

Bootcamp IGTI: Engenheiro(a) de Dados

Desafio

Módulo 3	Armazenamento de Dados
-----------------	-------------------------------

Objetivos

Exercitar os seguintes conceitos trabalhados no Módulo:

- ✓ Aplicar os conceitos de Computação em Nuvem (*Cloud Computing*), utilizando uma plataforma como serviço (*Platform as a Service – PaaS*) na Microsoft Azure.
- ✓ Aumentar o entendimento dos conceitos de banco de dados NoSQL, por meio do uso prático de um Sistema Gerenciador de Banco de Dados (SGBD) de mercado capaz de prover todos os tipos de dados NoSQL abordados ao longo da disciplina.
- ✓ Implementar Azure Cosmos DB baseadas em API SQL distribuído globalmente e utilizar linguagem similar à SQL para acesso aos dados semiestruturados.
- ✓ Implementar Azure Cosmos DB baseadas em API Grafo e utilizar a linguagem Gremlin para manipulação de dados.

Introdução

O Azure Cosmos DB é um banco de dados NoSQL totalmente gerenciado, oferecido pela Microsoft no formato de plataforma como serviço (PaaS) por meio de seu portal de nuvem Azure. Trata-se de um sistema de banco de dados distribuído globalmente, que permite ler e gravar os dados das réplicas locais do banco de dados. Ele replica de forma transparente os dados para todas as regiões associadas à conta do Cosmos sendo, portanto, um serviço de banco de dados distribuído globalmente, projetado para fornecer baixa latência, escalabilidade elástica de taxa de transferência, semântica bem definida para consistência de dados e alta disponibilidade.

Cada conta do Azure Cosmos DB é a unidade fundamental de distribuição global e alta disponibilidade sendo associada a um dos modelos de dados compatíveis. A API do SQL é o modelo de dados preferencial para a criação de um novo aplicativo. No entanto, ao se trabalhar grafos, tabelas ou em migração de dados do MongoDB ou do Cassandra para o Azure, contas adicionais podem ser criadas para esses modelos de dados específicos.

Um contêiner do Azure Cosmos é a unidade de escalabilidade para a taxa de transferência provisionada e para o armazenamento. Um contêiner é particionado horizontalmente e, em seguida, é replicado em várias regiões. Os itens adicionados ao contêiner são agrupados automaticamente em partições lógicas, que são distribuídas entre partições físicas, com base na chave de partição.

Ao criar-se um contêiner, configura-se a taxa de transferência em um dos seguintes modos:

- Modo de taxa de transferência provisionada dedicada: a taxa de transferência provisionada em um contêiner é reservada exclusivamente para esse contêiner e tem o suporte dos SLAs.
- Modo de taxa de transferência provisionada compartilhada: esses contêineres compartilham a taxa de transferência provisionada com outros contêineres no mesmo banco de dados (exceto pelos contêineres que foram configuradas com taxa de transferência provisionada dedicada). Em outras palavras, a taxa de transferência provisionada no banco de dados é compartilhada entre todos os contêineres "de taxa de transferência compartilhada".

O Azure Cosmos DB mede a taxa de transferência usando algo chamado RU (unidade de solicitação). O uso da unidade de solicitação é medido por segundo e, portanto, a unidade de medida é RU/s (unidades de solicitação por segundo). Você precisa reservar com antecedência o número de RU/s que deseja que o Azure Cosmos DB provisione, de modo que ele possa manipular a carga estimada e você possa aumentar ou reduzir as RU/s a qualquer momento para atender à demanda atual.

Uma chave de partição é o valor pelo qual o Azure organiza os dados em divisões lógicas. Ela deve ter como objetivo distribuir de maneira uniforme as operações no banco de dados para evitar partições ativas. Uma partição ativa é uma partição individual que recebe muito mais solicitações do que as outras, o que pode criar um gargalo de taxa de transferência.

A quantidade de RUs e armazenamento necessários determina o número de partições físicas necessárias para o contêiner, que são gerenciadas por completo pelo Azure Cosmos DB. Quando são necessárias partições físicas adicionais, o Cosmos DB cria essas partições automaticamente, dividindo as existentes. Não há nenhum impacto de tempo de inatividade ou de desempenho para o aplicativo.

Para cada contêiner do Azure Cosmos DB, deve-se especificar uma chave de partição que satisfaça as seguintes propriedades básicas:

- Tenha uma alta cardinalidade. Essa opção permite que os dados sejam distribuídos de maneira uniforme entre todas as partições físicas.
- Distribua as solicitações de maneira uniforme. Lembrando de que o número total de RU/s é dividido de maneira uniforme entre todas as partições físicas.
- Distribua o armazenamento de maneira uniforme. O tamanho de cada partição pode chegar a 20 GB.

Como se sabe, a consistência pode ser um problema quando se utiliza sistemas distribuídos. O Azure Cosmos DB melhora essa situação oferecendo cinco diferentes níveis de consistência de dados: forte (*strong*), desatualização limitada (*bounded staleness*), sessão (*session*), prefixo consistente (*consistent prefix*) e eventual (*eventual*). Esses níveis permitem maximizar a disponibilidade e o desempenho do banco de dados, dependendo das necessidades. Para casos em que os dados precisam ser processados em uma ordem específica, a consistência forte pode ser a escolha certa. Já para situações em que os dados não precisam ter uma coerência imediata, a coerência eventual pode ser uma melhor opção.

Todas as opções acima são compatíveis com uma abordagem de múltiplos modelos do Azure Cosmos DB, que fornece a capacidade de usar dados de documentos, chave-valor, família de colunas ou baseados em grafo.

O Core (SQL) é a API padrão do Azure Cosmos DB, que fornece uma exibição dos dados semelhante a um repositório tradicional de documentos NoSQL. É possível consultar os documentos JSON hierárquicos com uma linguagem semelhante ao SQL que fornece várias cláusulas conhecidas, como por exemplo: SELECT, FROM, WHERE, COUNT, SUM, MIN, MAX e ORDER BY.

A API do Azure Cosmos DB para MongoDB é compatível com o protocolo de transmissão do MongoDB. Essa API permite que os SDKs de cliente, os drivers e as ferramentas existentes do MongoDB interajam com os dados de forma transparente, como se estivessem sendo executados em um banco de dados real do MongoDB. Os dados são armazenados no formato de documento, que é o mesmo que usa o Core (SQL).

A compatibilidade do Azure Cosmos DB com a API do Cassandra possibilita a consulta de dados usando o CQL (Linguagem de Consulta do Cassandra), e seus dados parecerão um repositório de linhas particionadas. Assim como a API do MongoDB, os clientes ou as ferramentas conseguem se conectar de forma transparente ao Azure Cosmos DB; somente as configurações de conexão devem precisar ser atualizadas. O Azure Cosmos DB fornece várias cláusulas e instruções CQL familiares, como por exemplo: CREATE KEYSPACE, CREATE TABLE, INSERT, SELECT, UPDATE e DELETE.

A API de Tabela do Azure Cosmos DB é compatível com aplicativos escritos para o Azure Table Storage que precisam de funcionalidades como distribuição global, alta disponibilidade e taxa de transferência escalonável. O armazenamento de dados de tabela no Cosmos DB indexa automaticamente todas as propriedades e não requer gerenciamento de índice. A consulta é realizada usando consultas OData e LINQ no código e a API REST original para operações GET.

A escolha do Gremlin como a API, fornece uma exibição baseada em grafo dos dados, o que significa que os dados são compostos por vértices (item individual no banco de dados) e arestas (relação entre itens no banco de dados). O Azure Cosmos DB é compatível com a linguagem Gremlin do Apache Tinkerpop.

No que diz respeito à distribuição de dados, o suporte de vários mestres é uma opção que pode ser habilitada no Azure Cosmos DB. Depois que a conta é replicada em

várias regiões, cada região é uma região mestre que participa igualmente de um modelo de gravação em qualquer lugar, também conhecido como um padrão ativo-ativo. As regiões do Azure Cosmos DB que operam como regiões mestres em uma configuração de vários mestres funcionam automaticamente para convergir os dados gravados em todas as réplicas e garantir a integridade dos dados e a consistência global, sendo que os dados gravados são propagados para todas as outras regiões imediatamente.

Como benefício de múltiplos mestres, tem-se o seguinte:

- Latência de gravação de dígito único: as contas de vários mestres têm uma latência de gravação aprimorada de <10 ms para 99% das gravações, em comparação com <15 ms para contas que não têm vários mestres.
- Disponibilidade de leitura/gravação de 99,999%: a disponibilidade de gravação das contas de vários mestres aumenta para 99,999%, comparado a 99,99% das contas que não têm vários mestres.
- Escalabilidade de gravação e produtividade ilimitadas: com contas de vários mestres, você pode fazer gravações em todas as regiões, fornecendo escalabilidade de gravação e produtividade ilimitadas para dar suporte a bilhões de dispositivos.

Contudo, com a adição do suporte de vários mestres, surge a possibilidade de encontrar conflitos para gravações em regiões diferentes. Devido à rapidez com que a replicação ocorre globalmente, os conflitos tendem a ser raros e só podem ocorrer quando um item é alterado simultaneamente em várias regiões antes da propagação entre as regiões. Ainda assim, o Azure Cosmos DB fornece modos de resolução de conflitos que permite aos usuários decidir como lidar com cenários em que o mesmo registro é atualizado simultaneamente por autores diferentes em duas ou mais regiões. Há três modos de resolução de conflitos oferecidos pelo Azure Cosmos DB:

- LWW (último gravador vence): os conflitos são resolvidos com base no valor de uma propriedade de inteiro definida pelo usuário no documento. Por padrão, `_ts` é usado para determinar o último documento gravado. O último gravador vence é o mecanismo padrão de tratamento de conflitos.

- Personalizado: função definida pelo usuário, em que se pode controlar por completo a resolução de conflitos, registrando uma função definida pelo usuário na coleção.
- Personalizado Assíncrono: o Azure Cosmos DB exclui a confirmação de todos os conflitos e registra-os no feed de conflitos somente leitura para resolução adiada pelo aplicativo do usuário. O aplicativo pode executar a resolução de conflitos de forma assíncrona e usar qualquer lógica ou referenciar qualquer fonte externa, aplicativo ou serviço para resolver o conflito.

Enunciado

Após criar uma conta para acesso ao ambiente de computação em nuvem da Microsoft, podendo ser a mesma já utilizada para o trabalho prático, conecte-se ao portal da Azure para criar dois recursos de Cosmos DB (NoSQL), conforme o seguinte:

- ✓ Cosmos DB baseado em API SQL: API nativa do Azure Cosmos DB para trabalhar com documentos. Dá suporte ao desenvolvimento rápido e flexível com a linguagem de consulta SQL e as bibliotecas de clientes para .NET, JavaScript, Python e Java.
- ✓ Cosmos DB baseado em API Gremlin (Grafo): Serviço de banco de dados de grafo totalmente gerenciado que usa a linguagem de consulta Gremlin, com base no projeto Apache TinkerPop. Recomendado para novas cargas de trabalho que precisam armazenar relações entre os dados.

Atividades

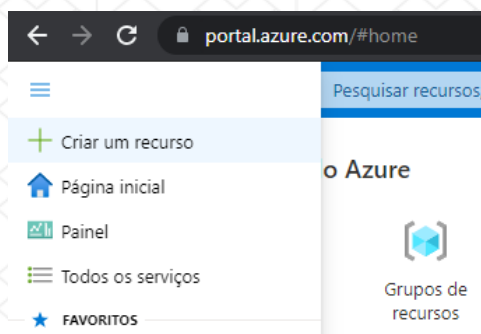
Os alunos deverão desempenhar as seguintes atividades:

1. Com base na demonstração e nas dicas repassadas na aula 3.1 do capítulo 3, Demonstração de SGBDR (Parte 1), crie sua conta para acesso à Microsoft Azure (<https://azure.microsoft.com>). Se isso já tiver sido realizado anteriormente, ignore essa atividade e passe para a seguinte.

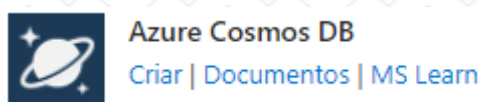
2. Seguindo os passos descritos a seguir, crie e manipule um recurso de Azure Cosmos DB baseado em API Core (SQL).

a. Entre em sua conta Azure.

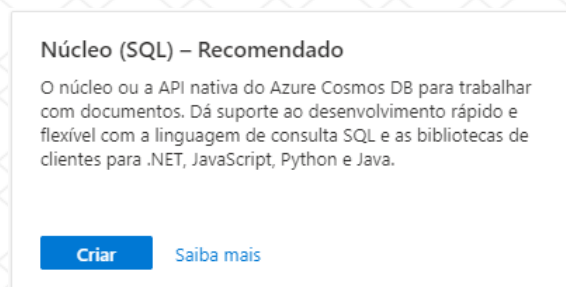
b. No menu do portal do Azure, selecione “Criar um recurso”.



c. Encontre e selecione Azure Cosmos DB.



d. Clique em “Criar” na caixa de Core (SQL).



e. Navegue por todas as abas de criação utilizando os seguintes valores (o que não for explicitamente citado deve ficar com o padrão):

- Grupo de recursos: Crie um grupo de recursos ou selecione um existente na sua assinatura.
- Nome da Conta: Escolha um nome exclusivo para a conta (por exemplo, cdbsqligti2021 + <seu_nome>, algo como cdbsqligti2021jose).
- Localidade: (US) Leste dos EUA.

- Gravações de Várias Regiões: Habilitar.
- Redundância do armazenamento de backup: Armazenamento de backup com redundância local.

f. Selecione “Criar”.

g. A criação da conta leva alguns minutos. Aguarde até o portal exibir a notificação de que a implantação foi bem-sucedida e selecione “Ir para o recurso”.

✓ A implantação foi concluída



Nome da implantação: Microsoft.Azure.CosmosDB-20210629034518

Assinatura: [Assinatura do Azure 1](#)

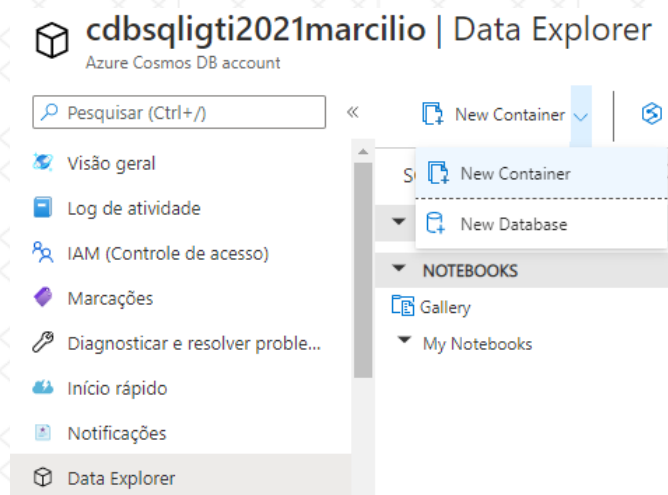
Grupo de recursos: [igti](#)

▼ Detalhes de implantação [\(Baixar\)](#)

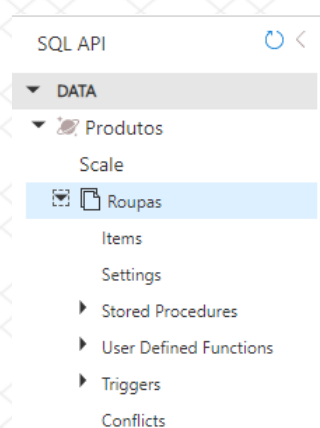
▲ Próximas etapas

[Ir para o recurso](#)

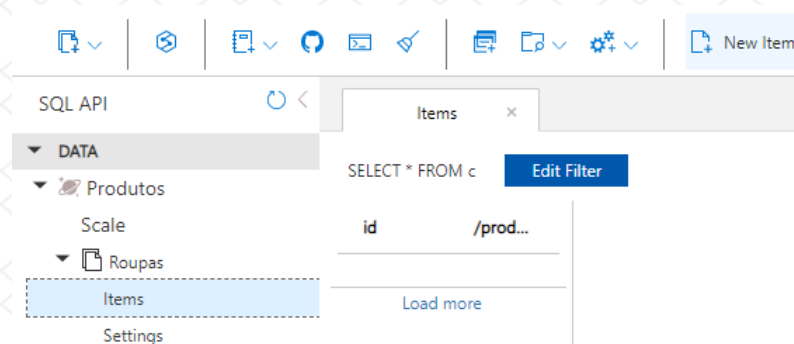
h. No menu à esquerda, clique em “Data Explorer” e, em seguida, no botão “New Container” na barra de ferramentas. O Data Explorer é uma ferramenta incluída no portal do Azure usada para gerenciar os dados armazenados em um Azure Cosmos DB. Ele fornece uma interface do usuário para navegar e exibir dados, consultar e modificá-los e criar e executar procedimentos armazenados. O Data Explorer é uma boa ferramenta para se familiarizar com os trabalhos internos e a funcionalidade oferecidos pelo Azure Cosmos DB.



- i. A área “New Container” é exibida mais à direita. Talvez você precise rolar a tela para a direita para vê-la. Nela, entre com as configurações a seguir e clique em “OK”:
 - Database id: Produtos.
 - Container id: Roupas.
 - Partition Keys: /productId.
- j. O Data Explorer exibirá o novo banco de dados Produtos e o contêiner Roupas.



- k. Para criar um documento JSON, expanda Roupas no painel da API do SQL, selecione Itens e escolha “New Item” na barra de ferramentas.



- l. Agora, adicione um documento ao contêiner. Cole o seguinte bloco JSON na guia Itens, substituindo o conteúdo atual:

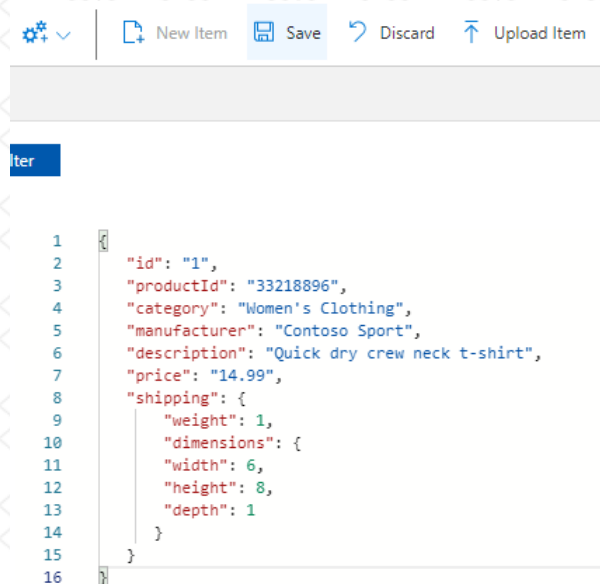
```
{
  "id": "1",
  "productId": "33218896",
```

```

    "category": "Women's Clothing",
    "manufacturer": "Contoso Sport",
    "description": "Quick dry crew neck t-shirt",
    "price": "14.99",
    "shipping": {
      "weight": 1,
      "dimensions": {
        "width": 6,
        "height": 8,
        "depth": 1
      }
    }
  }
}

```

m. Selecione “Save” na barra de ferramentas.



n. Selecione “New Item” novamente para criar e salvar outro documento, cole o objeto JSON a seguir no Data Explorer e salve o item.

```

{
  "id": "2",
  "productId": "33218897",
  "category": "Women's Outerwear",
  "manufacturer": "Contoso",
  "description": "Black wool pea-coat",
  "price": "49.99",
  "shipping": {
    "weight": 2,

```

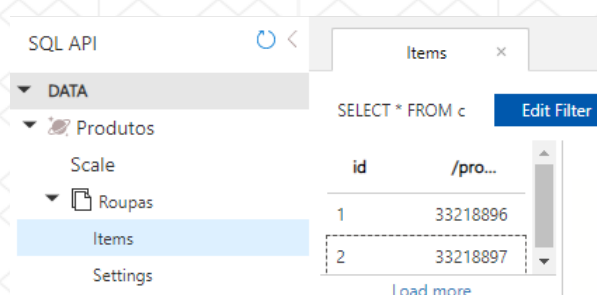


```

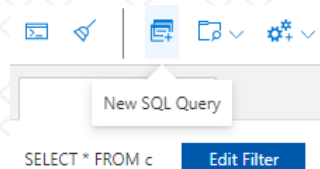
    "dimensions": {
      "width": 8,
      "height": 11,
      "depth": 3
    }
  }
}

```

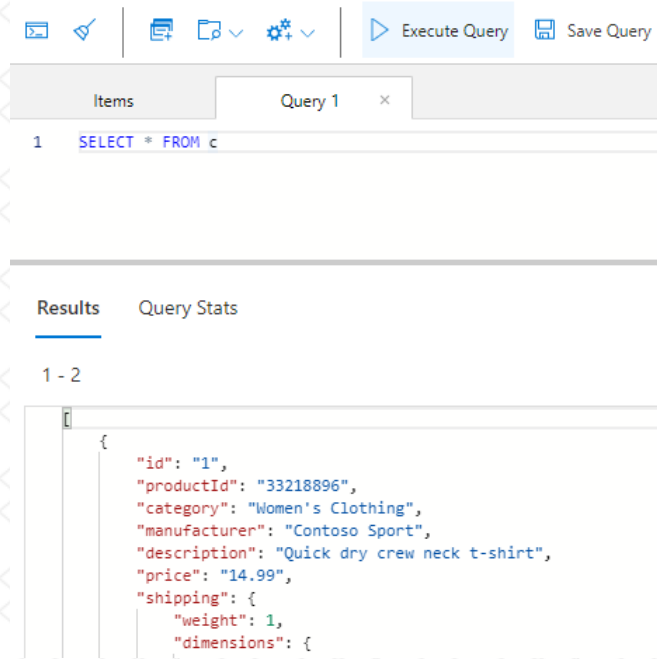
- o. Confirme se os documentos foram salvos clicando em “Itens” no menu à esquerda. Você deverá ver dois itens, conforme mostrado na captura de tela a seguir.



- p. No Data Explorer, selecione “New SQL Query” na barra de ferramentas, conforme mostrado na imagem a seguir.



- q. A consulta padrão na nova guia “Query 1” é “SELECT * FROM c”. Essa consulta retorna todos os resultados no banco de dados. Selecione “Execute Query” para executar a instrução SQL.



- r. Substitua o “SELECT * FROM c” e cole a seguinte consulta na caixa de texto:

```
SELECT *
FROM Products p
WHERE p.id = "1"
```

- s. Selecione “Execute Query” para executar o SQL modificado. Os resultados retornam o produto cuja id é 1 conforme trecho mostrado abaixo:

```
[
  {
    "id": "1",
    "productId": "33218896",
    "category": "Women's Clothing",
    "manufacturer": "Contoso Sport",
    "description": "Quick dry crew neck t-shirt",
    "price": "14.99",
    "shipping": {
      "weight": 1,
      "dimensions": {
        "width": 6,
        "height": 8,
        "depth": 1
      }
    }
  },
]
```



```
"_rid": "9T5OALXVGr0BAAAAAAAAA==",  
"_self":
```

```
...
```



- t. Substitua a consulta pelo texto a seguir e clique em “Execute Query”. Essa consulta retorna o preço, a descrição e a ID do produto para todos os produtos, ordenados por preço em ordem crescente.


```
SELECT p.price, p.description, p.productId  
FROM Products p  
ORDER BY p.price ASC
```

- u. O painel de saída deve mostrar resultados de maneira semelhante a:

```
[  
  {  
    "price": "14.99",  
    "description": "Quick dry crew neck t-shirt",  
    "productId": "33218896"  
  },  
  {  
    "price": "49.99",  
    "description": "Black wool pea-coat",  
    "productId": "33218897"  
  }  
]
```

- v. Teste realizar novas consultas para familiarizar-se ainda mais com a interface e linguagem de consulta.
- w. Para que os dados estejam replicados entre regiões, no menu à esquerda, clique em “Replicar dados globalmente”. Em seguida, no mapa, selecione “Brazil South” e clique em “Salvar”.

 Salvar
  Descartar
 

 Ao adicionar uma região à sua com 400 RU/s e 5 GB de arma

Clique em uma localização para


* Cada região terá uma cobrança b
mais

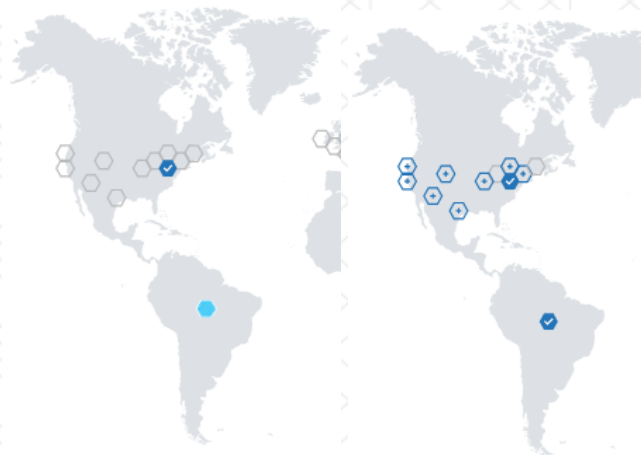


Configure as regiões para leituras, gravações e zona de disponibilidade (com suporte nas regiões selecionadas e pode ser configurado apenas quando uma nova região for adicionada). [+ Adicionar região](#)

Regiões	Leituras Habilitadas	Gravações Habilitadas	Zona de Disponibilid...	Ação
East US	✓	✓		
Brazil South	✓	✓	<input type="checkbox"/>	

- x. A página exibirá uma mensagem de “Atualizando” enquanto os dados são gravados para a nova região, o que deve ocorrer em até 30 minutos.

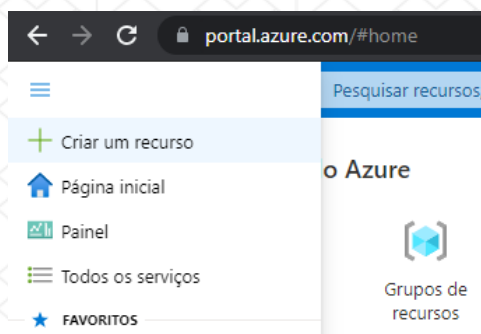
 Atualizando



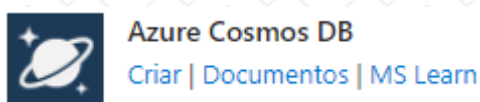
3. Seguindo os passos descritos a seguir, crie e manipule um recurso de Azure Cosmos DB baseado em API Gremlin (Grafo).

a. Entre em sua conta Azure.

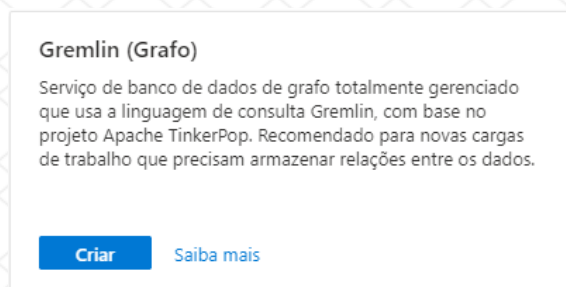
b. No menu do portal do Azure, selecione “Criar um recurso”.



c. Encontre e selecione Azure Cosmos DB.



d. Clique em “Criar” na caixa de Core (SQL).



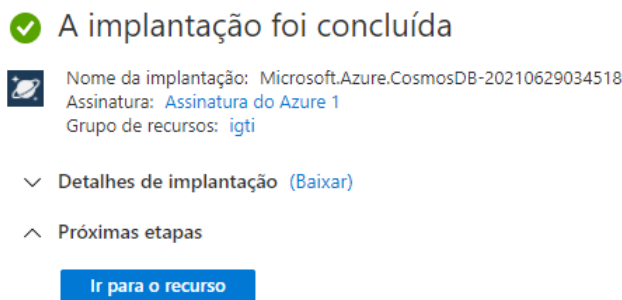
e. Navegue por todas as abas de criação utilizando os seguintes valores (o que não for explicitamente citado deve ficar com o padrão):

- Grupo de recursos: Crie um grupo de recursos ou selecione um existente na sua assinatura.
- Nome da Conta: Escolha um nome exclusivo para a conta (por exemplo, cdbgraphigti2021 + <seu_nome>, algo como cdbgraphigti2021jose).
- Localidade: (US) Leste dos EUA.

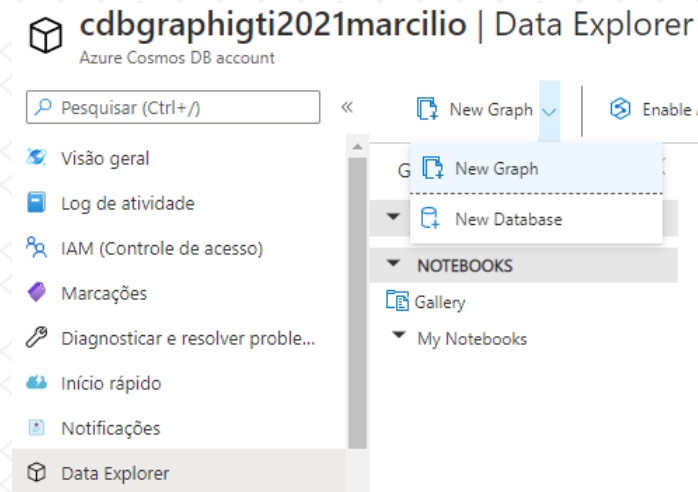
- Redundância do armazenamento de backup: Armazenamento de backup com redundância local.

f. Selecione “Criar”.

g. A criação da conta leva alguns minutos. Aguarde até o portal exibir a notificação de que a implantação foi bem-sucedida e selecione “Ir para o recurso”.



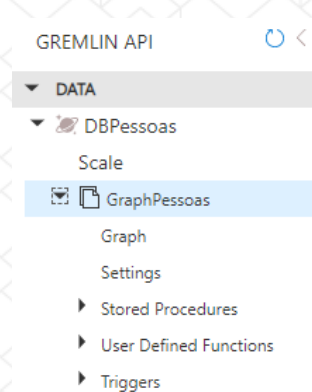
h. No menu à esquerda, clique em “Data Explorer” e, em seguida, no botão “New Graph” na barra de ferramentas.



i. A área “New Graph” é exibida mais à direita. Talvez você precise rolar a tela para a direita para vê-la. Nela, entre com as configurações a seguir e clique em “OK”:

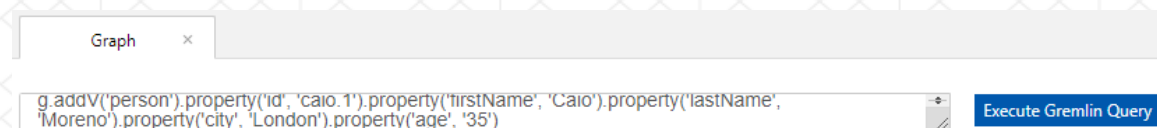
- Database id: DBPessoas.
- Container id: GraphPessoas.
- Partition Keys: /city.

- j. O Data Explorer exibirá o novo banco de dados DBPessoas e o grafo GraphPessoas.

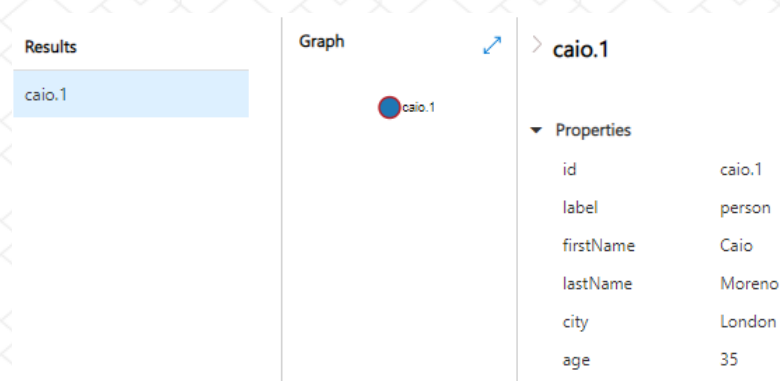


- k. Para adicionar novos vértices e arestas, expanda GraphPessoas no painel da API do Gremlin, selecione Graph, cole o comando abaixo e clique em “Execute Gremlin Query”.

```
g.addV('person').property('id', 'caio.1').property('firstName', 'Caio').property('lastName', 'Moreno').property('city', 'London').property('age', '35')
```



- l. O resultado deve ser o seguinte:



- m. Insira agora os dados de Ana Eliza:

```
g.addV('person').property('id', 'anaeliza.1').property('firstName', 'Ana Eliza').property('lastName', 'Motta').property('city', 'London').property('age', '5')
```

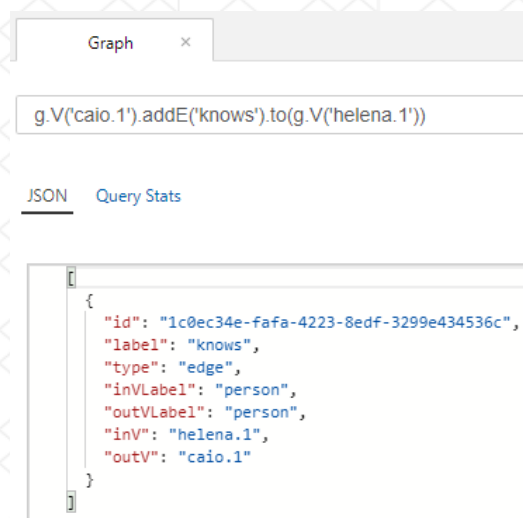

n. Insira agora os dados de Helena:

```
g.addV('person').property('id', 'helena.1').property('firstName',  
'Helena').property('lastName', 'Motta').property('city',  
'London').property('age', '2')
```

o. Agora adicione uma aresta que representa o relacionamento entre Caio e Helena:

```
g.V('caio.1').addE('knows').to(g.V('helena.1'))
```

p. O resultado deve ser similar ao seguinte:



q. Agora o relacionamento entre Caio e Ana Eliza.

```
g.V('caio.1').addE('knows').to(g.V('anaeliza.1'))
```

r. Exiba os vértices já incluídos executando o seguinte comando:

```
g.V()
```

Graph

g.V()

JSON

Graph

Query Stats

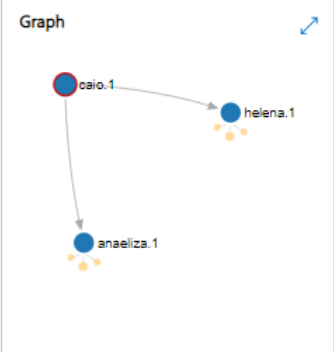
Results

caio.1

anaeliza.1

helena.1

Graph



> caio.1

Properties

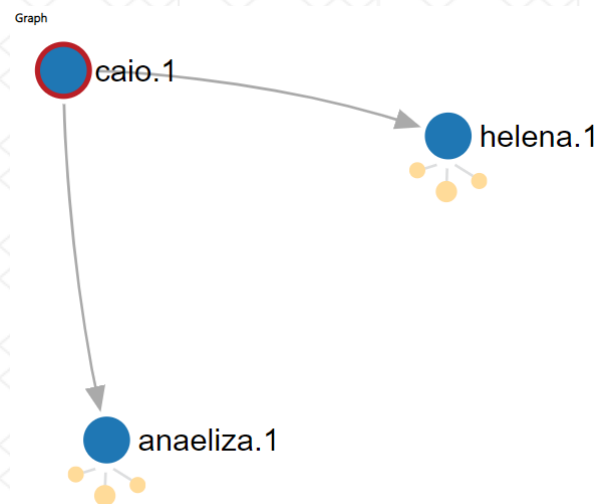
id	caio.1
label	person
firstName	Caio
lastName	Moreno
city	London
age	35

s. Para uma melhor visualização do grafo, clique no ícone.

Graph



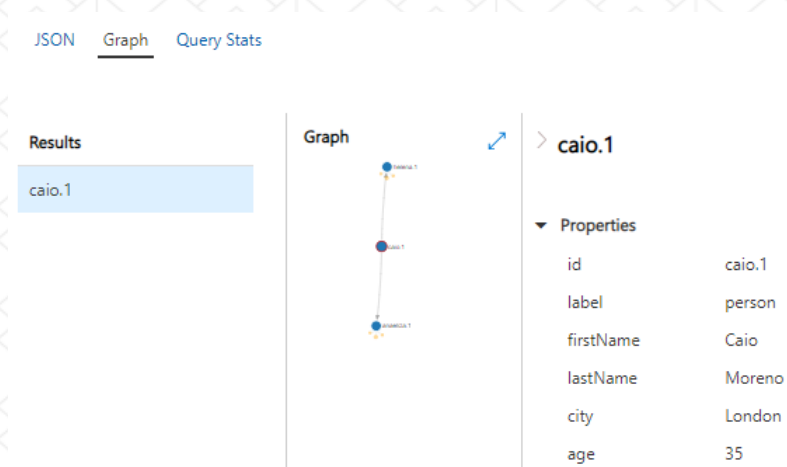
t. Isso permite uma exibição completa do grafo conforme o seguinte:



u. Busque todas as pessoas que têm idade de 35 anos com uso da seguinte consulta:

```
g.V().hasLabel('person').has('age', '35')
```

v. O resultado deve exibir Caio:



w. Teste realizar a inserção de novos dados e novas buscas para familiarizar-se ainda mais com a interface.

Obs.: Baseado no conteúdo disponível em:

ANDERSON, Christopher et al. *Início Rápido: Criar, consultar e percorrer um banco de dados de grafo do Azure Cosmos DB usando o console do Gremlin*. Microsoft, jul. 2020. Disponível em: <<https://docs.microsoft.com/pt-br/azure/cosmos-db/create-graph-gremlin-console>>. Acesso em: 29 jun. 2021.

BROWN, Mark et al. *Distribuir dados globalmente com o Azure Cosmos DB*. Microsoft, jan. 2021. Disponível em: <<https://docs.microsoft.com/pt-br/azure/cosmos-db/distribute-data-globally>>. Acesso em: 29 jun. 2021.

BROWN, Mark; OLPROD. *Modelo de recurso do Azure Cosmos DB*. Microsoft, out. 2020. Disponível em: <<https://docs.microsoft.com/pt-br/azure/cosmos-db/account-databases-containers-items>>. Acesso em: 29 jun. 2021.

MICROSOFT. *Trabalhar com os dados NoSQL no Azure Cosmos DB*. Disponível em: <<https://docs.microsoft.com/pt-br/learn/paths/work-with-nosql-data-in-azure-cosmos-db/>>. Acesso em: 29 jun. 2021.

MORENO, Caio. *Graph database using Cosmos DB Gremlin API*. Medium, out. 2018. Disponível em: <<https://caiomsouza.medium.com/graph-database-using-cosmos-db-gremlin-api-ef0b6e0e517c>>. Acesso em: 29 jun. 2021.

