Teste Técnico para Engenharia de Dados – Árvore

Introdução

Como desenvolver um pipeline de ingestão de dados que integre dados de um banco de dados relacional para o Amazon Redshift, utilizando o Apache Airflow ou uma ferramenta similar? Esse é o desafio proposto pela Árvore para a vaga de Engenharia de Dados. Neste documento, eu apresento a solução que eu desenvolvi para esse desafio, explicando os passos executados, as ferramentas utilizadas e os resultados obtidos.

Para realizar esse teste técnico, eu utilizei as seguintes plataformas de nuvem: Azure, Google Cloud e Amazon Redshift. A escolha dessas plataformas se deu pelos seguintes motivos:

- Azure: Eu escolhi o My SQL hospedado num servidor Azure como o banco de dados relacional de origem dos dados, pois ele oferece uma alta disponibilidade, segurança e escalabilidade para armazenar e gerenciar dados estruturados.

- Google Cloud: Eu escolhi o Google Cloud como a plataforma de nuvem para hospedar e executar o pipeline de dados, pois ele possui uma integração nativa com o Apache Airflow, uma ferramenta de orquestração de workflows que permite automatizar e monitorar as tarefas de ingestão de dados.

- Amazon Redshift: Eu escolhi o Amazon Redshift como o destino dos dados, pois ele é um serviço de armazenamento de dados em colunas, otimizado para análises de grandes volumes de dados, com alta performance e baixo custo.

Configuração do Banco de Dados MySQL

O MySQL é um banco de dados relacional que oferece alta disponibilidade, segurança e escalabilidade.

Neste tutorial, vamos configurar um banco de dados MySQL como um banco de dados único no Azure, usando os seguintes passos:

1. **Criação do grupo de recursos Arvore** em assinatura própria do Azure na localidade Brazil South. Um grupo de recursos é um contêiner lógico que agrupa os recursos relacionados a uma solução. Escolhemos a localidade Brazil South para minimizar a latência e os custos de rede.

2. **Criação de servidor flexível** em assinatura própria do Azure de nome my-sql, na região Brazil Southeast, versão do MySQL 8.0, com tipo de carga de trabalho para projetos de desenvolvimento ou hobby, tendo como nome de usuário administrador my_sql e senha Admin123.

3. **Criação de estrutura de tabelas para o projeto**: Foi criado um banco de dados de exemplo chamado **classicmodels** disponível em MySQL Sample Database (mysqltutorial.org).

Configuração do Amazon Redshift

O Amazon Redshift é um serviço de armazenamento de dados em nuvem que permite executar consultas analíticas complexas e rápidas em grandes volumes de dados1. Neste tutorial, vamos configurar uma instância do Amazon Redshift para servir como o repositório final dos dados, usando os seguintes passos:

1. Foi criado um cluster chamado **redshift-cluster-arvore** utilizando a avaliação gratuita. Um cluster é um conjunto de nós que executam o banco de dados do Amazon Redshift. O nome de usuário administrador foi configurado como awssuser e a senha como Awsuser123. Essas credenciais são usadas para acessar o console do Amazon Redshift e gerenciar o cluster.

2. No cluster redshift-cluster-arvore, foi criado um usuário chamado **redshift** com a senha Redshift123. Um usuário é uma conta de banco de dados que pode se conectar ao cluster e executar operações no banco de dados. Para criar um usuário, usamos o seguinte comando SQL:

```
CREATE USER redshift PASSWORD 'Redshift123';
```

3. No cluster redshift-cluster-arvore, foi criado um banco de dados chamado **arvore** e foram concedidas permissões ao usuário redshift. Um banco de dados é um contêiner lógico que armazena os dados e os objetos do banco de dados, como tabelas, visões e funções. Para criar um banco de dados, usamos o seguinte comando SQL:

```
CREATE DATABASE arvore;
```

Para conceder permissões ao usuário redshift, usamos o seguinte comando SQL:

```
GRANT ALL ON DATABASE arvore TO redshift;
```

Esse comando concede ao usuário redshift todos os privilégios no banco de dados arvore, incluindo criar, modificar e excluir objetos do banco de dados.

4. No banco de dados arvore, foi criado um esquema chamado **dados** e foram concedidos privilégios no mesmo para o usuário u_arvore. Um esquema é um espaço de nomes que agrupa os objetos do banco de dados relacionados, como tabelas e visões. Para criar um esquema, usamos o seguinte comando SQL:

```
CREATE SCHEMA dados;
```

Para conceder privilégios no esquema dados para o usuário u_arvore, usamos os seguintes comandos SQL:

```
GRANT USAGE, CREATE ON SCHEMA dados TO redshift;
GRANT SELECT, INSERT, DELETE, UPDATE ON ALL TABLES IN SCHEMA dados TO
redshift;
```

O primeiro comando concede ao usuário redshift os privilégios de uso e criação no esquema dados, ou seja, o usuário pode acessar e criar objetos no esquema. O segundo comando concede ao usuário redshift os privilégios de seleção, inserção, exclusão e atualização em todas as tabelas do esquema dados, ou seja, o usuário pode consultar e modificar os dados nas tabelas.

Configuração da Plataforma Cloud

O Google Cloud é uma plataforma de computação em nuvem que oferece diversos serviços e recursos para desenvolver, executar e gerenciar aplicações.

Neste teste técnico, vamos configurar a plataforma Google Cloud para realizar a sincronização de dados entre dois bancos de dados diferentes: o Azure SQL Database e o Amazon Redshift. Para esta configuração, foram realizados os seguintes passos:

1. Foi criado um ambiente chamado **ambiente-arvore** no local southamerica-east1 no Cloud Composer. O Cloud Composer é um serviço gerenciado do Google Cloud que permite criar, programar e monitorar fluxos de trabalho usando o Apache Airflow.

2. Um ambiente é uma instância do Cloud Composer que contém todos os recursos necessários para executar os fluxos de trabalho, como um cluster do Kubernetes, um bucket do Cloud Storage e um banco de dados do Cloud SQL. O nome do ambiente é uma forma de identificá-lo dentro do projeto do Google Cloud. O local é a região onde o ambiente é hospedado. Neste caso, escolhemos a região southamerica-east1, que corresponde à América do Sul (São Paulo).

3. Foi criado o arquivo **requirements.txt** para incluir as bibliotecas psycopg2 e mysql-connector-python. O arquivo requirements.txt é um arquivo de texto que lista as bibliotecas Python que são necessárias para executar os fluxos de trabalho do Cloud Composer. As bibliotecas são instaladas automaticamente pelo Cloud Composer quando o ambiente é criado ou atualizado. Neste caso,

incluímos as bibliotecas psycopg2 e mysql-connector-python, que são usadas para conectar e interagir com os bancos de dados PostgreSQL e Azure, respectivamente .

4. Foi criado o script de pipeline **arvore.py** que gerou a DAG arvore_dag. Este script sincroniza tabelas de um esquema de banco de dados de origem para o destino. Uma DAG (Directed Acyclic Graph) é uma representação gráfica dos fluxos de trabalho do Cloud Composer, que consiste em um conjunto de tarefas e suas dependências. O script arvore.py define a DAG arvore_dag, que contém as tarefas de decisao, carga_total e carga_incremental. O funcionamento do script é descrito a seguir:

4.1. A função **decisao** é o ponto de partida do script. Ela lista todas as tabelas do banco de dados de origem (Azure SQL) e verifica a existência das mesmas no banco de dados de destino (Amazon Redshift). Se uma tabela existir no Azure SQL, mas não no Redshift, a função carga_total é chamada para a respectiva tabela. Se a tabela já existir no Redshift, a função carga_incremental é chamada para a respectiva tabela.

4.2. A função **carga_total** é responsável por criar e preencher a tabela no Amazon Redshift com os dados provenientes do Azure SQL. Essa função recebe como parâmetro o nome da tabela que deve ser sincronizada entre os dois bancos de dados. Primeiro, ela verifica se a tabela já existe no Amazon Redshift. Se não existir, ela cria a tabela com a mesma estrutura da tabela do Azure SQL, usando o comando CREATE TABLE. Esse comando define o nome, o tipo de dados e as restrições de cada coluna da tabela. Em seguida, ela realiza a cópia integral dos dados da tabela de origem para a tabela de destino, usando o comando INSERT INTO. Esse comando insere os valores de cada registro da tabela do Azure SQL na tabela correspondente do Amazon Redshift. Dessa forma, a função garante que a tabela de destino seja criada e preenchida com todos os dados da tabela de origem. Essa função é executada apenas uma vez para cada tabela, quando ela não existe ou está vazia no Amazon Redshift.

4.3. A função **carga_incremental** é responsável por atualizar a tabela no Amazon Redshift com os dados mais recentes do Azure SQL. Essa função recebe como parâmetro o nome da tabela que deve ser sincronizada entre os dois bancos de dados. Primeiro, ela consulta a hora da última carga realizada na tabela de destino, usando o comando SELECT. Esse comando retorna o valor da coluna que armazena a data e a hora da última modificação dos registros na tabela. Em seguida, ela compara esse valor com a data e a hora dos registros na tabela de origem, usando o comando WHERE. Esse comando filtra os registros que foram inseridos ou atualizados após a última carga na tabela de destino.

Depois, ela exclui os registros que já existem na tabela de destino, mas que foram alterados na tabela de origem, usando o comando DELETE. Esse comando remove os registros que possuem o mesmo identificador (id) na tabela de origem e na tabela de destino. Por fim, ela insere todos os registros filtrados na tabela de destino, usando o comando INSERT. Esse comando adiciona os valores de cada registro da tabela de origem na tabela correspondente do Amazon Redshift. Dessa forma, a função garante que a tabela de destino reflita as alterações mais recentes da tabela de origem. Essa função é executada periodicamente para cada tabela, quando ela já existe e contém dados no Amazon Redshift.