Teste Técnico para Engenharia de Dados – Árvore

Este documento descreve os passos executados para a conclusão do teste técnico, que envolve a configuração de um banco de dados MySQL como origem dos dados, a configuração do Amazon Redshift como destino dos dados e a configuração do Google Cloud como a plataforma de nuvem para executar o pipeline de dados.

1. Configuração do Banco de Dados de Origem

Um banco de dados MySQL foi configurado em uma máquina virtual no Azure. Os passos para a configuração são os seguintes:

1.1. Criação de uma máquina virtual chamada `vm-mysql` com a imagem Ubuntu Server 20.04 LTS. A conta de administrador foi configurada com o usuário `administrador` e a senha `Administrador123`.

1.2. A porta 3306 foi aberta para tráfego TCP na máquina virtual `vm-mysql`.

1.3. O MySQL foi instalado na máquina virtual `vm-mysql` usando os seguintes comandos:

```
sudo apt-get update
sudo apt-get install mysql-server
```

1.4. A conexão externa no MySQL foi liberada editando o arquivo `/etc/mysql/mysql.conf.d/mysqld.cnf` e alterando o parâmetro `bind_address` para `0.0.0.0`.

1.5. Os objetos do banco de dados foram criados para gerar os dados que servirão de origem para este teste. Os comandos usados foram:

```
sudo mysql -u root -p

CREATE DATABASE arvore;
USE arvore
CREATE USER 'u_arvore'@'%' IDENTIFIED BY 'u_arvore';
GRANT ALL PRIVILEGES ON arvore.* TO 'u_arvore'@'%';
FLUSH PRIVILEGES;
CREATE TABLE `people` (
`id` mediumint(8) unsigned NOT NULL auto_increment,
`name` varchar(255) default NULL,
`email` varchar(255) default NULL,
`phone` varchar(100) default NULL,
`address` varchar(255) default NULL,
`postalZip` varchar(10) default NULL,
`region` varchar(50) default NULL,
`country` varchar(100) default NULL,
`created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
`updated_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
```

```
PRIMARY KEY (`id`)
) AUTO_INCREMENT=1;
INSERT INTO `people`
(`name`,`email`,`phone`,`address`,`postalZip`,`region`,`country`)
VALUES
("Latifah Mcmahon","mauris.erat.eget@hotmail.org","1-857-615-
0684","P.O. Box 234, 414 Nisl. Avenue","V4U 0MK","Puno","Colombia"),
("Leah Terry","vel@outlook.couk","1-549-167-1615","Ap #445-7064
Praesent Av.","1744","New South Wales","Costa Rica"),
("Alea Wong","cum.sociis@google.net","(161) 380-7391","P.O. Box 853,
2109 Congue. Avenue","587389","San José","Ukraine"),
("Maggie Barlow","enim.sit@outlook.org","(487) 432-8190","P.O. Box
556, 8203 Eu St.","23753","San José","Spain"),
("Briar Hayes","ante.lectus.convallis@google.net","(512) 583-
0462","6716 At Street","69759","North Sumatra","Brazil");
```

2. Configuração do Amazon Redshift

Uma instância do Amazon Redshift foi configurada para servir como o repositório final dos dados. Os passos para a configuração são os seguintes:

2.1. Foi criado um cluster chamado `redshift-cluster-arvore` utilizando a avaliação gratuita. O nome de usuário administrador foi configurado como `awssuser` e a senha como `Awsuser123`.

2.2. No cluster `redshift-cluster-arvore`, foi criado um usuário chamado `u_arvore` com a senha `u_Arvore123`.

2.3. No cluster `redshift-cluster-arvore`, foi criado um banco de dados chamado `arvore` e foram concedidas permissões ao usuário `u_arvore`.

2.4. No banco de dados `arvore`, foi criado um esquema chamado `dados` e foram concedidos privilégios no mesmo para o usuário `u_arvore` usando os seguintes comandos:

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

3. Configuração da Plataforma Cloud

A plataforma Google Cloud foi escolhida para executar este teste técnico. Para esta configuração, foram realizados os seguintes passos:

3.1. Foi criada uma conta pessoal no Google Cloud.

3.2. Foi criado um ambiente chamado `ambiente-arvore` no local `southamerica-east1` no Cloud Composer.

3.3. Foi criado o arquivo `requirements.txt` para incluir as bibliotecas `psycopg2` e `mysql-connector-python`.

3.4. 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. O funcionamento do script é descrito a seguir:

3.4.1. A função `decisao` é o ponto de partida do script. Ela lista todas as tabelas do banco de dados de origem (MySQL) e verifica a existência das mesmas no banco de dados de destino (Amazon Redshift). Se uma tabela existir no MySQL, 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.

3.4.2. A função `carga_total` cria a tabela no Amazon Redshift com uma estrutura compatível com a tabela existente no MySQL. Além disso, realiza a cópia dos dados de maneira integral da origem para o destino. Isso significa que todos os dados da tabela de origem são transferidos para a tabela de destino.

3.4.3. A função `carga_incremental` verifica a hora da última carga na tabela de destino e busca os registros que foram inseridos ou atualizados após essa hora no banco de dados de origem. Em seguida, insere os novos registros e atualiza os registros existentes no banco de dados de destino. Isso garante que a tabela de destino esteja sempre atualizada com as alterações mais recentes da tabela de origem.