Funcionamento do DynamoDB

Funcionamento do Amazon DynamoDB

Bem-vindo! Nesta seção do curso, vamos falar sobre como o DynamoDB funciona.

Tabelas, itens e partições

No DynamoDB, os dados são armazenados em tabelas. Você pode criar ou excluir uma tabela com uma simples chamada de API. São colocados itens na tabela, semelhantes às linhas em uma tabela de banco de dados relacional.

Cada item tem atributos, análogos às colunas em um banco de dados relacional. Cada item requer pelo menos um atributo, a chave de partição.

Opcionalmente, você pode definir uma chave de classificação, que se tornaria outro atributo obrigatório em cada item.

Juntas, a chave de partição e a chave de classificação (se definida) compõem a chave primária, que deve ser única. Este também é um conceito familiar para aqueles que têm conhecimentos sobre bancos de dados relacionais.

Cada item também pode ter outros atributos, e o conjunto de outros atributos é flexível de um item para outro. Essa flexibilidade do esquema é uma diferença fundamental com as tecnologias relacionais. Falaremos sobre algumas formas interessantes de aproveitar isso mais tarde no curso.

Quando precisam de ajuda com escalabilidade, os bancos de dados relacionais empregam um conceito chamado fragmentação, que divide o conjunto de dados em vários servidores ou clusters.

Essa funcionalidade é automatizada no DynamoDB e permite a escalabilidade horizontal para lidar com qualquer volume de carga e dados imaginável, sem qualquer esforço administrativo contínuo! É feito um hash no valor da chave de partição para criar um valor e distribuir os itens em várias partições.

Esse particionamento é um recurso gerenciado do DynamoDB. Tudo é tratado internamente ao serviço, mas é útil ter uma compreensão conceitual para entender algumas práticas recomendadas de design.

O DynamoDB pode particionar a tabela por vários motivos, geralmente para distribuir o volume de dados de forma mais ampla ou atender a taxas de solicitação maiores nas chaves primárias.

Detalhes sobre itens

Para aprofundar um pouco mais, o DynamoDB suporta os seguintes tipos de dados para atributos: número, string, binário (codificado em base 64), booleano e nulo.

Você também pode usar conjuntos de números, strings e binários (entradas em um conjunto devem ser do mesmo tipo e únicas). Os conjuntos não mantêm a ordem.

Além disso, e se a sua aplicação tende a trabalhar com documentos JSON?

Você pode adicionar os documentos JSON como atributos individuais ou empregar um modelo onde cada item da tabela é um documento JSON, usando mapa (atributos aninhados – entradas não ordenadas de tipo misto) e lista (semelhante a uma matriz JSON – listas ordenadas de valores de qualquer tipo de dados).

Esses tipos de dados de documentos podem ser aninhados em até 32 níveis de profundidade. Somente os atributos marcados em verde são obrigatórios. Eles fazem parte da chave primária. O restante é opcional e varia de item para item. Estamos começando a perceber o valor na flexibilidade do esquema.

Observe que os atributos de chave primária devem ser do tipo string, número ou binário. As entradas de mapa e lista podem ser referenciadas diretamente, mas se você quiser usá-las como chaves primárias é preciso expô-las como atributos separados.

Mais sobre chaves primárias

Selecionar chaves primárias apropriadas é fundamental para o sucesso do DynamoDB. Vamos falar mais sobre elas. Sabemos que cada item de uma tabela deve ser identificado exclusivamente por uma chave primária.

No caso mais simples, a chave primária é baseada em um único atributo: a chave de partição.

Aqui temos uma tabela usada em uma aplicação de exemplo que pode monitorar a temperatura atual através de sensores instalados em vários locais diferentes. O sensor é ativado periodicamente e grava a temperatura atual na tabela do DynamoDB, usando SensorId como a chave de partição. Os itens de vários sensores são distribuídos em todas as partições do DynamoDB na tabela.

E para manter um registro histórico de leituras de temperatura para cada sensor? A partir da nossa aplicação, precisamos ser capazes de apresentar eficientemente uma visualização do ponto de dados mais recente ou os cinco pontos de dados mais recentes.

Poderíamos usar uma chave primária composta: SensorId como chave de partição e timestamp como chave de classificação. SensorId é usado para distribuir a carga de trabalho entre partições do DynamoDB e, para qualquer SensorId, os registros de temperatura são mantidos em ordem classificada.

Durabilidade e disponibilidade

Os dados são altamente resilientes e disponíveis em uma tabela do DynamoDB. Todos os dados são replicados automaticamente em vários discos de estado sólido independentes, em servidores separados, em data centers isolados de falhas.

Você provavelmente está familiarizado com o conceito de zonas de disponibilidade da AWS. Qualquer chamada de gravação criada não será bem-sucedida até que os dados sejam armazenados de forma redundante (pelo menos duas cópias). Outras cópias convergem muito rapidamente (geralmente em um segundo ou menos).

Se algum servidor que armazena os dados sofre uma falha, ele é removido do conjunto de replicação em segundos e substituído automaticamente. Isso permite que o DynamoDB forneça um Acordo de Nível de Serviço com quatro noves (ou seja, 99,99%) de disponibilidade.

Consistência

Agora que sabemos sobre a replicação no DynamoDB, vamos discutir a consistência. Consistência é a capacidade de ler dados com o entendimento de que todas as gravações anteriores serão refletidas nos resultados retornados.

Muitas vezes isso é conhecido como consistência de leitura após gravação. As gravações por definição são consistentes, mas as leituras podem ser “fortemente” consistentes ou “eventualmente” consistentes. Esta escolha é feita pelo cliente a cada solicitação de leitura.

Consistência eventual é o comportamento padrão, com consistência forte disponível como uma opção para cada operação de leitura. Podemos ver um exemplo da diferença neste diagrama. O cliente grava uma atualização na Chave 1, que é persistente de forma duradoura.

A cópia na zona de disponibilidade A é uma das gravadas imediatamente. A cópia na zona de disponibilidade B ainda não foi replicada.

Agora um cliente quer ler o item. Vamos começar com um cenário de solicitação de leitura eventualmente consistente.

Vemos a solicitação de leitura eventualmente consistente para o DynamoDB. O DynamoDB pode optar por encaminhar a solicitação para qualquer uma das cópias da zona de disponibilidade mostradas. O resultado retornado pode ser o valor atualizado “B” ou o valor obsoleto “A”.

Aqui temos o cenário de solicitação de leitura fortemente consistente. O DynamoDB garante que a solicitação seja encaminhada para uma cópia da zona de disponibilidade com as atualizações mais recentes.

O cliente sempre verá o valor atualizado “B”. É tentador usar sempre leituras altamente consistentes, mas o DynamoDB cobra mais por elas porque o trabalho está concentrado em um número menor de cópias replicadas.

Além disso, pode ser que em alguns cenários de falha as leituras fortemente consistentes (SC) fiquem brevemente indisponíveis. Por fim, uma leitura fortemente consistente pode não fornecer inerentemente as garantias esperadas. Ela não é um mecanismo de bloqueio.

Se você fizer uma leitura fortemente consistente e voltar para gravar no mesmo item com base nos dados lidos, pode descobrir que outro cliente fez uma atualização diferente nesse intervalo.

Como prática recomendada, tente trabalhar com leituras eventualmente consistentes (EC) sempre que possível. Vamos ver algumas estratégias para isso mais tarde no curso.

Taxa de transferência de solicitações

O DynamoDB permite especificar a quantidade necessária de taxa de transferência de solicitações e cuida de tudo para atender a essa expectativa.

Você pode alterar a taxa de transferência provisionada conforme necessário. Os dados serão distribuídos em quantas partições forem necessárias para fornecer o nível de desempenho que você precisa. A capacidade da solicitação de leitura e a capacidade da solicitação de gravação são gerenciadas separadamente.

Uma unidade de capacidade de leitura (RCU) é consumida durante a leitura de um item de até 4 KB a cada segundo. Uma unidade de capacidade de gravação (WCU) é consumida ao gravar um item de até 1 kB a cada segundo.

Observe que a atualização de um único atributo em um item requer a gravação do item inteiro. A taxa de transferência é geralmente dividida uniformemente entre as partições. Por isso, é importante projetar solicitações que sejam distribuídas uniformemente entre as chaves.

O DynamoDB tem recursos chamados de burst (que seria como minutos de transferência em um plano de celular) e capacidade adaptável (que é uma compensação inteligente, permitindo "emprestar" uma taxa de transferência não utilizada de chaves menos ativas para cobrir as necessidades de uma chave mais movimentada).

Isso permite um certo desequilíbrio de tráfego entre as chaves. Com o tempo, o serviço tornou-se muito mais tolerante com cargas de trabalho desequilibradas e continua a evoluir nessa direção.

Observe que um único item nunca pode ser lido em mais de 3000 RCU, ou gravado em mais de 1000 WCU (ou uma combinação linear das duas). Falaremos sobre como projetar isso mais tarde.

Observe também que leituras eventualmente consistentes são cobradas à metade do custo de leituras fortemente consistentes. Duas leituras eventualmente consistentes de 4 kB consumirão apenas 1 RCU. O que acontece se você tentar usar mais taxa de transferência do que a provisionada?

O DynamoDB limita sua solicitação. Você receberá uma resposta dizendo "volte mais tarde".

Solicitações básicas de itens

Agora vamos falar brevemente sobre os tipos de solicitações que podem ser feitas relacionadas aos itens na tabela.

PutItem é usado para inserir um novo item ou sobrescrever um existente com base na chave primária especificada.

UpdateItem permite alterar atributos específicos de um item identificado por uma determinada chave primária, mas basicamente substitui o item inteiro. Portanto, a cobrança é feita de acordo com todas as WCUs necessárias.

DeleteItem é usado para excluir um item especificado pela chave primária. Custa o mesmo número de WCUs para excluir um item que para criá-lo.

GetItem recupera o item associado a uma chave primária específica.

BatchWriteItem e BatchGetItem permitem enviar vários puts/gets de uma só vez. Isso pode ser mais eficiente do que abrir novas conexões para solicitações individuais.

Scan (Varredura) faz a leitura de toda a tabela. Não é algo que você deva fazer com frequência e, quando for o caso, é recomendável um controle para evitar consumir toda a taxa de transferência provisionada para a tabela.

Finalmente, a minha favorita, a consulta. Query (Consulta) só pode ser usada em tabelas com chaves primárias compostas (partição e classificação). Na consulta você especifica a chave de partição de interesse e, em seguida, uma expressão de chave de classificação correspondente (igual a, menor que, maior que, começa com, etc.).

A consulta não é apenas poderosa, mas eficiente. Em vez de cobrar RCUs com base em tamanhos de item individuais (arredondando para incrementos de 4 kB), ela resume o tamanho de todos os itens no conjunto de resultados e, em seguida, arredonda para os 4 kB mais próximos.

Todas as operações de leitura podem ser eventualmente consistentes (como padrão) ou fortemente consistentes. Você também pode pedir ao DynamoDB que retorne apenas um determinado conjunto de atributos ou que filtre determinados resultados do conjunto.

Esses filtros podem ser úteis para simplificar sua aplicação. Eles reduzem a transferência de rede, mas não alteram o consumo de taxa de transferência associado ao conjunto completo de resultados com todos os atributos.

Índices secundários

E se você quiser ler os dados na tabela através da perspectiva de uma chave primária diferente? E se você quiser reduzir o consumo da taxa de transferência buscando apenas alguns dos itens ou atributos? Os índices secundários do DynamoDB são a resposta.

Com um índice secundário, você pode direcionar as chamadas de consulta e varredura para o índice em vez da tabela base.

Os itens no índice podem conter todos os atributos do item da tabela base, apenas os atributos da tabela base e da chave primária de índice, ou esses atributos de chave mais um conjunto específico de atributos adicionais (isto é chamado de projeção).

Isso pode ser usado para fazer leituras seletivas muito eficientes de subconjuntos de dados na tabela base. Existem dois tipos de índices secundários disponíveis no DynamoDB: índices secundários locais e índices secundários globais.

Índices secundários locais (LSI)

Os índices secundários locais, chamados LSIs, são locais a uma chave de partição particular. Um LSI sempre tem a mesma chave de partição que a tabela base. Estão localizados nas mesmas partições que a tabela base e compartilham a capacidade provisionada da tabela base.

Como o LSI está localizado na mesma partição que a tabela base, o tamanho total de dados dos itens associados a uma determinada chave de partição é limitado a aproximadamente 10 GB.

Normalmente, esse é o ponto em que o DynamoDB dividiria partições, mas o LSI não pode existir em dois lugares. Os LSIs podem ser usados para consultar itens dentro de uma chave de partição especificada, muitas vezes para classificação com base em um atributo diferente da tabela base.

Aqui temos um exemplo de tabela base usada para armazenar registros de uma pesquisa de pacientes. RepId (chave de partição) é o indivíduo que realizou a pesquisa, e PatientId (chave de classificação) é o número de identificação de um paciente. O atributo adicional é ContactDate, a data em que a pesquisa foi conduzida.

A tabela base suporta consultas para pacientes classificados (por PatientId) que falaram com um determinado RepId.

Aqui temos um LSI que permitirá que a aplicação consulte a lista de PatientIds que fizeram a pesquisa com um RepId específico dentro de um intervalo de dados fornecido.

Os LSIs podem lidar com leituras eventualmente consistentes e fortemente consistentes. Observe que eles só podem ser definidos quando a tabela base é criada e não podem ser modificados ou excluídos durante o tempo de vida da tabela base.

Índice secundário global (GSI)

Os índices secundários globais, chamados GSIs, são muito mais flexíveis do que os LSIs. Eles não são locais para uma chave de partição e podem abranger todas as chaves de partição na tabela base. Pense em um GSI como outra tabela completamente separada replicada pelo DynamoDB a partir da tabela base.

Os GSIs não fornecem consistência forte para leituras. Os GSIs não estão sujeitos à limitação de tamanho da coleção como os LSIs. Um GSI pode usar um atributo totalmente diferente como chave de partição, e qualquer atributo escalar pode se tornar uma chave de classificação.

Os GSIs têm sua própria taxa de transferência provisionada, que pode ser gerenciada independentemente da tabela base. Isso permite um grau de isolamento que pode ser operacionalmente muito útil na separação de casos de uso. Observe que um GSI pode ser criado e excluído a qualquer momento, o que significa que você pode criar um GSI para atender a uma necessidade temporária e excluí-lo assim que terminar.

Aqui temos uma tabela de exemplo em que um departamento de saúde registra relatórios de infecção. PatientId é usado como chave primária para distribuir adequadamente a carga entre as partições.

Mais tarde, decide-se ativar um recurso na aplicação que fornece uma lista de relatórios de infecção em uma determinada cidade classificada pela data do relatório.

Aqui temos o GSI criado para apoiar essa solicitação. Observe que os GSIs não exigem chaves primárias exclusivas. Portanto, é possível ter vários relatórios de infecção na mesma cidade, no mesmo dia. Uma consulta pode retornar vários itens para a mesma chave primária.

Streams

Os streams do DynamoDB são muito parecidos com os streams do Kinesis. Na verdade, eles são compatíveis com a biblioteca de cliente do Kinesis. Todas as gravações na tabela são registradas no stream, como um log de alterações.

O stream é estritamente ordenado de acordo com as mudanças na tabela, e a contagem de fragmentos de streams cresce com a tabela. Ao habilitar um stream de tabela, você pode escolher quantos detalhes incluir no log. Os streams são duráveis e mantidos por até 24 horas.

Neste exemplo, podemos ver que Jane decidiu que prefere fazer spinning no lugar de caminhar. O item foi atualizado e a atualização é registrada no stream da tabela.

Isso termina nosso módulo sobre o funcionamento do DynamoDB. No próximo módulo, falaremos sobre como operar o DynamoDB.