAX.

- O painel do DAX mostra o seguinte:

- Integridade de serviço
- A página inicial do CloudWatch mostra o seguinte:
 - Alertas e status atual
 - Gráficos de alertas e recursos
 - Estado de integridade do serviço

Além disso, você pode usar o CloudWatch para fazer o seguinte:

- Criar [painéis personalizados](#) para monitorar os serviços de seu interesse.
- Colocar em gráfico dados de métrica para solucionar problemas e descobrir tendências.
- Pesquisar e procurar todas as métricas de recursos da AWS.
- Criar e editar alertas para ser notificado sobre problemas.

Monitoramento com Amazon CloudWatch

É possível monitorar o Amazon DynamoDB Accelerator (DAX) usando o Amazon CloudWatch, que coleta e processa dados brutos do DAX em métricas legíveis quase que em tempo real. Essas estatísticas são gravadas durante um período de duas semanas. Você pode acessar as informações históricas para obter uma perspectiva melhor sobre o desempenho do aplicativo web ou do serviço. Por padrão, os dados de métrica do DAX são enviados para o CloudWatch automaticamente. Para obter mais informações, consulte [O que é o Amazon CloudWatch?](#) no Guia do usuário do Amazon CloudWatch.

Tópicos

- [Como usar métricas do DAX? \(p. 790\)](#)
- [Exibição de dimensões e métricas do DAX \(p. 791\)](#)
- [Criação de alarmes do CloudWatch para monitorar o DAX \(p. 802\)](#)

Como usar métricas do DAX?

As métricas informadas pelo DAX fornecem informações que você pode analisar de diferentes maneiras. A lista a seguir mostra alguns usos comuns para as métricas. Essas são sugestões para você começar, e não uma lista abrangente.

Como?	Métricas relevantes
Determinar se ocorreu algum erro do sistema	Monitore <code>FaultRequestCount</code> para determinar se alguma solicitação resultou em um código HTTP 500 (erro de servidor). Isso pode indicar um erro de serviço interno do DAX ou um HTTP 500 na Métrica SystemErrors .
Determinar se ocorreu algum erro do usuário	Monitore <code>ErrorRequestCount</code> para determinar se alguma solicitação resultou em um código HTTP 400 (erro de cliente). Se você vir a contagem de erros aumentando, investigue e verifique se você está enviando solicitações de clientes corretas.
Determinar se ocorreu alguma ausência no cache	Monitore <code>ItemCacheMisses</code> para determinar o número de vezes que um item não foi encontrado no cache e <code>QueryCacheMisses</code> e <code>ScanCacheMisses</code> para determinar o número de vezes que uma consulta ou um resultado de verificação não foi encontrado no cache.
Monitorar taxas de acerto do cache	Use a Matemática de métricas do CloudWatch para definir uma métrica da taxa de acertos do cache usando expressões matemáticas.

Como?	Métricas relevantes
	Por exemplo, para o cache do item, você pode usar a expressão $m1/\text{SUM}([m1, m2]) * 100$, em que $m1$ é a métrica <code>ItemCacheHits</code> e $m2$ é a métrica <code>ItemCacheMisses</code> para seu cluster. Para os caches de verificação e consulta, você pode seguir o mesmo padrão usando a métrica do cache de verificação e consulta correspondente.

Exibição de dimensões e métricas do DAX

Quando você interage com o Amazon DynamoDB, ele envia métricas e dimensões para o Amazon CloudWatch. Use os procedimentos a seguir para visualizar as métricas do DynamoDB Accelerator (DAX).

Para exibir métricas (console)

As métricas são agrupadas primeiro pelo namespace do serviço e, em seguida, por várias combinações de dimensão dentro de cada namespace.

1. Abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. No painel de navegação, selecione Metrics (Métricas).
3. Selecione o namespace DAX.

Para visualizar as métricas (AWS CLI)

- Em um prompt de comando, use o seguinte comando.

```
aws cloudwatch list-metrics --namespace "AWS/DAX"
```

Métricas e dimensões do DAX

As seções a seguir contêm as métricas e as dimensões que o DAX envia para o CloudWatch.

Métricas do DAX

As métricas a seguir estão disponíveis no DAX. O DAX envia métricas ao CloudWatch somente quando elas têm um valor diferente de zero.

Note

O CloudWatch agrupa as seguintes métricas do DAX em intervalos de um minuto:

- `CPUUtilization`
- `CacheMemoryUtilization`
- `NetworkBytesIn`
- `NetworkBytesOut`
- `NetworkPacketsIn`
- `NetworkPacketsOut`
- `GetItemRequestCount`
- `BatchGetItemRequestCount`
- `BatchWriteItemRequestCount`
- `DeleteItemRequestCount`
- `PutItemRequestCount`
- `UpdateItemRequestCount`

- TransactWriteItemCount
- TransactGetItemCount
- ItemCacheHits
- ItemCacheMisses
- QueryCacheHits
- QueryCacheMisses
- ScanCacheHits
- ScanCacheMisses
- TotalRequestCount
- ErrorRequestCount
- FaultRequestCount
- FailedRequestCount
- QueryRequestCount
- ScanRequestCount
- ClientConnections
- EstimatedDbSize
- EvictedSize
- CPUCreditUsage
- CPUCreditBalance
- CPUSurplusCreditBalance
- CPUSurplusCreditsCharged

Nem todas as estatísticas, como Average ou Sum, são aplicáveis a todas as métricas do. No entanto, todos esses valores estão disponíveis por meio do console DAX ou usando o console do CloudWatch, AWS CLI, ou AWS SDKs para todas as métricas. Na tabela a seguir, cada métrica tem uma lista de estatísticas válidas que são aplicáveis a essa métrica.

Métrica	Descrição
CPUUtilization	<p>O percentual de utilização da CPU do nó do ou do cluster.</p> <p>Unidades de :Percent</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average
CacheMemoryUtilization	<p>A porcentagem de memória cache disponível que está em uso pelo cache de item e cache de consulta no nó ou cluster. Os dados armazenados em cache começam a ser despejados antes que a utilização da memória atinja 100% (consulte EvictedSizeMétrica). Se CacheMemoryUtilization atinge 100% em qualquer nó, as solicitações de gravação serão limitadas e você deve considerar a mudança para um cluster com um tipo de nó maior.</p> <p>Unidades de :Percent</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum

Métrica	Descrição
	<ul style="list-style-type: none"> • Maximum • Average
NetworkBytesIn	<p>O número de bytes recebidos em todas as interfaces de rede pelo nó ou cluster.</p> <p>Unidades de :Bytes</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average
NetworkBytesOut	<p>O número de bytes enviados em todas as interfaces de rede pelo nó ou cluster. Essa métrica identifica o volume de tráfego de saída em termos do número de bytes em um único nó ou cluster.</p> <p>Unidades de :Bytes</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average
NetworkPacketsIn	<p>O número de pacotes recebidos em todas as interfaces de rede pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average
NetworkPacketsOut	<p>O número de pacotes enviados em todas as interfaces de rede pelo nó ou cluster. Essa métrica identifica o volume de tráfego de saída em termos do número de pacotes em um único nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average

Métrica	Descrição
GetItemRequestCount	<p>O número de <code>GetItem</code> manipuladas pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
BatchGetItemRequestCount	<p>O número de <code>BatchGetItem</code> manipuladas pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
BatchWriteItemRequestCount	<p>O número de <code>BatchWriteItem</code> manipuladas pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
DeleteItemRequestCount	<p>O número de <code>DeleteItem</code> manipuladas pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>

Métrica	Descrição
<code>PutItemRequestCount</code>	<p>O número de <code>PutItem</code> manipuladas pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>UpdateItemRequestCount</code>	<p>O número de <code>UpdateItem</code> manipuladas pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>TransactWriteItemCount</code>	<p>O número de <code>TransactWriteItems</code> manipuladas pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>TransactGetItemCount</code>	<p>O número de <code>TransactGetItems</code> manipuladas pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>

Métrica	Descrição
<code>ItemCacheHits</code>	<p>O número de vezes que um item foi retornado do cache pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>ItemCacheMisses</code>	<p>O número de vezes que um item não estava no cache de nó ou cluster e precisava ser recuperado do DynamoDB.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>QueryCacheHits</code>	<p>O número de vezes que um resultado de consulta foi retornado do cache de nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>QueryCacheMisses</code>	<p>O número de vezes que um resultado de consulta não estava no cache de nó ou cluster e precisava ser recuperado do DynamoDB.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>

Métrica	Descrição
ScanCacheHits	<p>O número de vezes que um resultado de varredura foi retornado do cache de nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • SampleCount • Sum
ScanCacheMisses	<p>O número de vezes que um resultado de varredura não estava no cache de nó ou cluster e precisava ser recuperado do DynamoDB.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • SampleCount • Sum
TotalRequestCount	<p>Número total de solicitações tratadas pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • SampleCount • Sum
ErrorRequestCount	<p>Número total de solicitações que resultaram em um erro de usuário relatado pelo nó ou cluster. As solicitações que foram limitadas pelo nó ou cluster são incluídas.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • SampleCount • Sum

Métrica	Descrição
<code>ThrottledRequestCount</code>	<p>Número total de solicitações limitadas pelo nó ou cluster. As solicitações que foram controladas pelo DynamoDB não são incluídas e podem ser monitoradas usando Métricas do DynamoDB (p. 930).</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>FaultRequestCount</code>	<p>Número total de solicitações que resultaram em um erro interno relatado pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>FailedRequestCount</code>	<p>O número total de solicitações que resultaram em erro informado pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>QueryRequestCount</code>	<p>O número de solicitações de consulta tratadas pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>

Métrica	Descrição
ScanRequestCount	<p>O número de solicitações de varredura tratadas pelo nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • SampleCount • Sum
ClientConnections	<p>O número de conexões simultâneas feitas pelos clientes ao nó ou cluster.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • SampleCount • Sum
EstimatedDbSize	<p>Uma aproximação da quantidade de dados armazenados em cache no cache de item e no cache de consulta pelo nó ou cluster.</p> <p>Unidades de :Bytes</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average
EvictedSize	<p>A quantidade de dados que foi despejada pelo nó ou cluster para criar espaço para dados recém-solicitados. Se a taxa de erros subir, e você ver essa métrica também crescendo, isso provavelmente significa que seu conjunto de trabalho aumentou. Você deve considerar alternar para um cluster com um tipo de nó maior.</p> <p>Unidades de :Bytes</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • Sum

Métrica	Descrição
CPUCreditUsage	<p>O número de créditos de CPU gastos pelo nó para utilização de CPU. Um crédito de CPU equivale a um vCPU em execução em 100% de utilização por um minuto ou a uma combinação equivalente de vCPUs, utilização e tempo (por exemplo, um vCPU em execução a 50% de utilização por dois minutos ou dois vCPUs em execução a 25% de utilização por dois minutos).</p> <p>As métricas de crédito de CPU estão disponíveis a uma frequência de apenas 5 minutos. Se você especificar um período de mais cinco minutos, use o <code>Sum</code> estatística em vez da <code>Average</code>.</p> <p>Unidades de :<code>Credits</code> (vCPU-minutes)</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
CPUCreditBalance	<p>O número de créditos ganhos de CPU que um nó acumulou desde que foi executado ou iniciado.</p> <p>Os créditos são acumulados no saldo de créditos após terem sido ganhos e são removidos do saldo de créditos quando são gastos. O saldo de crédito tem um limite máximo, determinado pelo tamanho do nó DAX. Depois que o limite for atingido, todos os novos créditos ganhos serão descartados.</p> <p>Os créditos no <code>CPUCreditBalance</code> estão disponíveis para que o nó gaste e apresente intermitência com uma utilização de CPU acima da linha de base.</p> <p>Unidades de :<code>Credits</code> (vCPU-minutes)</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>

Métrica	Descrição
CPUSurplusCreditBalance	<p>O número de créditos excedentes gastos por um nó DAX quando o CPUCreditBalance valor é zero.</p> <p>O valor CPUSurplusCreditBalance é pago pelos créditos de CPU ganhos. Se o número de créditos excedentes ultrapassar o número máximo de créditos que o nó pode ganhar em um período de 24 horas, os créditos excedentes gastos acima do limite máximo incorrerão em uma taxa adicional.</p> <p>Unidades de :Credits (vCPU-minutes)</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • SampleCount • Sum
CPUSurplusCreditsCharged	<p>O número de créditos excedentes gastos que não são pagos pelos créditos de CPU ganhos e que, portanto, incorrem em uma cobrança adicional.</p> <p>Os créditos excedentes gastos são cobrados quando os créditos excedentes ultrapassaram o número máximo de créditos que o nó pode ganhar em um período de 24 horas. Os créditos excedentes gastos acima do limite máximo são cobrados no final da hora ou quando o nó é encerrado.</p> <p>Unidades de :Credits (vCPU-minutes)</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • SampleCount • Sum

Note

OCPUCreditUsage, CPUCreditBalance, CPUSurplusCreditBalance, e CPUSurplusCreditsCharged As métricas estão disponíveis apenas para nós T3.

Dimensões para métricas do DAX

As métricas do DAX são qualificadas pelos valores da conta, ID de cluster ou combinação de ID de cluster e ID de nó. Você pode usar o console do CloudWatch para recuperar dados do DAX em qualquer uma das dimensões da tabela a seguir.

Dimensão	Descrição
Account	Exibe estatísticas agregadas em todos os nós em uma conta.
ClusterId	Limita os dados a um cluster. Metrics
ClusterId, NodeId	Limita os dados a um nó dentro de um cluster

Criação de alarmes do CloudWatch para monitorar o DAX

Você pode criar um alarme do Amazon CloudWatch que envia uma mensagem do Amazon Simple Notification Service (Amazon SNS) quando o alarme muda de estado. Um alarme observa uma única métrica por um período tempo que você especifica. Ele executa uma ou mais ações com base no valor da métrica em relação a um limite especificado ao longo de vários períodos. A ação é uma notificação enviada para um tópico do Amazon SNS ou uma política de Auto Scaling do. Os alertas invocam ações apenas para alterações de estado mantidas. Os alarmes do CloudWatch não invocam ações só porque estão em um determinado estado. O estado deve ter sido alterado e mantido por uma quantidade especificada de períodos.

Como posso ser notificado sobre as ausências no cache de consulta?

1. Crie um tópico do Amazon SNSarn:aws:sns:us-west-2:522194210714:QueryMissAlarm.

Para obter mais informações, consulte [Configurar o Amazon Simple Notification Service](#) no Guia do usuário do Amazon CloudWatch.

2. Crie o alerta.

```
aws cloudwatch put-metric-alarm \
--alarm-name QueryCacheMissesAlarm \
--alarm-description "Alarm over query cache misses" \
--namespace AWS/DAX \
--metric-name QueryCacheMisses \
--dimensions Name=ClusterID,Value=myCluster \
--statistic Sum \
--threshold 8 \
--comparison-operator GreaterThanOrEqualToThreshold \
--period 60 \
--evaluation-periods 1 \
--alarm-actions arn:aws:sns:us-west-2:522194210714:QueryMissAlarm
```

3. Teste o alerta.

```
aws cloudwatch set-alarm-state --alarm-name QueryCacheMissesAlarm --state-reason
"initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name QueryCacheMissesAlarm --state-reason
"initializing" --state-value ALARM
```

Note

Você também pode aumentar ou diminuir o limite para que seja adequado ao seu aplicativo. Você também pode usar a [Matemática de métricas do CloudWatch](#) para definir uma métrica da taxa de ausências no cache e definir um alarme para essa métrica.

Como posso ser notificado se as solicitações causarem algum erro interno no cluster?

1. Crie um tópico do Amazon SNSarn:aws:sns:us-west-2:123456789012:notify-on-system-errors.

Para obter mais informações, consulte [Configurar o Amazon Simple Notification Service](#) no Guia do usuário do Amazon CloudWatch.

2. Crie o alerta.

```
aws cloudwatch put-metric-alarm \
    --alarm-name FaultRequestCountAlarm \
    --alarm-description "Alarm when a request causes an internal error" \
    --namespace AWS/DAX \
    --metric-name FaultRequestCount \
    --dimensions Name=ClusterID,Value=myCluster \
    --statistic Sum \
    --threshold 0 \
    --comparison-operator GreaterThanThreshold \
    --period 60 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:notify-on-system-errors
```

3. Teste o alerta.

```
aws cloudwatch set-alarm-state --alarm-name FaultRequestCountAlarm --state-reason
    "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name FaultRequestCountAlarm --state-reason
    "initializing" --state-value ALARM
```

Registrar operações do DAXAWS CloudTrail

O Amazon DynamoDB Accelerator (DAX) é integrado ao AWS CloudTrail, um serviço que fornece um registro de ações executadas por um usuário, uma função ou um AWSno DAX.

Para saber mais sobre o DAX e o CloudTrail, consulte a seção do DynamoDB Accelerator (DAX) em [Registro de operações do DynamoDB usando o AWS CloudTrail \(p. 954\)](#).

Instâncias intermitentes DAX T3/T2

O DAX permite que você escolha entre instâncias de desempenho fixo (como R4 e R5) e instâncias de desempenho intermitente (como T2 e T3). As instâncias com desempenho que podem ser ampliadas e que oferecem um nível básico de desempenho da CPU com a capacidade de intermitência acima da linha de base quando necessário.

O desempenho de linha de base e a capacidade de intermitência acima dele são governados por créditos de CPU. Instâncias de desempenho intermitente acumulam créditos de CPU continuamente, a uma taxa

determinada pelo tamanho da instância, quando a carga de trabalho está abaixo do limite de linha de base. Eses créditos podem ser consumidos quando a carga de trabalho aumenta. Um crédito de CPU oferece o desempenho de um núcleo de CPU completo por um minuto.

Muitas cargas de trabalho não precisam de níveis consistentemente altos de CPU, mas se beneficiam significativamente de ter acesso total a CPUs muito rápidas quando precisam delas. Instâncias de desempenho intermitente são projetadas especificamente para esses casos de uso. Se você precisar de alto desempenho de CPU consistentemente para seu banco de dados, recomendamos que você use instâncias de desempenho fixo.

Família de instâncias DAX T2

As instâncias DAX T2 são instâncias com desempenho de uso geral com capacidade de intermitência que oferecem um nível básico de desempenho de CPU com capacidade de intermitência acima da linha de base. As instâncias T2 são uma boa opção para cargas de trabalho de teste e desenvolvimento que precisam de previsibilidade de preços. As instâncias T2 do DAX são configuradas para o modo padrão, o que significa que se a instância estiver ficando sem créditos acumulados, a utilização da CPU será gradualmente reduzida para o nível da linha de base. Para obter mais informações sobre o modo padrão, consulte[Modo padrão de instâncias de desempenho com capacidade de intermitência](#)No Guia do usuário do Amazon EC2 para instâncias do Linux.

Família de instâncias do DAX T3

As instâncias DAX T3 são o tipo de instância de uso geral com capacidade de intermitência de última geração, oferecendo um nível básico de desempenho de CPU com capacidade de intermitência para uso de CPU a qualquer momento e pelo tempo necessário. As instâncias T3 oferecem recursos equilibrados de computação, memória e rede e são ideais para cargas de trabalho com uso moderado de CPU que experimentam picos temporários em uso. As instâncias DAX T3 são configuradas para o modo ilimitado, o que significa que elas podem explodir além da linha de base em uma janela de 24 horas por um custo adicional. Para obter mais informações sobre o modo ilimitado, consulte[Modo ilimitado de instâncias de desempenho com capacidade de intermitência](#)No Guia do usuário do Amazon EC2 para instâncias do Linux.

As instâncias DAX T3 podem sustentar alto desempenho da CPU enquanto uma carga de trabalho o exigir. Para a maioria das cargas de trabalho de uso geral, as instâncias T3 oferecem um desempenho amplo sem encargos adicionais. O preço por hora da instância T3 cobre automaticamente todos os picos provisórios em uso quando a utilização média de CPU de uma instância T3 é igual ou menor que a linha de base durante uma janela de 24 horas.

Por exemplo, `mdax.t3.small`recebe créditos continuamente a uma taxa de 24 créditos de CPU por hora. Esse recurso fornece desempenho de linha de base equivalente a 20% de um núcleo de CPU (20% × 60 minutos = 12 minutos). Se a instância não usar os créditos recebidos, eles serão armazenados em seu saldo de crédito de CPU até um máximo de 576 créditos de CPU. Quando `ot3.small`precisa explodir para mais de 20% de um núcleo, ela tira de seu saldo de crédito da CPU para lidar com esse aumento automaticamente.

Embora as instâncias DAX T2 estejam restritas ao desempenho da linha de base, uma vez que o saldo de crédito da CPU é retirado para zero, as instâncias do DAX T3 podem explodir acima da linha de base mesmo quando o saldo de crédito da CPU for zero. Para a grande maioria das cargas de trabalho, onde a utilização média de CPU é igual ou inferior ao desempenho da linha de base, o preço básico por hora para `t3.small`cobre todas as explosões de CPU. Se a instância for executada com uma utilização média de 25% da CPU (5% acima da linha de base) durante um período de 24 horas após o saldo de crédito da CPU ser retirado para zero, será cobrado um adicional de 11,52 centavos (9,6 centavos/VCPU-hora × 1 vCPU × 5% × 24 horas). Consulte[Definição de preço do Amazon DynamoDB](#)Para obter detalhes de preço.

Controle de acesso DAX

O DynamoDB Accelerator (DAX) foi projetado para funcionar em conjunto com o DynamoDB e adicionar uma camada de cache imperceptível aos seus aplicativos. No entanto, o DAX e o DynamoDB têm mecanismos de controle de acesso separados. Ambos os serviços usam AWS Identity and Access Management para implementar suas respectivas políticas de segurança, mas os modelos de segurança para DAX e DynamoDB são diferentes.

É altamente recomendável que você entenda ambos os modelos de segurança para que você possa implementar as medidas de segurança adequadas aos seus aplicativos que usam DAX.

Esta seção descreve os mecanismos de controle de acesso fornecidos pelo DAX e fornece exemplos de políticas do IAM que você pode adaptar às suas necessidades.

Com o DynamoDB, você pode criar políticas do IAM que limitam as ações que um usuário pode realizar em recursos individuais do DynamoDB. Por exemplo, você pode criar uma função de usuário que permite que o usuário execute apenas ações somente leitura em uma determinada tabela do DynamoDB. (Para obter mais informações, consulte [the section called “Identity and Access Management” \(p. 883\)](#).) Por comparação, o modelo de segurança DAX se concentra na segurança do cluster e na habilidade do cluster de realizar ações de API do DynamoDB em seu nome.

Warning

Se, no momento, você estiver usando funções e políticas de IAM para restringir o acesso aos dados de tabelas do DynamoDB, então, o uso do DAX pode subverter essas políticas. Por exemplo, um usuário poderia ter acesso a uma tabela do DynamoDB via DAX, mas não ter acesso explícito à mesma tabela acessando o DynamoDB diretamente. Para mais informações, consulte [the section called “Identity and Access Management” \(p. 883\)](#).

O DAX não impõe separação no nível do usuário em dados no DynamoDB. Em vez disso, os usuários herdam as permissões da política do IAM do cluster DAX quando acessam o cluster. Sendo assim, ao acessar as tabelas do DynamoDB via DAX, os únicos controles de acesso que estão em vigor são as permissões da política do IAM do cluster DAX. Nenhuma outra permissão é reconhecida.

Se você precisar de isolamento, recomendamos a criação de clusters DAX adicionais e a atribuição da política do IAM a cada cluster. Por exemplo, é possível criar vários clusters DAX e permitir que cada cluster acesse apenas uma única tabela.

Função de serviço do IAM para DAX

Ao criar um cluster DAX, você deve associar o cluster a uma função do IAM. Isso é conhecido como função de serviço do cluster.

Suponha que você queria criar um novo cluster DAX chamado `daxcluster01`. Você poderia criar uma função de serviço chamada `DAXServiceRole` e associar a função ao `DAXCluster01`. A política para `DAXServiceRole` definiria as ações do DynamoDB que o `daxcluster01` poderia realizar, em nome dos usuários que interagem com o `daxcluster01`.

Ao criar uma função de serviço, você deve especificar uma relação de confiança entre `DAXServiceRole` e o serviço DAX em si. Uma relação de confiança determina quais entidades podem assumir uma função e usar suas permissões. A seguir há um exemplo de documento de relação de confiança de `DAXServiceRole`:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "dax.amazonaws.com"  
            }  
        }  
    ]  
}
```

```
        },
        "Action": "sts:AssumeRole"
    ]
}
```

Essa relação de confiança permite que um cluster DAX assumaDaxServiceRole execute chamadas de API do DynamoDB em seu nome.

As ações da API do DynamoDB permitidas são descritas em um documento da política do IAM, que você anexa à função deDaxServiceRole. Veja abaixo um exemplo de documento de política.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DaxAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "dynamodb:DescribeTable",
                "dynamodb:PutItem",
                "dynamodb:GetItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:ConditionCheckItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
            ]
        }
    ]
}
```

Essa política permite que o DAX execute as ações da API do DynamoDB necessárias em uma tabela do DynamoDB. Odynamodb : DescribeTableA ação é necessária para o DAX manter metadados sobre a tabela e as outras são ações de leitura e gravação executadas em itens na tabela. A tabela chamada Books está na região us-west-2 e pertence ao ID 123456789012 da conta da AWS.

Política do IAM para permitir o acesso ao cluster DAX

Depois de criar um cluster DAX, será necessário conceder permissões a um usuário do IAM, de forma que o usuário possa acessar o cluster DAX.

Por exemplo, suponha que você queira conceder acesso a uma conta IAM chamada Alice. Você primeiro criaria uma política do IAM (aliceAccessPolicy) que define os clusters DAX e as ações da API DAX do que o destinatário pode acessar. Em seguida, você deveria conceder acesso anexando essa política à usuária Alice.

O documento de política a seguir oferece acesso total ao destinatário no DAXCluster01.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dax:*"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/DAXCluster01"
            ]
        }
    ]
}
```

```
        "Resource": [
            "arn:aws:dax:us-west-2:123456789012:cache/DAXCluster01"
        ]
    }
}
```

O documento de política permite o acesso ao cluster DAX, mas não concede quaisquer permissões do DynamoDB. (As permissões do DynamoDB são atribuídas pela função de serviço do DAX.)

Para a usuária Alice, você deve primeiro criar `AliceAccessPolicy` com o documento de política mostrado anteriormente. Em seguida, você anexaria a política à Alice.

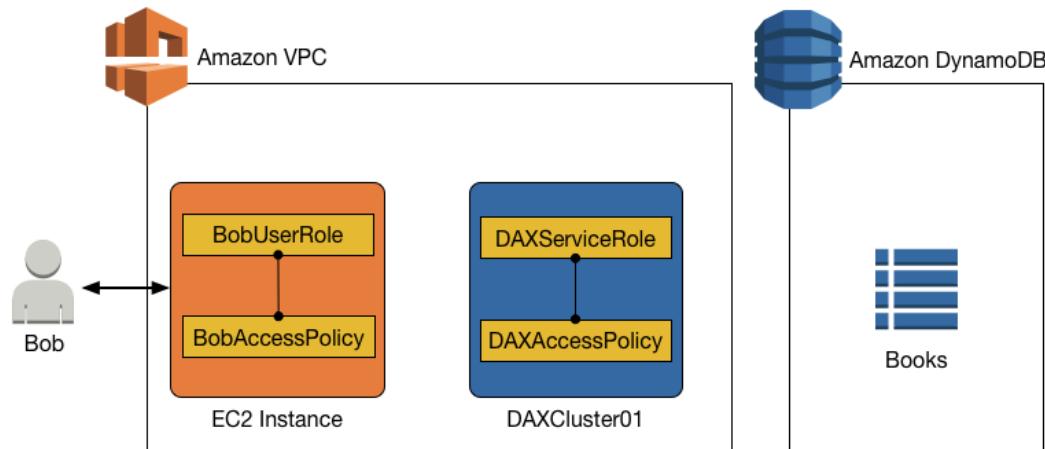
Note

Em vez de anexar a política a um usuário do IAM, você poderia anexá-la a uma função do IAM. Desse modo, todos os usuários que assumem essa função teriam as permissões que você definiu na política.

A política de usuário, junto com a função de serviço DAX, determina os recursos e as ações de API do DynamoDB que o destinatário pode acessar via DAX.

Estudo de caso: Acesso ao DynamoDB e ao DAX

O cenário a seguir pode ajudar a conhecer melhor as políticas do IAM para uso com o DAX. (Vamos falar sobre este cenário em todo o resto desta seção.) O diagrama a seguir mostra uma visão geral de alto nível do cenário.



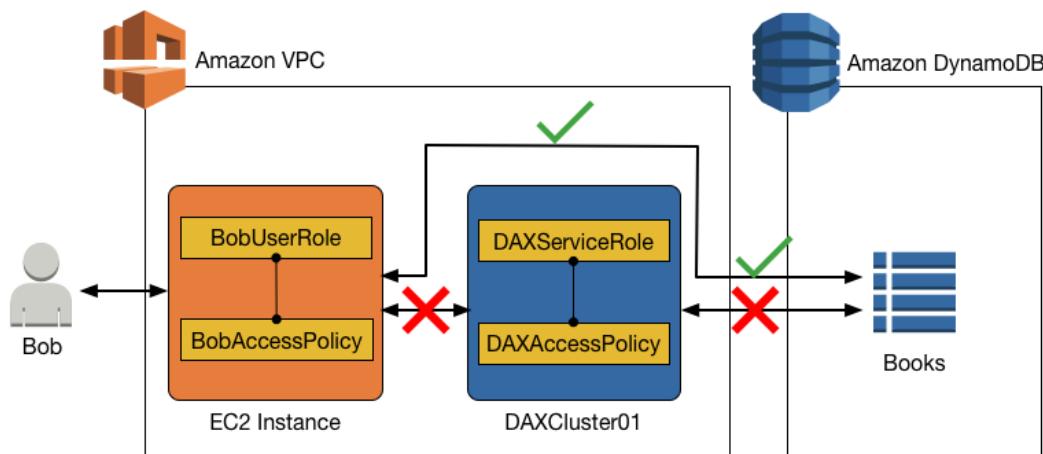
Nesse cenário, há as seguintes entidades:

- Um usuário do IAM (Bob).
- Uma função do IAM (`BobUserRole`). Bob assume essa função no tempo de execução.
- Uma política do IAM (`BobAccessPolicy`). Essa política é anexada a `BobUserRole`. `BobAccessPolicy` define os recursos do DynamoDB e DAX que `BobUserRole` tem permissão para acessar o.
- Um cluster DAX (`DAXCluster01`).
- Uma função de serviço do IAM (`DAXServiceRole`). Essa função permite `DAXCluster01` Para acessar o DynamoDB.

- Uma política do IAM (`DAXAccessPolicy`). Essa política é anexada a `DAXServiceRole.DAXAccessPolicy` define as APIs e os recursos do DynamoDB que `DAXCluster01` tem permissão para acessar o.
- Uma tabela do DynamoDB (`Books`).

A combinação de elementos de política no `BobAccessPolicy` e `DAXAccessPolicy` determinam o que Bob pode fazer com a tabela `Books`. Por exemplo, Bob pode ser capaz de acessar `Books` diretamente (usando o endpoint do DynamoDB), indiretamente (usando o cluster DAX), ou ambos. Bob talvez também possa ler os dados de `Books`, gravar dados em `Books`, ou as duas coisas.

Acesso ao DynamoDB, mas sem acesso com DAX



É possível permitir acesso direto a uma tabela do DynamoDB e, ao mesmo tempo, evitar o acesso indireto usando um cluster DAX. Para acesso direto ao DynamoDB, as permissões para `BobUserRole` são determinados por `BobAccessPolicy` (que está anexada à função).

Acesso somente leitura ao DynamoDB (apenas)

Bob pode acessar o DynamoDB com `BobUserRole`. A política do IAM anexada a essa função (`BobAccessPolicy`) determina as tabelas do DynamoDB que `BobUserRole` pode acessar, e quais APIs que `BobUserRole` pode invocar.

Considere o documento de política a seguir para `BobAccessPolicy`.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "DynamoDBAccessStmt",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:GetItem",  
                "dynamodb:BatchGetItem",  
                "dynamodb:Query",  
                "dynamodb:Scan"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"  
        }  
    ]  
}
```

```
        ]
    }
```

Quando este documento é anexado ao `BobAccessPolicy`, ele permite `BobUserRole` acessar o endpoint do DynamoDB e realizar operações somente leitura na `Books` Tabela do

O DAX não aparece nessa política, portanto, o acesso via DAX é negado.

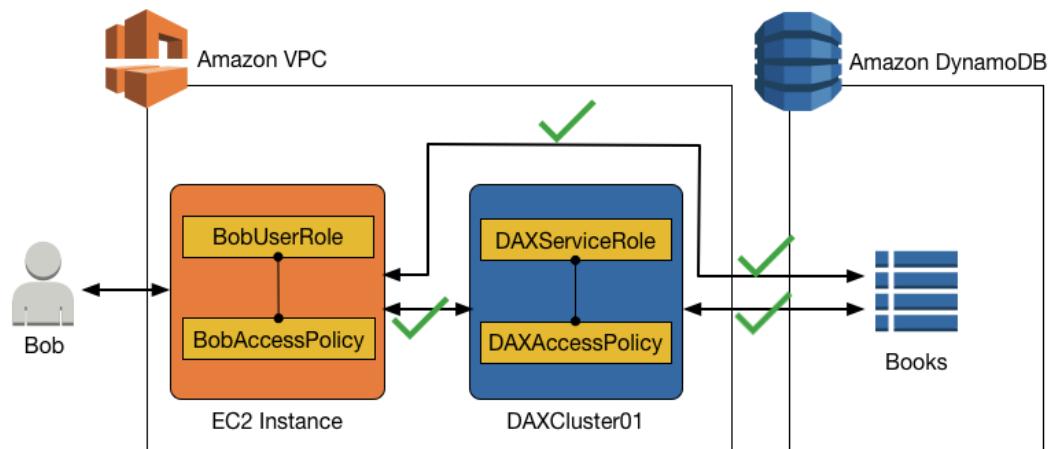
Acesso de leitura/gravação ao DynamoDB (apenas)

Se `BobUserRole` requer acesso de leitura/gravação ao DynamoDB, a política a seguir funcionaria.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DynamoDBAccessStmt",
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:BatchGetItem",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:ConditionCheckItem"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
        }
    ]
}
```

Novamente, o DAX não aparece nessa política, portanto, o acesso via DAX é negado.

Acesso ao DynamoDB e ao DAX



Para permitir acesso a um cluster DAX, você deve incluir ações específicas do DAX em uma política do IAM.

As ações específicas do DAX a seguir correspondem à suas equivalentes de nome semelhante na API do DynamoDB:

- `dax:GetItem`
- `dax:BatchGetItem`
- `dax:Query`
- `dax:Scan`
- `dax:PutItem`
- `dax:UpdateItem`
- `dax:DeleteItem`
- `dax:BatchWriteItem`
- `dax:ConditionCheckItem`

O mesmo é válido para a chave de condição `dax:EnclosingOperation`.

Acesso somente leitura ao DynamoDB e acesso somente leitura ao DAX

Suponha que Bob requeira acesso somente leitura à `Books`, do DynamoDB e do DAX. A política a seguir (anexada a `BobUserRole`) confere esse acesso.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DAXAccessStmt",
            "Effect": "Allow",
            "Action": [
                "dax:GetItem",
                "dax:BatchGetItem",
                "dax:Query",
                "dax:Scan"
            ],
            "Resource": "arn:aws:dax:us-west-2:123456789012:cache/DAXCluster01"
        },
        {
            "Sid": "DynamoDBAccessStmt",
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:BatchGetItem",
                "dynamodb:Query",
                "dynamodb:Scan"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
        }
    ]
}
```

A política tem uma declaração para acesso DAX (`DAXAccessStmt`) e outra instrução para o DynamoDBAccess (`DynamoDBAccessStmt`). Essas declarações permitem que Bob envie as solicitações `GetItem`, `BatchGetItem`, `Query` e `Scan` ao `DAXCluster01`.

No entanto, a função de serviço para `DAXCluster01` também precisaria de acesso somente leitura à `Books` no DynamoDB. A política do IAM a seguir anexada a `ADAXServiceRole`, cumpriria essa exigência.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "DynamoDBAccessStmt",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:GetItem",  
                "dynamodb:BatchGetItem",  
                "dynamodb:Query",  
                "dynamodb:Scan"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"  
        }  
    ]  
}
```

Acesso de leitura/gravação ao DynamoDB e somente de leitura com o DAX

Para uma determinada função de usuário, você pode fornecer acesso de leitura e gravação a uma tabela do DynamoDB e também permitir acesso somente leitura via DAX.

Para Bob, a política do IAM para `BobUserRole` precisaria permitir ações de leitura e gravação do DynamoDB no `Books`, além de oferecer suporte a ações somente leitura via `DAXCluster01`.

O seguinte exemplo de documento de política para `BobUserRole` conferiria esse acesso.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "DAXAccessStmt",  
            "Effect": "Allow",  
            "Action": [  
                "dax:GetItem",  
                "dax:BatchGetItem",  
                "dax:Query",  
                "dax:Scan"  
            ],  
            "Resource": "arn:aws:dax:us-west-2:123456789012:cache/DAXCluster01"  
        },  
        {  
            "Sid": "DynamoDBAccessStmt",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:GetItem",  
                "dynamodb:BatchGetItem",  
                "dynamodb:Query",  
                "dynamodb:Scan",  
                "dynamodb:PutItem",  
                "dynamodb:UpdateItem",  
                "dynamodb:DeleteItem",  
                "dynamodb:BatchWriteItem",  
                "dynamodb:DescribeTable",  
                "dynamodb:ConditionCheckItem"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"  
        }  
    ]  
}
```

```
        ]
    }
```

Além disso, o DAXServiceRole exigirá uma política do IAM que permitisse ao DAXCluster01 executar ações somente leitura na Books Tabela do

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DynamoDBAccessStmt",
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:DescribeTable"
      ],
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
    }
  ]
}
```

Acesso de leitura/gravação ao DynamoDB e acesso de leitura/gravação ao DAX

Agora, suponha que Bob precisasse de acesso de leitura/gravação ao Books, diretamente do DynamoDB ou indiretamente do DAXCluster01. O seguinte documento de política anexado a BobAccessPolicy confere esse acesso.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DAXAccessStmt",
      "Effect": "Allow",
      "Action": [
        "dax:GetItem",
        "dax:BatchGetItem",
        "dax:Query",
        "dax:Scan",
        "dax:PutItem",
        "dax:UpdateItem",
        "dax:DeleteItem",
        "dax:BatchWriteItem",
        "dax:ConditionCheckItem"
      ],
      "Resource": "arn:aws:dax:us-west-2:123456789012:cache/DAXCluster01"
    },
    {
      "Sid": "DynamoDBAccessStmt",
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem"
      ],
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
    }
  ]
}
```

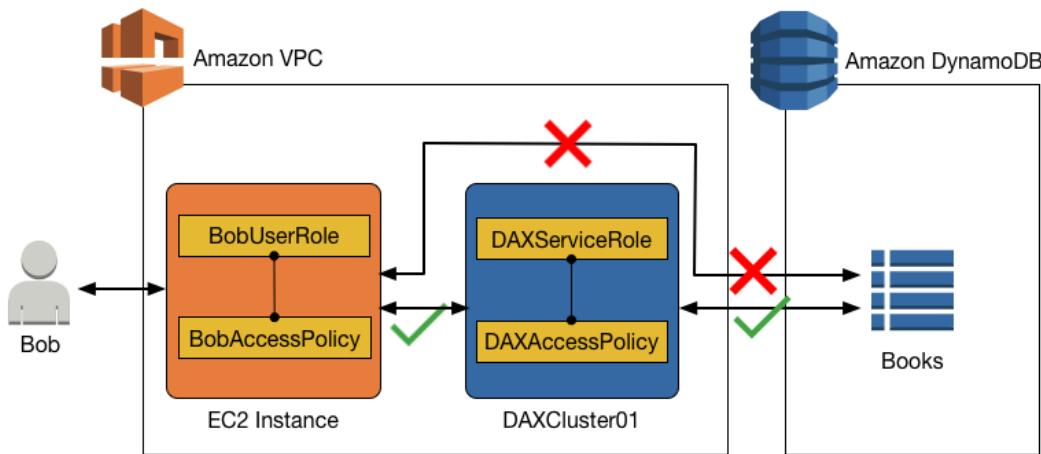
```
        "dynamodb>DeleteItem",
        "dynamodb>BatchWriteItem",
        "dynamodb>DescribeTable",
        "dynamodb>ConditionCheckItem"
    ],
    "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
}
}
```

Além disso, o DAXServiceRole exigiria uma política do IAM que permitisse o DAXCluster01 para executar ações de leitura/gravação na Books Tabela do

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DynamoDBAccessStmt",
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:BatchGetItem",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb>DeleteItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:DescribeTable"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
        }
    ]
}
```

Acesso ao DynamoDB via DAX, mas sem acesso direto ao DynamoDB

Nesse cenário, Bob pode acessar a Books Tabela via DAX, mas ele não tem acesso direto ao Books no DynamoDB. Sendo assim, quando Bob recebe acesso ao DAX, ele também obtém acesso a uma tabela do DynamoDB que ele de outra forma não poderia acessar. Ao configurar uma política do IAM para a função de serviço DAX, lembre-se de que qualquer usuário que receba acesso ao cluster DAX por meio da política de acesso do usuário tem acesso às tabelas especificadas nessa política. Nesse caso, o BobAccessPolicy ganha acesso às tabelas especificadas na DAXAccessPolicy.



Se, no momento, você estiver usando funções e políticas de IAM para restringir o acesso aos dados e às tabelas do DynamoDB, o uso do DAX pode subverter essas políticas. Na política a seguir, Bob tem acesso a uma tabela do DynamoDB via DAX, mas não tem acesso direto explícito à mesma tabela no DynamoDB.

O documento de política a seguir (`BobAccessPolicy`) anexado a `BobUserRole` conferiria esse acesso.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DAXAccessStmt",
            "Effect": "Allow",
            "Action": [
                "dax:GetItem",
                "dax:BatchGetItem",
                "dax:Query",
                "dax:Scan",
                "dax:PutItem",
                "dax:UpdateItem",
                "dax:DeleteItem",
                "dax:BatchWriteItem",
                "dax:ConditionCheckItem"
            ],
            "Resource": "arn:aws:dax:us-west-2:123456789012:cache/DAXCluster01"
        }
    ]
}
```

Nessa política de acesso, não há permissões para acessar o DynamoDB diretamente.

junto com `BobAccessPolicy`, os seguintes exemplos de `DAXAccessPolicy` dão `BobUserRole` acesso à tabela do DynamoDB `Books` mesmo que `BobUserRole` não pode acessar diretamente o `Books` tabela do

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DynamoDBAccessStmt",
            "Effect": "Allow",
            "Action": [
                "dynamodb:DescribeTable"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
        }
    ]
}
```

```
    "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem",
        "dynamodb:DescribeTable",
        "dynamodb:ConditionCheckItem"
    ],
    "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
}
]
```

Como o exemplo mostra, ao configurar o controle de acesso para a política de acesso do usuário e a política de acesso do cluster DAX, você tem que compreender totalmente o acesso de ponta a ponta para garantir que o princípio de privilégio mínimo seja observado. Garanta também que dar acesso a um cluster DAX a um usuário não subverte as políticas de controle de acesso estabelecidas anteriormente.

Criptografia DAX em repouso

A criptografia do Amazon DynamoDB Accelerator (DAX) em repouso fornece uma camada adicional de proteção de dados, ajudando a proteger os dados contra acesso não autorizado ao armazenamento subjacente. O uso de criptografia em repouso para proteção de dados pode ser requerida por políticas organizacionais, regulamentações setoriais ou governamentais e exigências de conformidade. Você pode usar criptografia para aumentar a segurança dos dados dos aplicativos que são implantados na nuvem.

Com a criptografia em repouso, os dados persistentes do DAX no disco são criptografados usando Advanced Encryption Standard de 256 bits, também conhecida como AES-256. O DAX grava dados ao disco como parte das alterações de propagação do nó primário para as réplicas de leitura.

A criptografia DAX em repouso se integra automaticamente ao AWS Key Management Service (AWS KMS) para gerenciamento da chave única de serviço padrão usada para criptografar seus clusters. Se ao criar um cluster DAX criptografado do não houver uma chave de serviço padrão, o AWS KMS cria automaticamente um novo AWS KMS chave gerenciada para você. Essa chave é usada com clusters criptografados que são criados no futuro. O AWS KMS integra hardware e software seguros e altamente disponíveis para oferecer um sistema de gerenciamento de chaves escalonado para a nuvem.

Após a criptografia dos seus dados, o DAX lida com a descriptografia dos seus dados de forma transparente com um mínimo impacto sobre o desempenho. Você não precisa modificar seus aplicativos para usar a criptografia.

Note

DAX não chama AWS KMS para cada operação DAX. O DAX somente usa a chave no lançamento do cluster. Mesmo que o acesso seja revogado, o DAX ainda pode acessar os dados até que o cluster seja desativado. As chaves AWS KMS especificadas pelo cliente não são aceitas.

A criptografia DAX em repouso está disponível para os tipos de nó de cluster:

Família	Node Type
Otimizado para memória (R4 e R5)	dax.r4.large
	dax.r4.xlarge

Família	Node Type
	dax.r4.2xlarge
	dax.r4.4xlarge
	dax.r4.8xlarge
	dax.r4.16xlarge
	dax.r5.large
	dax.r5.xlarge
	dax.r5.2xlarge
	dax.r5.4xlarge
	dax.r5.8xlarge
	dax.r5.12xlarge
	dax.r5.16xlarge
	dax.r5.24xlarge
Uso geral (T2)	dax.t2.small
	dax.t2.medium

Important

A criptografia DAX em repouso do não é compatível com dax.r3.* tipos de nó.

Não é possível ativar ou desativar a criptografia em repouso após a criação de um cluster. Você deve recriar o cluster para ativar a criptografia em repouso caso não tenha sido ativada na criação.

A criptografia DAX em repouso é fornecida sem custo adicional (AWS KMS cobranças de uso da chave de criptografia se aplicam) Para obter mais informações sobre definição de preços, consulte [Definição de preço do Amazon DynamoDB](#).

Habilitar a criptografia em repouso usando o AWS Management Console

Siga estas etapas para habilitar a criptografia DAX em repouso em uma tabela usando o console.

Para habilitar a criptografia DAX em repouso do

1. Faça login no AWS Management Console e abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação no lado esquerdo do console, em DAX, selecione Clusters.
3. Selecione Create cluster (Criar cluster).
4. Em Cluster name (Nome do cluster), insira um nome curto para o seu cluster. Selecione o node type (tipo de nó) para todos os nós no cluster e, para o tamanho do cluster, use 3 nós.
5. Em Encryption (Criptografia), selecione Enable encryption (Habilitar criptografia).

Encryption **Enable encryption**

You may enable encryption for your DAX cluster to help protect data at rest. [Learn more](#)

6. Após selecionar a função do IAM, o grupo de sub-rede, os grupos de segurança e as configurações do cluster, selecione Launch cluster (Iniciar cluster).

Para confirmar se o cluster está criptografado, verifique os detalhes dele no painel Clusters. O status da criptografia deve ser ENABLED (ATIVADA).

Criptografia do DAX em trânsito

O Amazon DynamoDB Accelerator (DAX) oferece suporte à criptografia em trânsito de dados entre seu aplicativo e seu cluster DAX, permitindo que você use o DAX em aplicativos com requisitos rigorosos de criptografia.

Independentemente de você escolher ou não a criptografia em trânsito, o tráfego entre o aplicativo e o cluster DAX permanece na Amazon VPC. Esse tráfego é roteado para Elastic Network Interfaces com IPs privados em sua VPC que estão conectados aos nós do cluster. Com sua VPC como limite de confiança, você tem controle significativo sobre a segurança de seus dados por meio do uso de ferramentas padrão, como grupos de segurança, segmentação de sub-rede com ACLs de rede e rastreamento de fluxo de VPC. A criptografia DAX em trânsito aumenta esse nível de confidencialidade de linha de base, garantindo que todas as solicitações e respostas entre o aplicativo e o cluster sejam criptografadas por TLS (Transport Level Security, segurança de nível de transporte) e que as conexões com o cluster possam ser autenticadas pela verificação de um certificado x509 de cluster. Os dados gravados em disco pelo DAX também podem ser criptografados se você escolher [Criptografia em repouso \(p. 815\)](#). Ao criar o cluster do DAX.

Usar criptografia em trânsito com DAX é fácil. Basta selecionar esta opção ao criar um novo cluster e usar uma versão recente de qualquer um dos [Clientes do DAX \(p. 738\)](#). Em seu aplicativo. Os clusters que usam criptografia em trânsito não oferecem suporte a tráfego não criptografado, portanto, não há chance de configurar incorretamente seu aplicativo e ignorar a criptografia. O cliente DAX usará o certificado x509 do cluster para autenticar a identidade do cluster quando estabelecer conexões, garantindo que suas solicitações DAX vão para onde pretendido. Todos os métodos de criação de clusters DAX oferecem suporte à criptografia em trânsito: o AWS Management Console, AWS CLI, todos os SDKs e AWS CloudFormation.

A criptografia em trânsito não pode ser habilitada em um cluster DAX existente. Para usar criptografia em trânsito em um aplicativo DAX existente, crie um novo cluster com criptografia em trânsito habilitada, mude o tráfego do aplicativo para ele e exclua o cluster antigo.

Uso de funções do IAM vinculadas ao serviço para DAX

O Amazon DynamoDB Accelerator (DAX) usa o AWS Identity and Access Management (IAM) [funções vinculadas ao serviço](#). Uma função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao DAX. As funções vinculadas ao serviço são predefinidas pelo DAX e incluem todas as permissões que o serviço requer para chamar outras AWS Serviços da em seu nome.

Uma função vinculada ao serviço facilita a configuração do DAX, pois você não precisa adicionar as permissões necessárias manualmente. O DAX define as permissões das funções vinculadas a serviços e, exceto se definido de outra forma, somente o DAX pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões. Essa política de permissões não pode ser anexada a nenhuma outra entidade do IAM.

Você pode excluir as funções somente depois de primeiro excluir seus recursos relacionados. Isso protege seus recursos do DAX, pois você não pode remover inadvertidamente a permissão para acessar os recursos.

Para obter informações sobre outros serviços que oferecem suporte a funções vinculadas ao serviço, consulte [AWS Serviços compatíveis com o IAM](#) no Guia do usuário do IAM. Procure os serviços que têm Yes (Sim) na coluna Service-linked roles (Funções vinculadas ao serviço). Selecione o link Yes (Sim) para exibir a documentação da função vinculada ao serviço para esse serviço.

Tópicos

- [Permissões de função vinculada ao serviço para o DAX \(p. 818\)](#)
- [Criar uma função vinculada ao serviço para o DAX \(p. 819\)](#)
- [Editar uma função vinculada ao serviço do DAX \(p. 819\)](#)
- [Exclusão de uma função vinculada ao serviço do DAX \(p. 819\)](#)

Permissões de função vinculada ao serviço para o DAX

O DAX usa a função vinculada ao serviço chamada `AWSServiceRoleForDAX`. Essa função permite que o DAX chame serviços em nome do cluster DAX.

Important

O `AWSServiceRoleForDAX` facilita a configuração e a manutenção de um cluster do DAX do. Contudo, ainda é necessário conceder a cada cluster acesso ao DynamoDB para poder usá-lo. Para mais informações, consulte [Controle de acesso DAX \(p. 805\)](#).

A função vinculada ao serviço `AWSServiceRoleForDAX` confia nos seguintes serviços para assumir a função:

- `dax.amazonaws.com`

A política de permissões da função permite que o DAX conclua as seguintes ações nos recursos especificados:

- Ações em ec2:
 - `AuthorizeSecurityGroupIngress`
 - `CreateNetworkInterface`
 - `CreateSecurityGroup`
 - `DeleteNetworkInterface`
 - `DeleteSecurityGroup`
 - `DescribeAvailabilityZones`
 - `DescribeNetworkInterfaces`
 - `DescribeSecurityGroups`
 - `DescribeSubnets`
 - `DescribeVpcs`

- [ModifyNetworkInterfaceAttribute](#)
- [RevokeSecurityGroupIngress](#)

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Para obter mais informações, consulte [Permissões de função vinculada ao serviço](#) no Guia do usuário do IAM.

Para permitir que uma entidade do IAM crie funções vinculadas ao serviço `AWSServiceRoleForDAX`

Adicione a seguinte declaração de política às permissões dessa entidade do IAM.

```
{  
    "Effect": "Allow",  
    "Action": [  
        "iam:CreateServiceLinkedRole"  
    ],  
    "Resource": "*",  
    "Condition": {"StringLike": {"iam:AWSServiceName": "dax.amazonaws.com"}}  
}
```

Criar uma função vinculada ao serviço para o DAX

Você não precisa criar manualmente uma função vinculada ao serviço. Quando você cria um cluster, o DAX cria a função vinculada ao serviço para você.

Important

Se você estava usando o serviço do DAX antes de 28 de fevereiro de 2018, quando ele começou a oferecer suporte a funções vinculadas ao serviço, o DAX criou o `AWSServiceRoleForDAX` na sua conta. Para obter mais informações, consulte [Uma nova função apareceu na minha AWS Conta no Guia do usuário do IAM](#).

Se você excluir essa função vinculada ao serviço e precisar criá-la novamente, poderá usar esse mesmo processo para recriar a função em sua conta. Quando você cria uma instância ou um cluster, o DAX cria a função vinculada ao serviço para você novamente.

Editar uma função vinculada ao serviço do DAX

O DAX não permite editar o `AWSServiceRoleForDAX` função vinculada ao serviço do. Depois que criar uma função vinculada ao serviço, você não poderá alterar o nome da função, pois várias entidades podem fazer referência a ela. No entanto, você poderá editar a descrição da função usando o IAM. Para obter mais informações, consulte [Editar uma função vinculada ao serviço](#) no Guia do usuário do IAM.

Exclusão de uma função vinculada ao serviço do DAX

Se você não precisar mais usar um recurso ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. Contudo, você deve excluir todos os seus clusters do DAX para poder excluir a função vinculada ao serviço.

Limpar uma função vinculada ao serviço

Antes de você poder usar o IAM para excluir uma função vinculada ao serviço, você deve primeiro confirmar que a função não tem sessões ativas e remover quaisquer recursos usados pela função.

Para verificar se a função vinculada ao serviço tem uma sessão ativa no console do IAM

1. Faça login no AWS Management Console e abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. No painel de navegação do console do IAM, escolha Roles. Em seguida, selecione o nome (não a caixa de seleção) do AWS Service Role para a função.
3. Na página Resumo para a função selecionada, escolha a guia Consultor de Acesso.
4. Na guia Consultor de Acesso, revise a atividade recente para a função vinculada ao serviço.

Note

Se você não tiver certeza se o DAX está usando o AWS Service Role para DAX, você pode tentar excluir a função. Se o serviço estiver usando a função, a exclusão falhará, e você poderá visualizar as regiões em que a função está sendo usada. Se a função estiver sendo usada, você deverá excluir os clusters do DAX para poder excluir a função. Você não pode revogar a sessão de uma função vinculada ao serviço.

Se desejar remover o AWS Service Role para DAX, você deve primeiro excluir todos os seus clusters do DAX.

Exclusão de todos os clusters do DAX

Use um destes procedimentos para excluir cada um de seus clusters do DAX.

Para excluir um cluster do DAX (console)

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, em DAX, escolha Clusters.
3. Escolha Ações e, em seguida, escolha Excluir.
4. Na caixa de diálogo Delete cluster confirmation (Confirmar exclusão de cluster), escolha Delete (Excluir).

Para excluir um cluster do DAX (AWS CLI)

Consulte [delete-cluster](#) no AWS CLI Referência de comandos da.

Para excluir um cluster do DAX (API)

Consulte [DeleteCluster](#) no Referência de API do Amazon DynamoDB.

Excluir uma função vinculada a serviços

Como excluir manualmente a função vinculada ao serviço usando o IAM

Use o console, a CLI ou a API do IAM para excluir o AWS Service Role para DAX vinculado ao serviço do. Para obter mais informações, consulte [Excluir uma função vinculada ao serviço](#) no Guia do usuário do IAM.

Acessando DAX entreAWSContas

Imagine que você tenha um cluster do DynamoDB Accelerator (DAX) em execução em uma AWS Conta A) e o cluster do DAX precisa ser acessível de uma instância do Amazon Elastic Compute Cloud (Amazon EC2) em outra instância da AWS Conta (conta B). Neste tutorial, você pode fazer isso ao executar uma instância

do EC2 na conta B com uma função do IAM da conta B. Em seguida, use credenciais de segurança temporárias da instância do EC2 para assumir uma função do IAM da conta A. Finalmente, você usa as credenciais de segurança temporárias do assumir a função do IAM na conta A para fazer chamadas de aplicativos através de uma conexão de emparelhamento da Amazon VPC para o cluster DAX na conta A. Para executar essas tarefas, você precisará de acesso administrativo em ambos AWS Contas.

Tópicos

- [Configurar o IAM \(p. 821\)](#)
- [Configuração de um VPC \(p. 823\)](#)
- [Modificar o cliente do DAX para permitir acesso entre contas \(p. 824\)](#)

Configurar o IAM

1. Crie um arquivo de texto chamado `AssumeDaxRoleTrust.json` com o seguinte conteúdo, que permite ao Amazon EC2 trabalhar em seu nome.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "ec2.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

2. Na conta B, crie uma função que o Amazon EC2 possa usar ao executar instâncias.

```
aws iam create-role \  
    --role-name AssumeDaxRole \  
    --assume-role-policy-document file://AssumeDaxRoleTrust.json
```

3. Crie um arquivo de texto chamado `AssumeDaxRolePolicy.json` com o seguinte conteúdo, que permite que o código executado na instância do EC2 na conta B assuma uma função do IAM na conta A. Substitua `accountA` com o ID real da conta A.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "sts:AssumeRole",  
            "Resource": "arn:aws:iam::accountA:role/DaxCrossAccountRole"  
        }  
    ]  
}
```

4. Adicione essa política à função que acabou de criar.

```
aws iam put-role-policy \  
    --role-name AssumeDaxRole \  
    --policy-name AssumeDaxRolePolicy \  
    --policy-document file://AssumeDaxRolePolicy.json
```

5. Crie um perfil de instância para permitir que as instâncias usem a função.

```
aws iam create-instance-profile \  
    --instance-profile-name AssumeDaxInstanceProfile
```

6. Associe a função ao perfil de instância.

```
aws iam add-role-to-instance-profile \  
    --instance-profile-name AssumeDaxInstanceProfile \  
    --role-name AssumeDaxRole
```

7. Crie um arquivo de texto chamado `DaxCrossAccountRoleTrust.json` com o seguinte conteúdo, que permite que a conta B assuma uma função da conta A. Substitua `accountB` pelo ID real da conta B.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::accountB:role/AssumeDaxRole"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

8. Na conta A, crie a função que a conta B poderá assumir.

```
aws iam create-role \  
    --role-name DaxCrossAccountRole \  
    --assume-role-policy-document file://DaxCrossAccountRoleTrust.json
```

9. Crie um arquivo de texto chamado `DaxCrossAccountPolicy.json` que permite o acesso ao cluster do DAX. Substituir `dax-cluster-arn` com o nome de recurso da Amazon (ARN) correto do seu cluster do DAX.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dax:GetItem",  
                "dax:BatchGetItem",  
                "dax:Query",  
                "dax:Scan",  
                "dax:PutItem",  
                "dax:UpdateItem",  
                "dax:DeleteItem",  
                "dax:BatchWriteItem",  
                "dax:ConditionCheckItem"  
            ],  
            "Resource": "dax-cluster-arn"  
        }  
    ]  
}
```

10. Na conta A, adicione a política à função.

```
aws iam put-role-policy \  
    --role-name DaxCrossAccountRole \  
    --policy-name DaxCrossAccountPolicy \  
    --policy-document file://DaxCrossAccountPolicy.json
```

```
--role-name DaxCrossAccountRole \
--policy-name DaxCrossAccountPolicy \
--policy-document file://DaxCrossAccountPolicy.json
```

Configuração de um VPC

1. Localize o grupo de sub-redes do cluster do DAX da conta A. Substituir *nome do cluster* pelo nome do cluster do DAX que a conta B deve acessar.

```
aws dax describe-clusters \
--cluster-name cluster-name \
--query 'Clusters[0].SubnetGroup'
```

2. Usando o *subnet-group*, encontre a VPC do cluster.

```
aws dax describe-subnet-groups \
--subnet-group-name subnet-group \
--query 'SubnetGroups[0].VpcId'
```

3. Usando o *vpc-id*, encontre o CIDR da VPC.

```
aws ec2 describe-vpcs \
--vpc vpc-id \
--query 'Vpcs[0].CidrBlock'
```

4. Na conta B, crie uma VPC usando um CIDR não sobreposto diferente daquele encontrado na etapa anterior. Em seguida, crie pelo menos uma sub-rede. Você pode usar o [assistente de criação da VPC](#) no AWS Management Console ou a [AWS CLI](#).
5. Na conta B, solicite uma conexão de emparelhamento com a VPC da conta A conforme descrito em [Criar e aceitar uma conexão de emparelhamento de VPC](#). Na conta A, aceite a conexão.
6. Na conta B, localize a nova tabela de roteamento da VPC. Substitua *vpc-id* pelo ID da VPC que você criou na conta B.

```
aws ec2 describe-route-tables \
--filters 'Name=vpc-id,Values=vpc-id' \
--query 'RouteTables[0].RouteTableId'
```

7. Adicione uma rota para enviar o tráfego destinado ao CIDR da conta A para a conexão de emparelhamento da VPC. Lembre-se de substituir cada *espaço reservado de entrada de usuário* pelos valores corretos para suas contas.

```
aws ec2 create-route \
--route-table-id accountB-route-table-id \
--destination-cidr accountA-vpc-cidr \
--vpc-peering-connection-id peering-connection-id
```

8. Na conta A, localize a tabela de rotas do cluster do DAX usando a *vpc-id* que você encontrou anteriormente.

```
aws ec2 describe-route-tables \
--filters 'Name=vpc-id, Values=accountA-vpc-id' \
--query 'RouteTables[0].RouteTableId'
```

9. Na conta A, adicione uma rota para enviar o tráfego destinado ao CIDR da conta B para a conexão de emparelhamento da VPC. Substitua cada *espaço reservado de entrada de usuário* pelos valores corretos para suas contas.

```
aws ec2 create-route \
--route-table-id accountA-route-table-id \
--destination-cidr accountB-vpc-cidr \
--vpc-peering-connection-id peering-connection-id
```

10. Na conta B, execute uma instância do EC2 na VPC que você criou anteriormente. Forneça o `AssumeDaxInstanceProfile`. Você pode usar o [assistente de inicialização](#) no AWS Management Console ou a [AWS CLI](#). Anote o grupo de segurança da instância.
11. Na conta A, localize o grupo de segurança usado pelo cluster do DAX. Lembre-se de substituir *nome do cluster* pelo nome do cluster do DAX.

```
aws dax describe-clusters \
--cluster-name cluster-name \
--query 'Clusters[0].SecurityGroups[0].SecurityGroupIdentifier'
```

12. Atualize o grupo de segurança do cluster do DAX para permitir o tráfego de entrada do grupo de segurança da instância do EC2 que você criou na conta B. Lembre-se de substituir a *Espaços reservados de entrada de usuário* com os valores corretos para suas contas.

```
aws ec2 authorize-security-group-ingress \
--group-id accountA-security-group-id \
--protocol tcp \
--port 8111 \
--source-group accountB-security-group-id \
--group-owner accountB-id
```

Nesse ponto, um aplicativo na instância do EC2 da conta B pode usar o perfil de instância para assumir `aarn:aws:iam::accountA-id:role/DaxCrossAccountRoleFunção` e use o cluster do DAX.

Modificar o cliente do DAX para permitir acesso entre contas

Note

As credenciais do AWS Security Token Service (AWS STS) são temporárias. Alguns clientes lidam com a atualização automaticamente, enquanto outros exigem lógica adicional para atualizar as credenciais. Recomendamos seguir as orientações da documentação apropriada.

Java

Esta seção ajuda você a modificar seu código de cliente do DAX do existente para permitir o acesso ao DAX entre contas. Caso ainda não tenha um código de cliente do DAX, você pode encontrar exemplos de código que funcionam na [Java e DAX \(p. 744\)](#) para "Hello, World!".

1. Adicione as seguintes importações.

```
import com.amazonaws.auth.STSSummary;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
```

2. Obtenha um provedor de credenciais do AWS STS e crie um objeto de cliente DAX. Lembre-se de substituir cada *espaço reservado de entrada de usuário* pelos valores corretos para suas contas.

```
AWSSecurityTokenService awsSecurityTokenService =
AWSSecurityTokenServiceClientBuilder
```

```
.standard()
.withRegion(region)
.build();

STSAssumeRoleSessionCredentialsProvider credentials = new
STSAssumeRoleSessionCredentialsProvider.Builder("arn:aws:iam::accountA:role/
RoleName", "TryDax")
.withStsClient(awsSecurityTokenService)
.build();

DynamoDB client = AmazonDaxClientBuilder.standard()
.withRegion(region)
.withEndpointConfiguration(dax_endpoint)
.withCredentials(credentials)
.build();
```

.NET

Esta seção ajuda você a modificar seu código de cliente do DAX do existente para permitir o acesso ao DAX entre contas. Caso ainda não tenha um código de cliente do DAX, você pode encontrar exemplos de código que funcionam na [.NET e DAX \(p. 753\)](#) para "Hello, World!".

1. Adicione o pacote [AWSSDK.SecurityToken](#) NuGet à solução.

```
<PackageReference Include="AWSSDK.SecurityToken" Version="latest version" />
```

2. Use os pacotes `SecurityToken` e `SecurityToken.Model`.

```
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
```

3. Obtenha credenciais temporárias do `AmazonSimpleTokenService` e crie um objeto `ClusterDaxClient`. Lembre-se de substituir cada *espaço reservado de entrada de usuário* pelos valores corretos para suas contas.

```
IAmazonSecurityTokenService sts = new AmazonSecurityTokenServiceClient();

var assumeRoleResponse = sts.AssumeRole(new AssumeRoleRequest
{
    RoleArn = "arn:aws:iam::accountA:role/RoleName",
    RoleSessionName = "TryDax"
});

Credentials credentials = assumeRoleResponse.Credentials;

var clientConfig = new DaxClientConfig(dax_endpoint, port)
{
    AwsCredentials = assumeRoleResponse.Credentials
};

var client = new ClusterDaxClient(clientConfig);
```

Go

Esta seção ajuda você a modificar seu código de cliente do DAX do existente para permitir o acesso ao DAX entre contas. Caso ainda não tenha um código de cliente do DAX, você pode encontrar [exemplos de código de trabalho no GitHub](#).

1. Importe os pacotes de AWS STS e sessão.

```
import (
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/sts"
    "github.com/aws/aws-sdk-go/aws/credentials/stscreds"
)
```

2. Obtenha credenciais temporárias do AmazonSimpleTokenService e crie um objeto de cliente DAX. Lembre-se de substituir cada *espaço reservado de entrada de usuário* pelos valores corretos para suas contas.

```
sess, err := session.NewSession(&aws.Config{
    Region: aws.String(region)},
)
if err != nil {
    return nil, err
}

stsClient := sts.New(sess)
arp := &stscreds.AssumeRoleProvider{
    Duration:      900 * time.Second,
    ExpiryWindow: 10 * time.Second,
    RoleARN:       "arn:aws:iam::accountA:role/role_name",
    Client:        stsClient,
    RoleSessionName: "session_name",
}cfg := dax.DefaultConfig()

cfg.HostPorts = []string{dax_endpoint}
cfg.Region = region
cfg.Credentials = credentials.NewCredentials(arp)
daxClient := dax.New(cfg)
```

Python

Esta seção ajuda você a modificar seu código de cliente do DAX do existente para permitir o acesso ao DAX entre contas. Caso ainda não tenha um código de cliente do DAX, você pode encontrar exemplos de código que funcionam na [Python e DAX \(p. 770\)](#) para "Hello, World!".

1. Importe o boto3.

```
import boto3
```

2. Obtenha credenciais temporárias do sts e crie um objeto do AmazonDaxClient. Lembre-se de substituir cada *espaço reservado de entrada de usuário* pelos valores corretos para suas contas.

```
sts = boto3.client('sts')
stsresponse =
sts.assume_role(RoleArn='arn:aws:iam::accountA:role/
RoleName',RoleSessionName='tryDax')
credentials = botocore.session.get_session()['Credentials']

dax = amazondax.AmazonDaxClient(session, region_name=region,
endpoints=[dax_endpoint], aws_access_key_id=credentials['AccessKeyId'],
aws_secret_access_key=credentials['SecretAccessKey'],
aws_session_token=credentials['SessionToken'])
client = dax
```

Node.js

Esta seção ajuda você a modificar seu código de cliente do DAX do existente para permitir o acesso ao DAX entre contas. Caso ainda não tenha um código de cliente do DAX, você pode encontrar exemplos de código que funcionam na [Node.js e DAX \(p. 762\)](#) para "Hello, World!". Lembre-se de substituir cada *espaço reservado de entrada de usuário* pelos valores corretos para suas contas.

```
const AmazonDaxClient = require('amazon-dax-client');
const AWS = require('aws-sdk');
const region = 'region';
const endpoints = [daxEndpoint1, ...];

const getCredentials = async() => {
    return new Promise((resolve, reject) => {
        const sts = new AWS.STS();
        const roleParams = {
            RoleArn: 'arn:aws:iam::accountA:role/RoleName',
            RoleSessionName: 'tryDax',
        };
        sts.assumeRole(roleParams, (err, session) => {
            if(err) {
                reject(err);
            } else {
                resolve({
                    accessKeyId: session.Credentials.AccessKeyId,
                    secretAccessKey: session.Credentials.SecretAccessKey,
                    sessionToken: session.Credentials.SessionToken,
                });
            }
        });
    });
};

const createDaxClient = async() => {
    const credentials = await getCredentials();
    const daxClient = new AmazonDaxClient({endpoints: endpoints, region: region,
        accessKeyId: credentials.accessKeyId, secretAccessKey: credentials.secretAccessKey,
        sessionToken: credentials.sessionToken});
    return new AWS.DynamoDB.DocumentClient({service: daxClient});
};

createDaxClient().then((client) => {
    client.get(...);
    ...
}).catch((error) => {
    console.log('Caught an error: ' + error);
});
```

Guia de dimensionamento de cluster do DAX

Este guia fornece conselhos para escolher um tamanho de cluster e um tipo de nó do Amazon DynamoDB Accelerator (DAX) para seu aplicativo. Essas instruções orientam você nas etapas de estimar do tráfego DAX do aplicativo do, selecionar uma configuração do cluster e testá-lo.

Se você já tem um cluster DAX do e deseja avaliar se ele tem o número e o tamanho apropriados de nós, consulte [Escalabilidade de um cluster DAX \(p. 784\)](#).

Tópicos

- [Overview \(p. 828\)](#)

- [Estimativa do tráfego \(p. 828\)](#)
- [Testes de carregamento \(p. 829\)](#)

Overview

É importante dimensionar o cluster DAX do adequadamente para sua carga de trabalho, quer você esteja criando um novo cluster ou mantendo um cluster existente. Conforme o tempo passa e a carga de trabalho do aplicativo é alterada, você deve revisar periodicamente suas decisões de escalabilidade para garantir que elas ainda são apropriadas.

O processo normalmente segue estas etapas:

1. Estimativa do tráfego. Nesta etapa, você faz previsões sobre o volume de tráfego que o aplicativo enviará ao DAX, a natureza do tráfego (operações de leitura versus gravação) e a taxa de acerto do cache esperada.
2. Testes de carga Nesta etapa, você cria um cluster e envia tráfego para ele espelhando suas estimativas da etapa anterior. Repita essa etapa até encontrar uma configuração de cluster adequada.
3. Monitoramento da produção. Enquanto seu aplicativo estiver usando DAX em produção, você deve[Monitore o cluster \(p. 788\)](#)Para validar continuamente se ele ainda está dimensionado corretamente conforme a carga de trabalho muda ao longo do tempo.

Estimativa do tráfego

Existem três fatores principais que caracterizam uma carga de trabalho DAX típica do:

- Taxa de acertos do cache
- [Unidades de capacidade de leitura \(p. 346\)](#) (RCUs - Read capacity units) por segundo
- [Unidades de capacidade de gravação \(p. 347\)](#) (WCUs - Write capacity units) por segundo

Estimativa da taxa de acertos do cache

Se você já tiver um cluster DAX, poderá usar o `ItemCacheHit`s/`ItemCacheMisses` [Métricas do Amazon CloudWatch \(p. 791\)](#)para determinar a taxa de acertos do cache. A taxa de acertos do cache é igual a $\text{ItemCacheHits} / (\text{ItemCacheHits} + \text{ItemCacheMisses})$. Se sua carga de trabalho incluir operações `Query` ou `Scan`, você também deverá examinar as métricas `QueryCacheHits`, `QueryCacheMisses`, `ScanCacheHit`s e `ScanCacheMisses`. As taxas de acerto do cache variam de um aplicativo para outro e são fortemente influenciadas pela configuração do Tempo para Vida (TL) do cluster. As taxas de acerto típicas de aplicativos que usam o DAX são de 85 a 95 por cento.

Estimativa de unidades de capacidade de leitura e gravação

Se você já tiver tabelas do DynamoDB para seu aplicativo, consulte `aConsumedReadCapacityUnit`s/`ConsumedWriteCapacityUnits` [Métricas do CloudWatch \(p. 791\)](#). Use a estatística `Sum` e divida pelo número de segundos no período.

Se você também já tiver um cluster do DAX, lembre-se DynamoDB que `oConsumedReadCapacityUnits`contabiliza apenas erros de cache. Portanto, para ter uma ideia das unidades de capacidade de leitura por segundo tratada pelo cluster DAX, divida o número pela taxa de erro do cache (ou seja, $1 - \text{taxa de acerto do cache}$).

Se você ainda não tem uma tabela do DynamoDB, consulte a documentação sobre[Unidades de capacidade de leitura \(p. 346\)](#) e[unidades de capacidade de gravação \(p. 347\)](#)Para estimar o tráfego com

base na taxa de solicitação estimada do aplicativo, nos itens acessados por solicitação e no tamanho do item.

Ao fazer estimativas de tráfego, planeje para crescimento futuro e picos esperados e inesperados para garantir que o cluster tenha espaço suficiente para aumentos do tráfego.

Testes de carregamento

A próxima etapa depois de estimar o tráfego é testar a configuração do cluster sob carga.

1. Para o teste de carga inicial, recomendamos que você comece com o tipo de nó `dax.r4.large`, o desempenho fixo de menor custo e o tipo de nó otimizado para memória.
2. Um cluster tolerante a falhas requer pelo menos três nós, distribuídos por três zonas de disponibilidade. Nesse caso, se uma zona de disponibilidade se tornar indisponível, o número efetivo de zonas de disponibilidade será reduzido em um terço. Para o teste de carga inicial, recomendamos começar com um cluster de dois nós, que simula a falha de uma zona de disponibilidade em um cluster de três nós.
3. Dirija o tráfego sustentado (conforme estimado na etapa anterior) para o cluster de teste durante o teste de carga.
4. Monitore o desempenho do cluster durante o teste de carga.

O ideal é que o perfil do tráfego que você dirige durante o teste de carga seja o mais semelhante possível ao tráfego real do aplicativo. Isso inclui a distribuição de operações (por exemplo, 70 por cento de `GetItem`, 25 por cento de `Query` e 5 por cento de `PutItem`), a taxa de solicitação para cada operação, o número de itens acessados por solicitação e a distribuição de tamanhos dos itens. Para obter uma taxa de acerto do cache semelhante à taxa esperada de acerto de cache do aplicativo, preste muita atenção à distribuição de chaves no tráfego de teste.

Note

Tenha cuidado ao carregar testes de tipos de nó T2 (`dax.t2.small` e `dax.t2.medium`). Os tipos de nó T2 fornecem [desempenho de CPU com capacidade de intermitência](#) que varia ao longo do tempo, dependendo do saldo de crédito da CPU do nó. Um cluster DAX do em execução em nós T2 pode parecer estar funcionando normalmente, mas se algum nó estiver com intermitência acima do[Desempenho de linha](#) de sua instância, o nó está gastando seu saldo de crédito de CPU acumulado. Quando o saldo de crédito está baixo, o [desempenho é gradualmente reduzido](#) para o nível de desempenho de linha de base.

[Monitore seu cluster DAX \(p. 788\)](#)Durante o teste de carga para determinar se o tipo de nó que você está usando para o teste de carga é o tipo de nó certo para você. Além disso, durante um teste de carga, você deve monitorar a taxa de solicitações e a taxa de acertos do cache para garantir que sua infraestrutura de teste esteja realmente direcionando a quantidade de tráfego planejada.

Se o teste de carga indicar que a configuração do cluster selecionada não pode sustentar a carga de trabalho do aplicativo, recomendamos[Alternar para um tipo de nó maior \(p. 785\)](#), especialmente se você perceber alta utilização da CPU no nó primário no cluster, altas taxas de remoção ou alta utilização da memória cache. Se as taxas de acerto forem consistentemente altas e a proporção do tráfego de leitura para gravação for alta, convém considerar a [adição de mais nós ao cluster \(p. 784\)](#). Consulte [Escalabilidade de um cluster DAX \(p. 784\)](#) para obter orientações adicionais sobre quando usar um tipo de nó maior (escalabilidade vertical) ou adicionar mais nós (escalabilidade horizontal).

Você deve repetir o teste de carga depois de fazer alterações na configuração do cluster.

Referência de API do.

Para obter mais informações sobre as APIs do Amazon DynamoDB Accelerator (DAX), consulte[Amazon DynamoDB Accelerator](#)Na Referência de API do Amazon DynamoDB.

NoSQL Workbench para o DynamoDB

O NoSQL Workbench para o Amazon DynamoDB é um aplicativo GUI cliente de plataforma cruzada para operações e desenvolvimento de bancos de dados modernos e está disponível para Windows, macOS e Linux. O NoSQL Workbench é uma ferramenta visual unificada que fornece recursos de modelagem de dados, visualização de dados e desenvolvimento de consulta para ajudá-lo a projetar, criar, consultar e gerenciar tabelas do DynamoDB.

Modelagem de dados

Com o NoSQL Workbench para o DynamoDB, você pode criar novos modelos de dados a partir de, ou projetar modelos com base em, modelos de dados existentes que satisfaçam os padrões de acesso a dados dos seus aplicativos. Pode também importar e exportar o modelo de dados designado no final do processo. Para obter mais informações, consulte [Criar modelos de dados com o NoSQL Workbench \(p. 831\)](#).

Visualização de dados

O visualizador de modelo de dados oferece uma tela na qual você pode mapear consultas e visualizar os padrões de acesso (facetas) do aplicativo sem precisar escrever código. Cada faceta corresponde a um padrão de acesso diferente no DynamoDB. É possível adicionar dados manualmente ao seu modelo de dados. Para obter mais informações, consulte [Visualizar padrões de acesso a dados \(p. 845\)](#).

Criação de operação

O NoSQL Workbench oferece uma avançada interface gráfica do usuário para você desenvolver e testar consultas. Você pode usar o criador de operações para visualizar, explorar e consultar conjuntos de dados. Pode também usar o criador de operações estruturadas para criar e executar operações de plano de dados. Ele oferece suporte para expressões de projeção e condição e permite que você gere código de exemplo em várias linguagens. Para obter mais informações, consulte [Explorar conjuntos de dados e criar operações com o NoSQL Workbench \(p. 853\)](#).

Tópicos

- [Fazer download do NoSQL Workbench \(p. 830\)](#)
- [Criar modelos de dados com o NoSQL Workbench \(p. 831\)](#)
- [Visualizar padrões de acesso a dados \(p. 845\)](#)
- [Explorar conjuntos de dados e criar operações com o NoSQL Workbench \(p. 853\)](#)
- [Exemplos de modelos de dados para o NoSQL Workbench \(p. 868\)](#)
- [Histórico de versões do NoSQL Workbench \(p. 871\)](#)

Fazer download do NoSQL Workbench

Siga estas instruções para fazer download do NoSQL Workbench Amazon DynamoDB.

Para fazer download do NoSQL Workbench e instalá-lo

1. Faça download da versão apropriada do NoSQL Workbench para seu sistema operacional.

Sistema operacional	Link para fazer download	Link para soma de verificação
macOS	Download para macOS	Soma de verificação
Windows	Download para Windows	Soma de verificação
Linux*	Download para Linux	Soma de verificação

* NoSQL Workbench oferece suporte para Ubuntu 12.04, Fedora 21 e Debian 8 ou quaisquer versões mais recentes destas distribuições do Linux.

2. Inicie o aplicativo que você baixou e siga as instruções na tela para instalar o NoSQL Workbench.

Criar modelos de dados com o NoSQL Workbench

Use a ferramenta de modelador de dados no NoSQL Workbench para Amazon DynamoDB para criar novos modelos de dados ou para criar modelos com base em modelos de dados existentes que satisfaçam os padrões de acesso a dados dos seus aplicativos. O modelador de dados inclui alguns modelos de dados de exemplo para ajudar você a começar.

Tópicos

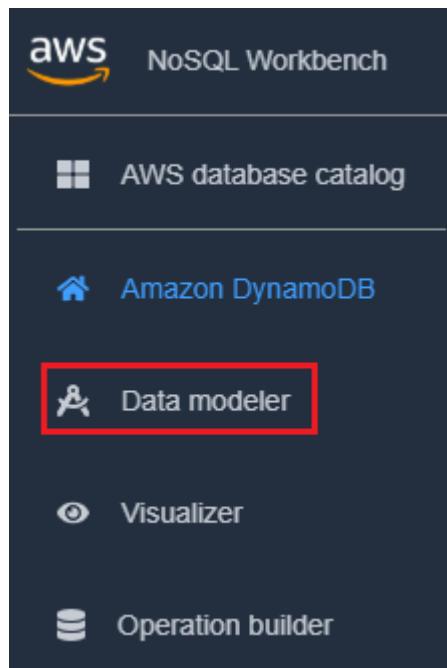
- [Criar um novo modelo de dados \(p. 831\)](#)
- [Importar um modelo de dados existente \(p. 838\)](#)
- [Exportar um modelo de dados \(p. 841\)](#)
- [Editar um modelo de dados existente \(p. 843\)](#)

Criar um novo modelo de dados

Siga estas etapas para criar um novo modelo de dados no Amazon DynamoDB usando o NoSQL Workbench.

Para criar um novo modelo de dados

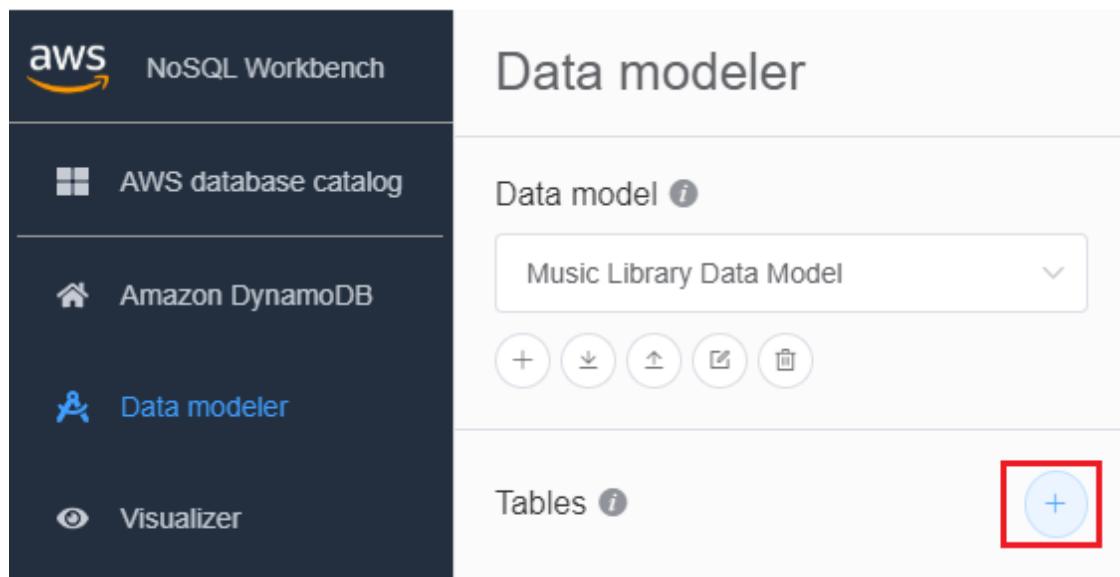
1. Abra o NoSQL Workbench e, no painel de navegação no lado esquerdo, escolha o ícone do Data modeler (Modelador de dados).



2. Escolha Create data model (Criar modelo de dados).

The screenshot shows the Data modeler screen. At the top, it says "Data modeler". Below that is a section labeled "Data model" with an "i" icon. To the right of this is a dropdown menu and a toolbar with several icons. One of the icons in the toolbar is a red-bordered "+" sign, which is highlighted with a red box.

3. Insira um nome, um autor e uma descrição para o modelo de dados e escolha Criar.
4. Escolha Add table (Adicionar tabela).



Para obter mais informações sobre tabelas, consulte [Trabalho com tabelas no DynamoDB](#).

5. Especifique o seguinte:

- Nome da tabela— Insira um nome exclusivo para a tabela.
- Chave de partição— Insira um nome de chave de partição e especifique seu tipo.
- Se desejar adicionar uma chave de classificação:
 1. Selecione Add sort key (Adicionar chave de classificação).
 2. Especifique o nome da chave de classificação e seu tipo.
 - 3.

* Table name ⓘ

Primary key attributes ⓘ

* Partition key String ⓘ

Add sort key ⓘ

6. Para adicionar outros atributos, faça o seguinte para cada atributo:

1. Escolha Add other attribute (Adicionar outro atributo).
2. Especifique o nome e o tipo do atributo.

Other attributes ⓘ

* Attribute name	FirstName	String	✖
* Attribute name	LastName	String	✖
* Attribute name	ManagerLoginAlias	String	✖
* Attribute name	Designation	String	✖
* Attribute name	Skills	String Set	✖
+ Add other attribute			

7. Adicionar uma faceta:

Note

Facetas representam diferentes padrões de acesso a dados de um aplicativo para o DynamoDB.

- Selecione Add facets (Adicionar facetas).
- Escolha Add facet (Adicionar faceta).

[+ Add other attribute](#)

[Add facets ⓘ](#)

ⓘ Facet is a virtual construct in NoSQL Workbench for Amazon DynamoDB, and not a functional construct in DynamoDB itself.

ⓘ No facet added

[+ Add facet](#)

- Especifique o seguinte:
 - O Facet name (Nome da faceta).
 - Um Partition key alias (Alias de chave de partição).
 - Um Sort key alias (Alias da chave de classificação).
 - Escolha os Other attributes (Outros atributos) que fazem parte desta faceta.

Escolha Add facet (Adicionar faceta).

Add facets [i](#)

[i](#) Facet is a virtual construct in NoSQL Workbench for Amazon DynamoDB, and not a functional construct in DynamoDB itself.

[i](#) No facet added

Add facet

* Facet name

* Partition key alias

* Sort key alias

Other attributes

[Cancel](#) [Add facet](#)



Repita a etapa 8 se quiser adicionar mais facetas.

8. Se desejar adicionar um índice secundário global, escolha Add global secondary index (Adicionar índice secundário global).

Especifique o Global secondary index name (Nome do índice secundário global), o atributo Partition key (Chave de partição) e o Projection type (Tipo de projeção).

Global secondary indexes

Global secondary [i](#)

index name

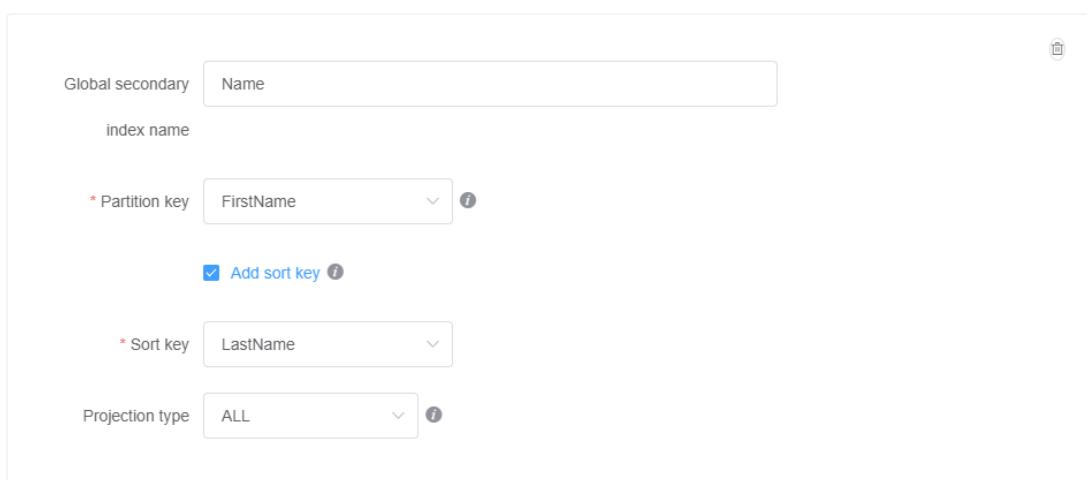
* Partition key [i](#)

Add sort key [i](#)

* Sort key [i](#)

Projection type [i](#)

[+ Add global secondary index](#)



Para obter mais informações sobre o trabalho com índices secundários globais no DynamoDB, consulte [índices secundários globais](#).

9. Por padrão, sua tabela usará o modo de capacidade provisionada com o dimensionamento automático ativado na capacidade de leitura e gravação. Se quiser alterar essas configurações, desmarque “Configurações padrão” em Configurações da capacidade.

Capacity settings ⓘ

Default settings

ⓘ The default settings use provisioned capacity mode with auto scaling enabled on both read and write capacity.

Selecione o modo de capacidade desejado, a capacidade de leitura e gravação e a função do IAM de dimensionamento automático (se aplicável).

Capacity settings

Default settings

 The default settings use provisioned capacity mode with auto scaling enabled on both read and write capacity.

Capacity mode

Provisioned 

On-demand 

Provisioned read capacity

Enable auto scaling 

* Minimum capacity units

1

* Maximum capacity units

10

* Target utilization (%)

70

Provisioned write capacity

Enable auto scaling 

* Minimum capacity units

1

* Maximum capacity units

10

* Target utilization (%)

70

Auto scaling IAM role

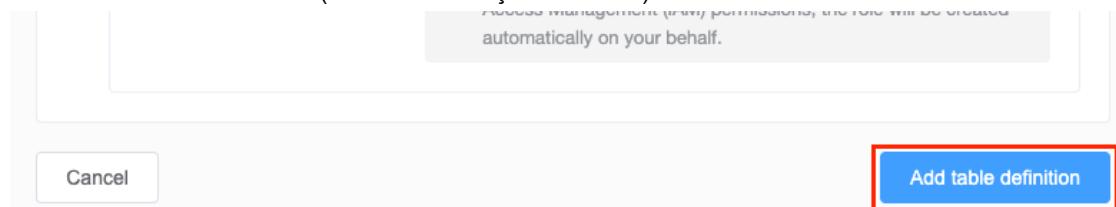
* Role version  Versão da API 2012-08-10 orApplicationAutc

837

This service-linked role is required to enable auto scaling through NoSQL Workbench. If the role does not already exist in your account and you have the required AWS Identity and

Para obter mais informações sobre as configurações de capacidade do DynamoDB, consulte[Modo de capacidade de leitura/gravação \(p. 18\)](#).

10. Escolha Add table definition (Adicionar definição de tabela).



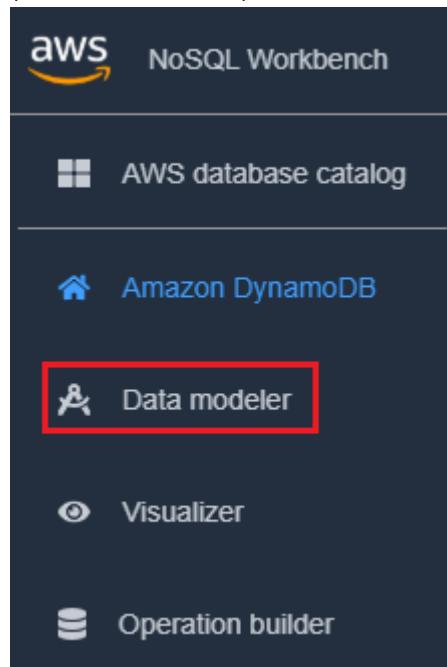
Para obter mais informações sobre o[CreateTable](#)Operação da API, consulte[CreateTable](#)noReferência de API do Amazon DynamoDB.

Importar um modelo de dados existente

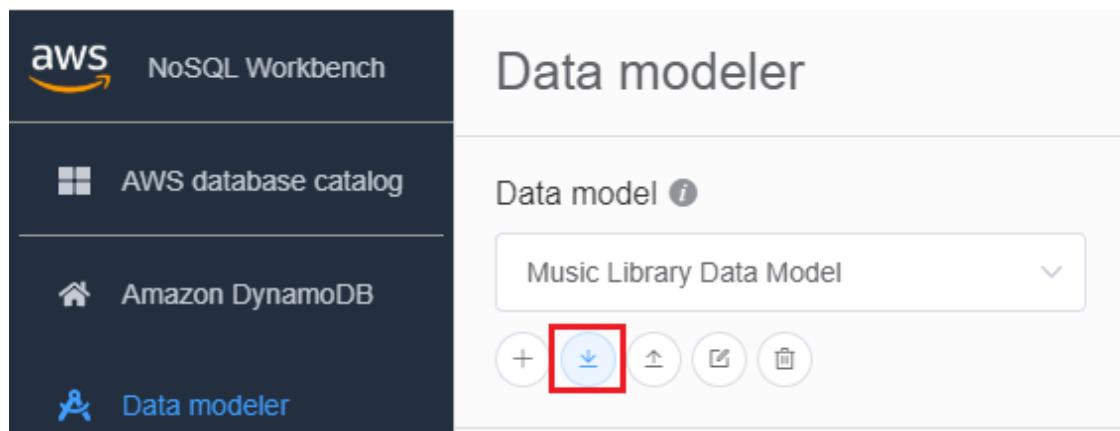
Você pode usar o NoSQL Workbench para Amazon DynamoDB para criar um modelo de dados importando e modificando um modelo existente. Você pode importar modelos de dados no formato de modelo NoSQL Workbench ou no[AWS CloudFormationFormato de modelo JSON](#).

Para importar um modelo de dados

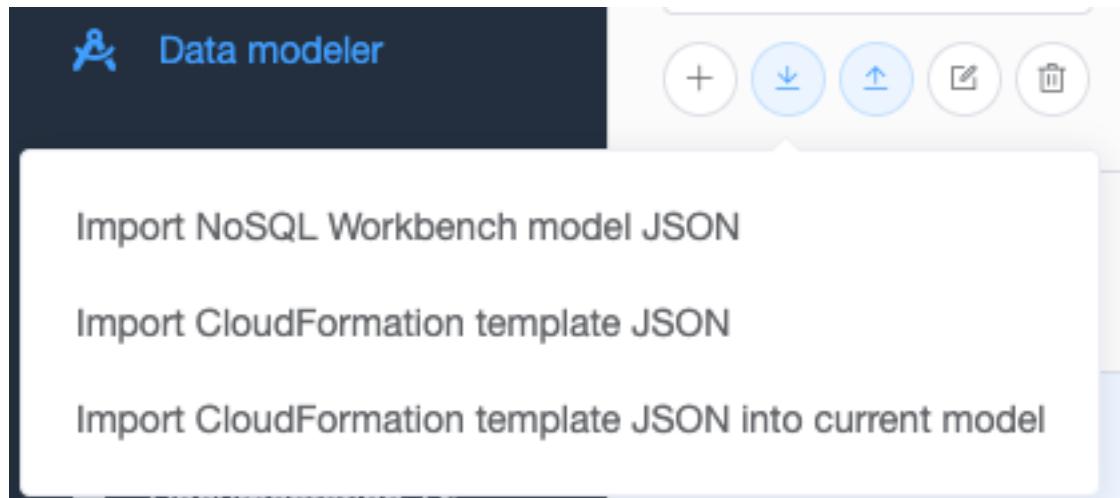
1. No NoSQL Workbench, no painel de navegação no lado esquerdo, escolha o ícone Data modeler (Modelador de dados).



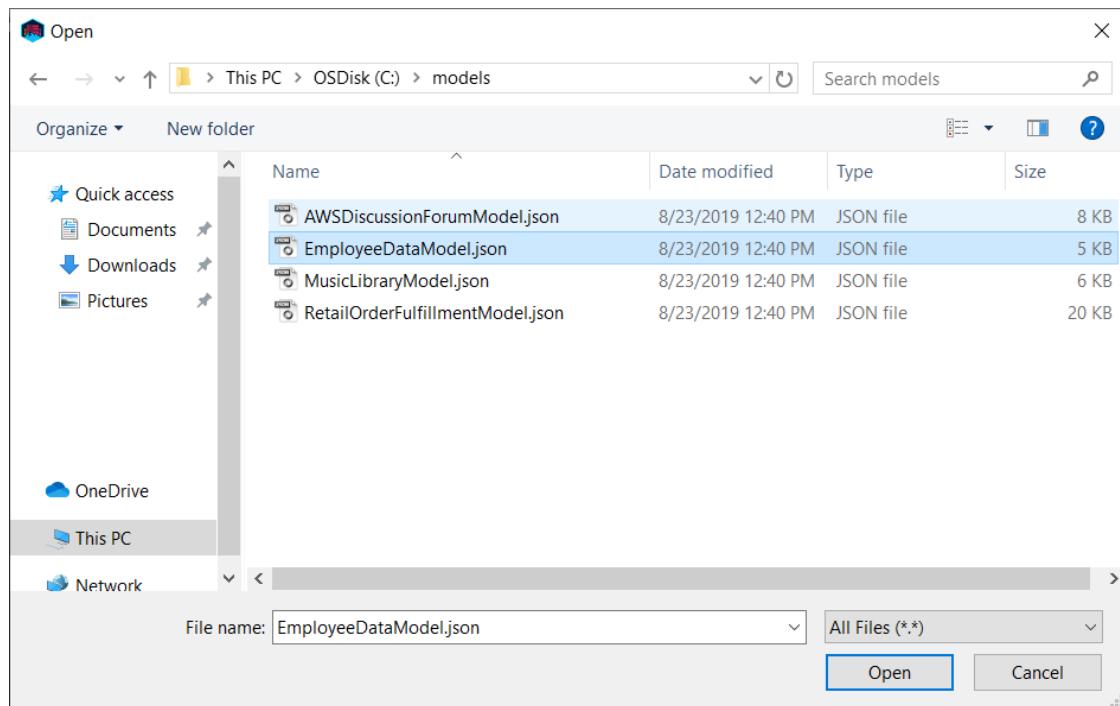
2. Passe o mouse sobre Importar modelo de dados.



Na lista suspensa, escolha se o modelo que deseja importar está no formato de modelo NoSQL Workbench ou no formato de modelo JSON do CloudFormation. Se você tiver um modelo de dados existente aberto no NoSQL Workbench, terá a opção de importar um modelo do CloudFormation para o modelo atual.



3. Escolha um modelo para importar.



4. Se o modelo que você está importando estiver no formato de modelo do CloudFormation, você verá uma lista de tabelas a serem importadas e terá a oportunidade de especificar um nome, autor e descrição do modelo de dados.

Create data model for Amazon DynamoDB

Only CloudFormation resources related to DynamoDB: tables and any related application auto scaling, **i** will be imported. Some fields within these resources are not supported by NoSQL Workbench and will also not be imported, including LocalSecondaryIndexes, RoleARN, and PolicyName.

Successfully imported tables (1)

 Employee

Data model information

* Name

Enter data model name

Author

Enter author name

Description

Describe this data model

[Cancel](#)

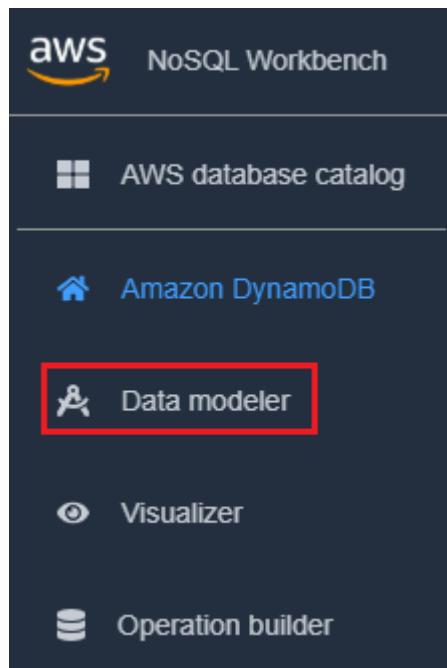
[Create](#)

Exportar um modelo de dados

Depois de criar um modelo de dados usando o NoSQL Workbench para Amazon DynamoDB, você poderá salvar e exportar o modelo no NoSQL Workbench do [AWS CloudFormation](#) formato de modelo JSON.

Para exportar um modelo de dados

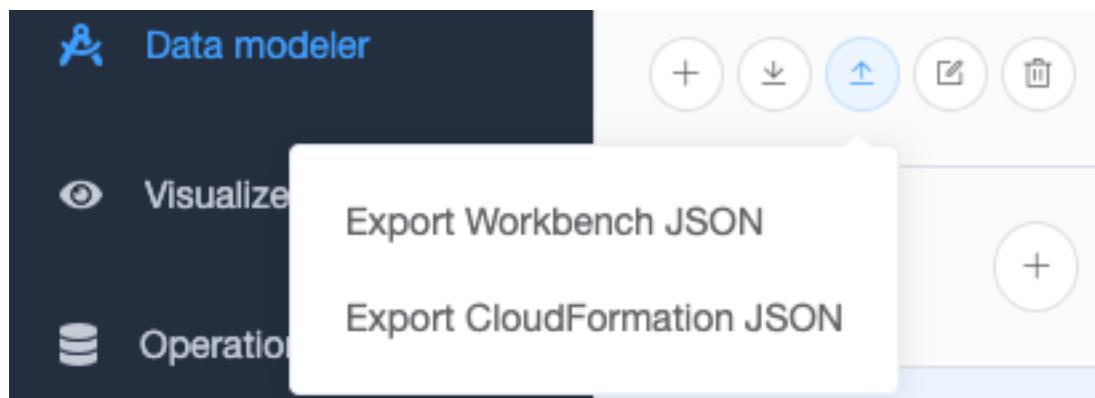
1. No NoSQL Workbench, no painel de navegação no lado esquerdo, escolha o ícone Data modeler (Modelador de dados).



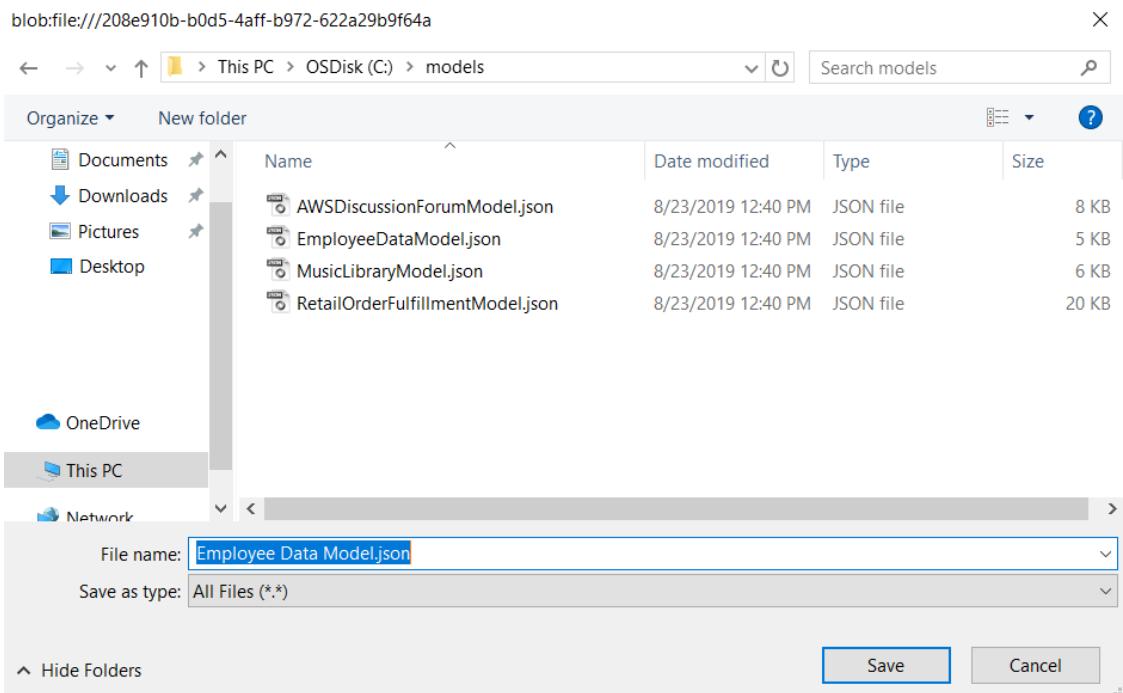
2. Passe o mouse sobre Exportar modelo de dados.

The screenshot shows the Data modeler page. It displays a data model named "Music Library Data Model". Below the data model name is a toolbar with several icons: a plus sign (+), a downward arrow (down), an upward arrow (up, highlighted with a red box), a checkmark (check), and a trash can (trash).

Na lista suspensa, escolha se deseja exportar seu modelo de dados no formato de modelo NoSQL Workbench ou no formato de modelo JSON do CloudFormation.



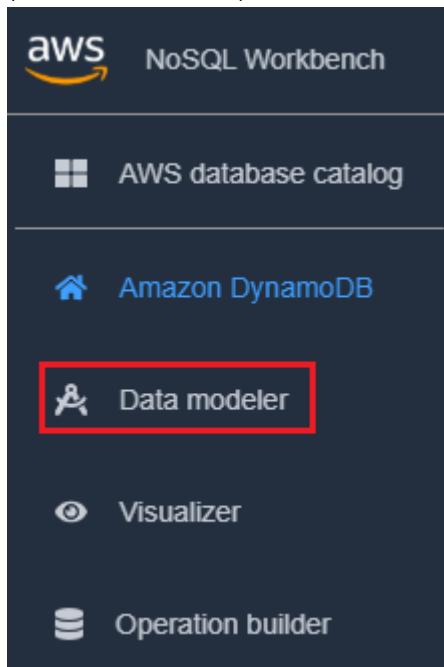
3. Escolha um local para salvar seu modelo.



Editar um modelo de dados existente

Como editar um usuário existente

1. No NoSQL Workbench, no painel de navegação no lado esquerdo, escolha o botão Data modeler (Modelador de dados).



2. Escolha a tabela que deseja editar.

Selecione Edit (Editar).

3. Faça as edições necessárias e escolha Save edits (Salvar edições).

Como editar manualmente um modelo existente e adicionar uma faceta

1. Exporte seu modelo. Para mais informações, consulte [Exportar um modelo de dados \(p. 841\)](#).
2. Abra o arquivo exportado em um editor.
3. Localize o objeto `DataModel` da tabela para a qual deseja criar uma faceta.

Adicione uma matriz `TableFacets` representando todas as facetas da tabela.

Para cada faceta, adicione um objeto à matriz `TableFacets`. Cada elemento da matriz tem as seguintes propriedades:

- `FacetName`: um nome para sua faceta. Esse valor deve ser exclusivo em todo o modelo.
- `PartitionKeyAlias`— um nome amigável para a chave de partição da tabela. Esse alias é exibido quando você visualiza a faceta no NoSQL Workbench.
- `SortKeyAlias`— um nome amigável para a chave de classificação da tabela. Esse alias é exibido quando você visualiza a faceta no NoSQL Workbench. Essa propriedade não será necessária se a tabela não tiver nenhuma chave de classificação definida.
- `NonKeyAttributes`— Uma matriz de nomes de atributo que são necessários para o padrão de acesso. Esses nomes devem ser mapeados para os nomes de atributo definidos para a tabela.

```
{  
  "modelName": "Music Library Data Model",  
  "dataModel": [  
    {  
      "tableName": "Songs",  
      "keyAttributes": {  
        "partitionKey": {  
          "attributeName": "Id",  
          "attributeType": "S"  
        },  
        "sortKey": {  
          "attributeName": "Metadata",  
          "attributeType": "S"  
        }  
      },  
      "nonKeyAttributes": [  
        {  
          "attributeName": "DownloadMonth",  
          "attributeType": "S"  
        },  
        {  
          "attributeName": "TotalDownloadsInMonth",  
          "attributeType": "S"  
        }  
      ]  
    }  
  ]  
}
```

```
        "AttributeType": "S"
    },
    {
        "AttributeName": "Title",
        "AttributeType": "S"
    },
    {
        "AttributeName": "Artist",
        "AttributeType": "S"
    },
    {
        "AttributeName": "TotalDownloads",
        "AttributeType": "S"
    },
    {
        "AttributeName": "DownloadTimestamp",
        "AttributeType": "S"
    }
],
"TableFacets": [
{
    "FacetName": "SongDetails",
    "KeyAttributeAlias": {
        "PartitionKeyAlias": "SongId",
        "SortKeyAlias": "Metadata"
    },
    "NonKeyAttributes": [
        "Title",
        "Artist",
        "TotalDownloads"
    ]
},
{
    "FacetName": "Downloads",
    "KeyAttributeAlias": {
        "PartitionKeyAlias": "SongId",
        "SortKeyAlias": "Metadata"
    },
    "NonKeyAttributes": [
        "DownloadTimestamp"
    ]
}
]
}
```

4. Agora você pode importar o modelo modificado para o NoSQL Workbench. Para mais informações, consulte [Importar um modelo de dados existente \(p. 838\)](#).

Visualizar padrões de acesso a dados

Você pode usar a ferramenta de visualização no NoSQL Workbench para o Amazon DynamoDB para mapear consultas e visualizar diferentes padrões de acesso (conhecidos como Facetas) de um aplicativo. Cada faceta corresponde a um padrão de acesso diferente no DynamoDB. Também é possível adicionar dados manualmente ao seu modelo de dados ou importar dados do MySQL.

Tópicos

- [Adicionar dados de exemplo a um modelo de dados \(p. 846\)](#)
- [Exibir padrões de acesso a dados \(p. 847\)](#)
- [Exibir todas as tabelas de um modelo de dados usando a exibição agregada \(p. 848\)](#)

- Confirmar um modelo de dados ao DynamoDB (p. 849)

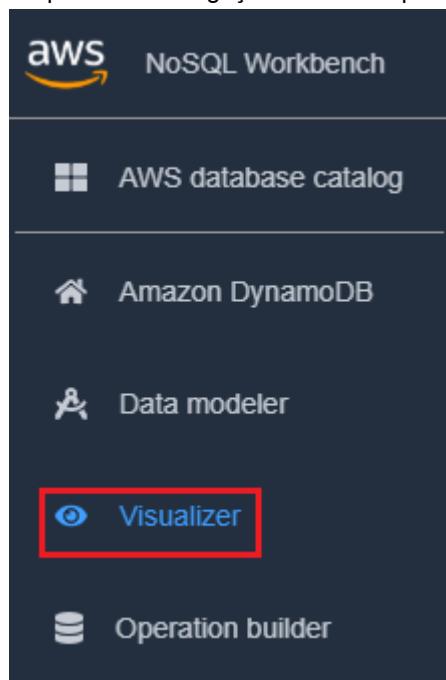
Adicionar dados de exemplo a um modelo de dados

Com a adição de dados de exemplo ao modelo, você pode exibir dados ao visualizar o modelo e seus vários padrões de acesso a dados ou facetas.

Siga estas etapas para adicionar dados de exemplo a um modelo de dados usando o NoSQL Workbench para o Amazon DynamoDB.

Para adicionar dados de exemplo

1. No painel de navegação no lado esquerdo, escolha o ícone de visualizador.



2. No visualizador, escolha Update (Atualizar) próximo ao nome da tabela.

A screenshot of the AWS NoSQL Workbench Visualizer interface. The interface has a dark sidebar on the left and a light-colored main area. The sidebar includes the same five items as the previous screenshot. The main area is titled "Visualizer". It shows a "Data model" dropdown set to "Music Library Data Model" with up and down arrows below it. In the main pane, there is a table row for "Songs" with an "Update" button to its right, which is also highlighted with a red rectangular box.

3. Escolha Add Data (Adicionar dados).

[FACET] SongDetails				
SongId (Partition key) : String	Metadata (Sort key) : String	Title : String	Artist : String	TotalDownloads : String

Add data Edit data

4. Insira os dados de exemplo nas caixas de texto vazias e escolhaAdd nova linhapara adicionar linhas adicionais. Quando terminar, escolhaSave (Salvar).

[FACET] SongDetails				
SongId (Partition key) : String	Metadata (Sort key) : String	Title : String	Artist : String	TotalDownloads : String
12	ACME Album	ACME Best Song	ACME	4

Cancel Save

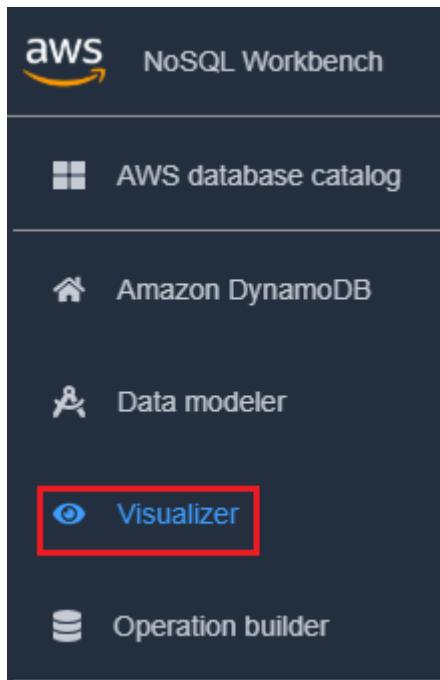
+ Add new row

Exibir padrões de acesso a dados

No NoSQL Workbench, Facetas Representam diferentes padrões de acesso a dados de um aplicativo para o Amazon DynamoDB.

Para exibir informações sobre facetas no NoSQL Workbench

1. No painel de navegação no lado esquerdo, escolha o ícone de visualizador.



2. No modelo de dados à esquerda, escolha uma tabela para exibir.
3. Escolha a seta da lista suspensa Facetas da tabela selecionada.
4. Na lista, escolha uma faceta para exibir.

The screenshot shows the AWS NoSQL Workbench interface with the 'Visualizer' tab selected. On the left, there's a sidebar with icons for AWS database catalog, Amazon DynamoDB, Data modeler, Visualizer (which is highlighted in blue), Operation builder, and Documentation. The main area is titled 'Visualizer' and shows a facet view for the 'SongDetails' table. It includes a 'Data model' dropdown set to 'Music Library Data Model', a table header with columns 'Songid (Partition key) : String', 'Metadata (Sort key) : String', 'Title : String', 'Artist : String', and 'TotalDownloads : String'. Below the header, there's a row with values: '12', 'ACME Band', 'Best Song', 'Attribute value', and 'Attribute value'. A 'Songs' section with an 'Update' button and a '+ Add new row' button is also visible.

Você também pode editar as definições de facetas usando o Modelador de dados. Para mais informações, consulte [Editar um modelo de dados existente \(p. 843\)](#).

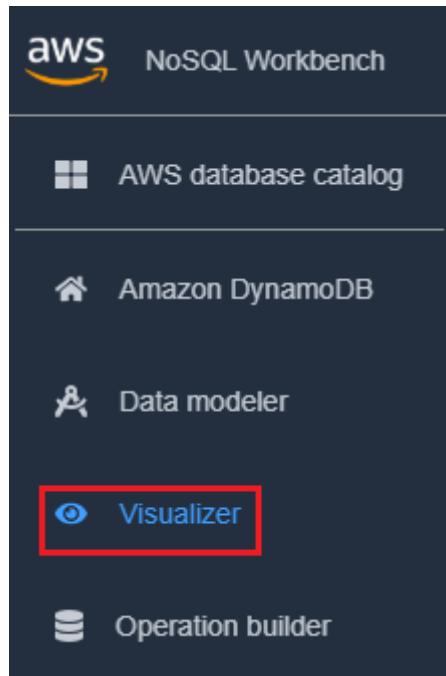
Exibir todas as tabelas de um modelo de dados usando a exibição agregada

A exibição agregada no NoSQL Workbench para Amazon DynamoDB representa todas as tabelas de um modelo de dados. Para cada tabela, são exibidas as seguintes informações:

- Nomes de colunas da tabela.
- Dados de exemplo.
- Todos os índices secundários globais associados à tabela. As informações a seguir são exibidas para cada índice:
 - Nomes de colunas do índice
 - Dados de exemplo

Para exibir todas as informações da tabela

1. No painel de navegação no lado esquerdo, escolha o ícone de visualizador.



2. No visualizador, escolha Aggregate view (Exibição agregada).

The screenshot shows the AWS NoSQL Workbench interface. On the left, there's a sidebar with icons for AWS database catalog, Amazon DynamoDB, Data modeler, Visualizer (which is selected and highlighted in blue), Operation builder, Documentation, and Email us. The main area is titled 'Visualizer' and 'Aggregate view'. It displays a table for the 'Forum' data model. The table has columns for Primary key (Partition key: ForumName), Category, Threads, Messages, and Views. Data rows include Amazon DynamoDB, Amazon Simple Notification Service, Amazon Simple Queue Service, Amazon MQ, and Amazon EMR. A red box highlights the 'Aggregate view' button at the bottom of the table.

Confirmar um modelo de dados ao DynamoDB

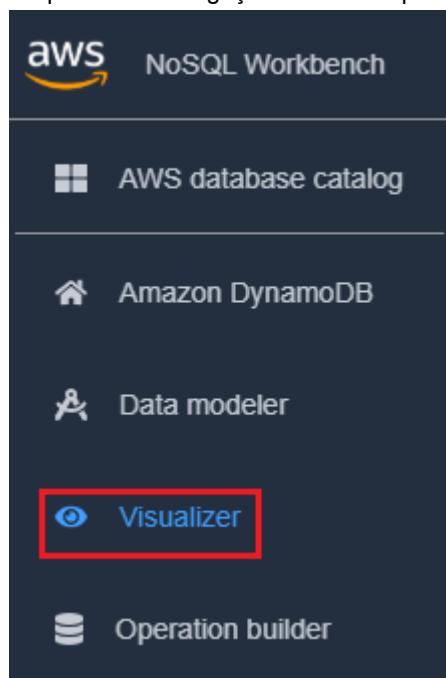
Quando estiver satisfeito com seu modelo de dados, você poderá confirmá-lo ao Amazon DynamoDB.

Note

- Essa ação resulta na criação de recursos no lado do servidor no AWSPara as tabelas e os índices secundários globais representados no modelo de dados.
- As tabelas são criadas com as seguintes características:
 - O Auto scaling é definido em 70% de utilização de destino.
 - A capacidade provisionada é definida em 5 unidades de capacidade de leitura e 5 unidades de capacidade de gravação.
 - Os índices secundários globais são criados com a capacidade provisionada de 10 unidades de capacidade de leitura e 5 unidades de capacidade de gravação.

Para confirmar o modelo de dados ao DynamoDB

1. No painel de navegação no lado esquerdo, escolha o ícone de visualizador.



2. Escolha Commit to DynamoDB (Confirmar ao DynamoDB).

AWS database catalog

Amazon DynamoDB

Data modeler

Visualizer

Operation builder

Documentation

Email us

Visualizer

Data model *i*

AWS Discussion Forum Data Model

Forum Update

Thread Update

Reply Update

sdfg Update

Aggregate view

Commit to Amazon DynamoDB *i*

3. Escolha uma conexão já existente ou crie uma nova conexão escolhendo a guia Add new connection remote (Adicionar nova conexão remota).

- Para adicionar uma nova conexão, especifique as seguintes informações:
 - Account Alias (Alias da conta)
 - AWS Região
 - ID de chave de acesso
 - Chave de acesso secreta

Para obter mais informações sobre como obter as chaves de acesso, consulte [Obter um AWS Chave de acesso](#).

- Opcionalmente, você pode especificar o seguinte:
 - [Token de sessão](#)
 - [ARN da função do IAM](#)
- Se não quiser se cadastrar em uma conta de nível gratuito e preferir usar o [DynamoDB Local \(versão disponível para download\)](#):

1. Escolha a guia DynamoDB local connection (Adicionar uma nova conexão local do DynamoDB)
2. Especifique o Connection name (Nome da conexão) e a Port (Porta).
4. Escolha Commit (Confirmar).

Commit to Amazon DynamoDB

On this page, you create server-side resources such as tables and global secondary indexes for the selected data model. If a table that uses auto scaling is being committed and a failure occurs while enabling auto scaling, table creation and item insertion will still be attempted and can be completed successfully. If the table has been created but auto scaling has not been enabled, use the DynamoDB console to enable auto scaling manually for the created table. If a table with auto scaling is being committed to DynamoDB local, auto scaling will not be enabled because it is not supported.

Use saved connections

[Add a new remote connection](#)

[Add a new DynamoDB local connection](#)

To use DynamoDB, you must have an AWS account. You can use AWS Identity and Access Management (IAM) roles and temporary security credentials to grant users in your AWS account access to your DynamoDB resources from NoSQL Workbench.

[Setting up DynamoDB](#)

Connection name

* AWS Region

AWS Region

* Access key ID

AWS access key ID

* Secret access key

AWS secret access key

Session token

AWS session token

(i)

IAM role ARN

IAM role ARN

(i)

Persist connection

If you select this check box, AWS connection secrets will be persisted in

/Users/[REDACTED]/.aws/credentials

[Cancel](#)

[Reset](#)

[Commit](#)

Explorar conjuntos de dados e criar operações com o NoSQL Workbench

O NoSQL Workbench para Amazon DynamoDB oferece uma avançada interface gráfica do usuário para desenvolvimento e teste de consultas. Você pode usar o criador de operações no NoSQL Workbench para visualizar, explorar e consultar conjuntos de dados. Também é possível usar o criador de operações estruturadas para criar e executar operações de plano de dados, com suporte para expressões de projeção e de condição, e gerar código de exemplo em várias linguagens.

Tópicos

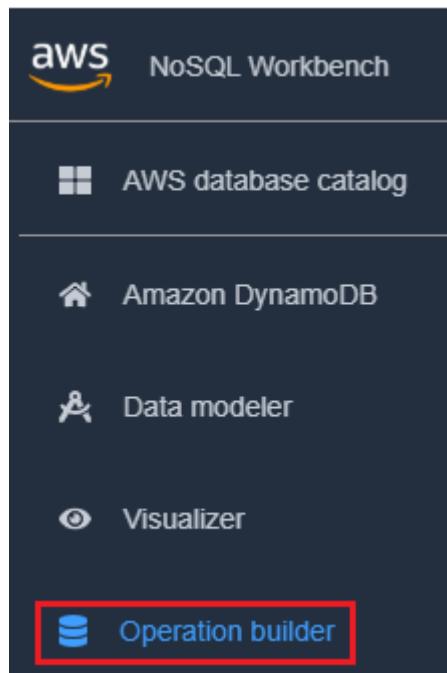
- [Explorar conjuntos de dados \(p. 853\)](#)
- [Criar operações complexas \(p. 857\)](#)

Explorar conjuntos de dados

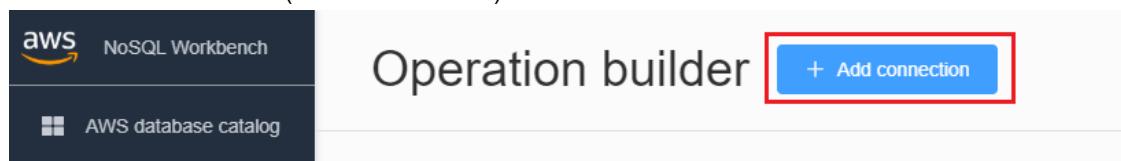
Para explorar suas tabelas do Amazon DynamoDB, primeiro você precisa se conectar ao AWS conta.

Para adicionar uma conexão ao seu banco de dados

1. No NoSQL Workbench, no painel de navegação no lado esquerdo, escolha o ícone Operation builder (Criador de operações).



2. Escolha Add Connection (Adicionar conexão).



3. Especifique as seguintes informações:

- Account alias (Alias da conta)
- AWS Região
- ID de chave de acesso
- Chave de acesso secreta

Para obter mais informações sobre como obter as chaves de acesso, consulte [Obter um AWS Chave de acesso](#).

Opcionalmente, você pode especificar o seguinte:

- [Token de sessão](#)
- [ARN da função do IAM](#)

4. Selecione Conectar.

Add a new database connection



A remote connection lets you access the DynamoDB web service in different AWS Regions.

A DynamoDB local connection lets you access the DynamoDB local server running on your computer.

Remote

DynamoDB local

* Connection name

Connection 3

* Default AWS Region

Default AWS Region

* Access key ID

AWS Access key ID

* Secret access key

AWS secret access key

Session token

AWS session token



IAM role ARN

IAM role ARN



Persist connection

If you select this check box, AWS connection secrets will be persisted in

C:\Users\m...ml.aws\credentials

Cancel

Reset

Connect

Se não quiser se cadastrar em uma conta de nível gratuito e preferir usar o [DynamoDB Local \(versão disponível para download\)](#):

- a. Selecione a guia Local na tela de conexão
- b. Especifique as seguintes informações:
 - Connection name
 - Porta
- c. Selecione o botão connect (conectar).

Add a new database connection

i A remote connection can let you interact with DynamoDB servers in different regions.
A local connection can let you interact with the DynamoDB Local server on your machine.

Remote Local

i Please setup the dynamoDB local server on your machine before adding a connection.
[Setting Up DynamoDB Local](#)

* Connection name: Connection 1

* Hostname: localhost

* Port: 8000

5. Na conexão criada, escolha Open (Abrir).

The screenshot shows the AWS NoSQL Workbench interface. On the left, there is a sidebar with the following options:

- AWS database catalog
- Amazon DynamoDB
- Data modeler
- Visualizer
- Operation builder

The main area is titled "Operation builder". It displays "Active connections" with one entry: "Connection 2" (Remote). Below this, there is a button labeled "Open" which is highlighted with a red box.

Após a conexão ao banco de dados do DynamoDB, a lista de tabelas disponíveis é exibida no painel esquerdo. Escolha uma das tabelas para retornar um exemplo dos dados armazenados na tabela.

Agora você pode executar consultas na tabela selecionada.

Para executar consultas em uma tabela

1. Na lista Attribute name (Nome do atributo), escolha o atributo que deseja consultar.
2. Especifique o operador de comparação.
3. Especifique o tipo de dados do valor.
4. Especifique o valor a ser consultado.
5. Selecione Scan (Verificar).

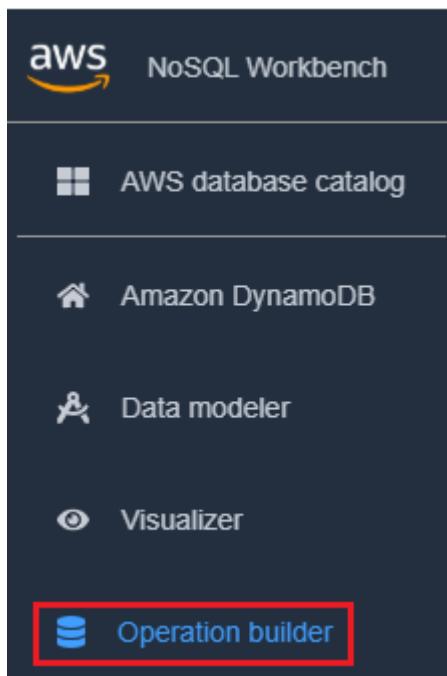
Para obter mais informações sobre essa operação, consulte [Scan](#) na Referência de API do Amazon DynamoDB.

Criar operações complexas

O criador de operações no NoSQL Workbench para Amazon DynamoDB oferece uma interface visual na qual você pode executar operações de plano de dados complexas. Ele inclui suporte para expressões de projeção e de condição. Você também pode optar por gerar código de exemplo para essas operações, em várias linguagens.

O NoSQL Workbench suporta a criação [PartiQL](#) para instruções do DynamoDB, que permite que você interaja com o Dynamodb usando uma linguagem de consulta compatível com SQL. O NoSQL Workbench também oferece suporte à criação de operações de API CRUD do DynamoDB.

Para usar o NoSQL Workbench para criar operações, no painel de navegação no lado esquerdo, escolha a opção Criador de operações ícone .



Tópicos

- [Construindo instruções do PartiQL \(p. 858\)](#)

- Criação de operações de API (p. 861)

Construindo instruções do PartiQL

Para usar a Bancada NoSQL para criar PartiQL para DynamoDB Instruções escolher Operações PartiQL No canto superior direito da Bancada NoSQL.

Você pode criar os seguintes tipos de instrução PartiQL no criador de operações.

Tópicos

- Declarações Singleton (p. 858)
- Transactions (p. 859)
- Batch (p. 860)

Declarações Singleton

Para executar ou gerar código para uma instrução PartiQL, faça o seguinte:

1 PartiQL statement 2 Statement

```
1 SELECT *
2 FROM Music
3 WHERE Artist=? and SongTitle=?
```

Optional request parameters 3.a

Enable strongly consistent reads 3.c

Parameters

Attribute type	Attribute value
String	Acme Band
String	PartiQL Rocks

+ Add new parameter 3.b

5 4

Clear form Run Generate code

1. Selecione Instrução PartiQL.
2. Insira um Instrução PartiQL.
3. Se sua instrução usa parâmetros
 - a. Selecione Parâmetros de solicitação.
 - b. Selecione Adicione novos parâmetros.
 - c. Informe o tipo e o valor do atributo
 - d. Se você quiser adicionar parâmetros adicionais, repita as etapas b e c.
4. Se desejar gerar código, escolha Generate code (Gerar código).

Selecione a linguagem desejada nas guias exibidas. Agora você pode copiar esse código e usá-lo no seu aplicativo.

5. Se desejar que a operação seja executada imediatamente, escolha Execução do.

Transactions

Para executar ou gerar código para uma transação PartiQL, faça o seguinte:

The screenshot shows the 'PartiQL transaction' tab selected in the top navigation bar. Below it are buttons for 'Delete all statements' (1), 'Collapse all statements' (2), 'Run' (3), and 'Generate code' (4). A note below the buttons states: 'Read and write operations are not supported in the same PartiQL transaction request. A SELECT statement cannot be in the same request with INSERT, UPDATE, and DELETE statements.' A dropdown menu is open, showing '1. Choose and then enter a title' (5). The main area is labeled 'Statement' and contains a code input field with the query '1 SELECT * FROM Music where Artist=? and SongTitle=?' (6). Below the code is a section for 'Optional request parameters' (7) with two entries: 'String' attribute type and 'Acme Band' attribute value (4.a), and another 'String' attribute type and 'PartiQL Rocks' attribute value (4.b). At the bottom right are 'Clear' and 'Delete' buttons.

1. Selecione Transações PartiQL.
2. Selecione Adicionar uma nova instrução.
3. Insira um Instrução PartiQL.

Note

Operações de leitura e gravação não são suportadas na mesma solicitação de transação do PartiQL. Uma instrução SELECT não pode estar na mesma solicitação com as instruções INSERT, UPDATE e DELETE. Consulte [Executando transações com o PartiQL para DynamoDB](#) para obter mais detalhes.

4. Se sua instrução usa parâmetros
 - a. Selecione Parâmetros de solicitação.
 - b. Selecione Adicione novos parâmetros.
 - c. Informe o tipo e o valor do atributo
 - d. Se você quiser adicionar parâmetros adicionais, repita as etapas b e c.
5. Para adicionar mais instruções, repita as etapas 2 a 4.
6. Se desejar gerar código, escolha Generate code (Gerar código).

Selecione a linguagem desejada nas guias exibidas. Agora você pode copiar esse código e usá-lo no seu aplicativo.

7. Se desejar que a operação seja executada imediatamente, escolha Execução do.

Batch

Para executar ou gerar código para um lote PartiQL, faça o seguinte:

1. Selecione Lote PartiQL.
2. Selecione Adicionar uma nova instrução.
3. Insira uma Instrução PartiQL.

Note

Operações de leitura e gravação não são suportadas na mesma solicitação de lote do PartiQL, o que significa que uma instrução SELECT não pode estar na mesma solicitação com instruções INSERT, UPDATE e DELETE. Operações de gravação para o mesmo item não são permitidas. Tal como acontece com a operação BatchGetItem, apenas operações de leitura singleton são suportadas. Operações de verificação e consulta não são aceitas. Consulte [Executando operações em Batch com o PartiQL para DynamoDB](#) para obter mais detalhes.

4. Se sua instrução usa parâmetros
 - a. Selecione Parâmetros de solicitação.
 - b. Selecione Adicione novos parâmetros.
 - c. Informe o tipo e o valor do atributo
 - d. Se você quiser adicionar parâmetros adicionais, repita as etapas b e c.
5. Para adicionar mais instruções, repita as etapas 2 a 4.
6. Se desejar gerar código, escolha Generate code (Gerar código).

Selecione a linguagem desejada nas guias exibidas. Agora você pode copiar esse código e usá-lo no seu aplicativo.

7. Se desejar que a operação seja executada imediatamente, escolha Execução do.

Criação de operações de API

Para usar o NoSQL Workbench para criar APIs CRUD do DynamoDB, escolha Operações baseadas em interface no canto superior direito da Bancada NoSQL.

Selecione Operações de plano de dados E escolha a operação que você deseja criar.

The screenshot shows the AWS NoSQL Workbench interface. On the left, there's a sidebar with links for AWS database catalog, Amazon DynamoDB (which is selected and highlighted in blue), Data modeler, Visualizer, Operation builder (which is also selected and highlighted in blue), Documentation, and Email us. The main area is titled "Operation builder". It has two tabs: "Connection" and "Tables". Under "Connection", there are fields for "Name" (set to "Connection 2") and "Region" (set to "us-east-1"). Under "Tables", there's a search bar. To the right, there's a "Build operations" section with a "Data plane operations" tab selected. A dropdown menu titled "Choose an operation" is open, listing several operations: UpdateItem, PutItem, DeleteItem, GetItem, Query, Scan, and TransactWriteItems. The "Data plane operations" tab is highlighted with a red box.

Você pode executar as operações a seguir no criador de operações.

Tópicos

- [Obter Item \(p. 861\)](#)
- [Put Item \(p. 862\)](#)
- [Update Item \(p. 863\)](#)
- [Delete Item \(p. 864\)](#)
- [Query \(p. 865\)](#)
- [Scan \(p. 865\)](#)
- [TransactGetItems \(p. 866\)](#)
- [TransactWriteItems \(p. 867\)](#)

Obter Item

Para executar ou gerar código para um `Get Item`, faça o seguinte.

1. Especifique o valor da chave de partição.
2. Para adicionar uma expressão de projeção, faça o seguinte:

- a. Selecione Expressão de projeção.
- b. Selecione o + (sinal de adição) ao lado de Expressão de projeção.
- c. Especifique o Nome do atributo,.

The screenshot shows the 'Put Item' form in the AWS Lambda console. At the top, there's a field for 'Partition key' with the value 'Attribute value for partition key: Name'. Below it is a section for 'Projection expression' with a checkbox checked and a '+' button. Underneath is a field for 'Projected attribute' with 'Attribute name' and a save icon. At the bottom, there's a 'Clear form' button, a 'Run' button, and a 'Generate code' button.

3. Se desejar gerar código, escolha Generate code (Gerar código).

Selecione a linguagem desejada nas guias exibidas. Agora você pode copiar esse código e usá-lo no seu aplicativo.

4. Se desejar que a operação seja executada imediatamente, escolha Execução do.

Para obter mais informações sobre essa operação, consulte [PutItem](#) na Referência de API do Amazon DynamoDB.

Put Item

Para executar ou gerar código para um `Put Item`, faça o seguinte.

1. Especifique o valor da chave de partição.
2. Especifique o valor da chave de classificação, se houver.
3. Se desejar adicionar atributos que não sejam de chave, faça o seguinte:
 - a. Escolha o + (sinal de adição) ao lado de Other attributes (Outros atributos).
 - b. Especifique o Attribute name (Nome do atributo), o Type (Tipo) e o Value (Valor).
4. Se a expressão de condição precisar ser satisfeita para a operação `Put Item` ser bem-sucedida, faça o seguinte:
 - a. Escolha Condition (Condição).
 - b. Especifique o nome do atributo, o operador de comparação, o tipo de atributo e o valor do atributo.
 - c. Se outras condições forem necessárias, escolha Condition (Condição) novamente.

Para mais informações, consulte [Expressões de condição \(p. 422\)](#).

The screenshot shows the 'PutItem' form in the Amazon DynamoDB developer guide. It includes fields for partition key ('Name'), other attributes (with a '+' button), condition expression (with '+ Condition' and '+ Child expression' buttons, and 'And', 'Or', 'Negate' options), and update expressions (with 'Condition', 'Negate' toggle, and dropdowns for attribute name, operator (=), attribute type, and value). Buttons at the bottom include 'Clear form', 'Run', and 'Generate code'.

5. Se desejar gerar código, escolha Generate code (Gerar código).

Selecione a linguagem desejada nas guias exibidas. Agora você pode copiar esse código e usá-lo no seu aplicativo.

6. Se desejar que a operação seja executada imediatamente, escolha Execução do.

Para obter mais informações sobre essa operação, consulte [PutItem](#) na Referência de API do Amazon DynamoDB.

Update Item

Para executar ou gerar código para um `Update Item`, faça o seguinte:

1. Insira o valor da chave de partição.
2. Insira o valor da chave de classificação, se houver.
3. Em Update expression (Expressão de atualização), escolha a expressão na lista.
4. Escolha o + (sinal de adição) para a expressão.
5. Insira o nome e o valor do atributo para a expressão selecionada.
6. Se desejar adicionar mais expressões, escolha outra expressão na lista suspensa Update Expression (Expressão de atualização) e selecione +.
7. Se a expressão de condição precisar ser satisfeita para a operação `Update Item` ser bem-sucedida, faça o seguinte:
 - a. Escolha Condition (Condição).
 - b. Especifique o nome do atributo, o operador de comparação, o tipo de atributo e o valor do atributo.
 - c. Se outras condições forem necessárias, escolha Condition (Condição) novamente.

Para mais informações, consulte [Expressões de condição \(p. 422\)](#).

The screenshot shows the 'Update expression' section of the configuration form. It includes fields for 'Partition key' (with a placeholder 'Attribute value for partition key: Name'), 'Update expression' (with a dropdown 'Select operation' and a '+' button), 'Condition expression' (with buttons '+ Condition' and '+ Child expression'), and a condition builder for 'Attribute name', 'Attribute type', and 'Attribute value'. At the bottom are 'Clear form', 'Run', and 'Generate code' buttons.

8. Se desejar gerar código, escolha Generate code (Gerar código).

Escolha a guia da linguagem desejada. Agora você pode copiar esse código e usá-lo no seu aplicativo.

9. Se desejar que a operação seja executada imediatamente, escolha Execução do.

Para obter mais informações sobre essa operação, consulte [UpdateItem](#) na Referência de API do Amazon DynamoDB.

Delete Item

Para executar ou gerar código para um `Delete Item`, faça o seguinte.

1. Insira o valor da chave de partição.
2. Insira o valor da chave de classificação, se houver.
3. Se a expressão de condição precisar ser satisfeita para a operação `Delete Item` ser bem-sucedida, faça o seguinte:
 - a. Escolha Condition (Condição).
 - b. Especifique o nome do atributo, o operador de comparação, o tipo de atributo e o valor do atributo.
 - c. Se outras condições forem necessárias, escolha Condition (Condição) novamente.

Para mais informações, consulte [Expressões de condição \(p. 422\)](#).

The screenshot shows the 'Delete expression' section of the configuration form. It includes fields for 'Partition key' (with a placeholder 'Attribute value for partition key: Name'), 'Condition expression' (with buttons '+ Condition' and '+ Child expression'), and a condition builder for 'Attribute name', 'Attribute type', and 'Attribute value'. At the bottom are 'Clear form', 'Run', and 'Generate code' buttons.

4. Se desejar gerar código, escolha Generate code (Gerar código).

Escolha a guia da linguagem desejada. Agora você pode copiar esse código e usá-lo no seu aplicativo.

5. Se desejar que a operação seja executada imediatamente, escolha Execução do.

Para obter mais informações sobre essa operação, consulte[DeleteItem](#)noReferência de API do Amazon DynamoDB.

Query

Para executar ou gerar código para um[Query](#), faça o seguinte.

1. Especifique o valor da chave de partição.
2. Se uma chave de classificação for necessária para a operação [Query](#):
 - a. Selecione Sort key (Chave de classificação).
 - b. Especifique o operador de comparação, o tipo de atributo e o valor do atributo.
3. Se nem todos os atributos precisarem ser retornados com o resultado da operação, selecione Projection expression (Expressão de projeção).
4. Escolha o + (sinal de adição).
5. Insira o atributo a ser retornado com o resultado da consulta.
6. Se mais atributos forem necessários, escolha o +.
7. Se a expressão de condição precisar ser satisfeita para a operação [Query](#) ser bem-sucedida, faça o seguinte:
 - a. Escolha Condition (Condição).
 - b. Especifique o nome do atributo, o operador de comparação, o tipo de atributo e o valor do atributo.
 - c. Se outras condições forem necessárias, escolha Condition (Condição) novamente.

Para mais informações, consulte [Expressões de condição \(p. 422\)](#).

The screenshot shows the AWS Lambda function configuration interface. The 'Handler' section is visible, containing fields for 'File name' (lambda_function.py), 'Handler' (lambda_function.lambda_handler), and 'Role' (arn:aws:iam::123456789012:role/lambdaBasicExecutionRole). Below these, there are sections for 'Environment variables' and 'Code' (with a 'Upload' button).

8. Se desejar gerar código, escolha Generate code (Gerar código).

Escolha a guia da linguagem desejada. Agora você pode copiar esse código e usá-lo no seu aplicativo.

9. Se desejar que a operação seja executada imediatamente, escolha Execução do.

Para obter mais informações sobre essa operação, consulte[Consulte](#)noReferência de API do Amazon DynamoDB.

Scan

Para executar ou gerar código para um[Scan](#), faça o seguinte.

1. Se nem todos os atributos precisarem ser retornados com o resultado da operação, selecione Projection expression (Expressão de projeção).
2. Escolha o + (sinal de adição).
3. Especifique o atributo a ser retornado com o resultado da consulta.
4. Se mais atributos forem necessários, escolha o + novamente.
5. Se a expressão de condição precisar ser satisfeita para a operação scan ser bem-sucedida, faça o seguinte:
 - a. Escolha Condition (Condição).
 - b. Especifique o nome do atributo, o operador de comparação, o tipo de atributo e o valor do atributo.
 - c. Se outras condições forem necessárias, escolha Condition (Condição) novamente.

Para mais informações, consulte [Expressões de condição \(p. 422\)](#).

6. Se desejar gerar código, escolha Generate code (Gerar código).

Escolha a guia da linguagem desejada. Agora você pode copiar esse código e usá-lo no seu aplicativo.

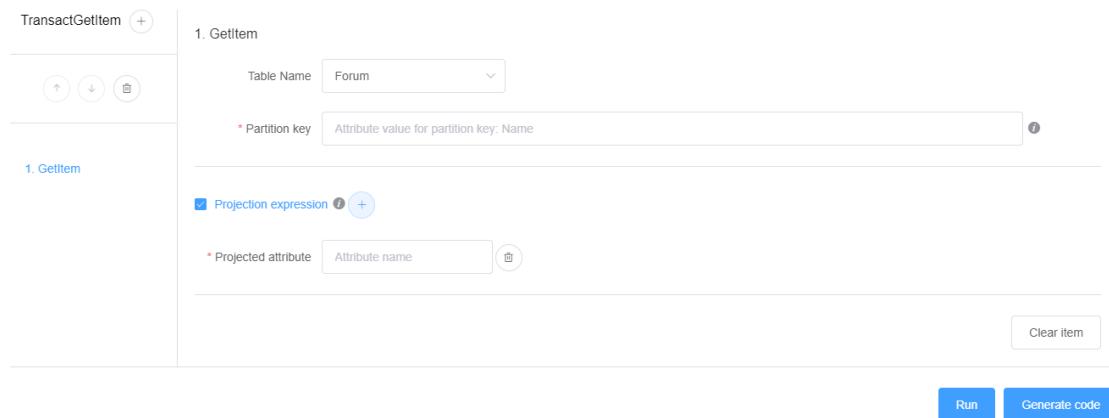
7. Se desejar que a operação seja executada imediatamente, escolha Execução do.

TransactGetItems

Para executar ou gerar código para um TransactGetItems, faça o seguinte.

1. Escolha o + (sinal de adição).
2. Siga as instruções para o [Obter Item \(p. 861\)](#) operação.

Quando terminar de especificar os detalhes da operação, escolha a opção +(sinal de adição) se quiser adicionar mais operações.



Para alterar a ordem das ações, escolha uma ação na lista à esquerda e selecione as setas para cima ou para baixo para movê-la para cima ou para baixo na lista.

Para excluir uma ação, escolha a ação na lista e selecione o ícone Delete (Excluir) (lixeira).

3. Se desejar gerar código, escolha Generate code (Gerar código).

Escolha a guia da linguagem desejada. Agora você pode copiar esse código e usá-lo no seu aplicativo.

4. Se desejar que a operação seja executada imediatamente, escolha Execução do.

Para obter mais informações, consulte [Amazon DynamoDB Transactions](#).

TransactWriteItems

Para executar ou gerar código para um `TransactWriteItems`, faça o seguinte.

1. Na lista suspensa Actions (Ações), escolha a operação desejada.

- Para `DeleteItem`, siga as instruções para a operação [Delete Item \(p. 864\)](#).
- Para `PutItem`, siga as instruções para a operação [Put Item \(p. 862\)](#).
- Para `UpdateItem`, siga as instruções para a operação [Update Item \(p. 863\)](#).

Quando terminar de especificar os detalhes da operação, escolha o botão +.

The screenshot shows the AWS Lambda Function Builder interface. On the left, there is a sidebar titled 'Actions' with a dropdown menu set to 'PutItem'. Other options in the dropdown include 'DeleteItem', 'PutItem', 'UpdateItem', and 'ConditionCheck'. Below the dropdown, there is a link '1. PutItem'. The main panel is titled 'PutItem' and contains the following fields:

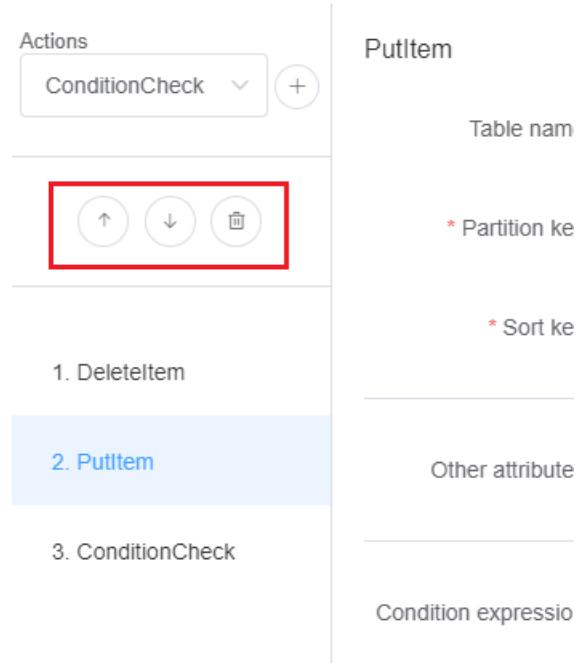
- 'Table name': A dropdown menu set to 'Forum'.
- '* Partition key': A text input field with the placeholder 'Attribute value for partition key: Name'.
- 'Other attributes': A section with a '+' button and a help icon.
- 'Condition expression': A section with '+ Condition' and '+ Child expression' buttons and a help icon.

At the bottom right of the main panel are three buttons: 'Run', 'Generate code', and 'Clear'.

Para alterar a ordem das ações, escolha uma ação na lista à esquerda e selecione as setas para cima ou para baixo para movê-la para cima ou para baixo na lista.

Para excluir uma ação, escolha a ação na lista e selecione o ícone Delete (Excluir) (lixo).

TransactWriteItems operation builder



2. Se desejar gerar código, escolha Generate code (Gerar código).

Escolha a guia da linguagem desejada. Agora você pode copiar esse código e usá-lo no seu aplicativo.

3. Se desejar que a operação seja executada imediatamente, escolha Execução do.

Para obter mais informações, consulte [Amazon DynamoDB Transactions](#).

Exemplos de modelos de dados para o NoSQL Workbench

A página inicial do modelador e visualizador exibe uma série de exemplos de modelos fornecidos com o NoSQL Workbench. Esta seção descreve esses modelos e seus usos potenciais.

Tópicos

- [Modelo de dados do funcionário \(p. 869\)](#)
- [Modelo de dados do fórum de discussão \(p. 869\)](#)
- [Modelo de dados da biblioteca de músicas \(p. 869\)](#)
- [Modelo de dados da estância de esqui \(p. 870\)](#)
- [Modelo de dados de ofertas de cartão de crédito \(p. 870\)](#)
- [Modelo de dados de favoritos \(p. 870\)](#)

Modelo de dados do funcionário

Este modelo de dados é um modelo introdutório. Ele representa os detalhes básicos de um funcionário, como um alias exclusivo, nome, sobrenome, designação, gerente e habilidades.

Este modelo de dados descreve algumas técnicas, como lidar com atributo complexo, como ter mais de uma habilidade. Este modelo também é um exemplo de relação do tipo “um para muitos” entre o gerente e seus funcionários subordinados, alcançado pelo índice secundário DirectReports.

Os padrões de acesso facilitados por este modelo de dados são:

- Recuperação de um registro de funcionário usando o alias de login do funcionário, facilitada por uma tabela chamada `Employee`.
- Pesquisa de funcionários por nome, facilitada pelo índice secundário global da tabela `Employee` (Funcionários) chamado `Name`.
- Recuperação de todos os relatórios diretos de um gerente usando o alias de login do gerente, facilitada pelo índice secundário global da tabela `Employee` (Funcionários) chamado `DirectReports`.

Modelo de dados do fórum de discussão

Este modelo de dados representa fóruns de discussão. Usando este modelo, os clientes podem se envolver com a comunidade de desenvolvedores, fazer perguntas e responder às postagens de outros clientes. Cada usuário tem um fórum dedicado. Qualquer usuário pode iniciar um novo tópico de discussão postando uma mensagem em um fórum, e cada assunto de tópico recebe um número de respostas.

Os padrões de acesso facilitados por este modelo de dados são:

- Recuperação de um registro de fórum usando o nome do fórum, facilitada por uma tabela chamada `Forum`.
- Recuperação de um thread específico ou de todos os threads de um fórum, facilitada por uma tabela chamada `Thread`.
- Pesquisa de respostas usando o endereço de e-mail do usuário de publicação, facilitada pelo índice secundário global da tabela `Reply` (Responder) chamado `PostedBy-MESSAGE-Index`.

Modelo de dados da biblioteca de músicas

Este modelo de dados representa uma biblioteca de músicas que tem uma grande coleção de músicas e mostra as mais baixadas quase em tempo real.

Os padrões de acesso facilitados por este modelo de dados são:

- Recuperação de um registro de músicas, facilitada por uma tabela chamada `Songs`.
- Recuperação de um registro de download específico ou de todos os registros de download de uma música, facilitada por uma tabela chamada `Songs`.
- Recuperação de um registro de contagem de downloads mensais específico ou de todos os registros de contagem de downloads mensais de uma música, facilitada por uma tabela chamada `Song`.
- Recuperação de todos os registros (incluindo gravação de músicas, registros de download e registros de contagem de downloads mensais) de uma música, facilitada por uma tabela chamada `Songs`.
- Pesquisa das músicas mais baixadas, facilitada pelo índice secundário global da tabela `Músicas` chamado `DownloadsByMonth`.

Modelo de dados da estância de esqui

Este modelo de dados representa uma estância de esqui que tem uma grande coleção de dados para cada subida mecânica recolhidos diariamente.

Os padrões de acesso facilitados por este modelo de dados são:

- Recuperação de todos os dados para um determinado teleférico ou resort geral, dinâmico e estático, facilitado por uma tabela chamada `SkiLifts`.
- Recuperação de todos os dados dinâmicos (incluindo ciclistas de elevadores exclusivos, cobertura de neve, perigo de avalanche e status de elevador) para um teleférico ou o resort em geral em uma data específica, facilitada por uma tabela chamada `SkiLifts`.
- Recuperação de todos os dados estáticos (incluindo se o elevador for apenas para ciclistas experientes, pés verticais e tempo de elevação) para um teleférico específico, facilitado por uma mesa chamada `SkiLifts`.
- Recuperação da data dos dados registrados para uma subida mecânica específica ou a instância geral classificada pelo total de esquiadores únicos, facilitada pelo índice secundário global da tabela `SkiLifts`, chamado `SkiLiftsByRiders`.

Modelo de dados de ofertas de cartão de crédito

Este modelo de dados é utilizado por uma Aplicação de Ofertas de Cartão de Crédito.

Um provedor de cartão de crédito produz ofertas ao longo do tempo. Essas ofertas incluem transferências de saldo sem taxas, aumento do limite de crédito, redução das taxas de juros, reembolso e milhas aéreas. Depois que um cliente aceita ou recusa essas ofertas, o respectivo status da oferta é atualizado de forma correspondente.

Os padrões de acesso facilitados por este modelo de dados são:

- Recuperação de registros de conta usando `AccountId`, facilitada pela tabela principal.
- Recuperação de todas as contas com poucos itens projetados, facilitada pelo índice secundário `AccountIndex`.
- Recuperação de contas e todos os registros de oferta associados a essas contas usando `AccountId`, facilitada pela tabela principal.
- Recuperação de contas e registros de ofertas específicas associados a essas contas usando `AccountId` e `OfferId`, facilitada pela tabela principal.
- Recuperação de todos `ACCEPTED/DECLINED` oferecer registros de `OfferType` associado a contas usando `AccountId`, `OfferType`, `eStatus`, como facilitado pelo índice secundário `GSI1`.
- Recuperação de ofertas e registros de itens de oferta associados usando `OfferId`, facilitada pela tabela principal.

Modelo de dados de favoritos

Este modelo de dados é usado para armazenar favoritos para os clientes.

Um cliente pode ter vários favoritos e um favorito pode pertencer a vários clientes. Este modelo de dados representa uma relação do tipo “muitos para muitos”.

Os padrões de acesso facilitados por este modelo de dados são:

- Agora, uma única consulta com o `customerId` pode retornar dados do cliente e dos favoritos.

- Consultar um `ByEmail` índice retorna os dados do cliente por endereço de e-mail. Observe que os marcadores não são recuperados por este índice.
- Um índice de consulta `ByUrl` obtém dados de favoritos por URL. Observe que `customerId` é a chave de classificação para o índice porque o mesmo URL pode ser adicionado aos favoritos por vários clientes.
- Consultar um `ByCustomerFolder` obtém marcadores por pasta para cada cliente.

Histórico de versões do NoSQL Workbench

A tabela a seguir descreve as alterações importantes em cada versão da ferramenta do cliente do NoSQL Workbench.

Alteração	Descrição	Data
Configurações de capacidade e importação/exportação do CloudFormation	O NoSQL Workbench for Amazon DynamoDB agora oferece suporte à especificação de um modo de capacidade de leitura/gravação para tabelas e agora pode importar e exportar modelos de dados no AWS CloudFormation format.	21 de abril de 2021
Support para o PartiQL	NoSQL Workbench para Amazon DynamoDB adiciona suporte para a criação de instruções PartiQL para DynamoDB.	4 de dezembro de 2020
Support para Linux.	O NoSQL Workbench para Amazon DynamoDB é compatível com Ubuntu, Fedora e Debian Linux.	4 de maio de 2020
NoSQL Workbench para Amazon DynamoDB – GA.	NoSQL Workbench para Amazon DynamoDB geralmente está disponível.	2 de março de 2020
Suporte para funções do IAM e credenciais de segurança temporárias.	NoSQL Workbench para Amazon DynamoDB adiciona suporte para o AWS Identity and Access Management Functions (IAM) e credenciais de segurança temporárias.	19 de dezembro de 2019
Suporte para DynamoDB Local (versão disponível para download).	Agora o NoSQL Workbench oferece suporte à conexão ao DynamoDB Local (versão disponível para download) para desenvolver, criar, consultar e gerenciar tabelas do DynamoDB.	8 de novembro de 2019
A demonstração do NoSQL Workbench foi lançada.	Essa é a versão inicial do NoSQL Workbench para DynamoDB. Use o NoSQL Workbench para projetar, criar, consultar e gerenciar tabelas do DynamoDB. Para obter mais informações,	16 de setembro de 2019

Alteração	Descrição	Data
	consulte NoSQL Workbench para Amazon DynamoDB (demonstração) .	

Segurança e conformidade no Amazon DynamoDB

A segurança da nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você se beneficiará de um datacenter e de uma arquitetura de rede criados para atender aos requisitos das empresas com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O [modelo de responsabilidade compartilhada](#) descreve isso como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem: AWSA é responsável por proteger a infraestrutura que executa oAWSServiços da naAWSNuvem. AWSO também fornece serviços que você pode usar com segurança. A eficácia de nossa segurança é regularmente testada e verificada por auditores terceirizados como parte dos [programas de conformidade da AWS](#). Para saber mais sobre os programas de conformidade que se aplicam ao DynamoDB, consulte [AWSServiços no escopo pelo programa de conformidade](#).
- Segurança na nuvem— Sua responsabilidade é determinada peloAWSque você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da sua organização e as leis e regulamentos aplicáveis.

Esta documentação ajudará você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o DynamoDB. Os tópicos a seguir mostram como configurar o DynamoDB para atender aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outrasAWSServiços da que podem ajudar você a monitorar e proteger seus recursos do DynamoDB.

Tópicos

- [Proteção de dados no DynamoDB \(p. 873\)](#)
- [Identity and Access Management \(p. 883\)](#)
- [Validação de conformidade por setor do DynamoDB \(p. 915\)](#)
- [Resiliência e recuperação de desastres no Amazon DynamoDB \(p. 916\)](#)
- [Segurança da infraestrutura no Amazon DynamoDB \(p. 916\)](#)
- [Análise de configuração e vulnerabilidade no Amazon DynamoDB \(p. 923\)](#)
- [Melhores práticas de segurança para o Amazon DynamoDB \(p. 923\)](#)

Proteção de dados no DynamoDB

O Amazon DynamoDB fornece uma infraestrutura de armazenamento resiliente projetada para armazenamento de dados de missão crítica e primários. Os dados são armazenados, de maneira redundante, em vários dispositivos de diversas instalações em uma região do Amazon DynamoDB.

O DynamoDB protege os dados de usuário armazenados em repouso e também os dados em trânsito entre os clientes no local e o DynamoDB, e entre o DynamoDB e outrosAWSrecursos dentro do mesmoAWSRegião :

Tópicos

- [Criptografia do DynamoDB em repouso \(p. 874\)](#)
- [Proteção de dados no DynamoDB Accelerator \(p. 882\)](#)
- [Privacidade do tráfego entre redes \(p. 883\)](#)

Criptografia do DynamoDB em repouso

Todos os dados do usuário armazenados no Amazon DynamoDB são totalmente criptografados em repouso. A criptografia de DynamoDB em repouso fornece segurança aprimorada criptografando todos os seus dados em repouso usando chaves de criptografia armazenadas no [AWS Key Management Service\(AWS KMS\)](#). Essa funcionalidade ajuda a reduzir a carga e complexidade operacionais necessárias para proteger dados confidenciais. Com a criptografia de dados em repouso, você pode criar aplicativos confidenciais que atendem a requisitos rigorosos de conformidade e regulamentação de criptografia.

A criptografia de DynamoDB em repouso fornece uma camada adicional de proteção de dados mantendo seus dados seguros na tabela criptografada — incluindo a chave principal, os índices secundários local e global, os fluxos, as tabelas globais, os backups e os clusters do DynamoDB Accelerator (DAX) sempre que dados forem armazenados em mídia durável. Políticas organizacionais, regulamentações setoriais ou governamentais e exigências de conformidade geralmente demandam o uso de criptografia em repouso para aumentar a segurança de dados de seus aplicativos.

A criptografia em repouso integra-se com AWS KMS para gerenciamento da chave de criptografia usada para criptografar suas tabelas. Para obter mais informações, consulte [AWS Key Management ServiceConceitos](#)no AWS Key Management ServiceGuia do desenvolvedor.

Ao criar uma nova tabela, você pode escolher uma das seguintes chaves mestre de cliente (CMK) para criptografar a tabela:

- AWSCMK de propriedade da — Tipo de criptografia padrão. A chave é pertencida pelo DynamoDB (sem custo adicional).
- AWSCMK gerenciado pela — A chave está armazenada em sua conta e é gerenciada pelo AWS KMS(AWS KMS Cobranças são aplicadas).
- CMK gerenciada pelo cliente — a chave é armazenada na sua conta e você a cria, a detém e a gerencia. Você tem total controle sobre a CMK (são aplicadas cobranças do AWS KMS).

Note

Ao criar um novo cluster DAX com criptografia em repouso habilitada, um AWSCMK gerenciado será usado para criptografar dados em repouso no cluster.

Quando você acessa uma tabela criptografada, o DynamoDB descriptografa os dados da tabela de forma transparente. Você pode alternar entre o AWSCMK de propriedade da, AWSCMK gerenciada pelo cliente e a CMK gerenciada pelo cliente a qualquer momento. Não é necessário alterar código nem aplicativos para usar ou gerenciar tabelas criptografadas. O DynamoDB continua a entregar a mesma latência milissegundo de dígito único que você espera, e todas as consultas do DynamoDB funcionam perfeitamente em seus dados criptografados.

É possível especificar uma chave de criptografia ao criar uma tabela ou alternar as chaves de criptografia em uma tabela existente usando o AWS Management Console,AWS Command Line Interface(AWS CLI) ou a API do Amazon DynamoDB. Para saber como, consulte [Gerenciamento das tabelas criptografadas no DynamoDB \(p. 878\)](#).

Criptografia em repouso usando o AWSA CMK pertencida pelo é oferecida gratuitamente. No entanto,AWS KMS Cobranças são aplicadas para uma AWSCMK gerenciada pelo cliente e a CMK gerenciada pelo cliente. Para obter mais informações sobre a definição de preço, consulte [Definição de preço do AWS KMS](#).

A criptografia do DynamoDB em repouso está disponível em todos os AWS Regiões, incluindo o AWS China (Pequim) e AWS Regiões da China (Ningxia) e do AWS Regiões GovCloud (US). Para obter mais informações, consulte [Criptografia em repouso: Como ele funciona \(p. 875\)](#) e [Notas de uso de criptografia em repouso do DynamoDB \(p. 876\)](#).

Criptografia em repouso: Como ele funciona

A criptografia do Amazon DynamoDB em repouso criptografa seus dados usando Advanced Encryption Standard de 256 bits (AES-256), ajudando a proteger os dados contra acesso não autorizado ao armazenamento subjacente.

A criptografia em repouso integra-se com AWS Key Management Service (AWS KMS) para gerenciamento da chave de criptografia usada para criptografar suas tabelas.

Ao criar uma nova tabela ou alternar as chaves de criptografia em uma tabela existente, você pode escolher uma das seguintes chaves mestras de cliente (CMK):

- AWSCMK de propriedade da — Tipo de criptografia padrão. A chave é pertencida pelo DynamoDB (sem custo adicional).
- AWSCMK gerenciado pela — A chave está armazenada em sua conta e é gerenciada pelo AWS KMS (AWS KMS Cobranças são aplicadas).
- CMK gerenciada pelo cliente — a chave é armazenada na sua conta e você a cria, a detém e a gerencia. Você tem total controle sobre a CMK (só aplicadas cobranças do AWS KMS).

AWSCMK de propriedade da

AWSCMKS de propriedade da não são armazenadas no AWS conta. Elas fazem parte de uma coleção de CMKs que a AWS possui e gerencia para uso em várias contas da AWS. Os serviços da AWS podem usar CMKs de propriedade da AWS para proteger seus dados.

Você não pode visualizar, gerenciar ou usar CMKs de propriedade da AWS, tampouco auditá-la seu uso. No entanto, você não precisa fazer nenhum trabalho nem alterar nenhum programa para proteger as chaves que criptografam seus dados.

Não é cobrada taxa mensal nem taxa de uso pelo uso do AWSCMKS de propriedade, e eles não contam em relação ao AWS KMS Cotas para sua conta do.

AWSCMK gerenciada pela

AWSAs CMKs gerenciadas pela são as CMKs em sua conta que são criadas, gerenciadas e usadas em seu nome por uma AWS que é integrado ao AWS KMS. Você pode visualizar as CMKs gerenciadas pela AWS na sua conta, visualizar suas políticas de chaves e auditá-las em logs do AWS CloudTrail. No entanto, não é possível gerenciar essas CMKs nem alterar suas permissões.

A criptografia em repouso integra-se automaticamente com o AWS KMS para gerenciar o AWSCMK gerenciada pela DynamoDB (`aws/dynamodb`) usado para criptografar as tabelas do. Se um AWSA CMK gerenciada pela não existe ao criar sua tabela do DynamoDB criptografada, o AWS KMS cria automaticamente uma chave para você. Essa chave é usada com tabelas criptografadas que são criados no futuro. O AWS KMS integra hardware e software seguros e altamente disponíveis para oferecer um sistema de gerenciamento de chaves escalonado para a nuvem.

Para obter mais informações sobre gerenciamento de permissões do AWSCMK gerenciada pela, consulte [Autorizar o uso do AWSCMK gerenciada pelo AWS Key Management Service](#) Guia do desenvolvedor.

CMK gerenciada pelo cliente

Os CMKs gerenciados pelo cliente são CMKs em seu AWS Conta da que você cria, possui e gerencia. Você tem controle total sobre essas CMKs, inclusive para estabelecer e manter as políticas de chaves, as políticas do IAM e as concessões; habilitar e desabilitar as CMKs; alternar seu material de criptografia; adicionar tags; criar aliases que fazem referência a elas; e programar a exclusão delas. Para obter mais informações sobre como gerenciar as permissões de uma CMK gerenciada pelo cliente, consulte [Política de chaves da CMK gerenciada pelo cliente](#).

Ao especificar uma CMK gerenciada pelo cliente como a chave de criptografia no nível da tabela, a tabela do DynamoDB, os índices secundários local e global e os fluxos serão criptografados com a mesma CMK gerenciada pelo cliente. Os backups sob demanda são criptografados com a chave de criptografia no nível da tabela que é especificada no momento da criação do backup. A atualização da chave de criptografia no nível da tabela não altera a chave de criptografia associada aos backups sob demanda existentes.

A configuração do estado da CMK gerenciada pelo cliente como desativada ou sua programação para exclusão evita que todos os usuários e o serviço do DynamoDB do possam criptografar ou descriptografar dados e realizar operações de leitura e gravação na tabela. O DynamoDB deve ter acesso à sua chave de criptografia para garantir que você possa continuar acessando sua tabela e evitar a perda de dados.

Se você desabilitar a CMK gerenciada pelo cliente ou programá-la para exclusão, o status da tabela passará para Inaccessible (Inacessível). Para garantir que você possa continuar trabalhando com a tabela, é necessário conceder ao DynamoDB acesso à chave de criptografia especificada em até sete dias. Assim que o serviço detectar que a chave de criptografia está inacessível, o DynamoDB enviará uma notificação por e-mail para alertar você.

Note

- Se a CMK gerenciada pelo cliente permanecer inacessível para o serviço do DynamoDB por mais de sete dias, a tabela será arquivada e não poderá mais ser acessada. O DynamoDB cria um backup sob demanda da tabela e você a cobra. É possível usar esse backup sob demanda a fim de restaurar seus dados para uma nova tabela. Para iniciar a restauração, a última CMK gerenciada pelo cliente na tabela deve estar habilitada e o DynamoDB deve ter acesso a ela.
- Se o seu cliente gerenciou o CMK que foi usado para criptografar uma réplica de tabela global estiver inacessível, o DynamoDB removerá essa réplica do grupo de replicação. A réplica não será excluída e a replicação será interrompida de e para essa região, 20 horas após detectar o CMK gerenciado pelo cliente como inacessível.

Notas sobre como usar CMKs gerenciadas

O Amazon DynamoDB não consegue ler os dados da tabela, a menos que tenha acesso à CMK armazenada no AWS KMS conta. O DynamoDB usa criptografia de envelope e hierarquia de chaves para criptografar dados. A chave de criptografia do AWS KMS é usada para criptografar a chave raiz dessa hierarquia de chaves. Para obter mais informações, consulte [Criptografia de envelope](#) no AWS Key Management Service Guia do desenvolvedor.

Você pode usar o AWS CloudTrail e o Amazon CloudWatch Logs para rastrear as solicitações que o DynamoDB envia para o AWS KMS em seu nome. Para obter mais informações, consulte [Monitorar interação do DynamoDB com o AWS KMS](#) no AWS Key Management Service Guia do desenvolvedor.

O DynamoDB não chama o AWS KMS para cada operação do DynamoDB. A chave é atualizada uma vez a cada 5 minutos por conexão de cliente com tráfego ativo.

Verifique se configurou o SDK para reutilizar conexões. Caso contrário, você terá latências do DynamoDB tendo de reestabelecer novas entradas de cache para cada operação do DynamoDB. Além disso, pode ser que você tenha que enfrentar o AWS KMSe custos do CloudTrail. Por exemplo, para fazer isso usando o SDK de Node.js, você pode criar um novo agente HTTPS com `keepAlive` ativado. Para mais informações, consulte [Configurar maxSockets em Node.js](#) no Guia do desenvolvedor do AWS SDK for JavaScript.

Notas de uso de criptografia em repouso do DynamoDB

Considere o seguinte ao usar criptografia em repouso no Amazon DynamoDB.

Todos os dados da tabela são criptografados

A criptografia do lado do servidor em repouso está habilitada em todos os dados de tabela do DynamoDB e não pode ser desabilitada. Não é possível criptografar apenas um subconjunto de itens em uma tabela.

O DynamoDB criptografou todas as tabelas existentes que foram previamente descriptografadas usando oAWS CMK pertencidos a uma chave mestra de cliente.

A criptografia em repouso só criptografa dados enquanto eles estão estáticos (em repouso) em uma mídia de armazenamento persistente. Se a segurança dos dados for motivo de preocupação para dados em trânsito ou dados em uso, talvez seja necessário tomar outras medidas:

- Dados em trânsito: Todos os seus dados no DynamoDB são criptografados em trânsito (exceto os dados no DAX). Por padrão, as comunicações de e para o DynamoDB usam o protocolo HTTPS, que protege o tráfego de rede usando a criptografia Secure Sockets Layer (SSL) /Transport Layer Security (TLS).
- Dados em uso: Proteja seus dados antes de enviá-los ao DynamoDB usando criptografia no lado do cliente. Para obter mais informações, consulte[Criptografia do lado do cliente e do lado do servidor](#)noGuia do desenvolvedor do cliente de criptografia do Amazon DynamoDB.

Você pode usar fluxos com tabelas criptografadas. Os fluxos do DynamoDB são sempre criptografados com uma chave de criptografia no nível da tabela. Para obter mais informações, consulte[Captura de dados de alteração para DynamoDB Streams \(p. 651\)](#).

Os backups do DynamoDB são criptografados e a tabela que é restaurada de um backup também tem a criptografia habilitada. Você pode usar oAWS CMK de propriedade da,AWS CMK gerenciada pelo cliente ou a CMK gerenciada pelo cliente para criptografar seus dados de backup. Para obter mais informações, consulte[Backup e restauração sob demanda para o DynamoDB \(p. 691\)](#).

Os índices secundários locais e os índices secundários globais são criptografados usando a mesma chave da tabela base.

Tipos de criptografia

NoAWS Management Console, o tipo de criptografia éKMSquando você usa oAWS CMK gerenciada pelo cliente ou a CMK gerenciada pelo cliente para criptografar seus dados. O tipo de criptografia é DEFAULT quando você usa o CMK de propriedade da AWS. Na API do Amazon DynamoDB, o tipo de criptografia éKMSquando você usa oAWS CMK gerenciada pelo cliente. Na ausência do tipo de criptografia, seus dados são criptografados usando oAWS CMK de propriedade da. Você pode alternar entre oAWS CMK de propriedade da,AWS CMK gerenciada pelo cliente e a CMK gerenciada pelo cliente a qualquer momento. Você pode usar o console doAWS Command Line Interface(AWS CLI) ou a API do Amazon DynamoDB para alternar as chaves de criptografia.

Observe as seguintes limitações ao usar CMKs gerenciadas pelo cliente:

- Não é possível usar uma CMK gerenciada pelo cliente com clusters do DynamoDB Accelerator (DAX). Para obter mais informações, consulte[Criptografia DAX em repouso \(p. 815\)](#).
- É possível usar uma CMK gerenciada pelo cliente para criptografar tabelas que usam transações. No entanto, para garantir a durabilidade da propagação de transações, uma cópia da solicitação de transação é temporariamente armazenada pelo serviço e criptografada usando umAWS CMK de propriedade da. Os dados comprometidos em suas tabelas e índices secundários são sempre criptografados em repouso usando o CMK gerenciado pelo cliente.
- É possível usar uma CMK gerenciada pelo cliente para criptografar tabelas que usam o Contributor Insights. No entanto, os dados transmitidos ao Amazon CloudWatch são criptografados com umaAWS CMK de propriedade da.
- Se você desabilitar a CMK gerenciada pelo cliente ou programá-la para exclusão, qualquer dado no DynamoDB Streams ainda estará sujeito a uma vida útil de 24 horas. Qualquer dado de atividade não recuperado é elegível para remoção quando tiver mais de 24 horas.
- Se você desabilitar a CMK gerenciada pelo cliente ou programá-la para exclusão, as exclusões do Time to Live (TTL) continuarão por 30 minutos. Essas exclusões de TTL continuarão a ser emitidas para DynamoDB Streams e estarão sujeitas ao intervalo de retenção/remoção padrão.

Gerenciamento das tabelas criptografadas no DynamoDB

Você pode usar oAWS Management Consoleou oAWS Command Line Interface(AWS CLI) para especificar a chave de criptografia em novas tabelas e atualizar as chaves de criptografia em tabelas existentes no Amazon DynamoDB.

Tópicos

- [Especificar a chave de criptografia para uma nova tabela \(p. 878\)](#)
- [Atualização de uma chave de criptografia \(p. 880\)](#)

Especificar a chave de criptografia para uma nova tabela

Siga estas etapas para especificar a chave de criptografia em uma nova tabela usando o console do Amazon DynamoDB ou aAWS CLI.

Criar uma tabela criptografada (Console)

1. Faça login noAWS Management Consolee abra o console do DynamoDB em<https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Selecione Create Table (Criar tabela). Para o Nome da tabela, insira **Music**. Para a chave primária, insira **Artist**. Para a chave de classificação, insira **SongTitle**, os dois como strings.
4. Em Table settings (Configurações da tabela), certifique-se de que a opção Use default settings (Usar configurações padrão) não esteja selecionada.

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

Note

Se Usar configurações padrão estiver selecionada, as tabelas são criptografadas em repouso com oAWSCMK pertencido a chave mestra de cliente sem custo adicional.

5. Em Criptografia em repouso, escolha um tipo de criptografia:
 - Padrão:AWSCMK de propriedade da. A chave é pertencida pelo DynamoDB (sem custo adicional).
 - KMS— CMK gerenciada pelo cliente. A chave é armazenada na sua conta e você a cria, a detém e a gerencia (se aplicam cobranças do AWS KMS).
 - KMS:AWSCMK gerenciada pela. A chave é armazenada na sua conta e é gerenciada pelo AWS Key Management Service (se aplicam cobranças do AWS KMS).

Encryption At Rest

Select Server-side encryption settings for your DynamoDB table to help protect data at rest. [Learn more](#)

DEFAULT

The key is owned by Amazon DynamoDB. You are not charged any fee for using these CMKs.

KMS - Customer managed CMK

The key is stored in your account that you create, own, and manage. AWS Key Management Service (KMS) charges apply. [Learn more](#)

KMS - AWS managed CMK

The key is stored in your account and is managed by AWS Key Management Service (KMS). AWS KMS charges apply.

6. Selecione Create (Criar) para criar a tabela criptografada. Para confirmar o tipo de criptografia, verifique os detalhes da tabela na guia Overview (Visão geral).

Criar uma tabela criptografada (AWS CLI)

Usar a AWS CLI para criar uma tabela com o padrão AWSCMK de propriedade da, o AWSCMK gerenciado ou um CMK gerenciado pelo cliente para o Amazon DynamoDB.

Para criar uma tabela criptografada com o padrão do AWSCMK de propriedade da

- Crie a tabela Music criptografada conforme o seguinte.

```
aws dynamodb create-table \
--table-name Music \
--attribute-definitions \
    AttributeName=Artist,AttributeType=S \
    AttributeName=SongTitle,AttributeType=S \
--key-schema \
    AttributeName=Artist,KeyType=HASH \
    AttributeName=SongTitle,KeyType=RANGE \
--provisioned-throughput \
    ReadCapacityUnits=10,WriteCapacityUnits=5
```

Note

Essa tabela agora está criptografada usando o padrão do AWSCMK de propriedade na conta de serviço do DynamoDB.

Para criar uma tabela criptografada com o AWSCMK gerenciada pela DynamoDB

- Crie a tabela Music criptografada conforme o seguinte.

```
aws dynamodb create-table \
--table-name Music \
--attribute-definitions \
```

```
AttributeName=Artist,AttributeType=S \
AttributeName=SongTitle,AttributeType=S \
--key-schema \
    AttributeName=Artist,KeyType=HASH \
    AttributeName=SongTitle,KeyType=RANGE \
--provisioned-throughput \
    ReadCapacityUnits=10,WriteCapacityUnits=5 \
--sse-specification Enabled=true,SSEType=KMS
```

O status SSEDescription da descrição da tabela é definido como ENABLED, e o SSEType é KMS:

```
"SSEDescription": {
    "SSEType": "KMS",
    "Status": "ENABLED",
    "KMSMasterKeyArn": "arn:aws:kms:us-east-1:123456789012:key/abcd1234-abcd-1234-a123-
ab1234a1b234",
}
```

Como criar uma tabela criptografada com uma CMK gerenciada pelo cliente para o DynamoDB

- Crie a tabela Music criptografada conforme o seguinte.

```
aws dynamodb create-table \
    --table-name Music \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput \
        ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=abcd1234-abcd-1234-a123-
ab1234a1b234
```

O status SSEDescription da descrição da tabela é definido como ENABLED, e o SSEType é KMS:

```
"SSEDescription": {
    "SSEType": "KMS",
    "Status": "ENABLED",
    "KMSMasterKeyArn": "arn:aws:kms:us-east-1:123456789012:key/abcd1234-abcd-1234-a123-
ab1234a1b234",
}
```

Atualização de uma chave de criptografia

Você também pode usar o console do DynamoDB ou o AWS CLI para atualizar as chaves de criptografia de uma tabela existente entre um AWSCMK de propriedade da AWS e um CMK gerenciado pelo cliente a qualquer momento.

Atualizar uma chave de criptografia (console)

1. Faça login no AWS Management Console e abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Escolha a tabela que deseja atualizar e selecione a guia Overview (Visão Geral).
4. Escolha Manage Encryption (Gerenciar criptografia).

Table details

Table name	Music
Primary partition key	Artist (String)
Primary sort key	SongTitle (String)
Point-in-time recovery	ENABLED Disable
Encryption Type	DEFAULT Manage Encryption

5. Escolha um tipo de criptografia.

Manage Encryption

DEFAULT
The key is owned by Amazon DynamoDB. You are not charged any fee for using these CMKs.

KMS - Customer managed CMK
The key is stored in your account that you create, own, and manage. AWS Key Management Service (KMS) charges apply. [Learn more](#)

KMS - AWS managed CMK
The key is stored in your account and is managed by AWS Key Management Service (KMS). AWS KMS charges apply.

[Cancel](#) [Save](#)

- Padrão: AWSCMK de propriedade da. A chave é pertencida pelo DynamoDB (sem custo adicional).
- KMS— CMK gerenciada pelo cliente. A chave é armazenada na sua conta e você a cria, a detém e a gerencia (se aplicam cobranças do AWS KMS).
- KMS: AWSCMK gerenciada pela. A chave é armazenada na sua conta e é gerenciada pelo AWS Key Management Service (se aplicam cobranças do AWS KMS).

Depois escolha Save (Salvar) para atualizar a tabela criptografada. Para confirmar o tipo de criptografia, verifique os detalhes da tabela na guia Overview (Visão geral).

[Atualizar uma chave de criptografia \(AWS CLI\)](#)

Os exemplos a seguir mostram como atualizar uma tabela de criptografia usando a AWS CLI.

Para criar uma tabela criptografada com o padrão do AWSCMK de propriedade da

- Atualize a tabela Music, como o exemplo a seguir.

```
aws dynamodb update-table \  
  --table-name Music \  
  --sse-specification Enabled=false
```

Note

Essa tabela agora está criptografada usando o padrão do AWSCMK de propriedade na conta de serviço do DynamoDB.

Para atualizar uma tabela criptografada com o AWSCMK gerenciada pela DynamoDB

- Atualize a tabela `Music`, como o exemplo a seguir.

```
aws dynamodb update-table \  
  --table-name Music \  
  --sse-specification Enabled=true
```

O status SSEDescription da descrição da tabela é definido como `ENABLED`, e o SSEType é `KMS`:

```
"SSEDescription": {  
    "SSEType": "KMS",  
    "Status": "ENABLED",  
    "KMSMasterKeyArn": "arn:aws:kms:us-east-1:123456789012:key/abcd1234-abcd-1234-a123-  
ab1234a1b234",  
}
```

Para criar uma tabela criptografada com uma CMK gerenciada pelo cliente para o DynamoDB

- Atualize a tabela `Music`, como o exemplo a seguir.

```
aws dynamodb update-table \  
  --table-name Music \  
  --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=abcd1234-abcd-1234-a123-  
ab1234a1b234
```

O status SSEDescription da descrição da tabela é definido como `ENABLED`, e o SSEType é `KMS`:

```
"SSEDescription": {  
    "SSEType": "KMS",  
    "Status": "ENABLED",  
    "KMSMasterKeyArn": "arn:aws:kms:us-east-1:123456789012:key/abcd1234-abcd-1234-a123-  
ab1234a1b234",  
}
```

Proteção de dados no DynamoDB Accelerator

A criptografia do Amazon DynamoDB Accelerator (DAX) em repouso fornece uma camada adicional de proteção de dados, ajudando a proteger os dados contra acesso não autorizado ao armazenamento subjacente. O uso de criptografia em repouso para proteção de dados pode ser requerida por políticas organizacionais, regulamentações setoriais ou governamentais e exigências de conformidade. Você pode usar criptografia para aumentar a segurança dos dados dos aplicativos que são implantados na nuvem.

Para obter mais informações sobre a proteção de dados no DAX, consulte [Criptografia DAX em repouso](#) (p. 815).

Privacidade do tráfego entre redes

As conexões são protegidas entre o Amazon DynamoDB e os aplicativos no local, e entre o DynamoDB e outros AWS Recursos dentro do mesmo AWS Região:

Tráfego entre clientes de serviço e no local e os aplicativos

Você tem duas opções de conectividade entre sua rede privada e a AWS:

- Uma conexão AWS Site-to-Site VPN. Para obter mais informações, consulte [O que é o AWS Site-to-Site VPN?](#) no Guia do usuário do AWS Site-to-Site VPN.
- Uma conexão AWS Direct Connect. Para obter mais informações, consulte [O que é o AWS Direct Connect?](#) no Guia do usuário do AWS Direct Connect.

O acesso ao DynamoDB via rede é por meio do AWS APIs publicadas pela. Os clientes devem ter suporte ao Transport Layer Security (TLS) 1.0. Recomendamos TLS 1.2 ou posterior. Os clientes também devem ter suporte a pacotes de criptografia com sigilo de encaminhamento perfeito (PFS) como Ephemeral Diffie-Hellman (DHE) ou Ephemeral Elliptic Curve Diffie-Hellman (ECDHE). A maioria dos sistemas modernos como Java 7 e versões posteriores oferece suporte a esses modos. Além disso, você deve assinar solicitações usando um ID da chave de acesso e uma chave de acesso secreta associados a uma entidade principal do IAM. Ou você pode usar o [AWS Security Token Service \(STS\)](#) para gerar credenciais de segurança temporárias para assinar solicitações.

Tráfego entre recursos da AWS na mesma região

Um endpoint da Amazon Virtual Private Cloud (Amazon VPC) para DynamoDB é uma entidade lógica em uma VPC que permite conectividade apenas com o DynamoDB. A Amazon VPC roteia as solicitações para o DynamoDB e as respostas de volta para a VPC. Para obter mais informações, consulte [VPC endpoints](#) no Guia do usuário da Amazon VPC. Para obter políticas de exemplo que podem ser usadas para controlar o acesso de VPC endpoints, consulte [Uso de políticas do IAM para controlar o acesso ao DynamoDB](#).

Identity and Access Management

Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) para usar os recursos do Amazon DynamoDB. Você pode usar o AWS Identity and Access Management (IAM) para gerenciar as permissões de acesso e implementar políticas de segurança para o Amazon DynamoDB e o DynamoDB Accelerator (DAX).

Tópicos

- [Identity and Access Management no Amazon DynamoDB](#) (p. 883)
- [Identity and Access Management no DynamoDB Accelerator](#) (p. 915)

Identity and Access Management no Amazon DynamoDB

O acesso ao Amazon DynamoDB requer credenciais. Essas credenciais devem ter permissões para acessar AWS Recursos do, como uma tabela do Amazon DynamoDB ou uma instância do Amazon Elastic

Compute Cloud (Amazon EC2). As seguintes seções fornecem detalhes sobre como você pode usar o AWS Identity and Access Management(IAM) e o DynamoDB para ajudar a proteger o acesso aos seus recursos.

- [Authentication \(p. 884\)](#)
- [Controle de acesso \(p. 885\)](#)

Authentication

Você pode acessar a AWS como alguns dos seguintes tipos de identidade:

- AWS Usuário raiz da conta da— Quando você cria pela primeira vez um AWS, você começa com uma única identidade de login que tem acesso total a todos os AWS serviços e recursos da conta. Essa identidade é chamada de AWS account. O usuário raiz é acessado pelo login com o endereço de e-mail e a senha usados para criar a conta. Recomendamos que não use o usuário raiz para suas tarefas diárias, nem mesmo as administrativas. Em vez disso, siga as [práticas recomendadas para o uso do usuário raiz somente a fim de criar seu primeiro usuário do IAM](#). Depois, armazene as credenciais do usuário raiz com segurança e use-as para executar somente algumas tarefas de gerenciamento de contas e de serviços.
- Usuário do IAM— Um [usuário do IAM](#) é uma identidade dentro do seu AWS que tem permissões personalizadas específicas (por exemplo, permissões para criar uma tabela no DynamoDB). Você pode usar um nome de usuário e senha do IAM para fazer login e proteger o AWS. Páginas da web como o [AWS Management Console](#), [AWS Fóruns de discussão](#), ou o [AWS Support Centro](#).

Além de um nome e senha de usuário, você também pode gerar [chaves de acesso](#) para cada usuário. Você pode usar essas chaves ao acessar serviços da AWS de forma programática, seja com [um dos vários SDKs](#) ou usando a [AWS Command Line Interface \(CLI\)](#). As ferramentas de SDK e de CLI usam as chaves de acesso para o cadastramento criptográfico da sua solicitação. Se você não utilizar ferramentas da AWS, cadastre a solicitação você mesmo. DynamoDB é compatível com [Signature versão 4](#), um protocolo para autenticar solicitações de API de entrada. Para mais informações sobre autenticar solicitações, consulte [Processo de assinatura do Signature versão 4](#) no AWS Referência geral.

- Função do IAM: uma [função do IAM](#) é uma identidade do IAM que você pode criar em sua conta com permissões específicas. Uma função do IAM é semelhante a um usuário do IAM, pois é uma função do AWS Identity com políticas de permissões que determinam o que a identidade pode e não pode fazer no AWS. No entanto, em vez de ser exclusivamente associada a uma pessoa, uma função destina-se a ser assumida por qualquer pessoa que precisar dela. Além disso, uma função não tem credenciais de longo prazo padrão, como uma senha ou chaves de acesso, associadas a ela. Em vez disso, quando você assumir uma função, ela fornecerá credenciais de segurança temporárias para sua sessão de função. As funções do IAM com credenciais temporárias são úteis nas seguintes situações:
 - Acesso de usuário federado- Em vez de criar um usuário do IAM, é possível usar identidades existentes do AWS Directory Service, o diretório de usuário da sua empresa ou um provedor de identidades da web. Estes são conhecidos como usuários federados. A AWS atribui uma função a um usuário federado quando o acesso é solicitado por meio de um [provedor de identidades](#). Para obter mais informações sobre usuários federados, consulte [Usuários federados e funções](#) no Guia do usuário do IAM.
 - AWS Acesso a serviços da— Uma função de serviço é uma função do IAM role ([Função do IAM](#)). Que um serviço assume para realizar ações em seu nome. As funções de serviço fornecem acesso apenas dentro de sua conta e não podem ser usadas para conceder acesso a serviços em outras contas. Um administrador do IAM pode criar, modificar e excluir uma função de serviço do IAM. Para obter mais informações, consulte [Criar uma função para delegar permissões a um AWS serviço](#) no Guia do usuário do IAM.

- Aplicativos em execução no Amazon EC2—você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos em execução em uma instância do EC2 que fazem AWS CLI ou AWS Solicitações de API. É preferível fazer isso do que armazenar chaves de acesso na instância do EC2. Para atribuir uma função da AWS a uma instância do EC2 e disponibilizá-la para todos os seus aplicativos, crie um perfil de instância que esteja anexado à instância. Um perfil de instância contém a função e permite que programas que estão em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte [Usar uma função do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2](#) no Guia do usuário do IAM.

Controle de acesso

Você pode ter credenciais válidas para autenticar suas solicitações. No entanto, a menos que tenha permissões, você não pode criar nem acessar os recursos do Amazon DynamoDB. Por exemplo, você deve ter permissões para criar uma tabela do Amazon DynamoDB.

As seções a seguir descrevem como gerenciar permissões para o Amazon DynamoDB. Recomendamos que você leia a visão geral primeiro.

- [Visão geral do gerenciamento de acesso \(p. 885\)](#)
- [Uso de políticas baseadas em identidade \(políticas do IAM\) \(p. 888\)](#)
- [Referência de permissões da API do DynamoDB \(p. 898\)](#)
- [Uso de condições \(p. 899\)](#)

Visão geral do gerenciamento de permissões de acesso aos recursos do Amazon DynamoDB

Cada recurso da AWS é de propriedade de uma conta da AWS, e as permissões para criar ou acessar um recurso são regidas por políticas de permissões. Um administrador de conta pode anexar políticas de permissões a identidades do IAM (ou seja, usuários, grupos e funções), e alguns serviços (como o AWS Lambda) também oferecem suporte à anexação de políticas de permissões a recursos.

Note

Um administrador da conta (ou usuário administrador) é um usuário com privilégios de administrador. Para obter mais informações, consulte [Melhores práticas do IAM](#) no Guia do usuário do IAM.

Ao conceder permissões, você decide quem recebe as permissões, os recursos relacionados às permissões concedidas e as ações específicas que deseja permitir nesses recursos.

Tópicos

- [Recursos e operações do DynamoDB \(p. 885\)](#)
- [Entender a propriedade de recursos \(p. 886\)](#)
- [Gerenciamento do acesso aos recursos \(p. 886\)](#)
- [Especificação de elementos da política: Ações, efeitos e principais \(p. 887\)](#)
- [Especificação de condições em uma política \(p. 888\)](#)

Recursos e operações do DynamoDB

No DynamoDB, os principais recursos são Tabelas de. O DynamoDB também é compatível com outros tipos de recursosíndices, eStreams do. No entanto, você pode criar índices e streams somente no contexto de uma tabela existente do DynamoDB. Estes são chamados de sub-recursos.

Esses recursos e sub-recursos têm Nomes de recurso da Amazon (ARNs) exclusivos associados a eles, conforme mostrado na tabela a seguir.

Tipo de recurso	Formato de Nome de região da Amazon (ARN)
Tabela	<code>arn:aws:dynamodb:<i>region</i>:<i>account-id</i>:table/<i>table-name</i></code>
Índice	<code>arn:aws:dynamodb:<i>region</i>:<i>account-id</i>:table/<i>table-name</i>/index/<i>index-name</i></code>
Fluxo	<code>arn:aws:dynamodb:<i>region</i>:<i>account-id</i>:table/<i>table-name</i>/stream/<i>stream-label</i></code>

O DynamoDB fornece um conjunto de operações para trabalhar com recursos do DynamoDB. Para obter uma lista das operações disponíveis, consulte o Amazon DynamoDB [Ações](#).

Entender a propriedade de recursos

A conta da AWS possui os recursos criados na conta, independentemente de quem os criou. Especificamente, o proprietário do recurso é o AWS conta da [entidade principal](#) (isto é, o AWS Usuário raiz da conta, um usuário do IAM ou uma função do IAM) que autentica a solicitação de criação de recursos. Os exemplos a seguir ilustram como isso funciona:

- Se você usar seu AWS credenciais de usuário raiz da conta para criar uma tabela, sua AWS conta é a proprietária do recurso (no DynamoDB, o recurso é uma tabela).
- Se você criar um usuário do IAM no AWS conceder permissões para criar uma tabela para esse usuário, o usuário pode criar uma tabela. No entanto, sua AWS conta da, à qual o usuário pertence, é a proprietária do recurso de tabela.
- Se você criar uma função do IAM no AWS Com permissões para criar uma tabela, qualquer pessoa que puder assumir a função poderá criar uma tabela. Suas AWS conta da, à qual a função pertence, é a proprietária do recurso de tabela.

Gerenciamento do acesso aos recursos

A política de permissões descreve quem tem acesso a quê. A seção a seguir explica as opções disponíveis para a criação das políticas de permissões.

Note

Esta seção aborda o uso do IAM no contexto do DynamoDB. Não são fornecidas informações detalhadas sobre o serviço IAM. Para concluir a documentação do IAM, consulte [O que é o IAM?](#) no Guia do usuário do IAM. Para obter mais informações sobre a sintaxe as descrições da política do IAM [AWS Referência de política IAM](#) no Guia do usuário do IAM.

As políticas anexadas a uma identidade do IAM são conhecidas como políticas baseadas em identidade (políticas do IAM). As políticas anexadas a um recurso são chamadas de baseado em recursos do Políticas da AWS. O DynamoDB oferece suporte apenas às políticas baseadas em identidade (políticas do IAM).

Tópicos

- [Políticas baseadas em identidade \(políticas do IAM\) \(p. 887\)](#)
- [Políticas com base em recurso \(p. 887\)](#)

Políticas baseadas em identidade (políticas do IAM)

Você pode anexar as políticas a identidades do IAM. Por exemplo, você pode fazer o seguinte:

- Anexar uma política de permissões a um usuário ou grupo na sua conta— Para conceder a um usuário permissões para criar um recurso do Amazon DynamoDB, como uma tabela, você pode anexar uma política de permissões a um usuário ou grupo a que o usuário pertence.
- Anexar uma política de permissões a uma função (conceder permissões entre contas)— você pode associar uma política de permissões baseada em identidade a uma função do IAM para conceder permissões entre contas. Por exemplo, o administrador na conta A pode criar uma função para conceder permissões entre contas a outra conta da AWS (por exemplo, a conta B) ou a um serviço da AWS da seguinte forma:
 1. O administrador da Conta A cria uma função do IAM e anexa uma política de permissões à função que concede permissões a recursos na conta A.
 2. Um administrador da conta A anexa uma política de confiança à função identificando a conta B como a principal, que pode assumir a função.
 3. O administrador da conta B pode delegar permissões para assumir a função a todos os usuários na conta B. Isso permite que os usuários na conta B criem ou acessem recursos na conta A. O principal na política de confiança também pode ser um serviço principal da AWS, se você quiser conceder a um serviço da AWS permissões para assumir a função.

Para obter mais informações sobre como usar o IAM para delegar permissões, consulte [Gerenciamento de acesso](#) no Guia do usuário do IAM.

Veja a seguir um exemplo de política que concede permissões para uma ação do DynamoDB (`dynamodb:ListTables`). O caractere curinga (*) no `Resource` significa que você pode usar esta ação para obter os nomes de todas as tabelas de propriedade da AWS na conta atual AWS Região :

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "ListTables",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:ListTables"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

Para obter mais informações sobre políticas baseadas em identidade com o DynamoDB, consulte [Uso de políticas baseadas em identidade \(políticas do IAM\) com o Amazon DynamoDB](#) (p. 888). Para obter mais informações sobre usuários, grupos, funções e permissões, consulte [Identidades \(usuários, grupos e funções\)](#) no Guia do usuário do IAM.

Políticas com base em recurso

Outros serviços, como Amazon S3, também dão suporte a políticas de permissões baseadas em recursos. Por exemplo: você pode anexar uma política a um bucket do S3 para gerenciar permissões de acesso a esse bucket. O DynamoDB não é compatível com as políticas baseadas em recursos.

Especificação de elementos da política: Ações, efeitos e principais

Para cada recurso do DynamoDB, o serviço define um conjunto de operações da API. Para conceder permissões a essas operações de API, o DynamoDB define um conjunto de ações que podem ser

especificadas em uma política. Algumas operações da API podem exigir permissões para mais de uma ação a fim de realizar a operação da API. Para obter mais informações sobre recursos e operações da API, consulte [Recursos e operações do DynamoDB \(p. 885\)](#).

Estes são os elementos de política mais básicos:

- Recurso: use um nome de recurso da Amazon (ARN) para identificar o recurso ao qual a política se aplica. Para mais informações, consulte [Recursos e operações do DynamoDB \(p. 885\)](#).
- Ação: você usa palavras-chave de ação para identificar operações de recursos que você deseja permitir ou negar. Por exemplo, `dynamodb:Query` permite que o usuário execute o DynamoDB `Query` operação.
- Efeito— especifique o efeito, permita ou negue, quando o usuário solicita a ação específica. Se você não conceder (permitir) explicitamente acesso a um recurso, o acesso estará implicitamente negado. Você também pode negar explicitamente o acesso a um recurso, o que pode fazer para ter a certeza de que um usuário não consiga acessá-lo, mesmo que uma política diferente conceda acesso.
- Principal: em políticas baseadas em identidade (políticas do IAM), o usuário ao qual a política é anexada é implicitamente o principal. Para as políticas baseadas em recursos, você especifica quais usuários, contas, serviços ou outras entidades deseja que recebam permissões (aplica-se somente a políticas baseadas em recursos). O DynamoDB não é compatível com as políticas baseadas em recursos.

Para saber mais sobre a sintaxe da política do IAM e as descrições, consulte [AWS Referência de política IAM](#) no Guia do usuário do IAM.

Para um listando todas as operações da API do Amazon DynamoDB e os recursos aos quais elas se aplicam, consulte [Permissões da API do DynamoDB: Referência de ações, recursos e condições \(p. 898\)](#).

Especificação de condições em uma política

Ao conceder permissões, você pode usar a linguagem da política de acesso para especificar as condições quando uma política deve entrar em vigor. Por exemplo, convém que uma política só seja aplicada após uma data específica. Para obter mais informações sobre como especificar condições em uma linguagem de política, consulte [Condição](#) no Guia do usuário do IAM.

Para expressar condições, você usa chaves de condição predefinidas. Há AWS Chaves de condição para toda a e o DynamoDB: chaves específicas do que você pode usar conforme apropriado. Para obter uma lista completa de AWS-teclas largas, consulte [Chaves disponíveis para as condições](#) no Guia do usuário do IAM. Para obter uma lista completa de chaves específicas do DynamoDB, consulte [Uso de condições de política do IAM para controle de acesso refinado \(p. 899\)](#).

Uso de políticas baseadas em identidade (políticas do IAM) com o Amazon DynamoDB

Esse Tópico do Coberturas usando Baseado em identidade do AWS Identity and Access Management políticas (IAM) com o Amazon DynamoDB fornece exemplos. Exemplos de como um administrador de conta pode associar políticas de permissões a identidades do IAM (usuários, grupos e funções), assim, conceder permissões para realizar operações nos recursos do Amazon DynamoDB.

Important

Recomendamos analisar primeiro os tópicos introdutórios que explicam os conceitos básicos e as opções disponíveis para gerenciar o acesso aos recursos do DynamoDB. Para mais informações, consulte [Visão geral do gerenciamento de permissões de acesso aos recursos do Amazon DynamoDB \(p. 885\)](#).

As seções neste tópico abrangem o seguinte:

- Permissões do IAM necessárias para usar o console do Amazon DynamoDB (p. 889)
- AWSPolíticas gerenciadas pelo IAM (predefinidas) para o Amazon DynamoDB (p. 889)
- Exemplos de política gerenciada pelo cliente (p. 890)

Veja a seguir um exemplo de política de permissões.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "DescribeQueryScanBooksTable",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:DescribeTable",  
                "dynamodb:Query",  
                "dynamodb:Scan"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:account-id:table/Books"  
        }  
    ]  
}
```

A política anterior tem uma instrução que concede permissões para três ações do DynamoDB (`dynamodb:DescribeTable`, `dynamodb:Query`, e `dynamodb:Scan`) em uma tabela `nous-west-2` AWSRegião, que é propriedade do AWSconta especificada por**account-id**. OAmazon Resource Name (ARN)noResourcevalor especifica a tabelathatas permissões se aplicampara.

Permissões do IAM necessárias para usar o console do Amazon DynamoDB

Para trabalhar com o console do DynamoDB, um usuário deve ter um conjunto mínimo de permissões que lhe permita trabalhar com oAWSrecursos do DynamoDB da conta. Além dessas permissões do DynamoDB, o console exige permissões:

- Permissões do Amazon CloudWatch para exibir métricas e gráficos.
- AWS Data PipelinePermissões do para exportar e importar os dados do DynamoDB.
- Permissões do AWS Identity and Access Management para acessar funções necessárias para exportações e importações.
- Permissões do Amazon Simple Notification Service para notificá-lo sempre que um alerta do CloudWatch é acionado.
- AWS LambdaPermissões do para processar registros do DynamoDB Streams.

Se você criar uma política do IAM que seja mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para os usuários com essa política do IAM. Para garantir que esses usuários ainda consigam usar o console do DynamoDB, anexe também oAmazonDynamoDBReadOnlyAccess AWSPolítica gerenciada pela para o usuário do, conforme descrito em[AWSPolíticas gerenciadas pelo IAM \(predefinidas\) para o Amazon DynamoDB \(p. 889\)](#).

Não é necessário conceder permissões mínimas do console para os usuários que estão fazendo chamadas somente com aAWS CLIou a API do Amazon DynamoDB.

AWSPolíticas gerenciadas pelo IAM (predefinidas) para o Amazon DynamoDB

AWSA aborda alguns casos de uso comuns fornecendo políticas autônomas do IAM que são criadas e administradas peloAWS. EstesAWSAs políticas gerenciadas concedem as permissões necessárias para

casos de uso comuns. Assim, você não precisa investigar quais permissões são necessárias. Para obter mais informações, consulte [AWS Políticas gerenciadas pelo Guia do usuário do IAM](#).

O Seguir AWS As políticas gerenciadas pela, que você pode anexar aos usuários da sua conta, são específicas do DynamoDB e agrupadas por cenário de caso de uso:

- [AmazonDynamoDBReadOnlyAccess](#)— concede acesso somente leitura aos recursos do DynamoDB através do AWS Management Console.
- [AmazonDynamoDBFullAccess](#)— concede acesso total aos recursos do DynamoDB através do AWS Management Console.

Você pode revisar estes AWS Permissões gerenciadas políticas fazendo login no console do IAM e pesquisando políticas específicas.

Important

A prática recomendada é criar políticas personalizadas do IAM que concedem [Menos privilege](#) para os usuários, funções ou grupos do IAM que necessitam deles.

Exemplos de política gerenciada pelo cliente

Nesta seção, você pode encontrar exemplos de política que concedem permissões para diversas ações do DynamoDB. Essas políticas funcionam quando você usa o AWSSDKs ou o AWS CLI. Ao usar o console do, é necessário conceder permissões adicionais que são específicas para a consola. Para obter mais informações, consulte [Permissões do IAM necessárias para usar o console do Amazon DynamoDB \(p. 889\)](#).

Note

Todos os exemplos de política a seguir usam um dos AWS REgions e contêm IDs de conta e nomes de tabela fictícios.

Exemplos:

- [Política do IAM para conceder permissões a todas as ações do DynamoDB em uma tabela \(p. 891\)](#)
- [Política do IAM conceder permissões somente leitura em itens da em uma tabela do DynamoDB \(p. 891\)](#)
- [Política do IAM para conceder acesso a uma tabela específica do DynamoDB e seus índices \(p. 891\)](#)
- [Política do IAM para ler, gravar, atualizar e excluir acesso em uma tabela do DynamoDB \(p. 892\)](#)
- [Política do IAM para separar ambientes DynamoDB no mesmo AWS Conta \(p. 893\)](#)
- [Política do IAM para impedir a compra de capacidade reservada do DynamoDB \(p. 895\)](#)
- [Política do IAM para conceder acesso de leitura apenas para um stream do DynamoDB \(não para a tabela\) \(p. 896\)](#)
- [Política do IAM para permitir um AWS Lambda Função para acessar o DynamoDB Stream Records \(p. 897\)](#)
- [Política do IAM para acesso de leitura e gravação a um cluster do DynamoDB Accelerator \(DAX\) \(p. 898\)](#)

O Guia do usuário do IAM, inclui o [três exemplos adicionais do DynamoDB](#):

- [Amazon DynamoDB: Permite acesso a uma tabela específica](#)
- [Amazon DynamoDB: Permite acesso a colunas específicas](#)
- [Amazon DynamoDB: Permite acesso no nível de linha ao DynamoDB com base no ID do Amazon Cognito](#)

Política do IAM para conceder permissões a todas as ações do DynamoDB em uma tabela

A política a seguir concede permissões para todas as ações do DynamoDB em uma tabela chamada Books. O ARN de recurso especificado no arquivo Resource identifica uma tabela em uma AWS Região : Se você substituir o nome da tabela Books no Resource ARN com um caractere curinga (*), todas as ações do DynamoDB são permitidas para todas as tabelas na conta. Considere cuidadosamente as possíveis implicações de segurança antes de usar um caractere curinga nessa ou em qualquer política do IAM.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllAPIActionsOnBooks",  
            "Effect": "Allow",  
            "Action": "dynamodb:*",  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"  
        }  
    ]  
}
```

Note

Este é um exemplo do uso de um caractere curinga (*) para permitir que todos as ações, incluindo administração, operações de dados, monitoramento e aquisição de capacidade reservada do DynamoDB. Em vez disso, é uma prática recomendada especificar explicitamente cada ação a ser concedida e apenas o que esse usuário, função ou grupo precisa.

Política do IAM conceder permissões somente leitura em itens da em uma tabela do DynamoDB

A política de permissões a seguir concede permissões para o GetItem, BatchGetItem, Scan, Query, e ConditionCheckItem. Somente ações do DynamoDB e, como resultado, define o acesso somente leitura no Books Tabela do.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "ReadOnlyAPIActionsOnBooks",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:GetItem",  
                "dynamodb:BatchGetItem",  
                "dynamodb:Scan",  
                "dynamodb:Query",  
                "dynamodb:ConditionCheckItem"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"  
        }  
    ]  
}
```

Política do IAM para conceder acesso a uma tabela específica do DynamoDB e seus índices

A política a seguir concede ao permissões para ações de modificação de dados em uma tabela do DynamoDB chamada Bookse todos os índices dessa tabela. Para obter mais informações sobre como os índices funcionam, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AccessTableAllIndexesOnBooks",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:PutItem",  
                "dynamodb:UpdateItem",  
                "dynamodb:DeleteItem",  
                "dynamodb:BatchWriteItem",  
                "dynamodb:GetItem",  
                "dynamodb:BatchGetItem",  
                "dynamodb:Scan",  
                "dynamodb:Query",  
                "dynamodb:ConditionCheckItem"  
            ],  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:123456789012:table/Books",  
                "arn:aws:dynamodb:us-west-2:123456789012:table/Books/index/*"  
            ]  
        }  
    ]  
}
```

Política do IAM para ler, gravar, atualizar e excluir acesso em uma tabela do DynamoDB

Use esta política se precisar permitir que seu aplicativo crie, leia, atualize e exclua dados em tabelas, índices e streams do Amazon DynamoDB. Substituir oAWSNome da região, ID da conta eoNome da tabela ou caractere curinga (*), quando apropriado.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "DynamoDBIndexAndStreamAccess",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:GetShardIterator",  
                "dynamodb:Scan",  
                "dynamodb:Query",  
                "dynamodb:DescribeStream",  
                "dynamodb:GetRecords",  
                "dynamodb>ListStreams"  
            ],  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:123456789012:table/Books/index/*",  
                "arn:aws:dynamodb:us-west-2:123456789012:table/Books/stream/*"  
            ]  
        },  
        {  
            "Sid": "DynamoDBTableAccess",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:BatchGetItem",  
                "dynamodb:BatchWriteItem",  
                "dynamodb:ConditionCheckItem",  
                "dynamodb:PutItem",  
                "dynamodb:DescribeTable",  
                "dynamodb:DeleteItem",  
                "dynamodb:GetItem",  
                "dynamodb:Scan",  
                "dynamodb:UpdateItem"  
            ]  
        }  
    ]  
}
```

```
        "dynamodb:Query",
        "dynamodb:UpdateItem"
    ],
    "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
},
{
    "Sid": "DynamoDBDescribeLimitsAccess",
    "Effect": "Allow",
    "Action": "dynamodb:DescribeLimits",
    "Resource": [
        "arn:aws:dynamodb:us-west-2:123456789012:table/Books",
        "arn:aws:dynamodb:us-west-2:123456789012:table/Books/index/*"
    ]
}
}
```

Para expandir essa política para abranger todas as tabelas do DynamoDB em todas as AWS Regiões desta conta, use um curinga (*) para a região e nome da tabela. Por exemplo:

```
"Resource": [
    "arn:aws:dynamodb:*:123456789012:table/*",
    "arn:aws:dynamodb:*:123456789012:table/*/*"
]
```

Política do IAM para separar ambientes DynamoDB no mesmo AWS Conta

Suponha que você tenha ambientes separados, onde cada ambiente mantém sua própria versão de uma tabela chamada `ProductCatalog`. Se você criar dois `ProductCatalog` tabelas no mesmo AWS trabalho em um ambiente pode afetar o outro ambiente por causa da forma em que as permissões são configuradas. Por exemplo, as cotas no número de operações simultâneas de plano de controle (como `CreateTable`) são definidos no AWS nível de conta.

Como resultado, cada ação em um ambiente reduz o número de operações disponíveis em outro ambiente. Há também um risco de que o código em um ambiente pode acessar acidentalmente tabelas no outro ambiente.

Note

Se você quiser separar cargas de trabalho de produção e teste para ajudar a controlar o potencial “raio de explosão” de um evento, a prática recomendada é criar AWS contas para cargas de trabalho de teste e produção. Para obter mais informações, consulte [AWS Gerenciamento e separação de contas](#).

Suponha também que você tenha dois desenvolvedores, Amit e Alice, que estão testando o `ProductCatalog` tabela do. Em vez de cada desenvolvedor exigir um AWS Conta do, seus desenvolvedores podem compartilhar o mesmo teste AWS conta. Nesta conta de teste, você pode criar uma cópia da mesma tabela para cada desenvolvedor trabalhar, como `Alice_ProductCatalog` e `Amit_ProductCatalog`. Nesse caso, você pode criar usuários do IAM, Alice e Amit, no AWS que você criou para o ambiente de teste. Em seguida, você pode conceder permissões para esses usuários executarem ações do DynamoDB nas tabelas que eles possuem.

Para conceder essas permissões de usuário do IAM, é possível executar uma das seguintes ações:

- Crie uma política separada para cada usuário e, em seguida, anexe cada política ao seu usuário separadamente. Por exemplo, você pode anexar a política a seguir ao usuário Alice para permitir que ela acesse ações do DynamoDB na `Alice_ProductCatalog` Tabela:

```
{
    "Version": "2012-10-17",
```

```

"Statement": [
    {
        "Sid": "AllAPIActionsOnAliceTable",
        "Effect": "Allow",
        "Action": [
            "dynamodb>DeleteItem",
            "dynamodb>DescribeContributorInsights",
            "dynamodb>RestoreTableToPointInTime",
            "dynamodb>ListTagsOfResource",
            "dynamodb>CreateTableReplica",
            "dynamodb>UpdateContributorInsights",
            "dynamodb>CreateBackup",
            "dynamodb>DeleteTable",
            "dynamodb>UpdateTableReplicaAutoScaling",
            "dynamodb>UpdateContinuousBackups",
            "dynamodb>TagResource",
            "dynamodb>DescribeTable",
            "dynamodb>GetItem",
            "dynamodb>DescribeContinuousBackups",
            "dynamodb>BatchGetItem",
            "dynamodb>UpdateTimeToLive",
            "dynamodb>BatchWriteItem",
            "dynamodb>ConditionCheckItem",
            "dynamodb>UntagResource",
            "dynamodb>PutItem",
            "dynamodb>Scan",
            "dynamodb>Query",
            "dynamodb>UpdateItem",
            "dynamodb>DeleteTableReplica",
            "dynamodb>DescribeTimeToLive",
            "dynamodb>RestoreTableFromBackup",
            "dynamodb>UpdateTable",
            "dynamodb>DescribeTableAutoScaling",
            "dynamodb>GetShardIterator",
            "dynamodb>DescribeStream",
            "dynamodb>GetRecords",
            "dynamodb>DescribeLimits",
            "dynamodb>ListStreams"
        ],
        "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/
Alice_ProductCatalog/*"
    }
]
}

```

Em seguida, você pode criar uma política semelhante com um recurso diferente (`oAmit_ProductCatalog`) para o usuário Amit.

- Em vez de anexar políticas a usuários individuais, você pode usar variáveis de política do IAM para gravar uma única política e anexá-la a um grupo. Você precisa criar um grupo e, para este exemplo, adicionar ambos os usuários Alice e Amit ao grupo. O exemplo a seguir concede permissões para executar todas as ações do DynamoDB no `#{aws:username}_ProductCatalogTabela`. A variável de política `#{aws:username}` é substituída pelo nome de usuário do solicitante quando a política é avaliada. Por exemplo, se Alice envia uma solicitação para adicionar um item, a ação é permitida apenas se Alice estiver adicionando itens à tabela `Alice_ProductCatalog`.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ActionsOnUserSpecificTable",
            "Effect": "Allow",
            "Action": [
                "dynamodb>PutItem",

```

```
        "dynamodb:UpdateItem",
        "dynamodb>DeleteItem",
        "dynamodb:BatchWriteItem",
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Scan",
        "dynamodb:Query",
        "dynamodb:ConditionCheckItem"
    ],
    "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/
${aws:username}_ProductCatalog"
},
{
    "Sid": "AdditionalPrivileges",
    "Effect": "Allow",
    "Action": [
        "dynamodb>ListTables",
        "dynamodb>DescribeTable",
        "dynamodb>DescribeContributorInsights"
    ],
    "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/*"
}
]
```

Note

Ao usar variáveis de política do IAM, você deve especificar explicitamente o 2012-10-17 da versão do IAM linguagem de políticas na política. A versão padrão do IAM linguagem de políticas (2008-10-17) não é compatível com as variáveis de política.

Em vez de identificar uma tabela específica como um recurso do que normalmente faria, você poderia usar um caractere curinga (*) para conceder permissões em todas as tabelas em que o nome da tabela é prefixado com o usuário do IAM que está fazendo a solicitação, conforme mostrado no exemplo a seguir.

```
"Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/${aws:username}_*"
```

Política do IAM para impedir a compra de capacidade reservada do DynamoDB

Com a capacidade reservada do Amazon DynamoDB, você paga uma taxa única antecipada e se compromete a pagar por um nível mínimo de utilização com uma economia significativa ao longo de um período. Você pode usar o AWS Management Console para visualizar e comprar capacidade reservada. No entanto, talvez você não queira que todos os usuários em sua organização possam comprar capacidade reservada. Para mais informações sobre a capacidade reservada, consulte [Definição de preços do Amazon DynamoDB](#).

O DynamoDB fornece as seguintes operações de API para controlar o acesso ao gerenciamento de capacidade reservada:

- `dynamodb:DescribeReservedCapacity`— retorna as compras de capacidade reservada que estão em vigor no momento.
- `dynamodb:DescribeReservedCapacityOfferings`— retorna detalhes sobre os planos de capacidade reservada que são oferecidos no momento pelo AWS.
- `dynamodb:PurchaseReservedCapacityOfferings`— executa uma compra real de capacidade reservada.

O AWS Management Console usa essas ações de API para exibir as informações de capacidade reservada e para fazer compras. Você não pode chamar essas operações de um programa aplicativo, pois

elas podem ser acessadas somente pelo console. No entanto, é possível permitir ou negar o acesso a essas operações em uma política de permissões do IAM.

A política a seguir permite que os usuários visualizem compras e ofertas de capacidade reservada, usando o AWS Management Console—mas novas compras são negadas.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowReservedCapacityDescriptions",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:DescribeReservedCapacity",  
                "dynamodb:DescribeReservedCapacityOfferings"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:/*"  
        },  
        {  
            "Sid": "DenyReservedCapacityPurchases",  
            "Effect": "Deny",  
            "Action": "dynamodb:PurchaseReservedCapacityOfferings",  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:/*"  
        }  
    ]  
}
```

Observe que esta política usa o caractere curinga (*) para permitir permissões de descrição para todos os, e para negar a aquisição da capacidade reservada do DynamoDB para todos.

Política do IAM para conceder acesso de leitura apenas para um stream do DynamoDB (não para a tabela)

Quando você ativa o DynamoDB Streams em uma tabela, as informações são capturadas sobre todas as modificações em itens na tabela. Para mais informações, consulte [Captura de dados de alteração para DynamoDB Streams \(p. 651\)](#).

Em alguns casos, talvez você queira evitar que um aplicativo leia dados de uma tabela do DynamoDB, mas ainda permitir o acesso aos streams dessa tabela. Por exemplo, você pode configurar AWS Lambda para analisar um stream do e invocar uma função do Lambda quando as atualizações de itens forem detectadas e, em seguida, executar processamento adicional.

As ações a seguir estão disponíveis para controlar o acesso ao Streams do DynamoDB:

- dynamodb:DescribeStream
- dynamodb:GetRecords
- dynamodb:GetShardIterator
- dynamodb>ListStreams

O exemplo de política a seguir concede ao usuário permissões para acessar os streams de uma tabela chamada GameScores. O caractere curinga (*) no ARN corresponde a qualquer stream associado a essa tabela.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowGameScoresStreams",  
            "Effect": "Allow",  
            "Action": "dynamodb:DescribeStream",  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:GameScores/*"  
        }  
    ]  
}
```

```
        "Sid": "AccessGameScoresStreamOnly",
        "Effect": "Allow",
        "Action": [
            "dynamodb:DescribeStream",
            "dynamodb:GetRecords",
            "dynamodb:GetShardIterator",
            "dynamodb>ListStreams"
        ],
        "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores/stream/*"
    }
}
```

Observe que essa política concede acesso ao Game Scores Fluxos da tabela, mas não para a tabela em si.

Política do IAM para permitir um AWS Lambda Função para acessar o DynamoDB Stream Records

Se você deseja que determinadas ações sejam executadas com base em eventos em um stream do DynamoDB, é possível gravar um AWS Lambda Função do que é acionada por esses eventos. Uma função do Lambda como essa precisa de permissões para ler dados de um stream do DynamoDB. Para obter mais informações sobre como usar o Lambda com DynamoDB Streams, consulte [DynamoDB Streams e AWS Lambda Gatilhos \(p. 672\)](#).

Para conceder permissões ao Lambda, use a política de permissões que está associada à função do IAM do Lambda (também saiba [Como função de execução](#)). Especifique essa política ao criar a função do Lambda.

Por exemplo, você pode associar a política de permissões a seguir a uma função de execução para conceder as permissões ao Lambda para realizar as ações do DynamoDB Streams listadas.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLambdaFunctionInvocation",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "APIAccessForDynamoDBStreams",
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetRecords",
                "dynamodb:GetShardIterator",
                "dynamodb:DescribeStream",
                "dynamodb>ListStreams"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores/streams/*"
        }
    ]
}
```

Para obter mais informações, consulte [AWS Lambda permissions](#) no AWS Lambda Guia do desenvolvedor.

Política do IAM para acesso de leitura e gravação a um cluster do DynamoDB Accelerator (DAX)

A política seguinteO permite acesso de leitura, gravação, atualização e exclusão a umAcelerador do DynamoDB(DAX), mas não para a tabela do DynamoDB associada. Para usar esta política, substitua oAWSO nome da região, o ID da conta e o nome doDAXCluster do.

Note

Esta política fornece acesso aoCluster do DAX, mas não para oAssociatedTabela do DynamoDB. Certifique-se de quethatseu cluster DAX tem a política correta para executar essas mesmas operações na tabela do DynamoDB em seu nome.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AmazonDynamoDBDAXDataOperations",  
            "Effect": "Allow",  
            "Action": [  
                "dax:GetItem",  
                "dax:PutItem",  
                "dax:ConditionCheckItem",  
                "dax:BatchGetItem",  
                "dax:BatchWriteItem",  
                "dax>DeleteItem",  
                "dax:Query",  
                "dax:UpdateItem",  
                "dax:Scan"  
            ],  
            "Resource": "arn:aws:dax:eu-west-1:123456789012:cache/MyDAXCluster"  
        }  
    ]  
}
```

Para expandir esta política para abrangerDAXPara todos osAWSRegiões de uma conta, use um caractere curinga (*) para o nome da região.

```
"Resource": "arn:aws:dax:*:123456789012:cache/MyDAXCluster"
```

Permissões da API do DynamoDB: Referência de ações, recursos e condições

Quando você estiver configurandoControle de acesso (p. 885)Ao escrever uma política de permissões que pode ser anexada a uma identidade do IAM (políticas baseadas em identidade), você pode usar a lista deAções, recursos e chaves de condição do Amazon DynamoDBnoGuia do usuário do IAMComo referência do. A página lista cada operação da API do DynamoDB, as ações correspondentes às quais você pode conceder permissões para realizar a ação e oAWSO recurso da para o qual você pode conceder as permissões. Você especifica as ações no campo `Action` da política e o valor do recurso no campo `Resource` da política.

Você pode usar oAWS- chaves de condição em toda a nas suas políticas do DynamoDB para expressar condições. Para obter uma lista completa deAWS- teclas largas, consulte aReferência de elementos de política IAM JSONnoGuia do usuário do IAM.

Além doAWSCom chaves de condição no âmbito do, o DynamoDB tem suas próximas chaves específicas que você pode usar nas condições. Para mais informações, consulte Uso de condições de política do IAM para controle de acesso refinado (p. 899).

Tópicos relacionados

- [Controle de acesso \(p. 885\)](#)
- [Uso de condições de política do IAM para controle de acesso refinado \(p. 899\)](#)

Uso de condições de política do IAM para controle de acesso refinado

Ao conceder permissões no DynamoDB, você pode especificar as condições que determinam como uma política de permissões entra em vigor.

Overview

No DynamoDB, você tem a opção de especificar as condições ao conceder permissões usando uma política do IAM ([consulte Controle de acesso \(p. 885\)](#)). Por exemplo, é possível:

- Conceder permissões a fim de permitir acesso somente leitura aos usuários para determinados itens e atributos em uma tabela ou um índice secundário.
- Conceder permissões para permitir somente acesso de gravação aos usuários para determinados atributos em uma tabela com base na identidade desse usuário.

No DynamoDB, você pode especificar as condições em uma política do IAM usando chaves de condição, como ilustrado no caso de uso na seção a seguir.

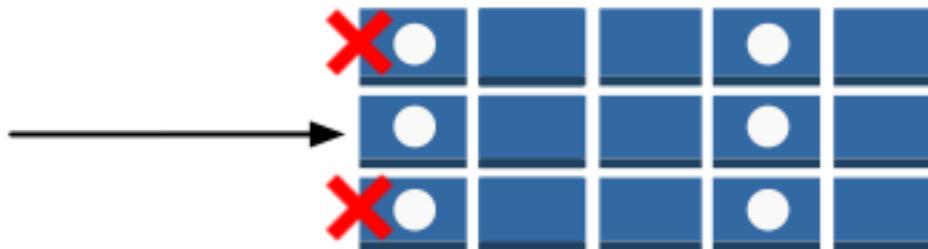
Note

Alguns AWSOs serviços da também oferecem suporte a condições com base em tag; no entanto, o DynamoDB não.

Caso de uso de permissões

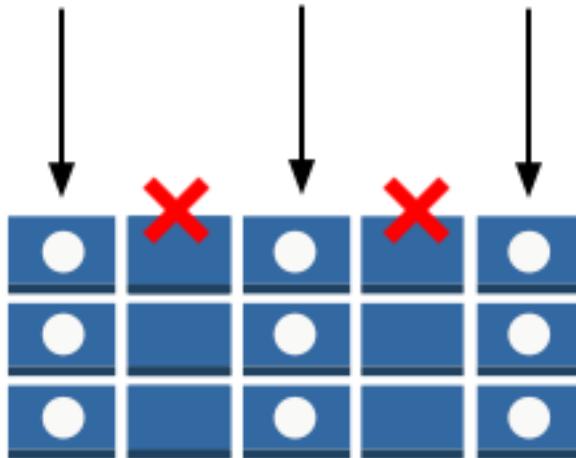
Além de controlar o acesso a ações de API do DynamoDB, você também pode controlar o acesso a itens de dados e atributos individuais. Por exemplo, você pode fazer o seguinte:

- Conceder permissões em uma tabela, mas restringir o acesso a itens específicos dessa tabela com base em determinados valores de chave primária. Um exemplo pode ser um aplicativo de rede social para jogos, onde todos os dados de jogos salvos dos usuários são armazenados em uma única tabela, mas nenhum usuário pode acessar itens de dados que não possui, como mostrado na ilustração a seguir:



- Oculte informações para que apenas um subconjunto de atributos fique visível para o usuário. Um exemplo pode ser um aplicativo que exibe os dados de voo para aeroportos próximos, com base na localização do usuário. Nomes de companhias aéreas e horas de partida, além de números de voo,

são exibidos. No entanto, atributos como nomes de piloto ou número de passageiros são ocultos, como mostrado na ilustração a seguir:



Para implementar esse tipo de controle de acesso refinado, grave uma política de permissões do IAM que especifica condições para acessar as credenciais de segurança e as permissões associadas. Em seguida, aplique a política aos usuários, grupos ou funções do IAM que você criar usando o console do IAM. Sua política do IAM pode restringir o acesso a cada item de uma tabela, aos atributos desses itens ou a ambas as coisas ao mesmo tempo.

Se preferir, você pode usar federação de identidades na web para controlar o acesso dos usuários que serão autenticados por meio do Login with Amazon, Facebook ou Google. Para mais informações, consulte [Uso de federação de identidades na web \(p. 909\)](#).

Use o elemento `Condition` do IAM para implementar uma política de controle de acesso refinada. Adição de um `Condition` para uma política de permissões, você pode permitir ou negar o acesso a itens e atributos em tabelas e índices do DynamoDB, com base em seus requisitos comerciais específicos.

Como um exemplo, considere um aplicativo de jogos móveis que permite que os jogadores selecionem e joguem uma variedade de jogos diferentes. O aplicativo usa uma tabela do DynamoDB chamada `GameScores` para manter o controle das pontuações máximas e outros dados do usuário. Cada item na tabela é exclusivamente identificado por um ID de usuário e o nome do jogo que o usuário jogou. A tabela `GameScores` tem uma chave primária que consiste em uma chave de partição (`UserID`) e a chave de classificação (`GameTitle`). Os usuários só têm acesso aos dados de jogos associados ao seu ID de usuário. Um usuário que deseja jogar um jogo deve pertencer a uma função do IAM chamada `GameRole`. O usuário tem uma política de segurança anexada a ele.

Para gerenciar permissões de usuário neste aplicativo, grave uma política de permissões, como a seguinte:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowAccessToOnlyItemsMatchingUserID",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:GetItem",  
                "dynamodb:BatchGetItem",  
                "dynamodb:Query",  
                "dynamodb:PutItem",  
                "dynamodb:UpdateItem"  
            ],  
            "Resource": "arn:aws:dynamodb:  
                [REDACTED]  
                :  
                [REDACTED]  
                /GameScores/  
                #UserID#/  
                #GameTitle#"  
        }  
    ]  
}
```

```
        "dynamodb:UpdateItem",
        "dynamodb>DeleteItem",
        "dynamodb:BatchWriteItem"
    ],
    "Resource": [
        "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores"
    ],
    "Condition": {
        "ForAllValues:StringEquals": {
            "dynamodb:LeadingKeys": [
                "${www.amazon.com:user_id}"
            ],
            "dynamodb:Attributes": [
                "UserId",
                "GameTitle",
                "Wins",
                "Losses",
                "TopScore",
                "TopScoreDateTime"
            ]
        },
        "StringEqualsIfExists": {
            "dynamodb:Select": "SPECIFIC_ATTRIBUTES"
        }
    }
}
]
```

Além de conceder permissões para ações específicas do DynamoDB (`Action`elemento) no `GameScoresTabela` (`Resource`elemento), o `Condition` elemento usa as chaves de condição a seguir específicas do DynamoDB que limitam as permissões da seguinte forma:

- `dynamodb:LeadingKeys`— esta chave de condição permite que os usuários acessem apenas os itens nos quais o valor de chave de partição coincide com seu ID de usuário. Este ID, `${www.amazon.com:user_id}`, é uma variável de substituição. Para obter mais informações sobre variáveis de substituição, consulte [Uso de federação de identidades na web \(p. 909\)](#).
- `dynamodb:Attributes`— esta chave de condição limita o acesso aos atributos especificados, para que apenas as ações listadas na política de permissões possam retornar valores para esses atributos. Além disso, a cláusula `StringEqualsIfExists` garante que o aplicativo sempre deve fornecer uma lista de atributos específicos nos quais atuar e que o aplicativo não pode solicitar todos os atributos.

Quando uma política do IAM é avaliada, o resultado é sempre verdadeiro (o acesso é permitido) ou falso (o acesso é negado). Se qualquer parte do elemento `Condition` é falsa, toda a política é avaliada como falsa e o acesso é negado.

Important

Caso você use `dynamodb:Attributes`, deve especificar os nomes de todos os atributos de chave de índice e chave primária da tabela, além de quaisquer índices secundários que estejam listados na política. Caso contrário, o DynamoDB não pode usar esses atributos de chave para executar a ação solicitada.

Os documentos de política do IAM podem conter apenas os seguintes caracteres Unicode: guia horizontal (U+0009), linefeeds (U+000A), retorno de carro (U+000D) e caracteres no intervalo U+0020 a U+00FF.

Especificação de condições: Uso de chaves de condição

AWS fornece um conjunto de chaves de condição predefinidas (AWS-chaves de condição amplas) para todos os AWS que oferecem suporte ao IAM para controle de acesso. Por exemplo, você pode usar a condição de chave `aws:SourceIp` para verificar o endereço IP do solicitante antes de permitir que uma

ação seja executada. Para obter mais informações e uma lista dos AWS-teclas largas, consulte[Chaves disponíveis para as condições](#)no Guia do usuário do IAM.

A tabela a seguir mostra as chaves de condição específicas do serviço DynamoDB que se aplicam ao DynamoDB.

Chave de condição do DynamoDB	Descrição
dynamodb:LeadingKeys	Representa o primeiro atributo de chave de uma tabela – em outras palavras, a chave de partição. O nome da chave LeadingKeys é no plural, mesmo se ela for usada com ações de um único item. Além disso, você deve usar o modificador <code>ForAllValues</code> ao utilizar LeadingKeys em uma condição.
dynamodb:Select	Representa o parâmetro <code>Select</code> de uma <code>Query</code> ou solicitação <code>Scan</code> . <code>Select</code> pode ter qualquer um dos seguintes valores: <ul style="list-style-type: none"> • <code>ALL_ATTRIBUTES</code> • <code>ALL_PROJECTED_ATTRIBUTES</code> • <code>SPECIFIC_ATTRIBUTES</code> • <code>COUNT</code>
dynamodb:Attributes	Representa uma lista dos nomes de atributo em uma solicitação, ou os atributos que são retornados de uma solicitação. Os valores de são nomeados da mesma forma e têm o mesmo significado que os parâmetros de determinadas ações de API do DynamoDB da, como mostrado a seguir: <ul style="list-style-type: none"> • <code>AttributesToGet</code> Usado por: <code>BatchGetItem</code>, <code>.GetItem</code>, <code>Query</code>, <code>Scan</code> • <code>AttributeUpdates</code> Usado por: <code>UpdateItem</code> • <code>Expected</code> Usado por: <code>DeleteItem</code>, <code>PutItem</code>, <code>UpdateItem</code> • <code>Item</code> Usado por: <code>PutItem</code> • <code>ScanFilter</code> Usado por: <code>Scan</code>
dynamodb:ReturnValue	Representa o parâmetro <code>ReturnValues</code> de uma solicitação. <code>ReturnValues</code> pode ter qualquer um dos seguintes valores: <ul style="list-style-type: none"> • <code>ALL_OLD</code> • <code>UPDATED_OLD</code> • <code>ALL_NEW</code> • <code>UPDATED_NEW</code> • <code>NONE</code>
dynamodb:ReturnConsumedCapacity	Representa o parâmetro <code>ReturnConsumedCapacity</code> de uma solicitação. <code>ReturnConsumedCapacity</code> pode ter um dos seguintes valores: <ul style="list-style-type: none"> • <code>TOTAL</code>

Chave de condição do DynamoDB	Descrição
	<ul style="list-style-type: none"> • NONE

Límite de acesso de usuário

Muitas políticas de permissões do IAM permitem que os usuários acessem esses itens em uma tabela, na qual o valor de chave de partição coincide com o identificador do usuário. Por exemplo, o aplicativo de jogo anterior limita o acesso dessa forma, para que os usuários possam acessar somente os dados do jogo que estão associados ao ID de usuário deles. As variáveis de substituição do IAM \${www.amazon.com:user_id}, \${graph.facebook.com:id}, \${accounts.google.com:sub} contêm identificadores de usuário para Login with Amazon, Facebook e Google. Para saber como um aplicativo se conecta a um desses provedores de identidade e obtém esses identificadores, consulte [Uso de federação de identidades na web \(p. 909\)](#).

Note

Cada um dos exemplos na seção a seguir define a cláusula `Effect` como `Allow` e especifica apenas as ações, os recursos e os parâmetros permitidos. O acesso é permitido apenas para o que está listado explicitamente na política do IAM.

Em alguns casos, é possível reescrever essas políticas para que elas sejam baseadas em negação (ou seja, definindo a cláusula `Effect` como `Deny` e invertendo toda a lógica na política). No entanto, recomendamos que você evite usar políticas baseadas em negação com o DynamoDB pois elas são difíceis de escrever corretamente, em comparação às políticas baseadas em permissão. Além disso, futuras alterações na API do DynamoDB (ou alterações nas entradas existentes da API) podem tornar ineficaz uma política baseada em negação.

Exemplos de políticas: Uso de condições para controle de acesso refinado

Esta seção mostra várias políticas para a implementação de um controle de acesso refinado em tabelas e índices do DynamoDB.

Note

Todos os exemplos usam a Região us-west-2 e contêm IDs de conta fictícios.

1: Conceder permissões que limitam o acesso a itens com um valor de chave de partição específico

A política de permissões a seguir concede ao usuário permissões que permitem um conjunto de ações do DynamoDB no GamesScoreTabela do. Ela usa a chave de condição `dynamodb:LeadingKeys` para limitar as ações do usuário apenas aos itens cujo valor da chave de partição `UserID` coincide com o ID do usuário único do Login with Amazon desse aplicativo.

Important

A lista de ações não inclui permissões para `Scan` porque `Scan` retorna todos os itens, independentemente das principais chaves.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessToUserItems",
            "Effect": "Allow",
            "Action": [
                "DynamoDB:PutItem",
                "DynamoDB:UpdateItem",
                "DynamoDB:DeleteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2::table/GamesScoreTabela/#/partitionKey=1"
            ],
            "Condition": {
                "dynamodb:LeadingKeys": "1"
            }
        }
    ]
}
```

```
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
    ],
    "Resource": [
        "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores"
    ],
    "Condition": {
        "ForAllValues:StringEquals": {
            "dynamodb:LeadingKeys": [
                "${www.amazon.com:user_id}"
            ]
        }
    }
]
}
```

Note

Ao usar variáveis de política, você deve especificar explicitamente a versão 2012-10-17 na política. A versão padrão da linguagem de políticas de acesso, 2008-10-17, não é compatível com as variáveis de política.

Para implementar o acesso somente leitura, você pode remover as ações que podem modificar os dados. Na política a seguir, somente as ações que fornecem acesso somente leitura são incluídas na condição.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadOnlyAccessToUserItems",
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:BatchGetItem",
                "dynamodb:Query"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": [
                        "${www.amazon.com:user_id}"
                    ]
                }
            }
        }
    ]
}
```

Important

Se você usar `dynamodb:Attributes`, você deve especificar os nomes de todos os atributos de chave de índice e chave primária da tabela, além de quaisquer índices secundários que estejam listados na política. Caso contrário, o DynamoDB não pode usar esses atributos de chave para executar a ação solicitada.

2: Conceder permissões que limitam o acesso a atributos específicos em uma tabela

A política de permissões a seguir permite o acesso apenas a dois atributos em uma tabela, adicionando a chave de condição dynamodb:Attributes. Esses atributos podem ser lidos, gravados ou avaliados em um filtro de verificação ou gravação condicional.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "LimitAccessToSpecificAttributes",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:UpdateItem",  
                "dynamodb:GetItem",  
                "dynamodb:Query",  
                "dynamodb:BatchGetItem",  
                "dynamodb:Scan"  
            ],  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores"  
            ],  
            "Condition": {  
                "ForAllValues:StringEquals": {  
                    "dynamodb:Attributes": [  
                        "UserId",  
                        "TopScore"  
                    ]  
                },  
                "StringEqualsIfExists": {  
                    "dynamodb:Select": "SPECIFIC_ATTRIBUTES",  
                    "dynamodb:ReturnValues": [  
                        "NONE",  
                        "UPDATED_OLD",  
                        "UPDATED_NEW"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

Note

A política utiliza umLista de permissõesabordagem, o que permite o acesso a um conjunto nomeado de atributos. Mas você pode escrever uma política equivalente que nega o acesso a outros atributos. Nós não recomendamos issoLista de negaçãoAbordagem. Os usuários podem determinar os nomes desses atributos negados seguindo aPrincípio do privilégio mínimo, como explicado na Wikipédia emhttp://en.wikipedia.org/wiki/Principle_of_least_privilege, e use umLista de permissõesPara enumerar todos os valores permitidos, em vez de especificar os atributos negados.

Essa política não permite PutItem, DeleteItem ou BatchWriteItem. Essas ações sempre substituem o item anterior inteiro, o que permitira aos usuários excluir os valores anteriores que eles não tem permissão para acessar.

A cláusula StringEqualsIfExists na política de permissões garante o seguinte:

- Se o usuário especifica o parâmetro Select, seu valor deve ser SPECIFIC_ATTRIBUTES. Essa exigência impede que a ação da API retorne quaisquer atributos que não sejam permitidos, tal como de uma projeção do índice.

- Se o usuário especifica o parâmetro `ReturnValues`, então seu valor deve ser `NONE`, `UPDATED_OLD` ou `UPDATED_NEW`. Isso é necessário porque a ação `UpdateItem` também executa operações de leitura implícitas para verificar se um item existe antes de substituí-lo, e de forma que os valores de atributo anteriores possam ser retornados, se solicitado. Restringir `ReturnValues` dessa forma garante que os usuários possam apenas ler ou gravar atributos permitidos.
- A cláusula `StringEqualsIfExists` garante que apenas um desses parâmetros – `Select` ou `ReturnValues` – pode ser usado por solicitação, no contexto das ações permitidas.

Veja a seguir algumas variações desta política:

- Para permitir apenas as ações de leitura, você pode remover `UpdateItem` da lista de ações permitidas. Como nenhuma das ações restantes aceitam `ReturnValues`, você pode remover `ReturnValues` da condição. Também é possível alterar `StringEqualsIfExists` para `StringEquals`, pois o parâmetro `Select` sempre tem um valor (`ALL_ATTRIBUTES`, a menos que especificado de outra forma).
- Para permitir apenas as ações de gravação, você pode remover tudo menos `UpdateItem` da lista de ações permitidas. Como `UpdateItem` não usa o parâmetro `Select`, você pode remover `Select` da condição. Você também tem que alterar `StringEqualsIfExists` para `StringEquals` porque o parâmetro `ReturnValues` sempre tem um valor (`NONE`, a menos que especificado de outra forma).
- Para permitir todos os atributos cujo nome corresponde a um padrão, use `StringLike` em vez de `StringEquals`, e use um caractere curinga de correspondência de padrão multicaractere (*).

3: Conceder permissões para evitar atualizações em determinadas atributos

A política de permissões a seguir limita o acesso do usuário à atualização apenas dos atributos identificados pela chave de condição `dynamodb:Attributes`. A condição `StringNotLike` impede que um aplicativo atualize os atributos especificados usando a condição de chave `dynamodb:Attributes`.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "PreventUpdatesOnCertainAttributes",  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:UpdateItem"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",  
            "Condition": {  
                "ForAllValues:StringNotLike": {  
                    "dynamodb:Attributes": [  
                        "FreeGamesAvailable",  
                        "BossLevelUnlocked"  
                    ]  
                },  
                "StringEquals": {  
                    "dynamodb:ReturnValues": [  
                        "NONE",  
                        "UPDATED_OLD",  
                        "UPDATED_NEW"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

Observe o seguinte:

- A ação `UpdateItem`, como outras ações de gravação, exige acesso de leitura aos itens para que possa retornar valores antes e depois da atualização. Na política, é possível limitar a ação para acessar somente os atributos que podem ser atualizados, especificando a chave de condição `dynamodb:ReturnValues`. A chave de condição restringe `ReturnValues` na solicitação para especificar apenas `NONE`, `UPDATED_OLD` ou `UPDATED_NEW` e não inclui `ALL_OLD` ou `ALL_NEW`.
- As ações `PutItem` e `DeleteItem` substituem um item inteiro e, assim, permite que os aplicativos modifiquem quaisquer atributos. Portanto, ao limitar um aplicativo para atualizar somente atributos específicos, você não deve conceder permissão para essas APIs.

[4: Conceder permissões para consultar somente atributos projetados em um índice](#)

A política de permissões a seguir permite consultas em um índice secundário (`TopScoreDateTimeIndex`) usando `dynamodb:AttributesChave` de condição. A política também limita as consultas para solicitar somente atributos específicos que foram projetados no índice.

Para solicitar que o aplicativo especifique uma lista de atributos na consulta, a política também especifica `dynamodb>Select` para exigir que a chave de condição `Select` parâmetro do `DynamoDBQuery` ação `SPECIFIC_ATTRIBUTES`. A lista de atributos é limitada a uma lista específica que é fornecida por meio da chave de condição `dynamodb:Attributes`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QueryOnlyProjectedIndexAttributes",
            "Effect": "Allow",
            "Action": [
                "dynamodb:Query"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores/index/TopScoreDateTimeIndex"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:Attributes": [
                        "TopScoreDateTime",
                        "GameTitle",
                        "Wins",
                        "Losses",
                        "Attempts"
                    ]
                },
                "StringEquals": {
                    "dynamodb:Select": "SPECIFIC_ATTRIBUTES"
                }
            }
        }
    ]
}
```

A política de permissões a seguir é semelhante, mas a consulta deve solicitar todos os atributos que foram projetados no índice.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QueryAllIndexAttributes",
            "Effect": "Allow",
            "Action": [
                "dynamodb:Query"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores/index/*"
            ]
        }
    ]
}
```

```
        "Action": [
            "dynamodb:Query"
        ],
        "Resource": [
            "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores/index/
TopScoreDateTimeIndex"
        ],
        "Condition": {
            "StringEquals": {
                "dynamodb:Select": "ALL_PROJECTED_ATTRIBUTES"
            }
        }
    }
}
```

5: Conceder permissões para limitar o acesso a determinados atributos e valores de chave de partição

A política de permissões a seguir permite ações específicas do DynamoDB (especificadas noActionElement) em uma tabela e em um índice de tabela (especificadas noResourceElement). A política usa a chave de condição dynamodb:LeadingKeys para restringir as permissões apenas aos itens cujo valor de chave de partição corresponde ao ID do usuário no Facebook

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LimitAccessToCertainAttributesAndKeyValues",
            "Effect": "Allow",
            "Action": [
                "dynamodb:UpdateItem",
                "dynamodb:GetItem",
                "dynamodb:Query",
                "dynamodb:BatchGetItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
                "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores/index/
TopScoreDateTimeIndex"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": [
                        "${graph.facebook.com:id}"
                    ],
                    "dynamodb:Attributes": [
                        "attribute-A",
                        "attribute-B"
                    ]
                },
                "StringEqualsIfExists": {
                    "dynamodb:Select": "SPECIFIC_ATTRIBUTES",
                    "dynamodb:ReturnValues": [
                        "NONE",
                        "UPDATED_OLD",
                        "UPDATED_NEW"
                    ]
                }
            }
        }
    ]
}
```

Observe o seguinte:

- As ações de gravação permitidas pela política (`UpdateItem`) só podem modificar `attribute-A` ou `attribute-B`.
- Como a política permite `UpdateItem`, um aplicativo pode inserir novos itens, e os atributos ocultos serão nulos em novos itens. Se esses atributos estiverem projetados em `TopScoreDateTimeIndex`, a política terá o benefício adicional de impedir consultas que causem buscas na tabela.
- Os aplicativos não podem ler quaisquer atributos que não estejam listados em `dynamodb:Attributes`. Com essa política implantada, um aplicativo deve definir o `Select` parâmetro para `SPECIFIC_ATTRIBUTE` sem solicitações de leitura, e somente atributos na lista de permissões podem ser solicitados. Para solicitações de gravação, o aplicativo não pode definir `ReturnValues` como `ALL_OLD` ou `ALL_NEW` e ele não pode executar operações de gravação condicional com base em outros atributos.

Tópicos relacionados

- [Controle de acesso \(p. 885\)](#)
- [Permissões da API do DynamoDB: Referência de ações, recursos e condições \(p. 898\)](#)

Uso de federação de identidades na web

Se você estiver criando um aplicativo direcionado a um grande número de usuários, você poderá, opcionalmente, usar a federação de identidades da web para autenticação e autorização. A federação de identidades da Web exclui a necessidade de criar usuários do IAM. Em vez disso, os usuários podem fazer login em um provedor de identidades e, então, obter credenciais de segurança temporárias do AWS Security Token Service (AWS STS). O aplicativo pode então usar essas credenciais para acessar AWS Serviços da .

A federação de identidades na web é compatível com os seguintes provedores de identidade:

- Login with Amazon
- Facebook
- Google

Recursos adicionais para federação de identidades na web

Os recursos a seguir podem ajudá-lo a saber mais sobre a federação de identidades na web:

- O post [Federação de identidades da web usando o AWS SDK for .NET](#) no AWSO blog de desenvolvedores ensina a usar a federação de identidades da web com o Facebook. Isso inclui snippets de código em C# que mostram como assumir uma função do IAM com identidade de web e usar credenciais de segurança temporárias para acessar um AWS Recurso do.
- O [AWS Mobile SDK for iOS](#) e a [AWS Mobile SDK for Android](#) contêm aplicativos de exemplo. Eles incluem código que mostra como chamar os provedores de identidade e, depois, como usar as informações desses provedores para obter e usar credenciais de segurança temporárias.
- O artigo [Web Identity Federation with Mobile Applications](#) discute a federação de identidades na web e mostra um exemplo de como usar a federação de identidades na web para acessar um recurso da AWS.

Política de exemplo para federação de identidades na web

Para mostrar como você pode usar a federação de identidades da web com o DynamoDB, revise a [GameScores](#) que foi introduzida no [Uso de condições de política do IAM para controle de acesso refinado \(p. 899\)](#). Esta é a chave primária para GameScores:

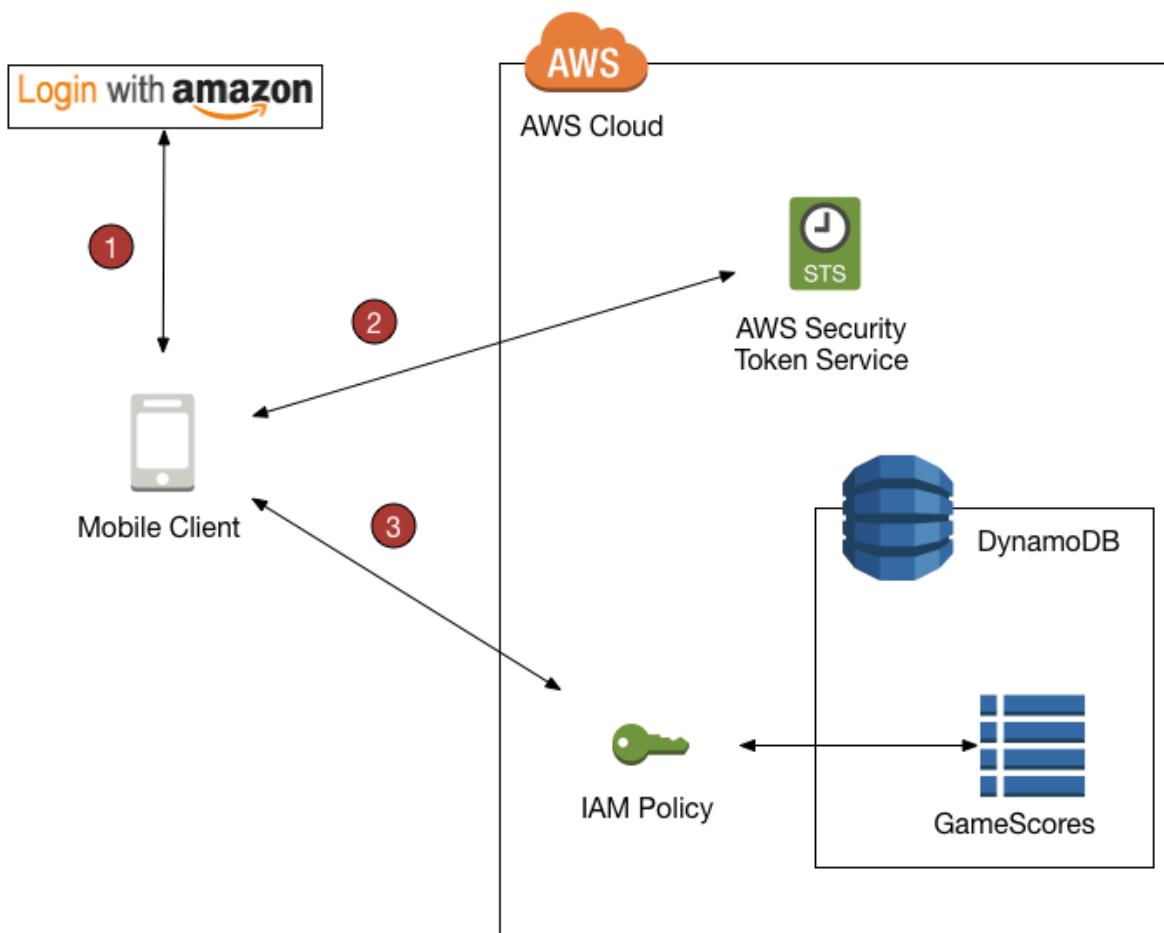
Nome da tabela	Tipo de chave primária	Nome e tipo de chave de partição	Nome e tipo de chave de classificação
GameScores (<u>UserId</u> , <u>GameTitle</u> , ...)	Composto	Nome do atributo: UserId Tipo: String	Nome do atributo: GameTitle Tipo: String

Agora, suponha que um aplicativo de jogos móveis use essa tabela, e que o aplicativo precise oferecer suporte a milhares, ou até mesmo milhões, de usuários. Nesta escala, torna-se muito difícil gerenciar usuários e aplicativos individuais e garantir que cada usuário possa acessar somente seus próprios dados na tabela GameScores. Felizmente, muitos usuários já têm contas com um provedor de identidade de terceiros, como o Facebook, o Google ou o Login with Amazon. Então faz sentido usar um desses provedores nas tarefas de autenticação.

Para fazer isso usando a federação de identidades na web, o desenvolvedor do aplicativo deve registrar o aplicativo com um provedor de identidade (como o Login with Amazon) e obter um ID de aplicativo exclusivo. Em seguida, o desenvolvedor precisa criar uma função do IAM. (Por exemplo, essa função chama GameRole.) A função deve ter um documento de política do IAM anexado a ela especificando as condições sob as quais o aplicativo pode acessar GameScoresTabela do.

Quando um usuário deseja reproduzir um jogo, ele faz login em sua conta do Login with Amazon de dentro do aplicativo do jogo. O aplicativo chama o AWS Security Token Service (AWS STS) fornecendo o ID do aplicativo do Login with Amazon e solicitando a associação ao GameRole. O AWS STS retorna as credenciais temporárias da AWS para o aplicativo e permite que ele acesse a tabela GameScores, de acordo com o documento de política de GameRole.

O diagrama a seguir mostra como essas partes se encaixam.



Visão geral da federação de identidades da web

- O aplicativo chama um provedor de identidade de terceiros para autenticar o usuário e o aplicativo. O provedor de identidade retorna um token de identidade da web para o aplicativo.
- O aplicativo chama AWS STS E passa o token de identidade da web como entrada. AWS STS O autoriza o aplicativo e oferece temporáriosAWScredenciais de acesso do. O aplicativo tem permissão para assumir uma função do IAM (GameRole) e acessoAWSde acordo com a política de segurança da função.
- O aplicativo chama o DynamoDB para acessar oGameScoresTabela do. Como ele assumiu a GameRole, o aplicativo está sujeito à política de segurança associada a essa função. O documento de política impede o aplicativo de acessar dados que não pertencem ao usuário.

Mais uma vez, essa é a política de segurança de GameRole que foi mostrada em [Uso de condições de política do IAM para controle de acesso refinado \(p. 899\)](#):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToOnlyItemsMatchingUserID",
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:Query"
      ],
      "Resource": [
        "arn:aws:dynamodb:us-east-1:123456789012:table/GameScores/*"
      ]
    }
  ]
}
```

```
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
    ],
    "Resource": [
        "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores"
    ],
    "Condition": {
        "ForAllValues:StringEquals": {
            "dynamodb:LeadingKeys": [
                "${www.amazon.com:user_id}"
            ],
            "dynamodb:Attributes": [
                "UserId",
                "GameTitle",
                "Wins",
                "Losses",
                "TopScore",
                "TopScoreDateTime"
            ]
        },
        "StringEqualsIfExists": {
            "dynamodb:Select": "SPECIFIC_ATTRIBUTES"
        }
    }
}
]
```

A cláusula **Condition** determina quais itens em GameScores são visíveis para o aplicativo. Isso é feito ao comparar o ID do Login with Amazon com os valores de chave de partição **UserId** em GameScores. Somente os itens que pertencem ao usuário atual podem ser processados usando uma das ações do DynamoDB que estão listadas nessa política. Outros itens na tabela não podem ser acessados. Além disso, somente os atributos específicos listados na política podem ser acessados.

Preparação para usar a federação de identidades na web

Se você é um desenvolvedor de aplicativo e deseja usar a federação de identidades na web para seu aplicativo, siga estas etapas:

1. Cadastre-se como desenvolvedor em um provedor de identidades de terceiros. Os links externos a seguir fornecem informações sobre como se cadastrar em provedores de identidades compatíveis:
 - [Centro do desenvolvedor do Login with Amazon](#)
 - [Registro no site do Facebook](#)
 - [Uso do OAuth 2.0 para acessar APIs do Google no site do Google](#)
2. Registre seu aplicativo com o provedor de identidades. Ao fazer isso, o provedor fornece um ID exclusivo para o seu aplicativo. Se você deseja que o seu aplicativo funcione com vários provedores de identidade, é necessário obter um ID do aplicativo de cada fornecedor.
3. Crie uma ou mais funções do IAM. Você precisa de uma função para cada provedor de identidade de cada aplicativo. Por exemplo, você pode criar uma função que pode ser assumida por um aplicativo em que o usuário se conectou usando Login with Amazon, uma segunda função para o mesmo aplicativo em que o usuário se conectou usando o Facebook, e uma terceira função para o aplicativo em que os usuários fazem login usando o Google.

Como parte do processo de criação de função, você precisa anexar uma política do IAM à função. Seu documento de política deve definir os recursos do DynamoDB exigidos por seu aplicativo, e as permissões para acessar esses recursos.

Para obter mais informações, consulte [Sobre a federação de identidades da web](#) em [Guia do usuário do IAM](#).

Note

Como alternativa ao AWS Security Token Service, você pode usar o Amazon Cognito. O Amazon Cognito é o serviço preferencial para o gerenciamento de credenciais temporárias para aplicativos móveis. Para obter mais informações, consulte as páginas a seguir:

- [Como autenticar usuários \(AWS Mobile SDK for iOS\)](#)
- [Como autenticar usuários \(AWS Mobile SDK for Android\)](#)

[Gerando uma política do IAM usando o Console do DynamoDB](#)

O console do DynamoDB pode ajudar você a criar uma política do IAM para usar com a federação de identidades na web. Para fazer isso, você escolhe uma tabela do DynamoDB e especifica o provedor de identidade, ações e atributos a serem incluídos na política. O console do DynamoDB gera uma política que você pode anexar a uma função do IAM.

1. Faça login no AWS Management Console e abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, selecione Tables (Tabelas).
3. Na lista de tabelas, escolha a tabela para a qual você deseja criar a política do IAM.
4. Escolha a guia Access control (Controle de acesso).
5. Escolha o provedor de identidade, as ações e os atributos para a política.

Quando estiver satisfeito com as configurações, clique em Create policy (Criar política). A política gerada aparece.

6. Clique em Anexar as instruções e siga as etapas necessárias para anexar a política gerada a uma função do IAM.

[Criar seu aplicativo para usar a federação de identidades na web](#)

Para usar a federação de identidades na web, seu aplicativo precisa assumir a função do IAM que você criou. A partir daí, o aplicativo honra a política de acesso que você anexou à função.

No tempo de execução, se o seu aplicativo usa a federação de identidades na web, ele deverá seguir estas etapas:

1. Autentique-se com um provedor de identidade de terceiros. Seu aplicativo deve chamar o provedor de identidade usando uma interface fornecida por ele. A maneira exata em que você autentica o usuário depende do provedor e em que plataforma seu aplicativo está em execução. Normalmente, se o usuário ainda não estiver conectado, o provedor de identidade exibe uma página de login para esse provedor.

Depois de autenticar o usuário, o provedor de identidade retorna um token de identidade da web para seu aplicativo. O formato desse token depende do fornecedor, mas geralmente é uma string muito longa.

2. Obter temporário AWS credenciais de segurança. Para fazer isso, seu aplicativo envia uma solicitação `AssumeRoleWithWebIdentity` para o AWS Security Token Service (AWS STS). Essa solicitação contém o seguinte:

- O token de identidade da web da etapa anterior
- O ID do aplicativo do provedor de identidade
- O Nome de recurso da Amazon (ARN) da função do IAM que você criou para este provedor de identidade para este aplicativo

AWS STS retorna um conjunto de AWS Credenciais de segurança que expira depois de um determinado período de tempo (3.600 segundos, por padrão).

Veja a seguir uma solicitação e resposta de exemplo de uma ação `AssumeRoleWithWebIdentity` no AWS STS . O token de identidade da web foi obtido do provedor de identidade Login with Amazon.

```
GET / HTTP/1.1
Host: sts.amazonaws.com
Content-Type: application/json; charset=utf-8
URL: https://sts.amazonaws.com/?ProviderId=www.amazon.com
&DurationSeconds=900&Action=AssumeRoleWithWebIdentity
&Version=2011-06-15&RoleSessionName=web-identity-federation
&RoleArn=arn:aws:iam::123456789012:role/GameRole
&WebIdentityToken=Atza|IQEBLjAsAhQluyKqyBiYZ8-kclvGTYM81e...(remaining characters
omitted)
```

```
<AssumeRoleWithWebIdentityResponse
  xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
  <AssumeRoleWithWebIdentityResult>
    <SubjectFromWebIdentityToken>amzn1.account.AGJZDKHJKAUUSW6C44CHPEXAMPLE</
  SubjectFromWebIdentityToken>
    <Credentials>
      <SessionToken>AQoDYXdzEMf//////////wEa8AP6nNDwcSLnf+cHupC...(remaining characters
omitted)</SessionToken>
      <SecretAccessKey>8Jhi60+EWUubbUShTEsjTxqQtM8UKvsM6XAjdA==</SecretAccessKey>
      <Expiration>2013-10-01T22:14:35Z</Expiration>
      <AccessKeyId>06198791C436IEXAMPLE</AccessKeyId>
    </Credentials>
    <AssumedRoleUser>
      <Arn>arn:aws:sts::123456789012:assumed-role/GameRole/web-identity-federation</Arn>
      <AssumedRoleId>AROAJU4SA2VW5SZRF2YMG:web-identity-federation</AssumedRoleId>
    </AssumedRoleUser>
  </AssumeRoleWithWebIdentityResult>
  <ResponseMetadata>
    <RequestId>c265ac8e-2ae4-11e3-8775-6969323a932d</RequestId>
  </ResponseMetadata>
</AssumeRoleWithWebIdentityResponse>
```

3. Acessar o AWS recursos da AWS. A resposta do AWS STS O contém informações de que seu aplicativo precisa para acessar recursos do DynamoDB:

- Os campos `AccessKeyId`, `SecretAccessKey` e `SessionToken` contêm credenciais de segurança válidas apenas para este usuário e este aplicativo.
- O campo `Expiration` significa o limite de tempo dessas credenciais. Depois disso, elas não são mais válidas.
- O campo `AssumedRoleId` O campo contém o nome de uma função do IAM específica da sessão que foi assumida pelo aplicativo. O aplicativo honra os controles de acesso do documento de política do IAM durante esta sessão.
- O campo `SubjectFromWebIdentityToken` O campo contém o ID exclusivo que aparece em uma variável de política do IAM para esse provedor de identidade específico. Veja a seguir as variáveis de política do IAM para provedores compatíveis, e alguns valores de exemplo para elas:

Variável de política	Valor de exemplo
<code>#{www.amazon.com:user_id}</code>	<code>amzn1.account.AGJZDKHJKAUUSW6C44CHPEXAMPLE</code>
<code>#{graph.facebook.com:id}</code>	<code>123456789</code>

Variável de política	Valor de exemplo
<code>#{accounts.google.com:sub}</code>	123456789012345678901

Por obter políticas do IAM do exemplo nas quais essas variáveis de política são usadas, consulte[Exemplos de políticas: Uso de condições para controle de acesso refinado \(p. 903\)](#).

Para obter mais informações sobre como o AWS STS gera credenciais de acesso temporárias, consulte[Solicitação de credenciais de segurança temporária](#)emGuia do usuário do IAM.

Identity and Access Management no DynamoDB Accelerator

O DynamoDB Accelerator (DAX) foi projetado para funcionar em conjunto com o DynamoDB, para adicionar uma camada de cache aos seus aplicativos sem problemas. No entanto, o DAX e o DynamoDB têm mecanismos de controle de acesso separados. Ambos os serviços usamAWS Identity and Access ManagementPara implementar suas respectivas políticas de segurança, mas os modelos de segurança para o DAX e o DynamoDB são diferentes.

Para obter mais informações sobre o Identity and Access Management para o DAX, consulte[Controle de acesso DAX \(p. 805\)](#).

Validação de conformidade por setor do DynamoDB

Auditores externos avaliam a segurança e a conformidade do DynamoDB como parte de váriosAWSprogramas de conformidade, incluindo o seguinte:

- Controles do Sistema e da Organização (CSO)
- PCI (Payment Card Industry)
- Federal Risk and Authorization Management Program (FedRAMP)
- Lei de Portabilidade e Responsabilidade de Provedores de Saúde (HIPAA) dos EUA

A AWS fornece uma lista atualizada com frequência de serviços da AWS no escopo de programas de conformidade específicos em [Serviços da AWS no escopo do programa de conformidade](#).

Os relatórios de auditoria de terceiros estão disponíveis para download por meio do AWS Artifact. Para obter mais informações, consulte [Baixar relatórios no AWS Artifact](#).

Para obter mais informações sobre programas de conformidade da AWS, consulte [Programas de conformidade da AWS](#).

Sua responsabilidade de conformidade ao usar o DynamoDB é determinada pela confidencialidade de seus dados, pelas metas de conformidade da sua empresa e pelas regulamentações e leis aplicáveis. Caso o uso do DynamoDB esteja sujeito à conformidade com padrões como HIPAA, PCI ou FedRAMP,AWSfornecer recursos para ajudar:

- [Guias de Início Rápido de segurança e conformidade](#)Guias de implantação que discutem as considerações de arquitetura e fornecem etapas para implantação de ambientes de linha de base focados em conformidade e segurança noAWS.
- [Whitepaper de Architecting for HIPAA Security and Compliance](#)— Um whitepaper que descreve como as empresas podem usarAWSpara criar aplicativos compatíveis com HIPAA.
- [AWSRecursos de conformidade](#)— uma coleção de manuais e guias que pode ser aplicada ao seu setor e local.

- [AWS Config](#)— um serviço que avalia até que ponto suas configurações de recursos estão em conformidade com práticas internas, diretrizes do setor e regulamentações.
- [AWS Security Hub](#)— uma visão abrangente do seu estado de segurança na AWS. Isso ajuda você a verificar a sua conformidade com os padrões e as melhores práticas do setor de segurança.

Resiliência e recuperação de desastres no Amazon DynamoDB

A AWS infraestrutura global da é criada com base em AWS Regiões e zonas de disponibilidade. As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, que são conectadas com baixa latência, altas taxas de transferência e redes altamente redundantes. Com as zonas de disponibilidade, você pode projetar e operar aplicativos e bancos de dados que executam o failover automaticamente entre as zonas de disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Se você precisar replicar seus dados ou aplicativos para distâncias geográficas maiores, use AWS Local Region é um único data center projetado para complementar uma AWS Região : Todos os AWS Regiões da AWS Local Region são completamente isolados de outros AWS Regiões da

Para obter mais informações sobre regiões e zonas de disponibilidade da AWS, consulte [Infraestrutura global da AWS](#).

Além da AWS infraestrutura global da, o Amazon DynamoDB oferece vários recursos para ajudar a atender às necessidades de resiliência de dados e backup.

Backup e restauração sob demanda

O DynamoDB oferece o recurso de backup sob demanda. Ele permite que você crie backups completos das suas tabelas para retenção e arquivamento em longo prazo. Para obter mais informações, consulte [Backup e restauração sob demanda para o DynamoDB](#).

Recuperação point-in-time

A recuperação point-in-time ajuda a proteger as tabelas do DynamoDB de operações acidentais de gravação ou exclusão. Com a recuperação point-in-time, você não precisa se preocupar com a criação, a manutenção ou a programação de backups sob demanda. Para obter mais informações, consulte [Recuperação point-in-time do DynamoDB](#).

Segurança da infraestrutura no Amazon DynamoDB

Como um serviço gerenciado, o Amazon DynamoDB é protegido pelo AWS Procedimentos de segurança de rede globais da descritos na [Amazon Web Services: Visão geral dos processos de segurança](#) Whitepaper.

Você usa o AWS As chamadas de API publicadas pela para acessar o DynamoDB pela rede. Os clientes devem ter suporte ao TLS (Transport Layer Security) 1.0. Recomendamos TLS 1.2 ou posterior. Os clientes também devem ter suporte a pacotes de criptografia com sigilo de encaminhamento perfeito (PFS) como Ephemeral Diffie-Hellman (DHE) ou Elliptic Curve Diffie-Hellman Encaminhamento (ECDHE). A maioria dos sistemas modernos como Java 7 e versões posteriores oferece suporte a esses modos. Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o [AWS Security Token Service \(AWS STS\)](#) para gerar credenciais de segurança temporárias para assinar solicitações.

Você também pode usar uma VPC endpoint para o DynamoDB para permitir que instâncias do Amazon EC2 em sua VPC usem seus endereços IP privados para acessar o DynamoDB sem se expor à Internet pública. Para mais informações, consulte [Uso de endpoints da Amazon VPC para acessar o DynamoDB \(p. 917\)](#).

Uso de endpoints da Amazon VPC para acessar o DynamoDB

Por razões de segurança, muitos AWS clientes executam seus aplicativos em um ambiente do Amazon Virtual Private Cloud (Amazon VPC). Com a Amazon VPC, você pode executar instâncias do Amazon EC2 em uma nuvem privada virtual, que é isolada logicamente de outras redes, incluindo a Internet pública. Com uma Amazon VPC, você tem controle sobre o intervalo de endereços IP, as sub-redes, as tabelas de roteamento, os gateways de rede e as configurações de segurança.

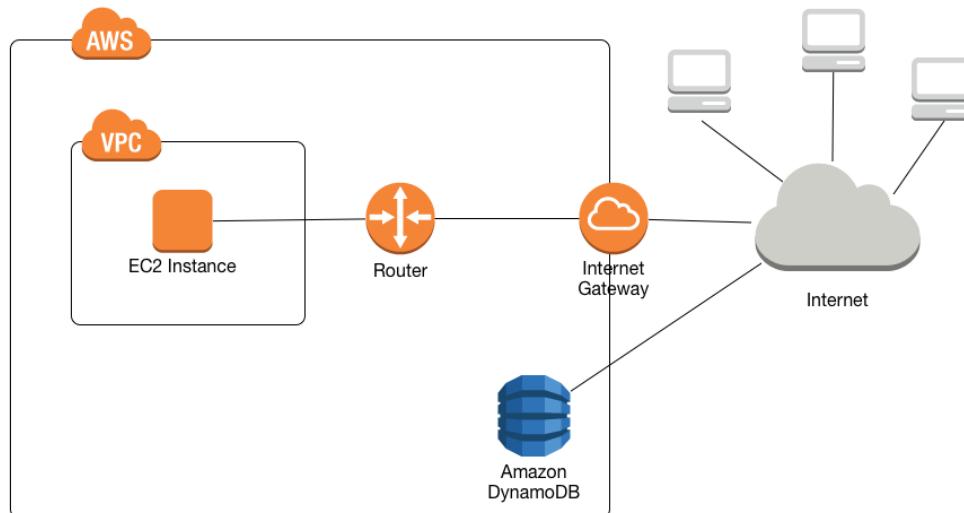
Note

Se você criou seu AWS depois de 4 de dezembro de 2013, você já tem uma VPC padrão em cada AWS Região : Uma VPC padrão está pronta para ser usada. Você pode começar a usá-la imediatamente, sem precisar executar quaisquer etapas de configuração adicionais.

Para obter mais informações, consulte [Padrão VPC e sub-redes padrão](#) no Guia do usuário da Amazon VPC.

Para acessar a Internet pública, a VPC deve ter um gateway de Internet, um roteador virtual que conecta a VPC à Internet. Isso permite que os aplicativos em execução no Amazon EC2 em sua VPC acessem os recursos da Internet, como o Amazon DynamoDB.

Por padrão, as comunicações de e para o DynamoDB usam o protocolo HTTPS, que protege o tráfego de rede usando a criptografia SSL/TLS. O diagrama a seguir mostra como uma instância do EC2 em uma VPC acessa o DynamoDB:

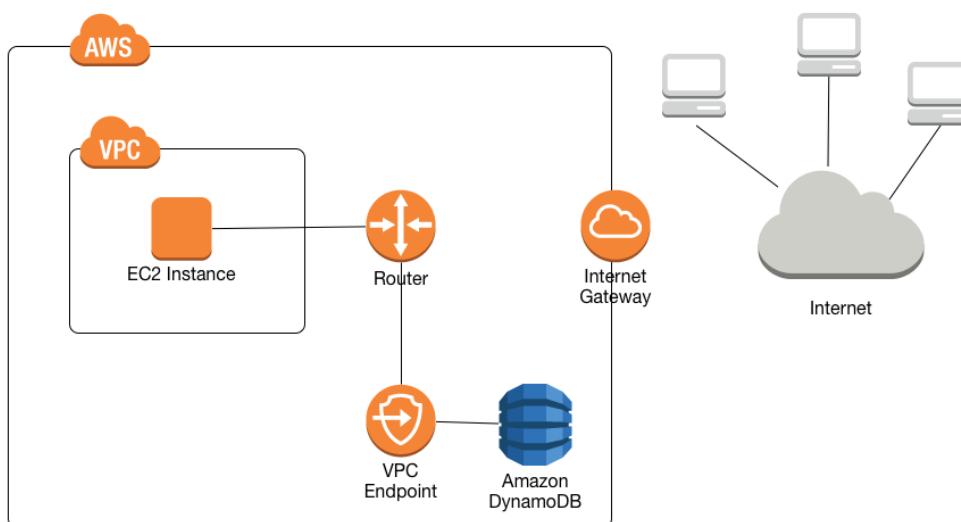


Muitos clientes têm preocupações legítimas com a segurança e a privacidade sobre o envio e o recebimento de dados pela Internet pública. Você pode abordar essas preocupações usando uma rede privada virtual (VPN) para rotear todo o tráfego de rede do DynamoDB por meio da infraestrutura de rede corporativa deles. No entanto, essa abordagem pode introduzir desafios de largura de banda e disponibilidade.

Os VPC endpoints para DynamoDB podem reduzir esses desafios. AVPC endpointPara o DynamoDB permite que instâncias do Amazon EC2 em sua VPC usem seus endereços IP privados para acessar o DynamoDB sem se expor à Internet pública. Suas instâncias do EC2 não precisam de endereços IP públicos, e você não precisa de um gateway da Internet, um dispositivo NAT ou um gateway privado virtual em sua VPC. Use as políticas de endpoint para controlar o acesso ao DynamoDB. Tráfego entre sua VPC e oAWSO serviço não deixa a rede da Amazon.

Ao criar um VPC endpoint para o DynamoDB, quaisquer solicitações para um DynamoDB endpoint dentro da região (por exemplo:dynamodb.us-west-2.amazonaws.com) são roteados para um endpoint privado do DynamoDB na rede da Amazon. Você não precisa modificar seus aplicativos em execução em instâncias do EC2 na VPC. O nome do endpoint permanece o mesmo, mas a rota para o DynamoDB permanece inteiramente dentro da rede da Amazon, e não acessa a Internet pública.

O diagrama a seguir mostra como uma instância do EC2 em uma VPC pode usar uma VPC endpoint para acessar o DynamoDB.



Para mais informações, consulte the section called “Tutorial: Uso de um VPC endpoint para DynamoDB” (p. 918).

Tutorial: Uso de um VPC endpoint para DynamoDB

Esta seção mostra a configuração e o uso de uma VPC endpoint para o DynamoDB.

Tópicos

- [Etapa 1: Executar uma instância do Amazon EC2 \(p. 918\)](#)
- [Etapa 2: Configurar sua instância do Amazon EC2 \(p. 920\)](#)
- [Etapa 3: Criar um VPC endpoint para DynamoDB \(p. 921\)](#)
- [Etapa 4: Limpar \(opcional\) \(p. 922\)](#)

Etapa 1: Executar uma instância do Amazon EC2

Nesta etapa, você executa uma instância do Amazon EC2 na Amazon VPC padrão. Em seguida, você pode criar e usar um VPC endpoint para DynamoDB.

1. Abra o console do Amazon EC2 em <https://console.aws.amazon.com/ec2/>.
2. Escolha Iniciar instância e faça o seguinte:

Etapa 1: Escolha uma Imagem de máquina da Amazon (AMI)

- Na parte superior da lista de AMIs, vá para Amazon Linux AMI (AMI do Amazon Linux) e escolha Select (Selecionar).

Etapa 2: Escolha um tipo de instância

- Na parte superior da lista de tipos de instância, escolha t2.micro.
- Selecione Next (Próximo): Configurar os detalhes da instância.

Etapa 3: Configurar os detalhes da instância

- Vá para Network (Rede) e escolha sua VPC padrão.

Selecione Next (Próximo): Add Storage.

Etapa 4: Add Storage

- Pule essa etapa escolhendo Próximo: Marcar a instância.

Etapa 5: Marcar a instância

- Pule essa etapa escolhendo Próximo: Configurar o grupo de segurança.

Etapa 6: Configurar o grupo de segurança

- Escolha Selecionar um security group existente.
- Na lista de security groups, escolha padrão. Este é o security group padrão para sua VPC.
- Selecione Next (Próximo): Revisar e lançar.

Etapa 7: Revisar o lançamento da instância

- Escolha Executar.

3. Na janela Selecionar um par de chaves existente ou criar um novo par de chaves, faça um dos seguintes:

- Se não tiver um key pair do Amazon EC2, escolha Criar um novo key pair e siga as instruções. Será solicitado que você faça download de um arquivo de chave privada (.pemArquivo), que será necessário no momento de fazer login na sua instância do Amazon EC2.
 - Se você já tiver um key pair existente do Amazon EC2, acesse o Selecionar um par de chaves e escolha o seu key pair na lista. Você já deve ter o arquivo de chave privada (.pem) disponível para fazer login na instância do Amazon EC2.
4. Após configurar o par de chaves, escolha Iniciar instâncias.
 5. Retorne à página inicial do console do Amazon EC2 e escolha a instância que você executou. No painel inferior, na guia Descrição, localize o DNS público para sua instância. Por exemplo: ec2-00-00-00-00.us-east-1.compute.amazonaws.com.

Anote esse nome DNS público, pois você precisará dele na próxima etapa deste tutorial ([Etapa 2: Configurar sua instância do Amazon EC2 \(p. 920\)](#)).

Note

Serão necessários alguns minutos para que a sua instância do Amazon EC2 se torne disponível. Antes de ir para a próxima etapa, verifique se o Estado da instância é `running` e se todas as suas Verificações de status foram aprovadas.

Etapa 2: Configurar sua instância do Amazon EC2

Quando sua instância do Amazon EC2 estiver disponível, você poderá fazer login e prepará-la para ser usada pela primeira vez.

Note

As etapas a seguir assumem que você está se conectando à sua instância do Amazon EC2 de um computador que executa o Linux. Para conhecer outras formas de se conectar, consulte [oConecte-se à sua instância do Linux](#)No Guia do usuário do Amazon EC2 para instâncias do Linux.

1. É necessário autorizar o tráfego SSH de entrada para a sua instância do Amazon EC2. Para fazer isso, crie um novo security group EC2 e, em seguida, atribua o security group à sua instância do EC2.
 - a. No painel de navegação, selecione Grupos de segurança.
 - b. Escolha Create Security Group. Na janela Security group, faça o seguinte:
 - Nome do security group – digite um nome para o security group. Por exemplo: `my-ssh-access`
 - Descrição – digite uma breve descrição para o security group.
 - VPC – escolha a VPC padrão.
 - Na seção Regras do security group, escolha Adicionar regra e faça o seguinte:
 - Tipo – escolha SSH.
 - Origem – escolha My IP.

Quando estiver satisfeito com as configurações, escolha Create.

- c. No painel de navegação, escolha Instances (Instâncias).
 - d. Escolha a instância do Amazon EC2 que você executou no[Etapa 1: Executar uma instância do Amazon EC2 \(p. 918\)](#).
 - e. Escolha Ações --> Rede --> Alterar security groups.
 - f. NoAlterar grupos de segurança, selecione o security group que você criou anteriormente neste procedimento (por exemplo: `my-ssh-access`). O existente default security group também deve ser selecionado. Quando estiver satisfeito com as configurações, escolha Atribuir security group.
2. Usar sshPara fazer login em sua instância do Amazon EC2, conforme o exemplo a seguir.

```
ssh -i my-keypair.pem ec2-user@public-dns-name
```

Você precisará especificar seu arquivo de chave privada (.pem) e o nome público do DNS da sua instância. (Consulte [Etapa 1: Executar uma instância do Amazon EC2 \(p. 918\)](#)).

O ID de login é `ec2-user`. Nenhuma senha é necessária.

3. Configure oAWS Credenciais, conforme mostrado a seguir. Insira oAWSID da chave de acesso, chave secreta e nome da região padrão quando solicitado.

```
aws configure
```

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
```

```
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-east-1
Default output format [None]:
```

Agora, você está pronto para criar um VPC endpoint para DynamoDB.

Etapa 3: Criar um VPC endpoint para DynamoDB

Nesta etapa, você criará um VPC endpoint para o DynamoDB e o testará para garantir que ele funciona.

1. Antes de começar, verifique se você pode se comunicar com o DynamoDB usando seu endpoint público.

```
aws dynamodb list-tables
```

A saída mostrará uma lista de tabelas do DynamoDB que você possui atualmente. (Se você não tiver tabelas, a lista estará vazia).

2. Verifique se o DynamoDB é um serviço disponível para a criação de VPC endpoints no AWSRegião : (O comando é mostrado em negrito, seguido pelo exemplo de saída).

```
aws ec2 describe-vpc-endpoint-services

{
    "ServiceNames": [
        "com.amazonaws.us-east-1.s3",
        "com.amazonaws.us-east-1.dynamodb"
    ]
}
```

No exemplo de saída, o DynamoDB é um dos serviços disponíveis, portanto, você pode prosseguir com a criação de um VPC endpoint para ele.

3. Determine o identificador da VPC.

```
aws ec2 describe-vpcs

{
    "Vpcs": [
        {
            "VpcId": "vpc-0bbc736e",
            "InstanceTenancy": "default",
            "State": "available",
            "DhcpOptionsId": "dopt-8454b7e1",
            "CidrBlock": "172.31.0.0/16",
            "IsDefault": true
        }
    ]
}
```

No exemplo de saída, o ID da VPC é vpc-0bbc736e.

4. Crie o VPC endpoint. Para o parâmetro --vpc-id, especifique o ID da VPC da etapa anterior. Use o parâmetro --route-table-ids para associar o endpoint às tabelas de rotas.

```
aws ec2 create-vpc-endpoint --vpc-id vpc-0bbc736e --service-name com.amazonaws.us-east-1.dynamodb --route-table-ids rtb-11aa22bb
```

```
{
    "VpcEndpoint": {
```

```
    "PolicyDocument": "{\"Version\":\"2008-10-17\", \"Statement\":[{\"Effect\": \"Allow\", \"Principal\":\"*\", \"Action\":\"*\", \"Resource\":\"*\"}]}",
    "VpcId": "vpc-0bbc736e",
    "State": "available",
    "ServiceName": "com.amazonaws.us-east-1.dynamodb",
    "RouteTableIds": [
        "rtb-11aa22bb"
    ],
    "VpcEndpointId": "vpce-9b15e2f2",
    "CreationTimestamp": "2017-07-26T22:00:14Z"
}
}
```

5. Verifique se você pode acessar o DynamoDB por meio do VPC endpoint.

```
aws dynamodb list-tables
```

Se quiser, experimente algum outro AWS CLI para DynamoDB. Para obter mais informações, consulte a [Referência de comandos da AWS CLI](#).

Etapa 4: Limpar (opcional)

Para excluir os recursos que você criou neste tutorial, siga estes procedimentos:

Para remover seu VPC endpoint para DynamoDB

1. Faça login na instância do Amazon EC2.
2. Determine o ID do VPC endpoint.

```
aws ec2 describe-vpc-endpoints

{
    "VpcEndpoint": {
        "PolicyDocument": "{\"Version\":\"2008-10-17\", \"Statement\":[{\"Effect\": \"Allow\", \"Principal\":\"*\", \"Action\":\"*\", \"Resource\":\"*\"}]}",
        "VpcId": "vpc-0bbc736e",
        "State": "available",
        "ServiceName": "com.amazonaws.us-east-1.dynamodb",
        "RouteTableIds": [],
        "VpcEndpointId": "vpce-9b15e2f2",
        "CreationTimestamp": "2017-07-26T22:00:14Z"
    }
}
```

No exemplo de saída, o ID do VPC endpoint é `vpce-9b15e2f2`.

3. Exclua o VPC endpoint.

```
aws ec2 delete-vpc-endpoints --vpc-endpoint-ids vpce-9b15e2f2

{
    "Unsuccessful": []
}
```

A matriz vazia [] indica o sucesso (não há solicitações malsucedidas).

Para encerrar sua instância do Amazon EC2

1. Abra o console do Amazon EC2 em <https://console.aws.amazon.com/ec2/>.

2. No painel de navegação, escolha Instances (Instâncias).
3. Escolha a instância do Amazon EC2.
4. Escolha Actions (Ações), Instance State (Estado da instância), Terminate (Encerrar).
5. Na janela de confirmação, clique em Sim, encerrar.

Análise de configuração e vulnerabilidade no Amazon DynamoDB

A AWS se encarrega das tarefas básicas de segurança, como aplicação de patches a bancos de dados e sistemas operacionais (SOs) convidados, configuração de firewalls e recuperação de desastres. Esses procedimentos foram revisados e certificados por terceiros certificados. Para obter mais detalhes, consulte os recursos a seguir:

- [Validação de conformidade para o Amazon DynamoDB](#)
- [Modelo de responsabilidade compartilhada](#)
- [Amazon Web Services: Visão geral dos processos de segurança do\(whitepaper\)](#)

As seguintes melhores práticas de segurança também abordam a análise de configuração e vulnerabilidade no Amazon DynamoDB:

- [Monitore a conformidade do DynamoDB com AWS Config Rules](#)
- [Monitorar a configuração do DynamoDB com o AWS Config](#)

Melhores práticas de segurança para o Amazon DynamoDB

O Amazon DynamoDB fornece uma série de recursos de segurança a serem considerados no desenvolvimento e na implementação das suas próprias políticas de segurança. As melhores práticas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas práticas recomendadas podem não ser adequadas ou suficientes no seu ambiente, trate-as como considerações úteis em vez de requisitos.

Tópicos

- [Melhores práticas de segurança preventiva do DynamoDB \(p. 923\)](#)
- [Melhores práticas de segurança do DynamoDB Detective \(p. 925\)](#)

Melhores práticas de segurança preventiva do DynamoDB

As práticas recomendadas a seguir podem ajudar a antecipar e evitar incidentes de segurança no Amazon DynamoDB.

Criptografia em repouso

O DynamoDB criptografa em repouso todos os dados de usuário gravados em tabelas, índices, streams e backups, usando chaves de criptografia armazenadas no[AWS Key Management](#)

[Service\(AWS KMS\)](#). Isso oferece uma camada de proteção de dados adicional ao proteger seus dados contra acesso não autorizado ao armazenamento subjacente.

Você pode especificar se o DynamoDB deve usar umAWS CMK de propriedade (tipo de criptografia padrão) ou umAWS gerenciado CMK para criptografar os dados do usuário. Para obter mais informações, consulte [Criptografia do Amazon DynamoDB em repouso](#).

Usar funções do IAM para autenticar o acesso ao DynamoDB

Para usuários, aplicativos e outros AWS para acessar o DynamoDB, eles devem incluir AWS sem suas credenciais do AWS. As solicitações de API. Você não deve armazenar AWS credenciais diretamente no aplicativo ou em instância do EC2. Essas são credenciais de longo prazo que não são automaticamente alternadas e, portanto, podem ter impacto comercial significativo se forem comprometidas. Uma função do IAM permite obter chaves de acesso temporárias que podem ser usadas para acessar o AWS serviços e recursos da.

Para obter mais informações, consulte [Autenticação](#).

Usar políticas do IAM para autorizações de base do DynamoDB

Ao conceder permissões, você decide quem as recebe, a quais APIs do DynamoDB as permissões se referem e as ações específicas que deseja permitir nesses recursos. A implementação do privilégio mínimo é fundamental para reduzir o risco de segurança e o impacto que pode resultar de erros ou usuários mal-intencionados.

Anexe políticas de permissões para identidades do IAM (ou seja, usuários, grupos e funções) e, assim, dê permissões para executarem operações nos recursos do DynamoDB.

Para isso, você pode usar o seguinte:

- [AWS Políticas gerenciadas \(predefinidas\)](#)
- [Políticas gerenciadas pelo cliente](#)

Uso de condições de política do IAM para controle de acesso refinado

Ao conceder permissões no DynamoDB, você pode especificar as condições que determinam como uma política de permissões entra em vigor. A implementação do privilégio mínimo é fundamental para reduzir o risco de segurança e o impacto que pode resultar de erros ou usuários mal-intencionados.

É possível especificar as condições ao conceder permissões usando uma política do IAM. Por exemplo, você pode fazer o seguinte:

- Conceder permissões a fim de permitir acesso somente leitura aos usuários para determinados itens e atributos em uma tabela ou um índice secundário.
- Conceder permissões para permitir somente acesso de gravação aos usuários para determinados atributos em uma tabela com base na identidade desse usuário.

Para obter mais informações, consulte [Como usar condições de política do IAM para um controle de acesso refinado](#).

Usar um VPC endpoint e políticas para acessar o DynamoDB

Se você só precisa de acesso ao DynamoDB a partir de uma nuvem privada virtual (VPC), use um VPC endpoint para limitar o acesso somente a partir da VPC requerida. Fazendo isso, você evita que o tráfego atravesse para a Internet pública e esteja sujeito a esse ambiente.

O uso de um VPC endpoint para o DynamoDB permite controle e limite de acesso do usando o seguinte:

- Políticas de VPC endpoint — Essas políticas são aplicadas ao DynamoDB VPC endpoint do. Elas permitem controle e limite de acesso da API à tabela do DynamoDB.

- Políticas do IAM — Usando o AWS SDK for Java, você pode reforçar que todo o acesso à tabela do DynamoDB é feito por meio do VPC endpoint especificado.

Para obter mais informações, consulte [Endpoints para o Amazon DynamoDB](#).

Considere utilizar a criptografia do lado do cliente

Se você armazena dados confidenciais e sensíveis no DynamoDB, talvez seja melhor criptografar os dados o mais próximo possível da origem, para que eles fiquem protegidos durante todo o ciclo de vida. A criptografia dos seus dados confidenciais em trânsito e em repouso ajuda você a garantir que os dados em texto simples não estejam disponíveis a terceiros.

O [Cliente de criptografia do Amazon DynamoDB](#) é uma biblioteca de software que ajuda você proteger seus dados de tabela antes de enviá-los ao DynamoDB.

No núcleo do DynamoDB Encryption Client está um criptografador de itens que criptografa, assina, verifica e descriptografa itens de tabela. Ele obtém informações sobre os itens da tabela e instruções sobre quais itens devem ser criptografados e assinados. Ele obtém os materiais de criptografia e as instruções sobre como usá-los de um provedor de materiais de criptografia que você seleciona e configura.

Melhores práticas de segurança do DynamoDB Detective

As práticas recomendadas a seguir para o Amazon DynamoDB podem ajudar a detectar pontos fracos e incidentes potenciais de segurança.

Usar o AWS CloudTrail Para monitorar o uso de chave do KMS gerenciada

Se você estiver usando um [AWS KMS](#) para gerenciar sua chave, o CloudTrail fornece visibilidade da atividade do usuário ao registrar ações realizadas em sua conta. O CloudTrail registra informações importantes sobre cada ação, incluindo quem fez a solicitação, os serviços usados, as ações realizadas, os parâmetros das ações e os elementos de resposta retornados pelo AWS KMS. Essas informações ajudam você a rastrear as alterações feitas em seu AWS KMS e solucionar problemas operacionais. O CloudTrail facilita garantir a conformidade com as políticas internas e os padrões regulatórios.

Use o CloudTrail para auditar o uso de chave. O CloudTrail cria arquivos de log que contêm um histórico de chamadas de API e eventos relacionados da sua conta. Esses arquivos de log incluem todos os solicitados de API feitos com o AWS Management Console, AWS SDKs e ferramentas de linha de comando, para além das feitas por meio do AWS CloudTrail. Use esses arquivos de log para obter informações sobre quando a CMK foi usada, a operação solicitada, a identidade do solicitante, o endereço IP de origem da solicitação e assim por diante. Para obter mais informações, consulte [Registro em log AWS KMS Chamadas de API do AWS CloudTrail](#) e o [AWS CloudTrail Guia do usuário](#).

Monitorar operações do DynamoDB usando o CloudTrail

O CloudTrail pode monitorar eventos de plano de controle e eventos de plano de dados. As operações de plano de controle permitem que você crie e gerencie as tabelas do DynamoDB. Elas também permitem que você trabalhe com índices, streams e outros objetos que são dependentes de tabelas. As operações de plano de dados permitem criar, ler, atualizar e excluir (também chamadas de CRUD) em dados de uma tabela do. Algumas operações de plano de dados também permitem que você leia dados de um índice secundário. Para ativar o registro de eventos de plano de dados no CloudTrail, você precisará ativar o registro da atividade da API do plano de dados no CloudTrail. Consulte [Registro de eventos de dados para trilhas](#) para obter mais informações.

Quando ocorre atividade no DynamoDB, essa atividade é registrada em um evento do CloudTrail junto com outros AWS eventos de serviço no histórico de eventos. Para obter mais informações, consulte [Registrar operações do DynamoDB ao usar o AWS CloudTrail](#). Você pode visualizar, pesquisar e fazer download de eventos recentes em sua conta da AWS. Para obter mais informações, consulte [Visualizar eventos com o histórico de eventos CloudTrail](#) no Guia do Usuário do AWS CloudTrail.

Para obter um registro contínuo de eventos em seu AWS, incluindo eventos para o DynamoDB, crie uma conta [Trilha do](#). Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon Simple Storage Service (Amazon S3). Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as regiões da AWS. A trilha registra eventos de todas as regiões na partição da AWS e fornece os arquivos de log ao bucket do S3 que você especificar. Além disso, você pode configurar outras AWS para analisar mais profundamente e agir sobre os dados de eventos coletados nos logs do CloudTrail.

Use o DynamoDB Streams para monitorar operações de plano de dados

O DynamoDB é integrado ao AWS Lambda para que você possa criar triggers (pedaços de código que respondem automaticamente a eventos no DynamoDB Streams). Com triggers, você pode criar aplicativos que reagem às modificações de dados em tabelas do DynamoDB.

Se habilitar o DynamoDB Streams em uma tabela, você poderá associar o Nome de recurso da Amazon (ARN) do stream a uma função do Lambda que você escrever. Imediatamente após um item da tabela ser modificado, um novo registro aparece no stream da tabela. O AWS Lambda consulta o stream e invoca sua função do Lambda de forma síncrona ao detectar novos registros de stream. A função do Lambda pode realizar qualquer ação que você especificar, como enviar uma notificação ou iniciar um fluxo de trabalho.

Para ver um exemplo, consulte [Tutorial: O uso do AWS Lambda Com o Amazon DynamoDB Streams](#). Este exemplo recebe uma entrada de evento do DynamoDB, processa as mensagens que ele contém e grava alguns dados de evento de entrada nos Amazon CloudWatch Logs.

Monitorar a configuração do DynamoDB com o AWS Config

O uso do [AWS Config](#), você pode monitorar e gravar alterações de configuração do AWS recursos da AWS. Você também pode usar o AWS Config para fazer o inventário do AWS recursos da AWS. Quando uma alteração em um estado anterior é detectada, uma notificação do Amazon Simple Notification Service (Amazon SNS) pode ser entregue para que você revise e tome uma ação. Siga as orientações em [Configuração AWS Config com o console do](#), garantindo que os tipos de recursos do DynamoDB sejam incluídos.

Você pode configurar o AWS Config para transmitir as alterações de configuração e notificações para um tópico do Amazon SNS. Por exemplo, quando um recurso é atualizado, você pode receber uma notificação em seu e-mail, para que possa visualizar as alterações. Você também pode ser notificado quando o AWS Config avaliar suas regras gerenciadas ou personalizadas em relação aos seus recursos.

Para ver um exemplo, consulte [Notificações que o AWS Config Envia para um tópico do Amazon SNS](#) no AWS Config Guia do desenvolvedor.

Monitore a conformidade do DynamoDB com o AWS Config Regras do

O AWS Config rastreia continuamente as alterações de configuração que ocorrem entre seus recursos. Ele verifica se as mudanças violam qualquer condição nas regras. Se um recurso viola uma regra, o AWS Config sinaliza o recurso e a regra como não compatíveis.

Usando o AWS Config para avaliar suas configurações de recursos, você pode avaliar se suas configurações de recursos atendem adequadamente a práticas internas e a diretrizes e regulamentações da indústria. O AWS Config fornece [AWS regras gerenciadas](#), que são regras predefinidas e personalizáveis que o AWS Config usa para avaliar se seu AWS está em conformidade com as práticas recomendadas comuns.

Marcar seus recursos do DynamoDB para identificação e automação

Você pode atribuir metadados ao seu AWS Recursos da forma de tags. Cada tag é um rótulo simples que consiste em uma chave definida pelo cliente e um valor opcional que pode facilitar o gerenciamento, a pesquisa e a filtragem de recursos.

A atribuição de tags (tagging) permite a implementação de controles agrupados. Embora não haja tipos de tags inerentes, elas permitem categorizar recursos por finalidade, proprietário, ambiente ou outros critérios. Veja os seguintes exemplos:

- Segurança — usada para determinar requisito como criptografia.
- Confidentiality (Confidencialidade) — um identificador para o nível de confidencialidade de dados específico compatível com um recurso.
- Ambiente — usado para distinguir entre as infraestruturas de desenvolvimento, teste e produção.

Para obter mais informações, consulte [AWS Estratégias de marcação e Marcação para o DynamoDB](#).

Monitoramento e registro em log

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e o desempenho do DynamoDB e do AWS Soluções. Você deve coletar dados de monitoramento de todas as partes do seu AWS Soluções para que você possa depurar uma falha de vários pontos com facilidade, caso ocorra. AWS fornece várias ferramentas para monitorar os recursos do DynamoDB e responder a incidentes potenciais:

Tópicos

- [Registro em log e monitoramento no DynamoDB \(p. 928\)](#)
- [Registro em log e monitoramento no DynamoDB Accelerator \(p. 970\)](#)
- [Analizando o acesso a dados usando o CloudWatch Contributor Insights para DynamoDB \(p. 970\)](#)

Registro em log e monitoramento no DynamoDB

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e o desempenho do DynamoDB e do AWS Soluções. Você deve coletar dados de monitoramento de todas as partes do AWS Para que você possa depurar mais facilmente uma falha de vários pontos, caso ocorra uma falha. Antes de começar a monitorar o DynamoDB, crie um plano de monitoramento que inclua respostas às seguintes perguntas:

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?
- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem realizará o monitoramento das tarefas?
- Quem deve ser notificado quando algo der errado?

A próxima etapa é estabelecer um parâmetro de desempenho normal do DynamoDB no ambiente medindo o desempenho em vários momentos e em diferentes condições de carga. Ao monitorar o DynamoDB, você deve pensar na possibilidade de armazenar os dados históricos de monitoramento. Esses dados armazenados proporcionam uma linha de base com a qual comparar os dados de desempenho atuais, identificar padrões normais de desempenho e anomalias de desempenho e criar métodos para a solução de problemas.

Para estabelecer uma linha de base, é preciso, no mínimo, monitorar os seguintes itens:

- O número de unidades de capacidade de leitura ou gravação consumidas ao longo do período de tempo especificado, para que você possa acompanhar quanto do throughput provisionado foi usado.
- As solicitações que excederam a capacidade de gravação ou de leitura provisionada de uma tabela durante o período especificado, para que você possa determinar as solicitações que excedem as cotas de taxa de transferência provisionada de uma tabela.
- Erros de sistema, para que você possa determinar se todas as solicitações resultaram em um erro.

Tópicos

- [Ferramentas de monitoramento \(p. 929\)](#)
- [Monitorar com o Amazon CloudWatch \(p. 930\)](#)
- [Registro de operações do DynamoDB usando o AWS CloudTrail \(p. 954\)](#)

Ferramentas de monitoramento

AWSA fornece ferramentas que você pode usar para monitorar o DynamoDB. Você pode configurar algumas dessas ferramentas para fazer o monitoramento em seu lugar; algumas delas exigem intervenção manual. Recomendamos que as tarefas de monitoramento sejam automatizadas ao máximo possível.

Ferramentas de monitoramento automatizadas

Você pode usar as seguintes ferramentas de monitoramento automatizadas para observar o DynamoDB e gerar relatórios quando algo estiver errado:

- Amazon CloudWatch Alarms (Alarmes do Amazon CloudWatch): observe uma única métrica ao longo de um período que você especificar e realize uma ou mais ações com base no valor da métrica em relação a um determinado limite ao longo de vários períodos. A ação é uma notificação enviada para um tópico do Amazon Simple Notification Service (Amazon SNS) ou por uma política do Amazon EC2 Auto Scaling. Os alarmes do CloudWatch não invocam ações simplesmente porque estão em um estado específico. O estado deve ter sido alterado e mantido por um número específico de períodos. Para obter mais informações, consulte [Monitorar com o Amazon CloudWatch \(p. 930\)](#).
- Amazon CloudWatch Logs: monitore, armazene e acesse seus arquivos de log do AWS CloudTrail ou de outras origens. Para obter mais informações, consulte [Monitorar arquivos de log](#) no Guia do usuário do Amazon CloudWatch.
- Amazon CloudWatch Events: faça correspondência de eventos e direcione-os a uma ou mais funções ou streams de destino para fazer alterações, capturar informações de estado e realizar ações corretivas. Para obter mais informações, consulte [O que é o Amazon CloudWatch Events?](#) no Guia do usuário do Amazon CloudWatch.
- AWS CloudTrail Log Monitoring: compartilhe arquivos de log entre contas, monitore os arquivos de log do CloudTrail em tempo real enviando-os para o CloudWatch Logs, escreva aplicações de processamento de logs em Java e confirme se os arquivos de log não foram alterados após a entrega pelo CloudTrail. Para obter mais informações, consulte [Trabalhando com arquivos de log do CloudTrail](#) no AWS CloudTrailGuia do usuário.

Ferramentas de monitoramento manual

Outra parte importante do monitoramento do DynamoDB é o monitoramento manual dos itens que os alarmes do CloudWatch não abrangem. O DynamoDB, CloudWatchTrusted Advisor, e outrosAWSOs painéis do console do fornecem uma visão rápida do estado do seuAWSMeio ambiente do. Recomendamos que você também verifique os arquivos de registro do Amazon DynamoDB.

- O painel do DynamoDB mostra:
 - Alertas recentes
 - Capacidade total
 - Integridade de serviço
- A página inicial do CloudWatch mostra:
 - Alertas e status atual
 - Gráficos de alertas e recursos
 - Estado de integridade do serviço

Além disso, você pode usar o CloudWatch para fazer o seguinte:

- Crie [painéis personalizados](#) para monitorar os serviços com os quais você se preocupa.
- Colocar em gráfico dados de métrica para solucionar problemas e descobrir tendências
- Pesquise e navegue por todas asAWSMétricas de recursos
- Criar e editar alertas para ser notificado sobre problemas

Monitorar com o Amazon CloudWatch

É possível monitorar o Amazon DynamoDB usando o CloudWatch, que coleta e processa dados brutos do DynamoDB em métricas legíveis e quase em tempo real. Essas estatísticas são mantidas por um período, para que você possa acessar informações históricas para obter uma perspectiva melhor sobre o desempenho de seu aplicativo web ou serviço. Por padrão, os dados de métrica do DynamoDB são enviados para o CloudWatch automaticamente. Para obter mais informações, consulte [O que é o Amazon CloudWatch?](#) e [Retenção de métricas](#) no Guia do usuário do Amazon CloudWatch.

Tópicos

- [Métricas e dimensões do DynamoDB \(p. 930\)](#)
- [Como faço para usar métricas do DynamoDB? \(p. 952\)](#)
- [Criação de alarmes do CloudWatch para monitorar o DynamoDB \(p. 952\)](#)

Métricas e dimensões do DynamoDB

Quando você interage com o DynamoDB, ele envia métricas e dimensões para o CloudWatch. Use os procedimentos a seguir para visualizar as métricas do DynamoDB.

Para exibir métricas (console)

As métricas são agrupadas primeiro pelo namespace do serviço e, em seguida, por várias combinações de dimensão dentro de cada namespace.

1. Abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. No painel de navegação, selecione Metrics (Métricas).
3. Selecione o namespace DynamoDB.

Para visualizar métricas (CLI)

- Em um prompt de comando, use o seguinte comando:

```
aws cloudwatch list-metrics --namespace "AWS/DynamoDB"
```

O CloudWatch exibe as seguintes métricas para DynamoDB:

Dimensões e métricas do DynamoDB

As métricas e as dimensões que o DynamoDB envia ao Amazon CloudWatch estão listadas aqui.

Métricas do DynamoDB

Note

O Amazon CloudWatch agrupa as seguintes métricas do DynamoDB em intervalos de um minuto:

- ConditionalCheckFailedRequests
- ConsumedReadCapacityUnits
- ConsumedWriteCapacityUnits
- ReadThrottleEvents
- ReturnedBytes
- ReturnedItemCount

- `ReturnedRecordsCount`
- `SuccessfulRequestLatency`
- `SystemErrors`
- `TimeToLiveDeletedItemCount`
- `ThrottledRequests`
- `TransactionConflict`
- `UserErrors`
- `WriteThrottleEvents`

Para todas as outras métricas do DynamoDB, a granularidade da agregação é de cinco minutos.

Nem todas as estatísticas, como Média ou Soma, são aplicáveis a todas as métricas do. No entanto, todos esses valores estão disponíveis por meio do console do Amazon DynamoDB ou usando o console do CloudWatch, AWS CLI, ou AWSSDKs para todas as métricas. Na tabela a seguir, cada métrica tem uma lista de estatísticas válidas que são aplicáveis a essa métrica.

Métrica	Descrição
<code>AccountMaxReads</code>	<p>O número máximo de unidades de capacidade de leitura que podem ser usadas por uma conta. Esse limite não se aplica a tabelas sob demanda ou índices secundários globais.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Maximum</code>— O número máximo de unidades de capacidade de leitura que podem ser usadas por uma conta.
<code>AccountMaxTableLevelReads</code>	<p>O número máximo de unidades de capacidade de leitura que podem ser usadas por uma tabela ou índice secundário global de uma conta. Para tabelas sob demanda, esse limite limita o máximo de unidades de solicitação de leitura que uma tabela ou um índice secundário global pode usar.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Maximum</code>— O número máximo de unidades de capacidade de leitura que podem ser usadas por uma tabela ou índice secundário global da conta.
<code>AccountMaxTableLevelWrites</code>	<p>O número máximo de unidades de capacidade de gravação que podem ser usadas por uma tabela ou índice secundário global de uma conta. Para tabelas sob demanda, esse limite limita as unidades máximas de solicitação de gravação que uma tabela ou um índice secundário global pode usar.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Maximum</code>— O número máximo de unidades de capacidade de gravação que podem ser usadas por uma tabela ou índice secundário global da conta.

Métrica	Descrição
<code>AccountMaxWrites</code>	<p>O número máximo de unidades de capacidade de gravação que podem ser usadas por uma conta. Esse limite não se aplica a tabelas sob demanda ou índices secundários globais.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Maximum</code>— O número máximo de unidades de capacidade de gravação que podem ser usadas por uma conta.
<code>AccountProvisionedReadCapacityUtilization</code>	<p>A porcentagem de unidades de capacidade de leitura provisionada utilizada por uma conta.</p> <p>Unidades de :Percent</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Maximum</code>— Porcentagem máxima de unidades de capacidade de leitura provisionada utilizadas pela conta da. • <code>Minimum</code>— Porcentagem mínima de unidades de capacidade de leitura provisionada utilizadas pela conta da. • <code>Average</code>— Porcentagem média de unidades de capacidade de leitura provisionada utilizadas pela conta da. A métrica é publicada para intervalos de cinco minutos. Portanto, se você ajustar rapidamente as unidades de capacidade de leitura provisionadas, essa estatística pode não refletir a média real.
<code>AccountProvisionedWriteCapacityUtilization</code>	<p>A porcentagem de unidades de capacidade de gravação provisionada utilizadas por uma conta.</p> <p>Unidades de :Percent</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Maximum</code>— Porcentagem máxima de unidades de capacidade de gravação provisionada utilizadas pela conta da. • <code>Minimum</code>— Porcentagem mínima de unidades de capacidade de gravação provisionada utilizadas pela conta da. • <code>Average</code>— Porcentagem média de unidades de capacidade de gravação provisionada utilizadas pela conta da. A métrica é publicada para intervalos de cinco minutos. Portanto, se você ajustar rapidamente as unidades de capacidade de gravação provisionadas, essa estatística pode não refletir a média real.

Métrica	Descrição
AgeOfOldestUnreplicatedRecord	<p>O tempo decorrido desde que um registro ainda não foi replicado para o fluxo de dados do Kinesis apareceu pela primeira vez na tabela do DynamoDB.</p> <p>Unidades de :Milliseconds</p> <p>Dimensões: TableName, DelegatedOperation</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Maximum. • Minimum. • Average.
ConditionalCheckFailedRequests	<p>O número de tentativas falhadas para executar gravações condicionais. O PutItem, UpdateItem, e DeleteItem permitem que você forneça uma condição lógica que deve ser avaliada como true antes que a operação possa prosseguir. Se essa condição avaliar como falsa, o ConditionalCheckFailedRequests é incrementado por um. ConditionalCheckFailedRequeststambém é incrementado por um para partiQL Update and Delete instruções onde uma condição lógica é fornecida e essa condição é avaliada como false.</p> <p>Note</p> <p>Uma gravação condicional com falha resultará em erro HTTP 400 (solicitação inválida). Esses eventos são refletidos no ConditionalCheckFailedRequests, mas não na UserErrors Métrica do.</p> <p>Unidades de :Count</p> <p>Dimensões: TableName</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • SampleCount • Sum

Métrica	Descrição
ConsumedChangeDataCaptureUnits	O número de unidades de captura de dados de alteração consumidas. Unidades de :Count Dimensões: TableName, DelegatedOperation Estatísticas válidas: <ul style="list-style-type: none">• Minimum• Maximum• Average

Métrica	Descrição
ConsumedReadCapacityUnits	<p>O número de unidades de capacidade de leitura consumidas ao longo do período de tempo especificado, para que você possa acompanhar quanto do throughput provisionado foi usado. Você pode recuperar a capacidade de leitura total consumida para uma tabela e todos os seus índices secundários globais ou para um índice secundário global específico. Para obter mais informações, consulte Modo de capacidade de Leitura/Gravação.</p> <p>O <code>TableName</code> retorna o <code>ConsumedReadCapacityUnits</code> para a tabela, mas não para nenhum índice secundário global. Como visualizar <code>ConsumedReadCapacityUnits</code> para um índice secundário global, você deve especificar tanto <code>TableName</code> e <code>GlobalSecondaryIndex</code>.</p> <p>Note</p> <p>Usar a <code>Sum</code> estatística para calcular a taxa de transferência consumida. Por exemplo, obtenha o <code>Sum</code> ao longo de um intervalo de um minuto, e dividi-lo pelo número de segundos em um minuto (60) para calcular a média <code>ConsumedReadCapacityUnits</code> por segundo (reconhecendo que essa média não destaca quaisquer picos grandes, mas breves na atividade de leitura que ocorreram durante esse minuto). Você pode comparar o valor calculado com o valor de throughput provisionado fornecido ao DynamoDB.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code>— O número mínimo de unidades de capacidade de leitura consumidas por qualquer solicitação individual para a tabela ou índice. • <code>Maximum</code>— O número máximo de unidades de capacidade de leitura consumidas por qualquer solicitação individual para a tabela ou índice. • <code>Average</code>— A capacidade média de leitura por solicitação consumida. <p>Note</p> <p>O <code>Average</code> é influenciado por períodos de inatividade onde o valor amostral será zero.</p> <ul style="list-style-type: none"> • <code>Sum</code>— O total de unidades de capacidade de leitura consumidas. Esta é a estatística mais útil para o <code>ConsumedReadCapacityUnits</code> Métrica do. • <code>SampleCount</code>— o número de solicitações de leitura para o DynamoDB, mesmo que nenhuma capacidade de leitura tenha sido consumida.

Métrica	Descrição
	<p>Note</p> <p>O SampleCount é influenciado por períodos de inatividade onde o valor amostral será zero.</p>

Métrica	Descrição
ConsumedWriteCapacityUnits	<p>O número de unidades de capacidade de gravação consumidas ao longo do período de tempo especificado, para que você possa acompanhar quanto do throughput provisionado foi usado. Você pode recuperar a capacidade de gravação total consumida para uma tabela e todos os seus índices secundários globais ou para um índice secundário global específico. Para obter mais informações, consulte Modo de capacidade de Leitura/Gravação.</p> <p>O <code>TableName</code> retorna o <code>ConsumedWriteCapacityUnits</code> para a tabela, mas não para nenhum índice secundário global. Como visualizar <code>ConsumedWriteCapacityUnits</code> para um índice secundário global, você deve especificar tanto <code>TableName</code> e <code>GlobalSecondaryIndex</code>.</p> <p>Note</p> <p>Usar a <code>sum</code> estatística para calcular a taxa de transferência consumida. Por exemplo, obtenha o <code>sum</code> ao longo de um intervalo de um minuto, e dividi-lo pelo número de segundos em um minuto (60) para calcular a média <code>ConsumedWriteCapacityUnits</code> por segundo (reconhecendo que essa média não destaca quaisquer picos grandes, mas breves na atividade de gravação que ocorreram durante esse minuto). Você pode comparar o valor calculado com o valor de throughput provisionado fornecido ao DynamoDB.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code>— O número mínimo de unidades de capacidade de gravação consumidas por qualquer solicitação individual para a tabela ou índice. • <code>Maximum</code>— O número máximo de unidades de capacidade de gravação consumidas por qualquer solicitação individual para a tabela ou índice. • <code>Average</code>— A capacidade média de gravação por solicitação consumida. <p>Note</p> <p>O <code>Average</code> é influenciado por períodos de inatividade onde o valor amostral será zero.</p> <ul style="list-style-type: none"> • <code>Sum</code>— O total de unidades de capacidade de gravação consumidas. Esta é a estatística mais útil para o <code>ConsumedWriteCapacityUnits</code> Métrica do. • <code>SampleCount</code>— o número de solicitações de gravação para o DynamoDB, mesmo que nenhuma capacidade de gravação tenha sido consumida.

Métrica	Descrição
	<p>Note</p> <p>O <code>SampleCount</code> é influenciado por períodos de inatividade onde o valor amostral será zero.</p>
<code>MaxProvisionedTableReadCapacity</code>	<p>A porcentagem de unidades de capacidade de leitura provisionadas utilizadas pela tabela de leitura provisionada mais alta ou pelo índice secundário global de uma conta.</p> <p>Unidades de :Percent</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Maximum— A porcentagem máxima de unidades de capacidade de leitura provisionadas utilizadas pela tabela de leitura provisionada mais alta da conta. • Minimum— a porcentagem mínima de unidades de capacidade de leitura provisionadas utilizadas pela tabela de leitura provisionada mais alta da conta. • Average— a porcentagem média de unidades de capacidade de leitura provisionadas utilizadas pela tabela de leitura provisionada mais alta da conta. A métrica é publicada para intervalos de cinco minutos. Portanto, se você ajustar rapidamente as unidades de capacidade de leitura provisionadas, essa estatística pode não refletir a média real.
<code>MaxProvisionedTableWriteCapacity</code>	<p>A porcentagem de capacidade de gravação provisionada utilizada pela tabela de gravação provisionada mais alta ou pelo índice secundário global de uma conta.</p> <p>Unidades de :Percent</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Maximum— a porcentagem máxima de unidades de capacidade de gravação provisionadas utilizadas pela tabela de gravação provisionada mais alta ou pelo índice secundário global de uma conta. • Minimum— a porcentagem mínima de unidades de capacidade de gravação provisionadas utilizadas pela tabela de gravação provisionada mais alta ou pelo índice secundário global de uma conta. • Average— a porcentagem média de unidades de capacidade de gravação provisionadas utilizadas pela tabela de gravação provisionada mais alta ou pelo índice secundário global da conta. A métrica é publicada para intervalos de cinco minutos. Portanto, se você ajustar rapidamente as unidades de capacidade de gravação provisionadas, essa estatística pode não refletir a média real.

Métrica	Descrição
<code>OnlineIndexConsumedWriteCapacity</code>	<p>O número de unidades de capacidade de gravação consumidas ao adicionar um novo índice secundário global a uma tabela. Se a capacidade de gravação do índice for muito baixa, a atividade de gravação de entrada durante a fase de preenchimento poderá ser limitada. Isso pode aumentar o tempo necessário para criar o índice. Você deve monitorar essa estatística enquanto o índice está sendo criado para determinar se a capacidade de gravação do índice está subprovisionada.</p> <p>Você pode ajustar a capacidade de gravação do índice usando a <code>UpdateTable</code> operação, mesmo enquanto o índice ainda está sendo construído.</p> <p>O <code>ConsumedWriteCapacityUnits</code> para o índice não inclui a taxa de transferência de gravação consumida durante a criação do índice.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>OnlineIndexPercentageProgress</code>	<p>A porcentagem de conclusão quando um novo índice secundário global está sendo adicionado a uma tabela. O DynamoDB deve primeiro alocar recursos para o novo índice e, em seguida, preencher atributos da tabela para o índice. Para tabelas grandes, esse processo pode levar muito tempo. Você deve monitorar essa estatística para visualizar o progresso relativo à medida que o DynamoDB constrói o índice.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>

Métrica	Descrição
<code>OnlineIndexThrottleEvents</code>	<p>O número de eventos de aceleração de gravação que ocorrem ao adicionar um novo índice secundário global a uma tabela. Esses eventos indicam que a criação do índice levará mais tempo para ser concluída, pois a atividade de gravação de entrada está excedendo a taxa de transferência de gravação provisionada do índice.</p> <p>Você pode ajustar a capacidade de gravação do índice usando a <code>UpdateTable</code> operação, mesmo enquanto o índice ainda está sendo construído.</p> <p>O <code>WriteThrottleEvents</code> para o índice não inclui quaisquer eventos de aceleração que ocorram durante a criação do índice.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>PendingReplicationCount</code>	<p>(Esta métrica é para tabelas globais do DynamoDB.) O número de atualizações de itens que foram gravadas em uma tabela de réplica, mas ainda não foram gravadas em outra réplica na tabela global.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>ReceivingRegion</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Average</code> • <code>Sample Count</code> • <code>Sum</code>

Métrica	Descrição
ProvisionedReadCapacityUnits	<p>O número de unidades de capacidade de leitura provisionada para uma tabela ou um índice secundário global.</p> <p>O <code>TableName</code> retorna o <code>ProvisionedReadCapacityUnits</code> para a tabela, mas não para nenhum índice secundário global. Como visualizar <code>ProvisionedReadCapacityUnits</code> para um índice secundário global, você deve especificar tanto <code>TableName</code> e <code>GlobalSecondaryIndexName</code>.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code>—A configuração mais baixa para capacidade de leitura provisionada. Se você usar <code>UpdateTable</code> para aumentar a capacidade de leitura, esta métrica mostra o valor mais baixo de <code>ReadCapacityUnits</code> durante este período de tempo. • <code>Maximum</code>—A configuração mais alta para capacidade de leitura provisionada. Se você usar <code>UpdateTable</code> para diminuir a capacidade de leitura, essa métrica mostra o valor mais alto de <code>ReadCapacityUnits</code> durante este período de tempo. • <code>Average</code>—A capacidade média de leitura provisionada. O <code>ProvisionedReadCapacityUnits</code> é publicada em intervalos de cinco minutos. Portanto, se você ajustar rapidamente as unidades de capacidade de leitura provisionadas, essa estatística pode não refletir a média real.

Métrica	Descrição
ProvisionedWriteCapacityUnits	<p>O número de unidades de capacidade de gravação provisionada para uma tabela ou um índice secundário global.</p> <p>O <code>TableName</code> retorna <code>ProvisionedWriteCapacityUnits</code> para a tabela, mas não para nenhum índice secundário global. Como visualizar <code>ProvisionedWriteCapacityUnits</code> para um índice secundário global, você deve especificar tanto <code>TableName</code> e <code>GlobalSecondaryIndexName</code>.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum— A configuração mais baixa para capacidade de gravação provisionada. Se você usar <code>UpdateTable</code> para aumentar a capacidade de gravação, essa métrica mostra o valor mais baixo de <code>WriteCapacityUnits</code> durante este período de tempo. • Maximum— A configuração mais alta para capacidade de gravação provisionada. Se você usar <code>UpdateTable</code> para diminuir a capacidade de gravação, essa métrica mostra o valor mais alto de <code>WriteCapacityUnits</code> durante este período de tempo. • Average— A capacidade média de gravação provisionada. A <code>ProvisionedWriteCapacityUnits</code> é publicada em intervalos de cinco minutos. Portanto, se você ajustar rapidamente as unidades de capacidade de gravação provisionadas, essa estatística pode não refletir a média real.

Métrica	Descrição
<code>ReadThrottleEvents</code>	<p>Solicitações ao DynamoDB que excedam as unidades de capacidade de leitura provisionadas para uma tabela ou um índice secundário global.</p> <p>Uma única solicitação pode resultar em vários eventos. Por exemplo, um <code>BatchGetItem</code> que lê 10 itens é processado como 10 <code>GetItem</code>. Para cada evento, <code>ReadThrottleEvents</code> é incrementado por um se esse evento é limitado. A métrica <code>ThrottledRequests</code> para todo o <code>BatchGetItem</code> não é incrementada a menos que todos os 10 <code>GetItem</code> eventos são limitados.</p> <p>O <code>TableName</code> retorna o <code>ReadThrottleEvents</code> para a tabela, mas não para nenhum índice secundário global. Como visualizar <code>ReadThrottleEvents</code> para um índice secundário global, você deve especificar tanto <code>TableName</code> e <code>GlobalSecondaryIndexName</code>.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>SampleCount</code> • <code>Sum</code>
<code>ReplicationLatency</code>	<p>(Esta métrica é para tabelas globais do DynamoDB.) O tempo decorrido entre um item atualizado que aparece no fluxo do DynamoDB para uma tabela de réplica e em outra réplica na tabela global.</p> <p>Unidades de :Milliseconds</p> <p>Dimensões: <code>TableName</code>, <code>ReceivingRegion</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Average</code> • <code>Minimum</code> • <code>Maximum</code>

Métrica	Descrição
<code>ReturnedBytes</code>	<p>O número de bytes retornados pelo <code>GetRecords</code> Operações (Streams do Amazon DynamoDB) durante o período especificado.</p> <p>Unidades de :Bytes</p> <p>Dimensões: <code>Operation</code>, <code>StreamLabel</code>, <code>TableName</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>
<code>ReturnedItemCount</code>	<p>O número de itens retornados por <code>Query</code>, <code>Scan</code> ou <code>ExecuteStatement</code> (selecionar) operações durante o período especificado.</p> <p>O número de itens retornados não é necessariamente o mesmo que o número de itens avaliados. Por exemplo, suponha que você tenha solicitado uma <code>Scan</code> em uma tabela ou um índice que tinha 100 itens, mas especificou um <code>FilterExpression</code> que reduziu os resultados para que apenas 15 itens fossem devolvidos. Nesse caso, a resposta da <code>Scan</code> conteria um <code>ScanCount</code> de 100 e um <code>Count</code> de 15 itens devolvidos.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>Operation</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • <code>Minimum</code> • <code>Maximum</code> • <code>Average</code> • <code>SampleCount</code> • <code>Sum</code>

Métrica	Descrição
ReturnedRecordsCount	<p>O número de registros de fluxo retornados porGetRecordsOperações (Streams do Amazon DynamoDB) durante o período especificado.</p> <p>Unidades de :Count</p> <p>Dimensões: Operation, StreamLabel, TableName</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • SampleCount • Sum
SuccessfulRequestLatency	<p>As solicitações bem-sucedidas para DynamoDB ou Amazon DynamoDB Streams durante o período especificado. SuccessfulRequestLatencyO pode fornecer dois tipos diferentes de informações:</p> <ul style="list-style-type: none"> • O tempo decorrido para solicitações bem-sucedidas (Minimum, Maximum, Sum, ou Average). • O número de solicitações bem-sucedidas (SampleCount). <p>SuccessfulRequestLatencyreflete a atividade somente nos Streams do DynamoDB ou do Amazon DynamoDB e não leva em consideração a latência da rede ou a atividade do lado do cliente.</p> <p>Unidades de :Milliseconds</p> <p>Dimensões: TableName, Operation</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Minimum • Maximum • Average • SampleCount
SystemErrors	<p>As solicitações ao DynamoDB ou ao Amazon DynamoDB Streams que geram um código de status HTTP 500 durante o período de tempo especificado. Um HTTP 500 geralmente indica um erro de serviço interno.</p> <p>Unidades de :Count</p> <p>Dimensões: TableName, Operation</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Sum • SampleCount

Métrica	Descrição
TimeToLiveDeletedItemCount	<p>O número de itens excluídos pelo Tempo de Vida (TTL) durante o período especificado. Essa métrica ajuda a monitorar a taxa de exclusões de TTL na tabela.</p> <p>Unidades de :Count</p> <p>Dimensões: TableName</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none">• Sum
ThrottledPutRecordCount	<p>O número de registros que falharam ao replicar para o fluxo de dados do Kinesis devido à capacidade insuficiente do Kinesis Data Stream.</p> <p>Unidades de :Count</p> <p>Dimensões: TableName, DelegateDoperation</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none">• Minimum• Maximum• Average• SampleCount

Métrica	Descrição
ThrottledRequests	<p>Solicitações ao DynamoDB que excedam os limites de throughput provisionados em um recurso (como uma tabela ou um índice).</p> <p>ThrottledRequests é incrementado em um se um evento em uma solicitação exceder o limite de throughput provisionado. Por exemplo, se você atualizar um item em uma tabela com índices secundários globais, haverá vários eventos — uma gravação na tabela e uma gravação em cada índice. Se um ou mais desses eventos forem limitados, então ThrottledRequests é incrementado por um.</p> <p>Note</p> <p>Em uma solicitação de lote (<code>BatchGetItem</code> ou <code>BatchWriteItem</code>), ThrottledRequests é incrementado somente se cada lote estiver limitado. Se qualquer solicitação individual dentro do lote for limitada, uma das seguintes métricas será incrementada:</p> <ul style="list-style-type: none"> • <code>ReadThrottleEvents</code> — Para um <code>GetItem</code> no <code>BatchGetItem</code>. • <code>WriteThrottleEvents</code> — Para um <code>PutItem</code> ou <code>DeleteItem</code> no <code>BatchWriteItem</code>. <p>Para obter uma ideia sobre qual evento está controlando a limitação de uma solicitação, compare ThrottledRequests com o <code>ReadThrottleEvents</code> e <code>WriteThrottleEvents</code> para a tabela e seus índices.</p> <p>Note</p> <p>Uma solicitação limitada resultará em um código de status HTTP 400. Todos esses eventos são refletidos no ThrottledRequests, mas não na <code>UserErrors</code> Métrica do.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>Operation</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> • Sum • SampleCount

Métrica	Descrição
TransactionConflict	<p>Solicitações de nível de item rejeitadas devido a conflitos transacionais entre solicitações concorrentes nos mesmos itens. Para obter mais informações, consulte Lidar com conflitos de transação no DynamoDB.</p> <p>Unidades de :Count</p> <p>Dimensões: TableName</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> Sum— O número de solicitações rejeitadas no nível do item devido a conflitos de transação. <p>Note</p> <p>Se várias solicitações de nível de item dentro de uma chamada para TransactWriteItems ou TransactGetItems foram rejeitados, Sum é incrementado em um para cada nível de item Put, Update, Delete, ou Get solicitação.</p> <ul style="list-style-type: none"> SampleCount— O número de solicitações rejeitadas devido a conflitos de transação. <p>Note</p> <p>Se várias solicitações de nível de item dentro de uma chamada para TransactWriteItems ou TransactGetItems são rejeitados, SampleCount é incrementada apenas em um.</p> <ul style="list-style-type: none"> Min— O número mínimo de solicitações rejeitadas em nível de item em uma chamada para TransactWriteItems, TransactGetItems, PutItem, UpdateItem ou DeleteItem. Max— O número máximo de solicitações rejeitadas em nível de item em uma chamada para TransactWriteItems, TransactGetItems, PutItem, UpdateItem ou DeleteItem. Average— O número médio de solicitações rejeitadas em nível de item em uma chamada para TransactWriteItems, TransactGetItems, PutItem, UpdateItem ou DeleteItem.

Métrica	Descrição
UserErrors	<p>Solicitações para DynamoDB ou Amazon DynamoDB Streams que geram um código de status HTTP 400 durante o período de tempo especificado. Um HTTP 400 geralmente indica um erro do lado do cliente, como uma combinação inválida de parâmetros, uma tentativa de atualizar uma tabela inexistente ou uma assinatura de solicitação incorreta.</p> <p>Todos esses eventos são refletidos no <code>UserErrors</code> Métrica, exceto pelo seguinte:</p> <ul style="list-style-type: none">• <code>ProvisionedThroughputExceededException</code>— Consulte o <code>ThrottledRequests</code> nesta seção.• <code>ConditionalCheckFailedException</code>— Consulte o <code>ConditionalCheckFailedRequests</code> nesta seção. <p><code>UserErrors</code> representa o agregado de erros HTTP 400 para solicitações do DynamoDB ou do Amazon DynamoDB Streams para o AWSRegião e a atualAWSconta.</p> <p>Unidades de :Count</p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none">• Sum• SampleCount

Métrica	Descrição
<code>WriteThrottleEvents</code>	<p>Solicitações ao DynamoDB que excedam as unidades de capacidade de gravação provisionadas para uma tabela ou um índice secundário global.</p> <p>Uma única solicitação pode resultar em vários eventos. Por exemplo, se <code>PutItem</code> em uma tabela com três índices secundários globais resultaria em quatro eventos — a gravação da tabela e cada uma das três gravações do índice. Para cada evento, <code>WriteThrottleEvents</code> é incrementada por um se esse evento for limitado. Para <code>singlePutItem</code> solicitações, se algum dos eventos for limitado, <code>ThrottledRequests</code> também é incrementada em um. Para <code>BatchWriteItem</code>, <code>ThrottledRequests</code> é a métrica para todo <code>BatchWriteItem</code> e não é incrementado a menos que todos os indivíduos <code>PutItem</code> ou <code>DeleteItem</code> eventos são limitados.</p> <p>O <code>TableName</code> retorna <code>WriteThrottleEvents</code> para a tabela, mas não para nenhum índice secundário global. Como visualizar <code>WriteThrottleEvents</code> para um índice secundário global, você deve especificar tanto <code>TableName</code> e <code>GlobalSecondaryIndexName</code>.</p> <p>Unidades de :Count</p> <p>Dimensões: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Estatísticas válidas:</p> <ul style="list-style-type: none"> Sum SampleCount

Dimensões para métricas do DynamoDB

As métricas para o DynamoDB são qualificadas de acordo com os valores de conta, nome da tabela, nome do índice secundário global ou operação. Você pode usar o console do CloudWatch para recuperar dados do DynamoDB em qualquer uma das dimensões da tabela abaixo.

Dimensão	Descrição
<code>DelegatedOperation</code>	<p>Essa dimensão limita os dados às operações que o DynamoDB executa em seu nome. As seguintes operações são capturadas:</p> <ul style="list-style-type: none"> Alterar a captura de dados para o Kinesis Data Streams.
<code>GlobalSecondaryIndexName</code>	<p>Essa dimensão limita os dados a um índice secundário global em uma tabela. Se você especificar <code>GlobalSecondaryIndexName</code>, também deve especificar <code>TableName</code>.</p>
<code>Operation</code>	<p>Essa dimensão limita os dados a uma das seguintes operações do DynamoDB:</p> <ul style="list-style-type: none"> <code>PutItem</code> <code>DeleteItem</code> <code>UpdateItem</code>

Dimensão	Descrição
	<ul style="list-style-type: none"> • <code>GetItem</code> • <code>BatchGetItem</code> • <code>Scan</code> • <code>Query</code> • <code>BatchWriteItem</code> • <code>TransactWriteItems</code> • <code>TransactGetItems</code> • <code>ExecuteTransaction</code> • <code>BatchExecuteStatement</code> • <code>ExecuteStatement</code> <p>Além disso, você pode limitar os dados à seguinte operação do Amazon DynamoDB Streams:</p> <ul style="list-style-type: none"> • <code>GetRecords</code>
<code>OperationType</code>	<p>Essa dimensão limita os dados a um dos seguintes tipos de operação:</p> <ul style="list-style-type: none"> • <code>Read</code> • <code>Write</code> <p>Esta dimensão é emitida para <code>ExecuteTransaction</code>, <code>BatchExecuteStatement</code> e <code>Solicitações</code>.</p>
<code>Verb</code>	<p>Essa dimensão limita os dados a um dos seguintes verbos do PartiQL DynamoDB:</p> <ul style="list-style-type: none"> • Inserir:<code>PartiQLInsert</code> • SELECT:<code>PartiQLSelect</code> • Atualização:<code>PartiQLUpdate</code> • Excluir:<code>PartiQLDelete</code> <p>Essa dimensão é emitida para a propriedade <code>ExecuteStatement</code> operação.</p>
<code>ReceivingRegion</code>	<p>Esta dimensão limita os dados a um determinado AWS Região. Ele é usado com métricas originadas de tabelas de réplica em uma tabela global do DynamoDB.</p>
<code>StreamLabel</code>	<p>Essa dimensão limita os dados a um rótulo de fluxo específico. Ele é usado com métricas originadas do Amazon DynamoDB Streams <code>GetRecords</code> Operações</p>
<code>TableName</code>	<p>Essa dimensão limita os dados a uma tabela específica. Este valor pode ser qualquer nome de tabela na região atual e o AWS conta.</p>

Como faço para usar métricas do DynamoDB?

As métricas informadas pelo DynamoDB fornecem informações que podem ser analisadas de diferentes maneiras. A lista a seguir mostra alguns usos comuns para as métricas. Essas são sugestões para você começar, e não uma lista abrangente.

Como?	Métricas relevantes
How can I monitor the rate of TTL deletions on my table?	Você pode monitorar <code>TimeToLiveDeletedItemCount</code> ao longo do período de tempo especificado, para rastrear a taxa de exclusões de TTL em sua tabela. Para obter um exemplo de um aplicativo sem servidor usando o comando <code>TimeToLiveDeletedItemCount</code> métrica, consulte Arquive itens automaticamente no S3 usando o DynamoDB Time to Live (TTL) com o AWS Lambda e Amazon Kinesis Data Firehose .
How can I determine how much of my provisioned throughput is being used?	Você pode monitorar <code>ConsumedReadCapacityUnits</code> ou <code>ConsumedWriteCapacityUnits</code> ao longo do período de tempo especificado, para rastrear a quantidade de throughput provisionado que está sendo usada.
How can I determine which requests exceed the provisioned throughput quotas of a table?	<code>ThrottledRequests</code> será incrementado em um se um evento em uma solicitação exceder a cota de uma taxa de transferência provisionada. Para ter uma ideia sobre qual evento está controlando a utilização de uma solicitação, compare <code>ThrottledRequests</code> com as métricas <code>ReadThrottleEvents</code> e <code>WriteThrottleEvents</code> da tabela e seus índices.
How can I determine if any system errors occurred?	Você pode monitorar <code>SystemErrors</code> para determinar se todas as solicitações resultaram em um código HTTP 500 (erro de servidor). Normalmente, essa métrica deve ser igual a zero. Se não for o caso, talvez você deva investigar. Note Você pode encontrar erros de servidor internos enquanto trabalha com itens. Esses são esperados durante a vida útil de uma tabela. Todas as solicitações com falha podem ser repetidas imediatamente.

Criação de alarmes do CloudWatch para monitorar o DynamoDB

Você pode criar um alarme do CloudWatch que envia uma mensagem do Amazon SNS quando o alarme muda de estado. Um alarme observa uma única métrica ao longo de um período especificado por você e realiza uma ou mais ações com base no valor da métrica relativo a um determinado limite ao longo de vários períodos. A ação é uma notificação enviada para um tópico do Amazon SNS ou uma política de Auto Scaling. Os alertas invocam ações apenas para alterações de estado mantidas. Os alarmes do CloudWatch não invocam ações simplesmente porque estão em um estado específico. O estado deve ter sido alterado e mantido por um número específico de períodos.

Note

Você deve especificar todas as dimensões necessárias ao criar o alarme do CloudWatch, pois o CloudWatch não agregará métricas para uma dimensão ausente. Criar um alarme do CloudWatch com uma dimensão ausente não resultará em um erro, ao criar o alarme.

Para obter uma lista de métricas suportadas e suas dimensões necessárias no DynamoDB, consulte [Dimensões e métricas do DynamoDB \(p. 930\)](#).

Como posso ser notificado antes de consumir toda a minha capacidade de leitura?

1. Crie um tópico do Amazon SNSarn:aws:sns:us-east-1:123456789012:capacity-alarm.

Para obter mais informações, consulte [Configurar o Amazon Simple Notification Service](#).

2. Crie o alerta. Neste exemplo, pressupomos uma capacidade provisionada de cinco unidades de leitura.

```
aws cloudwatch put-metric-alarm \
    --alarm-name ReadCapacityUnitsLimitAlarm \
    --alarm-description "Alarm when read capacity reaches 80% of my provisioned read
    capacity" \
    --namespace AWS/DynamoDB \
    --metric-name ConsumedReadCapacityUnits \
    --dimensions Name=TableName,Value=myTable \
    --statistic Sum \
    --threshold 240 \
    --comparison-operator GreaterThanOrEqualToThreshold \
    --period 60 \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:capacity-alarm
```

3. Teste o alerta.

```
aws cloudwatch set-alarm-state --alarm-name ReadCapacityUnitsLimitAlarm --state-reason
    "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name ReadCapacityUnitsLimitAlarm --state-reason
    "initializing" --state-value ALARM
```

Note

O alarme é ativado sempre que a capacidade de leitura consumida for de, pelo menos, 4 unidades por segundo (80% da capacidade de leitura provisionada de 5) para 1 minuto (60 segundos). Então, o threshold é 240 unidades de capacidade de leitura (4 unidades/s * 60 segundos). Sempre que a capacidade de leitura for atualizada, você deverá atualizar os cálculos de alarme adequadamente. Você pode evitar esse processo com a criação de alarmes por meio do DynamoDB Console. Dessa forma, os alarmes são atualizados automaticamente por você.

Como posso ser notificado caso alguma solicitação exceda as cotas de taxa de transferência provisionada de uma tabela?

1. Crie um tópico do Amazon SNSarn:aws:sns:us-east-1:123456789012:requests-exceeding-throughput.

Para obter mais informações, consulte [Configurar o Amazon Simple Notification Service](#).

2. Crie o alerta.

```
aws cloudwatch put-metric-alarm \
    --alarm-name RequestsExceedingThroughputAlarm \
    --alarm-description "Alarm when my requests are exceeding provisioned throughput
    quotas of a table" \
    --namespace AWS/DynamoDB \
    --metric-name ThrottledRequests \
    --dimensions Name=TableName,Value=myTable Name=Operation,Value=aDynamoDBOperation \
```

```
--statistic Sum \
--threshold 0 \
--comparison-operator GreaterThanThreshold \
--period 300 \
--unit Count \
--evaluation-periods 1 \
--alarm-actions arn:aws:sns:us-east-1:123456789012:requests-exceeding-throughput
```

3. Teste o alerta.

```
aws cloudwatch set-alarm-state --alarm-name RequestsExceedingThroughputAlarm --state-reason "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name RequestsExceedingThroughputAlarm --state-reason "initializing" --state-value ALARM
```

Como posso ser notificado se algum erro do sistema ocorreu?

1. Crie um tópico do Amazon SNSarn:aws:sns:us-east-1:123456789012:notify-on-system-errors.
Para obter mais informações, consulte [Configurar o Amazon Simple Notification Service](#).
2. Crie o alerta.

```
aws cloudwatch put-metric-alarm \
--alarm-name SystemErrorsAlarm \
--alarm-description "Alarm when system errors occur" \
--namespace AWS/DynamoDB \
--metric-name SystemErrors \
--dimensions Name=TableName,Value=myTable Name=Operation,Value=aDynamoDBOperation \
--statistic Sum \
--threshold 0 \
--comparison-operator GreaterThanThreshold \
--period 60 \
--unit Count \
--evaluation-periods 1 \
--alarm-actions arn:aws:sns:us-east-1:123456789012:notify-on-system-errors
```

3. Teste o alerta.

```
aws cloudwatch set-alarm-state --alarm-name SystemErrorsAlarm --state-reason "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name SystemErrorsAlarm --state-reason "initializing" --state-value ALARM
```

Registro de operações do DynamoDB usando oAWS CloudTrail

O DynamoDB é integrado aoAWS CloudTrail, um serviço que fornece um registro de ações executadas por um usuário, uma função ou umaAWSserviço no DynamoDB. O CloudTrail captura todas as chamadas de API do DynamoDB como eventos. As chamadas capturadas incluem chamadas ao console do DynamoDB e chamadas de código para as operações da API do DynamoDB, usando o PartiQL e a API clássica. Se você criar uma trilha, poderá habilitar a entrega contínua de eventos do CloudTrail para um bucket do

Amazon S3, incluindo eventos para DynamoDB. Se você não configurar uma trilha, ainda poderá visualizar os eventos mais recentes no console do CloudTrail em Event history (Histórico de eventos). Usando as informações coletadas pelo CloudTrail, é possível determinar a solicitação feita para o DynamoDB, o endereço IP no qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita, além de detalhes adicionais.

Para monitoramento e alertas robustos, você também pode integrar eventos do CloudTrail com [Amazon CloudWatch Logs](#). Para aprimorar sua análise da atividade de serviço do DynamoDB e identificar alterações nas atividades de umAWSConta, é possível consultar AWS CloudTrail logs usando [Amazon Athena](#). Por exemplo, é possível usar consultas para identificar tendências e isolar ainda mais a atividade por atributos como endereço IP de origem ou usuário.

Para saber mais sobre o CloudTrail, incluindo como configurá-lo e ativá-lo, consulte o [AWS CloudTrailGuia do usuário do](#).

Informações do DynamoDB no CloudTrail

O CloudTrail é habilitado em sua conta da AWS quando ela é criada. Quando a atividade do evento compatível ocorrer no DynamoDB, ela será registrada em um evento do CloudTrail juntamente com outrosAWSeventos de serviço em Histórico do evento. Você pode visualizar, pesquisar e fazer download de eventos recentes em sua conta da AWS. Para mais informações, consulte [Visualizar eventos com o histórico de eventos do CloudTrail](#).

Para obter um registro contínuo de eventos noAWSA conta, incluindo eventos do DynamoDB, crie uma trilha. Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as regiões da AWS. A trilha registra em log eventos de todas as regiões na partição da AWS e entrega os arquivos de log para o bucket do Amazon S3 especificado por você. Além disso, você pode configurar outros produtos da AWS para analisar mais profundamente e agir sobre os dados de eventos coletados nos logs do CloudTrail. Para obter mais informações, consulte:

- Visão geral da criação de uma trilha
- Serviços e integrações compatíveis com o CloudTrail
- Configuração de notificações do Amazon SNS para o CloudTrail
- Receber arquivos de log do CloudTrail de várias regiões e receber arquivos de log do CloudTrail de várias contas

Controlar eventos de plano no CloudTrail

As ações da seguinte API são registradas em log por padrão como eventos nos arquivos do CloudTrail:

Amazon DynamoDB

- [CreateBackup](#)
- [CreateGlobalTable](#)
- [CreateTable](#)
- [DeleteBackup](#)
- [DeleteTable](#)
- [DescribeBackup](#)
- [DescribeContinuousBackups](#)
- [DescribeGlobalTable](#)
- [DescribeLimits](#)
- [DescribeTable](#)

- [DescribeTimeToLive](#)
- [ListBackups](#)
- [ListTables](#)
- [ListTagsOfResource](#)
- [ListGlobalTables](#)
- [RestoreTableFromBackup](#)
- [RestoreTableToPointInTime](#)
- [TagResource](#)
- [UntagResource](#)
- [UpdateGlobalTable](#)
- [UpdateTable](#)
- [UpdateTimeToLive](#)
- [DescribeReservedCapacity](#)
- [DescribeReservedCapacityOfferings](#)
- [DescribeScalableTargets](#)
- [RegisterScalableTarget](#)
- [PurchaseReservedCapacityOfferings](#)

DynamoDB Streams

- [DescribeStream](#)
- [ListStreams](#)

DynamoDB Accelerator (DAX)

- [CreateCluster](#)
- [CreateParameterGroup](#)
- [CreateSubnetGroup](#)
- [DecreaseReplicationFactor](#)
- [DeleteCluster](#)
- [DeleteParameterGroup](#)
- [DeleteSubnetGroup](#)
- [DescribeClusters](#)
- [DescribeDefaultParameters](#)
- [DescribeEvents](#)
- [DescribeParameterGroups](#)
- [DescribeParameters](#)
- [DescribeSubNetGroups](#)
- [IncreaseReplicationFactor](#)
- [ListTags](#)
- [RebootNode](#)
- [TagResource](#)
- [UntagResource](#)
- [UpdateCluster](#)

- [UpdateParameterGroup](#)
- [UpdateSubnetGroup](#)

Eventos de plano de dados do DynamoDB no CloudTrail

Para habilitar o registro das seguintes ações de API nos arquivos do CloudTrail, você precisará ativar o registro da atividade da API do plano de dados no CloudTrail. Consulte[Registro de eventos de dados para trilhas](#)Para obter mais informações.

Amazon DynamoDB

- [BatchExecuteStatement](#)
- [BatchGetItem](#)
- [BatchWriteItem](#)
- [DeleteItem](#)
- [ExecuteStatement](#)
- [ExecuteTransaction](#)
- [GetItem](#)
- [PutItem](#)
- [Consulta](#)
- [Scan](#)
- [TransactGetItems](#)
- [TransactWriteItems](#)
- [UpdateItem](#)

Note

As ações do plano de dados do DynamoDB não são registradas em log pelo CloudTrail

DynamoDB Streams

- [GetRecords](#)
- [GetShardIterator](#)

Noções básicas sobre as entradas dos arquivos de log do

Uma trilha é uma configuração que permite a entrega de eventos como registros de log a um bucket do Amazon S3 especificado. Os arquivos de log do CloudTrail contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer origem e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros de solicitação e assim por diante.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário raiz ou do AWS Identity and Access Management (IAM).
- Se a solicitação foi feita com credenciais de segurança temporárias de uma função ou de um usuário federado.
- Se a solicitação foi feita por outro serviço da AWS.

Note

Valores de atributo não chave serão editados nos logs de ações do CloudTrail usando a API do PartiQL e não aparecerão em logs de ações usando a API clássica.

Para obter mais informações, consulte o [Elemento userIdentity do CloudTrail](#).

Os exemplos a seguir demonstram logs do CloudTrail desses tipos de eventos:

Amazon DynamoDB

- [UpdateTable](#) (p. 958)
 - [DeleteTable](#) (p. 959)
 - [CreateCluster](#) (p. 960)
 - [PutItem\(bem-sucedida\)](#) (p. 961)
 - [UpdateItem\(infrutífero\)](#) (p. 962)
 - [TransactWriteItems\(bem-sucedida\)](#) (p. 963)
 - [TransactWriteItems\(comTransactionCanceledException\)](#) (p. 965)
 - [ExecuteStatement](#) (p. 967)
 - [BatchExecuteStatement](#) (p. 968)

DynamoDB Streams

- `GetRecords` (p. 969)

UpdateTable

```
        "provisionedThroughput": {
            "writeCapacityUnits": 25,
            "readCapacityUnits": 25
        },
        "responseElements": {
            "tableDescription": {
                "tableName": "Music",
                "attributeDefinitions": [
                    {
                        "attributeType": "S",
                        "attributeName": "Artist"
                    },
                    {
                        "attributeType": "S",
                        "attributeName": "SongTitle"
                    }
                ],
                "itemCount": 0,
                "provisionedThroughput": {
                    "writeCapacityUnits": 10,
                    "numberOfDecreasesToday": 0,
                    "readCapacityUnits": 10,
                    "lastIncreaseDateTime": "May 3, 2015 11:34:14 PM"
                },
                "creationDateTime": "May 3, 2015 11:34:14 PM",
                "keySchema": [
                    {
                        "attributeName": "Artist",
                        "keyType": "HASH"
                    },
                    {
                        "attributeName": "SongTitle",
                        "keyType": "RANGE"
                    }
                ],
                "tableStatus": "UPDATING",
                "tableSizeBytes": 0
            }
        },
        "requestID": "AALNP0J2L244N5O15PKISJ1KUFVV4KQNSO5AEMVJF66Q9ASUAAJG",
        "eventID": "eb834e01-f168-435f-92c0-c36278378b6e",
        "eventType": "AwsApiCall",
        "apiVersion": "2012-08-10",
        "recipientAccountId": "111122223333"
    }
}
```

DeleteTable

```
{
    "Records": [
        {
            "eventVersion": "1.03",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "AKIAIOSFODNN7EXAMPLE:bob",
                "arn": "arn:aws:sts::111122223333:assumed-role/users/bob",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "sessionContext": {
                    "attributes": {
                        "mfaAuthenticated": "false",

```

```
        "creationDate": "2015-05-28T18:06:01Z"
    },
    "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::444455556666:role/admin-role",
        "accountId": "444455556666",
        "userName": "bob"
    }
},
"eventTime": "2015-05-04T13:38:20Z",
"eventSource": "dynamodb.amazonaws.com",
"eventName": "DeleteTable",
"awsRegion": "us-west-2",
"sourceIPAddress": "192.0.2.0",
"userAgent": "console.aws.amazon.com",
"requestParameters": {
    "tableName": "Music"
},
"responseElements": {
    "tableDescription": {
        "tableName": "Music",
        "itemCount": 0,
        "provisionedThroughput": {
            "writeCapacityUnits": 25,
            "numberOfDecreasesToday": 0,
            "readCapacityUnits": 25
        },
        "tableStatus": "DELETING",
        "tableSizeBytes": 0
    }
},
"requestID": "4KBNVRGD25RG1KEO9UT4V3FQDJVV4KQNSO5AEMVJF66Q9ASUAAJG",
"eventID": "a954451c-c2fc-4561-8aea-7a30ba1fdf52",
"eventType": "AwsApiCall",
"apiVersion": "2012-08-10",
"recipientAccountId": "111122223333"
}
]
```

CreateCluster

```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "AKIAIOSFODNN7EXAMPLE:bob",
                "arn": "arn:aws:sts::111122223333:assumed-role/users/bob",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "bob"
            },
            "eventTime": "2019-12-17T23:17:34Z",
            "eventSource": "dax.amazonaws.com",
            "eventName": "CreateCluster",
            "awsRegion": "us-west-2",
            "sourceIPAddress": "192.0.2.0",
            "userAgent": "aws-cli/1.16.304 Python/3.6.9
Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 botocore/1.13.40",
            "requestParameters": {

```

```
"sSESpecification": {
    "enabled": true
},
"clusterName": "daxcluster",
"nodeType": "dax.r4.large",
"replicationFactor": 3,
"iamRoleArn": "arn:aws:iam::111122223333:role/
DAXServiceRoleForDynamoDBAccess"
},
"responseElements": {
    "cluster": {
        "securityGroups": [
            {
                "securityGroupIdentifier": "sg-1af6e36e",
                "status": "active"
            }
        ],
        "parameterGroup": {
            "nodeIdsToReboot": [],
            "parameterGroupName": "default.dax1.0",
            "parameterApplyStatus": "in-sync"
        },
        "clusterDiscoveryEndpoint": {
            "port": 8111
        },
        "clusterArn": "arn:aws:dax:us-west-2:111122223333:cache/daxcluster",
        "status": "creating",
        "subnetGroup": "default",
        "sSDEDescription": {
            "status": "ENABLED",
            "kMSMasterKeyArn": "arn:aws:kms:us-
west-2:111122223333:key/764898e4-adb1-46d6-a762-e2f4225b4fc4"
        },
        "iamRoleArn": "arn:aws:iam::111122223333:role/
DAXServiceRoleForDynamoDBAccess",
        "clusterName": "daxcluster",
        "activeNodes": 0,
        "totalNodes": 3,
        "preferredMaintenanceWindow": "thu:13:00-thu:14:00",
        "nodeType": "dax.r4.large"
    }
},
"requestID": "585adc5f-ad05-4e27-8804-70ba1315f8fd",
"eventID": "29158945-28da-4e32-88e1-56d1b90c1a0c",
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}
]
```

PutItem(bem-sucedida)

```
{
    "Records": [
        {
            "eventVersion": "1.06",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "AKIAIOSFODNN7EXAMPLE:bob",
                "arn": "arn:aws:sts::111122223333:assumed-role/users/bob",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "sessionContext": {
                    "attributes": {

```

```
        "mfaAuthenticated": "false",
        "creationDate": "2015-05-28T18:06:01Z"
    },
    "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::44445556666:role/admin-role",
        "accountId": "44445556666",
        "userName": "bob"
    }
},
"eventTime": "2019-01-19T15:41:54Z",
"eventSource": "dynamodb.amazonaws.com",
"eventName": "PutItem",
"awsRegion": "us-west-2",
"sourceIPAddress": "192.0.2.0",
"userAgent": "aws-cli/1.15.64 Python/2.7.16 Darwin/17.7.0 botocore/1.10.63",
"requestParameters": {
    "tableName": "Music",
    "key": {
        "Artist": "No One You Know",
        "SongTitle": "Scared of My Shadow"
    },
    "item": [
        "Artist",
        "SongTitle",
        "AlbumTitle"
    ],
    "returnConsumedCapacity": "TOTAL"
},
"responseElements": null,
"requestID": "4KBNVRGD25RG1KEO9UT4V3FQDJVV4KQNSO5AEMVJF66Q9ASUAAJG",
"eventID": "a954451c-c2fc-4561-8aea-7a30ba1fdf52",
"readOnly": false,
"resources": [
    {
        "accountId": "111122223333",
        "type": "AWS::DynamoDB::Table",
        "ARN": "arn:aws:dynamodb:us-west-2:123456789012:table/Music"
    }
],
"eventType": "AwsApiCall",
"apiVersion": "2012-08-10",
"managementEvent": false,
"recipientAccountId": "111122223333",
"eventCategory": "Data"
}
]
```

UpdateItem(infrutífero)

```
{
    "Records": [
        {
            "eventVersion": "1.07",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "AKIAIOSFODNN7EXAMPLE:bob",
                "arn": "arn:aws:sts::111122223333:assumed-role/users/bob",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "sessionContext": {
                    "contextString": "string"
                }
            }
        }
    ]
}
```

```
"sessionIssuer": {  
    "type": "Role",  
    "principalId": "AKIAI44QH8DHBEEXAMPLE",  
    "arn": "arn:aws:iam::444455556666:role/admin-role",  
    "accountId": "444455556666",  
    "userName": "bob"  
},  
"attributes": {  
    "creationDate": "2020-09-03T22:14:13Z",  
    "mfaAuthenticated": "false"  
}  
}  
},  
"eventTime": "2020-09-03T22:27:15Z",  
"eventSource": "dynamodb.amazonaws.com",  
"eventName": "UpdateItem",  
"awsRegion": "us-west-2",  
"sourceIPAddress": "192.0.2.0",  
"userAgent": "aws-cli/1.15.64 Python/2.7.16 Darwin/17.7.0 botocore/1.10.63",  
"errorCode": "ConditionalCheckFailedException",  
"errorMessage": "The conditional request failed",  
"requestParameters": {  
    "tableName": "Music",  
    "key": {  
        "Artist": "No One You Know",  
        "SongTitle": "Call Me Today"  
    },  
    "updateExpression": "SET #Y = :y, #AT = :t",  
    "expressionAttributeNames": {  
        "#Y": "Year",  
        "#AT": "AlbumTitle"  
    },  
    "conditionExpression": "attribute_not_exists(#Y)",  
    "returnConsumedCapacity": "TOTAL"  
},  
"responseElements": null,  
"requestID": "4KBNVRGD25RG1KEO9UT4V3FQDJVV4KQNSO5AEMVJF66Q9ASUAAJG",  
"eventID": "a954451c-c2fc-4561-8aea-7a30ba1fdf52",  
"readOnly": false,  
"resources": [  
    {  
        "accountId": "111122223333",  
        "type": "AWS::DynamoDB::Table",  
        "ARN": "arn:aws:dynamodb:us-west-2:123456789012:table/Music"  
    }  
],  
"eventType": "AwsApiCall",  
"apiVersion": "2012-08-10",  
"managementEvent": false,  
"recipientAccountId": "111122223333",  
"eventCategory": "Data"  
}  
]  
}
```

TransactWriteItems(bem-sucedida)

```
{  
    "Records": [  
        {  
            "eventVersion": "1.07",  
            "userIdentity": {  
                "type": "AssumedRole",  
                "principalId": "AKIAIOSFODNN7EXAMPLE:bob",  
                "arn": "arn:aws:iam::123456789012:assumed-role/AdminRole/bob"  
            },  
            "awsRegion": "us-west-2",  
            "dynamodb": {  
                "RequestResponses": [  
                    {  
                        "DeleteRequest": {  
                            "Key": {  
                                "SongTitle": "Call Me Today",  
                                "Artist": "No One You Know"  
                            },  
                            "TableName": "Music"  
                        },  
                        "PutRequest": {  
                            "Item": {  
                                "SongTitle": "I Wanna Dance With Somebody",  
                                "Artist": "Gwen Stefani",  
                                "Year": 1995, "AlbumTitle": "No Doubt"  
                            },  
                            "TableName": "Music"  
                        }  
                    }  
                ],  
                "Version": "2012-08-10"  
            },  
            "eventSource": "aws:dynamodb",  
            "eventTime": "2020-09-03T22:27:15Z",  
            "eventName": "TransactWriteItems",  
            "awsRegion": "us-west-2",  
            "userAgent": "aws-sdk-go/1.4.10 Go/1.13.5",  
            "errorCode": null,  
            "errorMessage": null,  
            "requestParameters": {  
                "TransactStatements": [  
                    {  
                        "Statement": "TRANSACTIONAL  
                            BEGIN  
                                DELETE FROM Music WHERE SongTitle = :SongTitle AND Artist = :Artist  
                                PUT Item = :Item  
                            COMMIT  
                        ",  
                        "ConditionExpression": "attribute_not_exists(SongTitle)",  
                        "ExpressionAttributeNames": {  
                            ":SongTitle": "SongTitle",  
                            ":Artist": "Artist",  
                            ":Item": {  
                                "SongTitle": "I Wanna Dance With Somebody",  
                                "Artist": "Gwen Stefani",  
                                "Year": 1995, "AlbumTitle": "No Doubt"  
                            }  
                        },  
                        "ExpressionAttributeValues": {  
                            ":SongTitle": "Call Me Today",  
                            ":Artist": "No One You Know",  
                            ":Item": {  
                                "SongTitle": "I Wanna Dance With Somebody",  
                                "Artist": "Gwen Stefani",  
                                "Year": 1995, "AlbumTitle": "No Doubt"  
                            }  
                        }  
                    }  
                ]  
            },  
            "responseElements": null,  
            "requestID": "4KBNVRGD25RG1KEO9UT4V3FQDJVV4KQNSO5AEMVJF66Q9ASUAAJG",  
            "eventID": "a954451c-c2fc-4561-8aea-7a30ba1fdf52",  
            "readOnly": false,  
            "resources": [  
                {  
                    "accountId": "111122223333",  
                    "type": "AWS::DynamoDB::Table",  
                    "ARN": "arn:aws:dynamodb:us-west-2:123456789012:table/Music"  
                }  
            ],  
            "eventType": "AwsApiCall",  
            "apiVersion": "2012-08-10",  
            "managementEvent": false,  
            "recipientAccountId": "111122223333",  
            "eventCategory": "Data"  
        }  
    ]  
}
```

```
"arn": "arn:aws:sts::111122223333:assumed-role/users/bob",
"accountId": "111122223333",
"accessKeyId": "AKIAIOSFODNN7EXAMPLE",
"sessionContext": {
    "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::444455556666:role/admin-role",
        "accountId": "444455556666",
        "userName": "bob"
    },
    "attributes": {
        "creationDate": "2020-09-03T22:14:13Z",
        "mfaAuthenticated": "false"
    }
},
"eventTime": "2020-09-03T21:48:12Z",
"eventSource": "dynamodb.amazonaws.com",
"eventName": "TransactWriteItems",
"awsRegion": "us-west-1",
"sourceIPAddress": "192.0.2.0",
"userAgent": "aws-cli/1.15.64 Python/2.7.16 Darwin/17.7.0 botocore/1.10.63",
"requestParameters": {
    "requestItems": [
        {
            "operation": "Put",
            "tableName": "Music",
            "key": {
                "Artist": "No One You Know",
                "SongTitle": "Call Me Today"
            },
            "items": [
                "Artist",
                "SongTitle",
                "AlbumTitle"
            ],
            "conditionExpression": "#AT = :A",
            "expressionAttributeNames": {
                "#AT": "AlbumTitle"
            },
            "returnValuesOnConditionCheckFailure": "ALL_OLD"
        },
        {
            "operation": "Update",
            "tableName": "Music",
            "key": {
                "Artist": "No One You Know",
                "SongTitle": "Call Me Tomorrow"
            },
            "updateExpression": "SET #AT = :newval",
            "ConditionExpression": "attribute_not_exists(Rating)",
            "ExpressionAttributeNames": {
                "#AT": "AlbumTitle"
            },
            "returnValuesOnConditionCheckFailure": "ALL_OLD"
        },
        {
            "operation": "Delete",
            "TableName": "Music",
            "key": {
                "Artist": "No One You Know",
                "SongTitle": "Call Me Yesterday"
            },
            "conditionExpression": "#P between :lo and :hi",
            "expressionAttributeNames": {
```

```
        "#P": "Price"
    },
    "ReturnValuesOnConditionCheckFailure": "ALL_OLD"
},
{
    "operation": "ConditionCheck",
    "TableName": "Music",
    "Key": {
        "Artist": "No One You Know",
        "SongTitle": "Call Me Now"
    },
    "ConditionExpression": "#P between :lo and :hi",
    "ExpressionAttributeNames": {
        "#P": "Price"
    },
    "ReturnValuesOnConditionCheckFailure": "ALL_OLD"
}
],
"returnConsumedCapacity": "TOTAL",
"returnItemCollectionMetrics": "SIZE"
},
"responseElements": null,
"requestID": "45EN320M6TQSMV2MI6504L5TNFVV4KQNSO5AEMVJF66Q9ASUAAJG",
"eventID": "4f1cc78b-5c94-4174-a6ad-3ee78605381c",
"readOnly": false,
"resources": [
    {
        "accountId": "111122223333",
        "type": "AWS::DynamoDB::Table",
        "ARN": "arn:aws:dynamodb:us-west-2:123456789012:table/Music"
    }
],
"eventType": "AwsApiCall",
"apiVersion": "2012-08-10",
"managementEvent": false,
"recipientAccountId": "111122223333",
"eventCategory": "Data"
}
]
}
```

TransactWriteItems(comTransactionCanceledException)

```
{
    "Records": [
        {
            "eventVersion": "1.06",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "AKIAIOSFODNN7EXAMPLE:bob",
                "arn": "arn:aws:sts::111122223333:assumed-role/users/bob",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "sessionContext": {
                    "sessionIssuer": {
                        "type": "Role",
                        "principalId": "AKIAI44QH8DHBEEXAMPLE",
                        "arn": "arn:aws:iam::444455556666:role/admin-role",
                        "accountId": "444455556666",
                        "userName": "bob"
                    },
                    "attributes": {
                        "creationDate": "2020-09-03T22:14:13Z",
                        "mfaAuthenticated": "false"
                    }
                }
            }
        }
    ]
}
```

```
        }
    },
    "eventTime": "2019-02-01T00:42:34Z",
    "eventSource": "dynamodb.amazonaws.com",
    "eventName": "TransactWriteItems",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.16.93 Python/3.4.7
Linux/4.9.119-0.1.ac.277.71.329.metal1.x86_64 botocore/1.12.83",
    "errorCode": "TransactionCanceledException",
    "errorMessage": "Transaction cancelled, please refer cancellation reasons for
specific reasons [ConditionalCheckFailed, None]",
    "requestParameters": {
        "requestItems": [
            {
                "operation": "Put",
                "tableName": "Music",
                "key": {
                    "Artist": "No One You Know",
                    "SongTitle": "Call Me Today"
                },
                "items": [
                    "Artist",
                    "SongTitle",
                    "AlbumTitle"
                ],
                "conditionExpression": "#AT = :A",
                "expressionAttributeNames": {
                    "#AT": "AlbumTitle"
                },
                "returnValuesOnConditionCheckFailure": "ALL_OLD"
            },
            {
                "operation": "Update",
                "tableName": "Music",
                "key": {
                    "Artist": "No One You Know",
                    "SongTitle": "Call Me Tomorrow"
                },
                "updateExpression": "SET #AT = :newval",
                "ConditionExpression": "attribute_not_exists(Rating)",
                "ExpressionAttributeNames": {
                    "#AT": "AlbumTitle"
                },
                "returnValuesOnConditionCheckFailure": "ALL_OLD"
            },
            {
                "operation": "Delete",
                "TableName": "Music",
                "key": {
                    "Artist": "No One You Know",
                    "SongTitle": "Call Me Yesterday"
                },
                "conditionExpression": "#P between :lo and :hi",
                "expressionAttributeNames": {
                    "#P": "Price"
                },
                "ReturnValuesOnConditionCheckFailure": "ALL_OLD"
            },
            {
                "operation": "ConditionCheck",
                "TableName": "Music",
                "Key": {
                    "Artist": "No One You Know",
                    "SongTitle": "Call Me Now"
                }
            }
        ]
    }
}
```

```
        },
        "ConditionExpression": "#P between :lo and :hi",
        "ExpressionAttributeNames": {
            "#P": "Price"
        },
        "ReturnValuesOnConditionCheckFailure": "ALL_OLD"
    }
],
"returnConsumedCapacity": "TOTAL",
"returnItemCollectionMetrics": "SIZE"
},
"responseElements": null,
"requestID": "A0GTQEKLBB9VD8E05REA5A3E1VVV4KQNSO5AEMVJF66Q9ASUAAJG",
"eventID": "43e437b5-908a-46af-84e6-e27fffb9c5cd",
"readOnly": false,
"resources": [
{
    "accountId": "111122223333",
    "type": "AWS::DynamoDB::Table",
    "ARN": "arn:aws:dynamodb:us-west-2:123456789012:table/Music"
}
],
"eventType": "AwsApiCall",
"apiVersion": "2012-08-10",
"managementEvent": false,
"recipientAccountId": "111122223333",
"eventCategory": "Data"
}
]
}
```

ExecuteStatement

```
{
    "Records": [
        {
            "eventVersion": "1.08",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "AKIAIOSFODNN7EXAMPLE:bob",
                "arn": "arn:aws:sts::111122223333:assumed-role/users/bob",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "sessionContext": {
                    "sessionIssuer": {
                        "type": "Role",
                        "principalId": "AKIAI44QH8DHBEXAMPLE",
                        "arn": "arn:aws:iam::444455556666:role/admin-role",
                        "accountId": "444455556666",
                        "userName": "bob"
                    },
                    "attributes": {
                        "creationDate": "2020-09-03T22:14:13Z",
                        "mfaAuthenticated": "false"
                    }
                }
            },
            "eventTime": "2021-03-03T23:06:45Z",
            "eventSource": "dynamodb.amazonaws.com",
            "eventName": "ExecuteStatement",
            "awsRegion": "us-west-2",
            "sourceIPAddress": "192.0.2.0",
            "userAgent": "aws-cli/1.19.7 Python/3.6.13
Linux/4.9.230-0.1.ac.223.84.332.metal1.x86_64 botocore/1.20.7",
            "version": "1.08"
        }
    ]
}
```

```
        "requestParameters": {
            "statement": "SELECT * FROM Music WHERE Artist = 'No One You Know' AND
SongTitle = 'Call Me Today' AND nonKeyAttr = ***(Redacted)"
        },
        "responseElements": null,
        "requestID": "V7G2KCSFLP830RB7MMFG6RIAD3VV4KQNSO5AEMVJF66Q9ASUAAJG",
        "eventID": "0b5c4779-e169-4227-a1de-6ed01dd18ac7",
        "readOnly": false,
        "resources": [
            {
                "accountId": "111122223333",
                "type": "AWS::DynamoDB::Table",
                "ARN": "arn:aws:dynamodb:us-west-2:123456789012:table/Music"
            }
        ],
        "eventType": "AwsApiCall",
        "apiVersion": "2012-08-10",
        "managementEvent": false,
        "recipientAccountId": "111122223333",
        "eventCategory": "Data"
    }
}
```

BatchExecuteStatement

```
{
    "Records": [
        {
            "eventVersion": "1.08",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "AKIAIOSFODNN7EXAMPLE:bob",
                "arn": "arn:aws:sts::111122223333:assumed-role/users/bob",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "sessionContext": {
                    "sessionIssuer": {
                        "type": "Role",
                        "principalId": "AKIAI44QH8DHBEEXAMPLE",
                        "arn": "arn:aws:iam::444455556666:role/admin-role",
                        "accountId": "444455556666",
                        "userName": "bob"
                    },
                    "attributes": {
                        "creationDate": "2020-09-03T22:14:13Z",
                        "mfaAuthenticated": "false"
                    }
                }
            },
            "eventTime": "2021-03-03T23:24:48Z",
            "eventSource": "dynamodb.amazonaws.com",
            "eventName": "BatchExecuteStatement",
            "awsRegion": "us-west-2",
            "sourceIPAddress": "192.0.2.0",
            "userAgent": "aws-cli/1.19.7 Python/3.6.13
Linux/4.9.230-0.1.ac.223.84.332.metal1.x86_64 botocore/1.20.7",
            "requestParameters": {
                "requestItems": [
                    {
                        "statement": "UPDATE Music SET Album = ***(Redacted) WHERE Artist =
'No One You Know' AND SongTitle = 'Call Me Today'"
                    },
                    {

```

```
        "statement": "INSERT INTO Music VALUE {'Artist' : ***(**Redacted),  
'SongTitle' : ***(**Redacted), 'Album' : ***(**Redacted)}"  
    }  
]  
,  
"responseElements": null,  
"requestID": "23PE7ED291UD65P9SMS6TISNVBVV4KQNSO5AEMVJF66Q9ASUAAJG",  
"eventID": "f863f966-b741-4c36-b15e-f867e829035a",  
"readOnly": false,  
"resources": [  
    {  
        "accountId": "111122223333",  
        "type": "AWS::DynamoDB::Table",  
        "ARN": "arn:aws:dynamodb:us-west-2:123456789012:table/Music"  
    },  
,  
"eventType": "AwsApiCall",  
"apiVersion": "2012-08-10",  
"managementEvent": false,  
"recipientAccountId": "111122223333",  
"eventCategory": "Data"  
}  
]  
}
```

GetRecords

```
{  
    "Records": [  
        {  
            "eventVersion": "1.08",  
            "userIdentity": {  
                "type": "AssumedRole",  
                "principalId": "AKIAIOSFODNN7EXAMPLE:bob",  
                "arn": "arn:aws:sts::111122223333:assumed-role/users/bob",  
                "accountId": "111122223333",  
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",  
                "sessionContext": {  
                    "sessionIssuer": {  
                        "type": "Role",  
                        "principalId": "AKIAI44QH8DHBEEXAMPLE",  
                        "arn": "arn:aws:iam::444455556666:role/admin-role",  
                        "accountId": "444455556666",  
                        "userName": "bob"  
                    },  
                    "attributes": {  
                        "creationDate": "2020-09-03T22:14:13Z",  
                        "mfaAuthenticated": "false"  
                    }  
                }  
            },  
            "eventTime": "2021-04-15T04:15:02Z",  
            "eventSource": "dynamodb.amazonaws.com",  
            "eventName": "GetRecords",  
            "awsRegion": "us-west-2",  
            "sourceIPAddress": "192.0.2.0",  
            "userAgent": "aws-cli/1.19.50 Python/3.6.13  
Linux/4.9.230-0.1.ac.224.84.332.metal1.x86_64 botocore/1.20.50",  
            "requestParameters": {  
                "shardIterator": "arn:aws:dynamodb:us-west-2:123456789012:table/  
Music/stream/2021-04-15T04:02:47.428|1|AAAAAAAAAAAH7HF3xwDQHBrvk2UBZ1PKh8bX3F  
+JeH0rFwHCE7dz4VGv1ZoJ5bMxQwkmerA3wzCTL+zSseGLdSXNJP14EwrjLNvDNoZeRSJ/  
n6xc3I4NYOptR4zR8d7VrjMAD6h5nR12NtxGIgJ/dVsUpIluWsHyCW3PPbKsM1JSruVRWoitRhSd3S6s1EWEPB0bDC7+  
+ISH5mXrCHOneyezQKlqNshTSPZ5jWwqRj2VNNSXCMTGXv9P01/U0bpOU12cuRTchgpPSe3ur2s0rRj3KlbmIyCz7P  
        }  
    }  
]
```

```
+H3CYlugafi8fQ5kipDSkESkIWS605ejzibWKg/3izms1eVIm/
zLFdReihCYJ7G8fpHUSLX5JAk3ab68aUXGSFEZLONntgNIhQkcMo00/
mJlaIgkEdBUyqvZO1vtKUBH5YonIrZqSUhv8Coc+mh24v0g1YI+SPIxlr
+Ln154BG6AjrmascjHACVXoPDxPsXSJXC4c9HjoC3YSskCPV7uWi0f65/
n7JAT3cskcX2ISaLHwYzJPaMBSftxOgeRLm3BnisL32nT8uTj2gF/
PUrEjdyoqTX7EerQpcaekXm0gay5Kh8n4T2uPdM83f356vRpar/
DDp8pLFD0ddb6Yvz7zU2zGdAvTod3IScC1GpTqcjRxaMhlBVZy1TnI9Cs
+7fXMdUF6xYScjR2725icFBNLojSFVDmsfHabXaCEpmeyXzsLbp5CjcPAHa66R8mQ5tSoFjrzOEzeB4uconEXAMPLE=="
},
"responseElements": null,
"requestID": "1MOU1Q80P4LDPT7A7N1A758N2VVV4KQNSO5AEMVJF66Q9EXAMPLE",
"eventID": "09a634f2-da7d-4c9e-a259-54aceexample",
"readOnly": true,
"resources": [
{
"accountId": "111122223333",
"type": "AWS::DynamoDB::Table",
"ARN": "arn:aws:dynamodb:us-west-2:123456789012:table/Music"
}
],
"eventType": "AwsApiCall",
"apiVersion": "2012-08-10",
"managementEvent": false,
"recipientAccountId": "111122223333",
"eventCategory": "Data"
}
]
```

Registro em log e monitoramento no DynamoDB Accelerator

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e o desempenho do Amazon DynamoDB Accelerator (DAX) e do seu AWS Soluções. Você deve coletar dados de monitoramento de todas as partes do AWS Para que você possa depurar mais facilmente uma falha de vários pontos, caso ocorra uma falha.

Para obter mais informações sobre o registro e o monitoramento no DAX, consulte [Monitoramento DAX \(p. 788\)](#).

Analisando o acesso a dados usando o CloudWatch Contributor Insights para DynamoDB

O Amazon CloudWatch Contributor Insights para Amazon DynamoDB é uma ferramenta de diagnóstico para identificar rapidamente as chaves acessadas com mais frequência e mais limitadas na tabela ou no índice. Essa ferramenta usa o [CloudWatch Contributor Insights](#).

Ao habilitar o CloudWatch Contributor Insights para DynamoDB em uma tabela ou em um índice secundário global, é possível visualizar os itens mais acessados e limitados nesses recursos.

Note

São aplicadas cobranças do CloudWatch Insights para DynamoDB. Para obter mais informações sobre a definição de preço, consulte [Definição de preço do Amazon CloudWatch](#).

Tópicos

- [CloudWatch Contributor Insights para DynamoDB: Como ele funciona \(p. 971\)](#)
- [Noções básicas sobre o CloudWatch Contributor Insights para DynamoDB \(p. 975\)](#)
- [Uso do IAM com o CloudWatch Contributor Insights para DynamoDB \(p. 979\)](#)

CloudWatch Contributor Insights para DynamoDB: Como ele funciona

O Amazon DynamoDB integra-se ao [Informações sobre o Amazon CloudWatch](#) Para fornecer informações sobre os itens mais acessados e limitados em uma tabela ou em um índice secundário global. O DynamoDB fornece essas informações a você por meio do CloudWatch Contributor Insights [Regras do reporte](#) de gráficos de dados do relatório.

Para obter mais informações sobre o CloudWatch Contributor Insights, consulte [Usar o Contributor Insights para analisar dados de alta cardinalidade](#) no Guia do usuário do Amazon CloudWatch.

As seções a seguir descrevem os principais conceitos e o comportamento do CloudWatch Contributor Insights para DynamoDB.

Tópicos

- [Regras do CloudWatch Contributor Insights para DynamoDB \(p. 971\)](#)
- [Noções básicas sobre o CloudWatch Contributor Insights para gráficos do DynamoDB \(p. 972\)](#)
- [Interações com outros recursos do DynamoDB \(p. 973\)](#)
- [Faturamento do CloudWatch Contributor Insights para DynamoDB \(p. 974\)](#)

Regras do CloudWatch Contributor Insights para DynamoDB

Ao habilitar o CloudWatch Contributor Insights para o DynamoDB em uma tabela ou em um índice secundário global, o DynamoDB cria as seguintes [regras](#) em seu nome:

- Itens mais acessados (chave de partição)— identifica as chaves de partição dos itens mais acessados na tabela ou no índice secundário global.

Formato do CloudWatch:DynamoDBContributorInsights-PKC-[resource_name]-[creationtimestamp]
- A maioria das teclas limitadas (chave de partição)— identifica as chaves de partição dos itens mais limitados na tabela ou no índice secundário global.

Formato do CloudWatch:DynamoDBContributorInsights-PKT-[resource_name]-[creationtimestamp]

Se a tabela ou o índice secundário global tiver uma chave de classificação, o DynamoDB também criará as seguintes regras específicas para chaves de classificação:

- Chaves mais acessadas (chaves de partição e classificação)— identifica as chaves de partição e de classificação dos itens mais acessados na tabela ou no índice secundário global.

Formato do CloudWatch:DynamoDBContributorInsights-SKC-[resource_name]-[creationtimestamp]
- A maioria das chaves limitadas (chaves de partição e classificação)— identifica as chaves de partição e de classificação dos itens mais limitados na tabela ou no índice secundário global.

Formato do CloudWatch:DynamoDBContributorInsights-SKT-[resource_name]-[creationtimestamp]

Note

- Não é possível usar as APIs ou o console do CloudWatch para excluir ou modificar diretamente as regras criadas pelo CloudWatch Contributor Insights para DynamoDB. Desabilitar o CloudWatch Contributor Insights para DynamoDB em uma tabela ou em um índice secundário global exclui automaticamente as regras criadas para essa tabela ou para esse índice secundário global.
- Quando você usa o [GetInsightRuleReport](#) Operação com as regras do CloudWatch Contributor Insights criadas pelo DynamoDB, somente `MaxContributorValue` e `MaximumRetorna` estatísticas úteis. As outras estatísticas dessa lista não retornam valores significativos.

Você pode criar alarmes do CloudWatch usando as [regras](#) do CloudWatch Contributor Insights para DynamoDB. Isso permite que você seja notificado quando um item exceder ou atingir um limite específico para `ConsumedThroughputUnits` ou `ThrottleCount`. Para obter mais informações, consulte [Definir um alarme para os dados de métrica do Contributor Insights](#).

Noções básicas sobre o CloudWatch Contributor Insights para gráficos do DynamoDB

O CloudWatch Contributor Insights para DynamoDB exibe dois tipos de gráficos nos consoles do DynamoDB e do CloudWatch: Itens mais acessados e itens mais limitados.

Itens mais acessados

Use esse gráfico para identificar os itens mais acessados na tabela ou no índice secundário global. O gráfico exibe `ConsumedThroughputUnits` no eixo y e o tempo no eixo x. Cada uma das principais chaves N são exibidas em sua própria cor com uma legenda exibida abaixo do eixo x.

O DynamoDB mede a frequência de acesso a chaves usando `ConsumedThroughputUnits`, que mede o tráfego de leitura e de gravação combinados. `ConsumedThroughputUnits` é definido como o seguinte:

- Provisionado —(3 x unidades de capacidade de gravação consumidas) + unidades de capacidade de leitura consumidas
- Sob demanda —(3 x unidades de solicitação de gravação) + unidades de solicitação de leitura.

No console do DynamoDB, cada ponto de dados no gráfico representa o máximo `deConsumedThroughputUnits` durante um período de 1 minuto. Por exemplo, um valor do gráfico de 180.000 `ConsumedThroughputUnits` indica que o item foi acessado continuamente à taxa de transferência máxima por item de 1.000 unidades de solicitação de gravação ou 3.000 unidades de solicitação de leitura durante um intervalo de 60 segundos dentro desse período de 1 minuto (3.000 x 60 segundos). Em outras palavras, os valores no gráfico representam o minuto com tráfego mais alto dentro de cada período de 1 minuto. Você pode alterar a granularidade do tempo `doConsumedThroughputUnits` (por exemplo, para visualizar métricas de 5 minutos em vez de 1 minuto) no console do CloudWatch.

Se forem exibidas várias linhas estreitamente agrupadas sem pontos fora da curva óbvios, isso indicará que a carga de trabalho está relativamente equilibrada entre os itens na janela de tempo especificada. Se forem exibidos pontos isolados no gráfico em vez de linhas conectadas, isso indicará um item que foi frequentemente acessado somente por um breve período.

Se a tabela ou o índice secundário global tiver uma chave de classificação, o DynamoDB criará dois gráficos: um para as chaves de partição mais acessadas e outro para os pares de chave de classificação + partição mais acessados. É possível ver o tráfego no nível de chave de partição no gráfico apenas de chave de partição. Você pode ver o tráfego no nível do item nos gráficos de chave de partição + classificação.

Itens mais limitados

Use esse gráfico para identificar os itens mais limitados na tabela ou no índice secundário global. O gráfico exibe ThrottleCount no eixo y e o tempo no eixo x. Cada uma das principais chaves N são exibidas em sua própria cor com uma legenda exibida abaixo do eixo x.

O DynamoDB mede a frequência de limitação usando ThrottleCount, que é a contagem dos erros ProvisionedThroughputExceededException, ThrottlingException e RequestLimitExceeded.

A limitação de gravação causada pela capacidade de gravação insuficiente para um índice secundário global não é medida. Você pode usar oito chaves N mais acessadas do índice secundário global para identificar padrões de acesso desequilibrados que podem causar limitação de gravação. Para obter mais informações, consulte [Considerações sobre a taxa de transferência provisionada para índices secundários globais](#).

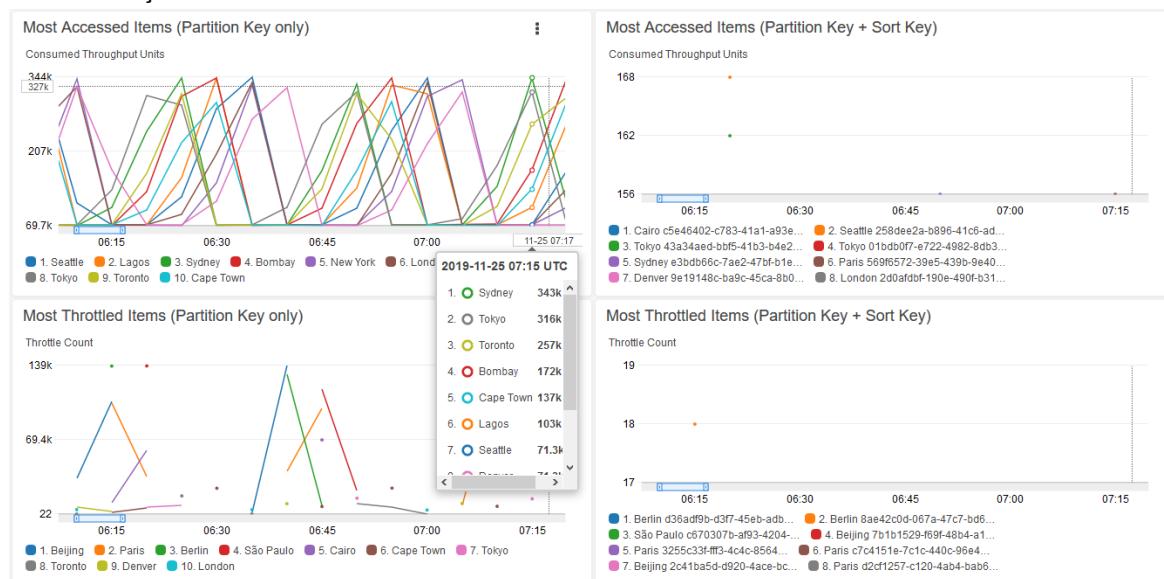
No console do DynamoDB, cada ponto de dados no gráfico representa a contagem de eventos de limitação em um período de 1 minuto.

Se não forem exibidos dados neste gráfico, isso indicará que as solicitações não estão sendo limitadas. Se você vir pontos isolados no gráfico em vez de linhas conectadas, isso indica que um item foi frequentemente limitado por um breve período.

Se a tabela ou o índice secundário global tiver uma chave de classificação, o DynamoDB criará dois gráficos: um para as chaves de partição mais limitadas e outro para os pares de chave de classificação + partição mais limitados. É possível ver a contagem de limitação no nível da chave de partição no gráfico somente da chave de partição, e a contagem de limitação no nível do item nos gráficos de chave de classificação + partição.

Exemplos de relatório

Veja a seguir exemplos dos relatórios gerados para uma tabela com uma chave de partição e uma chave de classificação



Interações com outros recursos do DynamoDB

As seções a seguir descrevem como o CloudWatch Contributor Insights para DynamoDB se comporta e interage com vários outros recursos no DynamoDB.

Tabelas globais

O CloudWatch Contributor Insights para DynamoDB monitora réplicas de tabelas globais como tabelas distintas. Os gráficos do Contributor Insights para uma réplica em umAWSA região pode não mostrar os mesmos padrões que outra região. Isso ocorre porque os dados de gravação são replicados em todas as réplicas em uma tabela global, mas cada réplica pode atender ao tráfego de leitura limitado à região.

DynamoDB Accelerator (DAX)

O CloudWatch Contributor Insights para DynamoDB não mostra respostas de cache do DAX. Ele mostra apenas respostas para acessar uma tabela ou um índice secundário global.

Criptografia em repouso

O CloudWatch Contributor Insights para DynamoDB não afeta a maneira como a criptografia funciona no DynamoDB. Os dados de chave primária publicados no CloudWatch são criptografados com oAWSChave mestra de cliente (CMK). No entanto, o DynamoDB também oferece suporte aoAWSCMK gerenciado e uma CMK gerenciada pelo cliente.

Se você precisar que seus dados de chave primária sejam criptografados com oAWSCMK gerenciada pelo cliente ou uma CMK gerenciada pelo cliente, você não deverá habilitar o CloudWatch Contributor Insights para DynamoDB para essa tabela.

Controle de acesso minucioso

O CloudWatch Contributor Insights para DynamoDB não funciona de forma diferente para tabelas com controle de acesso refinado (FGAC). Em outras palavras, qualquer usuário com as permissões apropriadas do CloudWatch pode visualizar chaves primárias protegidas por FGAC em gráficos do CloudWatch Contributor Insights.

Se a chave primária da tabela contiver dados protegidos por FGAC que você não deseja que sejam publicados no CloudWatch, não habilite o CloudWatch Contributor Insights para DynamoDB para essa tabela.

Controle de acesso

Você controla o acesso ao CloudWatch Contributor Insights para DynamoDB usando oAWS Identity and Access Management(IAM) limitando as permissões do plano de controle do DynamoDB e as permissões do plano de dados do CloudWatch. Para obter mais informações, consulte [Usar o IAM com o CloudWatch Contributor Insights para DynamoDB](#).

Faturamento do CloudWatch Contributor Insights para DynamoDB

As cobranças do CloudWatch Contributor Insights para DynamoDB são exibidas na[CloudWatch](#)da sua fatura mensal. Essas cobranças são calculadas com base no número de eventos do DynamoDB processados. Para tabelas e índices secundários globais com o CloudWatch Contributor Insights para DynamoDB ativado, cada item gravado ou lido por um[Plano de dados](#)operação representa um evento do.

Se uma tabela ou um índice secundário global incluir uma chave de classificação, cada item que for lido ou gravado representará dois eventos. Isso ocorre porque o DynamoDB está identificando os principais colaboradores de séries temporais separados: um para chaves de partição somente e outro para pares de chaves de partição e de classificação.

Por exemplo, suponha que o aplicativo execute as seguintes operações do DynamoDB: `umaGetItem`, `umPutItem`, e `umBatchWriteItem`que coloca cinco itens

- Se a tabela ou o índice secundário global tiver apenas uma chave de partição, isso resultará em 7 eventos (1 para `GetItem`, 1 para `PutItem` e 5 para `BatchWriteItem`).

- Se a tabela ou o índice secundário global tiver uma chave de partição e uma chave de classificação, isso resultará em 14 eventos (2 para `GetItem`, 2 para `PutItem` e 10 para `BatchWriteItem`).
- Uma operação `Query` sempre resulta em 1 evento, independentemente do número de itens retornados.

Ao contrário de outros recursos do DynamoDB, o CloudWatch Contributor Insights para DynamoDB não variam de acordo com o seguinte:

- O [modo de capacidade](#) (provisionado versus sob demanda)
- Se você executa solicitações de leitura ou gravação
- O tamanho (KB) dos itens lidos ou gravados

Noções básicas sobre o CloudWatch Contributor Insights para DynamoDB

Esta seção descreve como usar o Amazon CloudWatch Contributor Insights com o console do Amazon DynamoDB ou o AWS Command Line Interface(AWS CLI).

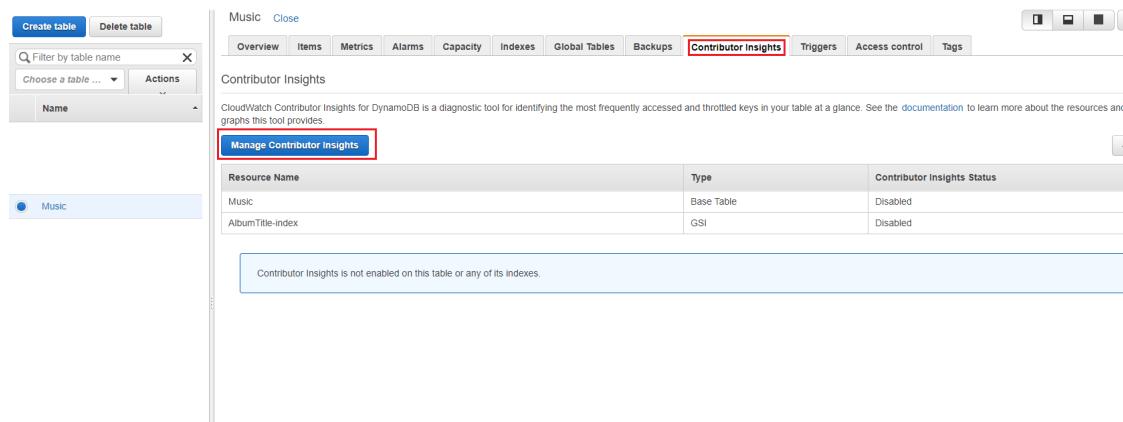
Nos exemplos a seguir, você usa a tabela do DynamoDB definida na[Conceitos básicos do DynamoDB](#)para "Hello, World!".

Tópicos

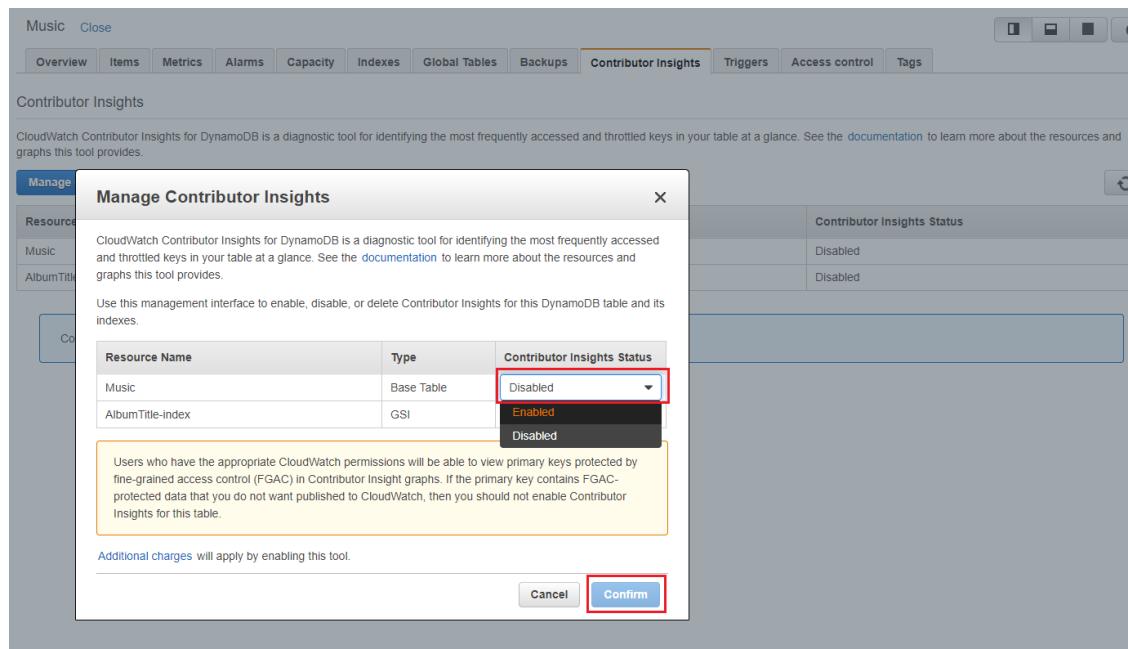
- [Usando o Colaborador Insights \(Console\) \(p. 975\)](#)
- [Usando o Contributor Insights \(AWS CLI\) \(p. 978\)](#)

Usando o Colaborador Insights (Console)

1. Faça login no AWS Management Console e abra o console do DynamoDB em<https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecione Tables (Tabelas).
3. Selecione o Music Tabela do.
4. Selecione o Contributor Insights Guia.
5. Selecione Gerenciar Contributor Insights.

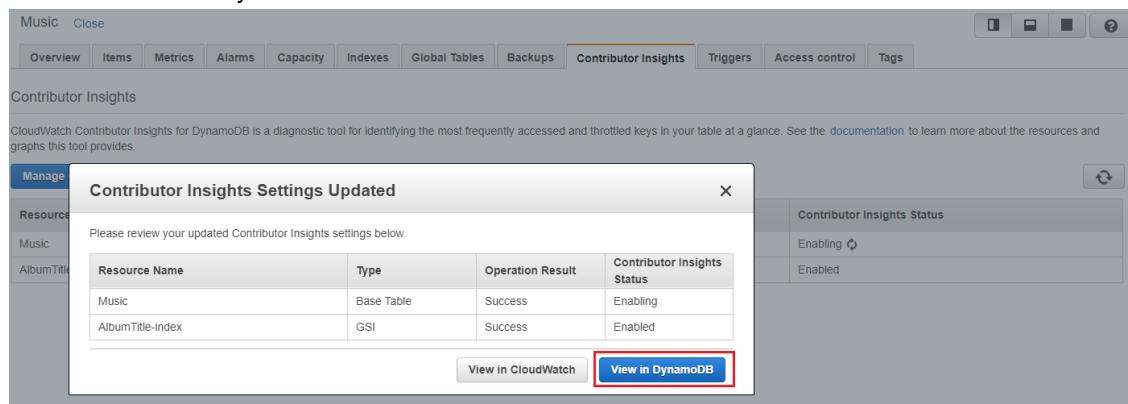


6. No Gerenciar Contributor Insights, em Status do Contributor Insights, escolha Enabled (Habilitado) para ambos os Music e a tabela base AlbumTitle-index índice secundário global. Depois, selecione Confirmar (Confirmar).

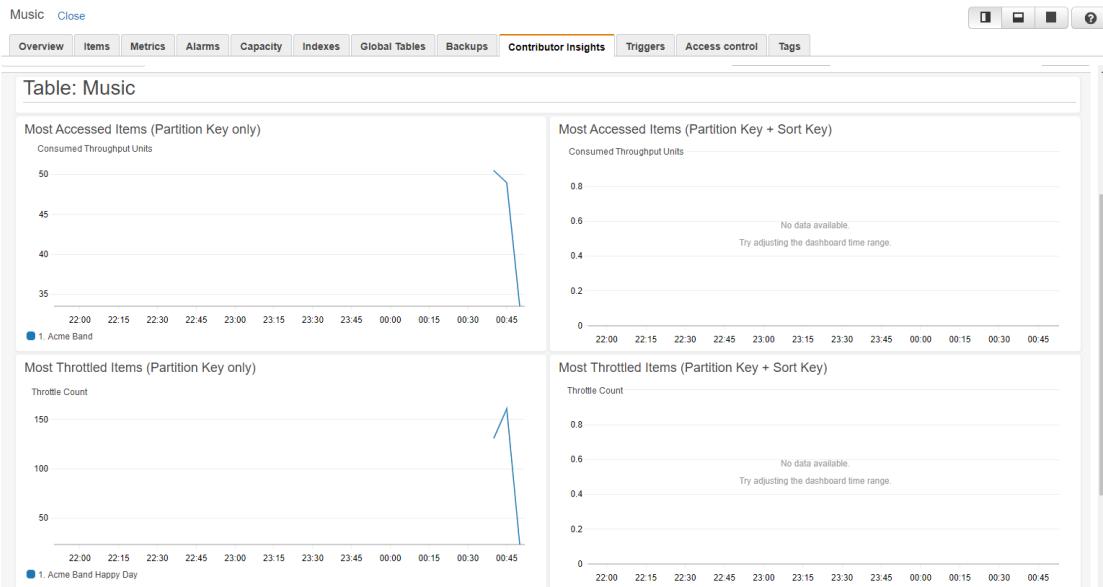


Se a operação falhar, consulte [DescribeContributorInsights FailureException](#) no [Referência de API do Amazon DynamoDB](#) por possíveis razões.

7. Selecione Exibir no DynamoDB.



8. Os gráficos do Colaborador Insights agora estão visíveis na página [Contributor Insights Guia do Music Tabela](#) do.



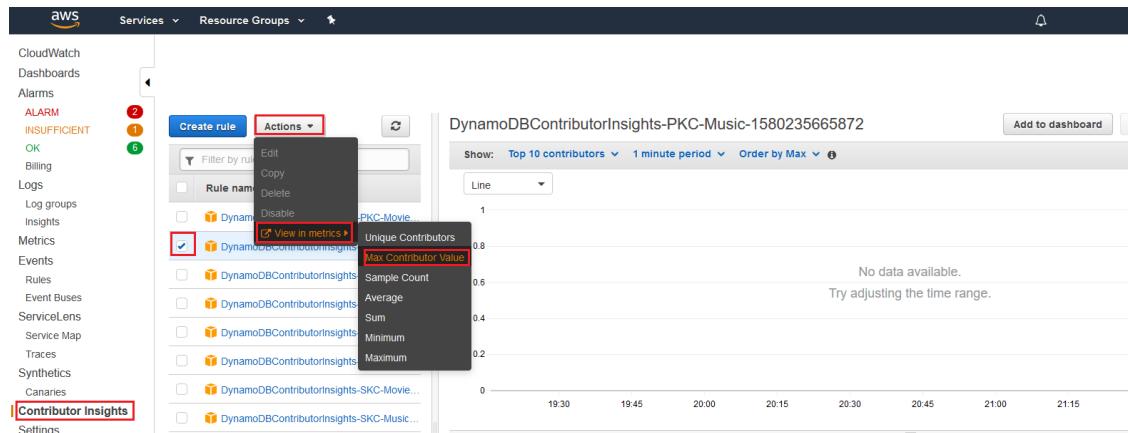
Criar alertas do CloudWatch

Siga estas etapas para criar um alarme do CloudWatch e ser notificado quando uma chave de partição consumir mais de 50.000 ConsumedThroughputUnits.

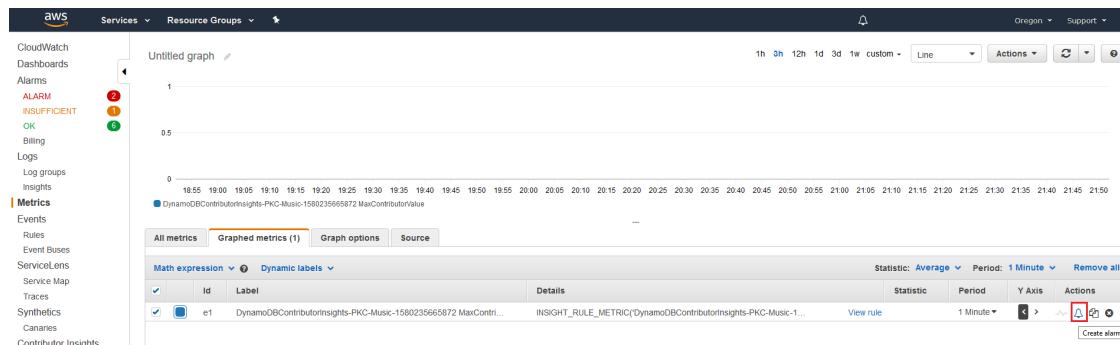
1. Faça login no AWS Management Console e abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>
2. No painel de navegação à esquerda do console, escolha Contributor Insights.
3. Selecione o DynamoDBContributorInsights-PKC-MusicRegra do.
4. Selecione o Ações escair.
5. Escolha View in metrics (Exibir nas métricas).
6. Escolha Max Contributor Value (Valor máximo do colaborador).

Note

SomenteMax Contributor Value e Maximum Retorna estatísticas úteis. As outras estatísticas dessa lista não retornam valores significativos.



7. Na coluna Actions (Ações), escolha Create Alarm (Criar alarme).



8. Insira um valor de 50000 para threshold e escolha Próximo.

Step 3: Add name and description

Step 4: Preview and create

Graph
This alarm will trigger when the blue line goes above the red line for 1 datapoints within 1 minute.

Label:

Math expression: INSIGHT_RULE_METRIC(DynamoDBContributorInsights-PKC-Music-1...)

Period: 1 minute

Conditions

Threshold type: Static Use a value as a threshold Anomaly detection Use a band as a threshold

Whenever DynamoDBContributorInsights-PKC-Music-1587490256272 MaxContributorValue is... Define the alarm condition.

Greater > threshold Greater/Equal >= threshold Lower/Equal <= threshold Lower < threshold

than... Define the threshold value: Must be a number

Next

9. Consulte [Usando alarmes do Amazon CloudWatch](#) Para obter detalhes sobre como configurar a notificação do alarme.

Usando o Contributor Insights (AWS CLI)

1. Ative o CloudWatch Contributor Insights para DynamoDB na Music mesa de base.

```
aws dynamodb update-contributor-insights --table-name Music --contributor-insights-action=ENABLE
```

2. Ative o Contributor Insights para DynamoDB no AlbumTitle índice secundário global.

```
aws dynamodb update-contributor-insights --table-name Music --index-name AlbumTitle-index --contributor-insights-action=ENABLE
```

3. Obtenha o status e as regras para o Music todos os seus índices.

```
aws dynamodb describe-contributor-insights --table-name Music
```

4. Desative o CloudWatch Contributor Insights para DynamoDB na `AlbumTitle-index` índice secundário global.

```
aws dynamodb update-contributor-insights --table-name Music --index-name AlbumTitle-index --contributor-insights-action=DISABLE
```

5. Obtenha o status do `Music` e todos os seus índices.

```
aws dynamodb list-contributor-insights --table-name Music
```

Uso do IAM com o CloudWatch Contributor Insights para DynamoDB

A primeira vez que você habilitar o Amazon CloudWatch Contributor Insights para Amazon DynamoDB, o DynamoDB criará automaticamente um AWS Identity and Access Management Função vinculada ao serviço (IAM) para você. Esta função do, AWS Service Role For DynamoDB CloudWatch Contributor Insights O DynamoDB gerencie as regras do CloudWatch Contributor Insights em seu nome. Não exclua essa função vinculada ao serviço. Se você excluí-la, todas as regras gerenciadas não serão mais removidas quando excluir a tabela ou o índice secundário global.

Para obter mais informações sobre funções vinculadas a serviços, consulte [Usar funções vinculadas a serviços](#) no Guia do usuário do IAM.

As seguintes permissões são necessárias:

- Para ativar ou desativar o CloudWatch Contributor Insights para DynamoDB, é necessário ter `dynamodb:UpdateContributorInsights` na tabela ou índice.
- Para visualizar os gráficos do CloudWatch Contributor Insights para DynamoDB, é necessário ter a permissão `cloudwatch:GetInsightRuleReport`.
- Para descrever o CloudWatch Contributor Insights para DynamoDB para determinada tabela ou índice do DynamoDB, é necessário ter `dynamodb:DescribeContributorInsightsPermissions`.
- Para listar os status do CloudWatch Contributor Insights para DynamoDB para cada tabela e cada índice secundário global, é necessário ter a permissão `dynamodb>ListContributorInsights`.

Exemplo: Ativar ou desativar o CloudWatch Contributor Insights para DynamoDB

A política do IAM a seguir concede permissões para ativar ou desativar o CloudWatch Contributor Insights para DynamoDB.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "iam>CreateServiceLinkedRole",  
            "Resource": "arn:aws:iam::*:role/aws-service-role/  
contributionorinsights.dynamodb.amazonaws.com/  
AWS Service Role For DynamoDB CloudWatch Contributor Insights",  
            "Condition": {"StringLike": {"iam:AWSServiceName":  
"contributionorinsights.dynamodb.amazonaws.com"}}  
    ]  
}
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:UpdateContributorInsights"
            ],
            "Resource": "arn:aws:dynamodb:*.*:table/*"
        }
    ]
}
```

Exemplo: Recuperar o CloudWatch Contributor Insights

A política do IAM a seguir concede permissões para recuperar o relatório de regras do CloudWatch Contributor Insights.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetInsightRuleReport"
            ],
            "Resource": "arn:aws:cloudwatch:.*:insight-rule/DynamoDBContributorInsights*"
        }
    ]
}
```

Exemplo: Aplicar seletivamente o CloudWatch Contributor Insights para permissões do DynamoDB com base no recurso

A política do IAM a seguir concede permissões para o `ListContributorInsights` e `DescribeContributorInsights`ações e nega a `UpdateContributorInsights`Ação para um índice secundário global específico.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb>ListContributorInsights",
                "dynamodb:DescribeContributorInsights"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "dynamodb:UpdateContributorInsights"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books/index/Author-
index"
        }
    ]
}
```

Uso de funções vinculadas ao serviço do CloudWatch Contributor Insights para DynamoDB

O CloudWatch Contributor Insights para DynamoDB usa o AWS Identity and Access Management(IAM)[Funções vinculadas ao serviço](#). Uma função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao CloudWatch Contributor Insights para DynamoDB. As funções vinculadas a serviços são predefinidas pelo CloudWatch Contributor Insights para DynamoDB e incluem todas as permissões exigidas pelo serviço para chamar outros AWS Serviços em seu nome.

Uma função vinculada ao serviço facilita a configuração do CloudWatch Contributor Insights para DynamoDB porque você não precisa adicionar as permissões necessárias manualmente. O CloudWatch Contributor Insights para DynamoDB define as permissões das funções vinculadas a serviços e, salvo outra definição, somente o CloudWatch Contributor Insights para DynamoDB pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

Para obter informações sobre outros produtos que oferecem suporte às funções vinculadas a serviços, consulte [Produtos da AWS compatíveis com o IAM](#) e procure os produtos que apresentam Yes (Sim) na coluna Service-Linked Role (Função vinculada a produtos). Escolha um Sim com um link para exibir a documentação da função vinculada a serviço desse serviço.

Permissões de função vinculada ao serviço do CloudWatch Contributor Insights para DynamoDB

O CloudWatch Contributor Insights para DynamoDB usa a função vinculada ao serviço chamada AWSServiceRoleForDynamoDBCloudWatchContributorInsights. O objetivo da função vinculada ao serviço é permitir que o Amazon DynamoDB gerencie as regras do Amazon CloudWatch Collaborator Insights criadas para tabelas do DynamoDB e índices secundários globais, em seu nome.

A função vinculada ao serviço AWSServiceRoleForDynamoDBCloudWatchContributorInsights confia nos seguintes serviços para assumir a função:

- `contributorinsights.dynamodb.amazonaws.com`

A política de permissões da função permite que o CloudWatch Contributor Insights para DynamoDB conclua as seguintes ações nos recursos especificados:

- Ação: `Create and manage Insight Rules` em `DynamoDBContributorInsights`

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Para obter mais informações, consulte [Permissões de função vinculada ao serviço](#) no Guia do usuário do IAM.

Criação de uma função vinculada ao serviço do CloudWatch Contributor Insights para DynamoDB

Você não precisa criar manualmente uma função vinculada a serviço. Quando você habilita o Colaborador Insights no AWS Management Console, o AWS CLI, ou o AWS CloudWatch Contributor Insights para DynamoDB cria uma função vinculada ao serviço para você.

Se você excluir essa função vinculada ao serviço e precisar criá-la novamente, poderá usar esse mesmo processo para recriar a função em sua conta. Quando você habilita o Contributor Insights, o CloudWatch Contributor Insights para DynamoDB cria novamente a função vinculada ao serviço.

Edição de uma função vinculada ao serviço do CloudWatch Contributor Insights para DynamoDB

O CloudWatch Contributor Insights para DynamoDB não permite editar a função vinculada ao serviço. Depois que criar uma função vinculada ao serviço, você não poderá alterar o nome da função, pois várias entidades podem fazer referência a ela. No entanto, você poderá editar a descrição da função usando o IAM. Para obter mais informações, consulte [Editar uma função vinculada ao serviço](#) no Guia do usuário do IAM.

Exclusão de uma função vinculada ao serviço do CloudWatch Contributor Insights para DynamoDB

Você não precisa excluir manualmente a função vinculada ao serviço. Quando você desabilita o Colaborador Insights no AWS Management Console, o AWS CLI, ou o AWS API do CloudWatch Contributor Insights para DynamoDB limpa os recursos.

Você também pode usar o console do IAM, o AWS CLI ou o AWS API para excluir manualmente a função vinculada ao serviço. Para isso, primeiro você deve limpar manualmente os recursos de sua função vinculada a serviço e depois excluí-la manualmente.

Note

Se o serviço CloudWatch Contributor Insights para DynamoDB estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Como excluir manualmente a função vinculada ao serviço usando o IAM

Use o console do IAM, o AWS CLI, ou o AWS API para excluir a função vinculada ao serviço. Para obter mais informações, consulte [Excluir uma função vinculada ao serviço](#) no Guia do usuário do IAM.

Práticas recomendadas para projetar e arquitetar com o DynamoDB

Use esta seção para localizar rapidamente as recomendações para maximizar o desempenho e minimizar os custos de throughput ao trabalhar com o Amazon DynamoDB.

Sumário

- [Design do NoSQL para o DynamoDB \(p. 984\)](#)
 - Diferenças entre o design relacional de dados e o NoSQL (p. 984)
 - Dois conceitos principais para o design de NoSQL (p. 985)
 - Abordagem de design do NoSQL (p. 985)
- [Práticas recomendadas para projetar chaves de partição e usá-las efetivamente \(p. 986\)](#)
 - Uso efetivo da capacidade de intermitência (p. 986)
 - Noções básicas da capacidade adaptável do DynamoDB (p. 987)
 - Aumentar a capacidade da taxa de transferência para partições de alto tráfego (p. 987)
 - Isolar itens acessados com frequência (p. 988)
 - Design de chaves de partição para distribuir uniformemente a carga de trabalho (p. 988)
 - Uso da fragmentação de gravação para distribuir uniformemente as cargas de trabalho (p. 989)
 - Fragmentação usando sufixos aleatórios (p. 989)
 - Fragmentação usando sufixos calculados (p. 989)
 - Distribuir a atividade de gravação com eficiência durante o upload de dados (p. 990)
- [Práticas recomendadas para usar Chaves de classificação para organizar dados \(p. 991\)](#)
 - Usar chaves de classificação para controle de versões (p. 992)
- [Práticas recomendadas para usar índices secundários no DynamoDB \(p. 993\)](#)
 - Diretrizes gerais para índices secundários no DynamoDB (p. 993)
 - Usar índices de modo eficiente (p. 994)
 - Escolher projeções com cuidado (p. 994)
 - Otimizar consultas frequentes para evitar buscas (p. 994)
 - Esteja ciente dos limites de tamanho da coleção de itens ao criar índices secundários locais (p. 995)
 - Aproveitar índices esparsos (p. 995)
 - Exemplos de índices esparsos no DynamoDB (p. 996)
 - Uso de índices secundários globais para consultas de agregação materializadas (p. 996)
 - Sobrecarga de índices secundários globais (p. 997)
 - Usando Sharding de gravação de índice secundário global para consultas seletivas de tabela (p. 998)
 - Usando índices secundários globais para criar uma réplica eventualmente consistente (p. 999)
- [Práticas recomendadas para armazenar itens e atributos grandes \(p. 1000\)](#)
 - Compactação de valores de atributos grandes (p. 1000)

- Armazenamento de valores de atributos grandes no Amazon S3 (p. 1001)
- Práticas recomendadas para lidar com dados de séries temporais no. (p. 1001)
 - Padrão de design para dados de séries temporais (p. 1001)
 - Exemplos de tabelas de séries temporais (p. 1002)
- Práticas recomendadas para gerenciar relações muitos para muitos (p. 1002)
 - Padrão de design da lista de adjacências (p. 1002)
 - Padrão de gráficos materializados (p. 1003)
- Práticas recomendadas para implementação de um sistema híbrido de banco de dados (p. 1006)
 - Se você não quiser migrar tudo para o DynamoDB (p. 1006)
 - Como um sistema híbrido pode ser implementado (p. 1007)
- Práticas recomendadas para modelagem de dados relacionais no DynamoDB (p. 1008)
 - Primeiras etapas para modelagem de dados relacionais no DynamoDB (p. 1011)
 - Exemplo de modelagem de dados relacionais no DynamoDB (p. 1012)
- Melhores práticas para consulta e verificação de dados (p. 1014)
 - Considerações sobre desempenho das verificações (p. 1014)
 - Evitar picos súbitos em atividade de leitura (p. 1014)
 - Aproveitar verificações paralelas (p. 1016)
 - Escolha TotalSegments (p. 1017)

Design do NoSQL para o DynamoDB

Os sistemas de bancos de dados NoSQL, como o Amazon DynamoDB, usam modelos alternativos para o gerenciamento de dados, como pares de chave-valor ou armazenamento de documentos. Ao mudar de um sistema de gerenciamento de banco de dados relacional para um sistema de banco de dados NoSQL como o DynamoDB, é importante compreender as principais diferenças e as abordagens específicas de design.

Tópicos

- Diferenças entre o design relacional de dados e o NoSQL (p. 984)
- Dois conceitos principais para o design de NoSQL (p. 985)
- Abordagem de design do NoSQL (p. 985)

Diferenças entre o design relacional de dados e o NoSQL

Os sistemas de bancos de dados relacionais (RDBMS) e os bancos de dados NoSQL têm diferentes pontos fortes e fracos:

- No RDBMS, as consultas de dados são flexíveis, mas têm um custo relativamente alto e não escalam com facilidade em situações de grande volume de tráfego (consulte [Primeiras etapas para modelagem de dados relacionais no DynamoDB \(p. 1011\)](#)).
- Em um banco de dados NoSQL, como o DynamoDB, os dados podem ser consultados com eficiência em um número limitado de maneiras limitadas, além das quais as consultas podem apresentar alto custo e baixo desempenho.

Essas diferenças tornam o design de banco de dados diferente entre os dois sistemas:

- Em RDBMS, você cria o design para obter flexibilidade sem se preocupar com detalhes de implementação ou desempenho. A otimização de consultas geralmente não afeta o design do esquema, mas a normalização é importante.
- No DynamoDB, você projeta o esquema especificamente para fazer as consultas mais comuns e importantes do modo mais rápido e barato possível. Suas estruturas de dados são adaptadas aos requisitos específicos de seus casos de uso de negócios.

Dois conceitos principais para o design de NoSQL

O design do NoSQL exige uma visão diferente daquela no design do RDBMS. Para um RDBMS, você pode criar um modelo de dados normalizado sem pensar nos padrões de acesso. Você poderá estendê-lo posteriormente quando surgirem novas perguntas e requisitos de consulta. Você pode organizar cada tipo de dados em sua própria tabela.

Como o design de NoSQL é diferente

- Por outro lado, você não deve iniciar o design do seu esquema para o DynamoDB até que saiba quais perguntas ele precisará responder. É essencial compreender os problemas de negócios e os casos de uso de aplicativo antecipadamente.
- Você deve manter o mínimo de tabelas possível em um aplicativo do DynamoDB.

Abordagem de design do NoSQL

A primeira etapa ao projetar seu aplicativo do DynamoDB é identificar os padrões específicos de consulta aos quais o sistema deve atender.

Especificamente, é importante compreender três propriedades fundamentais de padrões de acesso de seu aplicativo antes de começar:

- Tamanho dos dados: Saber o volume de dados que serão armazenados e solicitados ao mesmo tempo ajudará a determinar a maneira mais eficiente de particionar os dados.
- Forma dos dados: Em vez de remodelar dados quando uma consulta é processada (como um sistema RDBMS faz), um banco de dados NoSQL organiza os dados para que sua forma no banco de dados corresponda ao que será consultado. Esse é um fator importante no aumento da velocidade e da escalabilidade.
- Velocidade dos dados: O DynamoDB é dimensionado aumentando o número de partíciones físicas que estão disponíveis para processar consultas e distribuindo os dados com eficiência entre essas partíciones. Saber antecipadamente qual é o pico das cargas de consulta pode ajudar a determinar como particionar os dados para melhor utilização da capacidade de E/S.

Após identificar os requisitos específicos da consulta, você pode organizar dados de acordo com os princípios gerais que regem o desempenho:

- Mantenha os dados relacionados juntos. Uma pesquisa sobre a otimização de tabelas de rotas há 20 anos descobriu que a "localidade de referência" era o único fator o mais importante para agilizar o tempo de resposta: manter dados relacionados juntos em um só lugar. Isso também se aplica aos sistemas NoSQL hoje, em que manter dados relacionados juntos tem um impacto significativo no custo e no desempenho. Em vez da distribuição de itens de dados relacionados entre várias tabelas, você deve manter itens relacionados no sistema de NoSQL o mais próximo possível.

Como regra geral, você deve manter o mínimo de tabelas possível em um aplicativo do DynamoDB.

As exceções são os casos que envolvem dados de séries temporais de alto volume ou conjuntos de dados que têm padrões muito diferentes de acesso. Uma única tabela com índices invertidos pode

normalmente habilitar consultas simples para criar e recuperar estruturas de dados hierárquicas e complexas, exigidas pelo aplicativo.

- Use a ordem de classificação. Os itens relacionados podem ser agrupados juntos e consultados de modo eficiente se o design da chave fizer com que sejam classificados juntos. Essa é uma estratégia importante de design do NoSQL.
- Distribua as consultas. É importante também que um alto volume de consultas não se concentre em uma parte do banco de dados, onde podem exceder a capacidade de E/S. Em vez disso, você deve projetar chaves de dados para distribuir o tráfego entre as partições do modo mais uniforme possível, evitando “pontos de atividade”.
- Use índices secundários globais. Ao criar índices secundários globais específicos, você pode permitir consultas diferentes daquelas aceitas na tabela principal, que sejam rápidas e relativamente econômicas.

Esses princípios gerais traduzem-se em alguns padrões comuns de design que você pode usar para modelar dados no DynamoDB com eficiência.

Práticas recomendadas para projetar chaves de partição e usá-las efetivamente

A chave primária que identifica exclusivamente cada item em uma tabela do Amazon DynamoDB pode ser simples (apenas uma chave de partição) ou composta (uma chave de partição combinada com uma chave de classificação).

De modo geral, você deve projetar o aplicativo para atividade uniforme em todas as chaves de partição lógica na tabela e em seus índices secundários. É possível determinar os padrões de acesso que o aplicativo exige e estimar o total de unidades de capacidade de leitura (RCU) e unidades de capacidade de gravação (WCU) que cada tabela e índice secundário requerem.

O DynamoDB é compatível com seus padrões de acesso usando a taxa de transferência provisionada, contanto que o tráfego em uma determinada partição não exceda 3.000 RCU ou 1.000 WCUs.

Tópicos

- [Uso efetivo da capacidade de intermitência \(p. 986\)](#)
- [Noções básicas da capacidade adaptável do DynamoDB \(p. 987\)](#)
- [Design de chaves de partição para distribuir uniformemente a carga de trabalho \(p. 988\)](#)
- [Uso da fragmentação de gravação para distribuir uniformemente as cargas de trabalho \(p. 989\)](#)
- [Distribuir a atividade de gravação com eficiência durante o upload de dados \(p. 990\)](#)

Uso efetivo da capacidade de intermitência

O DynamoDB fornece alguma flexibilidade no provisionamento de throughput por partição, fornecendo Capacidade de intermitência. Sempre que você não está usando plenamente a taxa de transferência de uma partição, o DynamoDB reserva uma parte dessa capacidade não utilizada para posterior Intermittente de taxa de transferência para lidar com picos de uso.

No momento, o DynamoDB mantém até cinco minutos (300 segundos) de capacidade de leitura e gravação não utilizada. Durante uma intermitência ocasional de atividades de leitura e gravação, essas unidades de capacidade extra podem ser consumidas rapidamente, ainda mais rápido do que a capacidade de throughput provisionado por segundo que você definiu para a sua tabela.

O DynamoDB também pode consumir capacidade de intermitência para manutenção em segundo plano e para outras tarefas sem prévio aviso.

Note que esses detalhes da capacidade de intermitência podem mudar no futuro.

Noções básicas da capacidade adaptável do DynamoDB

Capacidade adaptável é um recurso que permite que o DynamoDB execute cargas de trabalho desequilibradas indefinidamente. Ela minimiza a limitação devido a exceções da taxa de transferência. Ela também ajuda a reduzir os custos permitindo que você provisione somente a capacidade da taxa de transferência necessária.

A capacidade adaptável é habilitada automaticamente para todas as tabelas do DynamoDB, sem custo adicional. Não é necessário habilitá-la ou desabilitá-la explicitamente.

Tópicos

- [Aumentar a capacidade da taxa de transferência para partições de alto tráfego \(p. 987\)](#)
- [Isolar itens acessados com frequência \(p. 988\)](#)

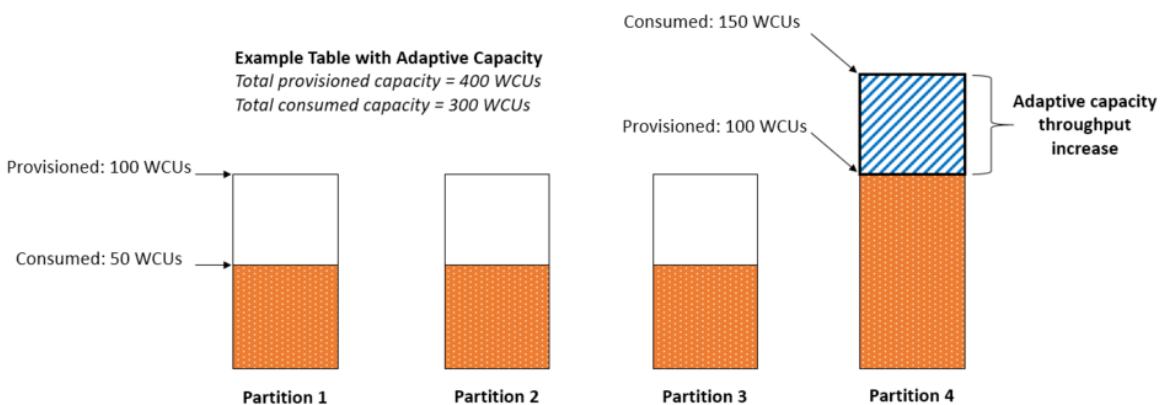
Aumentar a capacidade da taxa de transferência para partições de alto tráfego

Nem sempre é possível distribuir as atividades de leitura e gravação uniformemente. Quando o acesso aos dados é desequilibrado, uma partição "dinâmica" pode receber um volume mais alto de tráfego de leitura e gravação em comparação com outras partições. Em casos extremos, poderá ocorrer limitação, se uma única partição receber mais de 3000 RCUs ou 1000 WCUs.

Para acomodar melhor padrões desiguais de acesso, a capacidade adaptável do DynamoDB permite que o aplicativo continue a ler e gravar em partições dinâmicas sem ser limitado, desde que o tráfego não exceda a capacidade total provisionada da tabela ou a capacidade máxima da partição. A capacidade adaptável funciona por meio do aumento automático e instantâneo da capacidade de taxa de transferência para as partições que recebem mais tráfego.

O diagrama a seguir mostra como a capacidade adaptável funciona. A tabela de exemplo é provisionada com 400 WCUs compartilhadas uniformemente em quatro partições, permitindo que cada partição comporte até 100 WCUs por segundo. As partições 1, 2 e 3 recebem tráfego de gravação de 50 WCU/seg. Partição 4 recebe 150 WCU/seg. Essa partição dinâmica pode aceitar tráfego de gravação enquanto tiver capacidade não utilizada, mas em algum momento limitará o tráfego que excede 100 WCU/s.

A capacidade adaptável do DynamoDB responde por meio do aumento da capacidade da partição quatro para que ela comporte uma carga de trabalho mais alta de 150 WCU/s sem limitação.



Isolar itens acessados com frequência

Se o aplicativo direcionar tráfego desproporcionalmente alto para um ou mais itens, a capacidade adaptável reequilibrará as partições de modo que os itens acessados com frequência não residam na mesma partição. Esse isolamento dos itens acessados com frequência reduz a probabilidade da limitação de solicitação devido à carga de trabalho exceder a cota de taxa de transferência em uma única partição.

Se o aplicativo direciona alto tráfego constantemente a um único item, a capacidade adaptável pode reequilibrar os dados de modo que uma partição contenha somente esse único item acessado com frequência. Nesse caso, o DynamoDB pode fornecer throughput até o máximo de 3.000 RCUs da partição ou 1.000 WCUs para a chave primária desse único item.

Note

Esta funcionalidade de isolamento não está disponível para

- Tabelas que têm um [índice secundário local](#).
- Tabelas usando o [modo provisionado de capacidade de leitura/gravação](#) que habilitaram [Streams do DynamoDB](#).

Design de chaves de partição para distribuir uniformemente a carga de trabalho

A parte de chave de partição da chave primária de uma tabela determina as partições lógicas em que os dados de uma tabela são armazenados. Isso afeta, por sua vez, as partições físicas subjacentes. A capacidade provisionada de E/S para a tabela é dividida uniformemente entre essas partições físicas. Portanto, um design de chave de partição que não distribui as solicitações de E/S uniformemente pode criar partições “dinâmicas” que resultam em limitação e usam a capacidade provisionada de E/S de forma ineficiente.

O uso ideal da taxa de transferência provisionada de uma tabela depende não apenas dos padrões da carga de trabalho de itens individuais, mas também do design da chave de partícões. Isso não significa que você deve acessar todos os valores de chaves de partição para atingir um nível eficiente de throughput ou que a porcentagem de valores de chaves de partição acessados deve ser alta. Isso significa que quanto mais diferentes forem os valores de chave de partição que sua carga de trabalho acessa, mais essas solicitações serão espalhadas entre o espaço particionado. Em geral, você utilizará o throughput provisionado de modo mais eficiente, à medida que a proporção aumentar entre valores de chaves de partição acessados e o número total de valores de chaves de partição.

Veja a seguir uma comparação da eficiência da taxa de transferência provisionada de alguns esquemas comuns de chave de partição.

Valor da chave de partição	Uniformidade
ID de usuário, no qual o aplicativo tem muitos usuários.	Bom
Código de status, no qual existem apenas alguns códigos de status possíveis.	Ruim
Data de criação do item, arredondada para o período mais próximo (para, por exemplo, dia, hora ou minuto).	Ruim
ID do dispositivo, no qual cada dispositivo acessa dados em intervalos relativamente semelhantes.	Bom

Valor da chave de partição	Uniformidade
ID do dispositivo, no qual, mesmo que haja muitos dispositivos rastreados, um deles é muito mais popular do que todos os outros.	Ruim

Se uma única tabela tiver apenas um número pequeno de valores de chaves de partição, considere distribuir suas operações de gravação entre valores de chaves de partição mais distintos. Em outras palavras, estruture os elementos de chave primária para evitar um valor chave de partição "quente" (intensamente solicitada) que diminua o desempenho geral.

Por exemplo, considere uma tabela com uma chave primária composta. A chave de partição representa a data de criação do item, arredondada para o próximo dia. A chave de classificação é um identificador de item. Em um determinado dia, digamos 2014-07-09, todos os novos itens são gravados naquele valor de chave de partição único (e na partição física correspondente).

Se a tabela se encaixar inteiramente em uma única partição (considerando o crescimento dos seus dados ao longo do tempo) e se os requisitos de throughput de leitura e gravação do seu aplicativo não excederem os recursos de leitura e gravação de uma única partição, seu aplicativo não deverá se deparar com limitações inesperadas resultantes do particionamento.

No entanto, se você prevê a escalabilidade da tabela para além de uma única partição, deve arquitetar seu aplicativo para que ele possa usar mais do throughput provisionado completo da tabela.

Uso da fragmentação de gravação para distribuir uniformemente as cargas de trabalho

Uma forma de distribuir melhor as gravações entre o espaço de chaves de uma partição no Amazon DynamoDB é expandir o espaço. É possível fazer isso de várias formas diferentes. Você pode adicionar um número aleatório aos valores de chaves de partição para distribuir os itens entre partícões. Ou pode usar um número calculado com base em algo que você está consultando.

Fragmentação usando sufixos aleatórios

Uma estratégia para a distribuição mais uniforme de cargas em um espaço de chave de partição é adicionar um número aleatório ao final dos valores de chave de partição. Então, você randomiza as gravações no espaço maior.

Por exemplo, para uma chave de partição que represente a data de hoje, você pode escolher um número aleatório entre 1 e 200 e concatená-lo como um sufixo à data. Isso produz valores de chave de partição como 2014-07-09.1, 2014-07-09.2 e assim por diante, até 2014-07-09.200. Como você está randomizando a chave de partição, as gravações na tabela em cada dia são espalhadas uniformemente entre várias partícões. Isso resulta em melhor paralelismo e throughput geral mais alto.

Contudo, para ler todos os itens de um determinado dia, você teria que consultar os itens quanto a todos os sufixos e mesclar os resultados. Por exemplo, você primeiro emite uma solicitação de `Query` do valor da chave de partição 2014-07-09.1. Depois emite outra `Query` de 2014-07-09.2, e assim por diante, por meio de 2014-07-09.200. Por fim, seu aplicativo precisaria mesclar os resultados de todas essas solicitações `Query`.

Fragmentação usando sufixos calculados

Uma estratégia de randomização pode melhorar bastante o throughput de gravação. Mas é difícil ler um item específico, porque você não sabe que valor de sufixo foi usado na gravação do item. Para facilitar a

leitura de itens individuais, você pode usar uma estratégia diferente. Em vez de usar um número aleatório para distribuir os itens entre as partições, use um número calculado com base em algo que você deseja consultar.

Considere o exemplo anterior, em que uma tabela usa a data de hoje na chave de partição. Agora considere que cada item conta com um atributo `OrderId` acessível e que você precisa com frequência de localizar os itens por ID de pedidos, além da data. Antes que o aplicativo grave o item na tabela, ele pode calcular um sufixo hash com base no ID do pedido e anexá-lo à data da chave de partição. O cálculo pode gerar um número entre 1 e 200 que é distribuído uniformemente, semelhante à estratégia aleatória.

Bastaria um cálculo simples, como o produto dos valores de pontos de código UTF-8 para os caracteres no ID do pedido, módulo 200, + 1. O valor da chave de partição seria então a data concatenada ao resultado do cálculo.

Com essa estratégia, as gravações são distribuídas uniformemente entre os valores de chaves de partição e, portanto, entre as partições físicas. É possível executar uma operação `GetItem` facilmente para um determinado item e data, pois você pode calcular o valor de chave da partição de um valor `OrderId` específico.

Para ler todos os itens de um determinado dia, você ainda deve `query` uma data (por exemplo, 2014-07-09) e seu aplicativo terá que mesclar todos os resultados. O benefício é evitar de ter um valor de chave de partição "hot" único consumindo toda a carga de trabalho.

Note

Para obter uma estratégia mais eficiente, projetada especificamente para lidar com um alto volume de dados de séries temporais, consulte [Dados de séries temporais \(p. 1001\)](#).

Distribuir a atividade de gravação com eficiência durante o upload de dados

Normalmente, quando você carrega dados de outras fontes de dados, o Amazon DynamoDB partitiona seus dados de tabela em vários servidores. Você terá um melhor desempenho se o upload de dados for feito em todos os servidores alocados simultaneamente.

Por exemplo, suponha que você queira fazer upload das mensagens dos usuários em uma tabela do DynamoDB que use uma chave primária composta com `UserID` como a chave de partição e `MessageID` como a chave de classificação.

Quando você carrega os dados, uma abordagem que você pode tomar é carregar todos os itens de mensagem para cada usuário, um usuário após o outro:

UserID	MessageID
U1	1
U1	2
U1	...
U1	... até 100
U2	1
U2	2
U2	...

UserID	MessageID
U2	... até 200

Nesse caso, o problema é que as solicitações de gravação não estão sendo distribuídas para o DynamoDB entre seus valores de chaves de partição. Você está usando um valor de chave de partição de cada vez e fazendo upload de todos os seus itens antes de passar para o próximo valor de chave de partição e fazer o mesmo.

Nos bastidores, o DynamoDB está particionando os dados na sua tabela em vários servidores. Para usar plenamente toda a capacidade de throughput que é provisionada para a tabela, você deve distribuir sua carga de trabalho entre seus valores de chaves de partição. Ao direcionar uma quantidade irregular de carga de trabalho de upload para os itens, todos com o mesmo valor de chave de partição, você não está usando plenamente todos os recursos que o DynamoDB provisionou para a sua tabela.

Você pode distribuir seu trabalho de upload usando a chave de classificação para carregar um item de cada valor de chave de partição, depois outro item de cada valor de chave de partição e assim por diante:

UserID	MessageID
U1	1
U2	1
U3	1
...	...
U1	2
U2	2
U3	2
...	...

Cada upload nesta sequência usa um valor de chave de partição diferente, mantendo mais servidores do DynamoDB ocupados ao mesmo tempo e melhorando seu desempenho de throughput.

Práticas recomendadas para usar Chaves de classificação para organizar dados

Em uma tabela do Amazon DynamoDB, a chave primária que identifica exclusivamente cada item na tabela pode ser composta não apenas de uma chave de partição, mas também de uma chave de classificação.

As chaves de classificação bem-projetadas tem dois principais benefícios:

- Elas reúnem informações relacionadas em um só lugar para consulta eficiente. O design cuidadoso da chave de classificação permite recuperar grupos de itens relacionados comumente necessários, usando consultas de intervalo com operadores como `begins_with`, `between`, `>`, `<` e assim por diante.
- Chaves de classificação compostas permitem definir os relacionamentos hierárquicos (um para muitos) entre os dados que podem ser consultados em qualquer nível da hierarquia.

Por exemplo, em uma tabela que lista as localizações geográficas, você pode estruturar a chave de classificação desta forma.

[country]#[region]#[state]#[county]#[city]#[neighborhood]

Isso permite fazer consultas de intervalos eficientes de uma lista de localizações em qualquer um desses níveis de agregação, de `country` a `neighborhood` e tudo o que estiver nesse intervalo.

Usar chaves de classificação para controle de versões

Muitos aplicativos precisam manter um histórico de revisões no nível do item para fins de auditoria ou conformidade e para conseguir recuperar a versão mais recente com facilidade. Há um padrão de design eficiente que faz isso por meio do uso de prefixos de chave de classificação:

- Para cada novo item, crie duas cópias do item: Uma cópia deve ter um prefixo de número de versão igual a zero (como `v0_`) no início da chave de classificação, e a outra deve ter um prefixo de número de versão um (como `v1_`).
- Sempre que o item for atualizado, use o próximo prefixo de versão mais alto na chave de classificação da versão atualizada e copie o conteúdo atualizado para o item com o prefixo de versão zero. Isso significa que a versão mais recente de qualquer item pode ser localizada facilmente usando o prefixo zero.

Por exemplo, o fabricante de uma peça pode usar um esquema como esse ilustrado abaixo.

Primary Key		Data-Item Attributes...								
Partition Key	Sort Key (varies)	Attribute 1		Attribute 2		Attribute 3		Attribute 4		...
Equipment_1	Details	Name:	Biphasic Cardiometer (equipment name)	Factory_ID:	S14_Tukwilla (factory where manufactured)	Line_ID	R_7 (assembly-line ID)			
	v0_Audit	Auditor:	Padma (name of the auditor)	Latest:	3 (most recent audit version)	Time:	2018-04-15T11:00 (audit date and time)	Result	Passed (audit result)	...etc.
	v1_Audit	Auditor:	Rick (name of the auditor)	Time:	2018-03-14T11:00 (audit date and time)	Result	Open (audit result)	Report:	0943922EKG14 (detailed problem report in S3)	...etc.
	v2_Audit	Auditor:	George (name of the auditor)	Time:	2018-03-18T11:00 (audit date and time)	Result	Open (audit result)	Report:	0943923EKG15 (detailed problem report in S3)	...etc.
	v3_Audit	Auditor:	Pádmā (name of the auditor)	Time:	2018-04-15T11:00 (audit date and time)	Result	Passed (audit result)	Report:	x792 (pass confirmation report)	...etc.

O item `Equipment_1` passa por uma sequência de auditorias realizadas por vários auditores. Os resultados de cada nova auditoria são capturados em um novo item na tabela, começando com o número de versão um e depois incrementando o número para cada revisão sucessiva.

Quando cada nova revisão é adicionada, a camada do aplicativo substitui o conteúdo do item de versão zero (com chave de classificação igual a `v0_Audit`) com conteúdo da nova revisão.

Sempre que o aplicativo precisar recuperar para o status de auditoria mais recente, ele poderá consultar o prefixo da chave de classificação `v0_`.

Se o aplicativo precisar recuperar o histórico de revisão inteiro, poderá consultar todos os itens na chave de partição do item e filtrar o item `v0_`.

Esse design também funcionará para auditorias em várias peças de uma parte do equipamento, se você incluir IDs de peças individuais na chave de classificação após o prefixo de chave de classificação

Práticas recomendadas para usar índices secundários no DynamoDB

Os índices secundários são frequentemente essenciais para suporte aos padrões de consulta que o aplicativo exige. Ao mesmo tempo, o uso em excesso os índices secundários ou o uso ineficiente deles pode adicionar custo e reduzir o desempenho desnecessariamente.

Sumário

- [Diretrizes gerais para índices secundários no DynamoDB \(p. 993\)](#)
 - Usar índices de modo eficiente (p. 994)
 - Escolher projeções com cuidado (p. 994)
 - Otimizar consultas frequentes para evitar buscas (p. 994)
 - Esteja ciente dos limites de tamanho da coleção de itens ao criar índices secundários locais (p. 995)
- [Aproveitar índices esparsos \(p. 995\)](#)
 - Exemplos de índices esparsos no DynamoDB (p. 996)
- [Uso de índices secundários globais para consultas de agregação materializadas \(p. 996\)](#)
- [Sobrecarga de índices secundários globais \(p. 997\)](#)
- [Usando Sharding de gravação de índice secundário global para consultas seletivas de tabela \(p. 998\)](#)
- [Usando índices secundários globais para criar uma réplica eventualmente consistente \(p. 999\)](#)

Diretrizes gerais para índices secundários no DynamoDB

O Amazon DynamoDB oferece suporte a dois tipos de índices secundários:

- Índice secundário global: um índice com uma chave de partição e uma chave de classificação que podem ser diferentes das contidas na tabela base. Um índice secundário global é considerado “global” porque as consultas no índice podem abranger todos os dados em uma tabela base, além de todas as partições. Um índice secundário global não tem nenhuma limite de tamanho e tem suas próprias configurações de throughput provisionado para a atividade de leitura e gravação que são separadas dessas configurações da tabela.
- Índice secundário local: um índice que possui a mesma chave de partição da tabela base, mas uma chave de classificação diferente. Um índice secundário local é “local” no sentido de que cada partição de um índice secundário local tem a determinação de escopo para uma partição de tabela base com o mesmo valor de chave de partição. Como resultado, o tamanho total de itens indexados para qualquer valor de chave de partição não pode exceder 10 GB. Além disso, um índice secundário local compartilha configurações de throughput provisionado para atividade de leitura e gravação com a tabela que ele está indexando.

Cada tabela no DynamoDB pode ter até 20 índices secundários globais (cota padrão) e 5 índices secundários locais.

Para mais informações sobre as diferenças entre índices secundários globais e índices secundários locais, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

Em geral, você deve usar índices secundários globais em vez de índices secundários locais. A exceção é quando você precisa de consistência forte nos resultados de consulta, o que um índice secundário local

pode fornecer, mas um índice secundário global não pode (as consultas de índice secundário global são compatíveis apenas com consistência eventual).

Veja a seguir alguns princípios gerais e padrões de design para serem lembrados na criação de índices no DynamoDB:

Tópicos

- [Usar índices de modo eficiente \(p. 994\)](#)
- [Escolher projeções com cuidado \(p. 994\)](#)
- [Otimizar consultas frequentes para evitar buscas \(p. 994\)](#)
- [Esteja ciente dos limites de tamanho da coleção de itens ao criar índices secundários locais \(p. 995\)](#)

Usar índices de modo eficiente

Mantenha o número de índices no mínimo. Não crie índices secundários em atributos que você não consulta frequentemente. Os índices usados raramente contribuem para aumentar os custos de armazenamento e de E/S, sem melhorar o desempenho do aplicativo.

Escolher projeções com cuidado

Como os índices secundários consomem armazenamento e throughput provisionado, você deve manter o índice no menor tamanho possível. Além disso, quanto menor o índice, maior será a vantagem de desempenho em comparação com a consulta de toda a tabela. Se as suas consultas geralmente retornarem apenas um pequeno subconjunto de atributos, e o tamanho total dos atributos for muito menor que o item inteiro, projete apenas os atributos que você solicita regularmente.

Se você estima muitas atividades de gravação em uma tabela, em comparação com as atividades de leitura, siga estas práticas recomendadas:

- Considere projetar menos atributos para reduzir o tamanho dos itens gravados no índice. No entanto, isso se aplicará somente se o tamanho dos atributos projetados for maior do que uma única unidade de capacidade de gravação (1 KB). Por exemplo, se o tamanho de uma entrada de índice for apenas 200 bytes, o DynamoDB o arredondará até 1 KB. Em outras palavras, contanto que os itens de índice sejam pequenos, você pode projetar mais atributos sem nenhum custo extra.
- Evite projetar os atributos que serão raramente necessários nas consultas. Sempre que você atualizar um atributo projetado em um índice, haverá um custo extra também para a atualização do índice. Você pode ainda recuperar atributos não projetados em uma `Query` a um custo maior de throughput provisionado, mas o custo da consulta pode ser significativamente menor do que o custo da atualização frequente do índice.
- Especifique `ALL` somente se você deseja que suas consultas retornem itens da tabela inteira classificados por uma chave de classificação diferente. Proteger todos os atributos elimina a necessidade de buscas da tabela, mas, na maioria dos casos, isso dobra os custos das atividades de armazenamento e gravação.

Equilibre a necessidade de manter seus índices o menor possível e a necessidade de manter o mínimo de buscas, como explicado a próxima seção.

Otimizar consultas frequentes para evitar buscas

Para obter consultas mais rápidas com a menor latência possível, projete todos os atributos que você espera que essas consultas retornem. Especificamente, se você consultar um índice secundário local quanto a atributos que não estão planejados, o DynamoDB buscará esses atributos automaticamente

na tabela, o que requer a leitura do item inteiro na tabela. Isso pode gerar operações adicionais de E/S e latência que você pode evitar.

Lembre-se de que as consultas “ocasionais” podem se transformar em consultas “principais”. Se houver mais atributos que não se pretende projetar, porque você estima que as consultas serão apenas ocasionais, considere se as condições podem ser alteradas e se você pode lamentar não ter projetado os atributos.

Para obter mais informações sobre buscas de tabela, consulte [Considerações sobre taxa de transferência provisionada para índices secundários locais \(p. 610\)](#).

Esteja ciente dos limites de tamanho da coleção de itens ao criar índices secundários locais

Uma coleção de itens comprehende todos os itens em uma tabela e seus índices secundários locais que têm a mesma chave de partição. Nenhuma coleção de itens pode exceder 10 GB, por isso, é possível ficar sem espaço para um determinado valor de chave de partição.

Quando você adiciona ou atualiza um item de tabela, o DynamoDB atualiza todos os índices secundários locais que são afetados. Se os atributos indexados forem definidos na tabela, os índices secundários locais também crescerão.

Ao criar um índice secundário local, pense no volume de dados que serão gravados nele e quantos desses itens de dados terão o mesmo valor de chave de partição. Se você espera que a soma dos itens da tabela e do índice de um determinado valor de chave de partição exceda 10 GB, considere se é possível evitar a criação do índice.

Se você não puder evitar a criação do índice secundário local, deverá prever o limite de tamanho da coleção de itens e executar ações antes que ele seja excedido. Para estratégias sobre como trabalhar dentro do limite e tomar uma ação corretiva, consulte [Limite de tamanho da coleção de itens \(p. 614\)](#).

Aproveitar índices esparsos

Para qualquer item em uma tabela, o DynamoDB grava uma entrada de índice correspondente somente se o valor da chave de classificação do índice estiver presente no item. Se a chave de classificação não aparecer em cada item da tabela, o índice é considerado esparsos.

Os índices esparsos são úteis para consultas em uma subseção pequena de uma tabela. Por exemplo, suponha que você tem uma tabela que armazene todos os pedidos de clientes, com os seguintes atributos chaves:

- Chave de partição: `CustomerId`
- Chave de classificação: `OrderId`

Para monitorar os pedidos em aberto, você pode inserir um atributo chamado `isOpen`. Os itens de pedidos que ainda não foram enviados. Então, quando o pedido for enviado, você poderá excluir o atributo. Se você criar um índice em `CustomerId` (chave de partição) e em `isOpen` (chave de cada), somente os pedidos definidos com `isOpen` serão exibidos. Quando você tem milhares de pedidos e apenas um pequeno número deles está em aberto, é mais rápido e barato consultar o índice de pedidos em aberto do que analisar a tabela inteira.

Em vez de usar um tipo de atributo como `isOpen`, é possível usar um atributo com um valor que resulta em uma ordem de classificação útil no índice. Por exemplo, você pode usar um atributo `OrderOpenDate` definido como a data em que cada pedido foi feito e excluí-lo após o atendimento do pedido. Dessa forma, quando você consulta o índice esparsos, os itens são retornados classificados de acordo com a data na qual cada pedido foi feito.

Exemplos de índices esparsos no DynamoDB

Os índices secundários globais são esparsos por padrão. Ao criar um índice secundário global, você especifica uma chave de partição e, opcionalmente, uma chave de classificação. Somente os itens na tabela base que contenham esses atributos aparecem no índice.

Ao projetar um índice secundário global como esparsos, você pode provisioná-lo com uma taxa de transferência de gravação menor do que a da tabela base, ainda obtendo um excelente desempenho.

Por exemplo, um aplicativo de jogos pode acompanhar todas as pontuações de cada usuário, mas geralmente só precisa consultar algumas pontuações altas. O seguinte design lida com esse cenário de modo eficiente:

Table	Primary Key		Data Attributes...			
	Partition Key	Sort Key				
	Player_ID	Game_ID	Attribute 1	Attribute 2	Attribute 3	
Rick	Rick	Game_1	Score: 36,750 <i>(game score)</i>	Date: 2017-11-14 <i>(date of game)</i>		
		Game_2	Score: 69,450 <i>(game score)</i>	Date: 2017-12-31 <i>(date of game)</i>		
		Game_3	Score: 135,900 <i>(game score)</i>	Date: 2018-01-19 <i>(date of game)</i>	Award: Champ <i>(type of award)</i>	
Padma	Padma	Game_4	Score: 25,350 <i>(game score)</i>	Date: 2018-01-27 <i>(date of game)</i>		
		Game_5	Score: 69,450 <i>(game score)</i>	Date: 2018-01-19 <i>(date of game)</i>		
		Game_6	Score: 147,300 <i>(game score)</i>	Date: 2018-02-02 <i>(date of game)</i>	Award: Champ <i>(type of award)</i>	
		Game_7	Score: 169,100 <i>(game score)</i>	Date: 2018-03-10 <i>(date of game)</i>	Award: Champ <i>(type of award)</i>	

Aqui, Rick jogou três jogos e obteve o status Champ em um deles. Padma jogou quatro jogos e obteve o status Champ em dois deles. Observe que o atributo Award está presente apenas nos itens em que o usuário obteve um award. O índice secundário global associado é semelhante ao seguinte:

GSI	Primary Key		Projected Attributes...			
	Partition Key					
		Award	Player_ID	Game_ID	Score	Date
Champ	Champ	Rick	Game_3	135,900	2018-01-19	
		Padma	Game_6	147,300	2018-02-02	
		Padma	Game_7	169,100	2018-03-10	

O índice secundário global contém apenas as pontuações altas que são consultadas com frequência e representadas por um pequeno subconjunto de itens na tabela base.

Uso de índices secundários globais para consultas de agregação materializadas

Manter as agregações quase em tempo real e as métricas de chaves relacionadas aos dados em constante mudança é uma operação cada vez mais valiosa para a rápida tomada de decisões na empresa.

Por exemplo, uma biblioteca de músicas pode querer apresentar suas músicas que mais foram obtidas por download quase em tempo real.

Considere o seguinte layout de tabela da biblioteca de músicas:

Music Library Table						
Primary Key		Data-Item Attributes...				
Partition Key	Sort Key	Attribute 1		Attribute 2		Attribute 3
Song-129 <i>(song ID)</i>	Details	Title:	Wild Love <i>(song title)</i>	Artist:	Argyboots <i>(artist or band name)</i>	Downloads: 15,314,822 <i>(lifetime total downloads)</i>
		GSI Primary Key		GSI Secondary Key		
	Month-2018-01	Month:	2018-01 <i>(download month)</i>	MonthTotal:	1,746,992 <i>(month total downloads)</i>	
		Time:	2018-01-01T00:00:07 <i>(download timestamp)</i>			
	Dld-9349823681	Time:	2018-01-01T00:00:07 <i>(download timestamp)</i>			
	Dld-9349823682	Time:	2018-01-01T00:00:07 <i>(download timestamp)</i>			
	Dld-9349823683	Time:	2018-01-01T00:00:07 <i>(download timestamp)</i>			

A tabela neste exemplo armazena músicas com `songID` como a chave de partição. Você pode habilitar fluxos do Amazon DynamoDB nessa tabela e conectar uma função do Lambda aos fluxos, de modo que, à medida que cada música for transferida, uma entrada seja adicionada à tabela com `Partition-Key=SongID&Sort-Key=DownloadID`. Conforme essas atualizações são feitas, elas ativam uma função do Lambda nos DynamoDB Streams. A função do Lambda pode agregar e agrupar os downloads por `songID` e atualizar o item de nível superior, `Partition-Key=songID` e `Sort-Key=Month`. Tenha em mente que se uma execução do Lambda falhar logo após escrever o novo valor agregado, ela pode ser repetida e agregar o valor mais de uma vez, deixando você com um valor aproximado.

Para ler as atualizações quase em tempo real, com latência de milissegundo de um único dígito, use o índice secundário global com condições de consulta `Month=2018-01`, `ScanIndexForward=False`, `Limit=1`.

Outra otimização de chave usada aqui é que o índice secundário global é um índice esparsão e está disponível somente nos itens que precisam ser consultados para recuperar os dados em tempo real. O índice secundário global pode atender afluxos de trabalho adicionais que precisam de informações sobre as 10 primeiras músicas que eram populares, ou qualquer música transferida naquele mês.

Sobrecarga de índices secundários globais

Embora o Amazon DynamoDB tenha uma cota padrão de 20 índices secundários globais por tabela, na prática, é possível indexar muito mais do que 20 campos de dados. Ao contrário de uma tabela em um sistema de banco de dados relacional (RDBMS), em que o esquema é uniforme, uma tabela no DynamoDB pode armazenar muitos tipos diferentes de itens de dados ao mesmo tempo. Além disso, o mesmo atributo em diferentes itens pode conter tipos de informações totalmente diferentes.

Considere o seguinte exemplo de um layout de tabela do DynamoDB que salva uma variedade de tipos diferentes de dados.

Primary Key		Data-Item Attributes...				
Partition Key	Sort Key	Attribute 1		Attribute 2		...
HR-974 <i>(employee ID)</i>	Employee_Name	Data:	Murphy, John <i>(employee name)</i>	Start:	2008-11-08 <i>(start date)</i>	...etc.
	YYYY-Q1	Data:	\$5,477 <i>(order totals in USD)</i>	Name:	Murphy, John <i>(employee name)</i>	
	HR_confidential	Data:	2008-11-08 <i>(hire date)</i>	Name:	Murphy, John <i>(employee name)</i>	...etc.
	Warehouse_01	Data:	Murphy, John <i>(employee name)</i>			
	v0_Job_title	Data:	Operator-1 <i>(job title)</i>	Start:	2008-11-08 <i>(start date)</i>	...etc.
	v1_Job_title	Data:	Operator-2 <i>(job title)</i>	Start:	2016-11-04 <i>(start date)</i>	...etc.
	v2_Job_title	Data:	Supervisor-1 <i>(job title)</i>	Start:	2017-11-01 <i>(start date)</i>	...etc.

O atributo Data, que é comum para todos os itens, tem conteúdo diferente dependendo do item pai. Se você criar um índice secundário global para a tabela que usa a chave de classificação da tabela como sua chave de partição e o atributo Data como sua chave de classificação, poderá fazer uma variedade de consultas diferentes usando esse único índice secundário global. Essas consultas podem incluir o seguinte:

- Pesquise um funcionário pelo nome no índice secundário global, usando Employee_Name como o valor da chave de partição e o nome do funcionário (por exemplo Murphy, John) como o valor da chave de classificação.
- Use o índice secundário global para encontrar todos os funcionários que trabalham em um determinado armazém procurando um ID de armazém (como Warehouse_01).
- Obtenha uma lista de contratações recentes, consultando o índice secundário global no HR_confidential como um valor de chave de partição e usando um intervalo de datas no valor de chave de classificação.

Usando Sharding de gravação de índice secundário global para consultas seletivas de tabela

Os aplicativos freqüentemente precisam identificar um pequeno subconjunto de itens em uma tabela do Amazon DynamoDB que atendam a uma determinada condição. Quando esses itens são distribuídos aleatoriamente pelas chaves de partição da tabela, você pode recorrer a uma varredura de tabela para recuperá-los. Esta opção pode ser cara, mas funciona bem quando um grande número de itens na tabela atende à condição de pesquisa. No entanto, quando o espaço de chave é grande e a condição de pesquisa é muito seletiva, essa estratégia pode causar um monte de processamento desnecessário.

Para habilitar consultas seletivas em todo o espaço de chave, você pode usar o fragmento de gravação adicionando um atributo contendo um (0-N) para cada item que você usará para a chave de partição de índice secundário global.

Veja a seguir um exemplo de um esquema que o usa em um fluxo de trabalho de eventos críticos:

Table	Primary Key	Data Attributes...						
	Partition Key							
Event_ID	Attribute 1		Attribute 2		Attribute 3		Attribute 4	...
EID_12345	Time: <i>(event timestamp)</i>	2018-02-07T08:42:40	State: <i>(event state)</i>	INFO	GSI PK: <i>(random GSI-PK value)</i>	(random: 0-N)	GSI SK: <i>(composite state-time)</i>	INFO#2018-02-07T08:42:40 <i>(composite state-time)</i> ...etc.
EID_12346	Time: <i>(event timestamp)</i>	2018-02-07T08:32:40	State: <i>(event state)</i>	CRITICAL	GSI PK: <i>(random GSI-PK value)</i>	(random: 0-N)	GSI SK: <i>(composite state-time)</i>	CRITICAL#2018-02-07T08:32:40 <i>(composite state-time)</i> ...etc.
EID_12347	Time: <i>(event timestamp)</i>	2018-02-07T08:22:40	State: <i>(event state)</i>	WARN	GSI PK: <i>(random GSI-PK value)</i>	(random: 0-N)	GSI SK: <i>(composite state-time)</i>	WARN#2018-02-07T08:22:40 <i>(composite state-time)</i> ...etc.
EID_12348	Time: <i>(event timestamp)</i>	2018-02-07T08:12:40	State: <i>(event state)</i>	INFO	GSI PK: <i>(random GSI-PK value)</i>	(random: 0-N)	GSI SK: <i>(composite state-time)</i>	INFO#2018-02-07T08:12:40 <i>(composite state-time)</i> ...etc.

GSI	Primary Key		Data Attributes...		
	Partition Key	Sort Key			
GSI PK	GSI SK	...			
[0-N]	INFO#2018-02-07T08:42:40 <i>(composite state-time)</i>etc.		
[0-N]	CRITICAL#2018-02-07T08:32:40 <i>(composite state-time)</i>etc.		
[0-N]	WARN#2018-02-07T08:22:40 <i>(composite state-time)</i>etc.		
[0-N]	INFO#2018-02-07T08:12:40 <i>(composite state-time)</i>etc.		

Usando este design de esquema, os itens de evento são distribuídos entre o -N no GSI, permitindo uma leitura de dispersão usando uma condição de classificação na chave composta para recuperar todos os itens com um determinado estado durante um período de tempo especificado.

Esse padrão de esquema fornece um conjunto de resultados altamente seletivo a um custo mínimo, sem a necessidade de uma varredura de tabela.

Usando índices secundários globais para criar uma réplica eventualmente consistente

Você pode usar um índice secundário global para criar uma réplica eventualmente consistente de uma tabela. A criação de uma réplica pode permitir que você faça o seguinte:

- Defina diferentes capacidades de leitura provisionadas para diferentes leitores. Por exemplo, digamos que você tenha dois aplicativos: Um aplicativo lida com consultas de alta prioridade e precisa dos níveis mais altos de desempenho de leitura, enquanto o outro lida com consultas de baixa prioridade que podem tolerar a limitação da atividade de leitura.

Se ambos os aplicativos forem lidos da mesma tabela, uma carga de leitura pesada do aplicativo de baixa prioridade poderá consumir toda a capacidade de leitura disponível para a tabela. Isso aceleraria a atividade de leitura do aplicativo de alta prioridade.

Em vez disso, você pode criar uma réplica por meio de um índice secundário global cuja capacidade de leitura você pode definir separada da própria tabela. Em seguida, você pode fazer com que seu aplicativo de baixa prioridade consulte a réplica em vez da tabela.

- Elimine totalmente as leituras de uma tabela. Por exemplo, talvez você tenha um aplicativo que capture um alto volume de atividades de clickstream de um site, e não queira arriscar que as leituras interfiram com isso. Você pode isolar esta tabela e impedir leituras por outros aplicativos (consulte [Uso de condições de política do IAM para controle de acesso refinado \(p. 899\)](#)), enquanto permite que outros aplicativos leiam uma réplica criada usando um índice secundário global.

Para criar uma réplica, configure um índice secundário global que tenha o mesmo esquema de chave que o da tabela pai, com alguns ou todos os atributos não chave projetados nela. Em aplicativos, é possível direcionar uma parte da atividade de leitura ou toda ela para esse índice secundário global, em vez de para a tabela pai. Em seguida, você pode ajustar a capacidade de leitura provisionada do índice secundário global para lidar com essas leituras sem alterar a capacidade de leitura provisionada da tabela pai.

Há sempre um breve atraso de propagação entre uma gravação na tabela pai e o momento em que os dados gravados aparecem no índice. Em outras palavras, seus aplicativos devem levar em conta que a réplica de índice secundário global é apenaseventualmente consistente com a tabela pai.

Você pode criar várias réplicas de índice secundário global para oferecer suporte a diferentes padrões de leitura. Quando você cria as réplicas, projete somente os atributos que cada padrão de leitura realmente requer. Um aplicativo pode então consumir menos capacidade de leitura provisionada para obter apenas os dados necessários, em vez de ter que ler o item da tabela pai. Essa otimização pode resultar em uma economia significativa com o passar do tempo.

Práticas recomendadas para armazenar itens e atributos grandes

Atualmente, o Amazon DynamoDB limita o tamanho de cada item que você armazena em uma tabela (consulte [Cotas de serviço, conta e tabela no Amazon DynamoDB \(p. 1056\)](#)). Se o aplicativo precisar armazenar mais dados em um item do que o limite de tamanho permitido pelo DynamoDB, você poderá tentar compactar um ou mais atributos grandes ou dividir o item em vários itens (indexados adequadamente por chaves de classificação). Também é possível armazenar o item como um objeto no Amazon Simple Storage Service (Amazon S3) e armazenar o identificador de objeto do Amazon S3 no item do DynamoDB.

Compactação de valores de atributos grandes

A compactação de valores de atributos grandes pode permitir que eles se ajustem aos limites de item no DynamoDB e reduzam os custos de armazenamento. Os algoritmos de compactação como GZIP ou LZO geram resultado binário que você pode armazenar em um tipo de atributo `Binary`.

Por exemplo, a tabela `Reply` na seção [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#) armazena mensagens escritas por usuários do fórum. Essas respostas de usuários podem consistir em strings longas de texto, o que as torna excelentes candidatas para compactação.

Para obter o código de exemplo que demonstra como compactar essas mensagens no DynamoDB, consulte o seguinte:

- Exemplo: Tratamento de atributos do tipo binário usando o AWS SDK for JavaDocumentar API (p. 463)
- Exemplo: Tratamento de atributos do tipo binário usando o AWS SDK for .NETAPI de baixo nível (p. 487)

Armazenamento de valores de atributos grandes no Amazon S3

Como mencionado anteriormente, você também pode usar o Amazon S3 para armazenar valores de atributos grandes que não podem caber em um item do DynamoDB. Você pode armazená-los como um objeto no Amazon S3 e armazenar o identificador de objeto no item do DynamoDB.

Também é possível usar o suporte de metadados de objeto no Amazon S3 para fornecer um link para o item pai no DynamoDB. Armazene o valor da chave primária do item como metadados do Amazon S3 do objeto no Amazon S3. Fazer isso normalmente ajuda com a manutenção de objetos do Amazon S3.

Por exemplo, considere a tabela `ProductCatalog` na seção [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#). Os itens nessa tabela armazenam informações sobre preço do item, descrição, autores de livros e dimensões para outros produtos. Se você quisesse armazenar uma imagem de cada produto que era muito grande para se ajustar em um item, poderia armazenar as imagens no Amazon S3 em vez de no DynamoDB.

Ao implementar essa estratégia, lembre-se sempre do seguinte:

- O DynamoDB não é compatível com transações que cruzam o Amazon S3 e o DynamoDB. Portanto, seu aplicativo deve lidar com quaisquer falhas, que podem incluir a limpeza de objetos do Amazon S3 órfãos.
- O Amazon S3 limita o tamanho de identificadores de objeto. Então, você deve organizar seus dados de um modo que não permita a geração excessiva de identificadores de objeto nem viole outras restrições do Amazon S3.

Para obter mais informações sobre como usar o Amazon S3, consulte [Guia do desenvolvedor do Amazon Simple Storage Service](#).

Práticas recomendadas para lidar com dados de séries temporais no.

Os princípios gerais de design no Amazon DynamoDB recomendam que você mantenha o número mínimo de tabelas em uso. Para a maioria dos aplicativos, uma tabela única é tudo que você precisa. No entanto, no caso de dados de séries temporais, é possível lidar melhor com isso usando uma tabela por aplicativo por período.

Padrão de design para dados de séries temporais

Considere um cenário típico de séries temporais, em que você deseja monitorar um alto volume de eventos. O padrão de acesso de gravação é que todos os eventos sendo registrados tenham a data de hoje. O padrão de acesso de leitura pode ser: ler os eventos de hoje com mais frequência, os eventos de ontem com menos frequência e, então, os eventos mais antigos com muito menos frequência. Uma forma de lidar com isso é por meio da criação de data e hora atuais na chave primária.

O seguinte padrão de design normalmente lida com esse tipo de cenário de modo efetivo:

- Crie uma tabela por período, provisionada com a capacidade de leitura e gravação necessária e os índices necessários.
- Antes do final de cada período, crie previamente a tabela para o próximo período. Assim que o período atual terminar, direcione o tráfego de eventos para a nova tabela. É possível atribuir nomes a essas tabelas que especifiquem os períodos em que foram registradas.

- Quando uma tabela não estiver mais sendo gravada, reduza sua capacidade de gravação provisionada para um valor mais baixo (por exemplo, 1 WCU) e provisione a capacidade de leitura que for mais apropriada. Reduz a capacidade de leitura provisionada de tabelas anteriores conforme elas envelhecem. Você pode optar por arquivamento ou exclusão das tabelas cujo conteúdo raramente ou nunca é necessário.

A ideia é alocar os recursos necessários para o período atual que receberá o maior volume de tráfego e diminui o provisionamento para tabelas mais velhas que não são usadas ativamente, diminuindo assim os custos. Dependendo das necessidades de seus negócios, você pode considerar estilhaçar a gravação para distribuir o tráfego de forma uniforme para a chave de partição lógica. Para obter mais informações, consulte [Uso da fragmentação de gravação para distribuir uniformemente as cargas de trabalho \(p. 989\)](#).

Exemplos de tabelas de séries temporais

Veja a seguir um exemplo de dados de séries temporais no qual a tabela atual é provisionada com uma capacidade mais alta de leitura/gravação e as tabelas mais antigas são reduzidas porque não são acessadas com frequência:

Current table		Provisioned at: WCU=75 and RCU=200	
Primary Key	Attributes		
Partition key	Value	WCU	RCU
2010-01-15	00:00:00.002	17.59 WCU	793 RCU
2010-01-15	00:00:00.004	17.59 WCU	793 RCU
2010-01-15	00:00:00.006	17.59 WCU	793 RCU
2010-01-15	00:00:00.007	18.17 WCU	874 RCU
...

Previous table		Provisioned at: WCU=1 and RCU=100	
Primary Key	Attributes		
Partition key	Value	WCU	RCU
2010-01-14	00:00:00.001	14.47 WCU	512 RCU
2010-01-14	00:00:00.003	14.47 WCU	512 RCU
2010-01-14	00:00:00.006	14.47 WCU	512 RCU
2010-01-14	00:00:00.008	14.47 WCU	508 RCU
...

Older table		Provisioned at: WCU=1 and RCU=1	
Primary Key	Attributes		
Partition key	Value	WCU	RCU
2010-01-10	00:00:00.001	13.04 WCU	460 RCU
2010-01-10	00:00:00.003	13.04 WCU	460 RCU
2010-01-10	00:00:00.006	13.04 WCU	459 RCU
2010-01-10	00:00:00.008	13.04 WCU	457 RCU
2010-01-10	00:00:00.005	13.03 WCU	425 RCU
...

Práticas recomendadas para gerenciar relações muitos para muitos

As listas de adjacências são um padrão de design útil para modelar as relações muitos para muitos no Amazon DynamoDB. De modo mais geral, elas oferecem uma forma de representar os dados gráfico (nós e bordas) no DynamoDB.

Padrão de design da lista de adjacências

Quando entidades diferentes de um aplicativo têm uma relação muitos para muitos entre elas, a relação pode ser modelada como uma lista de adjacências. Nesse padrão, todas as entidades de nível superior (sinônimos para nós no modelo de gráfico) são representadas usando a chave de partição. Qualquer relação com outras entidades (bordas em um gráfico) é representada como um item na partição configurando o valor da chave de classificação como o ID da entidade de destino (nó de destino).

As vantagens desse padrão incluem a duplicação mínima de dados e padrões simplificados de consulta para encontrar todas as entidades (nós) relacionadas a uma entidade de destino (o ponto em um nó de destino).

Como exemplo real, esse padrão foi útil em um sistema de faturamento em que as faturas continham várias contas. Uma conta pode pertencer a várias faturas. A chave de partição nesse exemplo é um `InvoiceID` ou um `BillID`. As partícões `BillID` têm todos os atributos específicos para as contas. As partícões `InvoiceID` têm um item que armazena os atributos específicos à fatura e um item para cada `BillID` implementado para a fatura.

O esquema é semelhante ao seguinte.

Table	Primary Key		Data Attributes...	
	Partition Key	Sort Key (and GSI PK)		
Invoice-92551	Inv_ID:	Invoice-92551 <i>(invoice ID)</i>	Dated: 2018-02-07 <i>(date created)</i>	More attributes of this invoice...
	Bill_ID:	Bill-4224663 <i>(bill ID)</i>	Dated: 2017-12-03 <i>(date created)</i>	Attributes of this bill <i>in this invoice..</i>
	Bill_ID:	Bill-4224687 <i>(bill ID)</i>	Dated: 2018-01-09 <i>(date created)</i>	Attributes of this bill <i>in this invoice..</i>
Invoice-92552	Inv_ID:	Invoice-92552 <i>(invoice ID)</i>	Dated: 2018-03-04 <i>(date created)</i>	More attributes of this invoice...
	Bill_ID:	Bill-4224687 <i>(bill ID)</i>	Dated: 2018-01-09 <i>(date created)</i>	Attributes of this bill <i>in this invoice..</i>
Bill-4224663	Bill_ID:	Bill-4224663 <i>(bill ID)</i>	Dated: 2017-12-03 <i>(date created)</i>	More attributes of this bill...
Bill-4224687	Bill_ID:	Bill-4224687 <i>(bill ID)</i>	Dated: 2018-01-09 <i>(date created)</i>	More attributes of this bill...

Usando o esquema anterior, você pode ver que todas as contas para uma fatura podem ser consultadas usando a chave primária na tabela. Para pesquisar todas as faturas que contêm uma parte de uma conta, crie um índice secundário global na chave de classificação da tabela.

As projeções do índice secundário global são semelhantes ao seguinte.

GSI	Primary Key		Projected Attributes...	
	Partition Key			
Bill-4224663	Bill_ID:	Bill-4224663 <i>(table primary key)</i>	Attributes of this bill...	
	Inv_ID:	Invoice-92551 <i>(table primary key)</i>	Attributes of this bill <i>in this invoice..</i>	
Bill-4224687	Bill_ID:	Bill-4224687 <i>(table primary key)</i>	Attributes of this bill...	
	Inv_ID:	Invoice-92551 <i>(table primary key)</i>	Attributes of this bill <i>in this invoice..</i>	
Invoice-92551	Inv_ID:	Invoice-92552 <i>(table primary key)</i>	Attributes of this bill <i>in this invoice..</i>	
	Inv_ID:	Invoice-92551 <i>(table primary key)</i>	Attributes of this invoice...	
Invoice-92552	Inv_ID:	Invoice-92552 <i>(table primary key)</i>	Attributes of this invoice...	

Padrão de gráficos materializados

Muitos aplicativos são criados com base no entendimento das classificações entre os colegas, nas relações comuns entre entidades, no estado da entidade vizinha e em outros tipos de fluxos de trabalho de estilo de gráficos. Para esses tipos de aplicativos, considere o seguinte padrão de design de esquema.

	Primary Key		Attributes		
	PK (NodeId)	SK (TypeTarget, GSI 2 SK)	Data	GSI PK	Graph Projections
1	DATE 2 BIRTH	1980-12-19	Hash(Person.Data)	GSI PK	...
		Data (GSI1 SK)	GSI PK		
	PERSON 1	John Doe	Hash(Person.Data)		
		Data	GSI PK		
	PERSON 5 FRIEND	Jane Smith	Hash(Person.Data)		
		Data	GSI PK		
	PLACE 4 BIRTH	USA Texas Austin	Hash(Person.Data)		
		Data	GSI PK		
	SKILL 6	Java Developer Senior	Hash(Person.Data)		
		Data	GSI PK		
2	DATE 2	1980-12-19	0	GSI PK	...
		Data	GSI PK		
	PLACE 3	UK England London	0		
		Data	GSI PK		
	PLACE 4	USA Texas Austin	0		
		Data	GSI PK		
	DATE 2 BIRTH	1980-12-19	Hash(Person.Data)		
		Data	GSI PK		
5	PERSON 5	Jane Smith	Hash(Person.Data)	GSI PK	...
		Data	GSI PK		
	PERSON 1 FRIEND	John Doe	Hash(Person.Data)		
		Data	GSI PK		
	PLACE 3 BIRTH	UK England London	Hash(Person.Data)		
		Data	GSI PK		
	SKILL 7	Guitar Advanced	Hash(Person.Data)		
		Data	GSI PK		
6	SKILL 6	Java Developer	0	GSI PK	...
		Data	GSI PK		
7	SKILL 7	Guitar	0		
		Data	GSI PK		

GSI PK	Primary Key		Attributes				
	GSI 1 SK (Data)		Graph Projections				
O-N	1980-12-19	NodeID	TypeTarget	GSI PK	...		
		2	DATE 2				
		NodeID	TypeTarget				
		1	DATE 2 BIRTH				
		NodeID					
	Guitar	NodeID	TypeTarget				
		7	SKILL 7				
		NodeID					
	Jane Smith	5	SKILL 7				
		NodeID	TypeTarget				
		5					
		NodeID	Person 5				
O-N	Java Developer	NodeID	TypeTarget	GSI PK	...		
		6	SKILL 6				
		NodeID					
		1	SKILL 6				
		NodeID					
	John Doe	1	TypeTarget				
		NodeID	Person 1				
		5	TypeTarget				
		NodeID	Person 1 FRIEND				
		1	PLACE 3 BIRTH				
O-N	UK England London	NodeID	TypeTarget				
		3	PLACE 3				
		NodeID	TypeTarget				
		1	PLACE 3 BIRTH				
		NodeID	TypeTarget				
	USA Texas Austin	4	PLACE 4	GSI PK	...		
		NodeID	TypeTarget				
		5	PLACE 4 BIRTH				
		NodeID	TypeTarget				
		5	TypeTarget				

GSI PK	Primary Key GSI 2 SK (TypeTarget)	Attributes			
		NodeID	Data	Graph Projections	
GSI 2	DATE 2	2	1980-12-19	O-N ...	
	DATE 2 BIRTH	1			
	PERSON 1	1	Data		
	PERSON 1 FRIEND	5			
	PERSON 5	5	John Doe		
	PERSON 5 FRIEND	1			
	PLACE 3	3	Jane Smith		
	PLACE 3 BIRTH	5			
	PLACE 4	4	Data		
	PLACE 4 BIRTH	1			
SKILL 6	NodeID	6	UK England London	Data	
	NodeID	1	USA texas Austin		
	NodeID	6	Java Developer		
	NodeID	1	Java Developer Senior		
	NodeID	7	Data		
	NodeID	5	Guitar		
	NodeID	5	Data		
			Guitar Advanced		

O esquema anterior mostra uma estrutura de dados em gráfico que é definida por um conjunto de partições de dados contendo os itens que definem as bordas e os nós do gráfico. Os itens da borda contêm um atributo `Type` e um `Target`. Esses atributos são usados como parte do nome de uma chave composta “`TypeTarget`” para identificar o item em uma partição na tabela principal ou em outro índice secundário global.

O primeiro índice secundário global é criado no atributo `Data`. Esse atributo usa a sobrecarga do índice secundário global descrito anteriormente para indexar vários tipos diferentes de atributo, a saber `Dates`, `Names`, `Places` e `Skills`. Aqui, um índice secundário global está indexando efetivamente quatro atributos diferentes.

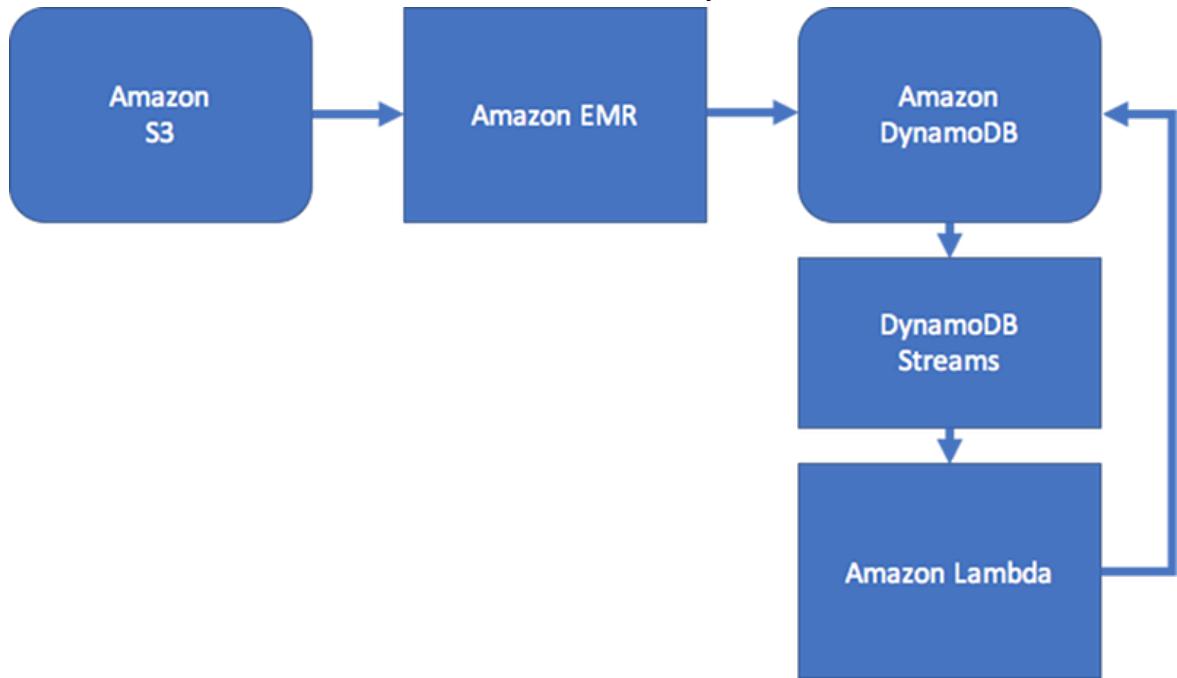
À medida que você inserir itens na tabela, poderá usar uma estratégia de fragmentação inteligente para distribuir os conjuntos de item com grandes agregações (data de nascimento, habilidade) nas partições lógicas nos índices secundários globais que forem necessárias para evitar problemas de leitura/gravação dinâmicas.

O resultado dessa combinação de padrões de design é um datastore sólido para fluxos de trabalho de gráfico altamente eficientes e em tempo real. Esses fluxos de trabalho podem fornecer consultas de alto desempenho a agregação de borda e ao estado da entidade vizinha para mecanismos de recomendação, aplicativos de rede social, classificações de nós, agregações de subárvore e outros casos de uso comuns de gráficos.

Se seu caso de uso não for sensível a consistência de dados em tempo real, você poderá usar um processo do Amazon EMR do programado para preencher as bordas com agregações relevantes de resumo de gráfico para seus fluxos de trabalho de um modo dinâmico. Se o aplicativo não precisar saber imediatamente quando uma borda é adicionada ao gráfico, será possível usar um processo programado para agregar resultados.

Para manter algum nível de consistência, o design poderia incluir os fluxos do Amazon DynamoDB e o AWS Lambda para processar atualizações de borda. Ele também poderia usar um trabalho do Amazon EMR para validar os resultados em um intervalo regular. Essa abordagem é ilustrada pelo diagrama a

seguir. Ela é usada comumente em aplicativos de rede social, onde o custo de uma consulta em tempo real é alto, e a necessidade de saber imediatamente as atualizações de usuário individual é baixa.



Os aplicativos de segurança e gerenciamento de serviços de TI (ITSM - IT service-management) geralmente precisa responder em tempo real às alterações de estado da entidade compostas de agregações complexas de borda. Esses aplicativos precisam de um sistema que seja compatível com várias agregações de nó em tempo real de relações de segundo e terceiros níveis ou de percursos complexos de borda. Se o seu caso de uso exigir esses tipos de fluxos de trabalho de consulta de gráficos em tempo real, recomendamos que você considere o uso do [Amazon Neptune](#) para gerenciar esses fluxos de trabalho.

Práticas recomendadas para implementação de um sistema híbrido de banco de dados

Em algumas circunstâncias, migrar de um ou mais sistemas de gerenciamento de banco de dados relacional (RDBMS) para o Amazon DynamoDB talvez não seja vantajoso. Nesses casos, pode ser preferível criar um sistema híbrido.

Se você não quiser migrar tudo para o DynamoDB

Por exemplo, algumas organizações fazem grandes investimentos no código que produz inúmeros relatórios necessários para contabilidade e operações. O tempo necessário para gerar um relatório não é importante para elas. A flexibilidade de um sistema relacional é adequada para esse tipo de tarefa, e recriar todos os relatórios em um contexto NoSQL pode ser proibitivamente complicado.

Algumas organizações também mantêm uma variedade de sistemas relacionais herdados que adquiriram ou herdaram ao longo de décadas. A migração dos dados desses sistemas podem ser muito arriscada e cara para justificar o esforço.

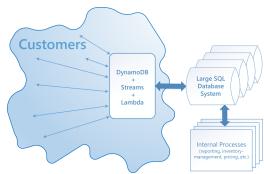
Contudo, as mesmas organizações agora podem achar que suas operações dependem de sites de alto tráfego, voltados para os clientes, onde a resposta em milissegundo é essencial. Os sistemas relacionais

não podem ser dimensionados para atender a esse requisito, exceto por uma despesa enorme (e frequentemente inaceitável).

Nessas situações, a resposta pode ser a criação de um sistema híbrido, em que o DynamoDB cria uma visualização materializada dos dados armazenados em um ou mais sistemas relacionais e lida com solicitações de alto tráfego nessa visualização. Esse tipo de sistema pode reduzir potencialmente os custos, ao eliminar as licenças de hardware de servidor, manutenção, e RDBMS que eram necessárias anteriormente para lidar com o tráfego voltado para o cliente.

Como um sistema híbrido pode ser implementado

O DynamoDB pode aproveitar os DynamoDB Streams e AWS LambdaPara integrar-se perfeitamente a um ou mais sistemas de banco de dados relacional existentes:



Um sistema que integra o DynamoDB Streams e AWS LambdaO pode fornecer várias vantagens:

- Ele pode funcionar como um cache persistente de visualizações materializadas.
- Ele pode ser configurado para ser preenchido gradualmente com dados, à medida que esses dados forem consultados e modificados no sistema SQL. Isso significa que a visualização inteira não precisa ser pré-preenchida. Isso, por sua vez, significa que a capacidade de taxa de transferência provisionada é mais provável de ser usada de forma eficiente.
- Ele tem baixos custos administrativos e é altamente disponível e confiável.

Para esse tipo de integração ser implementada, essencialmente três tipos de interoperação devem ser fornecidos.



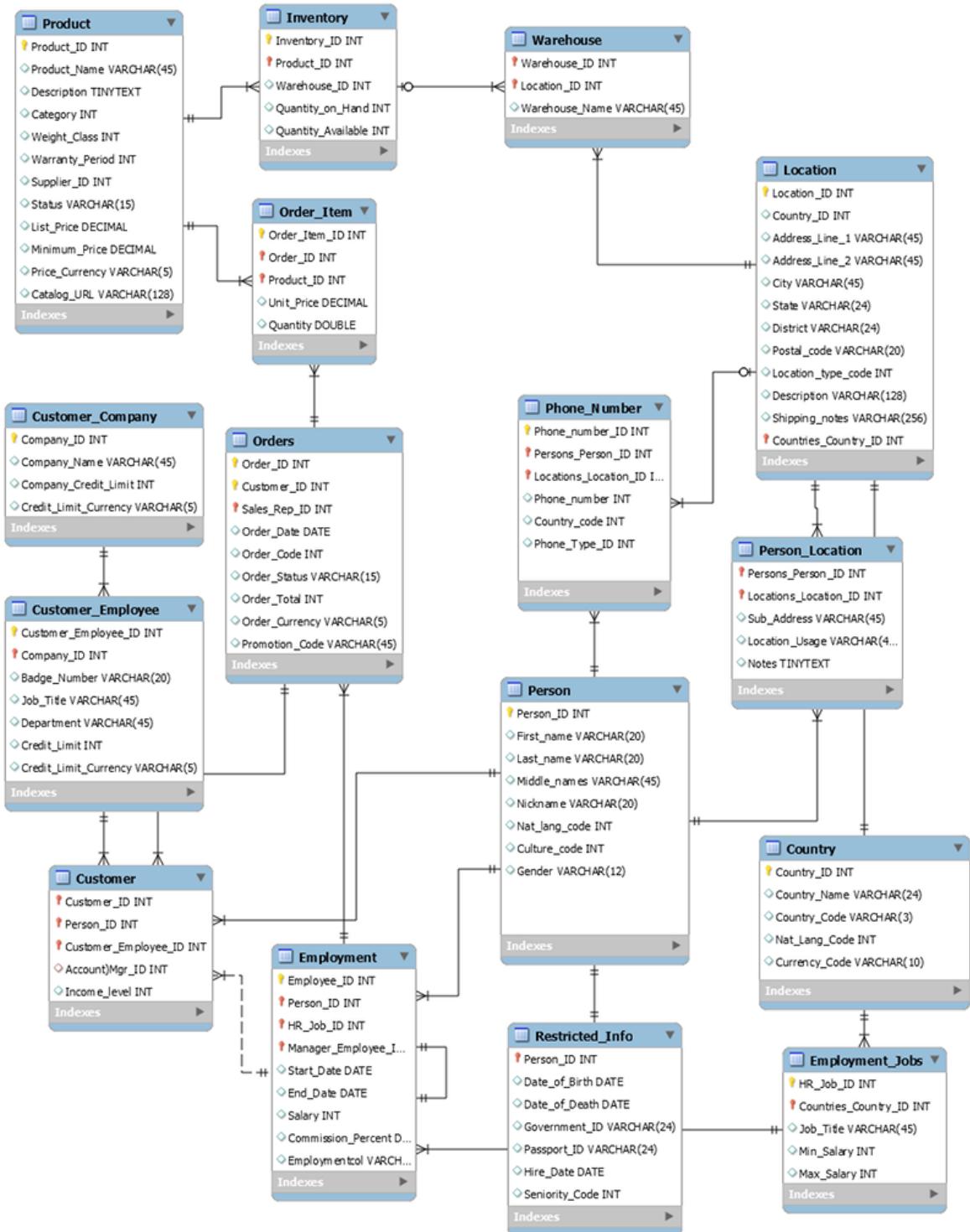
1. Preencha o cache do DynamoDB de modo incremental. Quando um item é consultado, procure-o primeiro no DynamoDB. Se ele não estiver lá, procure-o no sistema SQL e carregue-o no DynamoDB.
2. Grave por meio de um cache do DynamoDB. Quando um cliente altera um valor no DynamoDB, uma função do Lambda é acionada para gravar o novo dado de volta no sistema SQL.
3. Atualize o sistema SQL. Quando processos internos, como o gerenciamento de inventário ou a definição de preço, alteram um valor no sistema SQL, um procedimento armazenado é acionado para propagar a alteração para a visualização materializada do DynamoDB.

Essas operações são diretas, e nem todas são necessárias em todos os cenários.

Uma solução híbrida também pode ser útil quando você deseja confiar principalmente no DynamoDB, mas também deseja manter um pequeno sistema relacional para consultas únicas, ou para operações que a precisam de segurança especial ou que não são urgentes.

Práticas recomendadas para modelagem de dados relacionais no DynamoDB

As plataformas tradicionais do sistema de gerenciamento de banco de dados relacional (RDBMS) armazenam dados em uma estrutura relacional normalizada. Essa estrutura reduz as estruturas de dados hierárquicas para um conjunto de elementos comuns que são armazenados em várias tabelas. O esquema a seguir é um exemplo de um aplicativo genérico de entrada de pedidos, compatível com o esquema de RH por trás dos sistemas de suporte operacional e administrativo de um fabricante teórico.



As plataformas de RDBMS usam uma linguagem de consulta ad hoc (geralmente um tipo de SQL) para gerar ou materializar visualizações de dados normalizados para compatibilidade com os padrões de acesso à camada de aplicativos.

Por exemplo, para gerar uma lista de itens de pedidos de compra classificados pela quantidade em estoque em todos os warehouses que podem enviar cada item, você pode emitir a seguinte consulta no esquema anterior:

```
SELECT * FROM Orders
INNER JOIN Order_Items ON Orders.Order_ID = Order_Items.Order_ID
INNER JOIN Products ON Products.Product_ID = Order_Items.Product_ID
INNER JOIN Inventories ON Products.Product_ID = Inventories.Product_ID
ORDER BY Quantity_on_Hand DESC
```

Consultas desse tipo, que são feita apenas uma vez, fornecem um API flexível para acesso aos dados, mas exigem um volume significativo de processamento. Você deve consultar os dados em vários locais com frequência, e os resultados devem ser montados para apresentação. A consulta anterior inicia consultas complexas em várias tabelas e depois classifica e integra os dados resultantes.

Outro fator que pode retardar os sistemas RDBMS é a necessidade de suporte a uma estrutura de transações compatível com ACID. As estruturas de dados hierárquicas usadas pela maioria dos aplicativos de processamento de transações on-line (OLTP - Online transaction processing) devem ser divididas e distribuídas entre várias tabelas lógicas quando são armazenadas em um RDBMS. Portanto, uma estrutura de transações compatível com ACID é necessária para evitar condições de corrida que poderão ocorrer se um aplicativo tentar ler um objeto que esteja em processo de gravação. Essa estrutura de transações adiciona necessariamente uma sobrecarga significativa ao processo de gravação.

Esses dois fatores são as principais barreiras ao dimensionamento das plataformas tradicionais de RDBMS. Ela continua sendo considerada independentemente da comunidade de NewSQL ser bem-sucedida no fornecimento de uma solução distribuída de RDBMS. Mas é improvável que isso resolva as duas limitações descritas anteriormente. Não importa como a solução é fornecida; os custos de processamento da normalização e as transações de ACID devem permanecer significativas.

Por esse motivo, quando sua empresa exigir resposta de baixa latência a consultas de alto tráfego, aproveitar as vantagens de um sistema NoSQL geralmente faz sentido técnico e econômico. O Amazon DynamoDB ajuda a resolver os problemas que limitam a escalabilidade do sistema relacional ao evitá-los.

Um sistema de banco de dados relacional não tem um bom dimensionamento pelos seguintes motivos:

- Ele normaliza os dados e armazena-os em várias tabelas que exigem várias consultas para gravar em disco.
- Isso geralmente implica em custos de desempenho de um sistema de transações compatível com ACID.
- Ele usa junções caras para remontar as visualizações exigidas dos resultados de consulta.

O DynamoDB tem um bom dimensionamento por estes motivos:

- A flexibilidade do esquema permite que o DynamoDB armazene dados hierárquicos e complexos em um único item.
- O design da chave composta permite o armazenamento de itens relacionados na mesma tabela.

As consultas no armazenamento de dados tornam-se muito mais simples, normalmente da seguinte forma:

```
SELECT * FROM Table_X WHERE Attribute_Y = "somevalue"
```

O DynamoDB realiza muito menos trabalho para retornar os dados solicitados em comparação ao RDBMS no exemplo anterior.

Primeiras etapas para modelagem de dados relacionais no DynamoDB

Important

O design do NoSQL exige uma visão diferente daquela no design do RDBMS. Para um RDBMS, você pode criar um modelo de dados normalizado sem pensar nos padrões de acesso. Você poderá estendê-lo posteriormente quando surgirem novas perguntas e requisitos de consulta. Por outro lado, no Amazon DynamoDB, você não deve iniciar o design do esquema até que saiba quais perguntas ele precisa responder. Compreender os problemas de negócios e os casos de uso de aplicativo antecipadamente é absolutamente essencial.

Para iniciar o design de uma tabela do DynamoDB do que será dimensionada com eficiência, você deve realizar várias etapas primeiro para identificar os padrões de acesso exigidos pelos sistemas de suporte operacional e administrativo (OSS/BSS) que o design precisa comportar:

- Para novos aplicativos, analise as histórias dos usuários referentes a atividades e objetivos. Documente os vários casos de uso identificados e analise os padrões de acesso que eles exigem.
- Para aplicativos existentes, analise os logs de consulta para saber como as pessoas estão usando o sistema atualmente e quais são os principais padrões de acesso.

Após concluir esse processo, você deve encerrar com uma lista que pode ser semelhante à seguinte:

Most Common/Import Access Patterns in Our Organization	
1	Look up employee details by employee ID
2	Query employee details by employee name
3	Find an employee's phone number(s)
4	Find a customer's phone number(s)
5	Get orders for a given customer within a given date range
6	Show all open orders within a given date range across all customers
7	See all employees hired recently
8	Find all employees working in a given warehouse
9	Get all items on order for a given product
10	Get current inventories for a given product at all warehouses
11	Get customers by account representative
12	Get orders by account representative and date
13	Get all items on order for a given product
14	Get all employees with a given job title
15	Get inventory by product and warehouse
16	Get total product inventory
17	Get account representatives ranked by order total and sales period

Em um aplicativo real, sua lista pode ser muito mais longa. Mas essa coleção representa a faixa de complexidade dos padrões de consulta que você pode encontrar em um ambiente de produção.

Uma abordagem comum ao design de esquema do DynamoDB é identificar entidades da camada de aplicativo e usar a desnormalização e a agregação de chave composta para reduzir a complexidade da consulta.

No DynamoDB, isso significa usar as chaves de classificação compostas, os índices secundários globais sobrecarregados, as tabelas/índices particionados e outros padrões de design. Você pode usar esses elementos para estruturar os dados, para que um aplicativo possa recuperar o que for necessário para um determinado padrão de acesso, usando uma única consulta em uma tabela ou um índice. O padrão principal que você pode usar para modelar o esquema normalizado, mostrado em [Modelagem](#)

relacional ([p. 1008](#)), é o padrão da lista de adjacências. Outros padrões usados nesse design podem incluir a fragmentação de gravação do índice secundário global, a sobrecarga do índice secundário global, as chaves compostas e as agregações materializadas.

Important

Em geral, você deve manter o mínimo de tabelas possível em um aplicativo do DynamoDB. As exceções incluem os casos que envolvem dados de séries temporais de alto volume ou conjuntos de dados que têm padrões muito diferentes de acesso. Uma única tabela com índices invertidos pode normalmente habilitar consultas simples para criar e recuperar estruturas de dados hierárquicas e complexas, exigidas pelo aplicativo.

Exemplo de modelagem de dados relacionais no DynamoDB

Esse exemplo descreve como modelar dados relacionais no Amazon DynamoDB. Um design de tabela do DynamoDB corresponde ao esquema relacional de entrada de pedidos que é mostrado em [Modelagem relacional \(p. 1008\)](#). Segue-se a [Padrão de design da lista de adjacências \(p. 1002\)](#), que é um modo comum de representar as estruturas de dados relacionais no DynamoDB.

O padrão de design requer que você defina um conjunto de tipos de entidades que se correlacionam geralmente com várias tabelas no esquema relacional. Os itens da entidade são adicionados à tabela usando uma chave primária composta (partição e classificação). A chave de partição desses itens de entidade é o atributo que identifica exclusivamente o item e que é mencionado genericamente em todos os itens como **PK**. O atributo de chave de classificação contém um valor do atributo que pode ser usado para um índice invertido ou um índice secundário global. Ele é referido genericamente como **SK**.

Você define as seguintes entidades que são compatíveis com o esquema relacional de entrada de pedidos.

1. HR-Employee - PK: EmployeeID, SK: Nome do funcionário
2. HR-Region - PK: RegionID, SK: Nome da região
3. HR-Country - PK: CountryId, SK: Nome do país
4. HR-Location - PK: LocationID, SK: Nome do país
5. HR-Job - PK: JobID, SK: Título do Job
6. HR-Department - PK: DepartmentID, SK: DepartmentID
7. OE-Customer - PK: CustomerID, SK: AccountRepID
8. OE-Order - PK OrderID, SK: CustomerID
9. OE-Product - PK: ProductID, SK: Nome do produto
10. OE-Warehouse - PK: WarehouseID, SK: Nome da região

Após ter adicionado esses itens de entidade à tabela, você pode definir as relações entre eles adicionando itens de borda às partições de item de entidade. A tabela a seguir demonstra essa etapa.

Nesse exemplo, as partições `Employee`, `Order` e `Product Entity` na tabela têm itens de borda adicionais que contêm indicadores para outros itens de entidade na tabela. Em seguida, defina alguns índices secundários globais (GSIs – Global secondary indexes) para compatibilidade com todos os padrões de acesso definidos anteriormente. Os itens de entidade não usam o mesmo tipo de valor para a chave primária nem o atributo de chave de classificação. Tudo isso é necessário para que os atributos de chave primária e de chave de classificação presentes sejam inseridos na tabela.

O fato de algumas dessas entidades usarem nomes próprios e outras usarem outros IDs de entidade como os valores da chave de classificação permite que o mesmo índice secundário global seja compatível com vários tipos de consultas. Essa técnica é chamada de sobrecarga de GSI. Ela elimina efetivamente

o limite padrão de 20 índices secundários globais para as tabelas que contêm vários tipos de itens. Isso é mostrado no diagrama a seguir como GSI 1.

O GSI 2 foi projetado para compatibilidade com um padrão bastante comum de acesso a aplicativos, que é obter todos os itens na tabela que têm um determinado estado. Para uma tabela grande com uma distribuição desigual de itens entre os estados disponíveis, esse padrão de acesso pode resultar em uma chave dinâmica, a menos que os itens sejam distribuídos em mais de uma partição lógica que pode ser consultada simultaneamente. Esse padrão de design é chamado `write sharding`.

Para realizar isso para o GSI 2, o aplicativo adiciona o atributo de chave primária do GSI 2 a cada item de pedidos. Ele preenche o campo com um número aleatório em um intervalo de 0 a N, em que N pode ser genericamente calculado usando a seguinte fórmula, a menos que haja um motivo específico para se fazer de outra maneira.

```
ItemsPerRCU = 4KB / AvgItemSize  
PartitionMaxReadRate = 3K * ItemsPerRCU  
N = MaxRequiredIO / PartitionMaxReadRate
```

Por exemplo, suponha que você espere o seguinte:

- Até 2 milhões de pedidos estarão no sistema, aumentando para 3 milhões em 5 anos.
- Até 20% desses pedidos estarão em um estado ABERTO por um determinar tempo.
- O registro médio de pedidos é cerca de 100 bytes, com três registros de OrderItem na partição de pedidos com cerca de 50 bytes cada, oferecendo um tamanho médio de entidade de pedidos de 250 bytes.

Para a tabela, o cálculo do fator N seria semelhante ao seguinte.

```
ItemsPerRCU = 4KB / 250B = 16  
PartitionMaxReadRate = 3K * 16 = 48K  
N = (0.2 * 3M) / 48K = 13
```

Nesse caso, você precisa distribuir todos os pedidos em pelo menos 13 partícões lógicas em GSI 2 para garantir que uma leitura de todos os itens de `Order` com um status `OPEN` não cause uma partição dinâmica na camada de armazenamento físico. É uma boa prática preencher esse número para permitir anomalias no conjunto de dados. Então, um modelo que use `N = 15` é provavelmente bom. Como mencionado anteriormente, você faz isso adicionando o valor 0—N aleatório ao atributo PK GSI 2 de cada `Order` inserido na tabela.

Esse detalhamento supõe que o padrão de acesso que requer que a coleta de todas as faturas `OPEN` ocorra relativamente sem frequência, de modo que você possa usar a capacidade de intermitênciça para preencher a solicitação. Você pode consultar o seguinte índice secundário global usando uma condição de chave de classificação de `State` e de `Date Range` para produzir um subconjunto ou todos os `Orders` em um determinado estado, conforme necessário.

Neste exemplo, os itens são distribuídos de modo aleatório entre 15 partícões lógicas. Essa estrutura funciona, porque o padrão acesso exige a recuperação de um grande número de itens. Portanto, é improvável que alguns dos 15 threads retornarão conjuntos vazios de resultados que poderiam potencialmente representar a capacidade desperdiçada. Uma consulta sempre usa 1 unidade de capacidade de leitura (RCU) ou 1 unidade de capacidade de gravação (WCUs), mesmo se nada for retornado ou nenhum dado for gravado.

Se o padrão acesso exigir uma consulta de alta velocidade nesse índice secundário global que retorna um conjunto de resultados esparsos, é provavelmente melhor usar um algoritmo hash para distribuir os

itens em vez de um padrão aleatório. Nesse caso, você pode selecionar um atributo que seja conhecido quando a consulta for executada em tempo de execução e aplicar hash a esse atributo em um espaço de chaves de 0 a 14, quando os itens são inseridos. Então, eles podem ser lidos de modo eficiente no índice secundário global.

Por fim, você pode reanalisar os padrões de acesso que foram definidos anteriormente. A seguir está a lista de padrões de acesso e as condições de consulta que serão usadas com a nova versão do DynamoDB do aplicativo para acomodá-los.

Melhores práticas para consulta e verificação de dados

Esta seção aborda algumas práticas recomendadas para o uso do `Query` e `Scan` operações no Amazon DynamoDB.

Considerações sobre desempenho das verificações

Em geral, `Scan` operações são menos eficientes do que outras operações no DynamoDB. `AScan` sempre verifica toda a tabela ou índice secundário. Ele, em seguida, exclui valores para fornecer o resultado desejado, adicionando, finalmente, a etapa extra de remover os dados do conjunto de resultados.

Se possível, você deve evitar o uso de uma `Scan` operação em uma tabela ou índice grande com um filtro que remove muitos resultados. Além disso, conforme uma tabela ou índice cresce, o `Scan` operação fica mais lenta. O `Scan` operação examina cada item para os valores solicitados e pode usar o throughput provisionado para uma tabela ou índice grande em uma única operação. Para tempos de resposta mais rápidos, crie suas tabelas e índices para que seus aplicativos possam usar o `query` INSTOFGSCAN. (Para tabelas, você também pode considerar o uso do `GetItemBatchGetItem` APIs.)

Como alternativa, crie seu aplicativo para usar operações `Scan` de uma forma que minimize o impacto em sua taxa de solicitações.

Evitar picos súbitos em atividade de leitura

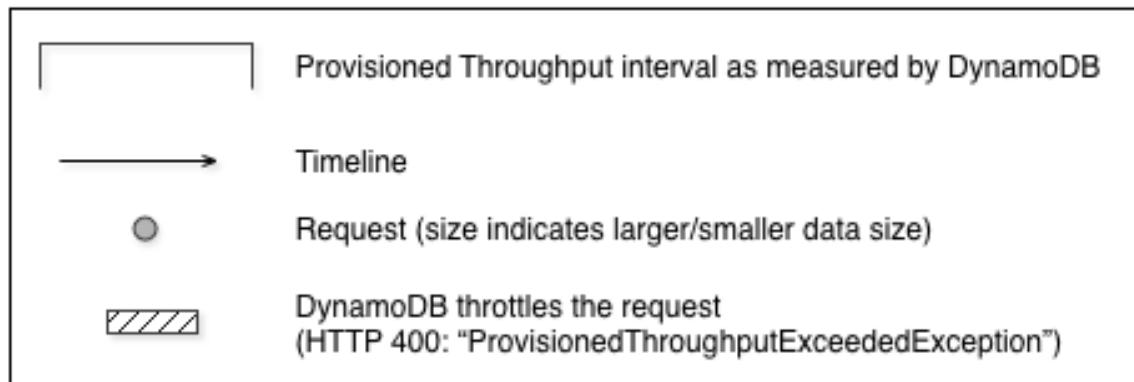
Ao criar uma tabela, você define seus requisitos de unidade de capacidade de leitura e gravação. Para leituras, as unidades de capacidade são expressas como o número de solicitações de leitura de dados de 4 KB fortemente consistentes por segundo. Para leituras eventualmente consistentes, uma unidade de capacidade de leitura é duas solicitações de leitura de 4 KB por segundo. `AScan` executa leituras eventualmente consistentes por padrão, e pode retornar até 1 MB (uma página) de dados. Portanto, um único `Scan` solicitação pode consumir (tamanho de página 1 MB / tamanho de item 4 KB) / 2 (leituras eventualmente consistentes) = 128 operações de leitura. Se você solicitar leituras fortemente consistentes, o `Scan` consumiria duas vezes mais throughput provisionado — 256 operações de leitura.

Isso representa um aumento repentino no uso, em comparação com a capacidade de leitura configurada para a tabela. Esse uso de unidades de capacidade por uma verificação impede que outras solicitações potencialmente mais importantes para a mesma tabela use as unidades de capacidade disponíveis. Como resultado, você provavelmente recebe um `ProvisionedThroughputExceeded` exceção para essas solicitações.

O problema não é apenas o aumento repentino em unidades de capacidade que o `Scan` usa. A verificação também provavelmente consome todas as suas unidades de capacidade da mesma partição, pois a verificação solicita itens de leitura que estão próximos uns dos outros na partição. Isso significa que a solicitação está alcançando a mesma partição, fazendo com que todas as suas unidades de capacidade

sejam consumidas, e limitando outras solicitações para essa partição. Se a solicitação para leitura de dados estiver dividida entre várias partições, a operação não limitaria uma partição específica.

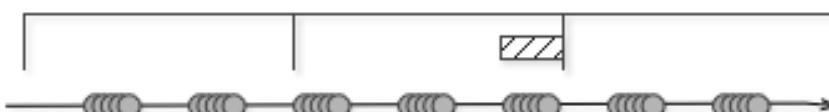
O diagrama a seguir ilustra o impacto de um pico súbito no uso de unidades de capacidade pelas operações `Query` e `Scan`, e seu impacto nas outras solicitações para a mesma tabela.



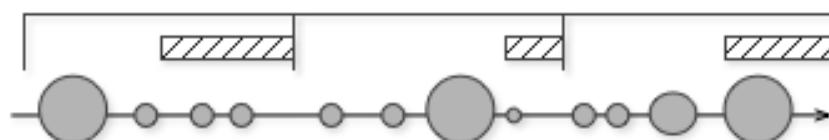
1. Good: Even distribution of requests and size



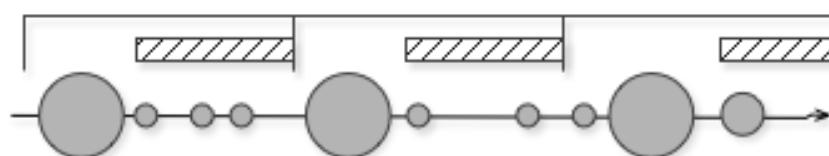
2. Not as Good: Frequent requests in bursts



3. Bad: A few random large requests



4. Bad: Large scan operations



Como ilustrado aqui, o pico de uso pode afetar a taxa de transferência provisionada da tabela de várias maneiras:

1. Bom: Distribuição uniforme de solicitações e tamanho
2. Não tão bom. Pedidos frequentes em explosões
3. Não ideal: Algumas solicitações grandes aleatórias
4. Não ideal: Operações Scan grandes

Em vez de usar uma operação Scan grande, você pode usar as técnicas a seguir para minimizar o impacto de uma verificação no throughput provisionado de uma tabela.

- Reduzir o tamanho da página

Como uma operação Scan lê uma página inteira (por padrão, 1 MB), você pode reduzir o impacto da operação Scan, configurando um tamanho de página menor. O Scanfornece uma operaçãoScanLimitO parâmetro que você pode usar para definir o tamanho da página para sua solicitação. CadaQueryyouScanque tenha um tamanho de página menor usa menos operações de leitura e cria uma "pausa" entre cada solicitação. Por exemplo, suponha que cada item tem 4 KB e defina o tamanho da página para 40 itens. AQuery, em seguida, consumir apenas 20 operações de leitura eventualmente consistentes ou 40 operações de leitura altamente consistentes. Um número maior de operações Query ou Scan menores permitiria que suas outras solicitações críticas fossem bem-sucedidas sem limitação.

- Operações Scan isoladas

O DynamoDB foi projetado para facilitar a escalabilidade. Como resultado, um aplicativo pode criar tabelas para fins distintos e, possivelmente, duplicar conteúdo em várias tabelas. Você deseja executar verificações em uma tabela que não está levando tráfego de "missão crítica". Alguns aplicativos lidam com essa carga, roteando tráfego entre duas tabelas por hora — uma para tráfego crítico e uma para contabilidade. Outros aplicativos podem fazer isso, executando todas as gravações em duas tabelas: uma tabela de "missão crítica" e uma tabela de "sombra".

Configure seu aplicativo para tentar novamente qualquer solicitação que recebe um código de resposta que indica que você excedeu seu throughput provisionado. Ou, aumente a taxa de transferência provisionada para sua tabela usando oUpdateTableoperação. Se você tiver picos temporárias na sua carga de trabalho que fazem com que seu throughput exceda, ocasionalmente, o nível provisionado, tente novamente a solicitação com recuo exponencial. Para obter mais informações sobre a implementação de recuo exponencial, consulte [Repetições de erro e recuo exponencial \(p. 226\)](#).

Aproveitar verificações paralelas

Muitos aplicativos podem se beneficiar do uso de paralelasScanOperações, em vez de verificações sequenciais. Por exemplo, um aplicativo que processa uma grande tabela de dados históricos pode executar uma verificação paralela muito mais rapidamente do que uma verificação sequencial. Vários threads de operadores em um processo de "varredura" em segundo plano poderiam verificar uma tabela com baixa prioridade sem afetar o tráfego de produção. Em cada um desses exemplos, uma operação Scan paralela é usada de forma a não enfraquecer outros aplicativos de recursos de throughput provisionado.

Embora as verificações paralelas sejam benéficas, elas podem representar uma demanda pesada sobre o throughput provisionado. Com uma verificação paralela, seu aplicativo tem vários trabalhadores que estão executandoScanOperações simultaneamente. Isso pode consumir rapidamente toda a capacidade de leitura provisionada da sua tabela. Neste caso, outros aplicativos que precisam acessar a tabela podem ser limitados.

A verificação paralela pode ser a escolha certa se as seguintes condições forem atendidas:

- O tamanho da tabela é 20 GB ou maior.
- O throughput de leitura provisionado da tabela não está sendo completamente usado.
- Operações Scan sequenciais são muito lentas.

Escolha TotalSegments

A melhor configuração para `TotalSegments` depende de seus dados específicos, das configurações de throughput provisionado da tabela e de seus requisitos de desempenho. Você pode precisar experimentar para conseguir acertar. Recomendamos que você comece com uma taxa simples, como um segmento por 2 GB de dados. Por exemplo, para uma tabela de 30 GB, você pode definir `totalSegments` até 15 (30 GB/2 GB). Seu aplicativo poderia usar 15 operadores, com cada operador verificando um segmento diferente.

Também é possível escolher um valor para `totalSegments` que é baseado em recursos do cliente. Você pode definir `totalSegments` para qualquer número de 1 a 1000000, e o DynamoDB permite que você execute uma verificação desse número de segmentos. Por exemplo, se o cliente limita o número de threads que podem ser executados simultaneamente, você pode gradualmente aumentar `totalSegments` até obter o melhor desempenho com seu aplicativo.

Monitore suas verificações paralelas para otimizar o uso de throughput provisionado e, ao mesmo tempo, garantir que seus outros aplicativos não fiquem carentes de recursos. Aumente o valor `totalSegments` se você não consumir todo o seu throughput provisionado, mas ainda houver limitação em suas solicitações de scan. Reduza o valor de `totalSegments` se as solicitações de scan consomem mais throughput provisionado do que você deseja usar.

Integração do DynamoDB com outros AWS Serviços

O Amazon DynamoDB é integrado a outros AWS para permitir que você automatize tarefas repetitivas ou crie aplicativos que abrangem vários serviços. Por exemplo:

Tópicos

- [Configurar o AWS Credenciais em seus arquivos usando o Amazon Cognito \(p. 1018\)](#)
- [Carregando dados do DynamoDB para o Amazon Redshift \(p. 1020\)](#)
- [Processamento de dados do DynamoDB com o Apache Hive no Amazon EMR \(p. 1021\)](#)
- [Exportando dados de tabela do DynamoDB para o Amazon S3 \(p. 1046\)](#)

Configurar o AWS Credenciais em seus arquivos usando o Amazon Cognito

A maneira recomendada de obter AWS para seus aplicativos móveis e da Web é usar o Amazon Cognito. O Amazon Cognito ajuda a evitar a codificação do AWS em seus arquivos. Ele usa AWS Identity and Access Management Funções do (IAM) com o objetivo de gerar credenciais temporárias para os usuários autenticados e não autenticados do seu aplicativo.

Por exemplo, para configurar seus arquivos JavaScript para usar uma função não autenticada do Amazon Cognito para acessar o Web service do Amazon DynamoDB, faça o seguinte:

Para configurar credenciais a serem integradas ao Amazon Cognito

1. Crie um pool de identidades do Amazon Cognito que permita identidades não autenticadas.

```
aws cognito-identity create-identity-pool \
    --identity-pool-name DynamoPool \
    --allow-unauthenticated-identities \
    --output json
{
    "IdentityPoolId": "us-west-2:12345678-1ab2-123a-1234-a12345ab12",
    "AllowUnauthenticatedIdentities": true,
    "IdentityPoolName": "DynamoPool"
}
```

2. Copie a seguinte política para um arquivo denominado `myCognitoPolicy.json`. Substitua o ID do grupo de identidades (`us-west-2:12345678-1ab2-123a-1234-a12345ab12`) pelo seu próprio `IdentityPoolId` obtido na etapa anterior.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
        "Federated": "cognito-identity.amazonaws.com"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
        "StringEquals": {
            "cognito-identity.amazonaws.com:aud": "us-west-2:12345678-1ab2-123a-1234-
a12345ab12"
        },
        "ForAnyValue:StringLike": {
            "cognito-identity.amazonaws.com:amr": "unauthenticated"
        }
    }
}
```

- Crie uma função do IAM que assuma a política anterior. Dessa forma, o Amazon Cognito torna-se uma entidade confiável capaz de assumir oCognito_DynamoPoolUnauthFunção do .

```
aws iam create-role --role-name Cognito_DynamoPoolUnauth \
--assume-role-policy-document file://PathToFile/myCognitoPolicy.json --output json
```

- Conceda oCognito_DynamoPoolUnauthAcesso total à função ao DynamoDB, anexando uma política gerenciada (AmazonDynamoDBFullAccess).

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonDynamoDBFullAccess \
--role-name Cognito_DynamoPoolUnauth
```

Note

Como alternativa, você pode conceder acesso refinado ao DynamoDB. Para obter mais informações, consulte [Como usar condições de política do IAM para um controle de acesso refinado](#).

- Obtenha e copie a função do Amazon Resource Name (ARN) do IAM.

```
aws iam get-role --role-name Cognito_DynamoPoolUnauth --output json
```

- Adicione a função Cognito_DynamoPoolUnauth ao pool de identidades DynamoPool. O formato a ser especificado é KeyName=string, onde KeyName é unauthenticated e a string é o ARN da função obtido na etapa anterior.

```
aws cognito-identity set-identity-pool-roles \
--identity-pool-id "us-west-2:12345678-1ab2-123a-1234-a12345ab12" \
--roles unauthenticated=arn:aws:iam::123456789012:role/Cognito_DynamoPoolUnauth --
output json
```

- Especifique as credenciais do Amazon Cognito nos seus arquivos. Modifique IdentityPoolId e RoleArn de acordo.

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
IdentityPoolId: "us-west-2:12345678-1ab2-123a-1234-a12345ab12",
RoleArn: "arn:aws:iam::123456789012:role/Cognito_DynamoPoolUnauth"
});
```

Agora, você pode executar seus programas JavaScript no serviço Web DynamoDB usando credenciais do Amazon Cognito. Para obter mais informações, consulte [Definir credenciais em um navegador da Web](#) no AWS SDK for JavaScript Guia de conceitos básicos.

Carregando dados do DynamoDB para o Amazon Redshift

O Amazon Redshift complementa o Amazon DynamoDB com recursos avançados de inteligência de negócios e uma poderosa interface baseada em SQL. Ao copiar os dados de uma tabela do DynamoDB para o Amazon Redshift, você pode realizar consultas de análise de dados complexas nesses dados, incluindo junções com outras tabelas em seu cluster Amazon Redshift.

Em termos de throughput provisionado, uma operação de cópia de uma tabela do DynamoDB entra na contagem da capacidade de leitura dessa tabela. Depois que os dados são copiados, as consultas SQL no Amazon Redshift não afetam o DynamoDB de forma alguma. Isso ocorre porque as consultas agem em uma cópia dos dados do DynamoDB, em vez de no próprio DynamoDB.

Antes de você poder carregar dados de uma tabela do DynamoDB, você deve primeiro criar uma tabela do Amazon Redshift para servir como o destino para os dados. Lembre-se de que você está copiando dados de um ambiente NoSQL para um ambiente SQL, e que há determinadas regras em um ambiente que não se aplicam ao outro. Veja algumas das diferenças a considerar:

- Os nomes de tabela do DynamoDB podem conter até 255 caracteres, incluindo os caracteres “.” (ponto) e “-” (traço) e diferenciam maiúsculas e minúsculas. Os nomes de tabela do Amazon Redshift são limitados a 127 caracteres, não podem conter pontos ou traços e não diferenciam maiúsculas e minúsculas. Além disso, nomes de tabela não podem entrar em conflito com quaisquer palavras reservadas do Amazon Redshift.
- O DynamoDB não é compatível com o conceito SQL de NULL. Você precisa especificar como o Amazon Redshift interpreta valores de atributo vazios ou em branco no DynamoDB, tratando-os como NULLs ou como campos vazios.
- Os tipos de dados do DynamoDB não correspondem diretamente aos do Amazon Redshift. Você precisa garantir que cada coluna na tabela do Amazon Redshift seja do tipo e tamanho corretos para acomodar os dados do DynamoDB.

Este é um exemplo do comando COPY do Amazon Redshift SQL:

```
copy favoritemovies from 'dynamodb://my-favorite-movies-table'  
credentials 'aws_access_key_id=<Your-Access-Key-ID>;aws_secret_access_key=<Your-Secret-  
Access-Key>'  
readratio 50;
```

Neste exemplo, a tabela de origem no DynamoDB é `my-favorite-movies-table`. A tabela de destino no Amazon Redshift é `favoritemovies`. A cláusula `readratio 50` regula a percentagem do throughput provisionado que é consumida; neste caso, o comando COPY usará não mais que 50% das unidades de capacidade de leitura provisionadas para `my-favorite-movies-table`. É altamente recomendável definir esse índice para um valor menor do que a média de throughput provisionado não utilizado.

Para obter instruções detalhadas sobre como carregar dados do DynamoDB no Amazon Redshift, consulte as seções a seguir no [Guia Amazon Redshift de banco de dados](#):

- [Carregamento de dados de uma tabela do DynamoDB](#)
- [O comando COPY](#)
- [Exemplos de COPY](#)

Processamento de dados do DynamoDB com o Apache Hive no Amazon EMR

O Amazon DynamoDB é integrado ao Apache Hive, um aplicativo de armazenamento de dados que pode ser executado no Amazon EMR. O Hive pode ler e gravar dados em tabelas do DynamoDB, permitindo que você:

- Consulte dados dinâmicos do DynamoDB usando uma linguagem semelhante ao SQL (HiveQL).
- Copie dados de uma tabela do DynamoDB para um bucket do Amazon S3, e vice-versa.
- Copie dados de uma tabela do DynamoDB para o Hadoop Distributed File System (HDFS) e vice-versa.
- Execute operações de junção em tabelas do DynamoDB.

Tópicos

- [Overview \(p. 1021\)](#)
- [Tutorial: Trabalho com o Amazon DynamoDB e o Apache Hive \(p. 1022\)](#)
- [Criação de uma tabela externa no Hive \(p. 1028\)](#)
- [Processamento de instruções HiveQL \(p. 1030\)](#)
- [Consulta de dados no DynamoDB \(p. 1031\)](#)
- [Cópia de dados para e do Amazon DynamoDB \(p. 1033\)](#)
- [Ajuste de desempenho \(p. 1043\)](#)

Overview

O Amazon EMR é um serviço que facilita o processamento de grandes quantidades de dados de maneira rápida e econômica. Para usar o Amazon EMR, você inicia um cluster gerenciado de instâncias do Amazon EC2 executando o framework de código aberto do Hadoop. HadoopO é um aplicativo distribuído que implementa o algoritmo MapReduce, onde uma tarefa é mapeada para vários nós no cluster. Cada nó processa seu trabalho designado, em paralelo com outros nós. Finalmente, as saídas são reduzidas em um único nó, gerando o resultado final.

Você pode optar por iniciar seu cluster do Amazon EMR para que ele seja persistente ou transitório:

- ApersistenteO cluster é executado até que você o desative. Os clusters persistentes são ideais para análise de dados, armazenamento de dados ou qualquer outro uso interativo.
- AtransitórioO cluster é executado por tempo suficiente para processar um fluxo de trabalho e, em seguida, é desativado automaticamente. Os clusters transitórios são ideais para tarefas de processamento periódicas, como a execução de scripts.

Para obter informações sobre o Amazon EMR, arquitetura e administração, consulte o[Guia de gerenciamento do Amazon EMR](#).

Ao iniciar um cluster Amazon EMR você especifica o número inicial e o tipo de instâncias do Amazon EC2. Você também especifica outros aplicativos distribuídos (além do Hadoop em si) que você deseja executar no cluster. Esses aplicativos incluem Matiz, Mahout, Pig, Spark e muito mais.

Para obter informações sobre aplicativos para o Amazon EMR, consulte o[Guia de apresentação do Amazon EMR](#).

Dependendo da configuração do cluster, é possível ter um ou mais dos seguintes tipos de nó:

- Nó líder — gerencia o cluster, coordenando a distribuição do MapReduce executável e os subconjuntos de dados brutos, o núcleo e os grupos de instâncias de tarefa. Ele também monitora o status de cada tarefa executada e monitora a integridade dos grupos de instâncias. Há apenas um nó principal em um cluster.
- Nós core - executam tarefas do MapReduce e armazenam dados usando o Hadoop Distributed File System (HDFS).
- Nós de tarefas (opcional) – executam tarefas do MapReduce.

Tutorial: Trabalho com o Amazon DynamoDB e o Apache Hive

Neste tutorial, você iniciará um cluster Amazon EMR e, em seguida, usará o Apache Hive para processar os dados armazenados em uma tabela do DynamoDB.

HiveO é um aplicativo de data warehouse para o Hadoop que permite processar e analisar dados de várias fontes. O Hive fornece uma linguagem semelhante ao SQL,HiveQL, que permite que você trabalhe com dados armazenados localmente no cluster Amazon EMR ou em uma fonte de dados externa (como o Amazon DynamoDB).

Para obter mais informações, consulte o [Tutorial do Hive](#).

Tópicos

- [Antes de começar \(p. 1022\)](#)
- [Etapa 1: Criar um par de chaves do Amazon EC2 \(p. 1022\)](#)
- [Etapa 2: Iniciar um cluster do Amazon EMR \(p. 1023\)](#)
- [Etapa 3: Connect ao nó líder \(p. 1024\)](#)
- [Etapa 4: Carregamento de dados no HDFS \(p. 1024\)](#)
- [Etapa 5: Copiar dados para o DynamoDB \(p. 1026\)](#)
- [Etapa 6: Consulte os dados na tabela do DynamoDB \(p. 1027\)](#)
- [Etapa 7: Limpar \(opcional\) \(p. 1027\)](#)

Antes de começar

Para este tutorial, você precisará do seguinte:

- Uma conta da AWS. Se você não tiver uma, consulte [Cadastrar-se no AWS \(p. 58\)](#).
- Um cliente SSH (Secure Shell). É possível usar o cliente SSH para se conectar ao nó principal do cluster Amazon EMR e executar comandos interativos. Os clientes SSH estão disponíveis por padrão na maioria das instalações de Linux, Unix e Mac OS X. Os usuários do Windows podem fazer download e instalar o cliente [PuTTY](#), que oferece suporte para SSH.

Próxima etapa

[Etapa 1: Criar um par de chaves do Amazon EC2 \(p. 1022\)](#)

Etapa 1: Criar um par de chaves do Amazon EC2

Nesta etapa, você criará o key pair do Amazon EC2 de que você precisa para se conectar a um nó principal do Amazon EMR e executar comandos do Hive.

1. Faça login no AWS Management Console E abra o console do Amazon EC2 em <https://console.aws.amazon.com/ec2/>.
2. Escolha uma região (por exemplo, US West (Oregon)). Essa deve ser a mesma região em que a tabela do DynamoDB está localizada.
3. No painel de navegação, selecione Key Pairs (Pares de chaves).
4. Escolha Criar par de chaves.
5. Em Key pair name, digite um nome para o seu par de chaves (por exemplo, mykeypair) e, em seguida, escolha Create.
6. Faça download do arquivo de chave privada. O nome do arquivo terminará com .pem (como mykeypair.pem). Mantenha esse arquivo de chave privada em um lugar seguro. Você precisará dele para acessar qualquer cluster do Amazon EMR que executar com esse key pair.

Important

Se você perder o key pair, não poderá se conectar ao nó principal de seu cluster Amazon EMR.

Para obter mais informações sobre pares de chaves, consulte [Pares de chaves do Amazon EC2](#) no Guia do usuário do Amazon EC2 para instâncias do Linux.

Próxima etapa

[Etapa 2: Iniciar um cluster do Amazon EMR \(p. 1023\)](#)

Etapa 2: Iniciar um cluster do Amazon EMR

Nesta etapa, você poderá configurar e iniciar um cluster do Amazon EMR. O Hive e um identificador de armazenamento para o DynamoDB já estarão instalados no cluster.

1. Abra o console do Amazon EMR em <https://console.aws.amazon.com/elasticmapreduce/>.
2. Selecione Create Cluster (Criar cluster).
3. Na página Create Cluster - Quick Options, faça o seguinte:
 - a. Em Cluster name, digite um nome para o seu cluster (por exemplo: My EMR cluster).
 - b. Em EC2 key pair, escolha o par de chaves que você criou mais cedo.

Deixe as outras configurações nos valores padrão.

4. Selecione Create cluster (Criar cluster).

Vai levar vários minutos para o seu cluster ser iniciado. Você pode usar o Detalhes do cluster no console do Amazon EMR para monitorar seu progresso.

Quando o status mudar para Waiting, o cluster estará pronto para uso.

Arquivos de log do cluster e Amazon S3

Um cluster Amazon EMR gera arquivos de log que contêm informações sobre o status do cluster e as informações de depuração. As configurações padrão para o Criar Cluster - Opções Rápidas incluem a configuração do registro em log do Amazon EMR.

Se um ainda não existir, o AWS Management Console cria um bucket do Amazon S3. O nome do bucket é `aws-logs-account-id-region`, em que `account-id` é seu AWS Número da conta e `region` é a região onde você iniciou o cluster (por exemplo, `aws-logs-123456789012-us-west-2`).

Note

Você pode usar o console do Amazon S3 para visualizar os arquivos de log. Para obter mais informações, consulte [Exibir arquivos de log](#) no Guia de gerenciamento do Amazon EMR.

Você pode usar esse bucket para outras finalidades, além de registro em log. Por exemplo, você pode usar o bucket como um local para armazenar um script do Hive ou como um destino ao exportar dados do Amazon DynamoDB para o Amazon S3.

Próxima etapa

[Etapa 3: Connect ao nó líder \(p. 1024\)](#)

Etapa 3: Connect ao nó líder

Quando o status do cluster do Amazon EMR for alterado para `waiting`, você poderá se conectar ao nó principal usando o SSH e executar operações de linha de comando.

1. No console do Amazon EMR, escolha o nome do cluster para visualizar seu status.
2. No Detalhes do cluster, localize o DNS público líder field. Este é o nome DNS público do nó principal do seu cluster do Amazon EMR.
3. À direita do nome DNS, escolha o link SSH.
4. Siga as instruções em [Connect ao nó principal usando SSH](#).

Dependendo do sistema operacional, escolha a opção [Windows](#) ou a guia [Mac/Linux](#) e siga as instruções para se conectar ao nó principal.

Depois de se conectar ao nó principal usando o SSH ou o PuTTY, você deverá ver um prompt de comando semelhante ao seguinte:

```
[hadoop@ip-192-0-2-0 ~]$
```

Próxima etapa

[Etapa 4: Carregamento de dados no HDFS \(p. 1024\)](#)

Etapa 4: Carregamento de dados no HDFS

Nesta etapa, você poderá copiar um arquivo de dados para o Hadoop Distributed File System (HDFS) e, em seguida, criar uma tabela do Hive externa que mapeia para o arquivo de dados.

Faça download dos dados de exemplo

1. Faça download do arquivo de dados de exemplo (`features.zip`):

```
wget https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/features.zip
```

2. Extraia o arquivo `features.txt` do arquivo:

```
unzip features.zip
```

3. Visualize as primeiras linhas do arquivo `features.txt`:

```
head features.txt
```

O resultado deve ter a seguinte aparência:

```
1535908|Big Run|Stream|WV|38.6370428|-80.8595469|794
875609|Constable Hook|Cape|NJ|40.657881|-74.0990309|7
1217998|Gooseberry Island|Island|RI|41.4534361|-71.3253284|10
26603|Boone Moore Spring|Spring|AZ|34.0895692|-111.410065|3681
1506738|Missouri Flat|Flat|WA|46.7634987|-117.0346113|2605
1181348|Minnow Run|Stream|PA|40.0820178|-79.3800349|1558
1288759|Hunting Creek|Stream|TN|36.343969|-83.8029682|1024
533060|Big Charles Bayou|Bay|LA|29.6046517|-91.9828654|0
829689|Greenwood Creek|Stream|NE|41.596086|-103.0499296|3671
541692|Button Willow Island|Island|LA|31.9579389|-93.0648847|98
```

O `features.txt` arquivo contém um subconjunto de dados do United States Board on Geographic Names (http://geonames.usgs.gov/domestic/download_data.htm). Os campos em cada linha representam o seguinte:

- ID do recurso (identificador exclusivo)
- Nome
- Classe (lago; floresta; riacho; e assim por diante)
- Estado
- Latitude (graus)
- Longitude (graus)
- Altura (em pés)

4. No prompt de comando, digite o seguinte comando:

```
hive
```

O prompt de comando muda para: `hive>`

5. Insira a seguinte instrução HiveQL para criar uma tabela nativa do Hive:

```
CREATE TABLE hive_features
  (feature_id          BIGINT,
   feature_name        STRING ,
   feature_class       STRING ,
   state_alpha         STRING,
   prim_lat_dec       DOUBLE ,
   prim_long_dec      DOUBLE ,
   elev_in_ft          BIGINT)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '|'
  LINES TERMINATED BY '\n';
```

6. Insira a seguinte instrução HiveQL para carregar a tabela com dados:

```
LOAD DATA
LOCAL
INPATH './features.txt'
OVERWRITE
INTO TABLE hive_features;
```

7. Você agora tem uma tabela nativa do Hive preenchida com dados do `features.txt` file. Para verificar, insira a seguinte instrução HiveQL:

```
SELECT state_alpha, COUNT(*)
FROM hive_features
```

```
GROUP BY state_alpha;
```

A saída deve mostrar uma lista de estados e o número de recursos geográficos em cada uma delas.

Próxima etapa

[Etapa 5: Copiar dados para o DynamoDB \(p. 1026\)](#)

Etapa 5: Copiar dados para o DynamoDB

Nesta etapa, você poderá copiar dados na tabela do Hive (`hive_features`) para uma nova tabela no DynamoDB.

1. Abra o console do DynamoDB em<https://console.aws.amazon.com/dynamodb/>.
2. Selecione Create Table (Criar tabela).
3. Na página Create DynamoDB table, faça o seguinte:
 - a. DentroTabela do, digite**Features**.
 - b. para oChave primária, noChave de partiçãofield, digite**ID**. Defina o tipo de dados como Number (Número).

ClearUse Default Settings. Para Provisioned Capacity, digite o seguinte:

- Unidades de capacidade de leitura—10
- Unidades de capacidade de gravação—10

Escolha Create (Criar).

4. No prompt do Hive, insira a seguinte instrução HiveQL:

```
CREATE EXTERNAL TABLE ddb_features
  (feature_id      BIGINT,
  feature_name    STRING,
  feature_class   STRING,
  state_alpha     STRING,
  prim_lat_dec   DOUBLE,
  prim_long_dec  DOUBLE,
  elev_in_ft      BIGINT)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES(
  "dynamodb.table.name" = "Features",
  "dynamodb.column.mapping"="feature_id:Id,feature_name:Name,feature_class:Class,state_alpha:State,p");

```

Você agora estabeleceu um mapeamento entre o Hive e a tabela Recursos no DynamoDB.

5. Insira a seguinte instrução HiveQL para importar dados para o DynamoDB:

```
INSERT OVERWRITE TABLE ddb_features
SELECT
  feature_id,
  feature_name,
  feature_class,
  state_alpha,
  prim_lat_dec,
  prim_long_dec,
  elev_in_ft
```

```
FROM hive_features;
```

O Hive enviará um trabalho do MapReduce, que será processado por seu cluster do Amazon EMR. Levará vários minutos para o trabalho ser concluído.

6. Verifique se os dados foram carregados para o DynamoDB:
 - a. No painel de navegação do console do DynamoDB, escolha Tabelas.
 - b. Escolha a tabela Recursos e, em seguida, escolha a guia Items para visualizar os dados.

Próxima etapa

[Etapa 6: Consulte os dados na tabela do DynamoDB \(p. 1027\)](#)

Etapa 6: Consulte os dados na tabela do DynamoDB

Nesta etapa, você usará o HiveQL para consultar a tabela Recursos no DynamoDB. Experimente as seguintes consultas Hive:

1. Todos os tipos de recursos (`feature_class`) em ordem alfabética:

```
SELECT DISTINCT feature_class
FROM ddb_features
ORDER BY feature_class;
```

2. Todos os lagos que começam com a letra "M":

```
SELECT feature_name, state_alpha
FROM ddb_features
WHERE feature_class = 'Lake'
AND feature_name LIKE 'M%'
ORDER BY feature_name;
```

3. Estados com pelo menos três recursos com mais de uma milha (1,61 km) (5.280 pés, 1.609 m):

```
SELECT state_alpha, feature_class, COUNT(*)
FROM ddb_features
WHERE elev_in_ft > 5280
GROUP by state_alpha, feature_class
HAVING COUNT(*) >= 3
ORDER BY state_alpha, feature_class;
```

Próxima etapa

[Etapa 7: Limpar \(opcional\) \(p. 1027\)](#)

Etapa 7: Limpar (opcional)

Agora que concluiu o tutorial, você pode continuar a ler esta seção para saber mais sobre como trabalhar com dados do DynamoDB no Amazon EMR. Você pode optar por manter seu cluster do Amazon EMR em execução ao mesmo tempo em que faz isso.

Se não precisar mais do cluster, você deverá encerrá-lo e remover quaisquer recursos associados. Isso ajudará você a não ser cobrado por recursos de que você não precisa.

1. Encerrar o cluster do Amazon EMR:
 - a. Abra o console do Amazon EMR em<https://console.aws.amazon.com/elasticmapreduce/>.

- b. Escolha o cluster do Amazon EMR, escolha Encerrare confirme.
2. Excluir a tabela Recursos no DynamoDB:
 - a. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
 - b. No painel de navegação, selecione Tables (Tabelas).
 - c. Escolha a tabela Recursos. No menu Actions, escolha Delete Table.
3. Exclua o bucket do Amazon S3 que contém os arquivos de log do Amazon EMR:
 - a. Abra o console do Amazon S3 em <https://console.aws.amazon.com/s3>.
 - b. Na lista de buckets, escolha aws-logs- *accountID-region*, em que *accountID* é seu AWS Número da conta e *região* é a região onde você iniciou o cluster.
 - c. No menu Action, escolha Delete.

Criação de uma tabela externa no Hive

Dentro [Tutorial: Trabalho com o Amazon DynamoDB e o Apache Hive \(p. 1022\)](#), você criou uma tabela externa do Hive mapeada para uma tabela do DynamoDB. Quando você emitiu instruções HiveQL na tabela externa, as operações de leitura e gravação foram repassadas para a tabela do DynamoDB.

Você pode pensar em uma tabela externa como um ponteiro para uma fonte de dados que é gerenciada e armazenada em outro lugar. Nesse caso, a fonte de dados subjacente é uma tabela do DynamoDB. (A tabela já deve existir. Não é possível criar, atualizar ou excluir uma tabela do DynamoDB de dentro do Hive.) Você pode usar o `CREATE EXTERNAL TABLE` para criar a tabela externa. Depois disso, você pode usar HiveQL para trabalhar com dados no DynamoDB, como se esses dados estivessem armazenados localmente no Hive.

Note

Você pode usar o `INSERT` instruções para inserir dados em uma tabela externa e `SELECT` instruções para selecionar dados a partir dele. No entanto, você não pode usar as instruções `UPDATE` ou `DELETE` para manipular dados na tabela.

Se você não precisa mais da tabela externa, remova-a usando `DROP TABLE` instrução. Nesse caso, `DROP TABLE` remove apenas a tabela externa no Hive. Ele não afeta a tabela subjacente do DynamoDB ou qualquer um dos seus dados.

Tópicos

- [Sintaxe CREATE EXTERNAL TABLE \(p. 1028\)](#)
- [Mapeamentos de tipo de dados \(p. 1029\)](#)

Sintaxe CREATE EXTERNAL TABLE

As considerações a seguir mostram a sintaxe HiveQL para criar uma tabela externa do Hive mapeada para uma tabela do DynamoDB:

```
CREATE EXTERNAL TABLE hive_table
  (hive_column1_name hive_column1_datatype, hive_column2_name hive_column2_datatype...)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES (
  "dynamodb.table.name" = "dynamodb_table",
  "dynamodb.column.mapping" =
  "hive_column1_name:dynamodb_attribute1_name,hive_column2_name:dynamodb_attribute2_name..."
```

A linha 1 é o início da instrução `CREATE EXTERNAL TABLE`, na qual você fornece o nome da tabela do Hive (`hive_table`) que você deseja criar.

A linha 2 especifica as colunas e os tipos de dados `dohive_table`. Você precisa definir colunas e tipos de dados que correspondem aos atributos da tabela do DynamoDB.

A linha 3 é `ASTORED BY`, na qual você especifica uma classe que lida com o gerenciamento de dados entre o Hive e a tabela do DynamoDB. No DynamoDB, `STORED BY` deve ser definido como '`org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler`'.

A linha 4 é o início da cláusula `TBLPROPERTIES`, na qual você define os parâmetros a seguir para `DynamoDBStorageHandler`:

- `dynamodb.table.name`— o nome da tabela do DynamoDB.
- `dynamodb.column.mapping`- pares de nomes de coluna na tabela do Hive e seus atributos correspondentes na tabela do DynamoDB. Cada par tem o formato `hive_column_name: dynamodb_attribute_name` e os pares são separados por vírgulas.

Observe o seguinte:

- O nome da tabela do Hive não precisa ser o mesmo nome da tabela do DynamoDB.
- Os nomes de coluna da tabela do Hive não precisam ser os mesmos que os da tabela do DynamoDB.
- A tabela especificada pelo `dynamodb.table.name` deve existir no DynamoDB.
- Para `dynamodb.column.mapping`:
 - Você deve mapear os atributos de esquema de chaves da tabela do DynamoDB. Isso inclui a chave de partição e a chave de classificação (se houver).
 - Não é necessário mapear os atributos que não são chave da tabela do DynamoDB. No entanto, você não verá os dados desses atributos ao consultar a tabela do Hive.
 - Se os tipos de dados de uma coluna de tabela do Hive e um atributo do DynamoDB forem incompatíveis, você verá `NULL` nessas colunas ao consultar a tabela do Hive.

Note

O `CREATE EXTERNAL TABLE` não executa qualquer validação `NOTBLPROPERTIES` Cláusula. Os valores fornecidos para `dynamodb.table.name` e `dynamodb.column.mapping` são avaliados apenas pela classe `DynamoDBStorageHandler` quando você tenta acessar a tabela.

Mapeamentos de tipo de dados

A tabela a seguir mostra os tipos de dados do DynamoDB e os compatíveis do Hive:

Tipo de dados DynamoDB	Tipo de dados do Hive
String	STRING
telefone	BIGINT OU DOUBLE
Binário	BINARY
String Set	ARRAY<STRING>
Number Set	ARRAY<BIGINT> OU ARRAY<DOUBLE>
Binary Set	ARRAY<BINARY>

Note

Os tipos de dados do DynamoDB a seguir não são compatíveis com `ODynamoDBStorageHandler`, portanto, eles não podem ser usados com `dynamodb.column.mapping`:

- Mapa
- Lista
- Booleano
- Nulo

Caso deseje mapear um atributo DynamoDB do tipo Number, você deve escolher um tipo apropriado do Hive:

- A Hive `BIGINT` tipo é para inteiros assinados de 8 bytes. Ele é o mesmo que o tipo de dados `long` em Java.
- A Hive `DOUBLE` tipo é para números de ponto flutuante de dupla precisão de 8 bits. Ele é o mesmo que o tipo `double` em Java.

Se você tiver dados numéricos armazenados no DynamoDB que tenham uma maior precisão do que o tipo de dados do Hive que você escolheu, então, acessar os dados do DynamoDB poderia causar uma perda de precisão.

Se você exportar dados do tipo Binary do DynamoDB para (Amazon S3) ou HDFS, os dados serão armazenados como uma string codificada por Base64. Caso importe dados do Amazon S3 ou do HDFS para o tipo Binary do DynamoDB, você deverá garantir que os dados sejam codificados como uma string Base64.

Processamento de instruções HiveQL

O Hive é um aplicativo que é executado no Hadoop, que é um framework orientado a lote para a execução de trabalhos do MapReduce. Quando você emite uma instrução HiveQL, o Hive determina se ele pode retornar os resultados imediatamente ou se ele deve enviar um trabalho do MapReduce.

Por exemplo, considere `adbb_features` Tabela (do [Tutorial: Trabalho com o Amazon DynamoDB e o Apache Hive \(p. 1022\)](#)). A consulta do Hive a seguir imprime as abreviações de estado e o número de conferências em cada:

```
SELECT state_alpha, count(*)
FROM ddb_features
WHERE feature_class = 'Summit'
GROUP BY state_alpha;
```

O Hive não retorna os resultados imediatamente. Em vez disso, ele envia um trabalho do MapReduce, que é processado pelo framework do Hadoop. O Hive aguardará até que o trabalho seja concluído antes de mostrar os resultados da consulta:

```
AK 2
AL 2
AR 2
AZ 3
CA 7
CO 2
CT 2
ID 1
KS 1
```

```
ME 2
MI 1
MT 3
NC 1
NE 1
NM 1
NY 2
OR 5
PA 1
TN 1
TX 1
UT 4
VA 1
VT 2
WA 2
WY 3
Time taken: 8.753 seconds, Fetched: 25 row(s)
```

Monitoramento e cancelamento de trabalhos

Quando o Hive executa um trabalho do Hadoop, ele imprime a saída desse trabalho. O status da conclusão do trabalho é atualizado conforme o trabalho está progredindo. Em alguns casos, o status pode não ser atualizado por um longo período. (Isso pode acontecer quando você está consultando uma tabela grande do DynamoDB que tem uma baixa capacidade de leitura provisionada configurada.)

Caso precise cancelar o trabalho antes que ele esteja concluído, digite **ctrl+c** em qualquer momento.

Consulta de dados no DynamoDB

Os exemplos a seguir mostram algumas maneiras em que você pode usar o HiveQL para consultar os dados armazenados no DynamoDB.

Estes exemplos referem-se ao db_features no tutorial ([Etapa 5: Copiar dados para o DynamoDB \(p. 1026\)](#)).

Tópicos

- [Uso de funções agregadas \(p. 1031\)](#)
- [Uso de cláusulas GROUP BY e HAVING \(p. 1031\)](#)
- [Junção de duas tabelas do DynamoDB \(p. 1032\)](#)
- [Junção de tabelas de origens diferentes \(p. 1032\)](#)

Uso de funções agregadas

O HiveQL fornece funções internas para resumir valores de dados. Por exemplo, você pode usar `MAX` para encontrar o maior valor para uma coluna selecionada. O exemplo a seguir retorna a elevação do recurso mais elevado no estado do Colorado.

```
SELECT MAX(elev_in_ft)
FROM ddb_features
WHERE state_alpha = 'CO';
```

Uso de cláusulas GROUP BY e HAVING

Você pode usar a cláusula `GROUP BY` para coletar dados em vários registros. Isso é muitas vezes usado com uma função agregada, como `SUM`, `COUNT`, `MIN`, ou `MAX`. Você também pode usar a cláusula `HAVING` para descartar os resultados que não atendem a determinados critérios.

O exemplo a seguir retorna uma lista das elevações mais altas dos estados que têm mais de cinco recursos na tabela `ddb_features`.

```
SELECT state_alpha, max(elev_in_ft)
FROM ddb_features
GROUP BY state_alpha
HAVING count(*) >= 5;
```

Junção de duas tabelas do DynamoDB

O exemplo a seguir mapeia outra tabela do Hive (`east_coast_states`) para uma tabela no DynamoDB. O SELECT é uma junção entre essas duas tabelas. A junção é calculada no cluster e retornada. A junção não ocorre no DynamoDB.

Considere uma tabela do DynamoDB chamada `EastCoastStates` que contenha os seguintes dados:

StateName	StateAbbrev
Maine	ME
New Hampshire	NH
Massachusetts	MA
Rhode Island	RI
Connecticut	CT
New York	NY
New Jersey	NJ
Delaware	DE
Maryland	MD
Virginia	VA
North Carolina	NC
South Carolina	SC
Georgia	GA
Florida	FL

Vamos supor que a tabela esteja disponível como uma tabela externa do Hive chamada `east_coast_states`:

```
CREATE EXTERNAL TABLE ddb_east_coast_states (state_name STRING, state_alpha STRING)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "EastCoastStates",
"YNAMOdb.column.mapping" = "state_name:StateName,state_alpha:StateAbbrev");
```

A junção a seguir retorna os estados na Costa Leste dos Estados Unidos que têm pelo menos três recursos:

```
SELECT ecs.state_name, f.feature_class, COUNT(*)
FROM ddb_east_coast_states ecs
JOIN ddb_features f ON ecs.state_alpha = f.state_alpha
GROUP BY ecs.state_name, f.feature_class
HAVING COUNT(*) >= 3;
```

Junção de tabelas de origens diferentes

No exemplo a seguir, `s3_east_coast_states` é uma tabela do Hive associada a um arquivo CSV armazenado no Amazon S3. `ddb_features` é uma tabela associada aos dados no DynamoDB. O exemplo a seguir junta essas duas tabelas, retornando os recursos geográficos de estados cujos nomes começam com "Nova".

```
create external table s3_east_coast_states (state_name STRING, state_alpha STRING)
```

```
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  
LOCATION 's3://bucketname/path/subpath/';
```

```
SELECT ecs.state_name, f.feature_name, f.feature_class  
FROM s3_east_coast_states ecs  
JOIN ddb_features f  
ON ecs.state_alpha = f.state_alpha  
WHERE ecs.state_name LIKE 'New%';
```

Cópia de dados para e do Amazon DynamoDB

NoTutorial: Trabalho com o Amazon DynamoDB e o Apache Hive (p. 1022), você copiou os dados de uma tabela nativa do Hive para uma tabela externa do DynamoDB e, em seguida, consultou a tabela externa do DynamoDB. A tabela é externa porque existe fora do Hive. Mesmo se você descartar a tabela do Hive mapeada para o DynamoDB, a tabela do DynamoDB não será afetada.

O Hive é uma excelente solução para copiar dados entre tabelas do DynamoDB, buckets do Amazon S3, tabelas nativas do Hive e o Hadoop Distributed File System (HDFS). Esta seção fornece exemplos dessas operações.

Tópicos

- [Cópia de dados entre o DynamoDB e uma tabela nativa do Hive \(p. 1033\)](#)
- [Copiando dados entre o DynamoDB e o Amazon S3 \(p. 1034\)](#)
- [Cópia de dados entre o DynamoDB e o HDFS \(p. 1038\)](#)
- [Uso de compactação de dados \(p. 1042\)](#)
- [Leitura de dados de caractere UTF-8 não imprimíveis \(p. 1042\)](#)

Cópia de dados entre o DynamoDB e uma tabela nativa do Hive

Caso tenha dados em uma tabela do DynamoDB, você pode copiar os dados para uma tabela nativa do Hive. Isso gerará um snapshot dos dados, a partir do momento em que você os copiou.

Você pode optar por fazer isso caso precise executar muitas consultas HiveQL, mas não deseja consumir a capacidade de throughput provisionado do DynamoDB. Como os dados da tabela nativa do Hive são uma cópia dos dados do DynamoDB, e não dados “ativos”, suas consultas não devem esperar que os dados sejam atualizados.

Note

Os exemplos desta seção são gravados com a suposição de que você seguiu as etapas deTutorial: Trabalho com o Amazon DynamoDB e o Apache Hive (p. 1022)e ter uma tabela externa no DynamoDB chamadddb_features.

Example Do DynamoDB para a tabela nativa do Hive

Você pode criar uma tabela nativa do Hive e preenche-la com dados de ddb_features, desta forma:

```
CREATE TABLE features_snapshot AS  
SELECT * FROM ddb_features;
```

Em seguida, você pode atualizar os dados a qualquer momento:

```
INSERT OVERWRITE TABLE features_snapshot
```

```
SELECT * FROM ddb_features;
```

Nestes exemplos, a subconsulta `SELECT * FROM ddb_features` Recuperará todos os dados da `ddb_features`. Se quiser copiar apenas um subconjunto dos dados, você pode usar uma cláusula `WHERE` na subconsulta.

O exemplo a seguir cria uma tabela nativa do Hive, contendo apenas alguns dos atributos para lagos e picos:

```
CREATE TABLE lakes_and_summits AS
SELECT feature_name, feature_class, state_alpha
FROM ddb_features
WHERE feature_class IN ('Lake', 'Summit');
```

Example Da tabela nativa do Hive para o DynamoDB

Use a instrução HiveQL a seguir para copiar os dados da tabela nativa do Hive para `ddb_features`:

```
INSERT OVERWRITE TABLE ddb_features
SELECT * FROM features_snapshot;
```

Copiando dados entre o DynamoDB e o Amazon S3

Caso tenha dados em uma tabela do DynamoDB, você poderá usar o Hive para copiar os dados para um bucket do Amazon S3.

Você pode fazer isso se quiser criar um arquivamento de dados na sua tabela do DynamoDB. Por exemplo, suponha que você tenha um ambiente de teste no qual precisa trabalhar com um conjunto de dados de teste de linha de base no DynamoDB. Você pode copiar os dados da linha de base para um bucket do Amazon S3 e, em seguida, executar os testes. Depois disso, você pode redefinir o ambiente de teste, restaurando os dados de linha de dados do bucket do Amazon S3 para o DynamoDB.

Se você trabalhou no [Tutorial: Trabalho com o Amazon DynamoDB e o Apache Hive \(p. 1022\)](#) Você já tem um bucket do Amazon S3 que contém seus logs do Amazon EMR. Você pode usar esse bucket para os exemplos desta seção, se souber o caminho raiz do bucket:

1. Abra o console do Amazon EMR em <https://console.aws.amazon.com/elasticmapreduce/>.
2. Para Name, escolha o seu cluster.
3. O URI é listado no Log URI em Configuration Details.
4. Anote o caminho raiz do bucket. A convenção de nomenclatura é:

`s3://aws-logs-accountID-region`

where `accountID` É seu AWS ID da conta e região é o AWS Region para o bucket.

Note

Para esses exemplos, usaremos um subcaminho no bucket, como neste exemplo:
`s3://aws-logs-123456789012-us-west-2/hive-test`

Os procedimentos a seguir são gravados com a suposição de que você seguiu as etapas do tutorial e tem uma tabela externa no DynamoDB chamada `ddb_features`.

Tópicos

- [Cópia de dados usando o formato padrão do Hive \(p. 1035\)](#)

- [Cópia de dados com um formato especificado pelo usuário \(p. 1035\)](#)
- [Cópia de dados sem um mapeamento de coluna \(p. 1036\)](#)
- [Exibição dos dados no Amazon S3 \(p. 1037\)](#)

Cópia de dados usando o formato padrão do Hive

Example Do DynamoDB para o Amazon S3

Usar uma `INSERT OVERWRITE` para gravar diretamente para o Amazon S3.

```
INSERT OVERWRITE DIRECTORY 's3://aws-logs-123456789012-us-west-2/hive-test'  
SELECT * FROM ddb_features;
```

O arquivo de dados no Amazon S3 é semelhante a:

```
920709^ASoldiers Farewell Hill^ASummit^ANM^A32.3564729^A-108.33004616135  
1178153^AJones Run^AStream^APA^A41.2120086^A-79.25920781260  
253838^ASentinel Dome^ASummit^ACA^A37.7229821^A-119.584338133  
264054^ANeversweet Gulch^AValley^ACA^A41.6565269^A-122.83614322900  
115905^AChacaloochee Bay^ABay^AAL^A30.6979676^A-87.97388530
```

Cada campo é separado por um caractere SOH (início do cabeçalho, 0x01). No arquivo, SOH aparece como ^A.

Example Do Amazon S3 para o DynamoDB

1. Crie uma tabela externa apontando para os dados não formatados no Amazon S3.

```
CREATE EXTERNAL TABLE s3_features_unformatted  
(feature_id          BIGINT,  
 feature_name         STRING ,  
 feature_class        STRING ,  
 state_alpha          STRING,  
 prim_lat_dec         DOUBLE ,  
 prim_long_dec        DOUBLE ,  
 elev_in_ft           BIGINT)  
LOCATION 's3://aws-logs-123456789012-us-west-2/hive-test';
```

2. Copie os dados para o DynamoDB.

```
INSERT OVERWRITE TABLE ddb_features  
SELECT * FROM s3_features_unformatted;
```

Cópia de dados com um formato especificado pelo usuário

Se você deseja especificar seu próprio caractere separador de campos, crie uma tabela externa mapeada para o bucket do Amazon S3. Você pode usar essa técnica para a criação de arquivos de dados com valores separados por vírgulas (CSV).

Example Do DynamoDB para o Amazon S3

1. Crie uma tabela externa do Hive mapeada para o Amazon S3. Ao fazer isso, certifique-se de que os tipos de dados estejam consistentes com os da tabela externa do DynamoDB.

```
CREATE EXTERNAL TABLE s3_features_csv
```

```
(feature_id      BIGINT,  
feature_name    STRING,  
feature_class   STRING,  
state_alpha     STRING,  
prim_lat_dec   DOUBLE,  
prim_long_dec  DOUBLE,  
elev_in_ft     BIGINT)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ','  
LOCATION 's3://aws-logs-123456789012-us-west-2/hive-test';
```

2. Copie os dados do DynamoDB.

```
INSERT OVERWRITE TABLE s3_features_csv  
SELECT * FROM ddb_features;
```

O arquivo de dados no Amazon S3 é semelhante a:

```
920709,Soldiers Farewell Hill,Summit,NM,32.3564729,-108.3300461,6135  
1178153,Jones Run,Stream,PA,41.2120086,-79.2592078,1260  
253838,Sentinel Dome,Summit,CA,37.7229821,-119.58433,8133  
264054,Neversweet Gulch,Valley,CA,41.6565269,-122.8361432,2900  
115905,Chacaloochee Bay,Bay,AL,30.6979676,-87.9738853,0
```

Example Do Amazon S3 para o DynamoDB

Com uma única instrução HiveQL, você pode preencher a tabela do DynamoDB usando os dados do Amazon S3:

```
INSERT OVERWRITE TABLE ddb_features  
SELECT * FROM s3_features_csv;
```

Cópia de dados sem um mapeamento de coluna

Você pode copiar dados do DynamoDB em um formato bruto e gravá-los no Amazon S3, sem especificar quaisquer tipos de dados ou mapeamento de coluna. Você pode usar esse método para criar um arquivamento de dados do DynamoDB e armazená-lo no Amazon S3.

Note

Se a sua tabela do DynamoDB contém atributos do tipo Map, List, Boolean ou Null, essa é a única forma que você poder usar o Hive para copiar dados do DynamoDB para o Amazon S3.

Example Do DynamoDB para o Amazon S3

1. Crie uma tabela externa associada à sua tabela do DynamoDB. (Não há dynamodb.column.mapping nessa instrução HiveQL.)

```
CREATE EXTERNAL TABLE ddb_features_no_mapping  
(item MAP<STRING, STRING>)  
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'  
TBLPROPERTIES ("dynamodb.table.name" = "Features");
```

2. Crie outra tabela externa associada ao seu bucket do Amazon S3.

```
CREATE EXTERNAL TABLE s3_features_no_mapping
```

```
(item MAP<STRING, STRING>)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
LOCATION 's3://aws-logs-123456789012-us-west-2/hive-test';
```

- Copie os dados do DynamoDB para o Amazon S3.

```
INSERT OVERWRITE TABLE s3_features_no_mapping
SELECT * FROM ddb_features_no_mapping;
```

O arquivo de dados no Amazon S3 é semelhante a:

```
Name^C{"s":"Soldiers Farewell
Hill"}^BState^C{"s":"NM"}^BClass^C{"s":"Summit"}^BElevation^C{"n":6135"}^BLatitude^C{"n":32.3564729"}^BLongitude^C{"n":-106.7851375"}^BId^C{"n":1}
Name^C{"s":"Jones
Run"}^BState^C{"s":"PA"}^BClass^C{"s":"Stream"}^BElevation^C{"n":1260"}^BLatitude^C{"n":41.2120086"}^BLongitude^C{"n":-75.1075471"}^BId^C{"n":2}
Name^C{"s":"Sentinel
Dome"}^BState^C{"s":"CA"}^BClass^C{"s":"Summit"}^BElevation^C{"n":8133"}^BLatitude^C{"n":37.7229821"}^BLongitude^C{"n":-122.1784328"}^BId^C{"n":3}
Name^C{"s":"Neversweet
Gulch"}^BState^C{"s":"CA"}^BClass^C{"s":"Valley"}^BElevation^C{"n":2900"}^BLatitude^C{"n":41.6565269"}^BLongitude^C{"n":-122.1784328"}^BId^C{"n":4}
Name^C{"s":"Chacaloochee
Bay"}^BState^C{"s":"AL"}^BClass^C{"s":"Bay"}^BElevation^C{"n":0"}^BLatitude^C{"n":30.6979676"}^BLongitude^C{"n":-86.7829591"}^BId^C{"n":5}
```

Cada campo começa com um caractere STX (início do texto, 0x02) e termina com um caractere ETX (fim do texto, 0x03). No arquivo, STX aparece como^Be ETX aparece como^C.

Example Do Amazon S3 para o DynamoDB

Com uma única instrução HiveQL, você pode preencher a tabela do DynamoDB usando os dados do Amazon S3:

```
INSERT OVERWRITE TABLE ddb_features_no_mapping
SELECT * FROM s3_features_no_mapping;
```

Exibição dos dados no Amazon S3

Caso use o SSH para se conectar ao nó principal, você poderá usar o AWS Command Line Interface(AWS CLI) para acessar os dados que o Hive gravou no Amazon S3.

As etapas a seguir são escritas com a suposição de que você copiou os dados do DynamoDB para o Amazon S3; usando um dos procedimentos desta seção.

- Se você estiver no momento no prompt de comando do Hive, saia do prompt de comando do Linux.

```
hive> exit;
```

- Liste o conteúdo do diretório hive-test em seu bucket do Amazon S3. (Esse é o local para o qual o Hive copiou os dados do DynamoDB.)

```
aws s3 ls s3://aws-logs-123456789012-us-west-2/hive-test/
```

A resposta deve ter a seguinte aparência:

```
2016-11-01 23:19:54 81983 000000_0
```

O nome do arquivo (000000_0) é gerado pelo sistema.

3. (Opcional) Você pode copiar o arquivo de dados do Amazon S3 para o sistema de arquivos local no nó principal. Depois de fazer isso, você pode usar os utilitários de linha de comando padrão do Linux para trabalhar com os dados no arquivo.

```
aws s3 cp s3://aws-logs-123456789012-us-west-2/hive-test/000000_0 .
```

A resposta deve ter a seguinte aparência:

```
download: s3://aws-logs-123456789012-us-west-2/hive-test/000000_0  
to ./000000_0
```

Note

O sistema de arquivos local no nó principal tem capacidade limitada. Não use este comando com arquivos maiores do que o espaço disponível no sistema de arquivos local.

Cópia de dados entre o DynamoDB e o HDFS

Caso tenha dados em uma tabela do DynamoDB, você poderá usar o Hive para copiar os dados para o Hadoop Distributed File System (HDFS).

Você pode fazer isso se estiver executando um trabalho do MapReduce que exige dados do DynamoDB. Se você copiar os dados do DynamoDB para o HDFS, o Hadoop poderá processá-los, usando todos os nós disponíveis no cluster do Amazon EMR em paralelo. Quando o trabalho do MapReduce for concluído, você poderá gravar os resultados do HDFS no DDB.

Nos exemplos a seguir, o Hive irá ler e gravar no diretório HDFS a seguir: /user/hadoop/hive-test

Note

Os exemplos desta seção são gravados com a suposição de que você seguiu as etapas de [Tutorial: Trabalho com o Amazon DynamoDB e o Apache Hive \(p. 1022\)](#) e você tem uma tabela externa no DynamoDB chamada ddb_features.

Tópicos

- [Cópia de dados usando o formato padrão do Hive \(p. 1038\)](#)
- [Cópia de dados com um formato especificado pelo usuário \(p. 1039\)](#)
- [Cópia de dados sem um mapeamento de coluna \(p. 1040\)](#)
- [Acesso aos dados no HDFS \(p. 1041\)](#)

Cópia de dados usando o formato padrão do Hive

Example Do DynamoDB para o HDFS

Use uma instrução `INSERT OVERWRITE DIRECTORY` para gravar diretamente para o HDFS.

```
INSERT OVERWRITE DIRECTORY 'hdfs://user/hadoop/hive-test'  
SELECT * FROM ddb_features;
```

O arquivo de dados do HDFS é semelhante a:

```
920709^ASoldiers Farewell Hill^ASummit^ANM^A32.3564729^A-108.33004616135  
1178153^AJones Run^ASTream^APA^A41.2120086^A-79.25920781260  
253838^ASentinel Dome^ASummit^ACA^A37.7229821^A-119.584338133
```

```
264054^ANeversweet Gulch^AValley^ACA^A41.6565269^A-122.83614322900
115905^AChacaloochee Bay^ABay^AAL^A30.6979676^A-87.97388530
```

Cada campo é separado por um caractere SOH (início do cabeçalho, 0x01). No arquivo, SOH aparece como^A.

Example Do HDFS para o DynamoDB

1. Crie uma tabela externa mapeada para os dados não formatados no HDFS.

```
CREATE EXTERNAL TABLE hdfs_features_unformatted
    (feature_id      BIGINT,
     feature_name    STRING ,
     feature_class   STRING ,
     state_alpha     STRING,
     prim_lat_dec   DOUBLE ,
     prim_long_dec  DOUBLE ,
     elev_in_ft      BIGINT)
LOCATION 'hdfs:///user/hadoop/hive-test';
```

2. Copie os dados para o DynamoDB.

```
INSERT OVERWRITE TABLE ddb_features
SELECT * FROM hdfs_features_unformatted;
```

Cópia de dados com um formato especificado pelo usuário

Caso queira usar um caractere separador de campos diferente, você poderá criar uma tabela externa mapeada para o diretório HDFS. Você pode usar essa técnica para a criação de arquivos de dados com valores separados por vírgulas (CSV).

Example Do DynamoDB para o HDFS

1. Crie uma tabela externa do Hive mapeada para o HDFS. Ao fazer isso, certifique-se de que os tipos de dados estejam consistentes com os da tabela externa do DynamoDB.

```
CREATE EXTERNAL TABLE hdfs_features_csv
    (feature_id      BIGINT,
     feature_name    STRING ,
     feature_class   STRING ,
     state_alpha     STRING,
     prim_lat_dec   DOUBLE ,
     prim_long_dec  DOUBLE ,
     elev_in_ft      BIGINT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION 'hdfs:///user/hadoop/hive-test';
```

2. Copie os dados do DynamoDB.

```
INSERT OVERWRITE TABLE hdfs_features_csv
SELECT * FROM ddb_features;
```

O arquivo de dados do HDFS é semelhante a:

```
920709,Soldiers Farewell Hill,Summit,NM,32.3564729,-108.3300461,6135
1178153,Jones Run,Stream,PA,41.2120086,-79.2592078,1260
```

```
253838, Sentinel Dome, Summit, CA, 37.7229821, -119.58433, 8133  
264054, Neversweet Gulch, Valley, CA, 41.6565269, -122.8361432, 2900  
115905, Chacaloochee Bay, Bay, AL, 30.6979676, -87.9738853, 0
```

Example Do HDFS para o DynamoDB

Com uma única instrução HiveQL, você pode preencher a tabela do DynamoDB usando os dados do HDFS:

```
INSERT OVERWRITE TABLE ddb_features  
SELECT * FROM hdfs_features_csv;
```

Cópia de dados sem um mapeamento de coluna

Você pode copiar dados do DynamoDB em um formato bruto e gravá-los no HDFS, sem especificar quaisquer tipos de dados ou mapeamento de coluna. Você pode usar esse método para criar um arquivamento de dados do DynamoDB e armazená-lo no HDFS.

Note

Se a sua tabela do DynamoDB contém atributos do tipo Map, List, Boolean ou Null, essa é a única forma que você poder usar o Hive para copiar dados do DynamoDB para o HDFS.

Example Do DynamoDB para o HDFS

1. Crie uma tabela externa associada à sua tabela do DynamoDB. (Não há dynamodb.column.mapping nessa instrução HiveQL.)

```
CREATE EXTERNAL TABLE ddb_features_no_mapping  
(item MAP<STRING, STRING>)  
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'  
TBLPROPERTIES ("dynamodb.table.name" = "Features");
```

2. Crie outra tabela externa associada ao seu diretório HDFS.

```
CREATE EXTERNAL TABLE hdfs_features_no_mapping  
(item MAP<STRING, STRING>)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '\t'  
LINES TERMINATED BY '\n'  
LOCATION 'hdfs://user/hadoop/hive-test';
```

3. Copie os dados do DynamoDB para o HDFS.

```
INSERT OVERWRITE TABLE hdfs_features_no_mapping  
SELECT * FROM ddb_features_no_mapping;
```

O arquivo de dados do HDFS é semelhante a:

```
Name^C{"s":"Soldiers Farewell  
Hill"}^BState^C{"s":"NM"}^BClass^C{"s":"Summit"}^BElevation^C{"n":6135}^BLatitude^C{"n":32.3564729}  
Name^C{"s":"Jones  
Run"}^BState^C{"s":PA}^BClass^C{"s":Stream"}^BElevation^C{"n":1260}^BLatitude^C{"n":41.2120086}  
Name^C{"s":"Sentinel  
Dome"}^BState^C{"s":CA}^BClass^C{"s":Summit"}^BElevation^C{"n":8133}^BLatitude^C{"n":37.7229821}
```

```
Name^C{"s": "Neversweet
Gulch"}^BState^C{"s": "CA"}^BClass^C{"s": "Valley"}^BElevation^C{"n": "2900"}^BLatitude^C{"n": "41.6565269
Name^C{"s": "Chacaloochee
Bay"}^BState^C{"s": "AL"}^BClass^C{"s": "Bay"}^BElevation^C{"n": "0"}^BLatitude^C{"n": "30.6979676"}^BId^C
```

Cada campo começa com um caractere STX (início do texto, 0x02) e termina com um caractere ETX (fim do texto, 0x03). No arquivo, STX aparece como ^B e ETX aparece como ^C.

Example Do HDFS para o DynamoDB

Com uma única instrução HiveQL, você pode preencher a tabela do DynamoDB usando os dados do HDFS:

```
INSERT OVERWRITE TABLE ddb_features_no_mapping
SELECT * FROM hdfs_features_no_mapping;
```

Acesso aos dados no HDFS

O HDFS é um sistema de arquivos distribuído, acessível para todos os nós do cluster Amazon EMR. Caso use o SSH para se conectar ao nó principal, você poderá usar as ferramentas de linha de comando para acessar os dados que o Hive gravou no HDFS.

O HDFS não é igual ao sistema de arquivos local no nó principal. Você não pode trabalhar com arquivos e diretórios no HDFS usando os comandos padrão do Linux (como `ocat`, `cp`, `mv`, `ourm`). Em vez disso, você executa essas tarefas usando o comando `hadoop fs`.

As etapas a seguir são escritas com a suposição de que você copiou os dados do DynamoDB para o HDFS; usando um dos procedimentos desta seção.

1. Se você estiver no momento no prompt de comando do Hive, saia do prompt de comando do Linux.

```
hive> exit;
```

2. Liste o conteúdo do diretório `/user/hadoop/hive-test` no HDFS. (Esse é o local para o qual o Hive copiou os dados do DynamoDB.)

```
hadoop fs -ls /user/hadoop/hive-test
```

A resposta deve ter a seguinte aparência:

```
Found 1 items
-rw-r--r-- 1 hadoop hadoop 29504 2016-06-08 23:40 /user/hadoop/hive-test/000000_0
```

O nome do arquivo (`000000_0`) é gerado pelo sistema.

3. Visualize o conteúdo do arquivo :

```
hadoop fs -cat /user/hadoop/hive-test/000000_0
```

Note

Neste exemplo, o arquivo é relativamente pequeno (aproximadamente 29 KB). Tenha cuidado ao usar este comando com arquivos que são muito grandes ou contêm caracteres não imprimíveis.

4. (Opcional) Você pode copiar o arquivo de dados do HDFS para o sistema de arquivos local no nó principal. Depois de fazer isso, você pode usar os utilitários de linha de comando padrão do Linux para trabalhar com os dados no arquivo.

```
hadoop fs -get /user/hadoop/hive-test/000000_0
```

Este comando não substituirá o arquivo.

Note

O sistema de arquivos local no nó principal tem capacidade limitada. Não use este comando com arquivos maiores do que o espaço disponível no sistema de arquivos local.

Uso de compactação de dados

Ao usar o Hive para copiar dados entre diferentes fontes de dados, você pode solicitar a compactação de dados durante o processo. O Hive fornece vários codecs de compactação. Você pode escolher um durante a sessão do Hive. Quando você faz isso, os dados são compactados no formato especificado.

O exemplo a seguir compacta os dados usando o algoritmo Lempel-Ziv-Oberhumer (LZO).

```
SET hive.exec.compress.output=true;
SET io.seqfile.compression.type=BLOCK;
SET mapred.output.compression.codec = com.hadoop.compression.lzo.LzopCodec;

CREATE EXTERNAL TABLE lzo_compression_table (line STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
LOCATION 's3://bucketname/path/subpath/';

INSERT OVERWRITE TABLE lzo_compression_table SELECT *
FROM hiveTableName;
```

O arquivo resultante no Amazon S3 terá um nome gerado pelo sistema com o .lzo no final (por exemplo, 8d436957-57ba-4af7-840c-96c2fc7bb6f5-000000.lzo).

Os codecs de compactação disponíveis são:

- org.apache.hadoop.io.compress.GzipCodec
- org.apache.hadoop.io.compress.DefaultCodec
- com.hadoop.compression.lzo.LzoCodec
- com.hadoop.compression.lzo.LzopCodec
- org.apache.hadoop.io.compress.BZip2Codec
- org.apache.hadoop.io.compress.SnappyCodec

Leitura de dados de caractere UTF-8 não imprimíveis

Para ler e gravar dados de caracteres UTF-8 não imprimíveis, você pode usar o comando `STORED AS SEQUENCEFILE` quando você cria uma tabela do Hive. Um SequenceFile é um formato de arquivo binário do Hadoop. Você precisa usar o Hadoop para ler esse arquivo. O exemplo a seguir mostra como exportar dados do DynamoDB para o Amazon S3. Você pode usar essa funcionalidade para lidar com caracteres de codificação UTF-8 não imprimíveis.

```
CREATE EXTERNAL TABLE s3_export(a_col string, b_col bigint, c_col array<string>)
STORED AS SEQUENCEFILE
LOCATION 's3://bucketname/path/subpath/';

INSERT OVERWRITE TABLE s3_export SELECT *
FROM hiveTableName;
```

Ajuste de desempenho

Ao criar uma tabela externa do Hive mapeada para uma tabela do DynamoDB, você não consome qualquer capacidade de leitura ou gravação do DynamoDB. No entanto, a atividade de leitura e gravação na tabela do Hive (como `INSERT` ou `SELECT`) se traduz diretamente em operações de leitura e gravação na tabela subjacente do DynamoDB.

O Apache Hive no Amazon EMR implementa sua próxima lógica para balancear a carga de E/S na tabela do DynamoDB e procura minimizar a possibilidade de exceder o throughput provisionado da tabela. No final de cada consulta do Hive, o Amazon EMR retorna as métricas de tempo de execução, incluindo o número de vezes em que o throughput provisionado foi excedido. Você pode usar essas informações, com as métricas do CloudWatch na sua tabela do DynamoDB, para melhorar o desempenho em solicitações subsequentes.

O console do Amazon EMR fornece ferramentas de monitoramento básicas para seu cluster. Para obter mais informações, consulte [Visualizar e monitorar um cluster](#) no Guia de gerenciamento do Amazon EMR.

É possível monitorar o cluster e trabalhos do Hadoop usando ferramentas com base na web, como Matiz, Ganglia e a interface da web do Hadoop. Para obter mais informações, consulte [Visualize interfaces Web hospedadas em clusters do Amazon EMR](#) no Guia de gerenciamento do Amazon EMR.

Esta seção descreve as etapas que você pode executar para ajustar o desempenho das operações no Hive em tabelas externas do DynamoDB.

Tópicos

- [Throughput provisionado de DynamoDB \(p. 1043\)](#)
- [Ajuste de mapeadores \(p. 1045\)](#)
- [Tópicos adicionais \(p. 1046\)](#)

Throughput provisionado de DynamoDB

Quando você emite instruções HiveQL em relação à tabela externa do DynamoDB, o `DynamoDBStorageHandler` classe faz as solicitações de API de baixo nível do DynamoDB apropriadas, que consomem throughput provisionado. Se não houver capacidade de leitura ou gravação suficiente na tabela do DynamoDB, a solicitação será limitada, o que resulta em um desempenho lento do HiveQL. Por esse motivo, você deve garantir que a tabela tenha capacidade de throughput suficiente.

Por exemplo, suponha que você tenha provisionado 100 unidades de capacidade de leitura para a sua tabela do DynamoDB. Isso permitirá que você leia 409.600 bytes por segundo (100×4 KB de tamanho de unidade de capacidade de leitura). Agora, suponha que a tabela contenha 20 GB de dados (21.474.836.480 bytes) e deseje usar o `SELECT` para selecionar todos os dados usando HiveQL. Você pode estimar quanto tempo a consulta levará para ser executada desta forma:

$$21.474.836.480 / 409.600 = 52.429 \text{ segundos} = 14,56 \text{ horas}$$

Nesse cenário, a tabela do DynamoDB é um gargalo. Ela não ajudará a adicionar mais nós do Amazon EMR, pois o throughput do Hive é limitado a apenas 409.600 bytes por segundo. A única maneira de reduzir o tempo necessário para o `SELECT` instrução é aumentar a capacidade de leitura provisionada da tabela do DynamoDB.

Você pode executar um cálculo semelhante para estimar a quantidade de tempo necessário para a carga de grandes volumes de dados em uma tabela externa do Hive mapeada para uma tabela do DynamoDB. Determine o número total de bytes nos dados que você deseja carregar e, em seguida, divida pelo tamanho de uma unidade de capacidade de gravação do DynamoDB (1 KB). Isso resultará no número de segundos necessários para carregar a tabela.

Você deve monitorar regularmente as métricas do CloudWatch da sua tabela. Para obter uma visão geral rápida no console do DynamoDB, escolha sua tabela e, em seguida, selecione o **MétricasGuia**. Aqui, você pode visualizar as unidades de capacidade de leitura e gravação consumidas e as solicitações que foram limitadas.

Capacidade de leitura

O Amazon EMR gerencia a sua tabela do DynamoDB, de acordo com as configurações de throughput provisionado. No entanto, se você notar um grande número de provisioned throughput exceeded na saída do trabalho, você poderá ajustar a taxa de leitura padrão. Para fazer isso, você pode modificar `adynamodb.throughput.read.percent` variável de configuração. Você pode usar o comando `SET` para definir essa variável no prompt de comando do Hive.

```
SET dynamodb.throughput.read.percent=1.0;
```

Esta variável persiste apenas durante a sessão atual do Hive. Se você sair do Hive e retornar mais tarde, `dynamodb.throughput.read.percent` retornará para o seu valor padrão.

O valor `dynamodb.read.percent` pode estar entre `0.1e1.5`, inclusivamente `0.5`. Representa a taxa de leitura padrão, o que significa que o Hive tentará consumir metade da capacidade de leitura da tabela. Se você aumentar o valor acima `0.5`, o Hive aumentará a taxa de solicitação; diminuindo o valor abaixo de `0.5` diminui a taxa de solicitações de leitura. (A taxa de leitura real poderá variar, dependendo de fatores como se há uma distribuição de chaves uniformes na tabela do DynamoDB.)

Se você notar que o Hive está frequentemente esgotando a capacidade de leitura provisionada da tabela, ou se suas solicitações de leitura estão sendo limitadas com muita frequência, tente reduzir `odYNAMODB.read.throughput.percent` para 0.5. Se você tiver capacidade de leitura suficiente e quiser operações HiveQL mais responsivas, defina o valor acima de 0.5.

Capacidade de gravação

O Amazon EMR gerencia a a a a a a a a a sua tabela do DynamoDB, de acordo com as configurações de throughput provisionado. No entanto, se você notar um grande número de ProvisionedThroughputExceededNa saída do trabalho, você pode ajustar a taxa de gravação padrão. Para fazer isso, você pode modificar adynamodb.throughput.write.percentVariável de configuração. Você pode usar o comando SET para definir essa variável no prompt de comando do Hive.

```
SET dynamodb.throughput.write.percent=1.0;
```

Esta variável persiste apenas durante a sessão atual do Hive. Se você sair do Hive e retornar mais tarde, `dynamodb.throughput.write.percent` retornará para o seu valor padrão.

O valor `dynamodb.throughput.write.percent` pode estar entre 0.1e1.5, inclusivamente. 0.5 representa a taxa de gravação padrão, o que significa que o Hive tentará consumir metade da capacidade de gravação da tabela. Se você aumentar o valor acima de 0.5, o Hive aumentará a taxa de solicitação; diminuindo o valor abaixo de 0.5 diminui a taxa de solicitações de gravação. (A taxa de gravação real poderá variar, dependendo de fatores como se há uma distribuição de chaves uniformes na tabela do DynamoDB.)

Se você notar que o Hive está frequentemente esgotando a capacidade de gravação provisionada da tabela, ou se suas solicitações de gravação estão sendo limitadas com muita frequência, tente reduzir `odynamodb.throughput.write.percent` para 0.5. Se você tiver capacidade suficiente na tabela e quiser operações HiveQL mais responsivas, defina o valor acima de 0.5.

Ao gravar dados no DynamoDB usando o Hive, certifique-se de que o número de unidades de capacidade de gravação seja maior do que o número de mapeadores no cluster. Por exemplo, considere um cluster do

Amazon EMR que consiste em 10m1.xlargeNó. Om1.xlargeO tipo de nó fornece 8 tarefas de mapeador, para que o cluster tenha um total de 80 mapeadores (10×8). Se a sua tabela do DynamoDB possuir menos que 80 unidades de capacidade de gravação, uma operação de gravação do Hive poderia consumir todo o throughput de gravação dessa tabela.

Para determinar o número de mapeadores dos tipos de nó do Amazon EMR, consulte[Configuração da tarefa](#)noGuia do desenvolvedor do Amazon EMR.

Para obter mais informações sobre mapeadores, consulte [Ajuste de mapeadores \(p. 1045\)](#).

Ajuste de mapeadores

Quando o Hive inicia um trabalho do Hadoop, esse trabalho é processado por uma ou mais tarefas do mapeador. Supondo que a tabela do DynamoDB tenha capacidade de throughput suficiente, é possível modificar o número de mapeadores no cluster, o melhora potencialmente o desempenho.

Note

O número de tarefas do mapeador usadas em um trabalho do Hadoop é influenciado peloDivisões de entrada, onde o Hadoop subdivide os dados em blocos lógicos. Se o Hadoop não realizar divisões de entrada suficientes, as operações de gravação poderão não ser capazes de consumir todo o throughput de gravação disponível na tabela do DynamoDB.

Aumento do número de mapeadores

Cada mapeador em um Amazon EMR tem uma taxa máxima de leitura de 1 MiB por segundo. O número de mapeadores em um cluster depende do tamanho dos nós no cluster. (Para obter informações sobre tamanhos de nó e o nome de mapeadores por nó, consulte[Configuração da tarefa](#)noGuia do desenvolvedor do Amazon EMR.)

Se a tabela do DynamoDB possuir ampla capacidade de throughput para leituras, você poderá tentar aumentar o número de mapeadores, ao executar um dos procedimentos a seguir:

- Aumente o tamanho dos nós no cluster. Por exemplo, se o cluster estiver usando nós m1.large (três mapeadores por nó), você pode tentar atualizar para os nós m1.xlarge (oito mapeadores por nó).
- Aumente o número de nós no cluster. Por exemplo, caso possua um cluster de três nós dom1.xlarge, você terá um total de 24 mapeadores disponíveis. Se você dobrar o tamanho do cluster, com o mesmo tipo de nó, terá 48 mapeadores.

É possível usar o AWS Management Console para gerenciar o tamanho ou o número de nós em seu cluster. (Você pode precisar reiniciar o cluster para que essas alterações entrem em vigor.)

Outra forma de aumentar o número de mapeadores é modificar `omapred.tasktracker.map.tasks.maximum`Parâmetro de configuração do Hadoop. (Este é um parâmetro do Hadoop, não um parâmetro do Hive. Não é possível modificá-lo interativamente a partir do prompt de comando.). Se você aumentar o valor de `omapred.tasktracker.map.tasks.maximum`, você poderá aumentar o número de mapeadores sem aumentar o tamanho e o número de nós. No entanto, é possível que os nós do cluster fiquem sem memória se você definir o valor muito alto.

Você pode definir o valor para `omapred.tasktracker.map.tasks.maximum`Como uma ação de bootstrap ao iniciar o primeiro cluster do Amazon EMR. Para obter mais informações, consulte[\(Opcional\) Criar ações de bootstrap para instalar softwares adicionais](#)noGuia de gerenciamento do Amazon EMR.

Diminuição do número de mapeadores

Se você usar `oSELECT`Para selecionar dados de uma tabela externa do Hive mapeada para o DynamoDB, o trabalho do Hadoop poderá usar quantas tarefas forem necessárias, até o número máximo de mapeadores no cluster. Neste cenário, é possível que uma consulta do Hive de longa duração possa

consumir toda a capacidade de leitura provisionada da tabela do DynamoDB, afetando negativamente outros usuários.

Você pode usar o parâmetro `dynamodb.max.map.tasks` para definir um limite superior para mapear tarefas:

```
SET dynamodb.max.map.tasks=1
```

Esse valor deve ser igual ou maior que 1. Quando o Hive processa sua consulta, o trabalho do Hadoop resultante não usará mais do que `dynamodb.max.map.tasks` ao ler a partir da tabela do DynamoDB.

Tópicos adicionais

A seguir estão algumas maneiras de ajustar os aplicativos que usam o Hive para acessar o DynamoDB.

Duração de repetição

Por padrão, o Hive executará novamente um trabalho do Hadoop se ele não tiver retornado quaisquer resultados do DynamoDB dentro de dois minutos. Você pode ajustar esse intervalo, modificando o parâmetro `dynamodb.retry.duration`:

```
SET dynamodb.retry.duration=2;
```

O valor deve ser um número inteiro diferente de zero, representando o número de minutos no intervalo de repetição. O padrão de `dynamodb.retry.duration` é 2 (minutos).

Solicitações de dados em paralelo

Várias solicitações de dados, seja de mais de um usuário ou de mais de um aplicativo, para uma única tabela pode esgotar o throughput provisionado de leitura e diminuir o desempenho.

Duração do processo

A consistência dos dados no DynamoDB depende da ordem de operações de leitura e gravação em cada nó. Embora uma consulta do Hive esteja em andamento, outro aplicativo pode carregar novos dados para a tabela do DynamoDB, modificar ou excluir dados existentes. Nesse caso, os resultados da consulta do Hive podem não refletir as alterações feitas nos dados enquanto a consulta estava em execução.

Tempo de solicitação

Agendar consultas do Hive que acessam uma tabela do DynamoDB quando há menor demanda na tabela do DynamoDB melhora o desempenho. Por exemplo, se a maioria dos usuários do aplicativo vive em São Francisco, talvez você opte por exportar os dados diariamente às 4:00 h (PST) quando a maioria dos usuários está dormindo e não está atualizando registros em seu banco de dados do DynamoDB.

Exportando dados de tabela do DynamoDB para o Amazon S3

Usando a exportação de tabela do DynamoDB, você pode exportar dados de uma tabela do Amazon DynamoDB a qualquer momento dentro da janela de recuperação point-in-time para um bucket do Amazon S3. A exportação de uma tabela do DynamoDB para um bucket do S3 permite que você execute análises e consultas complexas em seus dados usando outros AWS serviços como o Athena, AWS Glue, e a Lake Formation. A exportação de tabelas do DynamoDB é uma solução totalmente gerenciada para exportar

tabelas do DynamoDB em escala e é muito mais rápida do que outras soluções alternativas que envolvem varreduras de tabela. Para mais informações, consulte [Exportação de dados do DynamoDB para o Amazon S3: como ele funciona \(p. 1047\)](#).

Exportar uma tabela não consome capacidade de leitura na tabela e não afeta o desempenho e a disponibilidade da tabela. Você pode exportar dados de tabela para um bucket do S3 de propriedade de outro AWS para uma região diferente daquela em que sua tabela está. Os dados são sempre criptografados de ponta a ponta.

Você pode exportar uma tabela do DynamoDB usando o AWS Management Console, o AWS Command Line Interface ou a API do DynamoDB. Para mais informações, consulte [Solicitando uma exportação de tabela no DynamoDB \(p. 1048\)](#).

Para obter mais informações sobre AWS Disponibilidade e preços da região, consulte [Definição de preços do Amazon DynamoDB](#).

Tópicos

- [Exportação de dados do DynamoDB para o Amazon S3: como ele funciona \(p. 1047\)](#)
- [Solicitando uma exportação de tabela no DynamoDB \(p. 1048\)](#)
- [Formato de saída da tabela do DynamoDB \(p. 1051\)](#)
- [Uso de exportações de tabela do DynamoDB com outros AWS Serviços \(p. 1054\)](#)

Exportação de dados do DynamoDB para o Amazon S3: como ele funciona

Para exportar dados de uma tabela do Amazon DynamoDB para um bucket do Amazon S3, a recuperação point-in-time (PITR) deve estar habilitada na tabela de origem. Você pode exportar dados de tabela de qualquer ponto no tempo dentro da janela PITR, até 35 dias. Para mais informações, consulte [Recuperação point-in-time do DynamoDB \(p. 706\)](#).

Exportar uma tabela não consome capacidade de leitura na tabela e não afeta o desempenho e a disponibilidade da tabela. Você pode exportar dados de tabela para um bucket do S3 de propriedade de outro AWS para uma região diferente daquela em que sua tabela está. Os dados são sempre criptografados em repouso e em trânsito.

Você pode optar por exportar seus dados no formato JSON do DynamoDB ou no formato de texto do Amazon Ion. Para obter mais informações sobre formatos de exportação, consulte [Objetos de dados \(p. 1052\)](#).

Você pode exportar dados para um bucket do S3 de propriedade de uma conta diferente se você tiver as permissões corretas para gravar nessa tabela. O bucket de destino pode estar em uma região diferente da tabela de origem. Para mais informações, consulte [Configuração e permissões do Amazon S3 \(p. 1048\)](#).

Até 300 tarefas de exportação, ou até 100 TB de tamanho de tabela, podem ser exportadas simultaneamente.

O tempo de solicitação e a hora da última atualização incluída na solicitação de exportação do data lake podem variar dentro de uma janela de tempo de um minuto. Por exemplo, se você enviar a solicitação às 2:25 PM, a saída é garantida para conter todos os dados comprometidos com a tabela até 2:24, e os dados confirmados após 2:26 não serão incluídos. A saída pode ou não conter modificações de dados feitas entre 2:24 e 2:26. Os dados exportados também não são transacionalmente consistentes.

AWS CloudTrail registra todas as ações de console e API para exportação de tabela para habilitar registro, monitoramento contínuo e auditoria. Para mais informações, consulte [Registro de operações do DynamoDB usando o AWS CloudTrail \(p. 954\)](#).

A exportação de tabela do DynamoDB foi projetada para ser mais rápida do que exportar uma tabela usando uma verificação de tabela. No entanto, o tempo exato necessário para que a exportação seja concluída depende do tamanho da tabela e da distribuição uniforme dos dados da tabela. Se o seu caso de uso envolver análises em tempo real, você pode usar o Amazon Kinesis Data Streams. Para obter mais informações, consulte o [Guia do desenvolvedor do Amazon Kinesis Data Streams](#).

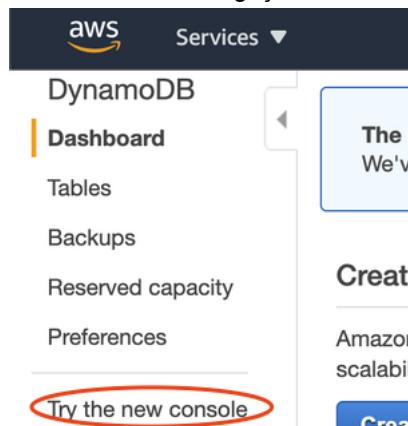
Para obter mais informações sobre o Amazon S3, consulte a [Guia do desenvolvedor do Amazon Simple Storage Service](#).

Solicitando uma exportação de tabela no DynamoDB

As exportações de tabela do DynamoDB permitem exportar dados de tabela para um bucket do Amazon S3, permitindo que você execute análises e consultas complexas em seus dados usando outros AWS serviços como o Athena, AWS Glue, e a Lake Formation. Você pode solicitar uma exportação de tabela usando o AWS Management Console, o AWS CLI ou a API do DynamoDB.

Note

A exportação de tabela do DynamoDB só está disponível no novo AWS Management Console. Se você ainda não estiver usando o novo console, procure o link “Experimente o novo console” na barra lateral de navegação.



Se quiser usar a AWS CLI, você precisa configurá-la primeiro. Para mais informações, consulte [Acessar o DynamoDB \(p. 60\)](#).

Tópicos

- [Configuração e permissões do Amazon S3 \(p. 1048\)](#)
- [Solicitando uma exportação usando o AWS Management Console \(p. 1049\)](#)
- [Obtendo detalhes sobre exportações anteriores no AWS Management Console \(p. 1050\)](#)
- [Solicitando uma exportação usando o AWS CLI \(p. 1050\)](#)
- [Obtendo detalhes sobre exportações anteriores no AWS CLI \(p. 1051\)](#)

Configuração e permissões do Amazon S3

Você pode exportar os dados da tabela para qualquer bucket do Amazon S3 no qual tenha permissão para gravar. O bucket de destino não precisa estar na mesma região ou ter o mesmo proprietário da tabela de origem. Suas AWS Identity and Access Management (IAM) deve permitir que os recursos `s3:AbortMultipartUpload`, `s3:PutObject`, `es3:PutObjectAcl`, além de `dynamodb:ExportTableToPointInTime`, conforme mostrado no exemplo a seguir.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
    {
        "Sid": "AllowDynamoDBExportAction",
        "Effect": "Allow",
        "Action": "dynamodb:ExportTableToPointInTime",
        "Resource": "arn:aws:dynamodb:us-east-1:111122223333:table/my-table"
    },
    {
        "Sid": "AllowWriteToDestinationBucket",
        "Effect": "Allow",
        "Action": [
            "s3:AbortMultipartUpload",
            "s3:PutObject",
            "s3:PutObjectAcl"
        ],
        "Resource": "arn:aws:s3:::your-bucket/*"
    }
]
```

O código a seguir é um exemplo de política do bucket do S3 (na conta de destino)).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleStatement",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::awsexamplebucket1/*"
        }
    ]
}
```

A revogação dessas permissões enquanto uma exportação estiver em andamento resultará em arquivos parciais.

Solicitando uma exportação usando oAWS Management Console

O exemplo a seguir demonstra como usar o console do DynamoDB para exportar uma tabela existente chamadaMusicCollection.

Note

Esse procedimento supõe que você tenha habilitado a recuperação point-in-time. Para habilitá-lo para oMusicCollection, na tabelaVisão geral, na guiaDetalhes da tabela, selecioneHabilitarparaRecuperação point-in-time.

Para solicitar uma exportação de tabela

1. Faça login noAWS Management Consolee abra o console do DynamoDB em<https://console.aws.amazon.com/dynamodb/>.
2. No painel de navegação, no lado esquerdo do console, selecioneFluxos e exportações.

3. No Exportar, selecione Exportar para o S3.
4. Escolha uma tabela de origem e um bucket do S3 de destino. Se o bucket de desinação for de propriedade da sua conta, você pode usar o Procurar o S3 para encontrá-lo. Caso contrário, insira o URL do bucket usando os3://**bucketname/prefix** formato. **oprefix** é uma pasta opcional para ajudar a manter o seu bucket de desinação organizado.
5. Por padrão, sua tabela será exportada no formato JSON do DynamoDB a partir da hora de restauração mais recente na janela de recuperação point-in-time e criptografada usando uma chave do Amazon S3 do Amazon S3 (SSE-S3). Se quiser alterar essas configurações, expanda a Configurações adicionais da página e insira as configurações de exportação desejadas.

Note

Se você optar por criptografar sua exportação usando uma chave protegida por AWS Key Management Service (AWS KMS), a chave deve estar na mesma região que o bucket do S3 de destino.

6. Clique no ícone da barra de ferramentas Exportar para iniciar a exportação.

Obtendo detalhes sobre exportações anteriores no AWS Management Console

Você pode encontrar informações sobre tarefas de exportação executadas no passado clicando em Fluxos e exportações na barra lateral de navegação e, em seguida, selecionando a opção Exportar Guia do. Esta guia contém uma lista de todas as exportações que você criou nos últimos 90 dias. Selecionar o ARN de uma tarefa listada na guia Exportações recuperará informações sobre essa exportação, incluindo todas as configurações avançadas escolhidas. Observe que, embora os metadados de tarefa de exportação expire após 90 dias e os trabalhos mais antigos do que isso não sejam mais encontrados nesta lista, os objetos no bucket do S3 permanecem enquanto suas políticas de bucket permitirem. O DynamoDB nunca exclui nenhum dos objetos que ele cria no bucket do S3 durante uma exportação.

Solicitando uma exportação usando o AWS CLI

O exemplo a seguir mostra como usar o AWS CLI para exportar uma tabela existente chamada `MusicCollection` para um bucket do S3 chamado `ddb-export-musiccollection`.

Note

Esse procedimento supõe que você tenha habilitado a recuperação point-in-time. Para ativar esse recurso para a tabela `MusicCollection`, execute o seguinte comando.

```
aws dynamodb update-continuous-backups \
    --table-name MusicCollection \
    --point-in-time-recovery-specification PointInTimeRecoveryEnabled=True
```

O comando a seguir exporta o `MusicCollection` para um bucket do S3 chamado `ddb-export-musiccollection` com um prefixo de `2020-Nov`. Os dados da tabela serão exportados no formato JSON do DynamoDB a partir de um horário específico na janela de recuperação point-in-time e criptografados usando uma chave do Amazon S3 do Amazon S3 (SSE-S3).

```
aws dynamodb export-table-to-point-in-time \
    --table-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection \
    --s3-bucket ddb-export-musiccollection \
    --s3-prefix 2020-Nov \
    --export-format DYNAMODB_JSON \
    --export-time 1604632434 \
    --s3-sse-algorithm AES256
```

Note

Se você optar por criptografar sua exportação usando uma chave protegida por AWS Key Management Service(AWS KMS), a chave deve estar na mesma região que o bucket do S3 de destino.

Obtendo detalhes sobre exportações anteriores noAWS CLI

Você pode encontrar informações sobre tarefas de exportação executadas no passado usando a opção `list-exports`Comando da. Esse comando retorna uma lista de todas as exportações criadas nos últimos 90 dias. Observe que, embora os metadados de tarefa de exportação expire após 90 dias e trabalhos mais antigos do que isso não sejam mais retornados pelo método `list-exports`, os objetos em seu bucket do S3 permanecerão enquanto suas políticas de bucket permitirem. O DynamoDB nunca exclui nenhum dos objetos que ele cria no bucket do S3 durante uma exportação.

No exemplo a seguir, usamos o opcional `table-arn`para listar apenas as exportações de uma tabela específica.

```
aws dynamodb list-exports \
  --table-arn arn:aws:dynamodb:us-east-1:123456789012:table/ProductCatalog
```

Para recuperar informações detalhadas sobre uma tarefa de exportação específica, incluindo quaisquer configurações avançadas, use o `describe-export`Comando da.

```
aws dynamodb describe-export \
  --export-arn arn:aws:dynamodb:us-east-1:123456789012:table/ProductCatalog/
  export/01234567890123-a1b2c3d4
```

Formato de saída da tabela do DynamoDB

Uma exportação de tabela do DynamoDB inclui dois arquivos de manifesto além dos arquivos que contêm os dados da tabela. Esses arquivos são salvos no bucket do Amazon S3 especificado no [Solicitação de exportação \(p. 1048\)](#). As seguintes seções descrevem o formato e o conteúdo de cada objeto de saída.

Tópicos

- [Arquivos manifesto \(p. 1051\)](#)
- [Objetos de dados \(p. 1052\)](#)

Arquivos manifesto

O DynamoDB cria dois arquivos manifesto, juntamente com seus arquivos de soma de verificação, no bucket S3 especificado para cada solicitação de exportação.

```
export-prefix/AWSDynoDB/ExportId/manifest-summary.json
export-prefix/AWSDynoDB/ExportId/manifest-summary.checksum
export-prefix/AWSDynoDB/ExportId/manifest-files.json
export-prefix/AWSDynoDB/ExportId/manifest-files.checksum
```

Você escolhe um `export-prefix`quando você solicita uma exportação de tabela. Isso ajuda você a manter os arquivos no bucket do S3 de destino organizados. O `ExportId`é um token exclusivo para garantir que várias exportações para o mesmo bucket S3 e `export-prefix`não sobrescrevam uns aos outros.

Note

O DynamoDB também cria um arquivo vazio chamado _startedNo mesmo diretório que os arquivos manifesto. Este arquivo verifica se o bucket de destino é gravável e se a exportação foi iniciada. Ele pode ser excluído com segurança.

O manifesto de resumo

Omanifest-summary.json contém informações resumidas sobre o trabalho de exportação. Seu formato é o seguinte:

```
{  
    "version": "2020-06-30",  
    "exportArn": "arn:aws:dynamodb:us-east-1:123456789012:table/ProductCatalog/  
export/01234567890123-alb2c3d4",  
    "startTime": "2020-11-04T07:28:34.028Z",  
    "endTime": "2020-11-04T07:33:43.897Z",  
    "tableArn": "arn:aws:dynamodb:us-east-1:123456789012:table/ProductCatalog",  
    "tableId": "12345a12-abcd-123a-ab12-1234abc12345",  
    "exportTime": "2020-11-04T07:28:34.028Z",  
    "s3Bucket": "ddb-productcatalog-export",  
    "s3Prefix": "2020-Nov",  
    "s3SseAlgorithm": "AES256",  
    "s3SseKmsKeyId": null,  
    "manifestFilesS3Key": "AWSDynamoDB/01345678901234-abc12345/manifest-files.json",  
    "billedSizeBytes": 0,  
    "itemCount": 8,  
    "outputFormat": "DYNAMODB_JSON"  
}
```

O manifesto de arquivos

O `manifest-files.json` contém informações sobre os arquivos que contêm seus dados da tabela exportados. O arquivo está em [Linhas JSON](#), o que significa que as novas linhas são usadas como delimitadores de item. No exemplo a seguir, os detalhes de um arquivo de dados de um manifesto de arquivos são formatados em várias linhas por razões de legibilidade.

Objetos de dados

O DynamoDB pode exportar os dados da tabela em dois formatos: DynamoDB JSON e Amazon Ion. Independentemente do formato escolhido, seus dados serão gravados em vários arquivos compactados nomeados pelas chaves encontradas na secção `manifest-files.jsonfile`.

```
export-prefix/AWSDynoDB/ExportId/data/bafybeiczs3yxay3o4abnabb.json.gz
export-prefix/AWSDynoDB/ExportId/data/qkes5o3lnrhzhnkwyax3hxvya.json.qz
```

DynamoDB JSON

Uma exportação de tabela no formato JSON do DynamoDB consiste em vários `Item`s do. Cada objeto individual está no formato JSON empacotado padrão do DynamoDB.

Ao criar analisadores personalizados para dados de exportação JSON do DynamoDB, lembre-se de que o formato é [Linhas JSON](#). Isso significa que as novas linhas são usadas como delimitadores de item. Muitos AWS serviços, como Athena e AWS Glue, irá analisar este formato automaticamente.

No exemplo a seguir, um único item de uma exportação JSON do DynamoDB foi formatado em várias linhas para fins de legibilidade.

```
{  
    "Item":{  
        "Authors":{  
            "SS":[  
                "Author1",  
                "Author2"  
            ]  
        },  
        "Dimensions":{  
            "S":"8.5 x 11.0 x 1.5"  
        },  
        "ISBN":{  
            "S":"333-3333333333"  
        },  
        "Id":{  
            "N":"103"  
        },  
        "InPublication":{  
            "BOOL":false  
        },  
        "PageCount":{  
            "N":"600"  
        },  
        "Price":{  
            "N":"2000"  
        },  
        "ProductCategory":{  
            "S":"Book"  
        },  
        "Title":{  
            "S":"Book 103 Title"  
        }  
    }  
}
```

Amazon Ion

[Amazon Ion](#) é um formato de serialização de dados hierárquico, com descrição automática, criado para lidar com os desafios rápidos de desenvolvimento, desacoplamento e eficiência enfrentados todos os dias enquanto cria arquiteturas orientadas a serviços em grande escala. O DynamoDB oferece suporte à exportação de dados de tabela nos [Formato de texto](#), que é um superconjunto de JSON.

Quando você exporta uma tabela para o formato Ion, os tipos de dados do DynamoDB usados na tabela são mapeados para [Tipos de dados do Ion](#). Os conjuntos do DynamoDB usam [Anotações de tipo Ion](#) para desambiguar o tipo de dados usado na tabela de origem.

Conversão de tipo de dados do DynamoDB para ion

Tipo de dados DynamoDB	Representação Ion
String (S)	string
booliano (BOOL)	bool
Número (N)	decimal

Tipo de dados DynamoDB	Representação Ion
Binário (B)	blob
Conjunto (SS, NS, BS)	lista (com anotação de tipo \$dynamodb_ss, \$dynamodb_ns ou \$dynamodb_bs)
Lista	lista
Mapa	struct

Os itens em uma exportação de íons são delimitados por novas linhas. Cada linha começa com um marcador de versão Ion, seguido por um item no formato Ion. No exemplo a seguir, um item de uma exportação de íons foi formatado em várias linhas para fins de legibilidade.

```
$ion_1_0 {
    Item: {
        Authors:$dynamodb_SS::["Author1", "Author2"],
        Dimensions:"8.5 x 11.0 x 1.5",
        ISBN:"333-333333333",
        Id:103.,
        InPublication:false,
        PageCount:6d2,
        Price:2d3,
        ProductCategory:"Book",
        Title:"Book 103 Title"
    }
}
```

Uso de exportações de tabela do DynamoDB com outros AWS Serviços do

Depois que os dados da tabela do DynamoDB forem exportados para o Amazon S3, você poderá processá-los a partir de uma variedade de outros AWS Serviços da . As seções a seguir detalham algumas das opções para consultar os dados exportados.

Utilização de exportações com o Amazon Athena

O Amazon Athena é um serviço de consulta interativo que facilita a análise de dados no Amazon S3 usando SQL padrão. O Athena não precisa de servidor, portanto, não há infraestrutura para gerenciar e você paga apenas pelas consultas executadas. Para obter mais informações, consulte o [Guia do usuário do Amazon Athena](#).

Usar exportações com AWS Glue

AWS Glue é um serviço de extração, transformação e carga (ETL) totalmente gerenciado que facilita o carregamento de seus dados para análise. Você pode criar e executar um trabalho ETL com alguns cliques no AWS Glue Editor visual. Você simplesmente aponta AWS Glue para os seus dados e AWS Glue detecta seus dados e armazena os metadados associados (como definição de tabela e esquema) no AWS Glue Data Catalog. Uma vez catalogados, seus dados são imediatamente pesquisáveis, consultáveis e disponíveis para ETL. Para obter mais informações, consulte o [Guia do desenvolvedor do AWS Glue](#).

Usar exportações com AWS Lake Formation

AWS Lake Formation facilita a configuração de um data lake seguro em dias. Um data lake é um repositório centralizado, curado e seguro que armazena todos os seus dados, tanto em sua forma original

quanto preparado para análise. Um data lake permite que você decompense silos de dados e combine diferentes tipos de análises para obter insights e orientar melhores decisões de negócios.

Criar um data lake com Lake Formation é tão simples quanto definir fontes de dados e quais políticas de segurança e acesso a dados você deseja aplicar. O Lake Formation ajuda a coletar e catalogar dados de bancos de dados e armazenamento de objetos, mover os dados para o novo data lake do Amazon S3, limpar e classificar seus dados usando algoritmos de aprendizado de máquina e proteger o acesso aos seus dados confidenciais. Para obter mais informações, consulte o [Guia do desenvolvedor do AWS Lake Formation](#).

Cotas de serviço, conta e tabela no Amazon DynamoDB

Esta seção descreve as cotas atuais do Amazon DynamoDB (ou nenhuma cota, em alguns casos). Salvo indicação em contrário, cada cota aplica-se por região.

Tópicos

- [Modo de capacidade de leitura/gravação e taxa de transferência \(p. 1056\)](#)
- [Tables \(p. 1058\)](#)
- [Tabelas globais \(p. 1059\)](#)
- [Índices secundários \(p. 1059\)](#)
- [Chaves de partição e chaves de classificação \(p. 1060\)](#)
- [Regras de nomenclatura \(p. 1060\)](#)
- [Tipos de dados \(p. 1061\)](#)
- [Items \(p. 1061\)](#)
- [Attributes \(p. 1062\)](#)
- [Parâmetros de expressão \(p. 1062\)](#)
- [Transações do DynamoDB \(p. 1063\)](#)
- [DynamoDB Streams \(p. 1063\)](#)
- [DynamoDB Accelerator \(DAX\) \(p. 1064\)](#)
- [Limites específicos de API \(p. 1064\)](#)
- [DynamoDB Encryption em repouso \(p. 1065\)](#)
- [Exportação de tabela para o Amazon S3 \(p. 1066\)](#)

Modo de capacidade de leitura/gravação e taxa de transferência

Você pode alternar entre os modos de capacidade de leitura/gravação uma vez a cada 24 horas.

Tamanhos de unidade de capacidade (para tabelas provisionadas)

Uma unidade de capacidade de leitura = uma leitura fortemente consistente por segundo, ou duas leituras eventualmente consistentes por segundo, para itens de até 4 KB de tamanho.

Uma unidade de capacidade de gravação = uma gravação por segundo para itens de até 1 KB de tamanho.

As solicitações de leitura transacional exigem duas unidades de capacidade de leitura para executar uma leitura por segundo para itens de até 4 KB.

As solicitações de gravação transacional exigem duas unidades de capacidade de gravação para executar uma gravação por segundo para itens de até 1 KB.

Tamanhos de unidade de solicitação (para tabelas sob demanda)

Uma unidade de solicitação de leitura = uma leitura fortemente consistente, ou duas leituras eventualmente consistentes, para itens de até 4 KB de tamanho.

Uma unidade de solicitação de gravação = uma gravação para itens de até 1 KB de tamanho.

As solicitações de leitura transacional exigem duas unidades de solicitação de leitura para executar uma leitura para itens de até 4 KB.

As solicitações de gravação transacional exigem duas unidades de solicitação de gravação para executar uma gravação para itens de até 1 KB.

Cotas padrão de taxa de transferência

AWSO impõe algumas cotas padrão para a taxa de transferência que você pode provisionar. Estas são as cotas, a menos que você solicite uma quantidade maior. Para solicitar um aumento da cota de serviço, consulte<https://aws.amazon.com/support>.

	Sob demanda	Provisionada
Per table	40,000 read request units and 40,000 write request units	40,000 read capacity units and 40,000 write capacity units
Per account	Not applicable	80,000 read capacity units and 80,000 write capacity units
Minimum throughput for any table or global secondary index	Not applicable	1 read capacity unit and 1 write capacity unit

Note

Todas as taxas de transferência disponíveis da conta podem ser aplicadas a uma única tabela ou entre várias tabelas.

A cota de taxa de transferência provisionada inclui a soma da capacidade da tabela com a capacidade de todos os seus índices secundários globais.

No AWS Management Console, é possível usar o Amazon CloudWatch para ver qual é sua taxa de transferência de leitura e gravação atual em determinado AWS Região, olhando para o `read capacity` e `write capacity` graphs no [Métricas Guia](#). Certifique-se de não se aproximar muito das cotas.

Se você aumentou as cotas padrão da taxa de transferência provisionada, será possível usar a operação `DescribeLimits` para ver os valores de cota atuais.

Aumentar ou diminuir a taxa de transferência (para tabelas provisionadas)

Aumento da taxa de transferência provisionada

Você pode aumentar as `ReadCapacityUnits` ou as `WriteCapacityUnits` sempre que necessário, usando o AWS Management Console ou a operação `UpdateTable`. Em uma única chamada, é possível aumentar a taxa de transferência provisionada de uma tabela, para quaisquer índices secundários globais na tabela ou para qualquer combinação dessas. As novas configurações não têm efeito até que a operação `UpdateTable` esteja concluída.

Você não pode exceder as cotas por conta ao adicionar a capacidade provisionada, e o DynamoDB não permite aumentar a capacidade provisionada de forma extremamente rápida. Além dessas restrições, você pode aumentar a capacidade provisionada de suas tabelas para o nível que for necessário. Para mais informações sobre cotas por conta, consulte a seção anterior, [Cotas padrão de taxa de transferência \(p. 1057\)](#).

Diminuição da taxa de transferência provisionada

Para cada tabela e índice secundário global em uma operação `UpdateTable`, é possível diminuir `ReadCapacityUnits` ou `WriteCapacityUnits` (ou ambas). As novas configurações não têm efeito até que a operação `UpdateTable` seja concluída. É permitido diminuir até quatro vezes a qualquer momento por dia. Um dia é definido pelo Tempo Universal Coordenado (UTC). Além disso, se não houver redução na última hora, será permitido fazer uma redução adicional. Isso leva o número máximo de diminuições em um dia para 27 vezes (4 diminuições na primeira hora e 1 diminuição para cada uma das janelas de 1 hora subsequentes em um dia).

Important

Como os limites de diminuição de tabela e índice secundário global são separados, qualquer índice secundário global de uma tabela específica tem os próprios limites de diminuição.

No entanto, se uma única solicitação diminuir a taxa de transferência de uma tabela e um índice secundário global, ela será rejeitada se exceder os limites atuais. Solicitações não são processadas parcialmente.

Example

Nas primeiras 4 horas de um dia, uma tabela com um índice secundário global pode ser modificada da seguinte forma:

- Diminuir `WriteCapacityUnits` ou `ReadCapacityUnits` (ou ambos) da tabela quatro vezes.
- Diminuir `WriteCapacityUnits` ou `ReadCapacityUnits` (ou ambos) do índice secundário global quatro vezes.

Ao final do mesmo dia, a tabela e a taxa de transferência do índice secundário global poderão ser diminuídas 27 vezes cada, no total.

Tables

Tamanho da tabela

Não há limite prático para o tamanho de uma tabela. As tabelas não são limitadas em termos de número de itens ou de bytes.

Tabelas por conta

Para qualquer AWS, há uma cota inicial de 256 tabelas por AWS Região:

Para solicitar um aumento da cota de serviço, consulte <https://aws.amazon.com/support>.

Tabelas globais

AWS A coloca algumas cotas padrão na taxa de transferência que você pode provisionar ou utilizar ao usar tabelas globais.

	Sob demanda	Provisionada
Per table	40,000 read request units and 40,000 write request units	40,000 read capacity units and 40,000 write capacity units
Per table, per day	10 TB for all source tables to which a replica was added	10 TB for all source tables to which a replica was added

Se você estiver adicionando uma réplica para uma tabela que está configurada para usar mais de 40.000 Write Capacity Units (WCU – Unidades de capacidade de gravação), será necessário solicitar um aumento de cota de serviço para a cota de WCU de adição de réplica. Para solicitar um aumento da cota de serviço, consulte <https://aws.amazon.com/support>.

As operações transacionais fornecem garantia de atomicidade, consistência, isolamento e durabilidade (ACID — atomicity, consistency, isolation, and durability) somente na AWS Região onde a gravação é feita originalmente. As transações não são compatíveis entre regiões em tabelas globais. Por exemplo, suponha que você tenha uma tabela global com réplicas nas regiões Leste dos EUA (Ohio) e Oeste dos EUA (Oregon), e execute uma operação TransactWriteItems na região Leste dos EUA (Norte da Virgínia). Nesse caso, você pode observar transações parcialmente concluídas na região Oeste dos EUA (Oregon) conforme as alterações são replicadas. As alterações só são replicadas para outras regiões quando forem confirmadas na região de origem.

Índices secundários

Índices secundários por tabela

Você pode definir um máximo de cinco índices secundários locais.

Há uma cota inicial de 20 índices secundários globais por tabela. Para solicitar um aumento da cota de serviço, consulte <https://aws.amazon.com/support>.

Você pode criar ou excluir somente um índice secundário global por `UpdateTable` operação.

Atributos de índice secundário projetados por tabela

Você pode projetar um total de até 100 atributos em todos os índices secundários locais e globais de uma tabela. Isso se aplica apenas a atributos projetados especificados pelo usuário.

Em `umCreateTable`, se você especificar `umProjectionType` de `INCLUDE`, a contagem total de atributos especificados em `emNonKeyAttributes`, somada em todos os índices secundários, não deve exceder 100. Se você projetar o mesmo nome de atributo em dois índices diferentes, isso conta como dois atributos distintos ao determinar o total.

Esse limite não se aplica a índices secundários com `ProjectionType` de `KEYS_ONLY` ou `ALL`.

Chaves de partição e chaves de classificação

Tamanho da chave de partição

O tamanho mínimo de um valor de chave de partição é 1 byte. O tamanho máximo é 2048 bytes.

Valores de chave de partição

Não há um limite prático para o número de valores de chave de partição distintos, para tabelas ou para índices secundários.

Tamanho da chave de classificação

O tamanho mínimo de um valor de chave de classificação é 1 byte. O tamanho máximo é de 1024 bytes.

Valores de chave de classificação

Em geral, não há limite prático para o número de valores de chave de classificação distintos por valor de chave de partição.

Há exceção para tabelas com índices secundários. Com um índice secundário local, há um limite para tamanhos de coleção de itens: Para cada valor de chave de partição distinto, os tamanhos totais de todos os itens de tabela ou índice não podem exceder 10 GB. Isso pode restringir o número de chaves de classificação por valor de chave de partição. Para mais informações, consulte [Limite de tamanho da coleção de itens \(p. 614\)](#).

Regras de nomenclatura

Nomes de tabela e nomes de índice secundário

Nomes de tabelas e índices secundários devem ter pelo menos 3 caracteres, mas não podem ser maiores do que 255 caracteres. Veja a seguir os caracteres permitidos:

- A–Z
- a–z
- 0–9
- _ (sublinhado)
- – (hífen)
- . (ponto)

Nomes de atributo

Em geral, um nome de atributo deve ter pelo menos 64 caracteres, mas não pode ser maior do que 64 KB.

Veja as exceções a seguir. Estes nomes de atributo não devem ser maiores do que 255 caracteres:

- Nomes de chave de partição de índice secundário.
 - Nomes de chave de classificação de índice secundário.
 - Os nomes de atributos projetados especificados pelo usuário (aplicável apenas a índices secundários locais). Em uma operação `CreateTable`, se você especificar um `ProjectionType` como `INCLUDE`, os nomes dos atributos no parâmetro `NonKeyAttributes` terão tamanho restrito. Os tipos de projeção `KEYS_ONLY` e `ALL` não são afetados.

Esses nomes de atributos devem ser codificados usando UTF-8 e o tamanho total de cada nome (após a codificação) não pode exceder 255 bytes.

Tipos de dados

String

O tamanho de uma string é restrito pelo tamanho de item máximo 400 KB.

Strings são Unicode com codificação binária UTF-8. Como a codificação UTF-8 tem largura variável, o DynamoDB determina o tamanho de uma string usando seus bytes UTF-8.

Number

Um número pode ter até 38 dígitos de precisão, e pode ser positivo, negativo ou zero.

O DynamoDB usa strings JSON para representar dados de números em solicitações e respostas. Para mais informações, consulte [API do DynamoDB de baixo nível \(p. 217\)](#).

Se a precisão dos números for importante, você deve passar os números para o DynamoDB usando strings que você converte de um tipo de número.

Binary

O tamanho de um binário é restrito pelo tamanho de item máximo 400 KB.

Os aplicativos que funcionam com atributos binários devem codificar os dados em formato base64 antes de enviá-los para o DynamoDB. Após o recebimento dos dados, o DynamoDB decodifica-os em uma matriz de bytes não assinados e usa isso como o tamanho do atributo.

Items

Tamanho do item

O tamanho de item máximo do DynamoDB é 400 KB, o que inclui o tamanho binário do nome do atributo (tamanho UTF-8) e os tamanhos de valor de atributo (novamente, tamanho binário). O nome do atributo conta para o limite de tamanho.

Por exemplo, considere um item com dois atributos: um atributo chamado “cor-camisa” com o valor “R” e outro atributo chamado “tamanho-camisa” com o valor “M”. O tamanho total desse item é 23 bytes.

Tamanho do Item para Tabelas com Índices Secundários Locais

Para cada índice secundário local em uma tabela, existe um limite de 400 KB no total do seguinte:

- O tamanho dos dados de um item na tabela.
- O tamanho da entrada de índice secundário local correspondente a esse item, incluindo seus valores de chave e atributos projetados.

Attributes

Pares de nome-valor de atributo por item

O tamanho cumulativo de atributos por item deve se adequar ao tamanho máximo de item do DynamoDB (400 KB).

Número de valores em lista, mapa ou conjunto

Não há limite para o número de valores em uma lista, um mapa ou um conjunto, desde que o item que contém os valores fique no limite de tamanho de item de 400 KB.

Valores de atributo

Valores de atributos binários e de string vazios serão permitidos se o atributo não for usado como um atributo de chave para uma tabela ou um índice. Valores binários e de string vazios são permitidos dentro dos tipos de conjunto, lista e mapa. Um valor de atributo não pode ser um conjunto vazio (conjunto de strings, conjunto de número ou conjunto binário). Entretanto, Lists e Maps vazios são permitidos.

Profundidade de atributo aninhado

O DynamoDB oferece suporte a atributos aninhados de até 32 níveis de profundidade.

Parâmetros de expressão

Os parâmetros de expressão incluem `ProjectionExpression`, `ConditionExpression`, `UpdateExpression` e `FilterExpression`.

Lengths

O tamanho máximo de qualquer string de expressão é 4 KB. Por exemplo, o tamanho de `ConditionExpression a=b` é 3 bytes.

O tamanho máximo de qualquer nome de atributo de expressão única ou valor de atributo de expressão é 255 bytes. Por exemplo, `#name` é 5 bytes; `:val` é 4 bytes.

O tamanho máximo de todas as variáveis de substituição em uma expressão é 2 MB. Esta é a soma dos tamanhos de todos os `ExpressionAttributeNames` e `ExpressionAttributeValues`.

Operadores e operandos

O número máximo de operadores ou funções permitidos em uma `UpdateExpression` é 300. Por exemplo, a `UpdateExpression` `SET a = :val1 + :val2 + :val3` contém dois operadores “+”.

O número máximo de operandos do comprador `IN` é 100.

Palavras reservadas

O DynamoDB não impede que você use nomes que conflitem com palavras reservadas. (Para obter uma lista completa, consulte [Palavras reservadas no DynamoDB \(p. 1125\)](#).)

No entanto, caso use uma palavra reservada em um parâmetro de expressão, você também deverá especificar `ExpressionAttributeNames`. Para mais informações, consulte [Nomes de atributo de expressão no DynamoDB \(p. 418\)](#).

Transações do DynamoDB

As operações da API transacionais do DynamoDB têm as seguintes restrições:

- Uma transação não pode conter mais de 25 itens únicos.
- Uma transação não pode conter mais de 4 MB de dados.
- Duas ações em uma transação não podem funcionar contra o mesmo item na mesma tabela. Por exemplo, você não pode usar `ConditionCheck` e `Update` no mesmo item em uma transação.
- Uma transação não pode operar nas tabelas em mais de um AWS Região.
- As operações transacionais fornecem garantia de atomicidade, consistência, isolamento e durabilidade (ACID — atomicity, consistency, isolation, and durability) somente na AWS Região onde a gravação é feita originalmente. As transações não são compatíveis entre regiões em tabelas globais. Por exemplo, suponha que você tenha uma tabela global com réplicas nas regiões Leste dos EUA (Ohio) e Oeste dos EUA (Oregon) e execute uma operação `TransactWriteItems` na região Leste dos EUA (Norte da Virgínia). Nesse caso, você pode observar transações parcialmente concluídas na região Oeste dos EUA (Oregon) conforme as alterações são replicadas. As alterações só são replicadas para outras regiões quando forem confirmadas na região de origem.

DynamoDB Streams

Leitores simultâneos de um estilhaço em DynamoDB Streams

Não permita que mais de dois processos leiam o mesmo estilhaço do DynamoDB Streams ao mesmo tempo. Exceder esse limite pode resultar em limitação de solicitação.

Capacidade de gravação máxima para uma tabela com um fluxo habilitado

AWSA coloca algumas cotas padrão na capacidade de gravação em tabelas do DynamoDB com DynamoDB Streams habilitados. Estas são as cotas, a menos que você solicite uma quantidade maior. Para solicitar um aumento da cota de serviço, consulte<https://aws.amazon.com/support>.

- Leste dos EUA (Norte da Virgínia), Leste dos EUA (Ohio), Oeste dos EUA (Norte da Califórnia), Oeste dos EUA (Oregon), América do Sul (São Paulo), Europa (Frankfurt), Europa (Irlanda), Ásia-Pacífico (Tóquio), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney), China (Pequim) Regiões:
 - Por tabela – 40.000 unidades de capacidade de gravação
- Todas as outras regiões:
 - Por tabela – 10.000 unidades de capacidade de gravação

Note

As cotas de taxa de transferência provisionada também se aplicam às tabelas do DynamoDB com o DynamoDB Streams habilitado. Para mais informações, consulte [Cotas padrão de taxa de transferência \(p. 1057\)](#).

DynamoDB Accelerator (DAX)

Disponibilidade de regiões do AWS

Para uma lista de AWS Regiões em que o DAX está disponível, consulte [DynamoDB Accelerator \(DAX\)](#) no AWS Referência geral.

Nodes

Um cluster DAX consiste em exatamente um nó principal e entre 0 e 10 nós de réplica de leitura.

O número total de nós (por AWS) não pode exceder 50 em um único AWS Região :

Grupos de parâmetros

Você pode criar até 20 grupos de parâmetro DAX do por região.

Grupos de sub-redes

Você pode criar até 50 grupos de sub-redes DAX do por região.

Em um grupo de sub-redes, você pode definir até 20 sub-redes.

Limites específicos de API

CreateTable/UpdateTable/DeleteTable

Em geral, você pode ter até 50 `CreateTable`, `UpdateTable`, e `DeleteTable` solicitações executadas simultaneamente (em qualquer combinação). Em outras palavras, o número total de tabelas no estado `CREATING`, `UPDATING` ou `DELETING` não pode exceder 50.

A única exceção é quando você está criando uma tabela com um ou mais índices secundários. Você pode ter até 25 solicitações desse tipo executadas ao mesmo tempo. Entretanto, se as especificações de tabela ou índice forem complexas, o DynamoDB poderá reduzir temporariamente o número de operações simultâneas.

BatchGetItem

Uma única `BatchGetItem` operação pode recuperar um máximo de 100 itens. O tamanho total de todos os itens recuperados não pode exceder 16 MB.

BatchWriteItem

Uma única `BatchWriteItem` operação pode conter até 25 `PutItem` ou `DeleteItem` solicitações. O tamanho total de todos os itens gravados não pode exceder 16 MB.

DescribeTableReplicaAutoScaling

O método `DescribeTableReplicaAutoScaling` oferece suporte a apenas 10 solicitações por segundo.

DescribeLimits

`DescribeLimits` deve ser chamado apenas periodicamente. Você pode esperar erros de limitação se chamá-lo mais de uma vez em um minuto.

DescribeContributorInsights/ListContributorInsights/UpdateContributorInsights

`DescribeContributorInsights`, `ListContributorInsights` e `UpdateContributorInsights` devem ser chamados apenas periodicamente. O DynamoDB oferece suporte a cinco solicitações por segundo para cada uma dessas APIs.

Query

O conjunto de resultados de `Query` é limitado a 1 MB por chamada. Você pode usar a `LastEvaluatedKey` da resposta da consulta para recuperar mais resultados.

Scan

O conjunto de resultados de `Scan` é limitado a 1 MB por chamada. Você pode usar a `LastEvaluatedKey` da resposta da verificação para recuperar mais resultados.

UpdateTableReplicaAutoScaling

`UpdateTableReplicaAutoScaling` O método oferece suporte a somente dez solicitações por segundo.

DynamoDB Encryption em repouso

Você pode alternar entre um AWSKMS mestra do cliente (CMK) de propriedade, um AWSKMS gerenciado e um cliente gerenciado CMK até quatro vezes, a qualquer hora por janela de 24 horas por tabela, a partir da criação da tabela. Se não houve alteração nas últimas seis horas, uma alteração adicional será permitida. Isso leva o número máximo de alterações em um dia para oito (quatro alterações nas primeiras seis horas e uma alteração para cada uma das janelas de seis horas subsequentes em um dia).

Você pode alternar as chaves de criptografia para usar uma AWSKMS CMK de propriedade sempre que necessário, mesmo que a cota acima tenha sido esgotada.

Estas são as cotas, a menos que você solicite uma quantidade maior. Para solicitar um aumento da cota de serviço, consulte<https://aws.amazon.com/support>.

Exportação de tabela para o Amazon S3

Até 300 tarefas de exportação, ou até 100 TB de tamanho de tabela, podem ser exportadas simultaneamente.

Referência de API de baixo nível

A Referência de API do Amazon DynamoDB contém uma lista completa de operações compatíveis com:

- [DynamoDB](#).
- [DynamoDB Streams](#).
- [DynamoDB Accelerator \(DAX\)](#).

Apêndice do DynamoDB

Tópicos

- Solução de problemas de estabelecimento de conexão SSL/TLS (p. 1068)
- Tabelas e dados de exemplo (p. 1070)
- Criar tabelas de exemplo e carregar dados (p. 1079)
- Aplicativo de exemplo do DynamoDB usando oAWS SDK for Python (Boto): Jogo da velha (p. 1095)
- Exportar e importar dados do DynamoDB usando oAWS Data Pipeline (p. 1114)
- Back-end de armazenamento do Amazon DynamoDB para Titan (p. 1125)
- Palavras reservadas no DynamoDB (p. 1125)
- Parâmetros condicionais herdados (p. 1134)
- Versão anterior da API de baixo nível (2011-12-05) (p. 1152)

Solução de problemas de estabelecimento de conexão SSL/TLS

O Amazon DynamoDB está em vias de mover nossos endpoints para certificados confiáveis assinados pela autoridade de certificação Amazon Trust Services (ATS), e não por uma autoridade de certificação de terceiros. Em dezembro de 2017, lançamos na região EU-WEST-3 (Paris) os certificados confiáveis emitidos pela Amazon Trust Services. Todas as novas regiões lançadas depois de dezembro de 2017 têm endpoints com certificados emitidos pela Amazon Trust Services. Este guia mostra como você valida e resolve problemas de conexão SSL/TLS.

Como testar seu aplicativo ou serviço

A maioria AWSOs SDKs e interfaces de linha de comando (CLIs) são compatíveis com a autoridade de certificação Amazon Trust Services. Se você estiver usando uma versão doAWSO SDK para Python ou CLI lançado antes de 29 de outubro de 2013, você deve atualizar. Os SDKs e CLIs para .NET, Java, PHP, Go, JavaScript e C++ não contêm nenhum certificado. Os certificados provém do sistema operacional subjacente. O SDK para Ruby tem incluído pelo menos uma das CAs exigidas desde o 10 de junho de 2015. Antes dessa data, o SDK para Ruby V2 não continha certificados. Se você usar uma versão não compatível, personalizada ou modificada doAWSSDK, ou se você usar um armazenamento personalizado, você pode não ter o suporte necessário para a autoridade de certificação Amazon Trust Services.

Para validar o acesso aos endpoints do DynamoDB, você precisará desenvolver um teste que acesse a API do DynamoDB ou a API do DynamoDB Streams na região EU-WEST-3 e validar se o handshake do TLS tem êxito. Os endpoints específicos que você precisará acessar nesse teste são:

- DynamoDB: <https://dynamodb.eu-west-3.amazonaws.com>
- DynamoDB Streams: <https://streams.dynamodb.eu-west-3.amazonaws.com>

Se seu aplicativo não for compatível com a autoridade de certificação Amazon Trust Services, você verá uma das falhas a seguir:

- Erros de negociação de SSL/TLS

- Um longo atraso antes de o software receber um erro, o que indica uma falha de negociação de SSL/TLS. O tempo de atraso depende da estratégia de repetição e de configuração de tempo limite do cliente.

Como testar o navegador cliente

Para verificar se seu navegador consegue se conectar ao Amazon DynamoDB, abra o URL a seguir: <https://dynamodb.eu-west-3.amazonaws.com>. Se o teste for bem-sucedido, você verá uma mensagem como esta:

```
healthy: dynamodb.eu-west-3.amazonaws.com
```

Se o teste não for bem-sucedido, será exibido um erro semelhante a este: <https://untrusted-root.badssl.com/>.

Como atualizar o aplicativo de software cliente

Os aplicativos que acessam endpoints de API do DynamoDB ou DynamoDB Streams (seja por meio de navegadores ou programaticamente) precisarão atualizar a lista de CA confiável nas máquinas clientes se ainda não forem compatíveis com as CAs a seguir:

- Amazon Root CA 1
- Starfield Services Root Certificate Authority – G2
- Starfield Class 2 Certification Authority

Se os clientes já confiarem em QUALQUER uma das três CAs acima, confiarão nos certificados usados pelo DynamoDB e nenhuma ação será necessária. Contudo, se os clientes ainda não confiarem em nenhuma das CAs acima, as conexões HTTPS com as APIs do DynamoDB ou do DynamoDB Streams falharão. Para obter mais informações, visite esta publicação de blog: <https://aws.amazon.com/blogs/security/how-to-prepare-for-aws-move-to-its-own-certificate-authority/>.

Como atualizar o navegador cliente

Você pode atualizar o pacote de certificado em seu navegador simplesmente atualizando seu navegador. Instruções para os navegadores mais comuns podem ser encontradas no site do respectivo navegador:

- Chrome: <https://support.google.com/chrome/answer/95414?hl=en>
- Firefox: <https://support.mozilla.org/en-US/kb/update-firefox-latest-version>
- Safari: <https://support.apple.com/en-us/HT204416>
- Internet Explorer: <https://support.microsoft.com/en-us/help/17295/windows-internet-explorer-which-version#ie=other>

Como atualizar manualmente seu pacote de certificado

Se você não conseguir acessar a API do DynamoDB API ou do DynamoDB Streams, precisará atualizar seu pacote de certificado. Para isso, você precisará importar pelo menos uma das CAs exigidas. Você pode encontrá-las em <https://www.amazontrust.com/repository>.

Os seguintes sistemas operacionais e linguagens de programação são compatíveis com os certificados da Amazon Trust Services:

- Versões do Microsoft Windows com atualizações em janeiro de 2005 ou mais recentes instaladas, Windows Vista, Windows 7, Windows Server 2008 e versões mais recentes.
- MacOS X 10.4 com Java para macOS X 10.4 Release 5, macOS X 10.5 e versões mais recentes.
- Red Hat Enterprise Linux 5 (março de 2007), Linux 6 e Linux 7 e CentOS 5, CentOS 6 e CentOS 7
- Ubuntu 8.10
- Debian 5.0
- Amazon Linux (todas as versões)
- Java 1.4.2_12, Java 5 atualização 2 e todas as versões mais recentes, incluindo Java 6, Java 7 e Java 8

Se ainda assim não conseguir se conectar, consulte a documentação do software ou o fornecedor do SO ou entre em contato com o AWS Suporte para <https://aws.amazon.com/support> Para obter mais ajuda.

Tabelas e dados de exemplo

O Guia do desenvolvedor do Amazon DynamoDB usa tabelas de exemplo para ilustrar diversos aspectos do DynamoDB.

Nome da tabela	Chave primária
ProductCatalog	Chave primária simples: <ul style="list-style-type: none">• <code>Id</code> (telefone)
Forum	Chave primária simples: <ul style="list-style-type: none">• <code>Name</code> (String)
Thread	Chave primária composta: <ul style="list-style-type: none">• <code>ForumName</code> (String)• <code>Subject</code> (String)
Responder	Chave primária composta: <ul style="list-style-type: none">• <code>Id</code> (String)• <code>ReplyDateTime</code> (String)

O ResponderA tabela tem um índice secundário global chamado `PostedBy-Message-Index`. Esse índice facilitará as consultas em dois atributos que não são de chave da tabela `Reply`.

Nome do índice	Chave primária
<code>PostedBy-Message-Index</code>	Chave primária composta: <ul style="list-style-type: none">• <code>PostedBy</code> (String)• <code>Message</code> (String)

Para obter mais informações sobre essas tabelas, consulte [Caso de uso 1: Catálogo de produtos \(p. 329\)](#) e [Caso de uso 2: Aplicativo de fórum \(p. 330\)](#).

Arquivos de dados de exemplo

Tópicos

- [ProductCatalogDados de exemplo \(p. 1071\)](#)
- [ForumDados de exemplo \(p. 1076\)](#)
- [ThreadDados de exemplo \(p. 1076\)](#)
- [ReplyDados de exemplo \(p. 1078\)](#)

As seções a seguir mostram os arquivos de dados de exemplo que são usados para carregar as tabelas ProductCatalog, Forum, Thread e Reply.

Cada arquivo de dados contém vários `PutRequest`. Cada um dos quais contêm um único item. Esses elementos `PutRequest` são usados como entrada para a operação `BatchWriteItem`, usando AWS Command Line Interface (AWS CLI).

Para obter mais informações, consulte [Etapa 2: Carregar dados em tabelas \(p. 332\)](#) em [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#).

ProductCatalogDados de exemplo

```
{  
    "ProductCatalog": [  
        {  
            "PutRequest": {  
                "Item": {  
                    "Id": {  
                        "N": "101"  
                    },  
                    "Title": {  
                        "S": "Book 101 Title"  
                    },  
                    "ISBN": {  
                        "S": "111-1111111111"  
                    },  
                    "Authors": {  
                        "L": [  
                            {  
                                "S": "Author1"  
                            }  
                        ]  
                    },  
                    "Price": {  
                        "N": "2"  
                    },  
                    "Dimensions": {  
                        "S": "8.5 x 11.0 x 0.5"  
                    },  
                    "PageCount": {  
                        "N": "500"  
                    },  
                    "InPublication": {  
                        "BOOL": true  
                    },  
                    "ProductCategory": {  
                        "S": "Book"  
                    }  
                }  
            }  
        }  
    ]  
},
```

```
{
    "PutRequest": {
        "Item": {
            "Id": {
                "N": "102"
            },
            "Title": {
                "S": "Book 102 Title"
            },
            "ISBN": {
                "S": "222-2222222222"
            },
            "Authors": {
                "L": [
                    {
                        "S": "Author1"
                    },
                    {
                        "S": "Author2"
                    }
                ]
            },
            "Price": {
                "N": "20"
            },
            "Dimensions": {
                "S": "8.5 x 11.0 x 0.8"
            },
            "PageCount": {
                "N": "600"
            },
            "InPublication": {
                "BOOL": true
            },
            "ProductCategory": {
                "S": "Book"
            }
        }
    }
},
{
    "PutRequest": {
        "Item": {
            "Id": {
                "N": "103"
            },
            "Title": {
                "S": "Book 103 Title"
            },
            "ISBN": {
                "S": "333-3333333333"
            },
            "Authors": {
                "L": [
                    {
                        "S": "Author1"
                    },
                    {
                        "S": "Author2"
                    }
                ]
            },
            "Price": {
                "N": "2000"
            },
            "Dimensions": {
                "S": "10.0 x 12.0 x 1.0"
            }
        }
    }
}
```

```
        "S": "8.5 x 11.0 x 1.5"
    },
    "PageCount": {
        "N": "600"
    },
    "InPublication": {
        "BOOL": false
    },
    "ProductCategory": {
        "S": "Book"
    }
}
},
{
    "PutRequest": {
        "Item": {
            "Id": {
                "N": "201"
            },
            "Title": {
                "S": "18-Bike-201"
            },
            "Description": {
                "S": "201 Description"
            },
            "BicycleType": {
                "S": "Road"
            },
            "Brand": {
                "S": "Mountain A"
            },
            "Price": {
                "N": "100"
            },
            "Color": {
                "L": [
                    {
                        "S": "Red"
                    },
                    {
                        "S": "Black"
                    }
                ]
            },
            "ProductCategory": {
                "S": "Bicycle"
            }
        }
    }
},
{
    "PutRequest": {
        "Item": {
            "Id": {
                "N": "202"
            },
            "Title": {
                "S": "21-Bike-202"
            },
            "Description": {
                "S": "202 Description"
            },
            "BicycleType": {
                "S": "Road"
            }
        }
    }
}
```

```
    "Brand": {
        "S": "Brand-Company A"
    },
    "Price": {
        "N": "200"
    },
    "Color": {
        "L": [
            {
                "S": "Green"
            },
            {
                "S": "Black"
            }
        ]
    },
    "ProductCategory": {
        "S": "Bicycle"
    }
}
},
{
    "PutRequest": {
        "Item": {
            "Id": {
                "N": "203"
            },
            "Title": {
                "S": "19-Bike-203"
            },
            "Description": {
                "S": "203 Description"
            },
            "BicycleType": {
                "S": "Road"
            },
            "Brand": {
                "S": "Brand-Company B"
            },
            "Price": {
                "N": "300"
            },
            "Color": {
                "L": [
                    {
                        "S": "Red"
                    },
                    {
                        "S": "Green"
                    },
                    {
                        "S": "Black"
                    }
                ]
            },
            "ProductCategory": {
                "S": "Bicycle"
            }
        }
    }
},
{
    "PutRequest": {
        "Item": {
            "Id": {
```

```
        "N": "204"
    },
    "Title": {
        "S": "18-Bike-204"
    },
    "Description": {
        "S": "204 Description"
    },
    "BicycleType": {
        "S": "Mountain"
    },
    "Brand": {
        "S": "Brand-Company B"
    },
    "Price": {
        "N": "400"
    },
    "Color": {
        "L": [
            {
                "S": "Red"
            }
        ]
    },
    "ProductCategory": {
        "S": "Bicycle"
    }
}
},
{
    "PutRequest": {
        "Item": {
            "Id": {
                "N": "205"
            },
            "Title": {
                "S": "18-Bike-204"
            },
            "Description": {
                "S": "205 Description"
            },
            "BicycleType": {
                "S": "Hybrid"
            },
            "Brand": {
                "S": "Brand-Company C"
            },
            "Price": {
                "N": "500"
            },
            "Color": {
                "L": [
                    {
                        "S": "Red"
                    },
                    {
                        "S": "Black"
                    }
                ]
            },
            "ProductCategory": {
                "S": "Bicycle"
            }
        }
    }
}
```

```
        }
    ]
}
```

ForumDados de exemplo

```
{
    "Forum": [
        {
            "PutRequest": {
                "Item": {
                    "Name": {"S": "Amazon DynamoDB"},
                    "Category": {"S": "Amazon Web Services"},
                    "Threads": {"N": "2"},
                    "Messages": {"N": "4"},
                    "Views": {"N": "1000"}
                }
            }
        },
        {
            "PutRequest": {
                "Item": {
                    "Name": {"S": "Amazon S3"},
                    "Category": {"S": "Amazon Web Services"}
                }
            }
        }
    ]
}
```

ThreadDados de exemplo

```
{
    "Thread": [
        {
            "PutRequest": {
                "Item": {
                    "ForumName": {
                        "S": "Amazon DynamoDB"
                    },
                    "Subject": {
                        "S": "DynamoDB Thread 1"
                    },
                    "Message": {
                        "S": "DynamoDB thread 1 message"
                    },
                    "LastPostedBy": {
                        "S": "User A"
                    },
                    "LastPostedDateTime": {
                        "S": "2015-09-22T19:58:22.514Z"
                    },
                    "Views": {
                        "N": "0"
                    },
                    "Replies": {
                        "N": "0"
                    },
                    "Answered": {
                        "N": "0"
                    },
                    "Tags": {

```

```
        "L": [
            {
                "S": "index"
            },
            {
                "S": "primarykey"
            },
            {
                "S": "table"
            }
        ]
    }
},
{
    "PutRequest": {
        "Item": {
            "ForumName": {
                "S": "Amazon DynamoDB"
            },
            "Subject": {
                "S": "DynamoDB Thread 2"
            },
            "Message": {
                "S": "DynamoDB thread 2 message"
            },
            "LastPostedBy": {
                "S": "User A"
            },
            "LastPostedDateTime": {
                "S": "2015-09-15T19:58:22.514Z"
            },
            "Views": {
                "N": "3"
            },
            "Replies": {
                "N": "0"
            },
            "Answered": {
                "N": "0"
            },
            "Tags": {
                "L": [
                    {
                        "S": "items"
                    },
                    {
                        "S": "attributes"
                    },
                    {
                        "S": "throughput"
                    }
                ]
            }
        }
    },
    "PutRequest": {
        "Item": {
            "ForumName": {
                "S": "Amazon S3"
            },
            "Subject": {
                "S": "S3 Thread 1"
            }
        }
    }
}
```

```
        },
        "Message": {
            "S": "S3 thread 1 message"
        },
        "LastPostedBy": {
            "S": "User A"
        },
        "LastPostedDateTime": {
            "S": "2015-09-29T19:58:22.514Z"
        },
        "Views": {
            "N": "0"
        },
        "Replies": {
            "N": "0"
        },
        "Answered": {
            "N": "0"
        },
        "Tags": {
            "L": [
                {
                    "S": "largeobjects"
                },
                {
                    "S": "multipart upload"
                }
            ]
        }
    }
}
]
```

ReplyDados de exemplo

```
{
    "Reply": [
        {
            "PutRequest": {
                "Item": {
                    "Id": {
                        "S": "Amazon DynamoDB#DynamoDB Thread 1"
                    },
                    "ReplyDateTime": {
                        "S": "2015-09-15T19:58:22.947Z"
                    },
                    "Message": {
                        "S": "DynamoDB Thread 1 Reply 1 text"
                    },
                    "PostedBy": {
                        "S": "User A"
                    }
                }
            }
        },
        {
            "PutRequest": {
                "Item": {
                    "Id": {
                        "S": "Amazon DynamoDB#DynamoDB Thread 1"
                    },
                    "ReplyDateTime": {

```

```
        "S": "2015-09-22T19:58:22.947Z"
    },
    "Message": {
        "S": "DynamoDB Thread 1 Reply 2 text"
    },
    "PostedBy": {
        "S": "User B"
    }
}
},
{
    "PutRequest": {
        "Item": {
            "Id": {
                "S": "Amazon DynamoDB#DynamoDB Thread 2"
            },
            "ReplyDateTime": {
                "S": "2015-09-29T19:58:22.947Z"
            },
            "Message": {
                "S": "DynamoDB Thread 2 Reply 1 text"
            },
            "PostedBy": {
                "S": "User A"
            }
        }
    }
},
{
    "PutRequest": {
        "Item": {
            "Id": {
                "S": "Amazon DynamoDB#DynamoDB Thread 2"
            },
            "ReplyDateTime": {
                "S": "2015-10-05T19:58:22.947Z"
            },
            "Message": {
                "S": "DynamoDB Thread 2 Reply 2 text"
            },
            "PostedBy": {
                "S": "User A"
            }
        }
    }
}
]
```

Criar tabelas de exemplo e carregar dados

Tópicos

- [Como criar tabelas de exemplo e carregar dados usando o AWS SDK for Java \(p. 1080\)](#)
- [Como criar tabelas de exemplo e carregar dados usando o AWS SDK for .NET \(p. 1086\)](#)

Dentro[Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#) Primeiro você cria tabelas usando o console do DynamoDB e depois usa o AWS CLI para adicionar dados às tabelas. Este apêndice fornece o código para criar as tabelas e adicionar dados programaticamente.

Como criar tabelas de exemplo e carregar dados usando o AWS SDK for Java

O seguinte exemplo de código Java cria tabelas e carrega dados nelas. A estrutura de tabela e os dados resultantes são mostrados em [Criar tabelas e carregar dados para exemplos de código no DynamoDB \(p. 329\)](#). Para obter instruções passo a passo para executar esse código usando o Eclipse, consulte [Exemplos de código Java \(p. 334\)](#).

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * This file is licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License. A copy of
 * the License is located at
 *
 * http://aws.amazon.com/apache2.0/
 *
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.amazonaws.codesamples;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.HashSet;
import java.util.TimeZone;

import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Item;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.LocalSecondaryIndex;
import com.amazonaws.services.dynamodbv2.model.Projection;
import com.amazonaws.services.dynamodbv2.model.ProjectionType;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;

public class CreateTablesLoadData {

    static AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().build();
    static DynamoDB dynamoDB = new DynamoDB(client);

    static SimpleDateFormat dateFormatter = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss.SSS'Z'");

    static String productCatalogTableName = "ProductCatalog";
    static String forumTableName = "Forum";
    static String threadTableName = "Thread";
    static String replyTableName = "Reply";

    public static void main(String[] args) throws Exception {

        try {
```

```
deleteTable(productCatalogTableName);
deleteTable(forumTableName);
deleteTable(threadTableName);
deleteTable(replyTableName);

// Parameter1: table name
// Parameter2: reads per second
// Parameter3: writes per second
// Parameter4/5: partition key and data type
// Parameter6/7: sort key and data type (if applicable)

createTable(productCatalogTableName, 10L, 5L, "Id", "N");
createTable(forumTableName, 10L, 5L, "Name", "S");
createTable(threadTableName, 10L, 5L, "ForumName", "S", "Subject", "S");
createTable(replyTableName, 10L, 5L, "Id", "S", "ReplyDateTime", "S");

loadSampleProducts(productCatalogTableName);
loadSampleForums(forumTableName);
loadSampleThreads(threadTableName);
loadSampleReplies(replyTableName);

}

catch (Exception e) {
    System.err.println("Program failed:");
    System.err.println(e.getMessage());
}
System.out.println("Success.");
}

private static void deleteTable(String tableName) {
    Table table = dynamoDB.getTable(tableName);
    try {
        System.out.println("Issuing DeleteTable request for " + tableName);
        table.delete();
        System.out.println("Waiting for " + tableName + " to be deleted...this may take
a while...");  

        table.waitForDelete();

    }
    catch (Exception e) {
        System.err.println("DeleteTable request failed for " + tableName);
        System.err.println(e.getMessage());
    }
}

private static void createTable(String tableName, long readCapacityUnits, long
writeCapacityUnits,
String partitionKeyName, String partitionKeyType) {

    createTable(tableName, readCapacityUnits, writeCapacityUnits, partitionKeyName,
partitionKeyType, null, null);
}

private static void createTable(String tableName, long readCapacityUnits, long
writeCapacityUnits,
String partitionKeyName, String partitionKeyType, String sortKeyName, String
sortKeyType) {

    try {

        ArrayList<KeySchemaElement> keySchema = new ArrayList<KeySchemaElement>();
        keySchema.add(new
KeySchemaElement().withAttributeName(partitionKeyName).withKeyType(KeyType.HASH)); //  

Partition
```

```
// key

ArrayList<AttributeDefinition> attributeDefinitions = new
ArrayList<AttributeDefinition>();
attributeDefinitions
    .add(new
AttributeDefinition().withAttributeName(partitionKeyName).withAttributeType(partitionKeyType));

if (sortKeyName != null) {
    keySchema.add(new
KeySchemaElement().withAttributeName(sortKeyName).withKeyType(KeyType.RANGE)); // Sort

    // key
    attributeDefinitions
        .add(new
AttributeDefinition().withAttributeName(sortKeyName).withAttributeType(sortKeyType));
}

CreateTableRequest request = new
CreateTableRequest().withTableName(tableName).withKeySchema(keySchema)
    .withProvisionedThroughput(new
ProvisionedThroughput().withReadCapacityUnits(readCapacityUnits)
    .withWriteCapacityUnits(writeCapacityUnits));

// If this is the Reply table, define a local secondary index
if (replyTableName.equals(tableName)) {

    attributeDefinitions
        .add(new
AttributeDefinition().withAttributeName("PostedBy").withAttributeType("S"));

    ArrayList<LocalSecondaryIndex> localSecondaryIndexes = new
ArrayList<LocalSecondaryIndex>();
    localSecondaryIndexes.add(new
LocalSecondaryIndex().withIndexName("PostedBy-Index")
    .withKeySchema(new
KeySchemaElement().withAttributeName(partitionKeyName).withKeyType(KeyType.HASH), // Partition

        // key
        new
KeySchemaElement().withAttributeName("PostedBy").withKeyType(KeyType.RANGE)) // Sort

    // key
    .withProjection(new
Projection().withProjectionType(ProjectionType.KEYS_ONLY)));

    request.setLocalSecondaryIndexes(localSecondaryIndexes);
}

request.setAttributeDefinitions(attributeDefinitions);

System.out.println("Issuing CreateTable request for " + tableName);
Table table = dynamoDB.createTable(request);
System.out.println("Waiting for " + tableName + " to be created...this may take
a while...");
table.waitForActive();

}
catch (Exception e) {
    System.err.println("CreateTable request failed for " + tableName);
    System.err.println(e.getMessage());
}
}
```

```
private static void loadSampleProducts(String tableName) {  
  
    Table table = dynamoDB.getTable(tableName);  
  
    try {  
  
        System.out.println("Adding data to " + tableName);  
  
        Item item = new Item().withPrimaryKey("Id", 101).withString("Title", "Book 101  
Title")  
            .withString("ISBN", "111-1111111111")  
            .withStringSet("Authors", new  
HashSet<String>(Arrays.asList("Author1"))).withNumber("Price", 2)  
            .withString("Dimensions", "8.5 x 11.0 x 0.5").withNumber("PageCount", 500)  
            .withBoolean("InPublication", true).withString("ProductCategory", "Book");  
        table.putItem(item);  
  
        item = new Item().withPrimaryKey("Id", 102).withString("Title", "Book 102  
Title")  
            .withString("ISBN", "222-2222222222")  
            .withStringSet("Authors", new HashSet<String>(Arrays.asList("Author1",  
"Author2")))  
            .withNumber("Price", 20).withString("Dimensions", "8.5 x 11.0 x  
0.8").withNumber("PageCount", 600)  
            .withBoolean("InPublication", true).withString("ProductCategory", "Book");  
        table.putItem(item);  
  
        item = new Item().withPrimaryKey("Id", 103).withString("Title", "Book 103  
Title")  
            .withString("ISBN", "333-3333333333")  
            .withStringSet("Authors", new HashSet<String>(Arrays.asList("Author1",  
"Author2")))  
            // Intentional. Later we'll run Scan to find price error. Find  
            // items > 1000 in price.  
            .withNumber("Price", 2000).withString("Dimensions", "8.5 x 11.0 x  
1.5").withNumber("PageCount", 600)  
            .withBoolean("InPublication", false).withString("ProductCategory", "Book");  
        table.putItem(item);  
  
        // Add bikes.  
  
        item = new Item().withPrimaryKey("Id", 201).withString("Title", "18-Bike-201")  
            // Size, followed by some title.  
            .withString("Description", "201 Description").withString("BicycleType",  
"Road")  
            .withString("Brand", "Mountain A")  
            // Trek, Specialized.  
            .withNumber("Price", 100).withStringSet("Color", new  
HashSet<String>(Arrays.asList("Red", "Black")))  
            .withString("ProductCategory", "Bicycle");  
        table.putItem(item);  
  
        item = new Item().withPrimaryKey("Id", 202).withString("Title", "21-Bike-202")  
            .withString("Description", "202 Description").withString("BicycleType",  
"Road")  
            .withString("Brand", "Brand-Company A").withNumber("Price", 200)  
            .withStringSet("Color", new HashSet<String>(Arrays.asList("Green",  
"Black")))  
            .withString("ProductCategory", "Bicycle");  
        table.putItem(item);  
  
        item = new Item().withPrimaryKey("Id", 203).withString("Title", "19-Bike-203")  
            .withString("Description", "203 Description").withString("BicycleType",  
"Road")  
            .withString("Brand", "Brand-Company B").withNumber("Price", 300)  
    }  
}
```

```
        .withStringSet("Color", new HashSet<String>(Arrays.asList("Red", "Green",
    "Black")))
        .withString("ProductCategory", "Bicycle");
    table.putItem(item);

    item = new Item().withPrimaryKey("Id", 204).withString("Title", "18-Bike-204")
        .withString("Description", "204 Description").withString("BicycleType",
    "Mountain")
        .withString("Brand", "Brand-Company B").withNumber("Price", 400)
        .withStringSet("Color", new HashSet<String>(Arrays.asList("Red")))
        .withString("ProductCategory", "Bicycle");
    table.putItem(item);

    item = new Item().withPrimaryKey("Id", 205).withString("Title", "20-Bike-205")
        .withString("Description", "205 Description").withString("BicycleType",
    "Hybrid")
        .withString("Brand", "Brand-Company C").withNumber("Price", 500)
        .withStringSet("Color", new HashSet<String>(Arrays.asList("Red", "Black")))
        .withString("ProductCategory", "Bicycle");
    table.putItem(item);

}
catch (Exception e) {
    System.err.println("Failed to create item in " + tableName);
    System.err.println(e.getMessage());
}

}

private static void loadSampleForums(String tableName) {

    Table table = dynamoDB.getTable(tableName);

    try {

        System.out.println("Adding data to " + tableName);

        Item item = new Item().withPrimaryKey("Name", "Amazon DynamoDB")
            .withString("Category", "Amazon Web Services").withNumber("Threads",
        2).withNumber("Messages", 4)
            .withNumber("Views", 1000);
        table.putItem(item);

        item = new Item().withPrimaryKey("Name", "Amazon S3").withString("Category",
    "Amazon Web Services")
            .withNumber("Threads", 0);
        table.putItem(item);

    }
    catch (Exception e) {
        System.err.println("Failed to create item in " + tableName);
        System.err.println(e.getMessage());
    }
}

private static void loadSampleThreads(String tableName) {
    try {
        long time1 = (new Date()).getTime() - (7 * 24 * 60 * 60 * 1000); // 7
        // days
        // ago
        long time2 = (new Date()).getTime() - (14 * 24 * 60 * 60 * 1000); // 14
        // days
        // ago
        long time3 = (new Date()).getTime() - (21 * 24 * 60 * 60 * 1000); // 21
        // days
        // ago
    }
}
```

```
Date date1 = new Date();
date1.setTime(time1);

Date date2 = new Date();
date2.setTime(time2);

Date date3 = new Date();
date3.setTime(time3);

dateFormatter.setTimeZone(TimeZone.getTimeZone("UTC"));

Table table = dynamoDB.getTable(tableName);

System.out.println("Adding data to " + tableName);

Item item = new Item().withPrimaryKey("ForumName", "Amazon DynamoDB")
    .withString("Subject", "DynamoDB Thread 1").withString("Message", "DynamoDB
thread 1 message")
    .withString("LastPostedBy", "User A").withString("LastPostedDateTime",
dateFormatter.format(date2))
    .withNumber("Views", 0).withNumber("Replies", 0).withNumber("Answered", 0)
    .withStringSet("Tags", new HashSet<String>(Arrays.asList("index",
"primarykey", "table")));
table.putItem(item);

item = new Item().withPrimaryKey("ForumName", "Amazon
DynamoDB").withString("Subject", "DynamoDB Thread 2")
    .withString("Message", "DynamoDB thread 2
message").withString("LastPostedBy", "User A")
    .withString("LastPostedDateTime",
dateFormatter.format(date3)).withNumber("Views", 0)
    .withNumber("Replies", 0).withNumber("Answered", 0)
    .withStringSet("Tags", new HashSet<String>(Arrays.asList("index",
"partitionkey", "sortkey")));
table.putItem(item);

item = new Item().withPrimaryKey("ForumName", "Amazon
S3").withString("Subject", "S3 Thread 1")
    .withString("Message", "S3 Thread 3 message").withString("LastPostedBy",
"User A")
    .withString("LastPostedDateTime",
dateFormatter.format(date1)).withNumber("Views", 0)
    .withNumber("Replies", 0).withNumber("Answered", 0)
    .withStringSet("Tags", new HashSet<String>(Arrays.asList("largeobjects",
"multipart upload")));
table.putItem(item);

}

catch (Exception e) {
    System.err.println("Failed to create item in " + tableName);
    System.err.println(e.getMessage());
}

}

private static void loadSampleReplies(String tableName) {
    try {
        // 1 day ago
        long time0 = (new Date()).getTime() - (1 * 24 * 60 * 60 * 1000);
        // 7 days ago
        long time1 = (new Date()).getTime() - (7 * 24 * 60 * 60 * 1000);
        // 14 days ago
        long time2 = (new Date()).getTime() - (14 * 24 * 60 * 60 * 1000);
        // 21 days ago
        long time3 = (new Date()).getTime() - (21 * 24 * 60 * 60 * 1000);
    }
}
```

```
Date date0 = new Date();
date0.setTime(time0);

Date date1 = new Date();
date1.setTime(time1);

Date date2 = new Date();
date2.setTime(time2);

Date date3 = new Date();
date3.setTime(time3);

dateFormatter.setTimeZone(TimeZone.getTimeZone("UTC"));

Table table = dynamoDB.getTable(tableName);

System.out.println("Adding data to " + tableName);

// Add threads.

Item item = new Item().withPrimaryKey("Id", "Amazon DynamoDB#DynamoDB Thread 1")
    .withString("ReplyDateTime", (dateFormatter.format(date3)))
    .withString("Message", "DynamoDB Thread 1 Reply 1")
    .withString("PostedBy", "User A");
table.putItem(item);

item = new Item().withPrimaryKey("Id", "Amazon DynamoDB#DynamoDB Thread 1")
    .withString("ReplyDateTime", dateFormatter.format(date2))
    .withString("Message", "DynamoDB Thread 1 Reply 2")
    .withString("PostedBy", "User B");
table.putItem(item);

item = new Item().withPrimaryKey("Id", "Amazon DynamoDB#DynamoDB Thread 2")
    .withString("ReplyDateTime", dateFormatter.format(date1))
    .withString("Message", "DynamoDB Thread 2 Reply 1")
    .withString("PostedBy", "User A");
table.putItem(item);

item = new Item().withPrimaryKey("Id", "Amazon DynamoDB#DynamoDB Thread 2")
    .withString("ReplyDateTime", dateFormatter.format(date0))
    .withString("Message", "DynamoDB Thread 2 Reply 2")
    .withString("PostedBy", "User A");
table.putItem(item);

}

catch (Exception e) {
    System.err.println("Failed to create item in " + tableName);
    System.err.println(e.getMessage());
}

}

}
```

Como criar tabelas de exemplo e carregar dados usando o AWS SDK for .NET

O seguinte exemplo de código C# cria tabelas e carrega dados nelas. A estrutura de tabela e os dados resultantes são mostrados em [Criar tabelas e carregar dados para exemplos de código no](#)

DynamoDB (p. 329). Para obter instruções passo a passo para executar esse código no Visual Studio, consulte [Exemplos de código .NET](#) (p. 336).

```
/**  
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * This file is licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License. A copy of  
 * the License is located at  
 *  
 * http://aws.amazon.com/apache2.0/  
 *  
 * This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
 * CONDITIONS OF ANY KIND, either express or implied. See the License for the  
 * specific language governing permissions and limitations under the License.  
 */  
using System;  
using System.Collections.Generic;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.DocumentModel;  
using Amazon.DynamoDBv2.Model;  
using Amazon.Runtime;  
using Amazon.SecurityToken;  
  
namespace com.amazonaws.codesamples  
{  
    class CreateTablesLoadData  
    {  
        private static AmazonDynamoDBClient client = new AmazonDynamoDBClient();  
  
        static void Main(string[] args)  
        {  
            try  
            {  
                //DeleteAllTables(client);  
                DeleteTable("ProductCatalog");  
                DeleteTable("Forum");  
                DeleteTable("Thread");  
                DeleteTable("Reply");  
  
                // Create tables (using the AWS SDK for .NET low-level API).  
                CreateTableProductCatalog();  
                CreateTableForum();  
                CreateTableThread(); // ForumTitle, Subject */  
                CreateTableReply();  
  
                // Load data (using the .NET SDK document API)  
                LoadSampleProducts();  
                LoadSampleForums();  
                LoadSampleThreads();  
                LoadSampleReplies();  
                Console.WriteLine("Sample complete!");  
                Console.WriteLine("Press ENTER to continue");  
                Console.ReadLine();  
            }  
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }  
            catch (Exception e) { Console.WriteLine(e.Message); }  
        }  
  
        private static void DeleteTable(string tableName)  
        {  
            try  
            {  
                var deleteTableResponse = client.DeleteTable(new DeleteTableRequest()  
            }  
        }  
    }  
}
```

```
        {
            TableName = tableName
        });
        WaitTillTableDeleted(client, tableName, deleteTableResponse);
    }
    catch (ResourceNotFoundException)
    {
        // There is no such table.
    }
}

private static void CreateTableProductCatalog()
{
    string tableName = "ProductCatalog";

    var response = client.CreateTable(new CreateTableRequest
    {
        TableName = tableName,
        AttributeDefinitions = new List<AttributeDefinition>()
        {
            new AttributeDefinition
            {
                AttributeName = "Id",
                AttributeType = "N"
            }
        },
        KeySchema = new List<KeySchemaElement>()
        {
            new KeySchemaElement
            {
                AttributeName = "Id",
                KeyType = "HASH"
            }
        },
        ProvisionedThroughput = new ProvisionedThroughput
        {
            ReadCapacityUnits = 10,
            WriteCapacityUnits = 5
        }
    });
    WaitTillTableCreated(client, tableName, response);
}

private static void CreateTableForum()
{
    string tableName = "Forum";

    var response = client.CreateTable(new CreateTableRequest
    {
        TableName = tableName,
        AttributeDefinitions = new List<AttributeDefinition>()
        {
            new AttributeDefinition
            {
                AttributeName = "Name",
                AttributeType = "S"
            }
        },
        KeySchema = new List<KeySchemaElement>()
        {
            new KeySchemaElement
            {
                AttributeName = "Name", // forum Title
                KeyType = "HASH"
            }
        }
    });
}
```

```
        },
        ProvisionedThroughput = new ProvisionedThroughput
    {
        ReadCapacityUnits = 10,
        WriteCapacityUnits = 5
    }
});

WaitTillTableCreated(client, tableName, response);
}

private static void CreateTableThread()
{
    string tableName = "Thread";

    var response = client.CreateTable(new CreateTableRequest
    {
        TableName = tableName,
        AttributeDefinitions = new List<AttributeDefinition>()
        {
            new AttributeDefinition
            {
                AttributeName = "ForumName", // Hash attribute
                AttributeType = "S"
            },
            new AttributeDefinition
            {
                AttributeName = "Subject",
                AttributeType = "S"
            }
        },
        KeySchema = new List<KeySchemaElement>()
        {
            new KeySchemaElement
            {
                AttributeName = "ForumName", // Hash attribute
                KeyType = "HASH"
            },
            new KeySchemaElement
            {
                AttributeName = "Subject", // Range attribute
                KeyType = "RANGE"
            }
        },
        ProvisionedThroughput = new ProvisionedThroughput
    {
        ReadCapacityUnits = 10,
        WriteCapacityUnits = 5
    }
});

WaitTillTableCreated(client, tableName, response);
}

private static void CreateTableReply()
{
    string tableName = "Reply";
    var response = client.CreateTable(new CreateTableRequest
    {
        TableName = tableName,
        AttributeDefinitions = new List<AttributeDefinition>()
        {
            new AttributeDefinition
            {
                AttributeName = "Id",
                AttributeType = "S"
            }
        }
    });

    WaitTillTableCreated(client, tableName, response);
}
```

```
        },
        new AttributeDefinition
        {
            AttributeName = "ReplyDateTime",
            AttributeType = "S"
        },
        new AttributeDefinition
        {
            AttributeName = "PostedBy",
            AttributeType = "S"
        }
    },
    KeySchema = new List<KeySchemaElement>()
    {
        new KeySchemaElement()
        {
            AttributeName = "Id",
            KeyType = "HASH"
        },
        new KeySchemaElement()
        {
            AttributeName = "ReplyDateTime",
            KeyType = "RANGE"
        }
    },
    LocalSecondaryIndexes = new List<LocalSecondaryIndex>()
    {
        new LocalSecondaryIndex()
        {
            IndexName = "PostedBy_index",

            KeySchema = new List<KeySchemaElement>() {
                new KeySchemaElement() {
                    AttributeName = "Id", KeyType = "HASH"
                },
                new KeySchemaElement() {
                    AttributeName = "PostedBy", KeyType = "RANGE"
                }
            },
            Projection = new Projection() {
                ProjectionType = ProjectionType.KEYS_ONLY
            }
        }
    },
    ProvisionedThroughput = new ProvisionedThroughput
    {
        ReadCapacityUnits = 10,
        WriteCapacityUnits = 5
    }
});

WaitTillTableCreated(client, tableName, response);
}

private static void WaitTillTableCreated(AmazonDynamoDBClient client, string
tableName,
                                         CreateTableResponse response)
{
    var tableDescription = response.TableDescription;

    string status = tableDescription.TableStatus;

    Console.WriteLine(tableName + " - " + status);

    // Let us wait until table is created. Call DescribeTable.
```

```
        while (status != "ACTIVE")
    {
        System.Threading.Thread.Sleep(5000); // Wait 5 seconds.
        try
        {
            var res = client.DescribeTable(new DescribeTableRequest
            {
                TableName = tableName
            });
            Console.WriteLine("Table name: {0}, status: {1}", res.Table.TableName,
                res.Table.TableStatus);
            status = res.Table.TableStatus;
        }
        // Try-catch to handle potential eventual-consistency issue.
        catch (ResourceNotFoundException)
        {
        }
    }

    private static void WaitTillTableDeleted(AmazonDynamoDBClient client, string
tableName,
                                            DeleteTableResponse response)
{
    var tableDescription = response.TableDescription;

    string status = tableDescription.TableStatus;

    Console.WriteLine(tableName + " - " + status);

    // Let us wait until table is created. Call DescribeTable
    try
    {
        while (status == "DELETING")
        {
            System.Threading.Thread.Sleep(5000); // wait 5 seconds

            var res = client.DescribeTable(new DescribeTableRequest
            {
                TableName = tableName
            });
            Console.WriteLine("Table name: {0}, status: {1}", res.Table.TableName,
                res.Table.TableStatus);
            status = res.Table.TableStatus;
        }
    }
    catch (ResourceNotFoundException)
    {
        // Table deleted.
    }
}

private static void LoadSampleProducts()
{
    Table productCatalogTable = Table.LoadTable(client, "ProductCatalog");
    // ***** Add Books *****
    var book1 = new Document();
    book1["Id"] = 101;
    book1["Title"] = "Book 101 Title";
    book1["ISBN"] = "111-1111111111";
    book1["Authors"] = new List<string> { "Author 1" };
    book1["Price"] = -2; // *** Intentional value. Later used to illustrate scan.
    book1["Dimensions"] = "8.5 x 11.0 x 0.5";
    book1["PageCount"] = 500;
    book1["InPublication"] = true;
    book1["ProductCategory"] = "Book";
    productCatalogTable.PutItem(book1);
```

```
var book2 = new Document();

book2["Id"] = 102;
book2["Title"] = "Book 102 Title";
book2["ISBN"] = "222-2222222222";
book2["Authors"] = new List<string> { "Author 1", "Author 2" }; ;
book2["Price"] = 20;
book2["Dimensions"] = "8.5 x 11.0 x 0.8";
book2["PageCount"] = 600;
book2["InPublication"] = true;
book2["ProductCategory"] = "Book";
productCatalogTable.PutItem(book2);

var book3 = new Document();
book3["Id"] = 103;
book3["Title"] = "Book 103 Title";
book3["ISBN"] = "333-3333333333";
book3["Authors"] = new List<string> { "Author 1", "Author2", "Author 3" }; ;
book3["Price"] = 2000;
book3["Dimensions"] = "8.5 x 11.0 x 1.5";
book3["PageCount"] = 700;
book3["InPublication"] = false;
book3["ProductCategory"] = "Book";
productCatalogTable.PutItem(book3);

// ***** Add bikes. *****
var bicycle1 = new Document();
bicycle1["Id"] = 201;
bicycle1["Title"] = "18-Bike 201"; // size, followed by some title.
bicycle1["Description"] = "201 description";
bicycle1["BicycleType"] = "Road";
bicycle1["Brand"] = "Brand-Company A"; // Trek, Specialized.
bicycle1["Price"] = 100;
bicycle1["Color"] = new List<string> { "Red", "Black" };
bicycle1["ProductCategory"] = "Bike";
productCatalogTable.PutItem(bicycle1);

var bicycle2 = new Document();
bicycle2["Id"] = 202;
bicycle2["Title"] = "21-Bike 202Brand-Company A";
bicycle2["Description"] = "202 description";
bicycle2["BicycleType"] = "Road";
bicycle2["Brand"] = "";
bicycle2["Price"] = 200;
bicycle2["Color"] = new List<string> { "Green", "Black" };
bicycle2["ProductCategory"] = "Bicycle";
productCatalogTable.PutItem(bicycle2);

var bicycle3 = new Document();
bicycle3["Id"] = 203;
bicycle3["Title"] = "19-Bike 203";
bicycle3["Description"] = "203 description";
bicycle3["BicycleType"] = "Road";
bicycle3["Brand"] = "Brand-Company B";
bicycle3["Price"] = 300;
bicycle3["Color"] = new List<string> { "Red", "Green", "Black" };
bicycle3["ProductCategory"] = "Bike";
productCatalogTable.PutItem(bicycle3);

var bicycle4 = new Document();
bicycle4["Id"] = 204;
bicycle4["Title"] = "18-Bike 204";
bicycle4["Description"] = "204 description";
bicycle4["BicycleType"] = "Mountain";
bicycle4["Brand"] = "Brand-Company B";
```

```
bicycle4["Price"] = 400;
bicycle4["Color"] = new List<string> { "Red" };
bicycle4["ProductCategory"] = "Bike";
productCatalogTable.PutItem(bicycle4);

var bicycle5 = new Document();
bicycle5["Id"] = 205;
bicycle5["Title"] = "20-Title 205";
bicycle5["Description"] = "205 description";
bicycle5["BicycleType"] = "Hybrid";
bicycle5["Brand"] = "Brand-Company C";
bicycle5["Price"] = 500;
bicycle5["Color"] = new List<string> { "Red", "Black" };
bicycle5["ProductCategory"] = "Bike";
productCatalogTable.PutItem(bicycle5);
}

private static void LoadSampleForums()
{
    Table forumTable = Table.LoadTable(client, "Forum");

    var forum1 = new Document();
    forum1["Name"] = "Amazon DynamoDB"; // PK
    forum1["Category"] = "Amazon Web Services";
    forum1["Threads"] = 2;
    forum1["Messages"] = 4;
    forum1["Views"] = 1000;

    forumTable.PutItem(forum1);

    var forum2 = new Document();
    forum2["Name"] = "Amazon S3"; // PK
    forum2["Category"] = "Amazon Web Services";
    forum2["Threads"] = 1;

    forumTable.PutItem(forum2);
}

private static void LoadSampleThreads()
{
    Table threadTable = Table.LoadTable(client, "Thread");

    // Thread 1.
    var thread1 = new Document();
    thread1["ForumName"] = "Amazon DynamoDB"; // Hash attribute.
    thread1["Subject"] = "DynamoDB Thread 1"; // Range attribute.
    thread1["Message"] = "DynamoDB thread 1 message text";
    thread1["LastPostedBy"] = "User A";
    thread1["LastPostedDateTime"] = DateTime.UtcNow.Subtract(new TimeSpan(14, 0, 0, 0));
    thread1["Views"] = 0;
    thread1["Replies"] = 0;
    thread1["Answered"] = false;
    thread1["Tags"] = new List<string> { "index", "primarykey", "table" };

    threadTable.PutItem(thread1);

    // Thread 2.
    var thread2 = new Document();
    thread2["ForumName"] = "Amazon DynamoDB"; // Hash attribute.
    thread2["Subject"] = "DynamoDB Thread 2"; // Range attribute.
    thread2["Message"] = "DynamoDB thread 2 message text";
    thread2["LastPostedBy"] = "User A";
    thread2["LastPostedDateTime"] = DateTime.UtcNow.Subtract(new TimeSpan(21, 0, 0, 0));
    thread2["Views"] = 0;
```

```
        thread2["Replies"] = 0;
        thread2["Answered"] = false;
        thread2["Tags"] = new List<string> { "index", "primarykey", "rangekey" };

        threadTable.PutItem(thread2);

        // Thread 3.
        var thread3 = new Document();
        thread3["ForumName"] = "Amazon S3"; // Hash attribute.
        thread3["Subject"] = "S3 Thread 1"; // Range attribute.
        thread3["Message"] = "S3 thread 3 message text";
        thread3["LastPostedBy"] = "User A";
        thread3["LastPostedDateTime"] = DateTime.UtcNow.Subtract(new TimeSpan(7, 0, 0,
0));
        thread3["Views"] = 0;
        thread3["Replies"] = 0;
        thread3["Answered"] = false;
        thread3["Tags"] = new List<string> { "largeobjects", "multipart upload" };
        threadTable.PutItem(thread3);
    }

    private static void LoadSampleReplies()
{
    Table replyTable = Table.LoadTable(client, "Reply");

    // Reply 1 - thread 1.
    var thread1Reply1 = new Document();
    thread1Reply1["Id"] = "Amazon DynamoDB#DynamoDB Thread 1"; // Hash attribute.
    thread1Reply1["ReplyDateTime"] = DateTime.UtcNow.Subtract(new TimeSpan(21, 0,
0, 0)); // Range attribute.
    thread1Reply1["Message"] = "DynamoDB Thread 1 Reply 1 text";
    thread1Reply1["PostedBy"] = "User A";

    replyTable.PutItem(thread1Reply1);

    // Reply 2 - thread 1.
    var thread1Reply2 = new Document();
    thread1Reply2["Id"] = "Amazon DynamoDB#DynamoDB Thread 1"; // Hash attribute.
    thread1Reply2["ReplyDateTime"] = DateTime.UtcNow.Subtract(new TimeSpan(14, 0,
0, 0)); // Range attribute.
    thread1Reply2["Message"] = "DynamoDB Thread 1 Reply 2 text";
    thread1Reply2["PostedBy"] = "User B";

    replyTable.PutItem(thread1Reply2);

    // Reply 3 - thread 1.
    var thread1Reply3 = new Document();
    thread1Reply3["Id"] = "Amazon DynamoDB#DynamoDB Thread 1"; // Hash attribute.
    thread1Reply3["ReplyDateTime"] = DateTime.UtcNow.Subtract(new TimeSpan(7, 0, 0,
0)); // Range attribute.
    thread1Reply3["Message"] = "DynamoDB Thread 1 Reply 3 text";
    thread1Reply3["PostedBy"] = "User B";

    replyTable.PutItem(thread1Reply3);

    // Reply 1 - thread 2.
    var thread2Reply1 = new Document();
    thread2Reply1["Id"] = "Amazon DynamoDB#DynamoDB Thread 2"; // Hash attribute.
    thread2Reply1["ReplyDateTime"] = DateTime.UtcNow.Subtract(new TimeSpan(7, 0, 0,
0)); // Range attribute.
    thread2Reply1["Message"] = "DynamoDB Thread 2 Reply 1 text";
    thread2Reply1["PostedBy"] = "User A";

    replyTable.PutItem(thread2Reply1);
```

```
// Reply 2 - thread 2.
var thread2Reply2 = new Document();
thread2Reply2["Id"] = "Amazon DynamoDB#DynamoDB Thread 2"; // Hash attribute.
thread2Reply2["ReplyDateTime"] = DateTime.UtcNow.Subtract(new TimeSpan(1, 0, 0,
0)); // Range attribute.
thread2Reply2["Message"] = "DynamoDB Thread 2 Reply 2 text";
thread2Reply2["PostedBy"] = "User A";

replyTable.PutItem(thread2Reply2);
}
}
```

Aplicativo de exemplo do DynamoDB usando oAWS SDK for Python (Boto): Jogo da velha

Tópicos

- [Etapa 1: Implantar e testar localmente \(p. 1096\)](#)
- [Etapa 2: Analise o modelo de dados e os detalhes da implantação \(p. 1100\)](#)
- [Etapa 3: Implantar em produção usando o serviço DynamoDB \(p. 1107\)](#)
- [Etapa 4: Liberação de recursos \(p. 1114\)](#)

O Jogo da velha é um aplicativo web de exemplo criado no Amazon DynamoDB. O aplicativo usa aAWS SDK for Python (Boto)Para fazer as chamadas do DynamoDB necessárias para armazenar os dados do jogo em uma tabela do DynamoDB, e o framework da web para Python chamado Flask para ilustrar o desenvolvimento completo do aplicativo no DynamoDB, incluindo como modelar os dados. Ele também demonstra as melhores práticas quando se trata de modelagem de dados no DynamoDB, incluindo a tabela que você cria para o aplicativo de jogo, a chave primária que você define, os índices adicionais necessários de acordo com as exigências da consulta e o uso de atributos de valor concatenados.

Você joga o aplicativo Jogo da velha na web da seguinte forma:

1. Você faz login na página inicial do aplicativo.
2. Em seguida, você convida outro usuário para jogar o jogo como o seu oponente.

Até que outro usuário aceite o convite, o status do jogo permanece como PENDING. Depois que um oponente aceita o convite, o status do jogo muda para IN_PROGRESS.

3. O jogo começa depois que o oponente faz login e aceita o convite.
4. O aplicativo armazena todas as movimentações e as informações de status do jogo em uma tabela do DynamoDB.
5. O jogo termina com uma vitória ou empate, o que define o status do jogo como FINISHED.

O exercício de criação do aplicativo de ponta a ponta é descrito em etapas:

- [Etapa 1: Implantar e testar localmente \(p. 1096\)](#)– Nesta seção, você faz download, implanta e testa o aplicativo em seu computador local. Você poderá criar as tabelas necessárias na versão disponível para download do DynamoDB.
- [Etapa 2: Analise o modelo de dados e os detalhes da implantação \(p. 1100\)](#)– Esta seção primeiro descreve o modelo de dados em detalhes, incluindo os índices e o uso do atributo de valor concatenado. Em seguida, a seção explica como o aplicativo funciona.
- [Etapa 3: Implantar em produção usando o serviço DynamoDB \(p. 1107\)](#)– Esta seção aborda as considerações de implantação em produção. Nesta etapa, você cria uma tabela usando o serviço

Amazon DynamoDB e implantando o aplicativo usando o AWS Elastic Beanstalk. Quando o aplicativo está em produção, você também concede as permissões apropriadas para que o aplicativo possa acessar a tabela do DynamoDB. As instruções desta seção orientam você durante a implantação da produção de ponta a ponta.

- [Etapa 4: Liberação de recursos \(p. 1114\)](#)- Esta seção destaca as áreas que não são cobertas por este exemplo. A seção também fornece etapas para você remover o AWS Recursos que você criou nas etapas anteriores para evitar qualquer cobrança.

Etapa 1: Implantar e testar localmente

Tópicos

- [1.1: Faça download e instale os pacotes obrigatórios \(p. 1096\)](#)
- [1.2: Testar o aplicativo do jogo \(p. 1097\)](#)

Nesta etapa, você faz download, implanta e testa o aplicativo Jogo da velha em seu computador local. Em vez de usar o web service do Amazon DynamoDB, você fará download do DynamoDB para seu computador e criará a tabela necessária aqui.

1.1: Faça download e instale os pacotes obrigatórios

Para testar este aplicativo localmente, será necessário o seguinte:

- Python
- Flask (um microframework para Python)
- AWS SDK for Python (Boto)
- DynamoDB em execução em seu computador
- Git

Para aproveitar essas ferramentas, faça o seguinte:

1. Instalar o Python. Para obter instruções detalhadas, acesse [Fazer download do Python](#).

O aplicativo Jogo da velha foi testando com a versão 2.7 do Python.

2. Instale o Flask e o AWS SDK for Python (Boto) usando o Python Package Installer (PIP):

- Instale o PIP.

Para obter instruções, consulte [Instalar PIP](#). Na página de instalação, selecione o link get-pip.py e, em seguida, salve o arquivo. Em seguida, abra um terminal de comando como administrador e digite as informações a seguir no prompt de comando.

```
python.exe get-pip.py
```

No Linux, você não especifica a extensão .exe. Você só especifica python get-pip.py.

- Usando o PIP, instale os pacotes Flask e Boto usando o código a seguir.

```
pip install Flask
pip install boto
pip install configparser
```

3. Faça download do DynamoDB para o seu computador. Para obter instruções sobre como executá-lo, consulte [Configuração do DynamoDB Local \(versão disponível para download\) \(p. 50\)](#).

4. Faça download do aplicativo Jogo da velha:
 - a. Instale o Git. Para obter instruções, consulte [Downloads do git](#).
 - b. Execute o código a seguir para fazer download do aplicativo.

```
git clone https://github.com/awslabs/dynamodb-tictactoe-example-app.git
```

1.2: Testar o aplicativo do jogo

Para testar o aplicativo Jogo da velha, você precisa executar o DynamoDB localmente no seu computador.

Para executar o aplicativo Jogo da velha

1. Inicie o DynamoDB.
2. Inicie o servidor web do aplicativo Jogo da velha.

Para isso, abra um terminal de comando, navegue para a pasta na qual você fez download do aplicativo Jogo da velha e execute o aplicativo localmente usando o código a seguir.

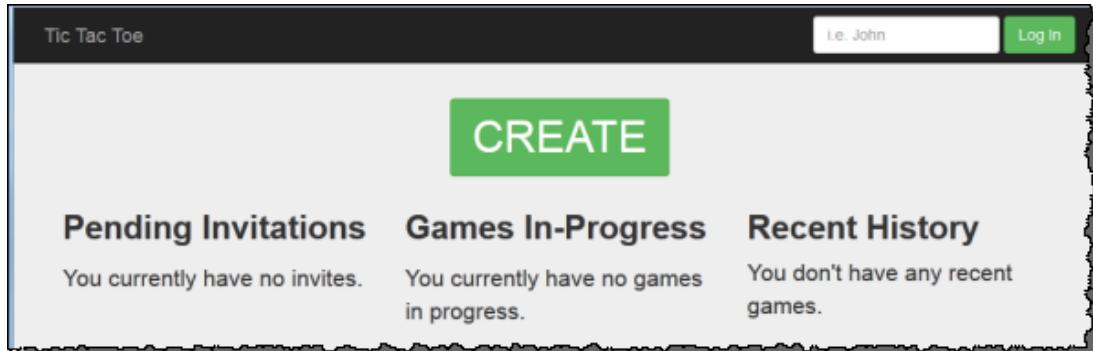
```
python.exe application.py --mode local --serverPort 5000 --port 8000
```

No Linux, você não especifica a extensão .exe.

3. Abra seu navegador da web e digite as informações a seguir.

```
http://localhost:5000/
```

O navegador mostra a página inicial.

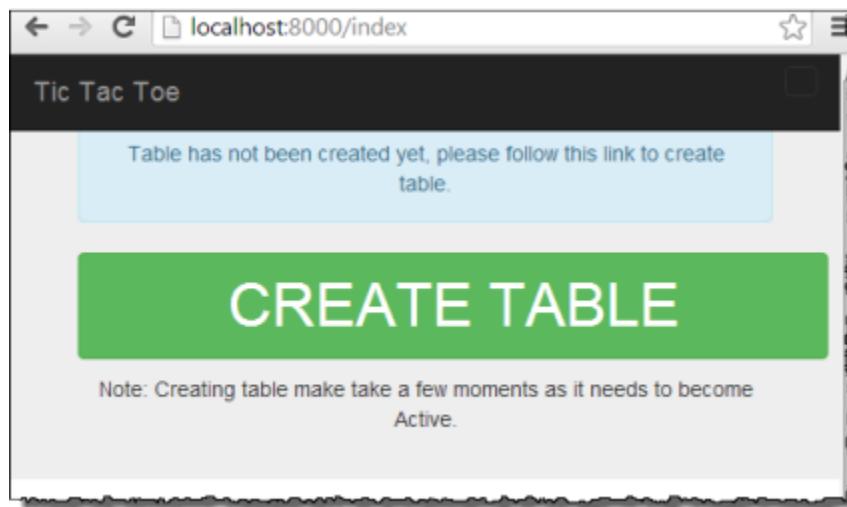


4. Digite **user1** na caixa Log in (Login) para fazer login como user1.

Note

Este aplicativo de exemplo não realiza autenticação de usuário. O ID de usuário só é usado para identificar os jogadores. Se dois jogadores fizerem login com o mesmo alias, o aplicativo funciona como se você estivesse jogando em dois navegadores diferentes.

5. Se esta for a primeira vez que você está jogando o jogo, aparecerá uma página solicitando a criação da tabela obrigatória (Games) no DynamoDB. Selecione CREATE TABLE (Criar tabela).



6. Selecione CREATE (Criar) para criar o primeiro jogo da velha.
7. Digite **user2** na caixa Choose an Opponent (Escolher um oponente) e selecione Create Game! (Criar jogo!).

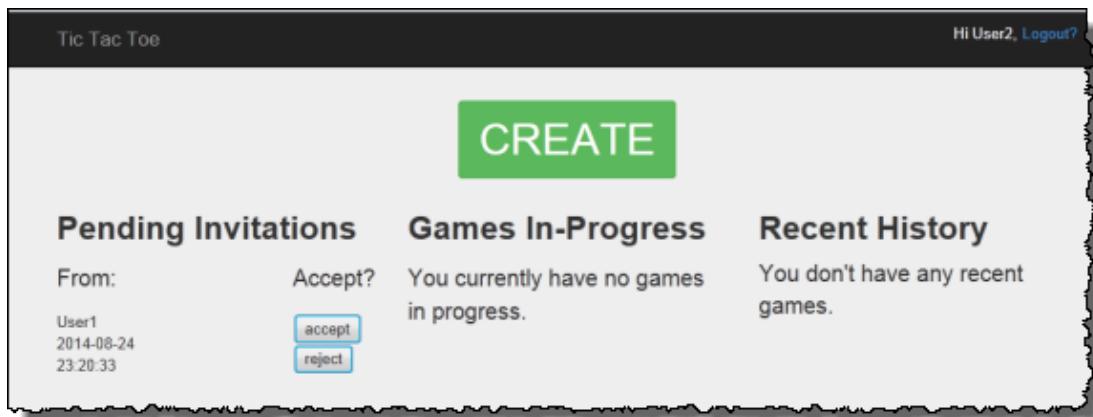


- Desse modo, o jogo é criado adicionando um item na tabela Games. Isso define o status do jogo como PENDING.
8. Abra outra janela do navegador e digite as informações a seguir.

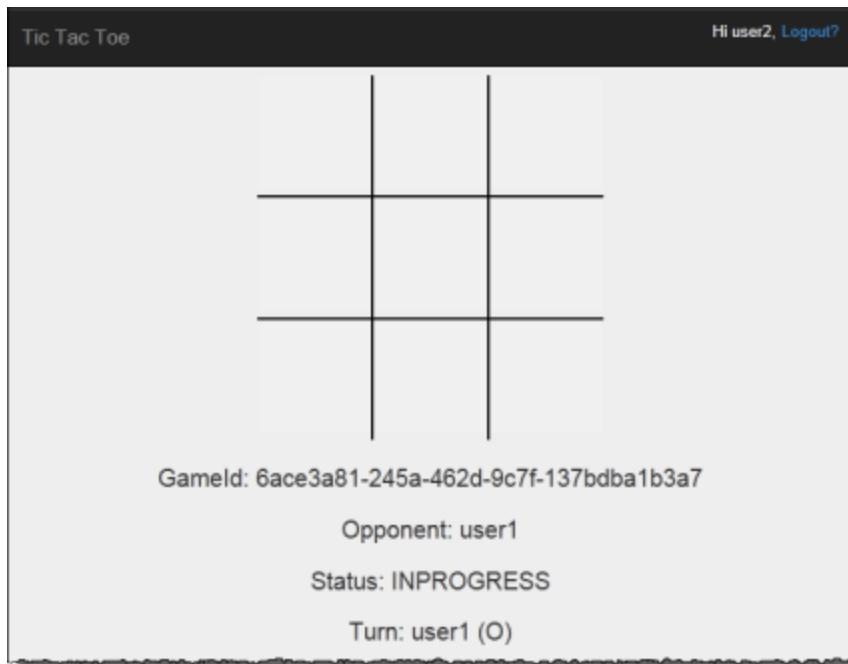
```
http://localhost:5000/
```

- O navegador passa informações por meio de cookies. Portanto, você deve usar o modo incognito ou navegação privada para que seus cookies não sejam estendidos.
9. Faça login como user2.

É exibida uma página que mostra um convite pendente do user1.



10. Selecione accept (aceitar) para aceitar o convite.



A página do jogo é exibida com uma grade de jogo da velha vazia. A página também mostra informações relevantes do jogo, como o ID do jogo, de quem é a vez e o status do jogo.

11. Jogue o jogo.

Para cada movimento do usuário, o web service envia uma solicitação para o DynamoDB atualizar condicionalmente o item do jogo no arquivo Games Tabela. Por exemplo, as condições garantem que o movimento é válido, que o quadrado que o usuário escolheu está disponível e que era a vez do usuário que fez o movimento. Para um movimento válido, a operação de atualização adiciona um novo atributo correspondente à seleção no quadro. A operação de atualização também define o valor do atributo existente para o usuário que pode fazer o próximo movimento.

Na página do jogo, o aplicativo faz chamadas JavaScript assíncronas a cada segundo, por até cinco minutos, para verificar se o estado do jogo no DynamoDB foi alterado. Caso tenha sido, o aplicativo atualiza a página com novas informações. Depois de cinco minutos, o aplicativo para de fazer as solicitações e você precisa atualizar a página para obter informações atualizadas.

Etapa 2: Analise o modelo de dados e os detalhes da implantação

Tópicos

- [2.1: Modelo de dados de base \(p. 1100\)](#)
- [2.2: Aplicativo em ação \(Code Walkthrough\) \(p. 1102\)](#)

2.1: Modelo de dados de base

Este aplicativo de exemplo destaca os seguintes conceitos de modelo de dados do DynamoDB:

- Tabela do— No DynamoDB, uma tabela é uma coleção de itens (ou seja, registros), e cada item é uma coleção de pares de nome-valor chamados atributos.

Neste exemplo de Jogo da velha, o aplicativo armazena todos os dados do jogo em uma tabela, Games. O aplicativo cria um item na tabela por jogo e armazena todos os dados de jogos como atributos. Um jogo da velha pode ter até nove movimentações. Como as tabelas do DynamoDB não possuem um esquema em casos nos quais apenas a chave primária é o atributo obrigatório, o aplicativo pode armazenar um número variável de atributos por item de jogo.

A tabela Games possui uma chave primária simples composta de um atributo, GameId, do tipo string. O aplicativo atribui um ID exclusivo a cada jogo. Para obter mais informações sobre chaves primárias do DynamoDB, consulte [Chave primária \(p. 6\)](#).

Quando um usuário inicia um jogo da velha, convidando outro usuário para jogar, o aplicativo cria um novo item na tabela Games com atributos que armazenam metadados de jogos, como os seguintes:

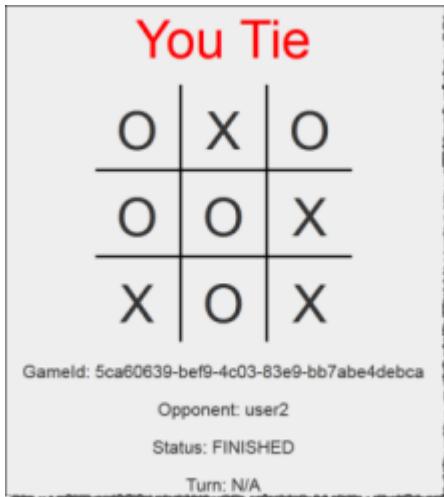
- HostId, o usuário que iniciou o jogo.
- Opponent, o usuário que foi convidado para jogar.
- O usuário que está na vez de jogar. O usuário que iniciou o jogo joga primeiro.
- O usuário que usa o símbolo O no quadro. O usuário que inicia os jogos usa o símbolo O.

Além disso, o aplicativo cria um atributo StatusDate concatenado, marcando o estado inicial do jogo como PENDING. A captura de tela a seguir mostra um item de exemplo como ele aparece no console do DynamoDB:

The screenshot shows the 'Amazon DynamoDB Explore Table' interface for the 'Games' table. The table has the following data:

Attribute	Type	Value
GameId (Hash Key)	String	"6ffd7f5-e293-4b4a-bacf-6ddde49ef0ae"
HostId	String	"user1"
O	String	"user1"
Opponent	String	"user2"
StatusDate	String	"PENDING_2014-07-06 21:28:02.354807"
Turn	String	"user1"

À medida que o jogo progride, o aplicativo adiciona um atributo à tabela para cada movimento do jogo. O nome do atributo é a posição no quadro, por exemplo `TopLeft` ou `BottomRight`. Por exemplo, um movimento pode ter um atributo `TopLeft` com o valor `0`, um atributo `TopRight` com o valor `0` e um atributo `BottomRight` com o valor `x`. O valor do atributo é `0` ou `x`, dependendo de qual usuário fez o movimento. Por exemplo, considere o quadro a seguir.



- Atributos de valor concatenados— O `statusDate` atributo ilustra um atributo de valor concatenado. Em essa abordagem, em vez de criar atributos separados para armazenar o status do jogo (`PENDING`, `IN_PROGRESS` e `FINISHED`) e a data (quando o último movimento foi feito), você pode combiná-los como um único atributo, por exemplo `IN_PROGRESS_2014-04-30 10:20:32`.

Em seguida, o aplicativo usa o atributo `statusDate` na criação de índices secundários, especificando `StatusDate` como uma chave de classificação para o índice. A vantagem de usar o atributo de valor concatenado `StatusDate` é melhor demonstrada nos índices discutidos a seguir.

- Índices secundários globais— Você pode usar a chave primária da tabela, `GameId` para consultar a tabela com eficiência para encontrar um item do jogo. Para consultar outros atributos que não sejam atributos da chave primária na tabela, o DynamoDB oferece suporte à criação de índices secundários. Neste aplicativo de exemplo, você cria os seguintes dois índices secundários:

Local Secondary Indexes															
Index Name	Hash Key	Range Key	Projected Attributes	Index Size (Bytes)*	Item Count*										
This table has no local secondary indexes.															
Global Secondary Indexes															
Index Name	Hash Key	Range Key	Projected Attributes	Status	Read Capacity Units	Write Capacity Units	Last Decrease Time	Last Increase Time	Index Size (Bytes)*	Item Count*					
hostStatusDate	HostId (String)	StatusDate (String)	All	Active	20	20		Sat May 31 10:35:42 GMT-700 2014	20305	125					
oppStatusDate	Opponent (String)	StatusDate (String)	All	Active	20	20		Sat May 31 10:35:42 GMT-700 2014	20305	125					

- `HostId-StatusDate-index`. O índice tem `HostId` como chave de partição e `StatusDate` como chave de classificação. Você pode usar esse índice para consultar o `HostId`, por exemplo, para localizar jogos hospedados por um determinado usuário.
- `OpponentId-StatusDate-index`. O índice tem `OpponentId` como chave de partição e `StatusDate` como chave de classificação. Você pode usar esse índice para consultar o `Opponent`, por exemplo, para localizar jogos nos quais um determinado usuário é o oponente.

Esses índices são chamados de índices secundários globais porque a chave de partição nesses índices não é igual à chave de partição (`GameId`), usada na chave primária da tabela.

Observe que ambos os índices especificam `StatusDate` como chave de classificação. Fazer isso permite o seguinte:

- Você pode consultar usando o operador de comparação `BEGINS_WITH`. Por exemplo, você pode encontrar todos os jogos com o atributo `IN_PROGRESS` hospedados por um determinado usuário. Neste caso, o operador `BEGINS_WITH` verifica o valor `StatusDate` que começa com `IN_PROGRESS`.
- O DynamoDB armazena os itens no índice em ordem classificada, por valor de chave de classificação. Portanto, se todos os prefixos de status forem os mesmos (por exemplo, `IN_PROGRESS`), o formato ISO usado para a parte de data terá itens classificados do mais antigo para o mais recente. Essa abordagem permite que determinadas consultas sejam executadas de forma eficiente, por exemplo, a seguinte:
 - Recupere até 10 dos jogos `IN_PROGRESS` mais recentes hospedados pelo usuário que está conectado. Para essa consulta, você especifica o índice `HostId-StatusDate-index`.
 - Recupere até 10 dos jogos `IN_PROGRESS` mais recentes nos quais o usuário conectado é o oponente. Para essa consulta, você especifica o índice `OpponentId-StatusDate-index`.

Para obter mais informações sobre índices secundários, consulte [Como melhorar o acesso a dados com índices secundários \(p. 559\)](#).

2.2: Aplicativo em ação (Code Walkthrough)

Este aplicativo tem duas páginas principais:

- Página inicial – esta página oferece ao usuário um login simples, um botão CRIAR para criar um novo jogo da velha, uma lista de jogos em andamento, o histórico de jogos e todos os convites de jogo pendentes ativos.

A página inicial não é atualizada automaticamente; você deve atualizar a página para atualizar as listas.

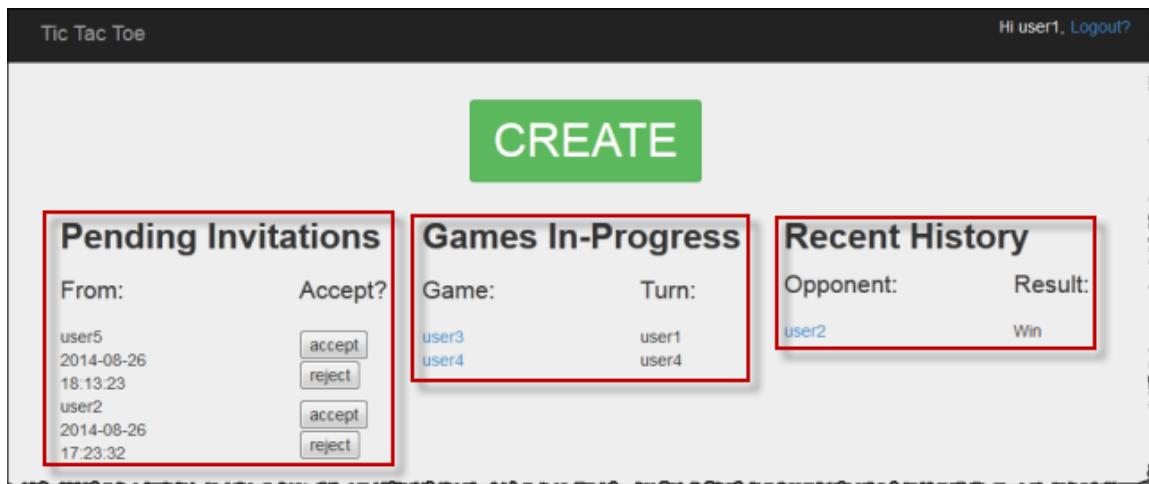
- Página do jogo – Esta página mostra a grade do jogo da velha na qual os usuários jogam.

O aplicativo atualiza a página de jogos automaticamente a cada segundo. O JavaScript no navegador chama o servidor web Python a cada segundo para consultar na tabela Games se os itens do jogo na tabela foram alterados. Se for o caso, o JavaScript aciona uma atualização da página para que o usuário veja o quadro atualizado.

Vamos ver em detalhes como o aplicativo funciona.

Página inicial

Depois que o usuário fizer login, o aplicativo exibe as três listas de informações a seguir.



- **invitations**— Esta lista mostra até 10 convites mais recentes de outros usuários que estão aguardando aceitação pelo usuário que está conectado. Na captura de tela anterior, o user1 tem convites de user5 e de user2 pendentes.
- **Jogos em andamento**— Esta lista mostra até 10 jogos mais recentes que estão em andamento. Esses são os jogos que o usuário está ativamente jogando, que têm o status IN_PROGRESS. Na captura de tela, o user1 está jogando ativamente um jogo da velha com user3 e user4.
- **Histórico recente**— Esta lista mostra até 10 jogos mais recentes que o usuário concluiu, que possuem o status FINISHED. No jogo apresentado na captura de tela, o user1 jogou anteriormente com o user2. Para cada jogo concluído, a lista mostra o resultado do jogo.

No código, a função `index` (em `application.py`) faz as seguintes três chamadas para recuperar informações de status do jogo:

```
inviteGames      = controller.getGameInvites(session["username"])
inProgressGames = controller.getGamesWithStatus(session["username"], "IN_PROGRESS")
finishedGames    = controller.getGamesWithStatus(session["username"], "FINISHED")
```

Cada uma dessas chamadas retorna uma lista de itens do DynamoDB que são empacotados pelo método `Game` objetos. É fácil extrair dados desses objetos na exibição. A função de índice passa essas listas de objetos para a exibição para renderizar o HTML.

```
return render_template("index.html",
                      user=session["username"],
                      invites=inviteGames,
                      inprogress=inProgressGames,
                      finished=finishedGames)
```

O aplicativo Jogo da velha define o `Game` para armazenar os dados do jogo recuperados do DynamoDB. Essas funções retornam listas de `Game` os objetos que permitem que você isole o resto do aplicativo do código relacionado a itens do Amazon DynamoDB. Portanto, essas funções ajudam a separar o código do seu aplicativo dos detalhes da camada de armazenamento de dados.

O padrão de arquitetura descrito aqui também é chamado de padrão de interface do usuário MVC (controlador de visualização de modelo). Neste caso, as instâncias do objeto `Game` (representando os dados) são o modelo, e a página HTML é a exibição. O controlador é dividido em dois arquivos. O arquivo `application.py` tem o controlador para o framework Flask, e a lógica de negócios é isolada no arquivo `gameController.py`. Ou seja, o aplicativo armazena tudo o que tem a ver com o DynamoDB SDK em seu próprio arquivo separado no diretório `dynamodb` folder.

Vamos analisar as três funções e como elas consultam a tabela Games usando índices secundários globais para recuperar dados relevantes.

Uso de getGameInvites para obter a lista de convites de jogo pendentes

A função `getGameInvites` recupera a lista dos 10 convites pendentes mais recentes. Esses jogos foram criados pelos usuários, mas os oponentes não aceitaram os convites de jogo. Para esses jogos, o status permanece `PENDING` até que o oponente aceite o convite. Se o oponente recusar o convite, o aplicativo removerá o item correspondente da tabela.

A função especifica a consulta da seguinte forma:

- Ela especifica o índice `OpponentId-StatusDate-index` para ser usado com os seguintes valores de chave de índice e operadores de comparação:
 - A chave de partição é `OpponentId` e usa a chave de índice `user ID`.
 - A chave de classificação é `StatusDate` e usa o operador de comparação e o valor de chave de índice `beginswith="PENDING_"`.

Você pode usar o índice `OpponentId-StatusDate-index` para recuperar jogos para os quais o usuário conectado é convidado – ou seja, nos quais o usuário conectado é o oponente.

- A consulta limita o resultado a 10 itens.

```
gameInvitesIndex = self.cm.getGamesTable().query(  
    Opponent__eq=user,  
    StatusDate__beginswith="PENDING_",  
    index="OpponentId-StatusDate-index",  
    limit=10)
```

No índice, para cada `OpponentId` (a chave de partição) O DynamoDB mantém os itens classificados por `StatusDate` (a chave de classificação). Portanto, os jogos que a consulta retorna serão os 10 jogos mais recentes.

Uso de getGamesWithStatus para obter a lista de jogos com um status específico

Depois que um oponente aceita um convite de jogo, o status do jogo muda para `IN_PROGRESS`. Depois que o jogo for concluído, o status mudará para `FINISHED`.

As consultas para encontrar jogos que estão em andamento ou concluídos são as mesmas, exceto para o valor de status diferente. Portanto, o aplicativo define a função `getGamesWithStatus`, que usa o valor de status como um parâmetro.

```
inProgressGames = controller.getGamesWithStatus(session["username"], "IN_PROGRESS")  
finishedGames = controller.getGamesWithStatus(session["username"], "FINISHED")
```

A seção a seguir aborda os jogos em andamento, mas a mesma descrição também se aplica a jogos concluídos.

Uma lista de jogos em andamento para um determinado usuário inclui o seguinte:

- Jogos em andamento hospedados pelo usuário
- Jogos em andamento nos quais o usuário é o oponente

A função `getGamesWithStatus` executa as duas consultas seguintes, cada vez usando o índice secundário apropriado.

- A função consulta a tabela Games usando o índice HostId-StatusDate-index. Para o índice, a consulta especifica valores de chave primária – tanto os valores de chave de partição (HostId) quanto os de chave de classificação (StatusDate), juntamente com operadores de comparação.

```
hostGamesInProgress = self.cm.getGamesTable().query(HostId__eq=user,
                                                    StatusDate__beginswith=status,
                                                    index="HostId-StatusDate-index",
                                                    limit=10)
```

Observe a sintaxe do Python para operadores de comparação:

- HostId__eq=user especifica o operador de comparação de igualdade.
- StatusDate__beginswith=status especifica o operador de comparação BEGINS_WITH.
- A função consulta a tabela Games usando o índice OpponentId-StatusDate-index.

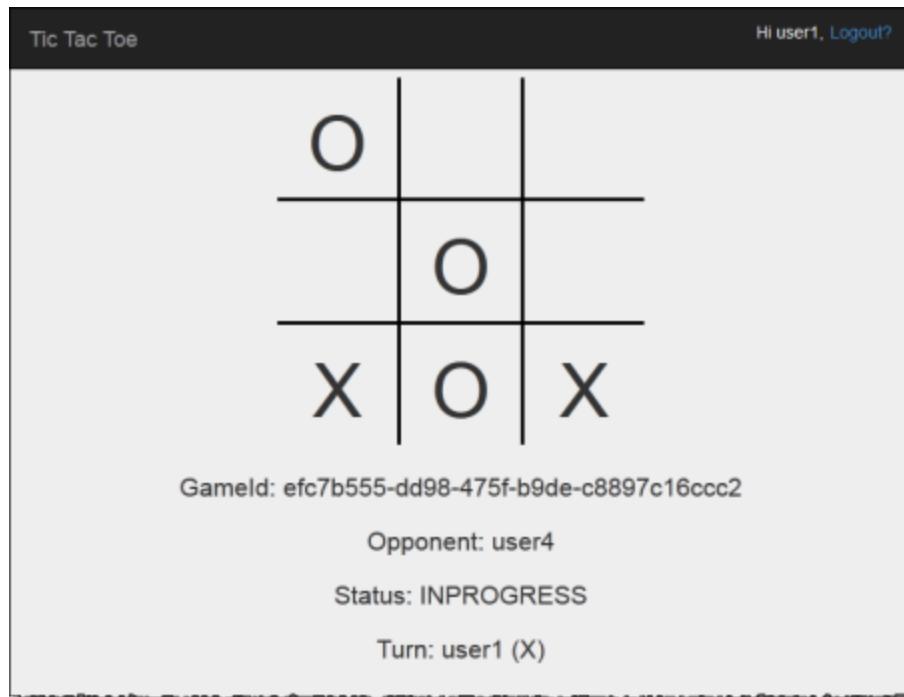
```
oppGamesInProgress = self.cm.getGamesTable().query(Opponent__eq=user,
                                                    StatusDate__beginswith=status,
                                                    index="OpponentId-StatusDate-index",
                                                    limit=10)
```

- Em seguida, a função combina as duas listas, classifica, e para os primeiros itens de 0 a 10, cria uma lista dos objetos Game e retorna a lista para a função de chamada (ou seja, o índice).

```
games = self.mergeQueries(hostGamesInProgress,
                           oppGamesInProgress)
return games
```

Página de jogos

A página de jogos é onde o usuário joga os jogos da velha. Ela mostra a grade do jogo junto com as informações relevantes do jogo. A captura de tela a seguir mostra um jogo de exemplo em andamento:



O aplicativo exibe a página de jogos nas seguintes situações:

- O usuário cria um jogo convidando outro usuário para jogar.

Neste caso, a página mostra o usuário como host e o status do jogo como PENDING, aguardando o oponente aceitar.

- O usuário aceita um dos convites pendentes na página inicial.

Neste caso, a página mostra o usuário como o oponente e o status do jogo como IN_PROGRESS.

Uma seleção do usuário no quadro gera uma solicitação POST de formato para o aplicativo. Ou seja, o Flask chama a função `selectSquare` (em `application.py`) com os dados no formato HTML. Essa função, por sua vez, chama a função `updateBoardAndTurn` (em `gameController.py`) para atualizar o item de jogo da seguinte forma:

- Ela adiciona um novo atributo específico ao movimento.
- Ela atualiza o valor do atributo `Turn` para o usuário cuja vez é a próxima.

```
controller.updateBoardAndTurn(item, value, session["username"])
```

A função retorna verdadeiro se a atualização do item foi bem-sucedida; caso contrário, retorna falso. Observe o seguinte sobre a função `updateBoardAndTurn`:

- A função chama `ouupdate_item` função do SDK para Python para fazer um conjunto finito de atualizações em um item existente. A função é mapeada para `updateItem` operação no DynamoDB. Para obter mais informações, consulte [UpdateItem](#).

Note

A diferença entre as operações `UpdateItem` e `PutItem` é que `PutItem` substitui o item inteiro. Para obter mais informações, consulte [PutItem](#).

Para a chamada `update_item`, o código identifica o seguinte:

- A chave primária da tabela Games (ou seja, `ItemId`).

```
key = { "GameId" : { "S" : gameId } }
```

- O novo atributo a ser adicionado, específico para o movimento do usuário atual, e seu valor (por exemplo, `TopLeft="X"`).

```
attributeUpdates = {
    position : {
        "Action" : "PUT",
        "Value" : { "S" : representation }
    }
}
```

- Condições que devem ser verdadeiras para a atualização acontecer:

- O jogo deve estar em andamento. Ou seja, o valor do atributo `StatusDate` deve começar com `IN_PROGRESS`.
- A vez atual deve ser de um usuário válido, conforme especificado pelo atributo `Turn`.
- O quadrado que o usuário escolheu deve estar disponível. Ou seja, o atributo correspondente ao quadrado não deve existir.

```
expectations = {"StatusDate" : {"AttributeValueList": [{"S" : "IN_PROGRESS_"}]},  
    "ComparisonOperator": "BEGINS_WITH"},  
    "Turn" : {"Value" : {"S" : current_player}},  
    position : {"Exists" : False}}
```

Agora, a função chama `update_item` para atualizar o item.

```
self.cm.db.update_item("Games", key=key,  
attribute_updates=attributeUpdates,  
expected=expectations)
```

Depois que a função retorna, as chamadas da função `selectSquare` são redirecionadas conforme mostrado no exemplo a seguir.

```
redirect("/game="+gameId)
```

Essa chamada faz com que o navegador seja atualizado. Como parte dessa atualização, o aplicativo verifica se o jogo terminou em uma vitória ou empate. Em caso afirmativo, o aplicativo atualizará o item de jogo adequadamente.

Etapa 3: Implantar em produção usando o serviço DynamoDB

Tópicos

- [3.1: Criar uma função do IAM para Amazon EC2 \(p. 1108\)](#)
- [3.2: Criar a tabela Games no Amazon DynamoDB \(p. 1109\)](#)
- [3.3: Empacotar e implantar o código do aplicativo Jogo da velha \(p. 1109\)](#)
- [3.4: Configurar o ambiente do AWS Elastic Beanstalk \(p. 1110\)](#)

Em seções anteriores, você implantou e testou o aplicativo Jogo da velha localmente no seu computador usando o DynamoDB Local. Agora, você implanta o aplicativo em produção da seguinte forma:

- **Implante o aplicativo usando o AWS Elastic Beanstalk, um serviço fácil de usar para implantação e escalabilidade de aplicativos web e web services. Para obter mais informações, consulte [Implantar um aplicativo Flask noAWS Elastic Beanstalk](#).**

O Elastic Beanstalk executa uma ou mais instâncias do Amazon Elastic Compute Cloud (Amazon EC2), que você configura por meio do Elastic Beanstalk, no qual seu aplicativo Jogo da velha será executado.

- Usando o serviço Amazon DynamoDB, crie um `Game` tabela que existe em AWS em vez de localmente no computador.

Além disso, você também tem que configurar as permissões. Quaisquer AWS recursos que você criar, como o `Game` no DynamoDB, são privadas por padrão. Somente o proprietário do recurso, que é a conta da AWS que criou a tabela `Game`, pode acessar essa tabela. Assim, por padrão, seu aplicativo Jogo da velha não pode atualizar a tabela `Game`.

Para conceder permissões necessárias, crie um AWS Identity and Access Management(IAM) e conceda a essa função permissões para acessar o arquivo `GameTable`. Sua instância do Amazon EC2 primeiro assume essa função. Em resposta, AWS retorna credenciais de segurança temporárias que a instância do Amazon EC2 pode usar para atualizar o `Game` nome do pedido Jogo da velha. Ao configurar seu aplicativo Elastic Beanstalk, você especifica a função do IAM que a instância ou as instâncias do Amazon

EC2 podem assumir. Para obter mais informações sobre funções do IAM, consulte[Funções do IAM para o Amazon EC2](#)noGuia do usuário do Amazon EC2 para instâncias do Linux.

Note

Antes de criar instâncias do Amazon EC2 para o aplicativo Jogo da velha, você deve primeiro decidir oAWSA região onde você deseja que o Elastic Beanstalk crie as instâncias. Depois de criar o aplicativo Elastic Beanstalk, você fornece o mesmo nome de região e endpoint em um arquivo de configuração. O aplicativo Jogo da velha usa informações desse arquivo para criar o arquivo oGamese enviar solicitações subsequentes em umAWSRegião : Tanto o DynamoDBGamesA tabela do e as instâncias do Amazon EC2 executadas pelo Elastic Beanstalk devem estar na mesma região. Para obter uma lista das regiões disponíveis, consulte[Amazon DynamoDB](#)noReferência geral do Amazon Web Services.

Resumindo, você faz o seguinte para implantar o aplicativo Jogo da velha em produção:

1. Crie uma função do IAM usando o serviço do IAM. Você poderá anexar uma política a essa função, concedendo permissões para as ações do DynamoDB acessarem oGamesTabela.
2. Empacote o código do aplicativo Jogo da velha e um arquivo de configuração, e crie um arquivo .zip. Você usa este .zipPara fornecer o código do aplicativo Jogo da velha ao Elastic Beanstalk a fim de inseri-los nos servidores. Para obter mais informações sobre como criar um pacote, consulte[Criar um pacote de origem do aplicativo](#)noAWS Elastic BeanstalkGuia do desenvolvedor.
3. Configure o ambiente do Elastic Beanstalk. O Elastic Beanstalk inicia uma instância ou instâncias do Amazon EC2 e implanta seu pacote de aplicativos Jogo da velha nelas. Depois que o ambiente do Elastic Beanstalk estiver pronto, você fornecerá o nome do arquivo de configuração, adicionando a variável de ambiente CONFIG_FILE.
4. Crie a tabela do DynamoDB. Usando o serviço Amazon DynamoDB, você cria oGamesTabela INTOAWS, em vez de localmente em seu computador. Lembre-se: essa tabela tem uma chave primária simples que consiste na chave de partição GameId do tipo string.
5. Teste o jogo em produção.

3.1: Criar uma função do IAM para Amazon EC2

Criar uma função do IAM doAmazon EC2O tipo permite que a instância do Amazon EC2 que está executando seu aplicativo Jogo da velha assuma a função correta e faça solicitações do aplicativo para acessar oGamesTabela. Ao criar a função, selecione a opção Custom Policy (Política personalizada), copie e cole a política a seguir.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "dynamodb>ListTables"  
            ],  
            "Effect": "Allow",  
            "Resource": "*"  
        },  
        {  
            "Action": [  
                "dynamodb:*"  
            ],  
            "Effect": "Allow",  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:922852403271:table/Games",  
                "arn:aws:dynamodb:us-west-2:922852403271:table/GameScore"  
            ]  
        }  
    ]  
}
```

```
        "arn:aws:dynamodb:us-west-2:922852403271:table/Games/index/*"
    }
}
```

Para obter mais instruções, consulte [Criar uma função para um AWS Service \(AWS Management Console\)](#) no Guia do usuário do IAM.

3.2: Criar a tabela Games no Amazon DynamoDB

O Games no DynamoDB armazena os dados do jogo. Se a tabela não existir, o aplicativo criará a tabela para você. Nesse caso, deixe o aplicativo criar a tabela Games.

3.3: Empacotar e implantar o código do aplicativo Jogo da velha

Se você seguiu as etapas deste exemplo, então, já fez download do aplicativo Jogo da velha. Caso contrário, faça download e extraia todos os arquivos para uma pasta no seu computador local. Para obter instruções, consulte [Etapa 1: Implantar e testar localmente \(p. 1096\)](#).

Após extrair todos os arquivos, você terá uma pasta code. Para entregar essa pasta para o Elastic Beanstalk, você agrupa o conteúdo dessa pasta como um .zip file. Primeiramente, você precisa adicionar um arquivo de configuração a essa pasta. Seu aplicativo usa as informações da região e do endpoint para criar uma tabela do DynamoDB na região especificada e fará solicitações de operação de tabela subsequentes usando o endpoint especificado.

1. Alterne para a pasta na qual você fez download do aplicativo Jogo da velha.
2. Na pasta raiz do aplicativo, crie um arquivo de texto chamado `beanstalk.config` com o conteúdo a seguir.

```
[dynamodb]
region=<AWS region>
endpoint=<DynamoDB endpoint>
```

Por exemplo, você pode usar o conteúdo a seguir.

```
[dynamodb]
region=us-west-2
endpoint=dynamodb.us-west-2.amazonaws.com
```

Para obter uma lista de regiões disponíveis, acesse [Amazon DynamoDB](#) na Referência geral da Amazon Web Services.

Important

A região especificada no arquivo de configuração é o local onde o aplicativo Jogo da velha cria o atributo Games no DynamoDB. Você deve criar o aplicativo Elastic Beanstalk discutido na próxima seção na mesma região.

Note

Ao criar seu aplicativo Elastic Beanstalk, você pode pedir para iniciar um ambiente no qual pode escolher o tipo. Para testar o aplicativo de exemplo Jogo da velha, você pode escolher o tipo de ambiente Single Instance (Instância única), ignorar o seguinte, e ir para a próxima etapa.

No entanto, o tipo de ambiente Load balancing, autoscaling (Balanceamento de carga, escalabilidade automática) oferece um ambiente de alta disponibilidade e escalável, algo que você deve considerar ao criar e implantar outros aplicativos. Se você escolher esse tipo de

ambiente, também será necessário gerar um UUID e adicioná-lo ao arquivo de configuração, como mostrado a seguir.

```
[dynamodb]
region=us-west-2
endpoint=dynamodb.us-west-2.amazonaws.com
[flask]
secret_key= 284e784d-1a25-4a19-92bf-8eeb7a9example
```

No comunicação entre cliente e servidor, quando o servidor envia a resposta, por segurança, o servidor envia um cookie assinado que o cliente envia de volta para o servidor na próxima solicitação. Quando há apenas um servidor, o servidor pode gerar localmente uma chave de criptografia quando ele for iniciado. Quando há vários servidores, todos eles precisam saber a mesma chave de criptografia; caso contrário, eles não poderão ler os cookies definidos pelos servidores do mesmo nível. Ao adicionar `secret_key` ao arquivo de configuração, todos os servidores são informados para usar essa chave de criptografia.

3. Compacte o conteúdo da pasta raiz do aplicativo (que inclui o arquivo `beanstalk.config`), por exemplo, `TicTacToe.zip`.
4. Faça upload do `.zip` para um bucket do Amazon Simple Storage Service (Amazon S3). Na próxima seção, você fornece este arquivo `.zip` para o Elastic Beanstalk para fazer upload no servidor ou servidores.

Para obter instruções sobre como fazer upload de um bucket do Amazon S3, consulte [Crie um bucket](#) e [Adicionar um objeto a um bucket](#) no Guia de conceitos básicos do Amazon Simple Storage Service.

3.4: Configurar o ambiente do AWS Elastic Beanstalk

Nesta etapa, você cria um aplicativo Elastic Beanstalk, que é um conjunto de componentes, incluindo ambientes. Para este exemplo, você inicia uma instância do Amazon EC2 para implantar e executar seu aplicativo Jogo da velha.

1. Insira o seguinte URL personalizado para configurar um console do Elastic Beanstalk para configurar o ambiente.

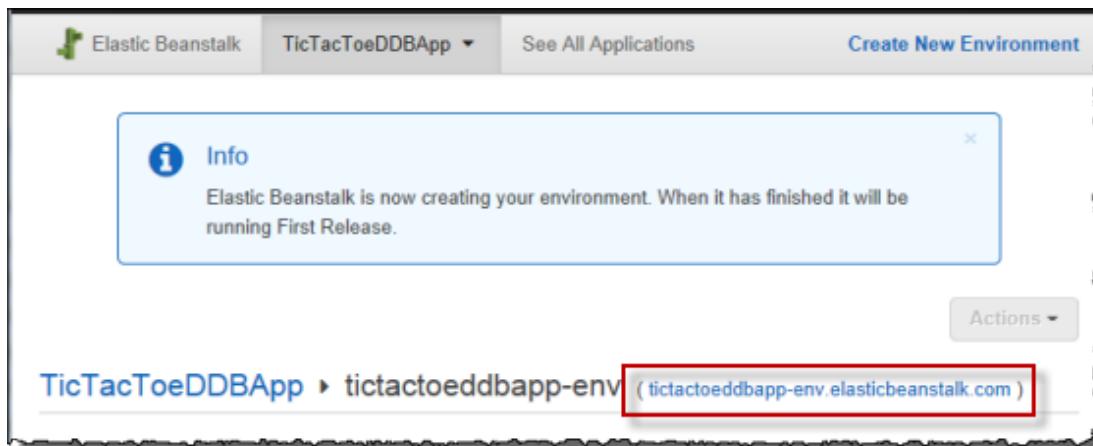
```
https://console.aws.amazon.com/elasticbeanstalk/?region=<AWS-Region>#/newApplication
?applicationName=TicTacToe<your-name>
&solutionStackName=Python
&sourceBundleUrl=https://s3.amazonaws.com/<bucket-name>/TicTacToe.zip
&environmentType=SingleInstance
&instanceType=t1.micro
```

Para obter mais informações sobre URLs personalizados, consulte [Construção de um URL "Iniciar agora"](#) no AWS Elastic Beanstalk Guia do desenvolvedor do . Para obter o URL, observe o seguinte:

- É necessário fornecer um AWS Nome da região (o mesmo que o fornecido no arquivo de configuração), um nome de bucket do Amazon S3 e o nome do objeto.
- Para testes, o URL solicita o tipo de ambiente SingleInstance e `t1.micro` como o tipo de instância.
- O nome do aplicativo precisa ser exclusivo. Assim, no URL anterior, sugerimos que você insira seu nome ao `applicationName`.

Essa ação abre o console do Elastic Beanstalk. Em alguns casos, talvez seja necessário fazer login.

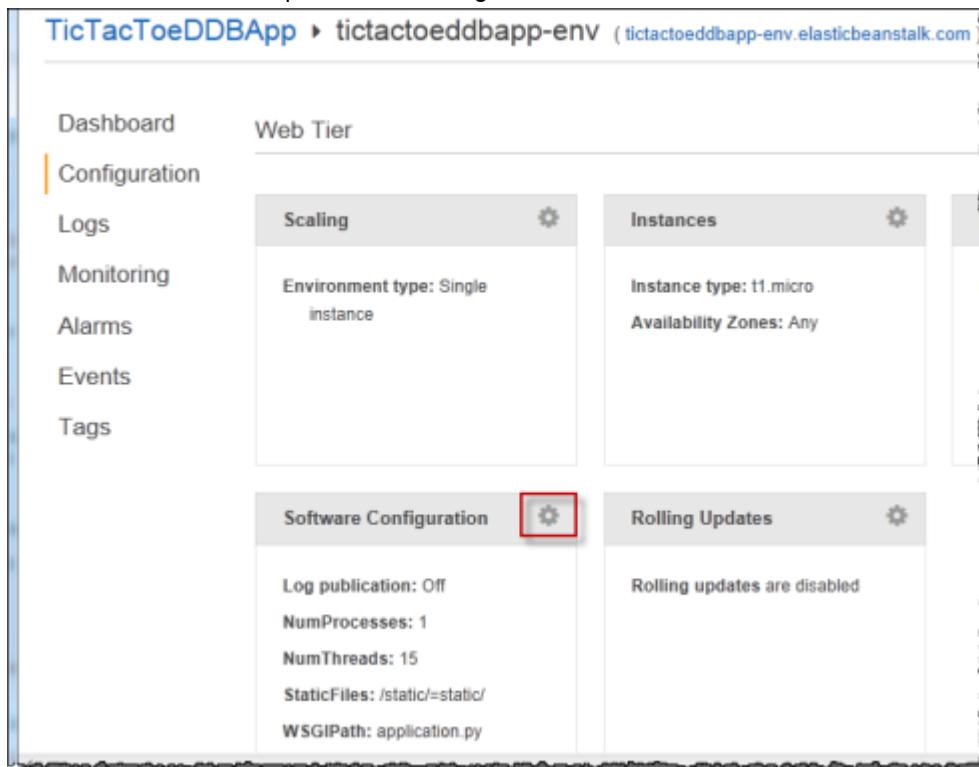
2. No console do Elastic Beanstalk, escolha Revisar e iniciar e, em seguida, selecione Iniciar.
3. Anote o URL para referência futura. Este URL abre a página inicial do seu aplicativo Jogo da velha.



4. Configure o aplicativo Jogo da velha para que ele saiba a localização do arquivo de configuração.

Após o Elastic Beanstalk criar o aplicativo, escolha Configuração.

- a. Selecione o ícone de engrenagem ao lado de Software Configuration (Configuração de software), conforme mostrado na captura de tela a seguir.



- b. No final da seção Environment Properties (Propriedades do ambiente), digite **CONFIG_FILE** e seu valor **beanstalk.config** e depois selecione Save (Salvar).

Pode levar alguns minutos para a atualização deste ambiente ser concluída.

PARAM3 A predefined environment property that will be available to your running application.	<input type="text"/>
PARAM4 A predefined environment property that will be available to your running application.	<input type="text"/>
PARAM5 A predefined environment property that will be available to your running application.	<input type="text"/>
CONFIG_FILE	<input type="text" value="beanstalk.config"/> +

Cancel Save

Depois que a atualização for concluída, você poderá jogar o jogo.

5. No navegador, digite o URL que você copiou na etapa anterior, como mostrado no exemplo a seguir.

`http://<app-name>.elasticbeanstalk.com`

Essa ação abre a página inicial do aplicativo.

The screenshot shows the main interface of the Tic Tac Toe application. At the top, there is a navigation bar with the title "Tic Tac Toe", a user dropdown set to "i.e. John", and a "Log In" button. Below the navigation bar, there is a large green "CREATE" button. Underneath the button, there are three sections: "Pending Invitations" (You currently have no invites.), "Games In-Progress" (You currently have no games in progress.), and "Recent History" (You don't have any recent games.).

6. Faça login como testuser1 e selecione CREATE (Criar) para iniciar um novo jogo da velha.
7. Digite **testuser2** na caixa Choose an Opponent (Escolher um oponente).

The screenshot shows the "Choose an Opponent" page. At the top, it says "Hi testuser1, Logout?". Below that, there is a text input field containing "testuser2". At the bottom, there is a "Create Game!" button.

8. Abra outra janela do navegador.

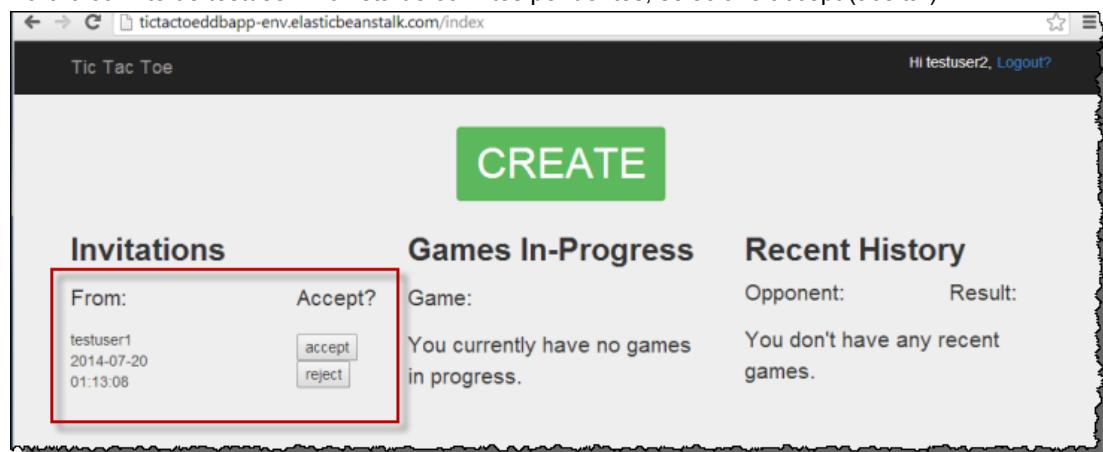
Certifique-se de limpar todos os cookies em sua janela do navegador, de modo que você não se conecte como o mesmo usuário.

9. Digite o mesmo URL para abrir a página inicial do aplicativo, como mostrado no exemplo a seguir:

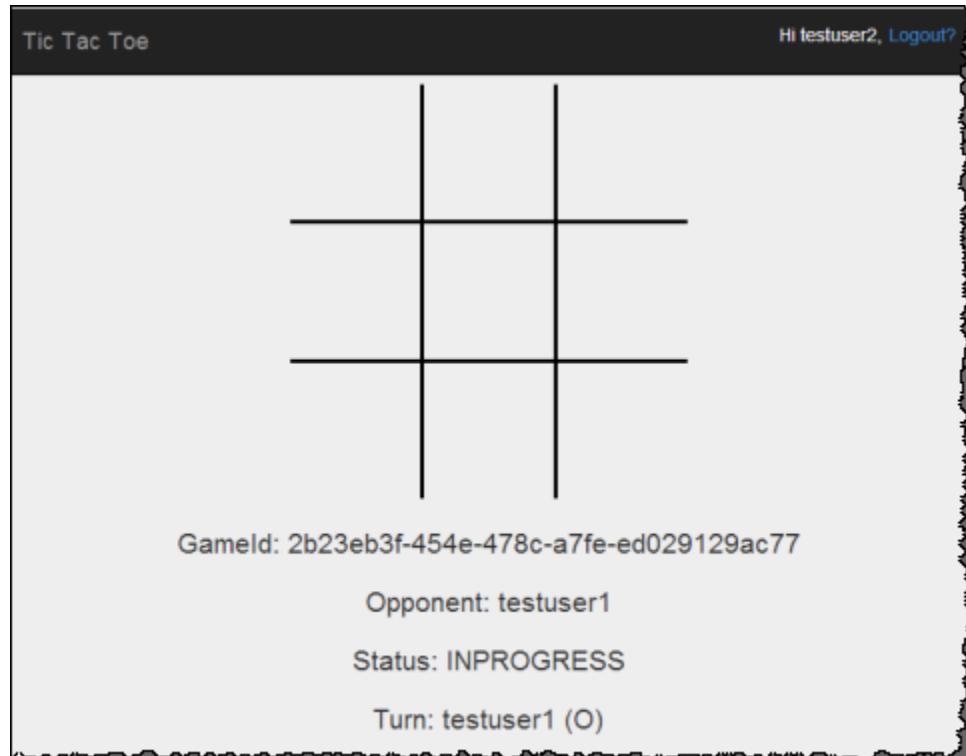
```
http://<env-name>.elasticbeanstalk.com
```

10. Faça login como testuser2.

11. Para o convite de testuser1 na lista de convites pendentes, selecione accept (aceitar).



12. Agora a página do jogo aparece.



testuser1 e testuser2 pode jogar o jogo. Para cada movimento, o aplicativo salva a movimentação no item correspondente na tabela Games.

Etapa 4: Liberação de recursos

Agora você concluiu a implantação e o teste do aplicativo Jogo da velha. O aplicativo aborda o desenvolvimento de aplicativos web de ponta a ponta no Amazon DynamoDB, com exceção da autenticação de usuários. O aplicativo usa as informações de login na página inicial apenas para adicionar o nome do jogador ao criar um jogo. Em um aplicativo de produção, você adicionaria o código necessário para realizar o login e a autenticação do usuário.

Se você concluiu o teste, poderá remover os recursos que criou para testar o aplicativo Jogo da velha para evitar ser cobrado.

Como remover recursos criados

1. Remover o GamesA tabela do que você criou no DynamoDB.
2. Encerre o ambiente do Elastic Beanstalk para liberar as instâncias do Amazon EC2.
3. Exclua a função do IAM que você criou.
4. Remova o objeto criado no Amazon S3.

Exportar e importar dados do DynamoDB usando o AWS Data Pipeline

Você pode usar o AWS Data PipelinePara exportar dados de uma tabela do DynamoDB para um arquivo em um bucket do Amazon S3. Também pode usar o console para importar dados do Amazon S3 para uma tabela do DynamoDB, no mesmoAWSregião ou em uma região diferente.

Note

O Console do DynamoDB agora oferece suporte ao seu próprio fluxo de Exportação para o Amazon S3. No entanto, ele não é compatível com o AWS Data Pipelinefluxo de importação. Para obter mais informações, consulte[Exportando dados de tabela do DynamoDB para o Amazon S3 \(p. 1046\)](#)e a publicação no blog[Exportar dados de tabela do Amazon DynamoDB para seu data lake no Amazon S3, sem necessidade de gravação de código](#).

A capacidade de exportar e importar dados é útil em muitos cenários. Por exemplo, suponha que você queira manter um conjunto de dados de linha de base para fins de teste. É possível colocar os dados de linha de base em uma tabela do DynamoDB e exportá-los para o Amazon S3. Dessa forma, depois de executar um aplicativo que modificasse os dados de teste, você poderia “redefinir” o conjunto de dados, importando a linha de base do Amazon S3 de volta para a tabela do DynamoDB. Outro exemplo envolve a exclusão acidental de dados, ou até mesmo uma operação DeleteTable acidental. Nesses casos, seria possível restaurar os dados de um arquivo de exportação anterior no Amazon S3. Você pode até mesmo copiar dados de uma tabela do DynamoDB em umAWSArmazene os dados no Amazon S3 e depois importá-los do Amazon S3 para uma tabela do DynamoDB idêntica em uma segunda região. Dessa forma, os aplicativos na segunda região poderiam acessar o endpoint do DynamoDB mais próximo e trabalhar com suas próprias cópias dos dados, com latência de rede reduzida.

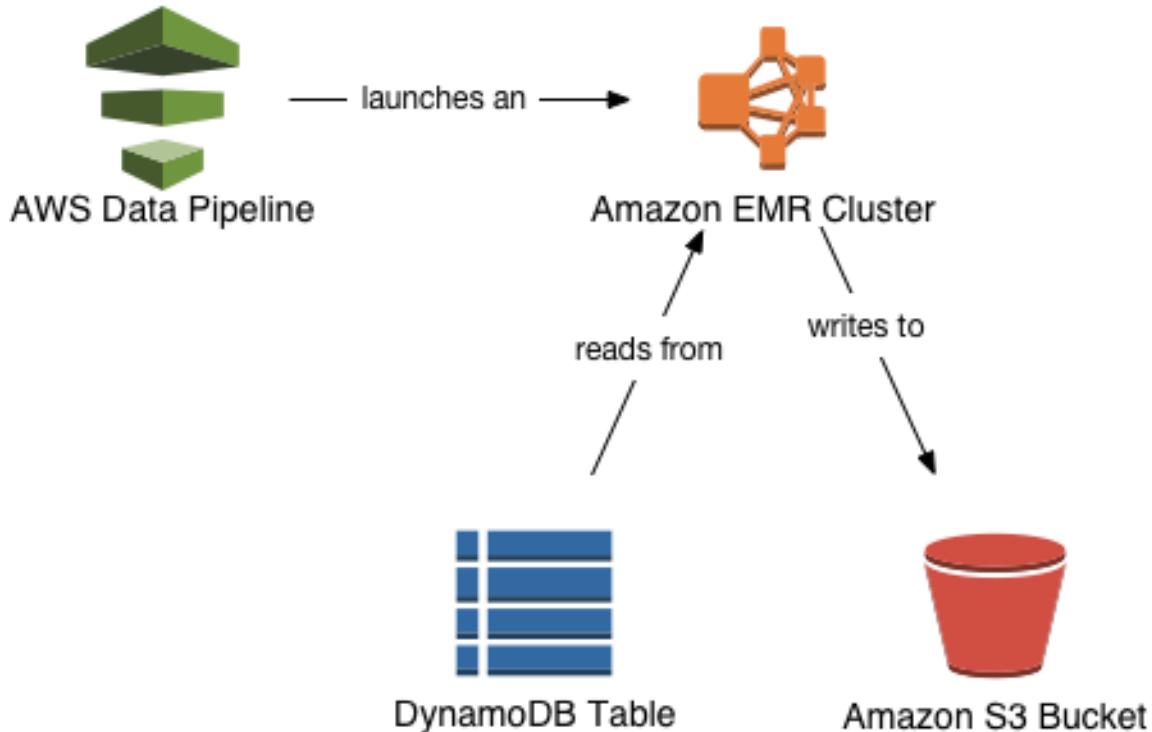
Important

O Backup e Restauração do DynamoDB é um recurso totalmente gerenciado. Você pode fazer backup de tabelas de alguns megabytes para centenas de terabytes de dados, sem

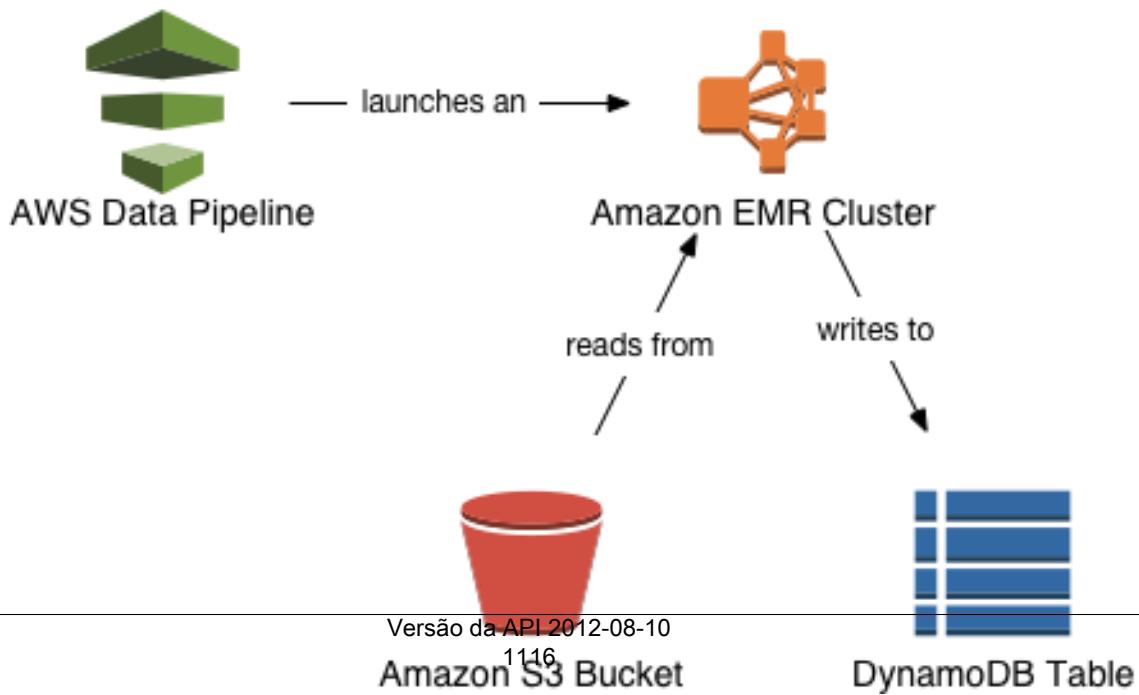
impactar o desempenho e a disponibilidade dos aplicativos de produção. Você pode restaurar sua tabela com um único clique na guia AWS Management Console ou uma única chamada de API. Recomendamos vivamente que você use o recurso nativo de backup e restauração do DynamoDB em vez de usar AWS Data Pipeline. Para mais informações, consulte [Backup e restauração sob demanda para o DynamoDB \(p. 691\)](#).

O diagrama a seguir mostra uma visão geral de como exportar e importar dados do DynamoDB usando o AWS Data Pipeline.

Exporting Data from DynamoDB to Amazon S3



Importing Data from Amazon S3 to DynamoDB



Para exportar uma tabela do DynamoDB, use o AWS Data Pipelineconsole do para criar um novo pipeline. O pipeline inicia um cluster do Amazon EMR para realizar a exportação propriamente dita. O Amazon EMR lê os dados do DynamoDB e os grava em um arquivo de exportação em um bucket do Amazon S3.

O processo é semelhante para uma importação, com a diferença de que os dados são lidos do bucket do Amazon S3 e gravados na tabela do DynamoDB.

Important

Ao exportar ou importar dados do DynamoDB, você contrair custos adicionais para oAWSserviços que são usados:

- AWS Data Pipeline — gerencia o fluxo de trabalho de importação/exportação para você.
- Amazon S3— contém os dados que você exporta ou importa do/para o DynamoDB.
- Amazon EMR— executa um cluster Hadoop gerenciado para realizar leituras e gravações entre o DynamoDB e o Amazon S3. A configuração do cluster é um `m3.xlarge` no líder da instância e um `m3.xlarge` core da instância.

Para obter mais informações, consulte [AWS Data PipelineDefinição de preços](#), [Definição de preços do Amazon EMR](#), e [Definição de preço do Amazon S3](#).

Pré-requisitos para exportar e importar dados

Ao usar o AWS Data Pipeline para exportar e importar dados, você deve especificar as ações que o pipeline tem permissão para realizar e quais recursos ele pode consumir. As ações e recursos permitidos são definidos usando oAWS Identity and Access ManagementFunções do (IAM).

Você também pode controlar o acesso criando políticas do IAM e anexando-as a usuários, funções ou grupos do IAM. Essas políticas permitem que você especifique quais usuários estão autorizados a importar e exportar seus dados do DynamoDB.

Important

O usuário do IAM que realiza as exportações e importações deve ter umativo AWSID da chave de acesso e chave secreta. Para obter mais informações, consulte [Como administrar chaves de acesso para usuários do IAM](#) no Guia do usuário do IAM.

Criação de funções do IAM paraAWS Data Pipeline

Para usarAWS Data Pipeline, as seguintes funções do IAM devem estar presentes no seuAWSConta da:

- `DataPipelineDefaultRole` — as ações que o pipeline pode realizar em seu nome.
- `DataPipelineDefaultResourceRole`— oAWSOs recursos que o pipeline provisionará em seu nome. Para exportar e importar dados do DynamoDB, esses recursos incluem um cluster do Amazon EMR e as instâncias do Amazon EC2 associadas a esse cluster.

Se você nunca usou o AWS Data Pipeline antes, precisará criar `DataPipelineDefaultRole` e `DataPipelineDefaultResourceRole` por conta própria. Depois de criar essas funções, você poderá usá-las sempre que quiser para exportar ou importar dados do DynamoDB.

Note

Se você já usou o console do AWS Data Pipeline para criar um pipeline, `DataPipelineDefaultRole` e `DataPipelineDefaultResourceRole` foram criadas naquela ocasião. Nenhuma ação adicional

será necessária, e você poderá pular essa seção e começar a criar pipelines usando o console do DynamoDB. Para mais informações, consulte e .

1. Faça login no AWS Management Console e abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. No painel do console do IAM, clique em Funções do.
3. Clique em Create Role (Criar função) e faça o seguinte:
 - a. No AWS Service Entity confiável, selecione Data Pipeline.
 - b. No painel Select your use case (Selecionar seu caso de uso), escolha Data Pipeline e, em seguida, escolha Next: Permissions (Próximo: permissões).
 - c. Observe que a política AWSDataPipelineRole é automaticamente anexada. Selecione Next:Review (Próximo: análise).
 - d. No campo Role name (Nome da função), digite DataPipelineDefaultRole para o nome da função e escolha Create role (Criar função).
4. Clique em Create Role (Criar função) e faça o seguinte:
 - a. No AWS Service Entity confiável, selecione Data Pipeline.
 - b. No painel Select your use case (Selecionar seu caso de uso), escolha EC2 Role for Data Pipeline (Função do EC2 para Data Pipeline) e, em seguida, escolha Next: Permissions (Próximo: permissões).
 - c. Observe que a política AmazonEC2RoleForDataPipelineRole é automaticamente anexada. Selecione Next:Review (Próximo: análise).
 - d. No campo Role name (Nome da função), digite DataPipelineDefaultResourceRole para o nome da função e escolha Create role (Criar função).

Agora que você criou essas funções, pode começar a criar pipelines usando o console do DynamoDB. Para mais informações, consulte e .

Como conceder a usuários e grupos do IAM permissão para realizar tarefas de exportação e importação

Para permitir que outros usuários, funções ou grupos do IAM exportem e importem seus dados de tabelas do DynamoDB, você pode criar uma política do IAM e anexá-la a usuários ou grupos designados. A política contém somente as permissões necessárias para executar essas tarefas.

Conceder acesso total

O procedimento a seguir descreve como anexar o AWS Policies gerenciadas do Amazon DynamoDB Full Access, AWS Data Pipeline Full Access uma política em linha do Amazon EMR para um usuário do IAM. Essas políticas gerenciadas fornecem acesso total ao AWS Data Pipeline para os recursos do DynamoDB, e usados com a política inline do Amazon EMR, permitem que o usuário execute as ações descritas nesta documentação.

Note

Para limitar o escopo das permissões sugeridas, a política embutida acima está impondo o uso da tag dynamodb:datapipline. Se desejar utilizar esta documentação sem essa limitação, você pode remover o Condition da política sugerida.

1. Faça login no AWS Management Console e abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. No painel do console do IAM, clique em Usuários e selecione o usuário que você deseja modificar.

3. NoPermissões, clique emAdicionar política.
4. NoAnexar permissões, clique emAssocie políticas existentes diretamente.
5. Selecione ambosAmazonDynamoDBFullAccessseAWSDataPipeline_FullAccessse clique emPróximo: Review.
6. Clique emAdicionar permissões.
7. Back onPermissões, clique emAdicionar política em linha.
8. NoCriar uma política, clique emJSONGuia.
9. Cole o conteúdo abaixo.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "EMR",  
            "Effect": "Allow",  
            "Action": [  
                "elasticmapreduce:DescribeStep",  
                "elasticmapreduce:DescribeCluster",  
                "elasticmapreduce:RunJobFlow",  
                "elasticmapreduce:TerminateJobFlows"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "Null": {  
                    "elasticmapreduce:RequestTag/dynamodbdatipeline": "false"  
                }  
            }  
        }  
    ]  
}
```

10. Clique emPolítica de revisão.
11. TipoEMRforDynamoDBDataPipelineno campo de nome.
12. Clique emCriar política.

Note

É possível usar um procedimento semelhante para anexar essa política gerenciada a uma função ou grupo em vez de a um usuário.

Como restringir o acesso a determinadas tabelas do DynamoDB

Para restringir o acesso para que um usuário só possa exportar ou importar um subconjunto das suas tabelas, você precisará criar um documento de política do IAM personalizado. Você pode usar o processo descrito emConceder acesso total (p. 1118)Como ponto de partida para sua política personalizada e depois modificá-la para que um usuário só possa trabalhar com as tabelas que você especificar.

Por exemplo, suponha que você queira permitir que um usuário do IAM exporte e importe somente oForum, Thread, eResponderTabelas. Este procedimento descreve como criar uma política personalizada, para que um usuário possa trabalhar com essas tabelas, mas não com outras.

1. Faça login no AWS Management Console e abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. No painel do console do IAM, clique emPolíticase clique emCriar política.
3. NoCriar política, vá paraCopiar umAWSPolítica gerenciadae clique emSelect.

4. No Copiar um AWS Política gerenciada, vá para [Amazon DynamoDB Full Access](#) e clique em Select.
5. No painel Review Policy (Revisar política), faça o seguinte:
 - a. Reveja os valores autogerados de Policy Name (Nome da política) e Description (Descrição). Se desejar, você pode modificar esses valores.
 - b. Na caixa de texto Policy Document (Documento da política), edite a política para restringir o acesso a tabelas específicas. Por padrão, a política permite todas as ações do DynamoDB em todas as suas tabelas:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "cloudwatch:DeleteAlarms",  
                "cloudwatch:DescribeAlarmHistory",  
                "cloudwatch:DescribeAlarms",  
                "cloudwatch:DescribeAlarmsForMetric",  
                "cloudwatch:GetMetricStatistics",  
                "cloudwatch>ListMetrics",  
                "cloudwatch:PutMetricAlarm",  
                "dynamodb:*",  
                "sns>CreateTopic",  
                "sns>DeleteTopic",  
                "sns>ListSubscriptions",  
                "sns>ListSubscriptionsByTopic",  
                "sns>ListTopics",  
                "sns:Subscribe",  
                "sns:Unsubscribe"  
            ],  
            "Effect": "Allow",  
            "Resource": "*",  
            "Sid": "DDBConsole"  
        },  
        ...remainder of document omitted...
```

Para restringir essa política, primeiro remova a seguinte linha:

```
"dynamodb:*",
```

Em seguida, construa uma nova Action que permita o acesso somente às tabelas Forum, Thread e Reply:

```
{  
    "Action": [  
        "dynamodb:*"  
    ],  
    "Effect": "Allow",  
    "Resource": [  
        "arn:aws:dynamodb:us-west-2:123456789012:table/Forum",  
        "arn:aws:dynamodb:us-west-2:123456789012:table/Thread",  
        "arn:aws:dynamodb:us-west-2:123456789012:table/Reply"  
    ]  
},
```

Note

Substituir `us-west-2` pela região na qual suas tabelas do DynamoDB residem. Substitua `123456789012` pelo número da sua conta da AWS.

Finalmente, adicione a nova Action ao documento de política:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "dynamodb:*"  
            ],  
            "Effect": "Allow",  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:123456789012:table/Forum",  
                "arn:aws:dynamodb:us-west-2:123456789012:table/Thread",  
                "arn:aws:dynamodb:us-west-2:123456789012:table/Reply"  
            ]  
        },  
        {  
            "Action": [  
                "cloudwatch:DeleteAlarms",  
                "cloudwatch:DescribeAlarmHistory",  
                "cloudwatch:DescribeAlarms",  
                "cloudwatch:DescribeAlarmsForMetric",  
                "cloudwatch:GetMetricStatistics",  
                "cloudwatch>ListMetrics",  
                "cloudwatch:PutMetricAlarm",  
                "sns>CreateTopic",  
                "sns>DeleteTopic",  
                "sns>ListSubscriptions",  
                "sns>ListSubscriptionsByTopic",  
                "sns>ListTopics",  
                "sns:Subscribe",  
                "sns:Unsubscribe"  
            ],  
            "Effect": "Allow",  
            "Resource": "*",  
            "Sid": "DDBConsole"  
        },  
        ...remainder of document omitted...  
    ]  
}
```

6. Quando estiver satisfeito com as configurações de política, clique em Create Policy (Criar política).

Depois de criar a política, você pode anexá-la a um usuário do IAM.

1. No painel do console do IAM, clique em Usuários e selecione o usuário que você deseja modificar.
2. Na guia Permissions (Permissões), clique em Attach Policy (Anexar política).
3. No painel Attach Policy (Anexar política), selecione o nome da sua política e clique em Attach Policy (Anexar política).

Note

É possível usar um procedimento semelhante para anexar sua política a uma função ou grupo em vez de a um usuário.

Exportação de dados do DynamoDB para o Amazon S3

Esta seção descreve como exportar dados de uma ou mais tabelas do DynamoDB para um bucket do Amazon S3. É necessário criar o bucket do Amazon S3 antes de realizar a exportação.

Important

Se você nunca usou AWS Data Pipeline Antes, você precisará configurar duas funções do IAM antes de seguir este procedimento. Para mais informações, consulte [Criação de funções do IAM para AWS Data Pipeline \(p. 1117\)](#).

1. Faça login no AWS Management Console e abra o AWS Data Pipeline console do em <https://console.aws.amazon.com/datapipeline/>.
2. Se você ainda não tem pipelines no AWS Região, selecione Comece a usar agora.

Caso contrário, se você já tem pelo menos um pipeline, escolha Create new pipeline (Criar um novo pipeline).

3. Na página Create Pipeline (Criar pipeline), faça o seguinte:
 - a. No campo Name (Nome), digite um nome para seu pipeline. Por exemplo: `MyDynamoDBExportPipeline`.
 - b. Para o parâmetro Source (Origem), selecione Build using a template (Criar usando um modelo). Na lista suspensa de modelos, escolha Export DynamoDB table to S3 (Exportar tabela do DynamoDB para o S3).
 - c. No Nome da tabela do DynamoDB de origem Digite o nome da tabela do DynamoDB que você deseja exportar.
 - d. No Pasta S3 de saída Insira um URI do Amazon S3 no qual o arquivo de exportação será gravado. Por exemplo: `s3://mybucket(exports`

O formato deste URI é `s3://bucketname/folder`, em que:

- `bucketname` É o nome do seu bucket do Amazon S3.
 - `folder` é o nome de uma pasta nesse bucket. Se a pasta não existir, ela será criada automaticamente. Se você não especificar um nome para a pasta, um nome será atribuído para ela no formulário `s3://bucketname/region tablename`.
- e. No Localização do S3 para logs Insira um URI do Amazon S3 no qual o arquivo de log da exportação será gravado. Por exemplo: `s3://mybucket/logs/`

O formato do URI de S3 Log Folder (Pasta de logs do S3) é idêntico ao de Output S3 Folder (Pasta de saída do S3). O URI deve ser resolvido para uma pasta. Arquivos de log não podem ser gravados no nível superior do bucket do S3.

4. Adicionar uma tag com a chave `dynamodb:datapipeline` e o Valor `true`.
5. Quando estiver satisfeito com as configurações, clique em Activate (Ativar).

Seu pipeline agora será criado; esse processo pode levar vários minutos para ser concluído. Você pode monitorar o progresso no console do AWS Data Pipeline.

Quando a exportação terminar, você poderá acessar o [Console do Amazon S3](#) para exibir o arquivo de exportação. O nome do arquivo de saída é um identificador valor sem extensão, como este exemplo: `ae10f955-fb2f-4790-9b11-fbfea01a871e_000000`. O formato interno desse arquivo está descrito em [Verificar Arquivo de Exportação de Dados](#) no AWS Data Pipeline Guia do desenvolvedor.

Importação de dados do Amazon S3 para o DynamoDB

Esta seção supõe que você já tenha exportado dados de uma tabela do DynamoDB e que o arquivo de exportação tenha sido gravado no seu bucket do Amazon S3. O formato interno desse arquivo está descrito em [Verificar Arquivo de Exportação de Dados](#) no AWS Data Pipeline Guia do desenvolvedor. Note que este é somente que o DynamoDB pode importar usando AWS Data Pipeline.

Usaremos os termos tabela de origem para a tabela original da qual os dados foram exportados e tabela de destino para a tabela que receberá os dados importados. Você pode importar dados de um arquivo de exportação no Amazon S3, com as seguintes condições:

- A tabela de destino já deve existir. (O processo de importação não criará a tabela para você.)
- A tabela de destino deve ter o mesmo esquema de chaves da tabela de origem.

A tabela de destino não precisa estar vazia. No entanto, o processo de importação substituirá todos os itens de dados na tabela que tiverem as mesmas chaves que os itens no arquivo de exportação. Por exemplo, suponha que você tenha uma tabela Customer com uma chave de CustomerId e que existam apenas três itens na tabela (CustomerId 1, 2 e 3). Se o arquivo de exportação também contiver itens de dados para CustomerID 1, 2 e 3, os itens na tabela de destino serão substituídos por aqueles do arquivo de exportação. Se o arquivo de exportação também contiver um item de dados para CustomerId 4, esse item será adicionado à tabela.

A tabela de destino pode estar em um AWS Região. Por exemplo, suponha que você tenha um Client na região Oeste dos EUA (Oregon) e exporte seus dados para o Amazon S3. Você poderia importar esses dados para um Client Tabela na região Europa (Irlanda). Esse processo é conhecido como exportação e importação entre regiões. Para uma lista de AWS Regiões, vá para [Regiões e endpoints](#) no AWS Referência geral.

Observe que o AWS Management Console permite exportar várias tabelas de origem ao mesmo tempo. No entanto, você só pode importar uma tabela de cada vez.

1. Faça login no AWS Management Console e abra o AWS Data Pipeline console do em <https://console.aws.amazon.com/datapipeline/>.
2. (Opcional) Se quiser realizar uma importação entre regiões, vá para o canto superior direito da janela e escolha a região de destino.
3. Escolha Create new pipeline (Criar um novo pipeline).
4. Na página Create Pipeline (Criar pipeline), faça o seguinte:
 - a. No campo Name (Nome), digite um nome para seu pipeline. Por exemplo: `MyDynamoDBImportPipeline`.
 - b. Para o parâmetro Source (Origem), selecione Build using a template (Criar usando um modelo). Na lista suspensa de modelos, escolha Import DynamoDB backup data from S3 (Importar dados de backup do DynamoDB do S3).
 - c. No campo S3 de entrada insira um URI do Amazon S3 no qual o arquivo de exportação pode ser encontrado. Por exemplo: `s3://mybucket(exports)`

O formato deste URI é `s3://bucketname/folder`, em que:

- `bucketname` é o nome do seu bucket do Amazon S3.
- `folder` é o nome da pasta que contém o arquivo de exportação.

O trabalho de importação espera encontrar um arquivo na localização especificada do Amazon S3. O formato interno do arquivo está descrito em [Verificar Arquivo de Exportação de Dados](#) no AWS Data Pipeline Guia do desenvolvedor.

- d. No Nome da tabela do DynamoDB de destino Digite o nome da tabela do DynamoDB na qual você deseja importar os dados.
- e. No Localização do S3 para logs Insira um URI do Amazon S3 no qual o arquivo de log da importação será gravado. Por exemplo: s3://mybucket/logs/

O formato do URI de S3 Log Folder (Pasta de logs do S3) é idêntico ao de Output S3 Folder (Pasta de saída do S3). O URI deve ser resolvido para uma pasta. Arquivos de log não podem ser gravados no nível superior do bucket do S3.

- f. Adicionar uma tag com a chave dynamodb:datapipeline e o valor true.
5. Quando estiver satisfeito com as configurações, clique em Activate (Ativar).

Seu pipeline agora será criado; esse processo pode levar vários minutos para ser concluído. O trabalho de importação começará imediatamente após a criação do pipeline.

Troubleshooting

Esta seção abrange alguns modos de falha básicos e métodos de solução de problemas para exportações do DynamoDB.

Se ocorrer um erro durante uma exportação ou importação, o status do pipeline no console do AWS Data Pipeline será exibido como **ERROR**. Se isso acontecer, clique no nome do pipeline com falha para acessar sua página de detalhes. Isso mostrará detalhes sobre todas as etapas no pipeline, bem como o status de cada um. Em particular, examine qualquer rastreamento de pilha de execução que você visualizar.

Por fim, acesse o bucket do Amazon S3 e procure qualquer arquivo de exportação ou importação que estejam gravados lá.

Veja a seguir alguns problemas comuns que podem fazer com que um pipeline falhe, juntamente com ações corretivas. Para diagnosticar seu pipeline, compare os erros que você viu com os problemas observados abaixo.

- Para uma importação, verifique se a tabela de destino já existe e tem o mesmo esquema de chave que a tabela de origem. Essas condições devem ser atendidas, ou a importação falhará.
- Certifique-se de que o pipeline tenha a tag `dynamodb:datapipeline`. Caso contrário, as chamadas de API do Amazon EMR não terão sucesso.
- Certifique-se de que o bucket do Amazon S3 especificado tenha sido criado e de ter permissões de leitura e gravação nele.
- O pipeline pode ter excedido o tempo limite de execução. (Você definiu esse parâmetro quando criou o pipeline.) Por exemplo, você pode ter definido o tempo limite de execução para 1 hora, mas o trabalho de exportação pode ter exigido mais tempo do que isso. Tente excluir e depois recriar o pipeline, mas com um intervalo de tempo limite de execução mais longo dessa vez.
- Atualize o arquivo manifesto se você estiver restaurando de um bucket do Amazon S3 que não seja o bucket original do qual a exportação foi realizada (que contém uma cópia da exportação).
- Talvez você não tenha as permissões corretas para realizar uma exportação ou importação. Para mais informações, consulte [Pré-requisitos para exportar e importar dados \(p. 1117\)](#).
- Você pode ter atingido uma cota de recursos em seu AWS Como o número máximo de instâncias do Amazon EC2 ou o número máximo de AWS Data Pipeline pipelines. Para obter mais informações, incluindo como solicitar o aumento dessas cotas, consulte [AWS Cotas de serviço do AWS Referência geral](#).

Note

Para obter mais detalhes sobre como solucionar problemas com um pipeline, acesse [Solução de problemas do AWS Data Pipeline](#) no AWS Data Pipeline Guia do desenvolvedor.

Modelos predefinidos para AWS Data Pipelinee DynamoDB

Se quiser ter uma compreensão mais profunda de como AWS Data Pipeline Para realizar o trabalho, recomendamos que você consulte o AWS Data Pipeline Guia do desenvolvedor. Este guia contém tutoriais passo a passo para criar e trabalhar com pipelines. Esses tutoriais podem ser usados como pontos de partida para você criar seus próprios pipelines. Convém ler o tutorial do AWS Data Pipeline, que fornece orientação pelas etapas necessárias para criar um pipeline de importação e exportação que você pode personalizar para os seus requisitos. Consulte [Tutorial: Importar e exportar o Amazon DynamoDB usando o AWS Data Pipeline](#) no AWS Data Pipeline Guia do desenvolvedor.

AWS Data Pipeline O oferece vários modelos para criar pipelines. Os seguintes modelos são relevantes para o DynamoDB.

Exportação de dados entre o DynamoDB e o Amazon S3

Note

O Console do DynamoDB agora oferece suporte ao seu próprio fluxo de Exportação para o Amazon S3. No entanto, ele não é compatível com o AWS Data Pipeline fluxo de importação. Para obter mais informações, consulte [Exportando dados de tabela do DynamoDB para o Amazon S3 \(p. 1046\)](#) e a publicação no blog [Exportar dados de tabela do Amazon DynamoDB para seu data lake no Amazon S3, sem necessidade de gravação de código](#).

O AWS Data Pipeline O console do fornece dois modelos predefinidos para a exportação de dados entre o DynamoDB e o Amazon S3. Para obter mais informações sobre esses modelos, consulte as seguintes seções do AWS Data Pipeline Guia do desenvolvedor:

- Exportar o DynamoDB para o Amazon S3
- Exportar o Amazon S3 para o DynamoDB

Back-end de armazenamento do Amazon DynamoDB para Titan

O DynamoDB Storage Backend para Titan foi substituído pelo Amazon DynamoDB JanusGraph Backend para JanusGraph, que está disponível no [GitHub](#).

Para obter instruções atualizadas sobre o DynamoDB Storage Backend para JanusGraph, consulte o [README.md](#) file.

Palavras reservadas no DynamoDB

As seguintes palavras-chave são reservadas para uso pelo DynamoDB. Não use nenhuma destas palavras como nomes de atributos em expressões. Essa lista não diferencia maiúsculas de minúsculas.

Se precisar escrever uma expressão que contém um nome de atributo em conflito com uma palavra reservada do DynamoDB, você poderá definir um nome de atributo de expressão para uso no lugar dessa palavra reservada. Para obter mais informações, consulte [Nomes de atributo de expressão no DynamoDB \(p. 418\)](#).

```
ABORT
ABSOLUTE
ACTION
ADD
AFTER
AGENT
AGGREGATE
ALL
ALLOCATE
ALTER
ANALYZE
AND
ANY
ARCHIVE
ARE
ARRAY
AS
ASC
ASCII
ASENSITIVE
ASSERTION
ASYMMETRIC
AT
ATOMIC
ATTACH
ATTRIBUTE
AUTH
AUTHORIZATION
AUTHORIZE
AUTO
AVG
BACK
BACKUP
BASE
BATCH
BEFORE
BEGIN
BETWEEN
BIGINT
BINARY
BIT
BLOB
BLOCK
BOOLEAN
BOTH
BREADTH
BUCKET
BULK
BY
BYTE
CALL
CALLED
CALLING
CAPACITY
CASCADE
CASCADED
CASE
CAST
CATALOG
```

CHAR
CHARACTER
CHECK
CLASS
CLOB
CLOSE
CLUSTER
CLUSTERED
CLUSTERING
CLUSTERS
COALESCE
COLLATE
COLLATION
COLLECTION
COLUMN
COLUMNS
COMBINE
COMMENT
COMMIT
COMPACT
COMPILE
COMPRESS
CONDITION
CONFLICT
CONNECT
CONNECTION
CONSISTENCY
CONSISTENT
CONSTRAINT
CONSTRAINTS
CONSTRUCTOR
CONSUMED
CONTINUE
CONVERT
COPY
CORRESPONDING
COUNT
COUNTER
CREATE
CROSS
CUBE
CURRENT
CURSOR
CYCLE
DATA
DATABASE
DATE
DATETIME
DAY
DEALLOCATE
DEC
DECIMAL
DECLARE
DEFAULT
DEFERRABLE
DEFERRED
DEFINE
DEFINED
DEFINITION
DELETE
DELIMITED
DEPTH
DEREF
DESC
DESCRIBE
DESCRIPTOR

```
DETACH
DETERMINISTIC
DIAGNOSTICS
DIRECTORIES
DISABLE
DISCONNECT
DISTINCT
DISTRIBUTE
DO
DOMAIN
DOUBLE
DROP
DUMP
DURATION
DYNAMIC
EACH
ELEMENT
ELSE
ELSEIF
EMPTY
ENABLE
END
EQUAL
EQUALS
ERROR
ESCAPE
ESCAPED
EVAL
EVALUATE
EXCEEDED
EXCEPT
EXCEPTION
EXCEPTIONS
EXCLUSIVE
EXEC
EXECUTE
EXISTS
EXIT
EXPLAIN
EXPLODE
EXPORT
EXPRESSION
EXTENDED
EXTERNAL
EXTRACT
FAIL
FALSE
FAMILY
FETCH
FIELDS
FILE
FILTER
FILTERING
FINAL
FINISH
FIRST
FIXED
FLATTERN
FLOAT
FOR
FORCE
FOREIGN
FORMAT
FORWARD
FOUND
FREE
```

```
FROM
FULL
FUNCTION
FUNCTIONS
GENERAL
GENERATE
GET
GLOB
GLOBAL
GO
GOTO
GRANT
GREATER
GROUP
GROUPING
HANDLER
HASH
HAVE
HAVING
HEAP
HIDDEN
HOLD
HOUR
IDENTIFIED
IDENTITY
IF
IGNORE
IMMEDIATE
IMPORT
IN
INCLUDING
INCLUSIVE
INCREMENT
INCREMENTAL
INDEX
INDEXED
INDEXES
INDICATOR
INFINITE
INITIALLY
INLINE
INNER
INNTER
INOUT
INPUT
INSENSITIVE
INSERT
INSTEAD
INT
INTEGER
INTERSECT
INTERVAL
INTO
INVALIDATE
IS
ISOLATION
ITEM
ITEMS
ITERATE
JOIN
KEY
KEYS
LAG
LANGUAGE
LARGE
LAST
```

LATERAL
LEAD
LEADING
LEAVE
LEFT
LENGTH
LESS
LEVEL
LIKE
LIMIT
LIMITED
LINES
LIST
LOAD
LOCAL
LOCALTIME
LOCALTIMESTAMP
LOCATION
LOCATOR
LOCK
LOCKS
LOG
LOGED
LONG
LOOP
LOWER
MAP
MATCH
MATERIALIZED
MAX
MAXLEN
MEMBER
MERGE
METHOD
METRICS
MIN
MINUS
MINUTE
MISSING
MOD
MODE
MODIFIES
MODIFY
MODULE
MONTH
MULTI
MULTISET
NAME
NAMES
NATIONAL
NATURAL
NCHAR
NCLOB
NEW
NEXT
NO
NONE
NOT
NULL
NULLIF
NUMBER
NUMERIC
OBJECT
OF
OFFLINE
OFFSET

```
OLD
ON
ONLINE
ONLY
OPAQUE
OPEN
OPERATOR
OPTION
OR
ORDER
ORDINALITY
OTHER
OTHERS
OUT
OUTER
OUTPUT
OVER
OVERLAPS
OVERRIDE
OWNER
PAD
PARALLEL
PARAMETER
PARAMETERS
PARTIAL
PARTITION
PARTITIONED
PARTITIONS
PATH
PERCENT
PERCENTILE
PERMISSION
PERMISSIONS
PIPE
PIPELINED
PLAN
POOL
POSITION
PRECISION
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVATE
PRIVILEGES
PROCEDURE
PROCESSED
PROJECT
PROJECTION
PROPERTY
PROVISIONING
PUBLIC
PUT
QUERY
QUIT
QUORUM
RAISE
RANDOM
RANGE
RANK
RAW
READ
READS
REAL
REBUILD
RECORD
```

RECURSIVE
REDUCE
REF
REFERENCE
REFERENCES
REFERENCING
REGEXP
REGION
REINDEX
RELATIVE
RELEASE
REMAINDER
RENAME
REPEAT
REPLACE
REQUEST
RESET
RESIGNAL
RESOURCE
RESPONSE
RESTORE
RESTRICT
RESULT
RETURN
RETURNING
RETURNS
REVERSE
REVOKE
RIGHT
ROLE
ROLES
ROLLBACK
ROLLUP
ROUTINE
ROW
ROWS
RULE
RULES
SAMPLE
SATISFIES
SAVE
SAVEPOINT
SCAN
SCHEMA
SCOPE
SCROLL
SEARCH
SECOND
SECTION
SEGMENT
SEGMENTS
SELECT
SELF
SEMI
SENSITIVE
SEPARATE
SEQUENCE
SERIALIZABLE
SESSION
SET
SETS
SHARD
SHARE
SHARED
SHORT
SHOW

SIGNAL
SIMILAR
SIZE
SKEWED
SMALLINT
SNAPSHOT
SOME
SOURCE
SPACE
SPACES
SPARSE
SPECIFIC
SPECIFICTYPE
SPLIT
SQL
SQLCODE
SQLERROR
SQLEXCEPTION
SQLSTATE
SQLWARNING
START
STATE
STATIC
STATUS
STORAGE
STORE
STORED
STREAM
STRING
STRUCT
STYLE
SUB
SUBMULTISET
SUBPARTITION
SUBSTRING
SUBTYPE
SUM
SUPER
SYMMETRIC
SYNONYM
SYSTEM
TABLE
TABLESAMPLE
TEMP
TEMPORARY
TERMINATED
TEXT
THAN
THEN
THROUGHPUT
TIME
TIMESTAMP
TIMEZONE
TINYINT
TO
TOKEN
TOTAL
TOUCH
TRAILING
TRANSACTION
TRANSFORM
TRANSLATE
TRANSLATION
TREAT
TRIGGER
TRIM

```
TRUE
TRUNCATE
TTL
TUPLE
TYPE
UNDER
UNDO
UNION
UNIQUE
UNIT
UNKNOWN
UNLOGGED
UNNEST
UNPROCESSED
UNSIGNED
UNTIL
UPDATE
UPPER
URL
USAGE
USE
USER
USERS
USING
UUID
VACUUM
VALUE
VALUED
VALUES
VARCHAR
VARIABLE
VARIANCE
VARINT
VARYING
VIEW
VIEWS
VIRTUAL
VOID
WAIT
WHEN
WHENEVER
WHERE
WHILE
WINDOW
WITH
WITHIN
WITHOUT
WORK
WRAPPED
WRITE
YEAR
ZONE
```

Parâmetros condicionais herdados

Esta seção compara os parâmetros condicionais herdados com os parâmetros de expressão no DynamoDB.

Com a introdução dos parâmetros de expressão (consulte [Uso de expressões no DynamoDB \(p. 414\)](#)), vários parâmetros mais antigos foram substituídos. Novos aplicativos não devem usar esses parâmetros herdados, mas devem usar os parâmetros de expressão. (Para obter mais informações, consulte [Uso de expressões no DynamoDB \(p. 414\)](#).)

Note

O DynamoDB não permite combinar parâmetros condicionais herdados e parâmetros de expressão em uma única chamada. Por exemplo, chamar a operação `Query` com `AttributesToGet` e `ConditionExpression` resultará em um erro.

A tabela a seguir mostra as APIs do DynamoDB, que ainda dão suporte a esses parâmetros herdados, e qual parâmetro de expressão usar. Esta tabela pode ser útil se você estiver considerando atualizar os aplicativos para que eles usem os parâmetros de expressão.

Se você usa esta API...	Com estes parâmetros herdados...	Use este parâmetro de expressão em vez disso
<code>BatchGetItem</code>	<code>AttributesToGet</code>	<code>ProjectionExpression</code>
<code>DeleteItem</code>	<code>Expected</code>	<code>ConditionExpression</code>
<code>GetItem</code>	<code>AttributesToGet</code>	<code>ProjectionExpression</code>
<code>PutItem</code>	<code>Expected</code>	<code>ConditionExpression</code>
<code>Query</code>	<code>AttributesToGet</code>	<code>ProjectionExpression</code>
	<code>KeyConditions</code>	<code>KeyConditionExpression</code>
	<code>QueryFilter</code>	<code>FilterExpression</code>
<code>Scan</code>	<code>AttributesToGet</code>	<code>ProjectionExpression</code>
	<code>ScanFilter</code>	<code>FilterExpression</code>
<code>UpdateItem</code>	<code>AttributeUpdates</code>	<code>UpdateExpression</code>
	<code>Expected</code>	<code>ConditionExpression</code>

As seções a seguir oferecem mais informações sobre os parâmetros condicionais herdados.

Tópicos

- [AttributesToGet \(p. 1135\)](#)
- [AttributeUpdates \(p. 1136\)](#)
- [ConditionalOperator \(p. 1138\)](#)
- [Expected \(p. 1138\)](#)
- [KeyConditions \(p. 1141\)](#)
- [QueryFilter \(p. 1143\)](#)
- [ScanFilter \(p. 1145\)](#)
- [Criação de condições com parâmetros herdados \(p. 1146\)](#)

AttributesToGet

`AttributesToGet` é um conjunto de um ou mais atributos a serem recuperados do DynamoDB. Se os nomes de atributo não forem fornecidos, todos os atributos serão retornados. Se qualquer um dos atributos solicitados não for encontrado, ele não aparecerá no resultado.

`AttributesToGet` permite que você recupere atributos do tipo List ou Map; no entanto, ele não pode recuperar elementos individuais em uma lista ou um mapa.

Observe que `AttributesToGet` não afeta o consumo de throughput provisionado. O DynamoDB determina as unidades de capacidade consumidas com base no tamanho do item, e não na quantidade de dados que são retornados para um aplicativo.

Usar o `ProjectionExpression` INST

Suponha que você queira recuperar um item do `Música`, mas você só queria retornar alguns dos atributos. Você poderia usar uma solicitação `GetItem` com um parâmetro `AttributesToGet`, como neste exemplo da AWS CLI:

```
aws dynamodb get-item \
  --table-name Music \
  --attributes-to-get '[ "Artist", "Genre" ]' \
  --key '{
    "Artist": {"S": "No One You Know"},
    "SongTitle": {"S": "Call Me Today"}
}'
```

Mas você também poderia usar uma `ProjectionExpression` em vez disso:

```
aws dynamodb get-item \
  --table-name Music \
  --projection-expression "Artist, Genre" \
  --key '{
    "Artist": {"S": "No One You Know"},
    "SongTitle": {"S": "Call Me Today"}
}'
```

AttributeUpdates

Em um `UpdateItem` operação, `AttributeUpdates` Os nomes de atributos a serem modificados, a ação a ser realizada em cada um e o novo valor de cada um. Se você estiver atualizando um atributo que seja um atributo de chave de índice de quaisquer índices dessa tabela, o tipo de atributo deverá coincidir com o tipo de chave de índice definido na `AttributesDefinition` da descrição da tabela. Você pode usar `UpdateItem` para atualizar todos os atributos que não são chave.

Os valores de atributo não podem ser nulos. Os atributos do tipo String e Binary devem ter tamanhos maior que zero. Os atributos do tipo Set não podem ficar vazios. As solicitações com valores vazios serão rejeitadas com uma exceção `ValidationException`.

Cada elemento `AttributeUpdates` consiste em um nome de atributo a ser modificado, junto com o seguinte:

- `Value` - O novo valor, se aplicável, desse atributo.
- `Action` - Um valor que especifica como executar a atualização. Essa ação só é válida para um atributo existente cujo tipo de dados é Number ou é Set; não use ADD para outros tipos de dados.

Se um item com a chave primária especificada for encontrado na tabela, os seguintes valores executam as ações a seguir:

- `PUT` - adiciona o atributo especificado ao item. Se o atributo já existir, ele será substituído pelo novo valor.
- `DELETE` - remove o atributo e seus valores, se nenhum valor for especificado para `DELETE`. O tipo de dados do valor especificado deve corresponder ao tipo de dados do valor existente.

Se um conjunto de valores for especificado, esses valores serão subtraídos do conjunto antigo. Por exemplo, se o valor do atributo era o conjunto `[a, b, c]` e a `DELETE` ação especifica `[a, c]`, então o valor final do atributo será `[b]`. Especificar um conjunto vazio é um erro.

- ADD- adiciona o valor especificado ao item, se o atributo ainda não existir. Se o atributo não existir, o comportamento de ADD dependerá do tipo de dados do atributo:
 - Se o atributo existente for um número, e se value é também um número, então value é adicionado matematicamente ao atributo existente. Se value for um número negativo, ele será subtraído do atributo existente.

Note

Se você usar ADD para aumentar ou reduzir um valor de número de um item que não existe antes da atualização, o DynamoDB usará 0 como o valor inicial.

Da mesma forma, se você usar ADD para um item existente para aumentar ou diminuir um valor de atributo que não existia antes da atualização, o DynamoDB usará 0 como o valor inicial. Por exemplo, suponha que o item que você deseja atualizar não tenha um atributo chamado contagem de itens, mas você decide ADD o número 3 para este atributo de qualquer maneira. O DynamoDB criará a contagem de itens atributo, defina seu valor inicial como 0, por fim, adicionar 3 a ele. O resultado será um novo atributo itemcount, com um valor 3.

- Se o tipo de dados existente for um conjunto, e se value também é um conjunto, então value é anexado ao conjunto existente. Por exemplo, se o valor do atributo é o conjunto [1, 2], e o ADDação especificada [3], então o valor final do atributo será [1, 2, 3]. Ocorrerá um erro se uma ação ADD for especificada para um atributo de conjunto e o tipo de atributo especificado não corresponder ao tipo de conjunto existente.

Ambos os conjuntos possuem o mesmo tipo de dados primitivo. Por exemplo, se o tipo de dados existente for um conjunto de strings, Value também deve ser um conjunto de strings.

Se nenhum item com a chave especificada for encontrado na tabela, os seguintes valores executam as ações a seguir:

- PUT - faz com que o DynamoDB crie um novo item com a chave primária especificada e, em seguida, adiciona o atributo.
- DELETE- nada acontece, pois os atributos não podem ser excluídos de um item inexistente. A operação é bem-sucedida, mas o DynamoDB não cria um novo item.
- ADD- faz com que o DynamoDB crie um item com o número e a chave primária fornecidos (ou conjunto de números) para o valor de atributo. Os únicos tipos de dados permitidos são Number e Number Set.

Se você fornecer quaisquer atributos que sejam parte de uma chave de índice, então, os tipos de dados desses atributos devem corresponder aos do esquema na definição de atributo da tabela.

Usar o UpdateExpressionINST

Suponha que você quisesse modificar um item no Músicatabela. Você poderia usar uma solicitação UpdateItem com um parâmetro AttributeUpdates, como neste exemplo da AWS CLI:

```
aws dynamodb update-item \
  --table-name Music \
  --key '{
    "SongTitle": {"S":"Call Me Today"},
    "Artist": {"S":"No One You Know"}
}' \
  --attribute-updates '{
    "Genre": {
      "Action": "PUT",
      "Value": {"S":"Rock"}
    }
}'
```

Mas você também poderia usar uma UpdateExpression em vez disso:

```
aws dynamodb update-item \
    --table-name Music \
    --key '{
        "SongTitle": {"S":"Call Me Today"},
        "Artist": {"S":"No One You Know"}
    }' \
    --update-expression 'SET Genre = :g' \
    --expression-attribute-values '{
        ":g": {"S":"Rock"}
    }'
```

ConditionalOperator

Um operador lógico para aplicar às condições em um mapa `Expected`, `QueryFilter` ou `ScanFilter`:

- AND – Se todas as condições forem avaliadas como verdadeiras, o mapa inteiro será avaliado como verdadeiro.
- OR – Se pelo menos uma das condições for avaliada como verdadeira, o mapa inteiro será avaliado como verdadeiro.

Se você omitir `ConditionalOperator`, então, AND será o padrão.

A operação será bem-sucedida somente se o mapa inteiro for avaliado como verdadeiro.

Note

Este parâmetro não tem suporte a atributos do tipo List ou Map.

Expected

`Expected` é um bloco condicional para uma `UpdateItem` operação. `Expected` é um mapa de pares de atributo/condição. Cada elemento do mapa consiste em um nome de atributo, um operador de comparação e um ou mais valores. O DynamoDB compara o atributo com o(s) valor(es) que você forneceu, usando o operador de comparação. Para cada elemento `Expected`, o resultado da avaliação é verdadeiro ou falso.

Se você especificar mais de um elemento na caixa `Expected`, por padrão, todas as condições devem ser avaliadas como verdadeiras. Em outras palavras, as condições são associadas. (Você pode usar o `ConditionalOperator` parâmetro para OU as condições. Se você fizer isso, pelo menos uma das condições deve ser avaliada como verdadeira, em vez de todas elas.)

Se o mapa `Expected` for avaliado como verdadeiro, a operação condicional será bem-sucedida; caso contrário, há uma falha.

`Expected` contém o seguinte:

- `AttributeValueList` - um ou mais valores para avaliar em relação ao atributo fornecido. O número de valores na lista depende do `ComparisonOperator` que está sendo usado.

Para o tipo Number, as comparações de valor são numéricas.

As comparações de valor String para "maior que", "igual a" ou "menor que" são baseadas em Unicode com codificação UTF-8 binária. Por exemplo, a é maior que A, e a é maior que B.

Para o tipo Binary, o DynamoDB trata cada byte de dados binários como não assinados ao comparar valores binários.

- `ComparisonOperator` - um comparador para avaliar atributos no `AttributeValueList`. Ao executar a comparação, o DynamoDB usa leituras fortemente consistentes.

Os seguintes operadores de comparação estão disponíveis:

`EQ | NE | LE | LT | GE | GT | NOT_NULL | NULL | CONTAINS | NOT_CONTAINS |
BEGINS_WITH | IN | BETWEEN`

Veja a seguir as descrições de cada operador de comparação.

- `EQ`: Igual.`EQ`tem suporte a todos os tipos de dados, incluindo listas e mapas.

AttributeValueListEle pode conter apenas umaAttributeValueelemento do tipo String, Number, Binary, String Set. Se um item contiver umAttributeValueelemento de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não é igual a { "NS": ["6", "2", "1"] }.

- `NE`: Não é igual.`NE`tem suporte a todos os tipos de dados, incluindo listas e mapas.

AttributeValueListEle pode conter apenas umaAttributeValueelemento do tipo String, Number, Binary, String Set. Se um item contiver umAttributeValueelemento de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não é igual a { "NS": ["6", "2", "1"] }.

- `LE` : Menor ou igual a.

AttributeValueListEle pode conter apenas umaAttributeValueelemento do tipo String, Number ou Binary (não um tipo Set). Se um item contiver umAttributeValueelemento de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.

- `LT` : Menor que.

AttributeValueListEle pode conter apenas umaAttributeValueelemento do tipo String, Number ou Binary (não um tipo Set). Se um item contiver umAttributeValueelemento de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.

- `GE` : Maior ou igual a.

AttributeValueListEle pode conter apenas umaAttributeValueelemento do tipo String, Number ou Binary (não um tipo Set). Se um item contiver umAttributeValueelemento de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.

- `GT` : Maior que.

AttributeValueListEle pode conter apenas umaAttributeValueelemento do tipo String, Number ou Binary (não um tipo Set). Se um item contiver umAttributeValueelemento de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.

- `NOT_NULL`: o atributo existe.`NOT_NULL`Item suporte a todos os tipos de dados, incluindo listas e mapas.

Note

Este operador testa a existência de um atributo, não o tipo de dados. Se o tipo de dados do atributo "a" é nulo, e você o avalia usando`NOT_NULL`, o resultado é um booleano`true`. Isso acontece porque o atributo "a" existe; o tipo de dados não é relevante para o operador de comparação `NOT_NULL`.

- `NULL`: o atributo não existe.`NULL`Item suporte a todos os tipos de dados, incluindo listas e mapas.

Note

Este operador testa a não existência de um atributo, não o tipo de dados. Se o tipo de dados do atributo "a" é nulo, e você o avalia usando`NULL`, o resultado é um booleano`false`. Isso

acontece porque o atributo "a" existe; o tipo de dados não é relevante para o operador de comparação **NULL**.

- **CONTAINS** : verifica uma subsequência ou valor em um conjunto.

AttributeValueListEle pode conter apenas uma **AttributeValue**elemento do tipo String, Number ou Binary (não um tipo Set). Se o atributo de destino da comparação for do tipo String, o operador procurará uma substring correspondente. Se o atributo de destino da comparação for do tipo Binary, o operador procurará uma subsequência do destino que corresponda à entrada. Se o atributo de destino da comparação for um conjunto ("SS", "NS" ou "BS"), o operador será avaliado como verdadeiro, se ele encontrar uma correspondência exata com qualquer membro do conjunto.

CONTAINS é suportado para listas: Ao avaliar "a **CONTAINS** b", "a"pode ser uma lista; no entanto, "b"não pode ser um conjunto, um mapa ou uma lista.

- **NOT_CONTAINS** : verifica a ausência de uma subsequência ou a ausência de um valor em um conjunto.

AttributeValueListEle pode conter apenas uma **AttributeValue**elemento do tipo String, Number ou Binary (não um tipo Set). Se o atributo de destino da comparação for String, o operador verificará a ausência de uma substring correspondente. Se o atributo de destino da comparação for Binary, o operador verificará a ausência de uma subsequência do destino que corresponda à entrada. Se o atributo de destino da comparação for um conjunto ("SS", "NS" ou "BS"), o operador será avaliado como verdadeiro se ele does **not** encontrar uma correspondência exata com qualquer membro do conjunto.

NOT_CONTAINS é suportado para listas: Ao avaliar "a **NOT CONTAINS** b", "a"pode ser uma lista; no entanto, "b"não pode ser um conjunto, um mapa ou uma lista.

- **BEGINS_WITH** : procura um prefixo.

AttributeValueListEle pode conter apenas uma **AttributeValue**do tipo String ou Binary (não um tipo Number ou Set). O atributo de destino da comparação deve ser do tipo String ou Binary (e não Number ou um tipo de conjunto).

- **IN** : procura elementos correspondentes dentro de dois conjuntos.

AttributeValueListpode conter uma ou mais **AttributeValue**elements do tipo String, Number ou Binary (não um tipo Set). Esses atributos são comparados com um atributo do tipo Set existente de um item. Se quaisquer elementos do conjunto de entradas estiverem presentes no atributo do item, a expressão será avaliada como verdadeira.

- **BETWEEN** : maior ou igual ao primeiro valor e menor ou igual ao segundo valor.

AttributeValueListdeve conter dois **AttributeValue**elements do mesmo tipo, seja String, Number ou Binary (não um tipo Set). Um atributo de destino corresponde se o valor de destino for maior que, ou igual, ao primeiro elemento e menor que, ou igual, ao segundo elemento. Se um item contiver um **AttributeValue**de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo, {"S": "6"}Não se compara a {"N": "6"}. Além disso, {"N": "6"} não se compara a {"NS": ["6", "2", "1"]}

Os parâmetros a seguir podem ser usados em vez de **AttributeValueList** e **ComparisonOperator**:

- **Value** - um valor para o DynamoDB comparar com um atributo.
- **Exists** - um valor Booleano que faz com que o DynamoDB avalie o valor antes de tentar a operação condicional:
- Se **Exists**étrue, o DynamoDB verificará se esse valor de atributo já existe na tabela. Se ele for encontrado, a condição será avaliada como verdadeira, caso contrário, a condição será avaliada como falsa.

- Se `Exists` é `false`, o DynamoDB irá pressupor que o valor do atributo não existem na tabela. Se, na verdade, o valor não existir, a suposição será válida e a condição será avaliada como verdadeira. Se o valor for encontrado, apesar da suposição de que ele não existe, a condição será avaliada como falsa.

Observe que o valor padrão para `Exists` é `true`.

O `Value` e `Exists` Os parâmetros do são incompatíveis com `AttributeValueList` e `ComparisonOperator`. Observe que se você usar os dois conjuntos de parâmetros de uma só vez, o DynamoDB retornará uma exceção `ValidationException`.

Note

Este parâmetro não tem suporte a atributos do tipo List ou Map.

Usar o ConditionExpression INST

Suponha que você quisesse modificar um item no `Música`, mas somente se uma determinada condição fosse verdadeira. Você poderia usar uma solicitação `UpdateItem` com um parâmetro `Expected`, como neste exemplo da AWS CLI:

```
aws dynamodb update-item \
  --table-name Music \
  --key '{
    "Artist": {"S": "No One You Know"},
    "SongTitle": {"S": "Call Me Today"}
}' \
  --attribute-updates '{
    "Price": {
      "Action": "PUT",
      "Value": {"N": "1.98"}
    }
}' \
  --expected '{
    "Price": {
      "ComparisonOperator": "LE",
      "AttributeValueList": [ {"N": "2.00"} ]
    }
}'
```

Mas você também poderia usar uma `ConditionExpression` em vez disso:

```
aws dynamodb update-item \
  --table-name Music \
  --key '{
    "Artist": {"S": "No One You Know"},
    "SongTitle": {"S": "Call Me Today"}
}' \
  --update-expression 'SET Price = :p1' \
  --condition-expression 'Price <= :p2' \
  --expression-attribute-values '{
    ":p1": {"N": "1.98"},
    ":p2": {"N": "2.00"}
}'
```

KeyConditions

`KeyCondition`s são os critérios de seleção para uma `query` operação. Para consultar uma tabela, você pode ter condições apenas em atributos de chave primária da tabela. Você deve fornecer o nome e o valor

da chave de partição como um `EQ`Condição. Opcionalmente, você pode fornecer uma segunda condição, referente à chave de classificação.

Note

Se você não fornecer uma condição de chave de classificação, todos os itens que correspondem à chave de partição serão recuperados. Se `FilterExpression` ou `QueryFilter` estiver presente, isso será aplicado depois que os itens forem recuperados.

Para consultar um índice, você pode ter condições apenas em atributos de chave do índice. Você deve fornecer o nome e o valor da chave de partição do índice como um `EQ`Condição. Opcionalmente, você pode fornecer uma segunda condição, referente à chave de classificação do índice.

Cada elemento `KeyConditions` consiste em um nome de atributo a ser comparado, junto com o seguinte:

- `AttributeValueList`- um ou mais valores para avaliar em relação ao atributo fornecido. O número de valores na lista depende do `ComparisonOperator` que está sendo usado.

Para o tipo Number, as comparações de valor são numéricas.

As comparações de valor String para "maior que", "igual a" ou "menor que" são baseadas em Unicode com codificação UTF-8 binária. Por exemplo, a é maior que A, e a é maior que B.

Para Binary, o DynamoDB trata cada byte de dados binários como não assinados ao comparar valores binários.

- `ComparisonOperator` - um comparador para avaliar atributos, por exemplo, é igual a, maior que, menor que e assim por diante.

Para `KeyConditions`, somente os seguintes operadores de comparação têm suporte:

`EQ` | `LE` | `LT` | `GE` | `GT` | `BEGINS_WITH` | `BETWEEN`

Veja a seguir as descrições desses operadores de comparação.

- `EQ` : Igual.

`AttributeValueList`Ele pode conter apenas uma `AttributeValue`do tipo String, Number ou Binary (não um tipo Set). Se um item contiver um `AttributeValue`de um tipo diferente do especificado na solicitação, os valores não coincidem. Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não é igual a { "NS": ["6", "2", "1"] }.

- `LE` : Menor ou igual a.

`AttributeValueList`Ele pode conter apenas uma `AttributeValue`elemento do tipo String, Number ou Binary (não um tipo Set). Se um item contiver um `AttributeValue`de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.

- `LT` : Menor que.

`AttributeValueList`Ele pode conter apenas uma `AttributeValue`do tipo String, Number ou Binary (não um tipo Set). Se um item contiver um `AttributeValue`de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.

- `GE` : Maior ou igual a.

`AttributeValueList`Ele pode conter apenas uma `AttributeValue`elemento do tipo String, Number ou Binary (não um tipo Set). Se um item contiver um `AttributeValue`de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.

- **GT** : Maior que.

AttributeValueList Ele pode conter apenas uma **AttributeValue** elemento do tipo String, Number ou Binary (não um tipo Set). Se um item contiver um **AttributeValue** de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo, { "S": "6" } Não igual { "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.

- **BEGINS_WITH** : procura um prefixo.

AttributeValueList Ele pode conter apenas uma **AttributeValue** do tipo String ou Binary (não um tipo Number ou Set). O atributo de destino da comparação deve ser do tipo String ou Binary (e não Number ou um tipo de conjunto).

- **BETWEEN** : maior ou igual ao primeiro valor e menor ou igual ao segundo valor.

AttributeValueList deve conter dois **AttributeValue** elementos do mesmo tipo, seja String, Number ou Binary (não um tipo Set). Um atributo de destino corresponde se o valor de destino for maior que, ou igual, ao primeiro elemento e menor que, ou igual, ao segundo elemento. Se um item contiver um **AttributeValue** de um tipo diferente do fornecido na solicitação, os valores não coincidem. Por exemplo, { "S": "6" } Não se compara a { "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }

Usar o KeyConditionExpression INST

Suponha que você quisesse recuperar vários itens com a mesma chave de partição da tabela Música. Você poderia usar uma solicitação **Query** com um parâmetro **KeyConditions**, como neste exemplo da AWS CLI:

```
aws dynamodb query \
    --table-name Music \
    --key-conditions '{
        "Artist": {
            "ComparisonOperator": "EQ",
            "AttributeValueList": [ {"S": "No One You Know"} ]
        },
        "SongTitle": {
            "ComparisonOperator": "BETWEEN",
            "AttributeValueList": [ {"S": "A"}, {"S": "M"} ]
        }
    }'
```

Mas você também poderia usar uma **KeyConditionExpression** em vez disso:

```
aws dynamodb query \
    --table-name Music \
    --key-condition-expression 'Artist = :a AND SongTitle BETWEEN :t1 AND :t2' \
    --expression-attribute-values '{
        ":a": {"S": "No One You Know"},
        ":t1": {"S": "A"},
        ":t2": {"S": "M"}
    }'
```

QueryFilter

Em uma operação **Query**, **QueryFilter** é uma condição que avalia os resultados da consulta depois que os itens são lidos e retorna apenas os valores desejados.

Este parâmetro não tem suporte a atributos do tipo List ou Map.

Note

Um `QueryFilter` é aplicado depois que os itens já foram lidos, o processo de filtragem não consome unidades de capacidade de leitura adicionais.

Se você fornecer mais de uma condição no `QueryFilter`, por padrão, todas as condições devem ser avaliadas como verdadeiras. Em outras palavras, as condições são associadas. (Você pode usar o `ConditionalOperator` (p. 1138) parâmetro para OU as condições. Se você fizer isso, pelo menos uma das condições deve ser avaliada como verdadeira, em vez de todas elas.)

Observe que `QueryFilter` não permite atributos de chave. Você não pode definir uma condição de filtro em uma chave de partição ou uma chave de classificação.

Cada elemento `QueryFilter` consiste em um nome de atributo a ser comparado, junto com o seguinte:

- `AttributeValueList`- um ou mais valores para avaliar em relação ao atributo fornecido. O número de valores na lista depende do operador especificado em `ComparisonOperator`.

Para o tipo Number, as comparações de valor são numéricas.

As comparações de valor String para "maior que", "igual a" ou "menor que" são baseadas em codificação UTF-8 binária. Por exemplo, a é maior que A, e a é maior que B.

Para o tipo Binary, o DynamoDB trata cada byte de dados binários como não assinados ao comparar valores binários.

Para obter mais informações sobre como especificar tipos de dados em JSON, consulte [API do DynamoDB de baixo nível \(p. 217\)](#).

- `ComparisonOperator`- um comparador para avaliar atributos. Por exemplo, é igual a, maior que, menor que etc.

Os seguintes operadores de comparação estão disponíveis:

`EQ | NE | LE | LT | GE | GT | NOT_NULL | NULL | CONTAINS | NOT_CONTAINS |
BEGINS_WITH | IN | BETWEEN`

Usar o `FilterExpression` INST

Suponha que você queira consultar o Música e aplicar uma condição aos itens correspondentes. Você poderia usar uma solicitação `Query` com um parâmetro `QueryFilter`, como neste exemplo da AWS CLI:

```
aws dynamodb query \  
    --table-name Music \  
    --key-conditions '{  
        "Artist": {  
            "ComparisonOperator": "EQ",  
            "AttributeValueList": [ {"S": "No One You Know"} ]  
        }'  
    }' \  
    --query-filter '{  
        "Price": {  
            "ComparisonOperator": "GT",  
            "AttributeValueList": [ {"N": "1.00"} ]  
        }'  
    }'
```

Mas você também poderia usar uma `FilterExpression` em vez disso:

```
aws dynamodb query \  
    --table-name Music \  
    --key-conditions '{  
        "Artist": {  
            "ComparisonOperator": "EQ",  
            "AttributeValueList": [ {"S": "No One You Know"} ]  
        }'  
    }' \  
    --filter-expression "Price > 1.00"
```

```
--table-name Music \
--key-condition-expression 'Artist = :a' \
--filter-expression 'Price > :p' \
--expression-attribute-values '{
    ":p": {"N":"1.00"}, 
    ":a": {"S":"No One You Know"}
}'
```

ScanFilter

Em uma operação Scan, ScanFilter é uma condição que avalia os resultados da verificação e retorna apenas os valores desejados.

Note

Este parâmetro não tem suporte a atributos do tipo List ou Map.

Se você especificar mais de uma condição na caixa ScanFilter, por padrão, todas as condições devem ser avaliadas como verdadeiras. Em outras palavras, as condições são associadas. (Você pode usar o [ConditionalOperator \(p. 1138\)](#) Para desassociar as condições. Se você fizer isso, pelo menos uma das condições deve ser avaliada como verdadeira, em vez de todas elas.)

Cada elemento ScanFilter consiste em um nome de atributo a ser comparado, junto com o seguinte:

- **AttributeValueList**- um ou mais valores para avaliar em relação ao atributo fornecido. O número de valores na lista depende do operador especificado em ComparisonOperator.

Para o tipo Number, as comparações de valor são numéricas.

As comparações de valor String para "maior que", "igual a" ou "menor que" são baseadas em codificação UTF-8 binária. Por exemplo, a é maior que A, e a é maior que B.

Para Binary, o DynamoDB trata cada byte de dados binários como não assinados ao comparar valores binários.

Para obter mais informações sobre como especificar tipos de dados em JSON, consulte [API do DynamoDB de baixo nível \(p. 217\)](#).

- **ComparisonOperator**- um comparador para avaliar atributos. Por exemplo, é igual a, maior que, menor que etc.

Os seguintes operadores de comparação estão disponíveis:

EQ | NE | LE | LT | GE | GT | NOT_NULL | NULL | CONTAINS | NOT_CONTAINS |
BEGINS_WITH | IN | BETWEEN

Usar o FilterExpression INST

Suponha que você queira verificar se Músicas e aplicar uma condição aos itens correspondentes. Você poderia usar uma solicitação Scan com um parâmetro ScanFilter, como neste exemplo da AWS CLI:

```
aws dynamodb scan \
--table-name Music \
--scan-filter '{
    "Genre": {
        "AttributeValueList": [ {"S": "Rock"} ],
        "ComparisonOperator": "EQ"
    }
}'
```

```
}
```

Mas você também poderia usar uma `FilterExpression` em vez disso:

```
aws dynamodb scan \  
  --table-name Music \  
  --filter-expression 'Genre = :g' \  
  --expression-attribute-values '{  
    ":g": {"S":"Rock"}  
}'
```

Criação de condições com parâmetros herdados

A seção a seguir descreve como criar condições a serem usadas com parâmetros herdados, como `Expected`, `QueryFilter` e `ScanFilter`.

Note

Os novos aplicativos devem usar os parâmetros de expressão em vez disso. Para mais informações, consulte [Uso de expressões no DynamoDB \(p. 414\)](#).

Condições simples

Com valores de atributo, você pode criar condições para comparações com atributos da tabela. Uma condição sempre é avaliada como verdadeira ou falsa, e consiste em:

- `ComparisonOperator` - maior que, menor que, igual a e assim por diante.
- `AttributeValueList`(opcional) – valor(es) de atributo para comparar. Dependendo do `ComparisonOperator` que está sendo usado, `AttributeValueList` pode conter um, dois ou mais valores ou pode não estar presente.

As seções a seguir descrevem os vários operadores de comparação, juntamente com exemplos de como usá-los em condições.

Operadores de comparação sem valores de atributo

- `NOT_NULL` - verdadeiro se um atributo existir.
- `NULL` - verdadeiro se um atributo não existir.

Use esses operadores para verificar se um atributo existe ou não. Como não há valor de comparação, não especifique `AttributeValueList`.

Exemplo

A expressão a seguir é avaliada como verdadeira se o atributo `Dimensions` existir.

```
...  
  "Dimensions": {  
    ComparisonOperator: "NOT_NULL"  
  }  
...
```

Operadores de comparação com um valor de atributo

- `EQ` - verdadeiro se um atributo for igual a um valor.

`AttributeValueList`pode conter apenas um valor do tipo String, Number, Binary, String Set, Number Set ou Binary Set. Se um item contém um valor de um tipo diferente do especificado na solicitação, os valores não coincidem. Por exemplo, a string "3" Não igual ao número 3. Além disso, o número 3 não é igual ao conjunto de números [3, 2, 1].

- `NE` - verdadeiro se um atributo não for igual a um valor.

`AttributeValueList`pode conter apenas um valor do tipo String, Number, Binary, String Set, Number Set ou Binary Set. Se um item contém um valor de um tipo diferente do especificado na solicitação, os valores não coincidem.

- `LE` - verdadeiro se um atributo for menor ou igual a um valor.

`AttributeValueList`pode conter apenas um valor do tipo String, Number ou Binary (não um tipo Set). Se um item contém um `AttributeValue` de um valor diferente do especificado na solicitação, os valores não coincidem.

- `LT` - verdadeiro se um atributo for menor que um valor.

`AttributeValueList`pode conter apenas um valor do tipo String, Number ou Binary (não um tipo Set). Se um item contém um valor de um tipo diferente do especificado na solicitação, os valores não coincidem.

- `GE` - verdadeiro se um atributo for maior ou igual a um valor.

`AttributeValueList`pode conter apenas um valor do tipo String, Number ou Binary (não um tipo Set). Se um item contém um valor de um tipo diferente do especificado na solicitação, os valores não coincidem.

- `GT` - verdadeiro se um atributo é maior que um valor.

`AttributeValueList`pode conter apenas um valor do tipo String, Number ou Binary (não um tipo Set). Se um item contém um valor de um tipo diferente do especificado na solicitação, os valores não coincidem.

- `CONTAINS` - verdadeiro se um valor estiver presente em um conjunto ou se um valor contiver outro.

`AttributeValueList`pode conter apenas um valor do tipo String, Number ou Binary (não um tipo Set). Se o atributo de destino da comparação for String, o operador procurará uma substring correspondente. Se o atributo de destino da comparação for Binary, o operador procurará uma subsequência do destino que corresponda à entrada. Se o atributo de destino da comparação for um conjunto, o operador será avaliado como verdadeiro caso encontre uma correspondência exata com qualquer membro do conjunto.

- `NOT_CONTAINS` - verdadeiro se um valor não estiver presente em um conjunto, ou se um valor não contiver outro valor.

`AttributeValueList`pode conter apenas um valor do tipo String, Number ou Binary (não um tipo Set). Se o atributo de destino da comparação for String, o operador verificará a ausência de uma substring correspondente. Se o atributo de destino da comparação for Binary, o operador verificará a ausência de uma subsequência do destino que corresponda à entrada. Se o atributo de destino da comparação for um conjunto, o operador será avaliado como verdadeiro se ele não encontrar uma correspondência exata com qualquer membro do conjunto.

- `BEGINS_WITH`- verdadeiro se os primeiros caracteres de um atributo corresponderem ao valor fornecido. Não use este operador para comparar números.

`AttributeValueList`pode conter apenas um valor do tipo String ou Binary (não um tipo Number ou Set). O atributo de destino da comparação deve ser String ou Binary (não Number ou um conjunto).

Use esses operadores para comparar um atributo com um valor. Você deve especificar um `AttributeValueList` consistindo em um único valor. Para a maioria dos operadores, esse valor deve ser escalar. No entanto, os operadores `EQ` e `NE` também dão suporte a conjuntos.

Exemplos

As expressões a seguir serão avaliadas como verdadeiras se:

- O preço de um produto for maior que 100.

```
...  
    "Price": {  
        ComparisonOperator: "GT",  
        AttributeValueList: [ {"N":"100"} ]  
    }  
...
```

- A categoria de um produto começar com "Bo".

```
...  
    "ProductCategory": {  
        ComparisonOperator: "BEGINS_WITH",  
        AttributeValueList: [ {"S":"Bo"} ]  
    }  
...
```

- Um produto estiver disponível em vermelho, verde ou preto:

```
...  
    "Color": {  
        ComparisonOperator: "EQ",  
        AttributeValueList: [  
            {"S":"Black"}, {"S":"Red"}, {"S":"Green"} ]  
    }  
...
```

Note

Ao comparar tipos de dados Set, a ordem dos elementos não importa. O DynamoDB retornará apenas os itens com o mesmo conjunto de valores, independentemente da ordem em que você especificá-los em sua solicitação.

Operadores de comparação com dois valores de atributo

- **BETWEEN** - verdadeiro se um valor for entre um limite inferior e um limite superior, inclusive endpoints.

AttributeValueList deve conter dois elementos do mesmo tipo, seja String, Number ou Binary (não Set). Um atributo de destino corresponde se o valor de destino for maior que, ou igual, ao primeiro elemento e menor que, ou igual, ao segundo elemento. Se um item contém um valor de um tipo diferente do especificado na solicitação, os valores não coincidem.

Use este operador para determinar se um valor de atributo está dentro de um intervalo. A AttributeValueList deve conter dois elementos escalares do mesmo tipo – String, Number ou Binary.

Exemplo

A expressão a seguir será avaliada como verdadeira, se o preço de um produto estiver entre 100 e 200.

```
...  
    "Price": {  
        ComparisonOperator: "BETWEEN",  
        AttributeValueList: [ {"N":"100"}, {"N":"200"} ]  
    }
```

```
    ...
}
```

Operadores de comparação com NValores de atributo

- **IN**- verdadeiro se um valor for igual a qualquer um dos valores em uma lista enumerada. Somente valores escalares têm suporte na lista, não conjuntos. O atributo de destino deve ser do mesmo tipo e o valor exato para corresponder.

`AttributeValueList`pode conter um ou mais elementos do tipo String, Number ou Binary (não um tipo Set). Esses atributos são comparados com um atributo do tipo que não é Set existente de um item. Se quaisquer elementos do conjunto de entradas estiverem presentes no atributo do item, a expressão será avaliada como verdadeira.

`AttributeValueList`pode conter um ou mais valores do tipo String, Number ou Binary (não um tipo Set). O atributo de destino da comparação deve ser do mesmo tipo e ter o valor exato para corresponder. Um atributo String nunca corresponde a um String set.

Use este operador para determinar se o valor fornecido está em uma lista enumerada. Você pode especificar qualquer número de valores escalares em `AttributeValueList`, mas todos eles devem ser do mesmo tipo de dados.

Exemplo

A expressão a seguir será avaliada como verdadeira se o valor para Id for 201, 203 ou 205.

```
...
  "Id": {
    ComparisonOperator: "IN",
    AttributeValueList: [ {"N":"201"}, {"N":"203"}, {"N":"205"} ]
}
...
```

Uso de várias condições

O DynamoDB permite combinar várias condições para formar expressões complexas. Isso é feito ao fornecer pelo menos duas expressões, com um [ConditionalOperator](#) (p. 1138) opcional.

Por padrão, quando você especifica mais de uma condição, otudoAs condições devem ser avaliadas como verdadeiras para que a expressão inteira também seja avaliada assim. Em outras palavras, uma operação AND implícita acontece.

Exemplo

A expressão a seguir será avaliada como verdadeira, se um produto for um livro que tem, pelo menos, 600 páginas. Ambas as condições devem ser avaliadas como verdadeiras, já que elas são implicitamente ANDed (associadas).

```
...
  "ProductCategory": {
    ComparisonOperator: "EQ",
    AttributeValueList: [ {"S":"Book"} ]
  },
  "PageCount": {
    ComparisonOperator: "GE",
    AttributeValueList: [ {"N":600} ]
  }
...
```

Você pode usar o [ConditionalOperator](#) (p. 1138) para esclarecer que uma operação terá lugar. O exemplo a seguir se comporta da mesma forma que o anterior.

```
...  
    "ConditionalOperator" : "AND",  
    "ProductCategory": {  
        "ComparisonOperator": "EQ",  
        "AttributeValueList": [ {"N":"Book"} ]  
    },  
    "PageCount": {  
        "ComparisonOperator": "GE",  
        "AttributeValueList": [ {"N":600} ]  
    }  
...
```

Você também pode definir `ConditionalOperator` como OR, o que significa que pelo menos uma das condições deve ser avaliada como verdadeira.

Exemplo

A expressão a seguir será avaliada como verdadeira, se um produto for uma mountain bike, se ele for de uma marca específica ou se o preço for maior que 100.

```
...  
    ConditionalOperator : "OR",  
    "BicycleType": {  
        "ComparisonOperator": "EQ",  
        "AttributeValueList": [ {"S":"Mountain"} ]  
    },  
    "Brand": {  
        "ComparisonOperator": "EQ",  
        "AttributeValueList": [ {"S":"Brand-Company A"} ]  
    },  
    "Price": {  
        "ComparisonOperator": "GT",  
        "AttributeValueList": [ {"N":100} ]  
    }  
...
```

Note

Em uma expressão complexa, as condições são processadas na ordem, da primeira condição para a última.

Não é possível usar AND e OR em uma única expressão.

Outros operadores condicionais

Nas versões anteriores do DynamoDB, o `Expected` se comportava de forma diferente em gravações condicionais. Cada item no `Expected` mapa representava um nome de atributo para o DynamoDB verificar, junto com o seguinte:

- `Value` - um valor para comparar com o atributo.
- `Exists` - determina se o valor existe antes de tentar a operação.

O `Value` e `Exists` continuam a ter suporte no DynamoDB; no entanto, elas só permitem que você teste uma condição de igualdade ou se um atributo existe. Recomendamos que você use `ComparisonOperator` e `AttributeValueList` em vez disso, pois essas opções permitem que você crie uma gama muito maior de condições.

Example

A `DeleteItem` pode verificar se um livro não está mais em publicação, e apenas exclui-lo se essa condição for verdadeira. Veja um exemplo da AWS CLI que usa uma condição herdada:

```
aws dynamodb delete-item \
--table-name ProductCatalog \
--key '{
    "Id": {"N":"600"}
}' \
--expected '{
    "InPublication": {
        "Exists": true,
        "Value": {"BOOL":false}
    }
}'
```

O exemplo a seguir executa a mesma ação, mas não usa uma condição herdada:

```
aws dynamodb delete-item \
--table-name ProductCatalog \
--key '{
    "Id": {"N":"600"}
}' \
--expected '{
    "InPublication": {
        "ComparisonOperator": "EQ",
        "AttributeValueList": [ {"BOOL":false} ]
    }
}'
```

Example

A `PutItem` operaçāo pode proteger contra a substituição de um item existente com os mesmos atributos da chave primária. Veja um exemplo da que usa uma condição herdada:

```
aws dynamodb put-item \
--table-name ProductCatalog \
--item '{
    "Id": {"N":"500"},
    "Title": {"S":"Book 500 Title"}
}' \
--expected '{
    "Id": { "Exists": false }
}'
```

O exemplo a seguir executa a mesma ação, mas não usa uma condição herdada:

```
aws dynamodb put-item \
--table-name ProductCatalog \
--item '{
    "Id": {"N":"500"},
    "Title": {"S":"Book 500 Title"}
}' \
--expected '{
    "Id": { "ComparisonOperator": "NULL" }
}'
```

Note

Para condições no `ExpectedMapa`, não use o legado `ValueeExistsOpções` do junto com `ComparisonOperatoreAttributeValueList`. Se você fizer isso, sua gravação condicional falhará.

Versão anterior da API de baixo nível (2011-12-05)

Esta seção documenta as operações disponíveis na versão anterior da API de baixo nível do DynamoDB (2011-12-05). Essa versão da API de baixo nível é mantida para compatibilidade com versões anteriores de aplicativos existentes.

Novos aplicativos devem usar a versão atual da API (2012-08-10). Para obter mais informações, consulte [Referência de API de baixo nível \(p. 1067\)](#).

Note

Recomendamos a migração de aplicativos para a versão mais recente da API (2012-08-10), pois não será feito o backport de novos recursos do DynamoDB para a versão anterior da API.

Tópicos

- [BatchGetItem \(p. 1152\)](#)
- [BatchWriteItem \(p. 1157\)](#)
- [CreateTable \(p. 1162\)](#)
- [DeleteItem \(p. 1167\)](#)
- [DeleteTable \(p. 1171\)](#)
- [DescribeTables \(p. 1174\)](#)
- [.GetItem \(p. 1177\)](#)
- [ListTables \(p. 1180\)](#)
- [PutItem \(p. 1182\)](#)
- [Query \(p. 1187\)](#)
- [Scan \(p. 1196\)](#)
- [UpdateItem \(p. 1208\)](#)
- [UpdateTable \(p. 1214\)](#)

BatchGetItem

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o [Referência de API do Amazon DynamoDB](#).

Description

O `BatchGetItem` define os atributos para vários itens de várias tabelas usando suas chaves primárias. O número máximo de itens que podem ser recuperados para uma única operação é 100. Além disso, o número de itens recuperados é restrito por um limite de tamanho de 1 MB. Se o limite de tamanho da resposta for excedido ou se um resultado parcial for retornado porque o throughput provisionado da tabela foi excedido, ou devido a uma falha de processamento interno, o DynamoDB retornará um `UnprocessedKeys` para que você possa tentar novamente a operação começando com o próximo item para obter. O DynamoDB ajusta automaticamente o número de itens retornados por página para impor

esse limite. Por exemplo, mesmo que você solicite o retorno de 100 itens, mas cada item tiver um tamanho de 50 KB, o sistema retornará 20 itens e um `UnprocessedKeys` para que você possa obter a próxima página de resultados. Se você desejar, o aplicativo pode incluir sua própria lógica para reunir as páginas de resultados em um único conjunto.

Se nenhum item pôde ser processado por causa de um throughput provisionado insuficiente em cada uma das tabelas envolvidas na solicitação, o DynamoDB retornará um `ProvisionedThroughputExceeded` exception.

Note

Por padrão, `BatchGetItem` realiza leituras eventualmente consistentes em cada tabela na solicitação. Você pode definir o parâmetro `ConsistentRead` como `true` para cada tabela se quiser leituras consistentes.

`BatchGetItem` busca itens em paralelo para minimizar latências de resposta.

Ao projetar seu aplicativo, tenha em mente que o DynamoDB não garante como os atributos são ordenados na resposta retornada. Inclua os valores de chaves primárias no `AttributesToGet` dos itens na sua solicitação para ajudar a analisar a resposta por item.

Se os itens solicitados não existirem, nada será retornado na resposta desses itens. Solicitações de itens inexistentes consomem as unidades mínimas de capacidade de leitura de acordo com o tipo de leitura. Para obter mais informações, consulte [Tamanhos e formatos de item do DynamoDB](#) (p. 349).

Requests

Syntax

```
// This header is abbreviated. For a sample of a complete header, see API do DynamoDB de
// baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.BatchGetItem
content-type: application/x-amz-json-1.0

{"RequestItems":
    {"Table1":
        {"Keys":
            [{"HashKeyElement": {"S": "KeyValue1"}, "RangeKeyElement": {"N": "KeyValue2"}},
             {"HashKeyElement": {"S": "KeyValue3"}, "RangeKeyElement": {"N": "KeyValue4"}},
             {"HashKeyElement": {"S": "KeyValue5"}, "RangeKeyElement": {"N": "KeyValue6"}],
            "AttributesToGet": ["AttributeName1", "AttributeName2", "AttributeName3"]},
        "Table2":
            {"Keys":
                [{"HashKeyElement": {"S": "KeyValue4"}},
                 {"HashKeyElement": {"S": "KeyValue5}}],
                "AttributesToGet": ["AttributeName4", "AttributeName5", "AttributeName6"]}
            }
    }
}
```

Nome	Descrição	Obrigatório
<code>RequestItems</code>	Um contêiner do nome da tabela e de itens correspondentes a serem obtidos por chave primária. Ao solicitar itens, cada nome de tabela pode ser invocado apenas uma vez por operação. Type: String	Sim

Nome	Descrição	Obrigatório
	Padrão: Nenhum	
Table	<p>O nome da tabela que contém os itens a serem obtidos. A entrada é simplesmente uma string especificando uma tabela existente, sem rótulo.</p> <p>Type: String</p> <p>Padrão: Nenhum</p>	Sim
Table:Keys	<p>Os valores de chave primária que definem os itens na tabela especificada. Para obter mais informações sobre chaves primárias, consulte Chave primária (p. 6).</p> <p>Type: Chaves</p>	Sim
Table:AttributesToGet	<p>Matriz de nomes de atributos dentro da tabela especificada. Se os nomes de atributos não forem especificados, todos os atributos serão retornados. Se alguns atributos não forem encontrados, eles não serão exibidos no resultado.</p> <p>Type: Array</p>	Não
Table:ConsistentRead	<p>Se definido como <code>true</code>, uma leitura consistente será emitida; caso contrário, uma leitura eventualmente consistente será usada.</p> <p>Type: Booliano</p>	Não

Responses

Syntax

```

    "AttributeName2": {"S": "AttributeValue"},  

    "AttributeName3": {"NS": ["AttributeValue", "AttributeValue", "AttributeValue"]}  

  ],  

  "ConsumedCapacityUnits":1},  

  "Table2":  

  {"Items":  

  [{"AttributeName1": {"S":"AttributeValue"},  

  "AttributeName2": {"N":"AttributeValue"},  

  "AttributeName3": {"SS":["AttributeValue", "AttributeValue", "AttributeValue"]}  

  },  

  {"AttributeName1": {"S": "AttributeValue"},  

  "AttributeName2": {"S": "AttributeValue"},  

  "AttributeName3": {"NS": ["AttributeValue", "AttributeValue","AttributeValue"]}  

  }],  

  "ConsumedCapacityUnits":1}  

},  

"UnprocessedKeys":  

 {"Table3":  

 {"Keys":  

 [{"HashKeyElement": {"S":"KeyValue1"}, "RangeKeyElement": {"N":"KeyValue2"}},  

 {"HashKeyElement": {"S":"KeyValue3"}, "RangeKeyElement": {"N":"KeyValue4"}},  

 {"HashKeyElement": {"S":"KeyValue5"}, "RangeKeyElement": {"N":"KeyValue6"}}],  

 "AttributesToGet":["AttributeName1", "AttributeName2", "AttributeName3"]}  

}
}
}

```

Nome	Descrição
Responses	Nomes de tabelas e os respectivos atributos de itens das tabelas. Type: Mapa
Table	O nome da tabela que contém os itens. A entrada é simplesmente uma string especificando a tabela, sem rótulo. Type: String
Items	Contêiner para nomes de atributos e valores que atendem aos parâmetros de operação. Type: Mapa de nomes de atributo e seus tipos de dados e valores.
ConsumedCapacityUnits	O número de unidades de capacidade de leitura consumidas, para cada tabela. Esse valor mostra o número utilizado no throughput provisionado. As solicitações de itens não existentes consomem o mínimo de unidades de capacidade de leitura, dependendo do tipo de leitura. Para obter mais informações, consulte Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB (p. 345) . Type: telefone
UnprocessedKeys	Contém uma matriz de tabelas e suas respectivas chaves que não foram processadas com a resposta atual, possivelmente devido a um limite atingido no tamanho da resposta.

Nome	Descrição
	O <code>UnprocessedKeys</code> está na mesma forma que <code>umRequestItems</code> (portanto, o valor pode ser fornecido diretamente a <code>umBatchGetItem</code> operação). Para obter mais informações, consulte o parâmetro <code>RequestItems</code> acima.
	Type: Array
<code>UnprocessedKeys: Table: Keys</code>	Os valores de atributos de chave primária que definem os itens e os atributos associados a esses itens. Para obter mais informações sobre chaves primárias, consulte Chave primária (p. 6) .
	Type: Matriz de pares de nome-valor de atributo.
<code>UnprocessedKeys: Table: AttributesToGet</code>	Nomes de atributos na tabela especificada. Se os nomes de atributos não forem especificados, todos os atributos serão retornados. Se alguns atributos não forem encontrados, eles não serão exibidos no resultado.
	Type: Matriz de nomes de atributo.
<code>UnprocessedKeys: Table: ConsistentRead</code>	Se definido como <code>true</code> , uma leitura consistente será usada para a tabela especificada; caso contrário, uma leitura eventualmente consistente será usada.
	Type: Booleano.

Erros especiais

Erro	Descrição
<code>ProvisionedThroughputExceededException</code>	O throughput provisionado máximo permitido foi excedido.

Examples

Os exemplos a seguir mostram uma solicitação e uma resposta HTTP POST usando a operação `BatchGetItem`. Para obter exemplos do usando AWS SDK, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Exemplo de solicitação

O exemplo a seguir solicita atributos de duas tabelas diferentes.

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.BatchGetItem
content-type: application/x-amz-json-1.0
content-length: 409
```

```
{"RequestItems":  
    {"comp1":  
        {"Keys":  
            [{"HashKeyElement":{"S":"Casey"}, "RangeKeyElement":{"N":"1319509152"}},  
             {"HashKeyElement":{"S":"Dave"}, "RangeKeyElement":{"N":"1319509155"}},  
             {"HashKeyElement":{"S":"Riley"}, "RangeKeyElement":{"N":"1319509158"}},  
             "AttributesToGet":["user", "status"]},  
        "comp2":  
            {"Keys":  
                [{"HashKeyElement":{"S":"Julie"}}, {"HashKeyElement":{"S":"Mingus"}]},  
                 "AttributesToGet":["user", "friends"]}  
    }  
}
```

Exemplo de resposta

O exemplo a seguir é a resposta.

```
HTTP/1.1 200 OK  
x-amzn-RequestId: GTPQVRM4VJS792J1UFJTKUBVV4KQNSO5AEMVJF66Q9ASUAAJG  
content-type: application/x-amz-json-1.0  
content-length: 373  
Date: Fri, 02 Sep 2011 23:07:39 GMT  
  
{"Responses":  
    {"comp1":  
        {"Items":  
            [{"status":{"S":"online"}, "user":{"S":"Casey"}},  
             {"status":{"S":"working"}, "user":{"S":"Riley"}},  
             {"status":{"S":"running"}, "user":{"S":"Dave"}},  
             "ConsumedCapacityUnits":1.5},  
        "comp2":  
            {"Items":  
                [{"friends":{"SS":["Elisabeth", "Peter"]}, "user":{"S":"Mingus"}},  
                 {"friends":{"SS":["Dave", "Peter"]}, "user":{"S":"Julie"}},  
                 "ConsumedCapacityUnits":1}  
            },  
        "UnprocessedKeys":{}  
    }  
}
```

BatchWriteItem

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o[Referência de API do Amazon DynamoDB](#).

Description

Essa operação permite que você insira ou exclua vários itens de várias tabelas em uma única chamada.

Para carregar um item, você pode usar[PutItem](#) para excluir um item, você pode usar[DeleteItem](#). No entanto, quando você deseja enviar ou excluir grandes quantidades de dados, por exemplo, carregar grandes quantidades de dados do Amazon EMR (Amazon EMR) ou migrar dados de outro banco de dados para o DynamoDB, [BatchWriteItem](#)ferece uma alternativa eficiente.

Se você usa linguagens como o Java, pode usar threads para carregar itens em paralelo. Isso adiciona complexidade ao seu aplicativo para manipular os threads. Outras linguagens não oferecem suporte para threads. Por exemplo, se você estiver usando o PHP, deverá carregar ou excluir itens um de cada vez.

Em ambas as situações, `BatchWriteItem` fornece uma alternativa na qual as operações de inserção e exclusão especificadas são processadas em paralelo, proporcionando a você o poder da abordagem de pools de threads sem a necessidade de introduzir complexidade no seu aplicativo.

Observe que cada inserção e exclusão individual especificada em um `BatchWriteItem` custo de operação é o mesmo em termos de unidades de capacidade consumidas. No entanto, porque `BatchWriteItem` realiza as operações especificadas em paralelo, você obtém menos latência. Operações de exclusão em itens inexistentes consomem 1 unidade de capacidade de gravação. Para obter mais informações sobre unidades de capacidade consumidas, consulte [Como trabalhar com tabelas e dados no DynamoDB \(p. 339\)](#).

Ao usar `BatchWriteItem`, observe as seguintes limitações:

- Operações máximas em uma única solicitação: é possível especificar um total de até 25 operações de inserção ou exclusão. No entanto, o tamanho total da solicitação não pode exceder 1 MB (a carga útil HTTP).
- Você pode usar o `BatchWriteItem` para inserir e excluir itens. Ela não pode ser usada para atualizar itens existentes.
- Não é uma operação atômica. Operações individuais especificadas em um `BatchWriteItem` são atômicas; no entanto, `BatchWriteItem` como um todo é uma operação de “melhor esforço”, e não uma operação atômica. Ou seja, em um `BatchWriteItem`, algumas operações podem ser bem-sucedidas e outras podem falhar. As operações que falham são retornadas em um `UnprocessedItems` na resposta. Algumas dessas falhas podem ocorrer porque você excedeu o throughput provisionado configurado para a tabela ou devido a uma falha transitória, como um erro de rede. Você pode investigar e, opcionalmente, reenviar as solicitações. Em geral, você chama `BatchWriteItem` em um loop e em cada verificação de iteração em busca de itens não processados e envia uma nova solicitação `BatchWriteItem` com esses itens não processados.
- Não retorna nenhum item—`BatchWriteItem` foi projetada para carregar grandes quantidades de dados de forma eficiente. Ela não oferece algumas das sofisticações de `PutItem` e `DeleteItem`. Por exemplo, `DeleteItem` oferece suporte ao `ReturnValues` no corpo da solicitação para solicitar o item excluído na resposta. A operação `BatchWriteItem` não retorna itens na resposta.
- Ao contrário de `PutItem` e `DeleteItem`, `BatchWriteItem` não permite que você especifique condições em solicitações de gravação individuais na operação.
- Valores de atributos não devem ser nulos, atributos do tipo String e Binário devem ter comprimentos maiores que zero, e atributos de tipo definido não devem estar vazios. Solicitações que têm valores vazios serão rejeitadas com uma `ValidationException`.

Se qualquer uma das seguintes situações: o DynamoDB rejeitará a operação de gravação em lote inteira em qualquer uma das seguintes situações:

- Se uma ou mais tabelas especificadas na solicitação `BatchWriteItem` não existir.
- Se atributos de chaves primárias especificados em um item na solicitação não corresponderem ao esquema de chave primária da tabela correspondente.
- Se você tentar realizar várias operações no mesmo item no mesmo `BatchWriteItem` solicitação. Por exemplo, não é possível inserir e excluir o mesmo item na mesma solicitação `BatchWriteItem`.
- Se o tamanho total da solicitação exceder o limite de 1 MB (a carga útil HTTP).
- Se qualquer item individual em um lote exceder o limite de tamanho de item de 64 KB.

Requests

Syntax

```
// This header is abbreviated. For a sample of a complete header, see API do DynamoDB de baixo nível.
```

```
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.BatchGetItem
content-type: application/x-amz-json-1.0

{
    "RequestItems" : RequestItems
}

RequestItems
{
    "TableName1" : [ Request, Request, ... ],
    "TableName2" : [ Request, Request, ... ],
    ...
}

Request ::=
PutRequest | DeleteRequest

PutRequest ::=
{
    "PutRequest" : {
        "Item" : {
            "Attribute-Name1" : Attribute-Value,
            "Attribute-Name2" : Attribute-Value,
            ...
        }
    }
}

DeleteRequest ::=
{
    "DeleteRequest" : {
        "Key" : PrimaryKey-Value
    }
}

PrimaryKey-Value ::= HashTypePK | HashAndRangeTypePK

HashTypePK ::=
{
    "HashKeyElement" : Attribute-Value
}

HashAndRangeTypePK
{
    "HashKeyElement" : Attribute-Value,
    "RangeKeyElement" : Attribute-Value,
}

Attribute-Value ::= String | Numeric | Binary | StringSet | NumericSet | BinarySet

Numeric ::=
{
    "N": "Number"
}

String ::=
{
    "S": "String"
}

Binary ::=
{
    "B": "Base64 encoded binary data"
}
```

```
StringSet ::=  
{  
    "SS": [ "String1", "String2", ... ]  
}  
  
NumberSet ::=  
{  
    "NS": [ "Number1", "Number2", ... ]  
}  
  
BinarySet ::=  
{  
    "BS": [ "Binary1", "Binary2", ... ]  
}
```

No corpo da solicitação, o objeto JSON descreve as operações que você deseja realizar. As operações são agrupadas por tabelas. Você pode usar `BatchWriteItem` para atualizar ou excluir vários itens em várias tabelas. Para cada solicitação de gravação específica, você deve identificar o tipo de solicitação (`PutItem`, `DeleteItem`) seguido de informações detalhadas sobre a operação.

- Para uma `PutRequest`, você fornece o item, ou seja, uma lista de atributos e seus valores.
- Para uma `DeleteRequest`, você fornece o nome e o valor da chave primária.

Responses

Syntax

Veja a seguir a sintaxe do corpo JSON retornado na resposta.

```
{  
    "Responses" : ConsumedCapacityUnitsByTable  
    "UnprocessedItems" : RequestItems  
}  
  
ConsumedCapacityUnitsByTable  
{  
    "TableName1" : { "ConsumedCapacityUnits" : NumericValue },  
    "TableName2" : { "ConsumedCapacityUnits" : NumericValue },  
    ...  
}  
  
RequestItems  
This syntax is identical to the one described in the JSON syntax in the request.
```

Erros especiais

Não há erros específicos para esta operação.

Examples

O exemplo a seguir mostra uma solicitação HTTP POST e a resposta de uma `BatchWriteItem` operação. A solicitação especifica as seguintes operações nas tabelas Reply e Thread:

- Inserir um item e excluí-lo da tabela Reply
- Inserir um item na tabela Thread

Para obter exemplos de usando o AWS SDK, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Exemplo de solicitação

```
// This header is abbreviated. For a sample of a complete header, see API do DynamoDB de
// baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.BatchGetItem
content-type: application/x-amz-json-1.0

{
    "RequestItems": {
        "Reply": [
            {
                "PutRequest": {
                    "Item": {
                        "ReplyDateTime": {
                            "S": "2012-04-03T11:04:47.034Z"
                        },
                        "Id": {
                            "S": "DynamoDB#DynamoDB Thread 5"
                        }
                    }
                }
            },
            {
                "DeleteRequest": {
                    "Key": {
                        "HashKeyElement": {
                            "S": "DynamoDB#DynamoDB Thread 4"
                        },
                        "RangeKeyElement": {
                            "S": "oops - accidental row"
                        }
                    }
                }
            }
        ],
        "Thread": [
            {
                "PutRequest": {
                    "Item": {
                        "ForumName": {
                            "S": "DynamoDB"
                        },
                        "Subject": {
                            "S": "DynamoDB Thread 5"
                        }
                    }
                }
            }
        ]
    }
}
```

Exemplo de resposta

A seguinte resposta de exemplo mostra uma operação de inserção bem-sucedida nas tabelas Thread e Reply e uma operação de exclusão com falha na tabela Reply (por motivos como a limitação causada quando você excede o throughput provisionado na tabela). Observe o seguinte na resposta JSON:

- O objeto Responses mostra que uma unidade de capacidade foi consumida em ambas as tabelas Thread e Reply como resultado da operação bem-sucedida em cada uma dessas tabelas.

- O `UnprocessedItems` mostra a operação de exclusão sem êxito no `Reply` da Tabela INTO Em seguida, você pode emitir uma nova chamada `BatchWriteItem` para solucionar essas solicitações não processadas.

```
HTTP/1.1 200 OK
x-amzn-RequestId: G8M9ANLOE5QA26AEUHJKJE0ASBVV4KQNSO5AEMVJF66Q9ASUAAJG
Content-Type: application/x-amz-json-1.0
Content-Length: 536
Date: Thu, 05 Apr 2012 18:22:09 GMT

{
    "Responses": {
        "Thread": {
            "ConsumedCapacityUnits": 1.0
        },
        "Reply": {
            "ConsumedCapacityUnits": 1.0
        }
    },
    "UnprocessedItems": {
        "Reply": [
            {
                "DeleteRequest": {
                    "Key": {
                        "HashKeyElement": {
                            "S": "DynamoDB#DynamoDB Thread 4"
                        },
                        "RangeKeyElement": {
                            "S": "oops - accidental row"
                        }
                    }
                }
            }
        ]
    }
}
```

CreateTable

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o [Referência de API do Amazon DynamoDB](#).

Description

A operação `CreateTable` adiciona uma nova tabela à sua conta.

O nome da tabela deve ser exclusivo entre aqueles associados ao AWS Conta que emite o pedido, e a AWS Região que recebe a solicitação (como `dynamodb.us-west-2.amazonaws.com`). Cada endpoint do DynamoDB é totalmente independente. Por exemplo, se você tiver duas tabelas chamadas “MyTable”, uma em `dynamodb.us-west-2.amazonaws.com` e outra em `dynamodb.us-west-1.amazonaws.com`, elas são completamente independentes e não compartilham nenhum dado.

O `CreateTable` opera desencadeia um fluxo de trabalho assíncrono para começar a criar a tabela. O DynamoDB retorna imediatamente o estado da tabela (`CREATING`) até que a tabela esteja no `ACTIVE` Estado. Quando a tabela estiver no estado `ACTIVE`, você poderá realizar as operações de plano de dados.

Use a operação [DescribeTables \(p. 1174\)](#) para verificar o status da tabela.

Requests

Syntax

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.CreateTable
content-type: application/x-amz-json-1.0

{"TableName": "Table1",
 "KeySchema":
     {"HashKeyElement": {"AttributeName": "AttributeName1", "AttributeType": "S"},
      "RangeKeyElement": {"AttributeName": "AttributeName2", "AttributeType": "N"}},
  "ProvisionedThroughput": {"ReadCapacityUnits": 5, "WriteCapacityUnits": 10}
}
```

Nome	Descrição	Obrigatório
TableName	<p>O nome da tabela a ser criada.</p> <p>Os caracteres permitidos são a-z, A-Z, 0-9, '_' (sublinhado), '-' (traço) e '.' (ponto). Os nomes podem ter de 3 a 255 caracteres.</p> <p>Type: String</p>	Sim
KeySchema	<p>A estrutura da chave primária (simples ou composta) da tabela. Um par nome-valor para oHashKeyElement é necessário e um par de nome/valor para oRangeKeyElement é opcional (necessário apenas para chaves primárias compostas). Para obter mais informações sobre chaves primárias, consulte Chave primária (p. 6).</p> <p>Nomes de elementos de chave primária podem ter entre 1 e 255 caracteres de comprimento, sem restrições de caracteres.</p> <p>Os valores possíveis para AttributeType são "S" (string), "N" (numérico) ou "B" (binário).</p> <p>Type: Mapa deHashKeyElement, ouHashKeyElement e RangeKeyElementPara uma chave primária composta.</p>	Sim
ProvisionedThroughput	<p>Novo throughput da tabela especificada,</p>	Sim

Nome	Descrição	Obrigatório
	<p>consistindo em valores para o <code>ReadCapacityUnits</code> e <code>WriteCapacityUnits</code>. Para obter mais detalhes, consulte Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB (p. 345).</p> <p>Note</p> <p>Para conhecer os valores máximos/mínimos atuais, consulte Cotas de serviço, conta e tabela no Amazon DynamoDB (p. 1056).</p> <p>Type: Array</p>	
<code>ProvisionedThroughput: ReadCapacityUnits</code>	<p>Define o número mínimo de consistentes <code>ReadCapacityUnits</code> consumidas por segundo para a tabela especificada antes que o DynamoDB balanceie a carga com outras operações.</p> <p>As operações de leitura eventualmente consistente requerem menos esforço que uma operação de leitura consistente, portanto, uma definição de 50 <code>ReadCapacityUnits</code> consistentes por segundo oferece 100 <code>ReadCapacityUnits</code> eventualmente consistentes por segundo.</p> <p>Type: telefone</p>	Sim
<code>ProvisionedThroughput: WriteCapacityUnits</code>	<p>Define o número mínimo de <code>writeCapacityUnits</code> consumidas por segundo para a tabela especificada antes que o DynamoDB balanceie a carga com outras operações.</p> <p>Type: telefone</p>	Sim

Responses

Syntax

```
HTTP/1.1 200 OK
```

```
x-amzn-RequestId: CSOC7TJPLR00OKIRLGOHVAICUFVV4KQNSO5AEMVJF66Q9ASUAAJG
content-type: application/x-amz-json-1.0
content-length: 311
Date: Tue, 12 Jul 2011 21:31:03 GMT

{"TableDescription":
  {"CreationDateTime":1.310506263362E9,
   "KeySchema":
     {"HashKeyElement":{"AttributeName":"AttributeName1","AttributeType":"S"},
      "RangeKeyElement":{"AttributeName":"AttributeName2","AttributeType":"N"}},
   "ProvisionedThroughput":{"ReadCapacityUnits":5,"WriteCapacityUnits":10},
   "TableName":"Table1",
   "TableStatus":"CREATING"
  }
}
```

Nome	Descrição
TableDescription	Um contêiner para as propriedades da tabela.
CreationDateTime	Data em que a tabela foi criada, no formato de tempo epoch UNIX . Type: telefone
KeySchema	A estrutura da chave primária (simples ou composta) da tabela. Um par nome-valor para oHashKeyElement é necessário e um par de nome/valor para oRangeKeyElement é opcional (necessário apenas para chaves primárias compostas). Para obter mais informações sobre chaves primárias, consulte Chave primária (p. 6) . Type: Mapa deHashKeyElement, ouHashKeyElementRangeKeyElementPara uma chave primária composta.
ProvisionedThroughput	O throughput da tabela especificada, consistindo em valores para oReadCapacityUnitseWriteCapacityUnits. Consulte Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB (p. 345) . Type: Array
ProvisionedThroughput :ReadCapacityUnits	O número mínimo deReadCapacityUnitsConsumidas por segundo antes que o DynamoDB balanceie a carga com outras operações Type: telefone
ProvisionedThroughput :WriteCapacityUnits	O número mínimo deReadCapacityUnitsConsumido por segundo antesWriteCapacityUnits. balanceie a carga com outras operações Type: telefone

Nome	Descrição
TableName	O nome da tabela criada. Type: String
TableStatus	O estado atual da tabela (CREATING). Quando a tabela estiver no estado ACTIVE, você poderá inserir dados nela. Use a API DescribeTables (p. 1174) para verificar o status da tabela. Type: String

Erros especiais

Erro	Descrição
ResourceInUseException	Tentativa de recriar uma tabela já existente.
LimitExceededException	O número de solicitações simultâneas de tabela (número cumulativo de tabelas no estado CREATING, DELETING ou UPDATING) excede o máximo permitido. Note Para conhecer os valores máximos/mínimos atuais, consulte Cotas de serviço, conta e tabela no Amazon DynamoDB (p. 1056).

Examples

O exemplo a seguir cria uma tabela com uma chave primária composta que contém uma string e um número. Para obter exemplos do usando AWSSDK, consulte [Como trabalhar com tabelas e dados no DynamoDB](#) (p. 339).

Exemplo de solicitação

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.CreateTable
content-type: application/x-amz-json-1.0

{
    "TableName": "comp-table",
    "KeySchema": [
        {"HashKeyElement": {"AttributeName": "user", "AttributeType": "S"}, "RangeKeyElement": {"AttributeName": "time", "AttributeType": "N"}},
        {"ProvisionedThroughput": {"ReadCapacityUnits": 5, "WriteCapacityUnits": 10}}
    ]
}
```

Exemplo de resposta

```
HTTP/1.1 200 OK
x-amzn-RequestId: CSOC7TJPLR00OKIRLGOHVAICUFVV4KQNSO5AEMVJF66Q9ASUAAJG
content-type: application/x-amz-json-1.0
content-length: 311
Date: Tue, 12 Jul 2011 21:31:03 GMT

{"TableDescription":
  {"CreationDateTime":1.310506263362E9,
  "KeySchema":
    {"HashKeyElement":{"AttributeName":"user","AttributeType":"S"},
     "RangeKeyElement":{"AttributeName":"time","AttributeType":"N"}},
  "ProvisionedThroughput":{"ReadCapacityUnits":5,"WriteCapacityUnits":10},
  "TableName":"comp-table",
  "TableStatus":"CREATING"
}
}
```

Ações relacionadas

- [DescribeTables \(p. 1174\)](#)
- [DeleteTable \(p. 1171\)](#)

DeleteItem

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o[Referência de API do Amazon DynamoDB](#).

Description

Exclui um item único em uma tabela por meio de uma chave primária. Você pode executar uma operação de exclusão condicional que exclui o item, se ele existir, ou se ele tiver um valor de atributo esperado.

Note

Se você especificar `DeleteItem` sem atributos ou valores, todos os atributos do item serão excluídos.

A não ser que você especifique as condições, `DeleteItem` é uma operação idempotente; executá-la várias vezes no mesmo item ou atributo não resultará em uma resposta de erro. As exclusões condicionais são úteis para excluir somente itens e atributos, se condições específicas forem atendidas. Se as condições forem atendidas, o DynamoDB executará a exclusão. Caso contrário, o item não é excluído.

Você pode executar a verificação condicional esperada em um atributo por operação.

Requests

Syntax

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.DeleteItem
```

```
content-type: application/x-amz-json-1.0

{
    "TableName": "Table1",
    "Key": {
        "HashKeyElement": {"S": "AttributeValue1"}, "RangeKeyElement": {"N": "AttributeValue2"}, "Expected": {"AttributeName3": {"Value": {"S": "AttributeValue3"}}, "ReturnValues": "ALL_OLD"}
    }
}
```

Nome	Descrição	Obrigatório
TableName	O nome da tabela que contém o item a ser excluído. Type: String	Sim
Key	A chave primária que define o item. Para obter mais informações sobre chaves primárias, consulte Chave primária (p. 6) . Type: Mapa de HashKeyElement ao seu valor e RangeKeyElement ao seu valor.	Sim
Expected	Designa um atributo para uma exclusão condicional. O Expected permite que você forneça o nome do atributo, e se o DynamoDB deve ou não verificar se o atributo tem um valor específico antes de excluí-lo. Type: Mapa de nomes de atributo.	Não
Expected:AttributeName	O nome do atributo da operação put condicional. Type: String	Não
Expected:AttributeName:ExpectedAttributeValue	Use esse parâmetro para especificar se o valor já existe ou não para o par de nome-valor do atributo. A notação JSON a seguir exclui o item se o atributo "Cor" não existir para esse item: <div style="border: 1px solid black; padding: 5px;"> <pre>"Expected" : {"Color":{"Exists":false}}</pre> </div> A notação JSON a seguir verifica se o atributo com o nome "Cor"	Não

Nome	Descrição	Obrigatório
	<p>tem um valor existente "Amarelo" antes de excluir o item:</p> <pre>"Expected" : {"Color":{"Exists":true}, {"Value":{"S":"Yellow"}}}</pre> <p>Por padrão, se você usar o <code>Expected</code> fornecido como parâmetro <code>Value</code>, o DynamoDB presumirá que o atributo existe e tem o valor atual a ser substituído. Portanto, você não precisa especificar <code>{"Exists":true}</code>, pois, ele é implícito. Você pode reduzir a solicitação para:</p> <pre>"Expected" : {"Color":{"Value": {"S":"Yellow"}}}</pre> <p>Note</p> <p>Se você especificar <code>{"Exists":true}</code> sem um valor de atributo para verificar, o DynamoDB retornará um erro.</p>	
<code>ReturnValues</code>	<p>Use este parâmetro se você deseja obter os pares de nome-valor de atributo antes que eles sejam excluídos. Os valores dos parâmetros possíveis são <code>NONE</code>(default) ou <code>ALL_OLD</code>. Se <code>ALL_OLD</code> for especificado, o conteúdo do item antigo será retornado. Se esse parâmetro não for fornecido ou for <code>NONE</code>, nada será retornado.</p> <p>Type: String</p>	Não

Responses

Syntax

```
HTTP/1.1 200 OK
x-amzn-RequestId: CSOC7TJPLR00OKIRLG0HVAICUFVV4KQNSO5AEMVJF66Q9ASUAAJG
content-type: application/x-amz-json-1.0
content-length: 353
Date: Tue, 12 Jul 2011 21:31:03 GMT
```

```
{
  "Attributes": [
    {"AttributeName3":{"SS":["AttributeValue3","AttributeValue4","AttributeValue5"]},
     "AttributeName2":{"S":"AttributeValue2"}, 
     "AttributeName1":{"N":"AttributeValue1"} 
   ],
  "ConsumedCapacityUnits":1
}
```

Nome	Descrição
Attributes	<p>Se o <code>ReturnValues</code> é fornecido como <code>ALL_OLD</code> na solicitação, o DynamoDB retornará uma matriz de pares de nome-valor de atributo (basicamente, o item excluído). Caso contrário, a resposta conterá um conjunto vazio.</p> <p>Type: Matriz de pares de nome-valor de atributo.</p>
ConsumedCapacityUnits	<p>O número de unidades de capacidade de gravação consumidas pela operação. Esse valor mostra o número utilizado no throughput provisionado. As solicitações de exclusão em itens não existentes consomem uma unidade de capacidade de gravação. Para obter mais informações, consulte Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB (p. 345).</p> <p>Type: telefone</p>

Erros especiais

Erro	Descrição
<code>ConditionalCheckFailedException</code>	Falha na verificação condicional. Um valor de atributo esperado não foi encontrado.

Examples

Exemplo de solicitação

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.DeleteItem
content-type: application/x-amz-json-1.0

{"TableName":"comp-table",
 "Key": {
   "HashKeyElement":{"S":"Mingus"}, "RangeKeyElement":{"N":"200"}},
 "Expected": {
   {"status":{"Value":{"S":"shopping"}}, },
 "ReturnValues":"ALL_OLD"
}
```

Exemplo de resposta

```
HTTP/1.1 200 OK
x-amzn-RequestId: U9809LI6BBFJA5N2R0TB0P017JVV4KQNSO5AEMVJF66Q9ASUAAJG
content-type: application/x-amz-json-1.0
content-length: 353
Date: Tue, 12 Jul 2011 22:31:23 GMT

{"Attributes":
  {"friends": {"SS": ["Dooley", "Ben", "Daisy"]},
   "status": {"S": "shopping"},
   "time": {"N": "200"},
   "user": {"S": "Mingus"}
 },
 "ConsumedCapacityUnits": 1
}
```

Ações relacionadas

- [PutItem \(p. 1182\)](#)

DeleteTable

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o[Referência de API do Amazon DynamoDB](#).

Description

ODeleteTableA operação exclui uma tabela e todos os seus itens. Depois deDeleteTable, a tabela especificada está noDELETINGaté que o DynamoDB conclua a exclusão. Se a tabela estiver noACTIVEEstado, você pode excluí-la. Se uma tabela estiver emCREATINGOUUPDATING, então o DynamoDB retorna umResourceInUseExceptionERROR Se a tabela especificada não existir, o DynamoDB retornará umResourceNotFoundException. Se a tabela já estiver no estado DELETING, nenhum erro será retornado.

Note

O DynamoDB pode continuar a aceitar solicitações de operação no plano de dados, como oGetItem ePutItem, em uma tabela noDELETINGaté que a exclusão da tabela esteja concluída.

As tabelas são únicas entre as associadas com oAWSConta que emite o pedido, e oAWSA região que recebe a solicitação (como dynamodb.us-west-1.amazonaws.com). Cada endpoint do DynamoDB é totalmente independente. Por exemplo, se você tiver duas tabelas chamadas "MyTable", uma em dynamodb.us-west-2.amazonaws.com e outra em dynamodb.us-west-1.amazonaws.com, elas são completamente independentes e não compartilham dados; excluir uma não exclui a outra.

Use a operação [DescribeTables \(p. 1174\)](#) para verificar o status da tabela.

Requests

Syntax

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
```

```
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.DeleteTable
content-type: application/x-amz-json-1.0

{"TableName": "Table1"}
```

Nome	Descrição	Obrigatório
TableName	O nome da tabela a ser excluída. Type: String	Sim

Responses

Syntax

```
HTTP/1.1 200 OK
x-amzn-RequestId: 4HONCKIVH1BFUDQ1U68CTG3N27VV4KQNSO5AEMVJF66Q9ASUAAJG
content-type: application/x-amz-json-1.0
content-length: 311
Date: Sun, 14 Aug 2011 22:56:22 GMT

{
    "TableDescription": {
        "CreationDateTime": 1.313362508446E9,
        "KeySchema": [
            {"HashKeyElement": {"AttributeName": "user", "AttributeType": "S"}, "RangeKeyElement": {"AttributeName": "time", "AttributeType": "N"}},
            {"ProvisionedThroughput": {"ReadCapacityUnits": 10, "WriteCapacityUnits": 10}, "TableName": "Table1", "TableStatus": "DELETING"}
        ]
    }
}
```

Nome	Descrição
TableDescription	Um contêiner para as propriedades da tabela.
CreationDateTime	Data em que a tabela foi criada. Type: telefone
KeySchema	A estrutura da chave primária (simples ou composta) da tabela. Um par nome-valor para oHashKeyElement é necessário e um par de nome/valor para oRangeKeyElement é opcional (necessário apenas para chaves primárias compostas). Para obter mais informações sobre chaves primárias, consulte Chave primária (p. 6) . Type: Mapa deHashKeyElement, ouHashKeyElementeRangeKeyElementPara uma chave primária composta.
ProvisionedThroughput	O throughput da tabela especificada, consistindo em valores para oReadCapacityUnitseWriteCapacityUnits. Consulte Gerenciamento de configurações

Nome	Descrição
	em tabelas de capacidade provisionada do DynamoDB (p. 345).
ProvisionedThroughput: ReadCapacityUnits	O número mínimo de <code>ReadCapacityUnits</code> consumidas por segundo para a tabela especificada antes que o DynamoDB balanceie a carga com outras operações. Type: telefone
ProvisionedThroughput: WriteCapacityUnits	O número mínimo de <code>WriteCapacityUnits</code> consumidas por segundo para a tabela especificada antes que o DynamoDB balanceie a carga com outras operações. Type: telefone
TableName	O nome da tabela excluída. Type: String
TableStatus	O estado atual da tabela (<code>DELETING</code>). Assim que a tabela é excluída, as solicitações subsequentes para a tabela retornam <code>resource not found</code> . Use a operação DescribeTables (p. 1174) para verificar o status da tabela. Type: String

Erros especiais

Erro	Descrição
<code>ResourceInUseException</code>	A tabela está no estado <code>CREATING</code> ou <code>UPDATING</code> e não pode ser excluída.

Examples

Exemplo de solicitação

```
// This header is abbreviated. For a sample of a complete header, see API do DynamoDB de
// baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.DeleteTable
content-type: application/x-amz-json-1.0
content-length: 40

{"TableName":"favorite-movies-table"}
```

Exemplo de resposta

```
HTTP/1.1 200 OK
```

```
x-amzn-RequestId: 4HONCKIVH1BFUDQ1U68CTG3N27VV4KQNSO5AEMVJF66Q9ASUAAJG
content-type: application/x-amz-json-1.0
content-length: 160
Date: Sun, 14 Aug 2011 17:20:03 GMT

{"TableDescription":
  {"CreationDateTime":1.313362508446E9,
   "KeySchema":
     {"HashKeyElement":{"AttributeName":"name","AttributeType":"S"}},
   "TableName":"favorite-movies-table",
   "TableStatus":"DELETING"
}
```

Ações relacionadas

- [CreateTable \(p. 1162\)](#)
- [DescribeTables \(p. 1174\)](#)

DescribeTables

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o[Referência de API do Amazon DynamoDB](#).

Description

Retorna informações sobre a tabela, incluindo seu status atual, o esquema de chave primária e quando a tabela foi criada. Os resultados de DescribeTable são eventualmente consistentes. Se você usar DescribeTable muito cedo no processo de criação de uma tabela, o DynamoDB retornará umResourceNotFoundException. Se você usar DescribeTable muito cedo no processo de atualizar uma tabela, os novos valores talvez não estejam imediatamente disponíveis.

Requests

Syntax

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.DescribeTable
content-type: application/x-amz-json-1.0

{"TableName":"Table1"}
```

Nome	Descrição	Obrigatório
TableName	O nome da tabela a ser descrita. Type: String	Sim

Responses

Syntax

```
HTTP/1.1 200
x-amzn-RequestId: 8966d095-71e9-11e0-a498-71d736f27375
content-type: application/x-amz-json-1.0
Content-Length: 543

{"Table":
  {"CreationDateTime":1.309988345372E9,
  "ItemCount":1,
  "KeySchema":
    {"HashKeyElement":{"AttributeName":"AttributeName1","AttributeType":"S"},
     "RangeKeyElement":{"AttributeName":"AttributeName2","AttributeType":"N"}},
  "ProvisionedThroughput":{"LastIncreaseDateTime": Date, "LastDecreaseDateTime": Date,
  "ReadCapacityUnits":10,"WriteCapacityUnits":10},
  "TableName":"Table1",
  "TableSizeBytes":1,
  "TableStatus":"ACTIVE"
  }
}
```

Nome	Descrição
Table	O contêiner da tabela que está sendo descrita. Type: String
CreationDateTime	Data em que a tabela foi criada, no formato de tempo epoch UNIX .
ItemCount	Número de itens na tabela especificada. O DynamoDB atualiza esse valor aproximadamente a cada seis horas. Alterações recentes podem não ser refletidas nesse valor. Type: telefone
KeySchema	A estrutura da chave primária (simples ou composta) da tabela. Um par nome-valor para o HashKeyElement é necessário e um par de nome/valor para o RangeKeyElement é opcional (necessário apenas para chaves primárias compostas). O tamanho máximo da chave de hash é 2048 bytes. O tamanho máximo da chave de intervalo é 1024 bytes. Ambos os limites são aplicados separadamente (ou seja, você pode ter uma chave combinada de hash + intervalo 2048 + 1024). Para obter mais informações sobre chaves primárias, consulte Chave primária (p. 6) .
ProvisionedThroughput	O throughput da tabela especificada, consistindo em valores para o LastIncreaseDateTime (se aplicável), LastDecreaseDateTime (se aplicável), ReadCapacityUnits e WriteCapacityUnits. Se o throughput da tabela nunca tiver sido aumentado ou diminuído, o DynamoDB não

Nome	Descrição
	retornará valores para esses elementos. Consulte Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB (p. 345). Type: Array
TableName	O nome da tabela solicitada. Type: String
TableSizeBytes	O tamanho total da tabela especificada, em bytes. O DynamoDB atualiza esse valor aproximadamente a cada seis horas. Alterações recentes podem não ser refletidas nesse valor. Type: telefone
TableStatus	O estado atual da tabela (CREATING,ACTIVE,DELETINGOUUPDATING). Quando a tabela estiver no estado ACTIVE, você poderá adicionar dados.

Erros especiais

Nenhum erro é específico dessa operação.

Examples

Os exemplos a seguir mostram uma solicitação e uma resposta HTTP POST usando a operação `DescribeTable` para uma tabela denominada "comp-table". A tabela tem uma chave primária composta.

Exemplo de solicitação

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.DescribeTable
content-type: application/x-amz-json-1.0

{"TableName": "users"}
```

Exemplo de resposta

```
HTTP/1.1 200
x-amzn-RequestId: 8966d095-71e9-11e0-a498-71d736f27375
content-type: application/x-amz-json-1.0
content-length: 543

{"Table":
  {"CreationDateTime": 1.309988345372E9,
   "ItemCount": 23,
   "KeySchema":
     {"HashKeyElement": {"AttributeName": "user", "AttributeType": "S"},
      "RangeKeyElement": {"AttributeName": "time", "AttributeType": "N"}},
   "ProvisionedThroughput": {"LastIncreaseDateTime": 1.309988345384E9,
    "ReadCapacityUnits": 10, "WriteCapacityUnits": 10},
```

```
    "TableName": "users",
    "TableSizeBytes": 949,
    "TableStatus": "ACTIVE"
}
```

Ações relacionadas

- [CreateTable](#) (p. 1162)
- [DeleteTable](#) (p. 1171)
- [ListTables](#) (p. 1180)

.GetItem

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o[Referência de API do Amazon DynamoDB](#).

Description

O `GetItem` operação retorna um conjunto de `Attributes` para obter informações sobre um item que corresponde à chave primária. Se não houver item correspondente, `GetItem` não retornará quaisquer dados.

O `GetItem` operação fornece uma leitura eventualmente consistente por padrão. Caso seu aplicativo não aceite leituras eventualmente consistentes, use o `ConsistentRead`. Embora essa operação possa demorar mais do que uma leitura padrão, ela sempre retorna o último valor atualizado. Para obter mais informações, consulte [Consistência de leituras](#) (p. 17).

Requests

Syntax

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.GetItem
content-type: application/x-amz-json-1.0

{"TableName": "Table1",
 "Key": {
     "HashKeyElement": {"S": "AttributeValue1"},
     "RangeKeyElement": {"N": "AttributeValue2"}
 },
 "AttributesToGet": ["AttributeName3", "AttributeName4"],
 "ConsistentRead": Boolean
}
```

Nome	Descrição	Obrigatório
TableName	O nome da tabela que contém o item solicitado.	Sim

Nome	Descrição	Obrigatório
	Type: String	
Key	Os valores de chave primária que definem o item. Para obter mais informações sobre chaves primárias, consulte Chave primária (p. 6) . Type: Mapa de HashKeyElement ao seu valor e RangeKeyElement ao seu valor.	Sim
AttributesToGet	Matriz de nomes de atributo. Se os nomes de atributos não forem especificados, todos os atributos serão retornados. Se alguns atributos não forem encontrados, eles não serão exibidos no resultado. Type: Array	Não
ConsistentRead	Se definido como true, uma leitura consistente será emitida; caso contrário, uma leitura eventualmente consistente será usada. Type: Booleano	Não

Responses

Syntax

```

HTTP/1.1 200
x-amzn-RequestId: 8966d095-71e9-11e0-a498-71d736f27375
content-type: application/x-amz-json-1.0
content-length: 144

{"Item": {
    "AttributeName3": {"S": "AttributeValue3"},
    "AttributeName4": {"N": "AttributeValue4"},
    "AttributeName5": {"B": "dmFsdWU="}
},
"ConsumedCapacityUnits": 0.5
}
  
```

Nome	Descrição
Item	Contém os atributos solicitados. Type: Mapa de pares de nome-valor de atributo.
ConsumedCapacityUnits	O número de unidades de capacidade de leitura consumidas pela operação. Esse valor mostra o

Nome	Descrição
	<p>número utilizado no throughput provisionado. As solicitações de itens não existentes consomem o mínimo de unidades de capacidade de leitura, dependendo do tipo de leitura. Para obter mais informações, consulte Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB (p. 345).</p> <p>Type: telefone</p>

Erros especiais

Não há erros específicos para esta operação.

Examples

Para obter exemplos do usando AWSSDK, consulte [Trabalho com itens e atributos](#) (p. 404).

Exemplo de solicitação

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.GetItem
content-type: application/x-amz-json-1.0

{"TableName":"comptable",
 "Key": {
     "HashKeyElement":{"S":"Julie"},
     "RangeKeyElement":{"N":"1307654345"}},
 "AttributesToGet":["status","friends"],
 "ConsistentRead":true
}
```

Exemplo de resposta

Observe que o valor ConsumedCapacityUnits é 1, pois o parâmetro opcional ConsistentRead é definido como true. Se ConsistentRead estiver definido como false (ou não especificado) para a mesma solicitação, a resposta é eventualmente consistente e o valor ConsumedCapacityUnits seria 0,5.

```
HTTP/1.1 200
x-amzn-RequestId: 8966d095-71e9-11e0-a498-71d736f27375
content-type: application/x-amz-json-1.0
content-length: 72

{"Item": {
    "friends": {"SS": ["Lynda, Aaron"]},
    "status": {"S": "online"}
},
"ConsumedCapacityUnits": 1
}
```

ListTables

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o[Referência de API do Amazon DynamoDB](#).

Description

Retorna uma matriz de todas as tabelas associadas à conta e ao endpoint atuais.

Cada endpoint do DynamoDB é totalmente independente. Por exemplo, se você tiver duas tabelas chamadas “MyTable”, uma em dynamodb.us-west-2.amazonaws.com e outra em dynamodb.us-east-1.amazonaws.com, elas são completamente independentes e não compartilham nenhum dado. A operação ListTables retorna todos os nomes de tabelas associados à conta que está fazendo a solicitação, para o endpoint que recebe a solicitação.

Requests

Syntax

```
// This header is abbreviated.  
// For a sample of a complete header, see API do DynamoDB de baixo nível.  
POST / HTTP/1.1  
x-amz-target: DynamoDB_20111205.ListTables  
content-type: application/x-amz-json-1.0  
  
{ "ExclusiveStartTableName": "Table1", "Limit": 3}
```

Por padrão, a operação ListTables solicita todos os nomes de tabelas associados à conta que está fazendo a solicitação, para o endpoint que recebe a solicitação.

Nome	Descrição	Obrigatório
Limit	Um número de nomes de tabela máximos a serem retornados. Type: Inteiro	Não
ExclusiveStartTableName	O nome da tabela que inicia a lista. Se você já executou uma operação ListTables e recebeu um valor LastEvaluatedTableName na resposta, use esse valor aqui para continuar a lista. Type: String	Não

Responses

Syntax

```
HTTP/1.1 200 OK
```

```
x-amzn-RequestId: S1LEK2DPQP8OJNHVHL8OU2M7KRVV4KQNSO5AEMVJF66Q9ASUAAJG
content-type: application/x-amz-json-1.0
content-length: 81
Date: Fri, 21 Oct 2011 20:35:38 GMT

{"TableNames": ["Table1", "Table2", "Table3"], "LastEvaluatedTableName": "Table3"}
```

Nome	Descrição
TableNames	Os nomes das tabelas associadas à conta atual no endpoint atual. Type: Array
LastEvaluatedTableName	O nome da última tabela na lista atual, somente se algumas tabelas da conta e do endpoint não tiverem sido retornadas. Esse valor não existirá em uma resposta se todos os nomes de tabelas já tiverem sido retornados. Use esse valor como o ExclusiveStartTableName em uma nova solicitação para continuar a lista até que todos os nomes das tabelas sejam retornados. Type: String

Erros especiais

Nenhum erro é específico dessa operação.

Examples

Os exemplos a seguir mostram uma solicitação e uma resposta HTTP POST usando a operação ListTables.

Exemplo de solicitação

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.ListTables
content-type: application/x-amz-json-1.0

{"ExclusiveStartTableName": "comp2", "Limit": 3}
```

Exemplo de resposta

```
HTTP/1.1 200 OK
x-amzn-RequestId: S1LEK2DPQP8OJNHVHL8OU2M7KRVV4KQNSO5AEMVJF66Q9ASUAAJG
content-type: application/x-amz-json-1.0
content-length: 81
Date: Fri, 21 Oct 2011 20:35:38 GMT

{"LastEvaluatedTableName": "comp5", "TableNames": ["comp3", "comp4", "comp5"]}
```

Ações relacionadas

- [DescribeTables \(p. 1174\)](#)
- [CreateTable \(p. 1162\)](#)
- [DeleteTable \(p. 1171\)](#)

PutItem

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o[Referência de API do Amazon DynamoDB](#).

Description

Cria um novo item, ou substitui um item antigo por um novo (incluindo todos os atributos). Se algum item existe em uma tabela específica com a mesma chave primária, o novo item substitui completamente o item já existente. Você pode executar uma operação Put condicional (inserir um novo item, caso não exista um com a chave primária) ou substituir um item existente se ele tiver determinados valores de atributo.

Os valores de atributo não podem ser nulos; os atributos do tipo string e binário devem ter tamanhos maiores que zero; e os atributos do tipo conjunto não devem estar vazios. As solicitações com valores vazios serão rejeitadas com `ValidationException`.

Note

Para garantir que um novo item não substitua um item existente, use uma operação put condicional com `Exists` definido como `false` para o atributo de chave primária, ou atributos.

Para obter mais informações sobre o uso de `PutItem`, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Requests

Syntax

```
// This header is abbreviated.  
// For a sample of a complete header, see API do DynamoDB de baixo nível.  
POST / HTTP/1.1  
x-amz-target: DynamoDB_20111205.PutItem  
content-type: application/x-amz-json-1.0  
  
{"TableName":"Table1",  
 "Item":{  
     "AttributeName1":{"S":"AttributeValue1"},  
     "AttributeName2":{"N":"AttributeValue2"},  
     "AttributeName5":{"B":"dmFsdWU="}  
 },  
 "Expected":{"AttributeName3":{"Value": {"S":"AttributeValue"}, "Exists":Boolean}},  
 "ReturnValues": "ReturnValuesConstant"}
```

Nome	Descrição	Obrigatório
TableName	O nome da tabela que conterá o item.	Sim

Nome	Descrição	Obrigatório
	Type: String	
Item	<p>Um mapa dos atributos do item, e deve incluir os valores de chave primária que definem o item. Outros pares de nome-valor de atributo podem ser fornecidos para o item. Para obter mais informações sobre chaves primárias, consulte Chave primária (p. 6).</p> <p>Type: Mapa de nomes de atributo para valores de atributo.</p>	Sim
Expected	<p>Designa um atributo para uma operação Put condicional. O <code>Expected</code> permite que você forneça o nome do atributo, e se o DynamoDB deve ou não verificar se o valor de atributo já existe; ou se o valor de atributo existe e tem um valor específico antes de alterá-lo.</p> <p>Type: Mapa de um nome de atributo para um valor de atributo, e se ele existe.</p>	Não
Expected:AttributeName	<p>O nome do atributo da operação put condicional.</p> <p>Type: String</p>	Não

Nome	Descrição	Obrigatório
<code>Expected:AttributeName: ExpectedAttributeValue</code>	<p>Use esse parâmetro para especificar se o valor já existe ou não para o par de nome-valor do atributo.</p> <p>A notação JSON seguinte substitui o item, se o atributo "Cor" ainda não existir para esse item:</p> <pre>"Expected" : {"Color":{"Exists":false}}</pre> <p>A notação JSON a seguir verifica se o atributo com o nome "Cor" tem um valor existente "Amarelo" antes de substituir o item:</p> <pre>"Expected" : {"Color":{"Exists":true, "Value":{"S":"Yellow"}}}</pre> <p>Por padrão, se você usar o <code>Expected</code> fornecendo um parâmetro <code>Value</code>, o DynamoDB presumirá que o atributo existe e tem o valor atual a ser substituído. Portanto, você não precisa especificar <code>{"Exists":true}</code>, pois, ele é implícito. Você pode reduzir a solicitação para:</p> <pre>"Expected" : {"Color":{"Value":{"S":"Yellow"}}}</pre> <p>Note</p> <p>Se você especificar <code>{"Exists":true}</code> sem um valor de atributo para verificar, o DynamoDB retornará um erro.</p>	Não

Nome	Descrição	Obrigatório
ReturnValues	<p>Use este parâmetro se você deseja obter os pares de nome-valor de atributo antes que eles sejam atualizados com o PutItem. Os valores dos parâmetros possíveis são <code>NONE</code>(default) ou <code>ALL_OLD</code>. Se <code>ALL_OLD</code> for especificado, o PutItemSubstitui um par de nome-valor de atributo, o conteúdo do item antigo é retornado. Se esse parâmetro não for fornecido ou for <code>NONE</code>, nada será retornado.</p> <p>Type: String</p>	Não

Responses

Syntax

A sintaxe a seguir presume que a solicitação especificou um parâmetro `ReturnValues` de `ALL_OLD`; caso contrário, a resposta terá apenas o elemento `ConsumedCapacityUnits`.

```
HTTP/1.1 200
x-amzn-RequestId: 8966d095-71e9-11e0-a498-71d736f27375
content-type: application/x-amz-json-1.0
content-length: 85

{"Attributes":
 {"AttributeName3":{"S":"AttributeValue3"},
 "AttributeName2":{"SS":"AttributeValue2"},
 "AttributeName1":{"SS":"AttributeValue1"},
 },
 "ConsumedCapacityUnits":1
}
```

Nome	Descrição
Attributes	<p>Os valores de atributo antes da operação Put, mas somente se o parâmetro <code>ReturnValues</code> for especificado <code>ALL_OLD</code> na solicitação.</p> <p>Type: Mapa de pares de nome-valor de atributo.</p>
ConsumedCapacityUnits	<p>O número de unidades de capacidade de gravação consumidas pela operação. Esse valor mostra o número utilizado no throughput provisionado. Para obter mais informações, consulte Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB (p. 345).</p> <p>Type: telefone</p>

Erros especiais

Erro	Descrição
ConditionalCheckFailedException	Falha na verificação condicional. Um valor de atributo esperado não foi encontrado.
ResourceNotFoundException	O item especificado ou o atributo não foi encontrado.

Examples

Para obter exemplos do usando AWSSDK, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Exemplo de solicitação

```
// This header is abbreviated. For a sample of a complete header, see API do DynamoDB de baixo nível.  
POST / HTTP/1.1  
x-amz-target: DynamoDB_20111205.PutItem  
content-type: application/x-amz-json-1.0  
  
{"TableName":"comp5",  
 "Item":  
     {"time":{"N":"300"},  
      "feeling":{"S":"not surprised"},  
      "user":{"S":"Riley"}  
    },  
 "Expected":  
     {"feeling":{"Value":{"S":"surprised"},"Exists":true}}  
 "ReturnValues":"ALL_OLD"  
}
```

Exemplo de resposta

```
HTTP/1.1 200  
x-amzn-RequestId: 8952fa74-71e9-11e0-a498-71d736f27375  
content-type: application/x-amz-json-1.0  
content-length: 84  
  
{"Attributes":  
     {"feeling":{"S":"surprised"},  
      "time":{"N":"300"},  
      "user":{"S":"Riley"}},  
 "ConsumedCapacityUnits":1  
}
```

Ações relacionadas

- [UpdateItem \(p. 1208\)](#)
- [DeleteItem \(p. 1167\)](#)
- [GetItem \(p. 1177\)](#)
- [BatchGetItem \(p. 1152\)](#)

Query

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o[Referência de API do Amazon DynamoDB](#).

Description

AQueryOperação obtém os valores de um ou mais itens e seus atributos por chave primária (Query só está disponível para tabelas de chave primária de hash e intervalo). É necessário fornecer umHashKeyValue, e pode restringir o escopo da consulta usando operadores de comparação noRangeKeyValueda chave primária. Use o parâmetro ScanIndexForward para obter resultados em ordem progressiva ou inversa, por chave de intervalo.

Consultas que não retornam resultados consomem as unidades de capacidade de leitura mínimas de acordo com o tipo de leitura.

Note

Se o número total de itens que atendem aos parâmetros de consulta exceder o limite de 1 MB, a consulta será interrompida, e os resultados serão retornados ao usuário com umLastEvaluatedKeypara continuar a consulta em uma operação subsequente. Ao contrário de uma operação de Verificação, uma operação de Consulta nunca retorna um conjunto de resultados vazioe aLastEvaluatedKey. A LastEvaluatedKey apenas será fornecida se o resultado exceder 1 MB, ou se você tiver usado o parâmetro Limit.

O resultado pode ser definido para uma leitura consistente usando o parâmetro ConsistentRead.

Requests

Syntax

```
// This header is abbreviated.  
// For a sample of a complete header, see API do DynamoDB de baixo nível.  
POST / HTTP/1.1  
x-amz-target: DynamoDB_20111205.Query  
content-type: application/x-amz-json-1.0  
  
{ "TableName": "Table1",  
  "Limit": 2,  
  "ConsistentRead": true,  
  "HashKeyValue": { "S": "AttributeValue1" },  
  "RangeKeyCondition": { "AttributeValueList":  
    [ { "N": "AttributeValue2" } ], "ComparisonOperator": "GT" }  
  "ScanIndexForward": true,  
  "ExclusiveStartKey": {  
    "HashKeyElement": { "S": "AttributeName1" },  
    "RangeKeyElement": { "N": "AttributeName2" }  
  },  
  "AttributesToGet": [ "AttributeName1", "AttributeName2", "AttributeName3" ]  
}
```

Nome	Descrição	Obrigatório
TableName	O nome da tabela que contém os itens solicitados.	Sim

Nome	Descrição	Obrigatório
	Type: String	
AttributesToGet	<p>Matriz de nomes de atributo. Se os nomes de atributos não forem especificados, todos os atributos serão retornados. Se alguns atributos não forem encontrados, eles não serão exibidos no resultado.</p> <p>Type: Array</p>	Não
Limit	<p>O número máximo de itens a serem retornados (não necessariamente o número de itens correspondentes). Se o DynamoDB processa o número de itens até o limite enquanto consulta a tabela, ele interromperá essa consulta e retornará os valores correspondentes até esse ponto, bem como umaLastEvaluatedKey para aplicar em uma operação subsequente para continuar a consulta. Além disso, se o conjunto de resultados exceder 1 MB antes de o DynamoDB atingir esse limite, ele interromperá a consulta e retornará os valores correspondentes, bem como umaLastEvaluatedKey para aplicar em uma operação subsequente para continuar a consulta.</p> <p>Type: telefone</p>	Não
ConsistentRead	<p>Se definido como <code>true</code>, uma leitura consistente será emitida; caso contrário, uma leitura eventualmente consistente será usada.</p> <p>Type: Booliano</p>	Não

Nome	Descrição	Obrigatório
Count	<p>Se definido como true, o DynamoDB retornará um número total de itens que correspondem aos parâmetros de consulta, em vez de uma lista dos itens correspondentes e seus atributos. É possível aplicar o parâmetro <code>Limit</code> a consultas somente de contagem.</p> <p>Não configuro-o <code>Count</code> para true enquanto fornece uma lista de <code>AttributesToGet</code>; caso contrário, o DynamoDB retornará um erro de validação. Para obter mais informações, consulte Contagem dos itens nos resultados (p. 495).</p> <p>Type: Booleano</p>	Não
HashKeyValue	<p>O valor do atributo do componente de hash da chave primária composta.</p> <p>Type: String, número ou binário</p>	Sim
RangeKeyCondition	<p>Um contêiner dos valores de atributos e operadores de comparação a serem usados para a consulta. Uma solicitação de consulta não requer uma <code>RangeKeyCondition</code>. Se você fornecer apenas <code>HashKeyValue</code>, o DynamoDB retornará todos os itens com o valor do elemento de chave de hash especificado.</p> <p>Type: Mapa</p>	Não
RangeKeyCondition: AttributeValueList	<p>Os valores de atributos a serem avaliados para os parâmetros de consulta. O <code>AttributeValueList</code> contém um valor de atributo, a menos que uma <code>BETWEEN</code> comparação é especificada. Para a comparação <code>BETWEEN</code>, o <code>AttributeValueList</code> contém dois valores de atributo.</p> <p>Type: Um mapa de <code>AttributeValue</code>s com <code>ComparisonOperator</code>.</p>	Não

Nome	Descrição	Obrigatório
<code>RangeKeyCondition: ComparisonOperator</code>	<p>Os critérios para avaliar os atributos fornecidos, como igual a, maior que etc. Veja a seguir os operadores de comparação válidos para uma operação de Consulta.</p> <p>Note</p> <p>As comparações de valor de string para maior que, igual a ou menor que são baseadas em valores de código de caracteres ASCII. Por exemplo, a é maior que A, e aa é maior que B. Para obter uma lista de valores de código, consulte http://en.wikipedia.org/wiki/ASCII#ASCII_printable_characters.</p> <p>Para Binary, o DynamoDB trata cada byte dos dados binários como não atribuído ao comparar valores binários, por exemplo, ao avaliar as expressões de consulta.</p> <p>Type: Strings ou Binário</p>	Não
	<p><code>EQ</code> : Igual.</p> <p>para o <code>EQ</code>, <code>AttributeValueList</code> pode conter apenas uma <code>AttributeValue</code> do tipo String, Number ou Binary (não um tipo Set). Se um item contém um <code>AttributeValue</code> de um valor diferente do especificado na solicitação, os valores não coincidem.</p> <p>Por exemplo, <code>{"S": "6"}</code> Não igual <code>{"N": "6"}</code>. Além disso, <code>{"N": "6"}</code> não é igual a <code>{"NS": ["6", "2", "1"]}</code>.</p>	

Nome	Descrição	Obrigatório
	<p><code>LE</code> : Menor ou igual a.</p> <p>para</p> <p><code>OLE</code>,<code>AttributeValueList</code>pode conter apenas uma<code>AttributeValue</code>do tipo String, Number ou Binary (não um tipo Set). Se um item contém um <code>AttributeValue</code> de um valor diferente do especificado na solicitação, os valores não coincidem.</p> <p>Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.</p>	
	<p><code>LT</code> : Menor que.</p> <p>para</p> <p><code>OLT</code>,<code>AttributeValueList</code>pode conter apenas uma<code>AttributeValue</code>do tipo String, Number ou Binary (não um tipo Set). Se um item contém um <code>AttributeValue</code> de um valor diferente do especificado na solicitação, os valores não coincidem.</p> <p>Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.</p>	
	<p><code>GE</code> : Maior ou igual a.</p> <p>para</p> <p><code>OGE</code>,<code>AttributeValueList</code>Ele pode conter apenas uma<code>AttributeValue</code>do tipo String, Number ou Binary (não um tipo Set). Se um item contém um <code>AttributeValue</code> de um valor diferente do especificado na solicitação, os valores não coincidem.</p> <p>Por exemplo,{ "S": "6" }Não igual{ "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.</p>	

Nome	Descrição	Obrigatório
	<p>GT : Maior que.</p> <p>para</p> <p>OBGT,AttributeValueListEle pode conter apenas umaAttributeValue do tipo String, Number ou Binary (não um tipo Set). Se um item contém um AttributeValue de um valor diferente do especificado na solicitação, os valores não coincidem. Por exemplo, { "S" : "6" } Não igual { "N" : "6" }. Além disso, { "N" : "6" } não se compara a { "NS" : ["6" , "2" , "1"] }.</p>	
	<p>BEGINS_WITH : procura um prefixo.</p> <p>para</p> <p>OBEGINS_WITH,AttributeValueListEle pode conter apenas umaAttributeValue do tipo String ou Binário (e não um Número ou conjunto). O atributo de destino da comparação deve ser String ou Binary (não Number ou um conjunto).</p>	
	<p>BETWEEN : Maior ou igual ao primeiro valor e menor que ou igual ao segundo valor.</p> <p>para</p> <p>OBETWEEN,AttributeValueList deve conter doisAttributeValueElementos do mesmo tipo, seja String, Number ou Binary (não Set). Um atributo de destino corresponde se o valor de destino for maior que, ou igual, ao primeiro elemento e menor que, ou igual, ao segundo elemento. Se um item contém um AttributeValue de um valor diferente do especificado na solicitação, os valores não coincidem. Por exemplo, { "S" : "6" } Não se compara a { "N" : "6" }. Além disso, { "N" : "6" } não se compara a { "NS" : ["6" , "2" , "1"] }.</p>	

Nome	Descrição	Obrigatório
<code>ScanIndexForward</code>	<p>Especifica o percurso crescente ou decrescente do índice. O DynamoDB retorna resultados refletindo a ordem solicitada determinada pela chave de intervalo: Se o tipo de dados for Número, os resultados serão retornados em ordem numérica; caso contrário, o percurso se baseará em valores do código de caracteres ASCII.</p> <p>Type: Booleano</p> <p>O padrão é <code>true</code> (crescente).</p>	Não
<code>ExclusiveStartKey</code>	<p>A chave primária do item a partir do qual a consulta anterior deve continuar. Uma consulta anterior pode fornecer esse valor como <code>LastEvaluatedKey</code>. Se essa operação de consulta tiver sido interrompida antes da conclusão da consulta, seja por causa do tamanho do conjunto de resultados ou do <code>Limit</code> parâmetro. A <code>LastEvaluatedKey</code> pode ser retornada em uma nova solicitação de consulta para continuar a operação a partir desse ponto.</p> <p>Tipo: <code>HashKeyElement</code> ou <code>HashKeyElement</code> e <code>RangeKeyElement</code> para uma chave primária composta.</p>	Não

Responses

Syntax

```

HTTP/1.1 200
x-amzn-RequestId: 8966d095-71e9-11e0-a498-71d736f27375
content-type: application/x-amz-json-1.0
content-length: 308

{"Count":2,"Items": [
    {"AttributeName1": {"S": "AttributeValue1"}, 
     "AttributeName2": {"N": "AttributeValue2"}, 
     "AttributeName3": {"S": "AttributeValue3"}}, 
    {"AttributeName1": {"S": "AttributeValue3"}, 
     "AttributeName2": {"N": "AttributeValue4"}, 
     "AttributeName3": {"S": "AttributeValue3"}, 
     ...
]
}

```

```

        "AttributeName5": {"B": "dmFsdWU="}
    },
    "LastEvaluatedKey": {"HashKeyElement": {"AttributeValue3": "S"},
                         "RangeKeyElement": {"AttributeValue4": "N"}}
    },
    "ConsumedCapacityUnits": 1
}

```

Nome	Descrição
Items	Atributos do item que atendem aos parâmetros de consulta. Type: Mapa de nomes de atributo e seus tipos de dados e valores.
Count	Número de itens na resposta. Para obter mais informações, consulte Contagem dos itens nos resultados (p. 495) . Type: telefone
LastEvaluatedKey	Chave primária do item no qual a operação de consulta foi interrompida, incluindo o conjunto de resultados anterior. Use esse valor para iniciar uma nova operação, excluindo o valor na nova solicitação. A <code>LastEvaluatedKey</code> é <code>null</code> quando o conjunto inteiro de resultados da consulta está completo (ou seja, a operação processou a "última página"). Tipo: <code>HashKeyElement</code> ou <code>HashKeyElement</code> e <code>RangeKeyElement</code> para uma chave primária composta.
ConsumedCapacityUnits	O número de unidades de capacidade de leitura consumidas pela operação. Esse valor mostra o número utilizado no throughput provisionado. Para obter mais informações, consulte Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB (p. 345) . Type: telefone

Erros especiais

Erro	Descrição
<code>ResourceNotFoundException</code>	A tabela especificada não foi encontrada.

Examples

Para obter exemplos do usando AWSSDK, consulte [Como trabalhar com consultas no DynamoDB \(p. 490\)](#).

Exemplo de solicitação

```
// This header is abbreviated. For a sample of a complete header, see API do DynamoDB de
// baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.Query
content-type: application/x-amz-json-1.0

{"TableName":"1-hash-rangetable",
 "Limit":2,
 "HashKeyValue":{"S":"John"},
 "ScanIndexForward":false,
 "ExclusiveStartKey": {
   "HashKeyElement":{"S":"John"},
   "RangeKeyElement":{"S":"The Matrix"}
 }
}
```

Exemplo de resposta

```
HTTP/1.1 200
x-amzn-RequestId: 3647e778-71eb-11e0-a498-71d736f27375
content-type: application/x-amz-json-1.0
content-length: 308

{"Count":2,"Items": [
  {"fans":{"SS":["Jody","Jake"]},
   "name":{"S":"John"},
   "rating":{"S":"***"},
   "title":{"S":"The End"}},
  {"fans":{"SS":["Jody","Jake"]},
   "name":{"S":"John"},
   "rating":{"S":"***"},
   "title":{"S":"The Beatles"}},
 ], "LastEvaluatedKey": {"HashKeyElement": {"S": "John"}, "RangeKeyElement": {"S": "The Beatles"}}, "ConsumedCapacityUnits": 1
}
```

Exemplo de solicitação

```
// This header is abbreviated. For a sample of a complete header, see API do DynamoDB de
// baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.Query
content-type: application/x-amz-json-1.0

{"TableName":"1-hash-rangetable",
 "Limit":2,
 "HashKeyValue":{"S":"Airplane"},
 "RangeKeyCondition": {"AttributeValueList": [{"N": "1980"}], "ComparisonOperator": "EQ"},
 "ScanIndexForward":false}
```

Exemplo de resposta

```
HTTP/1.1 200
x-amzn-RequestId: 8b9ee1ad-774c-11e0-9172-d954e38f553a
```

```
content-type: application/x-amz-json-1.0
content-length: 119

{"Count":1,"Items": [{"fans":{"SS":["Dave","Aaron"]}, "name":{"S":"Airplane"}, "rating":{"S":"***"}, "year":{"N":1980}}], "ConsumedCapacityUnits":1}
```

Ações relacionadas

- Scan (p. 1196)

Scan

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte oReferência de API do Amazon DynamoDB.

Description

OScanO retornará um ou mais itens e seus atributos, realizando uma verificação completa de uma tabela. Forneça um ScanFilter para obter resultados mais específicos.

Note

Se o número total de itens verificados exceder o limite de 1 MB, a verificação será interrompida e os resultados serão retornados para o usuário com umLastEvaluatedKeypara continuar a varredura em uma operação subsequente. Os resultados também incluem o número de itens que excedem o limite. O resultado de uma verificação pode ser que nenhum dado corresponda aos critérios de filtro.

O conjunto de resultados é eventualmente consistente.

Requests

Syntax

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.Scan
content-type: application/x-amz-json-1.0

{"TableName":"Table1",
 "Limit": 2,
 "ScanFilter": {
     "AttributeName1": {"AttributeValueList": [{"S": "AttributeValue"}], "ComparisonOperator": "EQ"}
 },
 "ExclusiveStartKey": {
     "HashKeyElement": {"S": "AttributeName1"},
```

```

        "RangeKeyElement": { "N": "AttributeName2" }
    },
    "AttributesToGet": [ "AttributeName1", "AttributeName2", "AttributeName3" ],
}

```

Nome	Descrição	Obrigatório
TableName	O nome da tabela que contém os itens solicitados. Type: String	Sim
AttributesToGet	Matriz de nomes de atributo. Se os nomes de atributos não forem especificados, todos os atributos serão retornados. Se alguns atributos não forem encontrados, eles não serão exibidos no resultado. Type: Array	Não
Limit	O número máximo de itens a serem avaliados (não necessariamente o número de itens correspondentes). Se o DynamoDB processa o número de itens até o limite ao processar os resultados, ele para e retorna os valores correspondentes até esse ponto, além de paraLastEvaluatedKeypara aplicar em uma operação subsequente para continuar recuperando itens. Além disso, se o tamanho do conjunto de dados exceder 1 MB antes de o DynamoDB atingir esse limite, a verificação será interrompida e retornará os valores correspondentes até o limite, além de umaLastEvaluatedKeypara aplicar em uma operação subsequente para continuar a varredura. Type: telefone	Não
Count	Se definido como true, o DynamoDB retornará um número total de itens para a operação Scan, mesmo se a operação não tiver itens correspondentes ao filtro designado. Você pode aplicar o parâmetro de limite somente para contar verificações.	Não

Nome	Descrição	Obrigatório
	<p>Não configuro-o <code>Count</code> para <code>true</code> enquanto fornece uma lista de <code>AttributesToGet</code>. Caso contrário, o DynamoDB retornará um erro de validação. Para obter mais informações, consulte Contagem dos itens nos resultados (p. 511).</p> <p>Type: Booleano</p>	
<code>ScanFilter</code>	<p>Avalia os resultados da verificação e retorna apenas os valores desejados. Várias condições são tratadas como operações "AND": todas as condições devem ser atendidas para serem incluídas nos resultados.</p> <p>Type: Um mapa de nomes de atributo para valores com operadores de comparação.</p>	Não
<code>ScanFilter.AttributeValueList</code>	<p>Os valores e as condições para avaliar os resultados da verificação do filtro.</p> <p>Type: Um mapa de <code>AttributeValue</code> para um <code>Condition</code>.</p>	Não

Nome	Descrição	Obrigatório
<code>ScanFilter:</code> <code>ComparisonOperator</code>	<p>Os critérios para avaliar os atributos fornecidos, como igual a, maior que etc. A seguir estão os operadores de comparação válidos para uma operação de verificação.</p> <p>Note</p> <p>As comparações de valor de string para maior que, igual a ou menor que são baseadas em valores de código de caracteres ASCII. Por exemplo, a é maior que A, e aa é maior que B. Para obter uma lista de valores de código, consulte http://en.wikipedia.org/wiki/ASCII#ASCII_printable_characters.</p> <p>Para Binary, o DynamoDB trata cada byte dos dados binários como não atribuído ao comparar valores binários, por exemplo, ao avaliar as expressões de consulta.</p> <p>Type: Strings ou Binário</p>	Não
	<p><code>EQ</code> : Igual.</p> <p>para <code>oEQ</code>, <code>AttributeValueList</code> Ele pode conter apenas uma <code>AttributeValue</code> do tipo String, Number ou Binary (não um tipo Set). Se um item contém um <code>AttributeValue</code> de um valor diferente do especificado na solicitação, os valores não coincidem.</p> <p>Por exemplo, <code>{"S": "6"}</code> Não igual <code>{"N": "6"}</code>. Além disso, <code>{"N": "6"}</code> não é igual a <code>{"NS": ["6", "2", "1"]}</code>.</p>	

Nome	Descrição	Obrigatório
	<p>NE : Não é igual.</p> <p>para</p> <p>oNE,AttributeValueListEle pode conter apenas umaAttributeValue do tipo String, Number ou Binary (não um tipo Set). Se um item contém um AttributeValue de um valor diferente do especificado na solicitação, os valores não coincidem.</p> <p>Por exemplo, { "S": "6" } Não igual { "N": "6" }. Além disso, { "N": "6" } não é igual a { "NS": ["6", "2", "1"] }.</p>	
	<p>LE : Menor ou igual a.</p> <p>para</p> <p>oLE,AttributeValueListEle pode conter apenas umaAttributeValue do tipo String, Number ou Binary (não um tipo Set). Se um item contém um AttributeValue de um valor diferente do especificado na solicitação, os valores não coincidem.</p> <p>Por exemplo, { "S": "6" } Não igual { "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.</p>	
	<p>LT : Menor que.</p> <p>para</p> <p>oLT,AttributeValueListEle pode conter apenas umaAttributeValue do tipo String, Number ou Binary (não um tipo Set). Se um item contém um AttributeValue de um valor diferente do especificado na solicitação, os valores não coincidem.</p> <p>Por exemplo, { "S": "6" } Não igual { "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.</p>	

Nome	Descrição	Obrigatório
	<p>GE : Maior ou igual a.</p> <p>para</p> <p>OGE,AttributeValueListEle pode conter apenas umaAttributeValue do tipo String, Number ou Binary (não um tipo Set). Se um item contém umAttributeValue de um valor diferente do especificado na solicitação, os valores não coincidem.</p> <p>Por exemplo, { "S": "6" } Não igual { "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.</p>	
	<p>GT : Maior que.</p> <p>para</p> <p>OGT,AttributeValueListEle pode conter apenas umaAttributeValue do tipo String, Number ou Binary (não um tipo Set). Se um item contém umAttributeValue de um valor diferente do especificado na solicitação, os valores não coincidem.</p> <p>Por exemplo, { "S": "6" } Não igual { "N": "6" }. Além disso, { "N": "6" } não se compara a { "NS": ["6", "2", "1"] }.</p>	
	NOT_NULL : o atributo existe.	
	NULL : o atributo não existe.	

Nome	Descrição	Obrigatório
	<p>CONTAINS : verifica uma subsequência ou valor em um conjunto.</p> <p>para oCONTAINS,AttributeValueListEle pode conter apenas umaAttributeValue do tipo String, Number ou Binary (não um tipo Set). Se o atributo de destino da comparação for uma String, a operação procurará uma substring correspondente. Se o atributo de destino da comparação for Binário, a operação procurará uma subsequência do destino que corresponda à entrada. Se o atributo de destino da comparação for um conjunto ("SS", "NS" ou "BS"), a operação procurará um membro do conjunto (não uma substring).</p>	
	<p>NOT_CONTAINS : verifica a ausência de uma subsequência ou a ausência de um valor em um conjunto.</p> <p>para oNOT_CONTAINS,AttributeValueListEle pode conter apenas umaAttributeValue do tipo String, Number ou Binary (não um tipo Set). Se o atributo de destino da comparação for String, a operação verificará a ausência de uma substring correspondente. Se o atributo de destino da comparação for Binário, a operação verificará a ausência de uma subsequência do destino que corresponda à entrada. Se o atributo de destino da comparação for um conjunto ("SS", "NS" ou "BS"), a operação verificará a ausência de um membro do conjunto (não uma substring).</p>	

Nome	Descrição	Obrigatório
	<p>BEGINS_WITH : procura um prefixo.</p> <p>para oBEGINS_WITH,AttributeValueListEle pode conter apenas umaAttributeValue do tipo String ou Binário (e não um Número ou conjunto). O atributo de destino da comparação deve ser String ou Binary (não Number ou um conjunto).</p>	
	<p>IN : verifica se existem correspondências exatas.</p> <p>para oIN,AttributeValueListEle pode conter mais de umAttributeValue do tipo String, Number ou Binary (não um tipo Set). O atributo de destino da comparação deve ser do mesmo tipo e ter o valor exato para corresponder. Um atributo String nunca corresponde a um String set.</p>	
	<p>BETWEEN : Maior ou igual ao primeiro valor e menor que ou igual ao segundo valor.</p> <p>para oBETWEEN,AttributeValueList deve conter doisAttributeValueElements do mesmo tipo, seja String, Number ou Binary (não Set). Um atributo de destino corresponde se o valor de destino for maior que, ou igual, ao primeiro elemento e menor que, ou igual, ao segundo elemento. Se um item contém umAttributeValue de um valor diferente do especificado na solicitação, os valores não coincidem. Por exemplo, { "S" : "6" } Não se compara a { "N" : "6" }. Além disso, { "N" : "6" } não se compara a { "NS": ["6" , "2" , "1"] }.</p>	

Nome	Descrição	Obrigatório
<code>ExclusiveStartKey</code>	<p>A chave primária do item a partir do qual a verificação anterior deve continuar. Uma verificação anterior deve fornecer esse valor, se essa operação Scan foi interrompida antes da verificação da tabela inteira, seja por causa do tamanho do conjunto de resultados ou do <code>Limit</code>parâmetro . O valor <code>LastEvaluatedKey</code> pode ser passado de volta em uma nova solicitação de verificação para continuar a operação a partir desse ponto.</p> <p>Tipo: <code>HashKeyElement</code> ou <code>HashKeyElement e RangeKeyElement</code> para uma chave primária composta.</p>	Não

Responses

Syntax

```

HTTP/1.1 200
x-amzn-RequestId: 8966d095-71e9-11e0-a498-71d736f27375
content-type: application/x-amz-json-1.0
content-length: 229

{"Count":2,"Items": [
    {"AttributeName1":{"S":"AttributeValue1"}, "AttributeName2":{"S":"AttributeValue2"}, "AttributeName3":{"S":"AttributeValue3"}}, {"AttributeName1":{"S":"AttributeValue4"}, "AttributeName2":{"S":"AttributeValue5"}, "AttributeName3":{"S":"AttributeValue6"}, "AttributeName5":{"B":"dmFsdWU="}}, {"LastEvaluatedKey": {"HashKeyElement":{"S":"AttributeName1"}, "RangeKeyElement":{"N":"AttributeName2"}}, "ConsumedCapacityUnits":1, "ScannedCount":2}
]}

```

Nome	Descrição
<code>Items</code>	<p>Contêiner dos atributos que atendem aos parâmetros da operação.</p> <p>Type: Mapa de nomes de atributo e seus tipos de dados e valores.</p>

Nome	Descrição
Count	Número de itens na resposta. Para obter mais informações, consulte Contagem dos itens nos resultados (p. 511) . Type: telefone
ScannedCount	Número de itens na verificação completa antes de todos os filtros serem aplicados. Uma alta ScannedCount Valor com poucos ou sem Count Os resultados indicam uma operação Scan ineficiente. Para obter mais informações, consulte Contagem dos itens nos resultados (p. 511) . Type: telefone
LastEvaluatedKey	Chave primária do item em que a operação Scan foi interrompida. Forneça esse valor em uma operação Scan subsequente para continuar a operação a partir desse ponto. O valor LastEvaluatedKey é null quando o conjunto de resultados da verificação inteira está completo (ou seja, a operação processou a “última página”).
ConsumedCapacityUnits	O número de unidades de capacidade de leitura consumidas pela operação. Esse valor mostra o número utilizado no throughput provisionado. Para obter mais informações, consulte Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB (p. 345) . Type: telefone

Erros especiais

Erro	Descrição
ResourceNotFoundException	A tabela especificada não foi encontrada.

Examples

Para obter exemplos do usando AWSSDK, consulte [Como trabalhar com verificações no DynamoDB \(p. 508\)](#).

Exemplo de solicitação

```
// This header is abbreviated. For a sample of a complete header, see API do DynamoDB de
// baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.Scan
content-type: application/x-amz-json-1.0
```

```
{"TableName": "1-hash-rangetable", "ScanFilter": {}}
```

Exemplo de resposta

```
HTTP/1.1 200
x-amzn-RequestId: 4e8a5fa9-71e7-11e0-a498-71d736f27375
content-type: application/x-amz-json-1.0
content-length: 465

{"Count":4,"Items": [{"date": {"S": "1980"}, "fans": {"SS": ["Dave", "Aaron"]}, "name": {"S": "Airplane"}, "rating": {"S": "****"}}, {"date": {"S": "1999"}, "fans": {"SS": ["Ziggy", "Laura", "Dean"]}, "name": {"S": "Matrix"}, "rating": {"S": "*****"}}, {"date": {"S": "1976"}, "fans": {"SS": ["Riley"]}, "name": {"S": "The Shaggy D.A."}, "rating": {"S": "***"}}, {"date": {"S": "1985"}, "fans": {"SS": ["Fox", "Lloyd"]}, "name": {"S": "Back To The Future"}, "rating": {"S": "****"}]}, "ConsumedCapacityUnits": 0.5, "ScannedCount": 4}
```

Exemplo de solicitação

```
// This header is abbreviated. For a sample of a complete header, see API do DynamoDB de
// baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.Scan
content-type: application/x-amz-json-1.0
content-length: 125

{"TableName": "comp5",
 "ScanFilter": {
    "time": {
        "AttributeValueList": [{"N": "400"}],
        "ComparisonOperator": "GT"
    }
}}
```

Exemplo de resposta

```
HTTP/1.1 200 OK
x-amzn-RequestId: PD1CQK9QCTERLTJP20VALJ60TRVV4KQNSO5AEMVJF66Q9ASUAAJG
content-type: application/x-amz-json-1.0
content-length: 262
Date: Mon, 15 Aug 2011 16:52:02 GMT

{"Count":2,
```

```
"Items": [
    {"friends":{"SS":["Dave","Ziggy","Barrie"]},
     "status":{"S":"chatting"},
     "time":{"N":"2000"},
     "user":{"S":"Casey"}},
    {"friends":{"SS":["Dave","Ziggy","Barrie"]},
     "status":{"S":"chatting"},
     "time":{"N":"2000"},
     "user":{"S":"Freddy"}}
 ],
"ConsumedCapacityUnits":0.5
"ScannedCount":4
}
```

Exemplo de solicitação

```
// This header is abbreviated. For a sample of a complete header, see API do DynamoDB de
// baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.Scan
content-type: application/x-amz-json-1.0

{"TableName":"comp5",
 "Limit":2,
 "ScanFilter":
    {"time":
        {"AttributeValueList":[{"N":"400"}],
         "ComparisonOperator":"GT"
    },
    "ExclusiveStartKey":
        {"HashKeyElement":{"S":"Freddy"}, "RangeKeyElement":{"N":"2000"}}
}
```

Exemplo de resposta

```
HTTP/1.1 200 OK
x-amzn-RequestId: PD1CQK9QCTERLTJP20VALJ60TRVV4KQNSO5AEMVJF66Q9ASUAAJG
content-type: application/x-amz-json-1.0
content-length: 232
Date: Mon, 15 Aug 2011 16:52:02 GMT

{"Count":1,
 "Items": [
    {"friends":{"SS":["Jane","James","John"]},
     "status":{"S":"exercising"},
     "time":{"N":"2200"},
     "user":{"S":"Roger"}}
 ],
"LastEvaluatedKey":{"HashKeyElement":{"S":"Riley"}, "RangeKeyElement":{"N":"250"}},
"ConsumedCapacityUnits":0.5
"ScannedCount":2
}
```

Ações relacionadas

- [Query \(p. 1187\)](#)
- [BatchGetItem \(p. 1152\)](#)

UpdateItem

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o[Referência de API do Amazon DynamoDB](#).

Description

Edita os atributos de um item existente. Você pode executar uma atualização condicional (insira um novo par de nome-valor de atributo se ele não existir, ou substitua um par de nome-valor existente se ele tiver determinados valores de atributos esperados).

Note

Não é possível atualizar os atributos da chave primária usando UpdateItem. Em vez disso, exclua o item e use PutItem para criar um novo item com novos atributos.

A operação UpdateItem inclui umAction, que define como executar a atualização. Você pode inserir, excluir ou adicionar valores de atributo.

Os valores de atributo não podem ser nulos; os atributos do tipo string e binário devem ter tamanhos maiores que zero; e os atributos do tipo conjunto não devem estar vazios. As solicitações com valores vazios serão rejeitadas com ValidationException.

Se um item existente tiver a chave primária especificada:

- COLOCARAdiciona o atributo especificado. Se o atributo existir, ele será substituído pelo novo valor.
- EXCLUIRSe nenhum valor for especificado, o atributo e seu valor serão removidos. Se um conjunto de valores for especificado, os valores no conjunto especificado serão removidos do conjunto antigo. Portanto, se o valor do atributo incluir [a, b, c] e a ação de exclusão incluir [a, c], o valor do atributo final será [b]. O tipo de valor especificado deve corresponder ao tipo de valor existente. Não é válido especificar um conjunto vazio.
- ADICIONAR—Use a ação de adicionar apenas para números ou se o atributo de destino for um conjunto (incluindo conjuntos de strings). ADD não funcionará se o atributo de destino for um único valor de string ou um valor binário escalar. O valor especificado é adicionado a um valor numérico (aumentando ou diminuindo o valor numérico existente) ou adicionado como um valor extra em um conjunto de strings. Se um conjunto de valores for especificado, os valores serão adicionados ao conjunto existente. Por exemplo, se o conjunto original for [1,2] e o valor fornecido for [3], após a operação de adição, o conjunto será [1,2,3] e não [4,5]. Ocorrerá um erro se uma ação Adicionar for especificada para um atributo de conjunto e o tipo de atributo especificado não corresponder ao tipo de conjunto existente.

Se você usar ADD para um atributo que não existe, o atributo e seus valores serão adicionados ao item.

Se nenhum item corresponder à chave primária especificada:

- COLOCARCria um novo item com a chave primária especificada. Em seguida, adiciona o atributo especificado.
- DELETE - Nada acontece.
- ADICIONAR—Cria um item com a chave primária e o número fornecidos (ou conjunto de números) para o valor de atributo. Não é válido para um tipo string ou binário.

Note

Se você usar `ADD` para aumentar ou reduzir um valor de número de um item que não existe antes da atualização, o DynamoDB usará 0 como o valor inicial. Além disso, se você atualizar um item usando `ADD` para aumentar ou reduzir um valor de número de um atributo que não existe antes da atualização (mas o item existe), o DynamoDB usará 0 como o valor inicial. Por exemplo, use `ADD` para adicionar +3 para um atributo que não existia antes da atualização. DynamoDB usa 0 para obter o valor inicial, e o valor após a atualização é 3.

Para obter mais informações sobre o uso dessa operação, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Requests

Syntax

```
// This header is abbreviated.
// For a sample of a complete header, see API do DynamoDB de baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.UpdateItem
content-type: application/x-amz-json-1.0

{"TableName": "Table1",
 "Key": {
     "HashKeyElement": {"S": "AttributeValue1"},
     "RangeKeyElement": {"N": "AttributeValue2"}},
 "AttributeUpdates": {"AttributeName3": {"Value": {"S": "AttributeValue3_New"}, "Action": "PUT"}},
     "Expected": {"AttributeName3": {"Value": {"S": "AttributeValue3_Current"}}, "ReturnValues": "ReturnValuesConstant"}
}
```

Nome	Descrição	Obrigatório
TableName	O nome da tabela que contém o item a ser atualizado. Type: String	Sim
Key	A chave primária que define o item. Para obter mais informações sobre chaves primárias, consulte Chave primária (p. 6) . Type: Mapa de HashKeyElement ao seu valor e RangeKeyElement ao seu valor.	Sim
AttributeUpdates	Mapa de nome de atributo para o novo valor e ação para a atualização. Os nomes de atributo especificam os atributos a serem modificados, e não podem conter quaisquer atributos da chave primária.	

Nome	Descrição	Obrigatório
	Type: Mapa de nome de atributo, valor e uma ação para a atualização do atributo.	
AttributeUpdates:Action	<p>Especifica como executar a atualização. Possíveis valores:<code>PUT</code>(padrão),<code>ADD</code> ou <code>DELETE</code>. As semânticas são explicadas na descrição de <code>UpdateItem</code>.</p> <p>Type: String</p> <p>Padrão: <code>PUT</code></p>	Não
Expected	<p>Designa um atributo para uma atualização condicional. O <code>Expected</code> permite que você forneça o nome do atributo, e se o DynamoDB deve ou não verificar se o valor de atributo já existe; ou se o valor de atributo existe e tem um valor específico antes de alterá-lo.</p> <p>Type: Mapa de nomes de atributo.</p>	Não
Expected:AttributeName	<p>O nome do atributo da operação put condicional.</p> <p>Type: String</p>	Não

Nome	Descrição	Obrigatório
<code>Expected:AttributeName: ExpectedAttributeValue</code>	<p>Use esse parâmetro para especificar se o valor já existe ou não para o par de nome-valor do atributo.</p> <p>A notação JSON seguinte atualiza o item, se o atributo "Cor" ainda não existir para esse item:</p> <pre>"Expected" : {"Color":{"Exists":false}}</pre> <p>A notação JSON seguinte verifica se o atributo com o nome "Cor" tem um valor existente "Amarelo" antes de atualizar o item:</p> <pre>"Expected" : {"Color":{"Exists":true}, {"Value":{"S":"Yellow"}}}</pre> <p>Por padrão, se você usar o <code>Expected</code> fornecendo um parâmetro <code>Value</code>, o DynamoDB presumirá que o atributo existe e tem o valor atual a ser substituído. Portanto, você não precisa especificar <code>{"Exists":true}</code>, pois, ele é implícito. Você pode reduzir a solicitação para:</p> <pre>"Expected" : {"Color":{"Value":{"S":"Yellow"}}}</pre> <p>Note</p> <p>Se você especificar <code>{"Exists":true}</code> sem um valor de atributo para verificar, o DynamoDB retornará um erro.</p>	Não

Nome	Descrição	Obrigatório
ReturnValues	<p>Use este parâmetro se você deseja obter os pares de nome-valor de atributo antes que eles sejam atualizados com a solicitação de UpdateItem. Os valores dos parâmetros possíveis são NONE (default) ou ALL_OLD, UPDATED_OLD, ALL_NEW ou UPDATED_NEW. Se ALL_OLD for especificado, eUpdateItemSubstitui um par de nome-valor de atributo, o conteúdo do item antigo será retornado. Se esse parâmetro não for fornecido ou for NONE, nada será retornado. Se ALL_NEW for especificado, todos os atributos da nova versão do item serão retornados. Se UPDATED_NEW for especificado, somente as novas versões dos atributos atualizados serão retornadas.</p> <p>Type: String</p>	Não

Responses

Syntax

A sintaxe a seguir presume que a solicitação especificou um parâmetro `ReturnValues` de `ALL_OLD`; caso contrário, a resposta terá apenas o elemento `ConsumedCapacityUnits`.

```

HTTP/1.1 200
x-amzn-RequestId: 8966d095-71e9-11e0-a498-71d736f27375
content-type: application/x-amz-json-1.0
content-length: 140

{
    "Attributes": {
        "AttributeName1": {"S": "AttributeValue1"},
        "AttributeName2": {"S": "AttributeValue2"},
        "AttributeName3": {"S": "AttributeValue3"},
        "AttributeName5": {"B": "dmFsdWU="}
    },
    "ConsumedCapacityUnits": 1
}

```

Nome	Descrição
<code>Attributes</code>	<p>Um mapa de pares de nome-valor de atributo, mas somente se o parâmetro <code>ReturnValues</code> for especificado como algo diferente de <code>NONE</code> na solicitação.</p> <p>Type: Mapa de pares de nome-valor de atributo.</p>

Nome	Descrição
ConsumedCapacityUnits	O número de unidades de capacidade de gravação consumidas pela operação. Esse valor mostra o número utilizado no throughput provisionado. Para obter mais informações, consulte Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB (p. 345) . Type: telefone

Erros especiais

Erro	Descrição
ConditionalCheckFailedException	Falha na verificação condicional. Atributo ("+ name +"), o valor é ("+ value +") mas o esperado era ("+ expValue +")
ResourceNotFoundExceptions	O item especificado ou o atributo não foi encontrado.

Examples

Para obter exemplos do usando AWSSDK, consulte [Trabalho com itens e atributos \(p. 404\)](#).

Exemplo de solicitação

```
// This header is abbreviated. For a sample of a complete header, see API do DynamoDB de
// baixo nível.
POST / HTTP/1.1
x-amz-target: DynamoDB_20111205.UpdateItem
content-type: application/x-amz-json-1.0

{"TableName": "comp5",
 "Key": {
     "HashKeyElement": {"S": "Julie"}, "RangeKeyElement": {"N": "1307654350"}},
 "AttributeUpdates": {
     "status": {"Value": {"S": "online"}, "Action": "PUT"}, "Expected": {"status": {"Value": {"S": "offline"}}}, "ReturnValues": "ALL_NEW"
 }
```

Exemplo de resposta

```
HTTP/1.1 200 OK
x-amzn-RequestId: 5IMHO7F01Q9P7Q6QMCKMMI3R3QRVV4KQNS05AEMVJF66Q9ASUAAJG
content-type: application/x-amz-json-1.0
content-length: 121
Date: Fri, 26 Aug 2011 21:05:00 GMT

{"Attributes": {
    "friends": {"SS": ["Lynda, Aaron"]}, "status": {"S": "online"}, "time": {"N": "1307654350"},
```

```
    "user":{"S":"Julie"}},  
    "ConsumedCapacityUnits":1  
}
```

Ações relacionadas

- [PutItem \(p. 1182\)](#)
- [DeleteItem \(p. 1167\)](#)

UpdateTable

Important

Esta seção refere-se à versão de API 2011-12-05, que está obsoleta e não deve ser usada para novos aplicativos.

Para obter a documentação sobre a API de nível inferior atual, consulte o[Referência de API do Amazon DynamoDB](#).

Description

Atualiza os valores de throughput provisionado da tabela especificada. Definir o throughput de uma tabela ajuda você a gerenciar o desempenho e é parte do recurso de throughput provisionado do DynamoDB. Para obter mais informações, consulte [Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB \(p. 345\)](#).

Os valores de throughput provisionado podem sofrer upgrade ou downgrade com base nos máximos e mínimos listados em [Cotas de serviço, conta e tabela no Amazon DynamoDB \(p. 1056\)](#).

A tabela deve estar na `ACTIVE`Para que essa operação seja bem-sucedida. `UpdateTable` é uma operação assíncrona, durante a execução da operação, a tabela está na `UPDATING`Estado Enquanto a tabela estiver no `UPDATING`Estado, a tabela ainda tem o throughput provisionado de antes da chamada. A nova configuração de throughput provisionado entra em vigor somente quando a tabela retorna para o estado `ACTIVE` após a operação `UpdateTable`.

Requests

Syntax

```
// This header is abbreviated.  
// For a sample of a complete header, see API do DynamoDB de baixo nível.  
POST / HTTP/1.1  
x-amz-target: DynamoDB_20111205.UpdateTable  
content-type: application/x-amz-json-1.0  
  
{ "TableName": "Table1",  
    "ProvisionedThroughput": { "ReadCapacityUnits": 5, "WriteCapacityUnits": 15 }  
}
```

Nome	Descrição	Obrigatório
TableName	O nome da tabela a ser atualizada. Type: String	Sim
ProvisionedThroughput	Novo throughput da tabela especificada,	Sim

Nome	Descrição	Obrigatório
	<p>consistindo em valores para o <code>ReadCapacityUnits</code> e <code>WriteCapacityUnits</code>. Consulte Gerenciamento de configurações em tabelas de capacidade provisionada do DynamoDB (p. 345).</p> <p>Type: Array</p>	
<code>ProvisionedThroughput</code> <code>:ReadCapacityUnits</code>	<p>Define o número mínimo de <code>ReadCapacityUnits</code> consumidas por segundo para a tabela especificada antes que o DynamoDB balanceie a carga com outras operações.</p> <p>As operações de leitura eventualmente consistente requerem menos esforço que uma operação de leitura consistente, portanto, uma definição de 50 <code>ReadCapacityUnits</code> consistentes por segundo oferece 100 <code>ReadCapacityUnits</code> eventualmente consistentes por segundo.</p> <p>Type: telefone</p>	Sim
<code>ProvisionedThroughput</code> <code>:WriteCapacityUnits</code>	<p>Define o número mínimo de <code>WriteCapacityUnits</code> consumidas por segundo para a tabela especificada antes que o DynamoDB balanceie a carga com outras operações.</p> <p>Type: telefone</p>	Sim

Responses

Syntax

```

HTTP/1.1 200 OK
x-amzn-RequestId: CSOC7TJPLR00OKIRLGOHVAICUFVV4KQNSO5AEMVJF66Q9ASUAAJG
Content-Type: application/json
Content-Length: 311
Date: Tue, 12 Jul 2011 21:31:03 GMT

{
    "TableDescription": {
        "CreationDateTime": 1.321657838135E9,
        "KeySchema": [
            {"HashKeyElement": {"AttributeName": "AttributeValue1", "AttributeType": "S"}, "RangeKeyElement": {"AttributeName": "AttributeValue2", "AttributeType": "N"}},
        "ProvisionedThroughput": {
    
```

```
{
    "LastDecreaseDateTime":1.321661704489E9,
    "LastIncreaseDateTime":1.321663607695E9,
    "ReadCapacityUnits":5,
    "WriteCapacityUnits":10},
    "TableName":"Table1",
    "TableStatus":"UPDATING"}}
```

Nome	Descrição
<code>CreationDateTime</code>	Data em que a tabela foi criada. Type: telefone
<code>KeySchema</code>	A estrutura da chave primária (simples ou composta) da tabela. Um par nome-valor para o <code>HashKeyElement</code> necessário e um par de nome/valor para o <code>RangeKeyElement</code> opcional (necessário apenas para chaves primárias compostas). O tamanho máximo da chave de hash é 2048 bytes. O tamanho máximo da chave de intervalo é 1024 bytes. Ambos os limites são aplicados separadamente (ou seja, você pode ter uma chave combinada de hash + intervalo 2048 + 1024). Para obter mais informações sobre chaves primárias, consulte Chave primária (p. 6) . Type: Mapa de <code>HashKeyElement</code> , <code>ouHashKeyElementRangeKeyElement</code> para uma chave primária composta.
<code>ProvisionedThroughput</code>	Configurações de throughput atuais da tabela especificada, incluindo valores para <code>LastIncreaseDateTime</code> (se aplicável), <code>LastDecreaseDateTime</code> (se aplicável), Type: Array
<code>TableName</code>	O nome da tabela atualizada. Type: String
<code>TableStatus</code>	O estado atual da tabela (<code>CREATING</code> , <code>ACTIVE</code> , <code>DELETING</code> ou <code>UPDATING</code>), que deve ser <code>UPDATING</code> . Use a operação DescribeTables (p. 1174) para verificar o status da tabela. Type: String

Erros especiais

Erro	Descrição
<code>ResourceNotFoundException</code>	A tabela especificada não foi encontrada.
<code>ResourceInUseException</code>	A tabela não está no estado <code>ACTIVE</code> .

Examples

Exemplo de solicitação

```
// This header is abbreviated.  
// For a sample of a complete header, see API do DynamoDB de baixo nível.  
POST / HTTP/1.1  
x-amz-target: DynamoDB_20111205.UpdateTable  
content-type: application/x-amz-json-1.0  
  
{"TableName":"comp1",  
 "ProvisionedThroughput":{"ReadCapacityUnits":5,"WriteCapacityUnits":15}  
}
```

Exemplo de resposta

```
HTTP/1.1 200 OK  
content-type: application/x-amz-json-1.0  
content-length: 390  
Date: Sat, 19 Nov 2011 00:46:47 GMT  
  
{"TableDescription":  
    {"CreationDateTime":1.321657838135E9,  
     "KeySchema":  
         {"HashKeyElement": {"AttributeName": "user", "AttributeType": "S"},  
          "RangeKeyElement": {"AttributeName": "time", "AttributeType": "N"}},  
     "ProvisionedThroughput":  
         {"LastDecreaseDateTime":1.321661704489E9,  
          "LastIncreaseDateTime":1.321663607695E9,  
          "ReadCapacityUnits":5,  
          "WriteCapacityUnits":10},  
     "TableName": "comp1",  
     "TableStatus": "UPDATING"}  
}
```

Ações relacionadas

- [CreateTable \(p. 1162\)](#)
- [DescribeTables \(p. 1174\)](#)
- [DeleteTable \(p. 1171\)](#)

Histórico de documentos do DynamoDB

A tabela a seguir descreve alterações importantes em cada versão do recurso deGuia do desenvolvedor do DynamoDBa partir de 3 de julho de 2018, em diante. Para receber notificações sobre atualizações nessa documentação, assine o RSS feed (no canto superior esquerdo da página).

update-history-change	update-history-description	update-history-date
Instâncias T3 do Amazon EC2 para DAX (p. 1218)	O DAX agora oferece suporte ao Tipos de instância do Amazon EC2 T3 , que fornece um nível de linha de base de desempenho da CPU com a capacidade de aumentar acima da linha de base quando necessário.	15 de fevereiro de 2021
Suporte para NoSQL Workbench para Amazon DynamoDB para PartiQL (p. 1218)	Agora, você pode usar o NoSQL Workbench para DynamoDB Para criar PartiQL para DynamoDB.	4 de dezembro de 2020
PartiQL para DynamoDB (p. 1218)	Agora você pode usar PartiQL para DynamoDB — uma linguagem de consulta compatível com SQL — para interagir com tabelas do DynamoDB e executar consultas ad hoc usando oAWS Management Console,AWS Command Line Interfacee as APIs do DynamoDB para o PartiQL.	23 de novembro de 2020
Amazon Kinesis Data Streams para Amazon DynamoDB (p. 1218)	Agora você pode usar Amazon Kinesis Data Streams para Amazon DynamoDB com suas tabelas do DynamoDB para capturar alterações no nível do item e replicá-las em um fluxo de dados do Kinesis.	23 de novembro de 2020
Exportação da tabela do DynamoDB (p. 1218)	Agora, você pode. Exportar tabelas do DynamoDB para o Amazon S3 , permitindo que você realize análises e consultas complexas em seus dados com serviços como o Athena,AWS Glue, e a Lake Formation.	9 de novembro de 2020
Support para valores vazios (p. 1218)	O DynamoDB agora oferece suporte a valores vazios para atributos binários e string não chave em tabelas do DynamoDB.	18 de maio de 2020

	O suporte a valores vazios oferece maior flexibilidade para usar atributos para um conjunto mais amplo de casos de uso sem precisar transformar esses atributos antes de enviá-los para o DynamoDB. Tipos de dados List, Map e Set também suportam valores de String e Binários vazios.	
Suporte para NoSQL Workbench para Amazon DynamoDB para Linux (p. 1218)	NoSQL Workbench para Amazon DynamoDB agora é compatível com o Linux- Ubuntu, Fedora e Debian.	4 de maio de 2020
Noções gerais sobre o CloudWatch Contributor Insights para DynamoDB (p. 1218)	Faturamento do CloudWatch Contributor Insights para DynamoDB disponível para o público geral. O CloudWatch Contributor Insights for DynamoDB é uma ferramenta de diagnóstico que fornece uma visão rápida das tendências de tráfego da tabela do DynamoDB e ajuda a identificar as chaves acessadas com mais frequência da tabela (também conhecidas como teclas de atalho)	2 de abril de 2020
Atualizando tabelas globais do (p. 1218)	Agora você pode atualizar suas tabelas globais da versão 2017.11.29 para oversão mais recente das tabelas globais (2019.11.21) , com alguns cliques no Console do DynamoDB. Ao atualizar a versão de suas tabelas globais, você pode aumentar a disponibilidade de suas tabelas do DynamoDB facilmente estendendo suas tabelas existentes para AWS Regiões, sem reconstruções de tabela necessárias.	16 de março de 2020
NoSQL Workbench para Amazon DynamoDB — GA GA (p. 1218)	NoSQL Workbench para Amazon DynamoDB disponível para o público geral. Use o NoSQL Workbench para projetar, criar, consultar e gerenciar tabelas do DynamoDB.	2 de março de 2020
Métricas do cluster de cache DAX (p. 1218)	Suporte ao DAX para o novo Métricas do CloudWatch , que permitem que você entenda melhor o desempenho do cluster DAX.	6 de fevereiro de 2020

CloudWatch Contributor Insights para DynamoDB — visualização prévia (p. 1218)	Faturamento do CloudWatch Contributor Insights para DynamoDBé uma ferramenta de diagnóstico que fornece uma visão rápida das tendências de tráfego da tabela do DynamoDB e ajuda a identificar as chaves acessadas com mais frequência da tabela (também conhecidas como teclas de atalho)	26 de novembro de 2019
Suporte de capacidade adaptável para carga de trabalho desequilibrada (p. 1218)	Capacidade adaptativa do Amazon DynamoDB agoraAlças cargas de trabalho desbalanceadas melhor isolando automaticamente os itens acessados com freqüência. Se o aplicativo direcionar tráfego desproporcionalmente alto para um ou mais itens, o DynamoDB reequilibrará as partições de modo que os itens acessados com freqüência não residam na mesma partição.	26 de novembro de 2019
Support para CMK gerenciada pelo cliente (p. 1218)	Agora, o DynamoDBO oferece suporte para CMKs gerenciados, o que significa que você pode ter controle total sobre como criptografar e gerenciar a segurança de seus dados do DynamoDB.	25 de novembro de 2019
Suporte ao NoSQL Workbench para DynamoDB Local (versão disponível para download) (p. 1218)	Agora o NoSQL Workbench oferece suporte à conexão ao DynamoDB Local (versão disponível para download) para desenvolver, criar, consultar e gerenciar tabelas do DynamoDB.	8 de novembro de 2019
NoSQL Workbench - Pré-visualização (p. 1218)	Essa é a versão inicial do NoSQL Workbench for DynamoDB. Use o NoSQL Workbench para projetar, criar, consultar e gerenciar tabelas do DynamoDB. Para obter mais informações, consulte NoSQL Workbench para Amazon DynamoDB (demonstração) .	16 de setembro de 2019
O DAX adiciona suporte para operações transacionais usando Python e .NET. (p. 1218)	O DAX é compatível com oTransactWriteItemsetTransactGetItemsAs APIs para aplicativos escritos em Go, Java, .NET, Node.js e Python. Para obter mais informações, consulte Aceleração na memória com DAX .	14 de fevereiro de 2019

Atualizações do Amazon DynamoDB local (versão disponível para download) (p. 1218)	O DynamoDB local (versão disponível para download) agora oferece suporte a APIs transacionais, capacidade de leitura/gravação sob demanda, relatórios de capacidade para operações de leitura e gravação e 20 índices secundários globais. Para obter mais informações, consulte Diferenças entre o DynamoDB disponível para download e serviço da Web do DynamoDB .	4 de fevereiro de 2019
Amazon DynamoDB sob demanda (p. 1218)	DynamoDB sob demanda é uma opção de faturamento flexível capaz de servir centenas de solicitações por segundo sem planejamento de capacidade. DynamoDB sob demanda oferece definição de preço "pague por solicitação" para solicitações de leitura e gravação, para que você pague apenas pelo o que usar. Para obter mais informações, consulte Modo de capacidade de Leitura/Gravação .	28 de novembro de 2018
Transações do Amazon DynamoDB (p. 1218)	As transações do DynamoDB realizam alterações de tudo ou nada em vários itens dentro e entre tabelas, fornecendo atomicidade, consistências, isolamento e durabilidade (ACID) no DynamoDB. Para obter mais informações, consulte Transações do Amazon DynamoDB .	27 de novembro de 2018
Amazon DynamoDB criptografa todos os dados ociosos do cliente (p. 1218)	A criptografia de DynamoDB em repouso fornece uma camada adicional de proteção de dados mantendo seus dados seguros na tabela criptografada, incluindo sua chave principal, índices secundários local e global, fluxos, tabelas globais, backups e clusters de DAX sempre que dados forem armazenados em mídia durável. Para obter mais informações, consulte Criptografia do Amazon DynamoDB em repouso .	15 de novembro de 2018

Use o Amazon DynamoDB local com mais facilidade com a nova imagem do Docker (p. 1218)	Agora, com a nova imagem do Docker do DynamoDB Local, ficou mais fácil usar o DynamoDB Local, a versão para download do DynamoDB, para ajudar a desenvolver e testar aplicativos do DynamoDB. Para obter mais informações, consulte DynamoDB (Versão baixável) e Docker.	22 de agosto de 2018
O DynamoDB Accelerator (DAX) adiciona Support à criptografia em repouso (p. 1218)	O DynamoDB Accelerator (DAX) agora oferece suporte a criptografia em repouso para novos clusters DAX para ajudar a acelerar leituras das tabelas do Amazon DynamoDB em aplicativos com segurança frágil que estão sujeitos a conformidade e requisitos regulatórios rigorosos. Para obter mais informações, consulte Criptografia de DAX em repouso .	9 de agosto de 2018
A recuperação point-in-time (PITR) do DynamoDB adicionar suporte à restauração de tabelas excluídas (p. 1218)	Se você excluir uma tabela com a recuperação point-in-time habilitada, um backup do sistema será criado automaticamente e será mantido por 35 dias (sem custo adicional). Para obter mais informações, consulte Antes de começar a usar a recuperação point-in-time .	7 de agosto de 2018
Atualizações agora disponíveis em RSS (p. 1218)	Agora, é possível assinar o feed RSS (no canto superior esquerdo desta página) para receber notificações sobre atualizações do Guia do desenvolvedor do Amazon DynamoDB.	3 de julho de 2018

Atualizações anteriores

A tabela a seguir descreve mudanças importantes no Guia do desenvolvedor do DynamoDB Antes de 3 de julho de 2018.

Alteração	Descrição	Alterado em
Suporte à linguagem Go para o DAX	Agora, você pode habilitar o desempenho de leitura de microsegundo para as tabelas do Amazon DynamoDB em seus aplicativos gravados na linguagem de programação Go usando o novo SDK do	26 de junho de 2018

Alteração	Descrição	Alterado em
	DynamoDB Accelerator (DAX) for Go. Para mais informações, consulte SDK do DAX para Go (p. 742) .	
DynamoDB anuncia o SLA	O DynamoDB lançou um Acordo de Nível de Serviço (SLA) de disponibilidade pública. Para obter mais informações, consulte o Acordo de nível de serviço do Amazon DynamoDB .	19 de junho de 2018
Backups contínuos do DynamoDB e recuperação point-in-time (PITR)	A recuperação point-in-time ajuda a proteger as tabelas do Amazon DynamoDB de operações acidentais de gravação ou exclusão. Com a recuperação point-in-time, você não precisa se preocupar com a criação, a manutenção ou a programação de backups sob demanda. Por exemplo, suponhamos que um script de teste seja gravado acidentalmente em uma tabela do DynamoDB de produção. Com a recuperação point-in-time, você pode recuperar a tabela para qualquer ponto durante os últimos 35 dias. O DynamoDB mantém backups incrementais da tabela. Para mais informações, consulte Recuperação point-in-time do DynamoDB (p. 706) .	25 de abril de 2018
Criptografia em repouso para DynamoDB	A criptografia em repouso do DynamoDB, disponível para novas tabelas do DynamoDB, ajuda você proteger os dados de seu aplicativo em tabelas do Amazon DynamoDB usando AWS Chaves de criptografia gerenciadas pelo AWS Key Management Service. Para mais informações, consulte Criptografia do DynamoDB em repouso (p. 874) .	8 de fevereiro de 2018

Alteração	Descrição	Alterado em
Backup e restauração do DynamoDB	O backup sob demanda permite que você crie backups completos dos dados das tabelas do DynamoDB para arquivamento de dados, o que ajudará você a atender aos requisitos regulamentares de sua corporação e do governo. Você pode fazer backup de tabelas de alguns megabytes para centenas de terabytes de dados, sem impactar o desempenho e a disponibilidade dos aplicativos de produção. Para mais informações, consulte Backup e restauração sob demanda para o DynamoDB (p. 691).	29 de novembro de 2017
Tabelas globais do DynamoDB	As tabelas globais aproveitam a presença global do DynamoDB para oferecer a você um banco de dados totalmente gerenciado com vários ativos e várias regiões que fornece desempenho rápido e local de leitura e gravação para aplicativos globais com escalabilidade muito alta. As tabelas globais replicam as tabelas do Amazon DynamoDB automaticamente na sua escolha de AWS Regiões. Para mais informações, consulte Tabelas globais Replicação em várias regiões com o DynamoDB (p. 365).	29 de novembro de 2017
Suporte de Node.js para DAX	Os desenvolvedores Node.js podem aproveitar o DynamoDB Accelerator (DAX), usando o cliente do DAX para Node.js. Para mais informações, consulte Aceleração na memória com o DynamoDB Accelerator (DAX) (p. 713).	5 de outubro de 2017

Alteração	Descrição	Alterado em
VPC endpoints para DynamoDB	<p>Os endpoints do DynamoDB permitem que instâncias do Amazon EC2 na sua Amazon VPC acessem o DynamoDB, sem exposição à Internet pública. O tráfego de rede entre a VPC e o DynamoDB não deixa a rede da Amazon. Para mais informações, consulte Uso de endpoints da Amazon VPC para acessar o DynamoDB (p. 917).</p>	16 de agosto de 2017
Auto Scaling para DynamoDB	<p>O Auto Scaling do DynamoDB elimina a necessidade de definir ou ajustar manualmente as configurações de taxa de transferência provisionada. Em vez disso, o Auto Scaling do DynamoDB ajusta dinamicamente a capacidade de leitura e gravação em resposta aos padrões de tráfego reais. Isso permite que uma tabela ou um índice secundário global aumente a capacidade provisionada de leitura e gravação para processar aumentos repentinos no tráfego, sem limitações. Quando a carga de trabalho diminui, o Auto Scaling do DynamoDB diminui a capacidade provisionada. Para mais informações, consulte Como gerenciar a capacidade de throughput automaticamente com o Auto Scaling do DynamoDB (p. 350).</p>	14 de junho de 2017
DynamoDB Accelerator (DAX)	<p>O DynamoDB Accelerator (DAX) é um cache de memória totalmente gerenciado e altamente disponível para o DynamoDB que proporciona melhorias até 10 vezes maiores no desempenho, de milissegundos até microssegundos, mesmo com milhões de solicitações por segundo. Para mais informações, consulte Aceleração na memória com o DynamoDB Accelerator (DAX) (p. 713).</p>	19 de abril de 2017

Alteração	Descrição	Alterado em
Agora o DynamoDB oferece suporte à expiração automática de itens com TTL (Time to Live, vida útil)	O Amazon DynamoDB Time to Live (TTL) permite excluir automaticamente itens expirados de suas tabelas, sem custo adicional. Para mais informações, consulte Itens expirando usando o Time to Live (TL — Time to Live — uso do DynamoDB (p. 439)) .	27 de fevereiro de 2017
Agora, o DynamoDB aceita Tags de alocação	Você pode adicionar tags às suas tabelas do Amazon DynamoDB para melhorar a categorização de uso e ter relatórios de custo mais granulares. Para mais informações, consulte Adicionar tags e rótulos a recursos (p. 388) .	19 de janeiro de 2017
Novo DynamoDBDescribeLimits API	O DescribeLimits API retorna os limites atuais de capacidade provisionada para o seu AWS em uma região, tanto para a região como um todo quanto para qualquer tabela do DynamoDB que você tenha criado nela. Ele permite que determinar quais são os seus limites de nível de conta atuais, para que você possa compará-los com a capacidade provisionada que está usando no momento e tenha tempo de sobra para solicitar um aumento antes de atingir um limite. Para obter mais informações, consulte Cotas de serviço, conta e tabela no Amazon DynamoDB (p. 1056) e a DescribeLimits no Referência de API do Amazon DynamoDB.	1 de março de 2016

Alteração	Descrição	Alterado em
Atualização do console do DynamoDB e novas terminologias para atributos de chave primária	<p>O console de gerenciamento do DynamoDB foi reprojetado para ser mais intuitivo e fácil de usar. Como parte dessa atualização, estamos introduzindo novas terminologias para os atributos de chave primária:</p> <ul style="list-style-type: none">• Chave de partição – também conhecida como um atributo de hash.• Chave de classificação – também conhecida como um atributo de intervalo. <p>Apenas os nomes foram alterados; a funcionalidade permanece a mesma.</p> <p>Ao criar uma tabela ou um índice secundário, você pode escolher uma chave primária simples (apenas a chave de partição) ou uma chave primária composta (chave de partição e chave de classificação). A documentação do DynamoDB foi atualizada para refletir essas alterações.</p>	12 de novembro de 2015

Alteração	Descrição	Alterado em
Back-end de armazenamento do Amazon DynamoDB para Titan	<p>O DynamoDB Storage Backend for Titan é um back-end de armazenamento para o banco de dados de gráficos Titan implementado com base no Amazon DynamoDB. Ao usar o DynamoDB Storage Backend for Titan do, seus dados se beneficiam com a proteção do DynamoDB, que é executado nos datacenters de alta disponibilidade da Amazon. O plug-in está disponível para o Titan versão 0.4.4 (principalmente para compatibilidade com aplicativos existentes) e para o Titan versão 0.5.4 (recomendado para novos aplicativos). Como outros back-ends de armazenamento para Titan, esse plug-in oferece suporte à pilha Tinkerpop (versões 2.4 e 2.5), incluindo a API Blueprints e o shell Gremlin. Para mais informações, consulte Back-end de armazenamento do Amazon DynamoDB para Titan (p. 1125).</p>	20 de agosto de 2015

Alteração	Descrição	Alterado em
DynamoDB Streams, replicação entre regiões e verificação com leituras fortemente consistentes	<p>O DynamoDB Streams captura uma sequência em ordem temporal de modificações em nível de item em qualquer tabela do DynamoDB e armazena essas informações em um log por até 24 horas. Os aplicativos podem acessar esse log e visualizar os itens de dados à medida que eles aparecem antes e depois que foram modificados, em tempo quase real. Para obter mais informações, consulteCaptura de dados de alteração para DynamoDB Streams (p. 651) e aReferência de API DynamoDB Streams.</p> <p>A replicação entre regiões do DynamoDB é uma solução no lado do cliente para manter cópias idênticas de tabelas do DynamoDB em diferentes AWS regiões do, quase em tempo real. Você pode usar a replicação entre regiões para fazer backup de tabelas do DynamoDB ou fornecer acesso de baixa latência aos dados da região em que os usuários estão distribuídos geograficamente.</p> <p>O DynamoDB <code>Scan</code> operação do usa leituras eventualmente consistentes por padrão. Você pode usar leituras fortemente consistentes em vez disso, definindo o parâmetro <code>ConsistentRead</code> como true. Para obter mais informações, consulteConsistência de leitura de Scan (p. 512) e aReferência de API do Amazon DynamoDB.</p>	16 de julho de 2015

Alteração	Descrição	Alterado em
AWS CloudTrailSuporte ao Amazon DynamoDB	Agora, o DynamoDB está integrado ao CloudTrail. O CloudTrail captura chamadas a API feitas do console do DynamoDB ou da API do DynamoDB e as rastreia em arquivos de log. Para obter mais informações, consulte Registro de operações do DynamoDB usando oAWS CloudTrail (p. 954) O e a AWS CloudTrailGuia do usuário .	28 de maio de 2015
Suporte aprimorado para expressões de consulta	Esta versão adiciona um novo parâmetro <code>KeyConditionExpression</code> à API <code>Query</code> . Uma <code>Query</code> lê itens de uma tabela ou um de índice usando valores de chave primária. O parâmetro <code>KeyConditionExpression</code> é uma string que identifica nomes de chaves primárias e as condições a serem aplicada aos valores de chaves; o <code>Query</code> recupera somente os itens que atendem à expressão. A sintaxe <code>doKeyConditionExpression</code> é semelhante ao de outros parâmetros de expressão no DynamoDB e permite que você defina variáveis de substituição de nomes e valores dentro da expressão. Para mais informações, consulte Como trabalhar com consultas no DynamoDB (p. 490) .	27 de abril de 2015

Alteração	Descrição	Alterado em
Novas funções de comparação para gravações condicionais	<p>No DynamoDB, o <code>ConditionExpression</code> determina se um parâmetro <code>PutItem</code>, <code>UpdateItem</code>, ou <code>DeleteItem</code> é executado: O item é escrito somente se a condição for avaliada como verdadeira.</p> <p>Esta versão adiciona duas novas funções, <code>attribute_type</code> e <code>size</code>, para uso com <code>ConditionExpression</code>. Essas funções permitem que você realize gravações condicionais com base no tipo de dados ou no tamanho de um atributo em uma tabela. Para mais informações, consulte Expressões de condição (p. 422).</p>	27 de abril de 2015
API Scan para índices secundários	<p>No DynamoDB, um <code>Scan</code> lê todos os itens em uma tabela, aplica critérios de filtragem definidos pelo usuário e retorna os itens de dados selecionados ao aplicativo. Essa mesma capacidade agora também está disponível para índices secundários. Para verificar um índice secundário local ou um índice secundário global, você especifica o nome do índice e o nome de sua tabela pai. Por padrão, uma <code>Scan</code> de índice retorna todos os dados do índice. Você pode usar uma expressão de filtro para restringir os resultados que são retornados ao aplicativo. Para mais informações, consulte Como trabalhar com verificações no DynamoDB (p. 508).</p>	10 de fevereiro de 2015

Alteração	Descrição	Alterado em
Operações online para índices secundários globais	<p>A indexação online permite que você adicione ou remova índices secundários globais em tabelas existentes. Com a indexação online, não é necessário definir todos os índices de uma tabela quando você cria uma tabela. Em vez disso, é possível adicionar um novo índice a qualquer momento. Da mesma forma, se você decidir que não precisa mais de um índice, pode removê-lo a qualquer momento. Operações de indexação online são não bloqueantes e, portanto, a tabela permanece disponível para atividades de leitura e gravação enquanto índices estão sendo adicionados ou removidos. Para mais informações, consulte Atualizar índices secundários globais (p. 571).</p>	27 de janeiro de 2015
Suporte para modelo de documento com JSON	<p>O DynamoDB permite armazenar e recuperar documentos com suporte completo para modelos de documento. Novos tipos de dados são totalmente compatíveis com o padrão JSON e permitem que você aninhe elementos de documento uns dentro dos outros. É possível usar operadores de desreferência de caminho de documento para ler e gravar elementos individuais, sem precisar recuperar o documento inteiro. Essa versão também apresenta novos parâmetros de expressão para especificar projeções, condições e ações de atualização ao ler ou gravar itens de dados. Para saber mais sobre o suporte a modelos de documentos com o JSON, consulte Tipos de dados (p. 14) e Uso de expressões no DynamoDB (p. 414).</p>	7 de outubro de 2014

Alteração	Descrição	Alterado em
Dimensionamento flexível	Para tabelas e índices secundários globais, você pode aumentar a capacidade de throughput provisionada de leitura e gravação em qualquer quantidade, desde que permaneça nos seus limites por tabela e por conta. Para mais informações, consulte Cotas de serviço, conta e tabela no Amazon DynamoDB (p. 1056) .	7 de outubro de 2014
Tamanhos maiores de itens	O tamanho máximo de itens no DynamoDB aumentou de 64 KB para 400 KB. Para mais informações, consulte Cotas de serviço, conta e tabela no Amazon DynamoDB (p. 1056) .	7 de outubro de 2014
Expressões condicionais aprimoradas	O DynamoDB expande os operadores disponíveis para expressões condicionais, dando a você flexibilidade adicional para inserções, atualizações e exclusões condicionais. Os operadores recentemente disponíveis permitem que você verifique se um atributo existe ou não, é maior ou menor que um determinado valor, está entre dois valores, começa com certos caracteres e muito mais. O DynamoDB também fornece um operador para avaliar várias condições. Por padrão, várias condições em uma expressão são unidas por AND e, portanto, a expressão apenas será verdadeira se todas as suas condições também forem verdadeiras. Se você especificar OR em vez disso, a expressão será verdadeira se uma ou mais condições forem verdadeiras. Para mais informações, consulte Trabalho com itens e atributos (p. 404) .	24 de abril de 2014

Alteração	Descrição	Alterado em
Filtro de consulta	<p>O DynamoDB API do oferece suporte ao novo <code>QueryFilter</code> opção. Por padrão, um <code>Query</code> localiza itens que correspondem a um valor de chave de partição específico e uma condição de chave de classificação opcional. Um filtro <code>Query</code> aplica expressões condicionais a outros atributos não chave. Se um filtro <code>Query</code> estiver presente, os itens que não corresponderem às condições de filtro serão descartados antes que os resultados de <code>Query</code> sejam retornados para o aplicativo.</p> <p>Para mais informações, consulte Como trabalhar com consultas no DynamoDB (p. 490).</p>	24 de abril de 2014
Exportação e importação de dados usando o AWS Management Console	<p>O console do DynamoDB foi aprimorado para simplificar exportações e importações de dados em tabelas do DynamoDB. Com apenas alguns cliques, você pode configurar um AWS Data Pipeline para orquestrar o fluxo de trabalho e um cluster do Amazon Elastic MapReduce para copiar dados de tabelas do DynamoDB para um bucket do Amazon S3, ou vice-versa. É possível executar uma exportação ou importação somente uma vez, ou configurar um trabalho de exportação diário. Você pode até mesmo executar exportações e importações entre regiões, copiando os dados do DynamoDB de uma tabela em um AWS para uma tabela em outro AWS região.</p> <p>Para mais informações, consulte Exportar e importar dados do DynamoDB usando o AWS Data Pipeline (p. 1114).</p>	6 de março de 2014

Alteração	Descrição	Alterado em
Documentação de APIs de nível superior reorganizada	<p>Agora é mais fácil localizar informações sobre as seguintes APIs:</p> <ul style="list-style-type: none"> • Java: DynamoDBMapper • .NET: Modelo de documento e modelo de persistência de objetos <p>Essas APIs de nível mais alto agora estão documentadas aqui: Interfaces de programação de nível superior para o DynamoDB (p. 227).</p>	20 de janeiro de 2014
Índices secundários globais	O DynamoDB adiciona suporte para índices secundários globais. Como no caso de um índice secundário local, você define um índice secundário global, usando uma chave alternativa de uma tabela e emitindo solicitações Query no índice. Ao contrário de um índice secundário local, a chave de partição do índice secundário global não precisa ser igual à da tabela. Ela pode ser qualquer atributo escalar da tabela. A chave de classificação é opcional e também pode ser qualquer atributo de tabela escalar. Um índice secundário global também tem suas próprias configurações de throughput provisionado, que são separadas daquelas da tabela pai. Para mais informações, consulte índices secundários globais .	12 de dezembro de 2013

Alteração	Descrição	Alterado em
Controle de acesso minucioso	<p>O DynamoDB adiciona suporte para controle de acesso minucioso. Esse recurso permite que os clientes especifiquem quais entidades (usuários, grupos ou funções) podem acessar itens e atributos individuais em uma tabela ou índice secundário do DynamoDB. Os aplicativos também podem tirar proveito da federação de identidades da Web para descarregar a tarefa de autenticação de usuários para um provedor de identidade de terceiros, como o Facebook, o Google ou o Login com o Amazon. Dessa forma, os aplicativos (incluindo aplicativos móveis) podem lidar com um grande número de usuários e, ao mesmo tempo, garantir que ninguém pode acessar itens de dados do DynamoDB sem autorização. Para mais informações, consulte Uso de condições de política do IAM para controle de acesso refinado (p. 899).</p>	29 de outubro de 2013
4 KB de tamanho da unidade de capacidade de leitura	<p>O tamanho da unidade de capacidade para leituras aumentou de 1 KB para 4 KB. Esse aumento pode reduzir o número de unidades de capacidade de leitura provisionadas necessárias para muitos aplicativos. Por exemplo, antes desta versão, a leitura de um item de 10 KB consumiria 10 unidades de capacidade de leitura. Agora, a mesma leitura de 10 KB consumiria apenas 3 unidades (10 KB/4 KB, arredondados para o próximo limite de 4 KB). Para mais informações, consulte Modo de capacidade de leitura/gravação (p. 18).</p>	14 de maio de 2013

Alteração	Descrição	Alterado em
Verificações paralelas	O DynamoDB adiciona suporte para operações de verificação paralelas. Agora, os aplicativos podem dividir uma tabela em segmentos lógicos e verificar todos esses segmentos simultaneamente. Esse recurso reduz o tempo necessário para uma conclusão de uma verificação e utiliza totalmente a capacidade de leitura provisionada da tabela. Para mais informações, consulte Como trabalhar com verificações no DynamoDB (p. 508) .	14 de maio de 2013
Índices secundários locais	O DynamoDB adiciona suporte para índices secundários locais. Você pode definir índices de tipo de chave em atributos não chave e depois usar esses índices em solicitações de Consulta. Com índices secundários locais, os aplicativos podem recuperar itens de dados eficientemente entre várias dimensões. Para mais informações, consulte Índices secundários locais (p. 604) .	18 de abril de 2013
Nova versão de API	Com este lançamento, o DynamoDB introduz uma nova versão de API (2012-08-10). A versão anterior da API (2011-12-05) ainda tem suporte para compatibilidade com aplicativos existentes. Novos aplicativos devem usar a nova API de versão 2012-08-10. Recomendamos que você migre seus aplicativos existentes para a API versão 2012-08-10, pois não será feito o backport de novos recursos do DynamoDB (como índices secundários) para a versão anterior da API. Para obter mais informações sobre a API versão 2012-08-10, consulte Referência de API do Amazon DynamoDB .	18 de abril de 2013

Alteração	Descrição	Alterado em
Supporte para variáveis de política do IAM	<p>Agora, a linguagem de política de acesso do IAM oferece suporte para variáveis. Quando uma política é avaliada, todas as variáveis de política são substituídas por valores fornecidos por informações baseadas no contexto da sessão do usuário autenticado. Você pode usar variáveis de política para definir políticas de uso geral sem, explicitamente, listar todos os componentes da política. Para obter mais informações sobre variáveis de políticas, visite Variáveis de políticas no AWS Identity and Access Management Uso de IAM Guia.</p> <p>Para obter exemplos de variáveis de políticas no DynamoDB, consulte Identity and Access Management no Amazon DynamoDB (p. 883).</p>	4 de abril de 2013
Exemplos de código PHP atualizados para o AWS SDK for PHP versão 2	Versão 2 do AWS SDK for PHP Agora, o está disponível. Os exemplos de código do PHP no Amazon DynamoDB Developer Guide foram atualizados para usar esse novo SDK. Para obter mais informações sobre a versão 2 do SDK, consulte AWS SDK for PHP .	23 de janeiro de 2013
Novo endpoint do	O DynamoDB se expande para o AWS Região GovCloud (Oeste dos EUA). Para obter a lista atual de endpoints de serviços e protocolos, consulte Regiões e endpoints .	3 de dezembro de 2012
Novo endpoint do	O DynamoDB se expande para a região da América do Sul (São Paulo). Para obter a lista atual de endpoints com suporte, consulte Regiões e endpoints .	3 de dezembro de 2012
Novo endpoint do	O DynamoDB se expande para a região Ásia-Pacífico (Sydney). Para obter a lista atual de endpoints com suporte, consulte Regiões e endpoints .	13 de novembro de 2012

Alteração	Descrição	Alterado em
O DynamoDB implementa suporte para somas de verificação CRC32, oferece suporte para obtenções de lote fortemente consistentes e remove restrições em atualizações de tabelas simultâneas.	<ul style="list-style-type: none"> O DynamoDB calcula uma soma de verificação CRC32 da carga útil HTTP e retorna essa soma em um novo cabeçalho, <code>x-amz-crc32</code>. Para mais informações, consulte API do DynamoDB de baixo nível (p. 217). Por padrão, as operações de leitura realizadas pela API <code>BatchGetItem</code> são eventualmente consistentes. Um novo parâmetro <code>ConsistentRead</code> em <code>BatchGetItem</code> permite que você escolha a consistência de leitura forte para qualquer tabela na solicitação. Para mais informações, consulte Description (p. 1152). Essa versão remove certas restrições ao atualizar muitas tabelas simultaneamente. O número total de tabelas que podem ser atualizadas ao mesmo tempo ainda é 10. No entanto, essas tabelas agora podem ser qualquer combinação de status <code>CREATING</code>, <code>UPDATING</code> ou <code>DELETING</code>. Além disso, não existe mais uma quantidade mínima para aumentar ou reduzir os valores de <code>ReadCapacityUnits</code> ou <code>WriteCapacityUnits</code> para uma tabela. Para mais informações, consulte Cotas de serviço, conta e tabela no Amazon DynamoDB (p. 1056). 	2 de novembro de 2012
Documentação de práticas recomendadas	O Amazon DynamoDB Developer Guide identifica as práticas recomendadas para trabalhar com tabelas e itens, juntamente com recomendações para operações de consulta e verificação.	28 de setembro de 2012

Alteração	Descrição	Alterado em
Suporte para tipos de dados binário	<p>Além dos tipos Número e String, o agora oferece suporte ao tipo de dados Binário.</p> <p>Antes desta versão, para armazenar dados binários, você convertia dados binários no formato de string e os armazenava no DynamoDB. Além do trabalho de conversão necessário no lado do cliente, a conversão muitas vezes aumentava o tamanho do item de dados, exigindo mais armazenamento e possivelmente capacidade adicional de throughput provisionado.</p> <p>Agora, com atributos do tipo Binário, você pode armazenar quaisquer dados binários, por exemplo, dados compactados, dados criptografados e imagens. Para obter mais informações, consulte Tipos de dados (p. 14). Para obter exemplos funcionais de como manipular dados do tipo Binário usando o recurso AWSSDKs, consulte as seções a seguir:</p> <ul style="list-style-type: none"> • Exemplo: Tratamento de atributos do tipo binário usando o AWS SDK for JavaDocumentar API (p. 463) • Exemplo: Tratamento de atributos do tipo binário usando o AWS SDK for .NETAPI de baixo nível (p. 487) <p>Para o suporte de tipo de dados binários adicionado no AWSSDKs, você precisará baixar os SDKs mais recentes e talvez também precise atualizar todos os aplicativos existentes. Para obter informações sobre como fazer download do AWSSDKs, consulte Exemplos de código .NET (p. 336).</p>	21 de agosto de 2012

Alteração	Descrição	Alterado em
Itens de tabela do DynamoDB podem ser atualizados e copiados usando o console do DynamoDB	Agora, os usuários do DynamoDB podem atualizar e copiar itens de tabela usando o Console do DynamoDB, além de serem capazes de adicionar e excluir itens. Essa nova funcionalidade simplifica o processo de fazer alterações em itens individuais usando o Console.	14 de agosto de 2012
O DynamoDB reduz os requisitos mínimos de throughput de tabela	Agora, o DynamoDB oferece suporte a uma taxa de transferência mínima de tabela, ou seja, 1 unidade de capacidade de gravação e 1 unidade de capacidade de leitura. Para obter mais informações, consulte o .Cotas de serviço, conta e tabela no Amazon DynamoDB (p. 1056) no Guia do desenvolvedor do Amazon DynamoDB.	9 de agosto de 2012
Suporte para o Signature Version 4	Agora, o DynamoDB oferece suporte ao Signature Version 4 para autenticar solicitações.	5 de julho de 2012
Suporte para agente de exploração de tabelas no Console do DynamoDB	Agora, o Console do DynamoDB oferece suporte para um agente de exploração de tabelas que permite navegar e consultar dados em tabelas. Você também pode inserir novos itens ou excluir itens existentes. As seções Criar tabelas e carregar dados para exemplos de código no DynamoDB (p. 329) e Usar o console (p. 60) foram atualizadas para esses recursos.	22 de maio de 2012
New endpoints (Novos endpoints)	A disponibilidade do DynamoDB se expande com novos endpoints na região Oeste dos EUA (Norte da Califórnia), Oeste dos EUA (Oregon) e região Ásia-Pacífico (Cingapura). Para obter a lista atual de endpoints com suporte, acesse Regiões e endpoints .	24 de abril de 2012

Alteração	Descrição	Alterado em
Suporte à API BatchWriteItem	<p>Agora, o DynamoDB oferece suporte a uma API de gravação em lote que permite que você insira e exclua vários itens de uma ou mais tabelas em uma única chamada a API. Para obter mais informações sobre a API de gravação em lote do DynamoDB, consulte BatchWriteItem (p. 1157).</p> <p>Para obter informações sobre como trabalhar com itens e usando o recurso de gravação em lote do usando o AWSSDKs, consulte Trabalho com itens e atributos (p. 404) e Exemplos de código .NET (p. 336).</p>	19 de abril de 2012
Mais códigos de erro documentados	Para mais informações, consulte Tratamento de erros com o DynamoDB (p. 221) .	5 de abril de 2012
Novo endpoint do	O DynamoDB se expande para a região Ásia-Pacífico (Tóquio). Para obter a lista atual de endpoints com suporte, consulte Regiões e endpoints .	29 de fevereiro de 2012
Métrica <code>ReturnedItemCount</code> adicionada	Uma nova métrica, <code>ReturnedItemCount</code> fornece o número de itens retornados na resposta de uma operação Query ou Scan para o DynamoDB que está disponível para monitoramento por meio do CloudWatch. Para mais informações, consulte Registro em log e monitoramento no DynamoDB (p. 928) .	24 de fevereiro de 2012
Exemplos adicionados para incrementar valores	<p>O DynamoDB oferece suporte a operações para incrementar e diminuir valores numéricos existentes. Exemplos mostram como somar a valores existentes nas seções "Como atualizar um item" em:</p> <p>Trabalho com itens: Java (p. 446).</p> <p>Trabalho com itens: .NET (p. 466).</p>	25 de janeiro de 2012

Alteração	Descrição	Alterado em
Versão inicial do produto	O DynamoDB é apresentado como um novo serviço na versão Beta.	18 de janeiro de 2012

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.