Relatório

Carlos Martins, 18836

João Azevedo, 18845

Barcelos, Fevereiro 2023

Contents

# Introduction

Uma empresa que comercializa bicicletas numa cadeia de lojas de mobilidade sustentável, está a mudar o seu sistema de dados de base de dados relacionais *on-premise*, para Big Data. Além dos dados novos pretende integrar no seu novo sistema os dados históricos de vendas, que foram exportados dos sistemas atuais em formato CSV. O armazenamento de dados será feito em HDFS, e será utilizado o Hive para gestão de dados.

## Project objectives

Dada a introdução feita no capítulo anterior (1) pretendemos que a elaboração deste projeto tenha em conta os seguintes objetivos:

* Deve ser realizada a melhor modelação dos dados de acordo com o ficheiro de dados disponibilizado.
* Garantir o melhor formato para armazenamento dos dados em termos de *performance* e armazenamento.
* Garantir a melhor *performance* das *queries* realizadas em Hive.
* Implementar algumas análises de dados em Zepplin.

# Modelação de Dados

Nesta secção será apresentado o modelo de dados escolhido após a análise dos dados.

O primeiro passo passou pela análise do CSV disponibilizado para a realização do trabalho. Este CSV continha um conjunto de dados que estava agrupado e, inicialmente, foi realizada a separação do CSV original em múltiplos CSVs, onde cada um correspondia a uma tabela como mostra a figura 1.

Com base na figura 1, podemos verificar que o “produto” final consiste em quatro tabelas, sendo elas as seguintes:

* **SalesOrder –** Esta tabela é a tabela que armazena todas as “faturas” do cliente, ou seja, cada linha desta tabela representa uma encomenda realizada por um certo cliente. O seu conteúdo consiste nas informações temporais, financeiras e de cliente para a respetiva fatura.
* **SalesOrderDetail –** Esta tabela armazena todas as linhas de uma “fatura”, ou seja, o seu conteúdo é, basicamente, uma linha da fatura que contém que produto foi comprado, a respetiva quantidade e ao valor total da linha.
* **Product –** Esta tabela armazena informações sobre os produtos, tais como, o nome, o preço e peso.
* **Customer –** Esta tabela armazena as informações sobre clientes, tais como, o nome, nome da empresa e contactos.

Figura - Modelo de dados

De modo a garantir o melhor desempenho do sistema, foram escolhidos certos tipos de dados que após investigação e conhecimento obtido através de aula, funcionam melhor.

Para guardar texto, foi escolhido o tipo de dados *string* porque em Hive, ao contrário do tipo de dados *varchar*, as *strings* utilizam um processo chamado vectorização\* quando são utilizados ficheiros ORC (que é o nosso caso).

Para guardar números, foram utilizados diferentes tipos de dados:

* Quanto a números inteiros dentro do intervalo -128 a 127 foi utilizado o tipo de dados *tinyint*. Este tipo de dados foi utilizado para as colunas “RevisionNumberId” e “OnlineOrderFlag” da tabela SalesOrder, por exemplo.
* Relativamente a números inteiros dentro do intervalo -32768 a 32767 foi utilizado o tipo de dados *smallint*. Este tipo de dados foi utilizado para as colunas “ShipToAddressId” e “BillToAddressId” da tabela SalesOrder, por exemplo.
* Por fim, para armazenar números flutuantes com precisão até 7 casas decimais, foi utilizado o tipo de dados *float*. Este tipo de dados foi utilizado nas colunas “UnitPrice” e “UnitPriceDiscount” da tabela Product, por exemplo.

Para armazenar datas com horas, foi utilizado o tipo de dados *timestamp* uma vez que é o único definido para esse efeito.

\*A vectorização é um processo que permite ao Hive processar linhas em *batch* em vez de uma de cada vez, o que aumenta a velocidade de processamento.

# Armazenamento de dados em Hive

O Hive suporta vários formatos de arquivos, cada um com suas próprias características, vantagens e usos adequados. Um dos formatos suportados é o **Text File Format**. este é o formato de arquivo mais simples suportado por Hive. Os arquivos de texto são armazenados como arquivos de texto simples no HDFS. Este formato é mais adequado para projetos de pequena escala ou para dados não estruturados. O **SequenceFile Format** é usado para armazenar séries de pares chave-valor. É mais adequado para dados com grande quantidade de dados, pois é mais eficiente em termos de espaço de armazenamento e desempenho de leitura. O **RCFile** (Record Columnar File) é semelhante ao SequenceFile, mas é otimizado para colunas em vez de linhas. Este formato é mais adequado para dados estruturados com muitas colunas, pois é mais eficiente em termos de armazenamento e desempenho de consulta. O **ORC** (Optimized Row Columnar) File é o formato de arquivo mais avançado suportado por Hive. Este formato é otimizado para consultas de colunas e tem muitas outras melhorias de desempenho, como compressão de dados e suporte para dados nulos. Este formato é mais adequado para projetos de grande escala com dados estruturados. Por fim, temos o **Parquet File** é outro formato de arquivo avançado suportado por Hive. É semelhante ao ORC, mas é projetado para ser utilizado com vários sistemas de base de dados, incluindo Hive, Impala, Spark, etc. Este formato é mais adequado para projetos que exigem colaboração entre sistemas de base de dados.

Após a análise dos formatos indicados acima foi escolhido como tipo de formato final o ORC por garantir um bom desempenho do sistema.

# Implementação do Modelo

# Upload dos Dados

Para efetuar o upload de um arquivo csv no Hive seguimos os seguintes passos:

1. Criar uma tabela no Hive de forma temporária com colunas correspondentes aos dados do arquivo csv.
2. Copiar o arquivo para o HDFS
3. Utilizar o comando LOAD DATA INPATH no Hive para carregar os dados do arquivo csv para a tabela temporária.
4. Criar a uma tabela com o formato ORC
5. Inserir os dados da tabela temporária para a tabela final com um INSERT INTO TABLE

O código \* mostra a criação da tabela temporária este processo foi efetuado para 4 as entidades, após isto criamos duas pastas na diretoria …. E fizemos upload dos ficheiros csv para a diretoria … a criação das duas pastas e não só uma deveu-se ao facto de querermos ter uma separação entre tabelas finais e tabelas temporárias, as tabelas finais encontram-se na pasta \*\* enquanto que na pasta \*\* estão as tabelas temporárias. Através do código \* procedemos ao import dos dados do csv para as tabelas temporárias, após isto criamos as tabelas finais com o formato de ORC, por fim fizemos um insert dos dados das tabelas temporárias para as tabelas finais.

# Implementação e análise das querys

# Implementação e análise das querys em Zeppelin