Pós-Graduação
DynamoDB

Título: Projeto Bloco C - DynamoDB

Autores: Amanda Campos de Santana

Matheus Ramalho Aires

Orientador: Eduardo Morelli

Rio de Janeiro, maio de 2016

Sumário

1. Detalhamento Teórico

1.1. Resumo

O Amazon DynamoDB é um serviço de banco de dados NoSQL rápido e flexível para todas as aplicações que precisam de latência constante abaixo de 10 milissegundos em qualquer escala.

O serviço é um banco de dados em nuvem totalmente gerenciado e é compatível com os modelos de armazenamento de documentos e de chave-valor.

O DynamoDB foi anunciado pelo CTO da Amazon, Werner Vogels, para integrar os serviços da Amazon Web Services em 18 de janeiro de 2012. Os crescimentos dos negócios da Amazon indicavam que era necessário investir em um banco de dados mais escalável.

Os problemas do SimpleDB como ter um limite de dados de 10 GB por domínio ele ainda não era auto escalável. Sendo assim, o DynamoDB é usado para gerenciar o estado dos serviços que possuem requisitos muito elevados de confiabilidade e precisam de um controle rígido sobre as compensações entre disponibilidade, consistência, custo, eficácia e desempenho.

Trata-se de um banco de dados não relacional (NoSQL) extremamente rápido (a base de servidores usa discos SSD), altamente disponível (utiliza múltiplas Availability Zones da AWS), e totalmente gerenciado pela AWS. Como todos os serviços da AWS, para usar o DynamoDB você paga somente o que precisa usar, com um custo reduzido.

Basicamente, é preciso criar uma tabela no DynamoDB e utilizar o respectivo SDK da AWS para integrar de maneira quase transparente a iniciação de sessões da aplicação no DynamoDB.

1.2. Aplicações

O DynamoDB possui modelo de dados flexível e desempenho confiável para aplicativos móveis e da Web, jogos, tecnologia de anúncios, IoT e outras aplicações.

- **BMW**

A BMW utilizou o DynamoDB para armazenar informações de sensores em seus automóveis para otimizar o GPS do BMW 7 Series, com isso o GPS é dinamicamente atualizado.

O projeto foi totalmente desenvolvido em apenas 6 meses e também utilizou outros recursos do AWS.

As melhorias foram limites de velocidade mais precisos, rotas otimizadas, diminuição dos erros de mapa e melhorias na geometria das estradas.

- **Duolingo**

Como uma empresa jovem, o Duolingo queria se concentrar em maximizar o crescimento e trazendo novos recursos para o mercado o mais rápido possível. "Nós decidimos que não queríamos para gerenciar hardware", diz CTO e cofundador, Severin Hacker. ” Nós queríamos concentrar no que somos bons . A aplicação e o código"

Os fundadores decidiram que a tecnologia de nuvem seria um ajuste melhor para a sua empresa e começaram a procurar um provedor de serviços em nuvem. Depois de considerar várias opções, Duolingo selecionou o Amazon Web Services (AWS). "O AWS tinha suporte ao desenvolvedor e boa documentação", diz Hacker. "Em 2011, a AWS foi o serviço de nuvem mais amplamente adotado e tinha a plataforma mais madura.

Na época, a AWS foi um dos poucos fornecedores de serviços em nuvem que oferece um serviço de banco de dados gerenciado. "Duolingo armazena dados para criar um modelo para a experiência de aprendizagem de cada usuário.

O aplicativo usa esse modelo para gerar lições personalizadas. Por exemplo, você pode entender a palavra para o sono em alemão, mas tem problemas com tempos de gramática. Se você não usar o aplicativo por um tempo, Duolingo vai entregar uma lição diferente para rever o que você aprendeu anteriormente. Para oferecer uma experiência de aprendizagem personalizada, Duolingo rastreia as palavras e conceitos que um usuário tenha visto. Inicialmente, estes dados foram armazenados em um banco de dados

MySQL usando um RDS Amazon. Hacker decidiu DynamoDB foi a solução para a ampliação de dados de vocabulário de Duolingo. "O serviço mudou fundamentalmente a nossa situação", diz Hacker. "Nós podemos usar DynamoDB para aumentar o rendimento, conforme necessário e não ter de lidar com qualquer outra coisa. Nós teríamos desintegrado até agora sem AWS".

A quantidade de dados que o Duolingo armazena aumentou de 100 milhões de itens para mais de quatro bilhões de itens. Os itens são atualizados quase mil vezes por segundo e recuperados muitos milhares de vezes por segundo. Duolingo ainda usa Amazon RDS e MySQL para muitas de suas necessidades, mas usa DynamoDB exclusivamente para mudar frequentemente de dados.

- **Supercell**

Supercell é uma empresa da Finlândia, fundada em 2010, é uma das empresas mais cresce no ramo de jogos sociais do mundo. Com mais de 100 funcionários, seus três jogos são de grande sucesso, atraindo dezenas de milhões de jogadores em iOS e dispositivos Android todos os dias. Estes jogos são *Hay Day*, um jogo de fazenda e *Clash of Clans* e *Boom Beach*, que combinam a gestão dos recursos sociais e elementos de combate estratégicos.

O DynamoDB armazena mais de 45 Bilhões de eventos por dia e foi escolhido por causa da escalabilidade e velocidade. Além disto, a Supercell também utiliza outros recursos AWS.

1.3. Arquitetura

1.3.1. Interface

O Dynamo possui duas operações – get() e put(). A operação put() determina a partir da chave do objeto onde as réplicas devem ser salvas, e grava as réplicas em disco. A operação get() retorna um objeto ou uma lista de objetos com versões conflitantes. Como mencionado previamente, o Dynamo deixa a resolução de conflitos para a aplicação cliente. As operações get() e put() são invocadas usando um framework específico da Amazon, que

encapsula HTTP. Há uma lógica de roteamento que considera a distribuição da carga e também a saúde atual dos nós, e aí as requisições são encaminhadas para o nó adequado.

1.3.2. Particionamento

Os nós do Dynamo se distribuem em um anel. Cada nó no sistema recebe um valor aleatório que determina a sua posição no anel. Cada novo objeto é associado a um nó pela aplicação de um hash ao seu ID, e aí percorremos o anel em sentido horário até chegar no primeiro nó com posição maior que a do objeto em questão. Para lidar melhor com as falhas em nós, o Dynamo atribui múltiplas posições para cada nó, em um conceito chamado de “nós virtuais”. Com essa abordagem o Dynamo consegue redistribuir bem os nós quando algum deles falha e a disponibilidade não fica comprometida.

1.3.3. Replicação

Para atingir alta disponibilidade e durabilidade, o Dynamo replica os dados em múltiplos nós. Cada item é replicado em N itens, sendo que este valor N é configurável. Além do nó original, o item é gravado nos N-1 nós seguintes, em sentido horário. A lista de nós responsáveis por guardar cada chave é chamada de lista de preferência, e possui mais de N nós, para conseguir lidar com falhas em nós individuais.

1.3.4. Versionamento

O Dynamo oferece consistência eventual, na qual as atualizações são propagadas para as réplicas assincronamente. Normalmente há um limite no tempo para as atualizações chegarem às réplicas, mas em cenários de falha este tempo pode ser muito longo. Quando um cliente envia uma atualização de um item, ele especifica qual é a versão que está sendo atualizada. Em cenários de falha pode ser necessário fazer um merge de versões conflitantes. Entretanto, isto é feito em tempo de leitura, pois o Dynamo se propõe a sempre aceitar escritas, como mencionado anteriormente.

1.3.5. Tratamento de falhas

O Dynamo tem um mecanismo para que operações de leitura e escrita envolvam N nós saudáveis. É feito um controle de nós saudáveis ao longo do tempo, e com isso o sistema se protege de eventuais falhas de rede e em nós individuais.

As operações de leitura e escrita são realizadas nos primeiros N nós saudáveis da lista de preferência, que pode não ser sempre os primeiros N nós encontrados ao andar o anel.

Neste exemplo, se um nó está temporariamente indisponível ou inacessível durante uma operação de gravação, uma réplica que normalmente teria estaria em A será agora enviada para o nó D. Isto é feito para manter a disponibilidade e durabilidade desejadas. A réplica enviada para D terá em seu metadados que o nó destinatário pretendido da réplica (neste caso A). Nós que recebem réplicas sugeridos irão mantê-las em um banco de dados local separado, que é escaneado periodicamente. Ao detectar que o nó A se recuperou, D tentará entregar a réplica A. Assim que a transferência for bem-sucedida, a réplica será excluída do nó D sem diminuir o número total de réplicas no sistema.

1.3.6. Detecção de novos nós e de falhas

O Dynamo usa um protocolo baseado em redes *peer-to-peer* para que os nós percebam rapidamente as falhas de nós existentes e a entrada de novos nós no anel.

1.4. Instalação

Além do serviço de web Amazon DynamoDB, AWS fornece uma versão para download do DynamoDB que pode ser executado localmente. Esta edição do DynamoDB permite escrever aplicações sem acessar o serviço web real Amazon DynamoDB.

A versão local do DynamoDB pode ajudar a economizar em taxa de transferência provisionada, armazenamento de dados e taxas de transferência

de dados. Além disso, não é preciso ter uma conexão com a Internet enquanto a aplicação estiver em desenvolvimento. Quando tudo estiver pronto para implantar o aplicativo em produção, podem ser feitas algumas pequenas alterações no código programado para ele usar o serviço Amazon DynamoDB.

A versão para download do DynamoDB é fornecido como um executável .jar arquivo. Ele será executado em Windows, Linux, Mac OS X, e outras plataformas que suportam Java.

Passos para instalação:

1) Baixar o DynamoDB.

Formato *.tar.gz* : http://dynamodb-local.s3-website-us-west-2.amazonaws.com/dynamodb_local_latest.tar.gz

Formato *.zip*: http://dynamodb-local.s3-website-us-west-2.amazonaws.com/dynamodb_local_latest.zip

2) Depois de ter baixado o arquivo, extrair o conteúdo e copiar o diretório extraído para um local de sua escolha.

3) Para iniciar DynamoDB no computador, basta abrir uma janela do prompt de comando, navegar até o diretório onde foi extraído o DynamoDBLocal.jar, e digitar o seguinte comando:

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

Serão visualizadas as mensagens de diagnóstico ocasionais na janela onde DynamoDB está em execução.

1.5. Interação

1.5.1. API

O DynamoDB dispõe de três linguagens para se conectar: Java, .Net e PHP.

Criação de um Item:

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient(new ProfileCredentialsPro
vider());
String tableName = "ProductCatalog";

Map<String, AttributeValue> item = new HashMap<String, AttributeValue>();
item.put("Id", new AttributeValue().withN("104"));
item.put("Title", new AttributeValue().withS("Book 104 Title"));
item.put("ISBN", new AttributeValue().withS("111-1111111111"));
item.put("Price", new AttributeValue().withS("25.00"));
item.put("Authors", new AttributeValue().withSS(Arrays.asList("Author1", "Author2")));

PutItemRequest putItemRequest = new PutItemRequest().withTableName(tableName).withItem
(item); PutItemResult result = client.putItem(putItemRequest);
```

Lendo um Item:

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient(new ProfileCredentialsPro
vider());
HashMap<String, AttributeValue> key = new HashMap<String, AttributeValue>();
key.put("Id", new AttributeValue().withN("101"));

GetItemRequest getItemRequest = new GetItemRequest().withTableName(tableName).withKey
(key);

GetItemResult result = client.getItem(getItemRequest);
Map<String, AttributeValue> map = result.getItem();
```

Atualizando um item:

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient(new ProfileCredentialsPro
vider());

HashMap<String, AttributeValue> key = new HashMap<String, AttributeValue>();
key.put("Id", new AttributeValue().withN("101"));

Map<String, AttributeValue> expressionAttributeValues = new HashMap<String,
AttributeValue>();
expressionAttributeValues.put(":val1", new AttributeValue().withSS("AuthorYY",
"AuthorZZ"));
expressionAttributeValues.put(":val2", new AttributeValue().withN("1"));

UpdateItemRequest updateItemRequest = new UpdateItemRequest()
        .withTableName(tableName)
        .withKey(key)
        .withReturnValues(ReturnValue.UPDATED_NEW)
        .withUpdateExpression("add Authors :val1 set Price = Price - :val2 remove ISBN")
        .withExpressionAttributeValues(expressionAttributeValues);

UpdateItemResult result = client.updateItem(updateItemRequest);
```

Removendo um item:

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient(new ProfileCredentialsPro
vider());

HashMap<String, AttributeValue> key = new HashMap<String, AttributeValue> ();
key.put("Id", new AttributeValue().withN("101"));
DeleteItemRequest deleteItemRequest = new DeleteItemRequest()
        .withTableName(tableName)
        .withKey(key);

DeleteItemResult deleteItemResult = client.deleteItem(deleteItemRequest);
```

1.5.2. Via console portal web

O console permite que seja feito o seguinte:

- Criar, atualizar e excluir tabelas.
- Ver os itens armazenados, adicionar, atualizar e excluir itens.
- Consultar uma tabela.
- Configurar alarmes para monitorar o uso da capacidade de uma tabela.
- Visualizar os alarmes configurados para cada tabela e criar alarmes personalizados.

1.6. Tarefas administrativas

O acesso ao DynamoDB requer credenciais que a AWS usará para autenticar as requisições, e essas credenciais devem ter permissões para acessar recursos da AWS, como uma tabela DynamoDB ou uma função da ***Amazon Web Service*** - **AWS**. É possível usar o Gerenciamento de Identificação e Acesso da AWS (*AWS Identity and Access Management* - IAM) e o DynamoDB para ajudar a proteger seus recursos controlando quem pode acessá-los.

- **Autenticação**

Quando um usuário se vincula à AWS, deve fornecer um endereço de e-mail e senha que são associados à sua conta AWS. Estas são credenciais com privilégios de administrador - *root* - e fornecem acesso completo a todos os recursos AWS.

Por razões de segurança, é recomendado que essas credenciais sejam usadas pela primeira vez apenas para criar um usuário administrador com permissões completas para a conta AWS, sendo possível criar outros usuários IAM e *roles* com permissões limitadas, por exemplo, ou criar um grupo de usuários, etc.

Um usuário IAM é simplesmente uma identidade dentro da conta AWS e tem permissões personalizadas específicas (por exemplo, a permissão para criar uma tabela). É possível usar um nome de usuário e senha IAM fazer login em páginas seguras da AWS, como o Console de Gestão AWS, os Fóruns de Discussão AWS, ou o Centro de Suporte AWS.

Pode ser gerada uma chave de acesso específica para cada usuário e autenticar as solicitações por meio de programação, permitindo um controle de

acesso mais refinado, quer através de um dos vários SDKs ou através do AWS *Command Line Interface* (CLI). Usando as chaves de acesso geradas, o SDK e ferramentas CLI criptograficamente "assinam" a requisição.

Por outro lado, ao invés de criar um usuário do IAM, é possível também usar identidades de usuários pré-existentes a partir do Diretório de Serviço da AWS, do diretório de usuários da empresa, ou de um provedor de identidade web. Estes são referidos como os **usuários federados**. Dizer que uma identidade é federada refere-se ao local onde o usuário armazena suas credenciais de acesso. Quando for realizado o acesso ao serviço, a própria AWS vai verificar se trata-se de um acesso confiável, autorizando-o.

Os usuários federados tem acesso aos serviços e recursos AWS através de um papel IAM, que é semelhante a um usuário IAM, mas não está associada a uma pessoa específica. Em vez disso, a função é atribuída a um usuário federado dinamicamente quando solicita acesso através de um provedor de identidade. As funções do IAM também pode ser usadas para outros fins, tais como a concessão de permissões para outra conta AWS acessar os recursos da conta do usuário atual. Um usuário federado é associado com um papel IAM que lhes permite obter chaves de acesso temporárias, que eles usam para autenticar requisições.

- **Controle de acesso**

É viável disponibilizar credenciais válidas para autenticacão, mas a menos que existam permissões, o usuário fica impossibilitado de criar ou acessar os recursos do DynamoDB. Por exemplo, o usuário deve ter permissões para criar uma tabela DynamoDB, ler e gravar dados em uma tabela, e permitir que uma função de AWS Lambda para processar registros de DynamoDB Streams.

Gerenciamento de permissões de acesso aos recursos

Cada AWS é gerenciado por uma conta AWS e as permissões para criar ou acessar um recurso são regidos pelas políticas de permissões. Um administrador de conta pode anexar permissões para identidades IAM (ou seja, usuários, grupos e funções), e alguns

serviços (tais como *AWS Lambda*) também suportam anexando políticas de permissões para recursos.

É preciso ressaltar que um administrador de conta (ou usuário administrador) é um usuário com privilégios de *root* e, ao conceder permissões, ele decide quem estará recebendo as permissões, os recursos que recebem permissões, e as ações específicas que deseja permitir a esses recursos

Exemplos de políticas baseadas em identidade onde um administrador de conta pode anexar políticas de permissões para identidades IAM - usuários, grupos e funções

Política de permissão baseada no IAM (quem tem acesso ao quê):

A regra tem uma declaração que concede permissões para três ações: `dynamoDB:DescribeTable`, `dynamoDB:Query` e `dynamoDB:Scan`) em uma tabela na região `us-west-2`, que é gerenciada pela conta AWS especificada pelo `account-id`. No *Amazon Resource Name* (ARN), o atributo `Resource` especifica a tabela à qual as permissões se aplicam:

```
{
    "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"
        }
    ]
}
```

O DynamoDB não suporta a Política de Permissão Baseada em Recursos.

Elementos da política de permissões: Ação, Efeito, Recurso

Para cada recurso do DynamoDB, o serviço define um conjunto de operações de API. Para conceder permissões para essas operações de API, o DynamoDB define um conjunto de ações que podem ser especificadas em uma política. Note-se que, realizando uma operação, a API pode exigir permissões para mais de uma ação. Ao conceder permissões para ações específicas, é preciso identificar também o recurso em que são permitidas ou negadas as ações.

A seguir estão os elementos básicos:

Recursos - Em uma política, é utilizado o ARN para identificar o recurso ao qual a política se aplica.

Ação - São usadas palavras-chave para identificar as operações de recursos que se deseja permitir ou negar. Por exemplo, `dynamodb:Query` autoriza as permissões de usuário para executar a operação `Query`.

Efeito - É especificado o efeito quando o usuário solicita uma ação específica, que pode ser permitida ou negada. Se não for concedido explicitamente a um recurso, o acesso é implicitamente negado. Também é possível negar explicitamente o acesso a um recurso, certificando-se de que um usuário não poderá acessá-lo, mesmo se for um acesso com *grant* diferente.

Permissões necessárias para utilizar o Console do DynamoDB

Para um usuário para trabalhar com o console DynamoDB, deve receber um conjunto mínimo de permissões, que permitirá trabalhar com os recursos DynamoDB para a sua conta de AWS. Além dessas permissões, o console requer permissões aos seguintes serviços:

- permissões Amazon *CloudWatch* para exibir métricas e gráficos.
- permissões AWS *Data Pipeline* para exportar e importar dados.
- permissões AWS *Identity and Access Management* para acessar as funções necessárias para as exportações e importações.
- permissões Amazon *Simple Notification Service* para notificá-lo sempre que um alarme CloudWatch é disparado.
- permissões AWS *Lambda* para processar registros do DynamoDB *Stream.*

Se for criada uma política de IAM que é mais restritiva do que as permissões mínimas exigidas, o console não funcionará como desejado para usuários criados sob a política de IAM. Para garantir que os usuários ainda podem usar o console DynamoDB, também é preciso vincular a regra `AmazonDynamoDBReadOnlyAccess` gestão política para o usuário, conforme descrito nas Políticas de Gerenciamento Pré-definidas - *AWS Managed (Predefined) Policies*.

Políticas de Gerenciamento Pré-definidas AWS

A AWS aborda muitos casos de uso comum, fornecendo políticas IAM autônomos que são criadas e administradas pela AWS. As políticas abaixo são específicas para o DynamoDB e são agrupados por cenário de caso de uso:

`AmazonDynamoDBReadOnlyAccess` - concede somente acesso *read-only* aos recursos do DynamoDB usando o *AWS Management Console*.

`AmazonDynamoDBFullAccess` - concede acesso completo aos recursos do DynamoDB usando o *AWS Management Console*.

`AmazonDynamoDBFullAccesswithDataPipeline` - concede acesso completo aos recursos do DynamoDB, incluindo exportação e importação usando o *AWS Data Pipeline*, usando o *AWS Management Console*.

O usuário também pode criar suas próprias políticas IAM personalizadas para permitir permissões para ações da API DynamoDB e recursos. Além disso, também pode anexar essas políticas personalizadas para os usuários IAM ou grupos que exigem tais permissões.

Exemplo 1: autorizar um usuário a executar qualquer ação em uma tabela

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllAPIActionsOnBooks",
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
        }
    ]
}
```

Exemplo 2: usuário com acesso *read-only* aos dados de uma tabela

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadOnlyAPIActionsOnBooks",
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:BatchGetItem"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
        }
    ]
}
```

Exemplo 3: autorizar as operações *Put*, *Update* e *Delete* em uma tabela específica

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PutUpdateDeleteOnBooks",
            "Effect": "Allow",
            "Action": [
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/Books"
        }
    ]
}
```

Exemplo 4: autorizar acesso a uma tabela específica e a todos os seus índices

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessAllIndexesOnBooks",
            "Effect": "Allow",
            "Action": [
                "dynamodb:*"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/Books",
                "arn:aws:dynamodb:us-west-2:123456789012:table/Books/index/*"
            ]
        }
    ]
}
```

Exemplo 5: configurar políticas de permissão para ambientes apartados de Produção e Teste

- Criar políticas em separado para cada usuário e anexar cada política a seu usuário específico. Por exemplo, você pode anexar a seguinte política ao usuário Alice para lhe permitir o acesso a todas as ações DynamoDB na tabela `Alice_ProductCatalog`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllAPIActionsOnAlicionsOnAliceTable",
            "Effect": "Allow",          ,
            "Action": [
                "dynamodb:*"
            ],
            "Resource": "arn:aws:dynamo aws:dynamodb:us-west-2:123456789012:table/Alice_ProductCatalog"
        }
    ]
}
```

- Ao invés de anexar políticas a usuários individualmente, é possível usar variáveis de políticas **IAM** para escrever uma única política e anexá-la a um grupo. Cria-se um grupo e, para este exemplo, são adicionado os usuários Alice e Bob.

  O exemplo a seguir concede permissões para executar todas as ações DynamoDB na tabela `${aws:username}_ProductCatalog`. A variável `${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 somente se Alice está adicionando dados à tabela `Alice_ProductCatalog`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllAPIActionsOnUserSpecificTable",
            "Effect": "Allow",
            "Action": [
                "dynamodb:*"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/${aws:username}_ProductCatalog"
        },
        {
            "Sid": "AdditionalPrivileges",
            "Effect": "Allow",
            "Action": [
                "dynamodb:ListTables",
                "dynamodb:DescribeTable",
                "cloudwatch:*",
                "sns:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Exemplo 6: impedir que um usuário aceite ofertas para a aquisição de espaço reservado

Controles de acesso:

- `dynamodb:DescribeReservedCapacity` – retorna as aquisições de espaço reservado ativas no momento.
- `dynamodb:DescribeReservedCapacityOfferings` – retorna detalhes sobre os planos de espaço reservado que estão sendo oferecidos no momento pela AWS.
- `dynamodb:PurchaseReservedCapacityOfferings` – efetua uma compra.

```
{
    "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:*"
        }
    ]
}
```

Exemplo 7: autorizar acesso de leitura somente ao DynamoDB Stream

Ações disponíveis para controle de acesso no *DynamoDB Streams*:

- `dynamodb:DescribeStream`
- `dynamodb:GetRecords`
- `dynamodb:GetShardIterator`
- `dynamodb:ListStream`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessGameScoresStreamOnly",
            "Effect": "Allow",
            "Action": [
                "dynamodb:DescribeStream",
                "dynamodb:GetRecords",
                "dynamodb:GetShardIterator",
                "dynamodb:ListStreams"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores/stream/*"
        }
    ]
}
```

Exemplo 8: permitir que a Função Lambda AWS processe os registros do DynamoDB Stream

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLambdaFunctionInvocation",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AllAPIAccessForDynamoDBStreams",
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetRecords",
                "dynamodb:GetShardIterator",
                "dynamodb:DescribeStream",
                "dynamodb:ListStreams"
            ],
            "Resource": "*"
        }
    ] d
}
```

2. Evidências

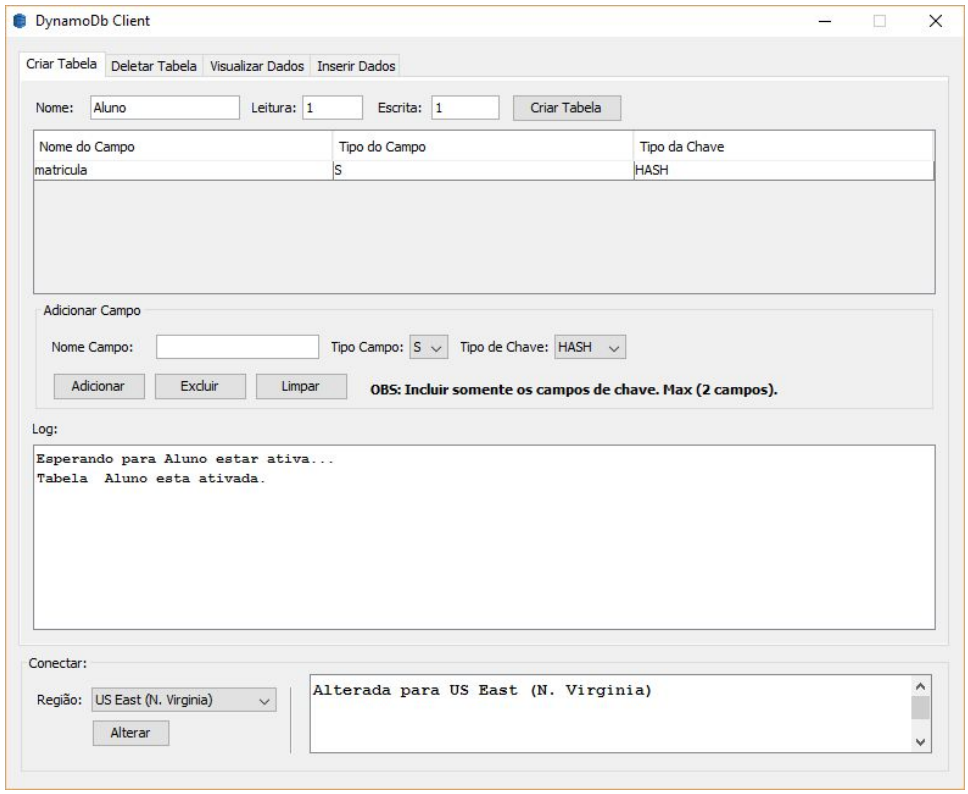
Foi criada uma aplicação Java utilizando o SDK do DynamoDB para demonstrar uma simulação de acesso utilizando o NetBeans 8.1.

Para a aplicação funcionar é necessário ter o arquivo com as credenciais de chave de acesso e a senha.

```
# Move this credentials file to (~/.aws/credentials)
# after you fill in your access and secret keys in the default
profile

# WARNING: To avoid accidental leakage of your credentials,
#          DO NOT keep this file in your source directory.
[default]
aws_access_key_id=AKIAIEVFDKHYEWDRISDO
aws_secret_access_key=PendQ3vakkBinKq0IClF1DNDCv6p7F2gRzPSF0ki
```

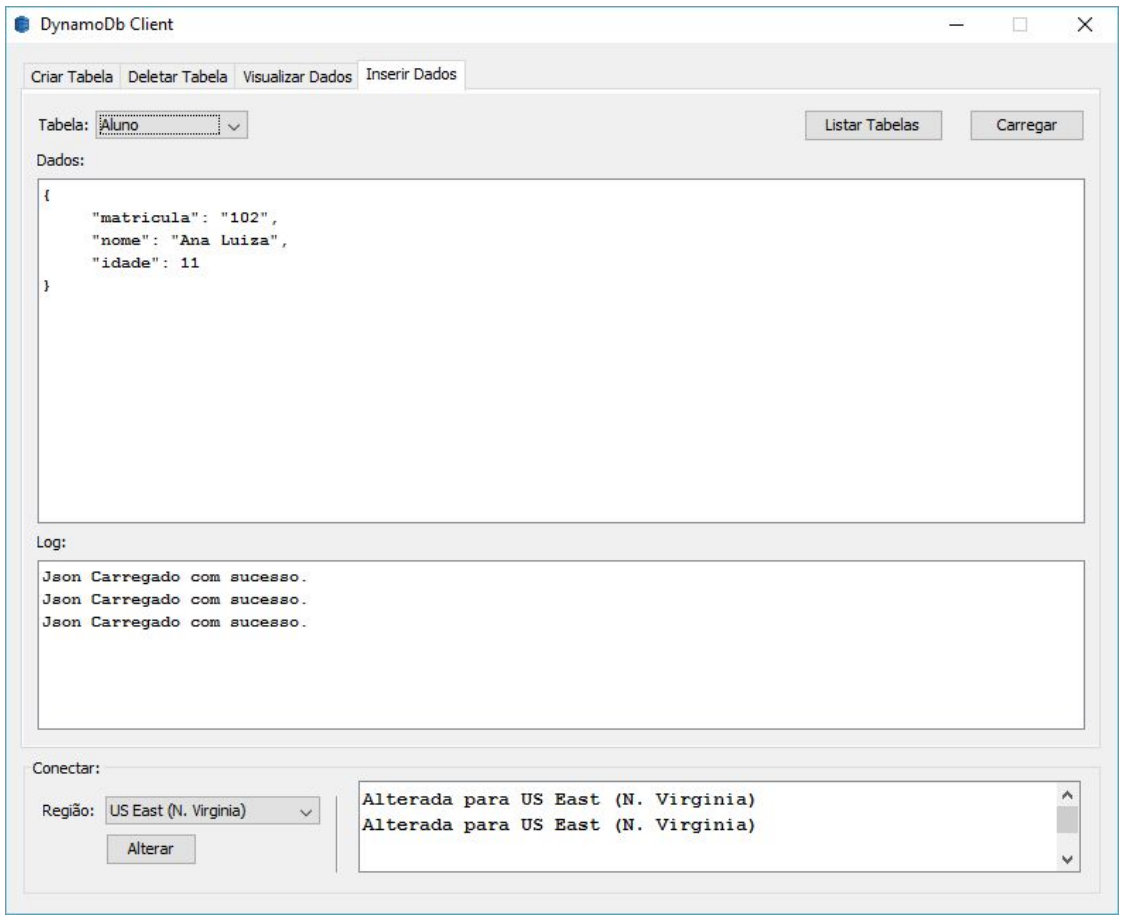
Na imagem abaixo, será gerada uma tabela chamada **Aluno** na região US East (N. Virginia) com o campo matricula como chave HASH.



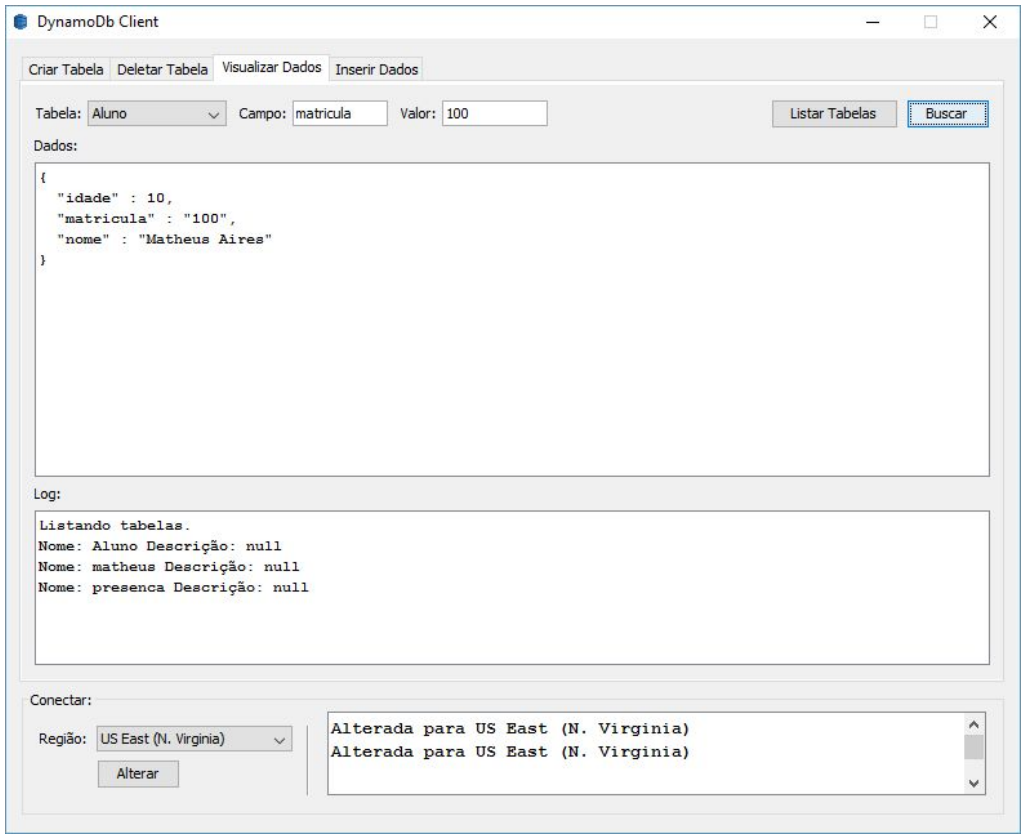
Na próxima etapa serão inseridos alguns itens na tabela **Aluno** com o JSON abaixo:

```
{
    "matricula": "100",
    "nome": "Matheus Aires",
    "idade": 10
}
{
    "matricula": "101",
    "nome": "Adriana",
    "idade": 12
}
{
    "matricula": "102",
    "nome": "Ana Luiza",
    "idade": 11
}
```

Pode ser verificado que o JSON foi carregado com sucesso.



A seguir, será verificado se o JSON está carregado na tabela **Aluno** utilizando o tributo matricula para encontrar o aluno com o valor “100”.



3. Conclusões

Com um banco de dados NoSQL, sem esquema predefinido (com exceção da chave) em que cada tabela é independente e suporta *queries* complexas, o DynamoDB se mostrou um banco fácil de se utilizar apesar de suas limitações, tais como tipos de dados limitados, não possuir *joins*, não ser possível copiar uma tabela para outra e possuir um limite de 64k por item.

Quanto ao SDK Java, os manuais fornecidos pela Amazon são bastante claros. Não foram encontradas grandes dificuldades para fazer uma conexão e executar operação simples de CRUD.

Sobre a administração, o DynamoDB possui regras claras e fáceis de aplicar, além de estar intimamente ligado às políticas de segurança da Amazon, especialmente o ***AWS Identity and Access Management*** **- IAM**,

onde o acrônimo permite a associação com a frase em inglês ***"I am"***, que significa ***"eu sou"*** em livre tradução para o Português. Foi possível concluir que a situação perfeita é utilizar o DynamoDB junto com a AWS, o que facilita muito o controle de acessos, gerenciamento de *roles* e perfis de usuários, além da segurança dos dados, reduzindo o trabalho de programação que o usuário terá para implementar seu próprio controle de acesso e autenticação.

4. Referências bibliográficas

Weblog Werner Vogels. Disponível em: http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html Acesso em 9 de maio de 2016.

Amazon DynamoDB Developer Guide. Disponível nos links a seguir:

- http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/UsingIAMWithDDB.htm
- http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/access-control-overview.htm
- http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/access-control-identity-based.htm
- http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/access-control-identity-based.html#access-policy-examples-aws-manage
- http://docs.aws.amazon.com/IAM/latest/UserGuide/introduction_access-management.html#intro-access-roles

Acesso em 25 de maio de 2016.

Information Security Stack Exchange. Disponível em http://security.stackexchange.com/questions/13803/what-is-single-sign-on-versus-federated-login . Acesso em 25 de maio de 2016.