📖Capítulo 7 – DynamoDB

Dia 1 – Let’s Go Shopping!

No primeiro dia com o **DynamoDB**, vamos entender como criar tabelas, inserir itens e realizar operações básicas de **CRUD**. O exemplo segue o contexto de um **e-commerce**, simulando carrinhos de compras e pedidos.

🗂️Criando uma Tabela

No DynamoDB, cada tabela precisa de uma **partition key** (obrigatória) e pode opcionalmente ter uma **sort key**.

Exemplo: criar uma tabela chamada `Pedidos`. - **Partition Key**: `UserId` (identifica o usuário). - **Sort Key**: `OrderId` (identifica pedidos diferentes do mesmo usuário).

```
aws dynamodb create-table
  --table-name Pedidos
  --attribute-definitions
      AttributeName=UserId,AttributeType=S
      AttributeName=OrderId,AttributeType=S
  --key-schema
      AttributeName=UserId,KeyType=HASH
      AttributeName=OrderId,KeyType=RANGE
  --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5
```

📌Aqui usamos a CLI da AWS. `S` indica atributo do tipo *String*.

Inserindo Itens (Create)

Cada item é um documento JSON.

```
aws dynamodb put-item
  --table-name Pedidos
  --item '{
    "UserId": {"S": "U123"},
    "OrderId": {"S": "A001"},
    "Data": {"S": "2025-08-29"},
    "Itens": {"L": [
      {"M": {"Produto": {"S": "Notebook"}, "Preco": {"N": "4500"}}},
      {"M": {"Produto": {"S": "Mouse"}, "Preco": {"N": "120"}}}
```

```
    ]}
  }'
```

📌`L` representa uma lista e `M` um mapa (objeto JSON).

📖Lendo Itens (Read)

- Buscar um pedido específico:

```
aws dynamodb get-item
  --table-name Pedidos
  --key '{"UserId": {"S": "U123"}, "OrderId": {"S": "A001"}}'
```

- Buscar todos os pedidos de um usuário (usando **Query**):

```
aws dynamodb query
  --table-name Pedidos
  --key-condition-expression "UserId = :uid"
  --expression-attribute-values '{":uid":{"S":"U123"}}'
```

- Buscar todos os pedidos (usando **Scan** – menos eficiente):

```
aws dynamodb scan --table-name Pedidos
```

✏️Atualizando Itens (Update)

```
aws dynamodb update-item
  --table-name Pedidos
  --key '{"UserId": {"S": "U123"}, "OrderId": {"S": "A001"}}'
  --update-expression "SET Status = :s"
  --expression-attribute-values '{":s":{"S":"Pago"}}'
```

📌DynamoDB usa `update-expression` para modificar atributos sem sobrescrever o item inteiro.

🗑️Deletando Itens (Delete)

```
aws dynamodb delete-item
  --table-name Pedidos
  --key '{"UserId": {"S": "U123"}, "OrderId": {"S": "A001"}}'
```

🔑Importância da Modelagem de Chaves

- **Partition Key** → deve ser bem distribuída para evitar *hotspots*.
- **Sort Key** → permite ordenar e filtrar dados dentro da mesma partição.
- Consultas eficientes dependem de **modelagem correta das chaves**.

Exemplo ruim: usar apenas `Data` como partition key (muitos pedidos no mesmo dia ficam na mesma partição). Exemplo bom: `UserId` como partition key + `OrderId` como sort key.

🔗Resumo do Dia 1

- Criamos uma tabela no DynamoDB (`Pedidos`).
- Inserimos itens (JSON aninhado com listas e mapas).
- Consultamos dados com **GetItem**, **Query** e **Scan**.
- Atualizamos atributos com `update-expression`.
- Deletamos itens específicos.
- Vimos que a **modelagem de chaves** é a base para eficiência.

🔔 No próximo passo: **Dia 2 – Building a Streaming Data Pipeline**, onde exploraremos como o DynamoDB se integra com outros serviços da AWS para processar dados em tempo real.