Projeto de Data Science

Slow Changing Dimensions in a Data Warehouse Using Hive

Fellipe Augusto Soares Silva

2020

Chapters

Figures

Tables

Summary

This project was developed with the purpose of applying rules of changes in dimensionality and storage of previous versions of data using the concept of Slow Changing Dimension (SCD) in a Hadoop cluster.

A Slow Changing Dimension will be applied to a Data Warehouse using Hive that is running on Apache Hadoop, and for that to happen it will be necessary to make total configuration of a pseudo-distributed cluster to store and process the data.

The cluster will be built in a Linux environment through a Virtual Machine, that is, through the virtualization of the environment on my physical machine. So I also performed the entire configuration of the Virtual Machine preparing to receive Hadoop ecosystem.

After configuring the Virtual Machine and the Hadoop Environment with Pseudo - Distributed architecture, I performed the transfer of the Physical Machine Database to the Virtual Machine storing it in MySQL configured on Linux.

With the Database in the Virtual Machine we started the business problem using Sqoop (data transfer tool for Apache Hadoop) to load a table from the source Database into a datalake in HDFS (Hadoop Distributed File System), so we leave only the columns necessary for testing the application.

Inside Hive we created the table schema that will receive and manage the occurrence of Slow Changing Dimension, preparing to be able to receive new data and compare with the information contained therein, making the versioning of altered data keeping a history of the last modifications.

Finally, I developed a sequence of codes so that Hive can make Slow Changing Dimension, that is, read, compare and add new information to the table, making changes to dimensions intelligently whenever they change.

Palavras – Chave: Data Engineering, Data Science, Apache Haddop, Hadoop Ecosystem, Structured Databases, Relational Database, Sqoop, Hive, Data Warehouse, Slow Changing Dimension, Virtual Machine, Linux, HDFS.

1. Introduction

This project was developed using an Oracle Virtual Machine[1] (VM) where I installed and configured the virtualization of the Linux[2] Operational System from start to finish through the CentOS[3] distribution.

A Virtual Machine is a computational environment that runs on the physical machine but is virtualized and isolated from the real machine. In the VM we can perform tests on different operational systems such as MacOS, Windows, Linux without leaving the base operational system of the machine and can even share items between them.

The choice of CentOS was made because it is a free and robust project regarding the installation of an open - source ecosystem. During the installation a series of configurations were adjusted in order to prepare the virtual environment to receive a Pseudo - Distributed environment from Hadoop Ecosystem[4].

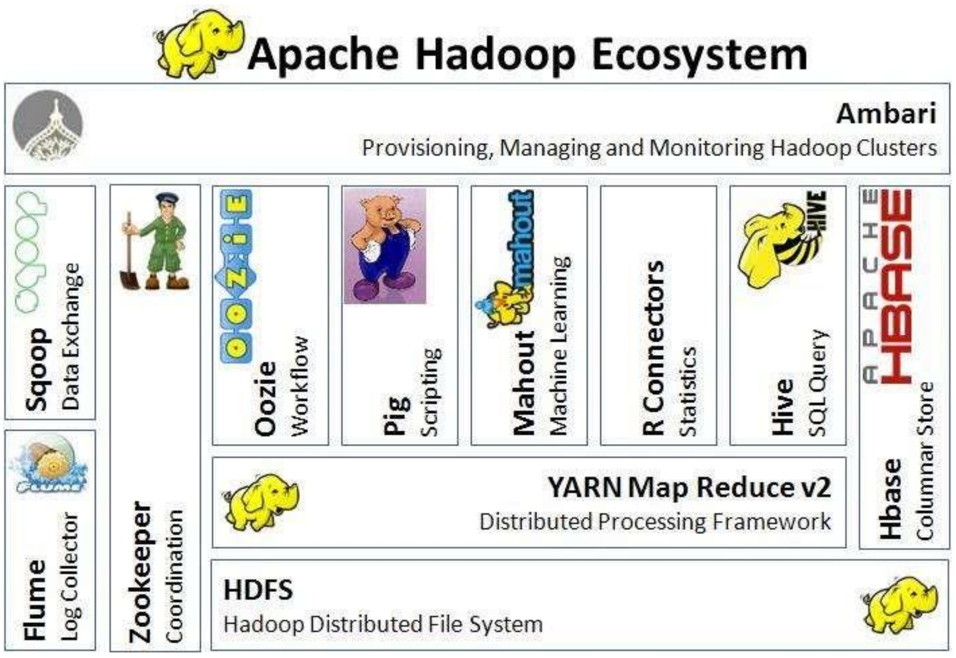


Figure 1 - Hadoop Ecosystem

Hadoop is an open source framework used for distributed storage and parallel processing, it is fault tolerant, scalable, secure and developed for distributed computing in a computer cluster. It is the basis for operations with large data sets, Big Data, as it offers a robust ecosystem

[1] https://www.oracle.com/br/virtualization/virtualbox/
[2] https://en.wikipedia.org/wiki/Linux
[3] https://www.centos.org/
[4] https://hadoop.apache.org/

with different modules that assist in data collection (such as Flume[5] and Sqoop[6]), storage with HDFS (Hadoop Distributed File System), data structuring (with Hive[7] and Hbase[8]), data processing with MapReduce operations using for example Spark[9], and can also perform Machine Learning on large data sets distributed by the cluster with Apache Mahout[10].

Hadoop is an Apache Foundation project, and is mainly composed of three modules:

- Apache HDFS: Responsible for distributed storage in the cluster;
- Hadoop Yarn: Responsible for resource management (memory, CPU ...);
- Hadoop MapReduce: Responsible for parallel processing of large data sets.

An HDFS architecture works with the Master / Worker functions, that is, a Master machine as the master of the entire cluster managing the storage and parallel processing communicating with Workers nodes using a trace of the operations performed.

In a Hadoop environment we have a computer running Master processes (management processes) that are:

- Secondary NameNode: function similar to a backup, although it can assume management functions.
- NameNode: manages HDFS;
- JobTracker: manages MapReduce Jobs.

The other computers in a cluster are the slaves (Slaves / Workers) and it is these processes that do the job itself:

- DataNode: stores and retrieves data from HDFS;
- TaskTracker: performs the Mapping and Reduction work.

DataNode and TaskTracker run on the same machine.

A Client machine, for example my PC running R or Python language, makes a request to the Master computer and this in turn puts the Workers to work either to store or retrieve data from the cluster or perform MapReduce operations.

The Master (or name node) must be the machine within the cluster with the best processing among all. It keeps all your information in memory and to manage all this it has two very important data structures (two files):

[5] https://flume.apache.org/
[6] https://sqoop.apache.org/
[7] https://hive.apache.org/
[8] https://hbase.apache.org/
[9] https://spark.apache.org/
[10] https://mahout.apache.org/

- Fsimage: responsible for storing structural information of the logs such as mapping and namespace of files, in addition to the location of replicas of these files.
- EditLog: responsible for storing all changes to file metadata.

A function as important as distributed storage and parallel processing is replication because in addition to dividing files into blocks, HDFS replicates blocks in an attempt to increase security. By default, HDFS has three (3) replicas allocated on different machines in the cluster (this amount can be configured).

There is still a recommendation for safety and reliability and performance to allocate two (2) replicas in the same rack, but on different machines and the other replica in a different rack.

As physically communication between machines in the same rack is faster than with other racks, for performance reasons when selecting a replica for the process, HDFS gives preference to the replica that belongs to the same rack.

Another benefit with Replication is greater fault tolerance and data reliability, because if a worker machine fails, processing will be done by another machine that contains the replica of that block without the need for data transfer or interrupting application execution.

There are many benefits to working with Apache Hadoop but each of the modules of the Hadoop ecosystem requires a specific configuration that can be found in the official documentation. During the presentation of this project I will highlight the most important configurations I used and for the other configurations I advise you to look for information in the reference material[11].

After installing CentOS with the Linux Operating System, you need to install Hadoop on the Virtual Machine as it forms the basis of the Ecosystem, other products of the Hadoop Ecosystem depend either on HDFS or MapReduce.

When installing CentOS, I created two users:

- root: who is the administrator of the environment
- fellipe: Linux environment user

I could use the fellipe user but for best practices I will build my Hadoop environment on a new user: the hadoop user. In this user I will parameterize my entire parallel and distributed computing environment using HDFS and other ecosystem modules.

[11] https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html

In practice, Apache Hadoop is installed without an installer, that is, a file is downloaded, unzipped, inserted into a folder on the VM and configured with its environment variables. After that it is ready for use.

The files needed for configuration are:

- .bashrc
- core-site-xml
- hdfs-site.xml

As stated earlier, the items to be configured within these files are in the official product documentation[12].

Another important module for the operation of HDFS is YARN[13]. This module is responsible for managing jobs within HDFS and must be started together with HDFS so that both the storage and processing of data distributed by the cluster can take place. The files needed for configuration are:

- mapred-site.xml
- yarn-site.xml

To initialize both HDFS and YARN after configurations done, we must type the following commands in the terminal:

- start-dfs.sh
- start-yarn.sh

As a result we have to have the following services running on the OS:

- NameNode
- DataNode
- SecondaryNameNode
- NodeManager
- ResourceManager

Continuing to install the Hadoop ecosystem,:

- Zookeeper: It aims to provide a coordination service for high-performance distributed applications that provides the means to facilitate the tasks of configuring a machine in the cluster, synchronizing distributed processes, and groups of services.

[12] https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html
[13] https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

- HBase: It is a distributed and scalable database, being a NoSQL database, that is, it allows you to store data without worrying about the schema.
- Hive: Its main functionality is to provide an infrastructure that allows the use of SQL language, in fact a modification of SQL that is HiveQL and allows to build a kind of data warehouse under Apache Hadoop. If the goal is to work with structured data and execute almost SQL queries, Apache Hive is an excellent alternative.
- Pig: It is a high level language oriented to data flow and execution for parallel computing. The optimization of Apache Pig does not change the configuration of the Hadoop cluster because it is used in client mode providing a language called Pig Latin and a controller capable of transforming programs of the Pig type into sequences of the MapReduce programming model. What it does is convert the Pig Latin code to JAVA or Scala format (Hadoop programming languages) in order to work on Jobs without much difficulty.
- Spark: Processing of mapreduce jobs within the cluster
- Sqoop: Data transfer tool for Apache Hadoop that can put data from a database into HDFS, Hive or HBase, or do the opposite way of exporting data from this distributed platform back to the relational database.
- Flume: Allows me to bring data from the most varied sources to Apache Hadoop, including log data (when running any application it generates output - information of who accessed the application, when it accessed, what resources it used, if there was memory overflow and so on)
- Mahout: Hadoop Ecosystem product to work with Machine Learning. Simplifies (as Pig does) the Machine Learning process on distributed data.

We have three modes of execution of Haddop where each item of the ecosystem must be configured according to the chosen mode:

- Standalone: In this mode, hadoop is configured to run in local mode. This mode is most recommended for the development phase, which is when most errors normally occur, being necessary to carry out several tests of the execution of the application of my data analysis.

- Pseudo - Distributed: In this mode, all the configurations that are necessary for the execution of a cluster are applied, however, the whole application is processed in Local Mode. Although it is not actually executed in parallel, this mode allows its simulation as it uses all the processes of an effective parallel execution: NameNode, DataNode, JobTracker, TaskTracker and SecondaryNameNode.
- Fully Distributed: Mode used for distributed processing of the Hadoop application on a computer cluster. In this option it is also necessary to edit the XML files previously presented, defining specific parameters and the location of SecondaryNameNode and Nodes Workers. However, as we have several computers in this mode, we must indicate which machines will actually run each component.

Each module was configured following the Hadoop Pseudo - Distributed execution mode so that we would be getting as close as possible to a real environment.

The beginning of the solution to any business problem lies in the understanding of the problem itself, which will be presented in the next chapter.

2. Business Problem

As described in the title of the project we will develop Slow Changing Dimensions in a Data Warehouse Using Hive but to fully understand the problem we will divide the concepts involved here.

Data Warehouse is a repository that is oriented by variables, subjects, integrated and variable over time, being organized to assist in making strategic decisions. In addition, a Data Warehouse has the characteristic of using all company data and storing historical data maintaining the authenticity and veracity of the information, thus matching the facts with the time of its occurrence.

A Data Warehouse is fed through an ETL process (Extract - Transform - Load) where the collection of data at the source (from operating systems, ERPs, CRMs, web files, databases, among others) undergoes a transformation by adjusting information applying standards and later uploading this data to the Data Warehouse making them available to analysis teams like Data Scientists.

The dimensions of a Data Warehouse correspond to the data stored in this DW (such as name, telephone, email, in case of registration tables for example) and a Slow Changing Dimension (SCD) is a dimension that changes slowly in this Data Warehouse and which rarely changes, but when it does change it is necessary to classify the type of existing change and update the corresponding fields, adopting an organized approach in order to correctly document the type of change. Let's consider the following dimension as an example:

Item	ID	Name	Occupation
749	63	Thais Pagani	Marketing

Table 1 - Slow Changing Dimension Example

Each Slow Changing Dimensions varies according to the data update characteristic in the Data Warehouse and we can list 3 alternatives to deal with them:

- Slow Changing Dimensions Type 1: in this type the changes overlap the existing data without leaving a trace of previous data. It is the simplest SCD model but may not be the most suitable for corporate environments as there is no version control for the modified registry. Note in the table below that when changing the user's Occupation, we lost the previous Occupation.

Item	ID	Name	Occupation
749	63	Thais Pagani	Brand Manager

Table 2 - Slow Changing Dimension Type 1

- Slow Changing Dimensions Type 2: in this type we have a greater control of versioning because the previous data is preserved with the insertion of a new line preserving what was not changed and inserting the modifications. Note in the table below that when changing the occupation a new line was inserted keeping the previous one, the only problem here is that some kind of control of the number of stored changes is necessary because the table can assume much greater proportions if each change is inserted in a new line.

Item	ID	Name	Occupation	Version
749	63	Thais Pagani	Marketing	001
749	63	Thais Pagani	Brand Manager	002

Table 3 - Slow Changing Dimension Type 2

- Slow Changing Dimensions Type 3: this type is similar to type 2 but we do not add rows with all the data but a column with only the modifications. Note in the table below that we preserve the previous data and add the user's new occupation. This case must also follow a systematic control so that the table size does not grow without control and compromises the Data Warehouse.

Item	ID	Name	Occupation	New_Occupation
749	63	Thais Pagani	Marketing	Brand Manager

Table 4 - Slow Changing Dimension Type 3

A more complex and complete model would be a Slow Changing Dimension Hybrid where the combination of different SCDs mentioned above are assumed according to the business problem and the company's wishes. For our case we will use a Slow Changing Dimension Type 3.

Another concept involved in the project is the use of Apache Hive, a framework that allows the user to work in a structured way with data stored in HDFS through commands close to SQL, HiveQL.

With Hive we will take advantage of Apache Hadoop features like distributed storage and parallel processing as it was developed on top of Hadoop. Hive allows us to work in a structured way in a Big Data universe where data is mostly unstructured, allowing us to define the structure of the data to be stored and later summarized, consulted and analyzed.

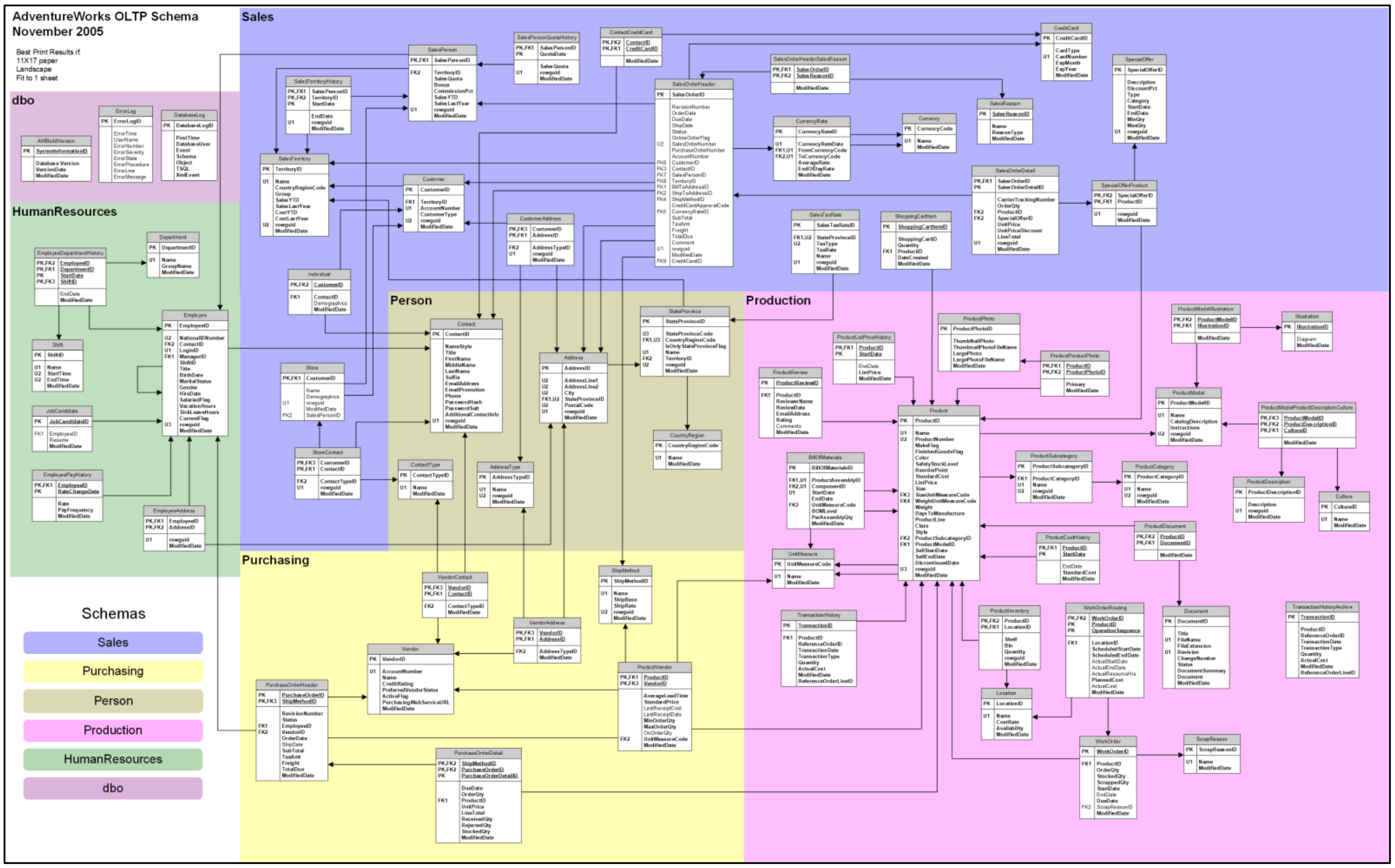
Returning to the business problem, after all the concepts elucidated, what we are going to do here is load a piece of a relational database to a datalake inside Apache Hadoop using HDFS for storage, assemble a schema inside Hive loading this data from the datalake and then assemble a HiveQL statement so that when changes occur in the dimensions of the original database, SCDs also occur within the Hive table.

3. Database Used

To carry out the business problem, a relational database was provided. In it we have several tables that communicate with each other with different information from a company called AdventureWorks. The Database has 5 main sectors:

- Sales: Sales Sector
- Purchasing: Purchasing Sector
- Person: Customer Information
- Production: Products Production Sector
- HumanResources: Human Resources Sector

Below you can see the schema of this database and the relationship between its tables:



Figuea 2 - Relational Database

To show how Slow Changing Dimension works, we will use only one table (figure below) so that we can replicate the concept for any other table, be it a table in this database or any other source of information. Following table used:

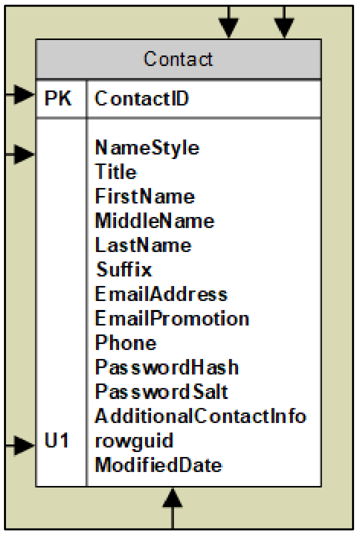


Figure 3 - Business problem table

Note in the image above that we have a table with user information such as First Name, Middle Name, Last Name, Email, Telephone, Password, Additional Information and Modification Date. As the objective here is to study the concept involved in a Slow Changing Dimensions in a Data Warehouse using Hive, I will use part of the information within this table indicating the success in the operation of updating its dimensions.

4. Project Development

To facilitate understanding, I developed the following pipeline in order to explain each step that we will go through during development:

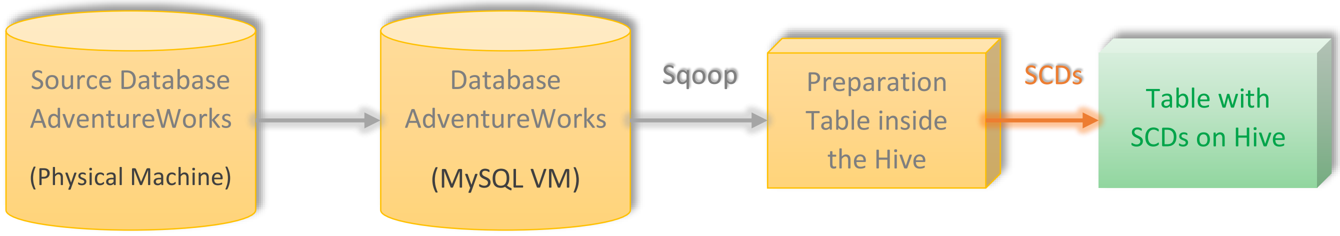

Source Database AdventureWorks (Physical Machine)
Database AdventureWorks (MySQL VM)
Sqoop
Preparation Table inside the Hive
SCDs
Table with SCDs on Hive

Figura 4 - Pipeline Projeto

4.1 Source Database

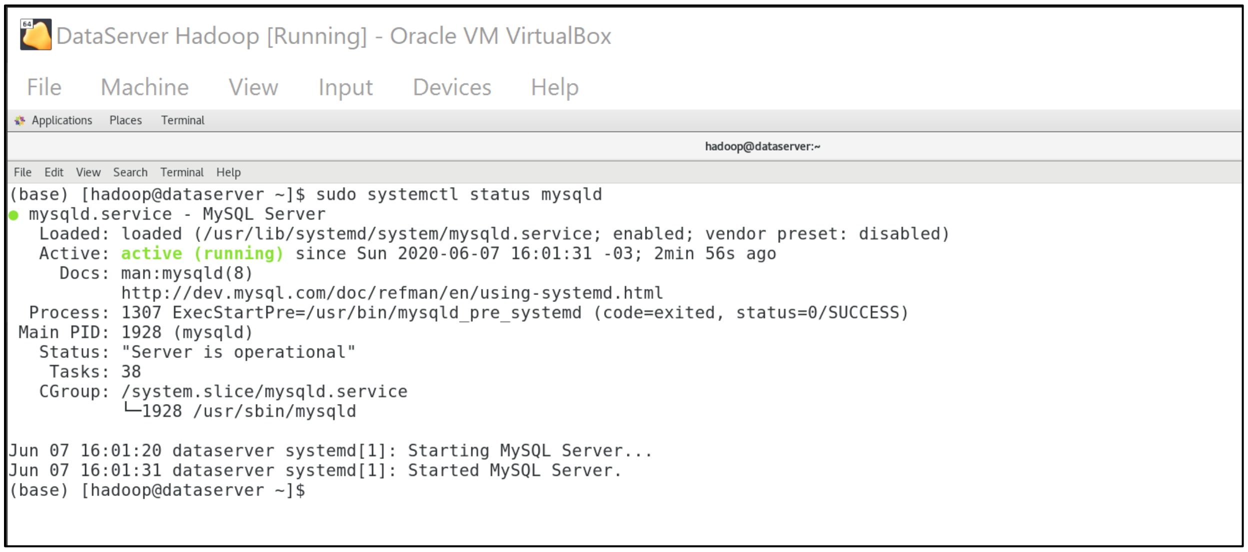
The source database is the one on the AdventureWorks company server that is constantly updated by customers, users and company employees as new transactions take place.

This is the database that we will use for the next step and it is from it that the changes come so that I can perform Slow Changing Dimensions in a Data Warehouse Using Hive.

4.2 Database and MySQL on the VM

During the configuration phase of my Virtual Machine I performed the installation and parameterization of MySQL on Linux so that I can import the source database. In this way I downloaded the database on my physical machine, shared it with the virtual machine, granted access permissions and transferred the complete database to MySQL.

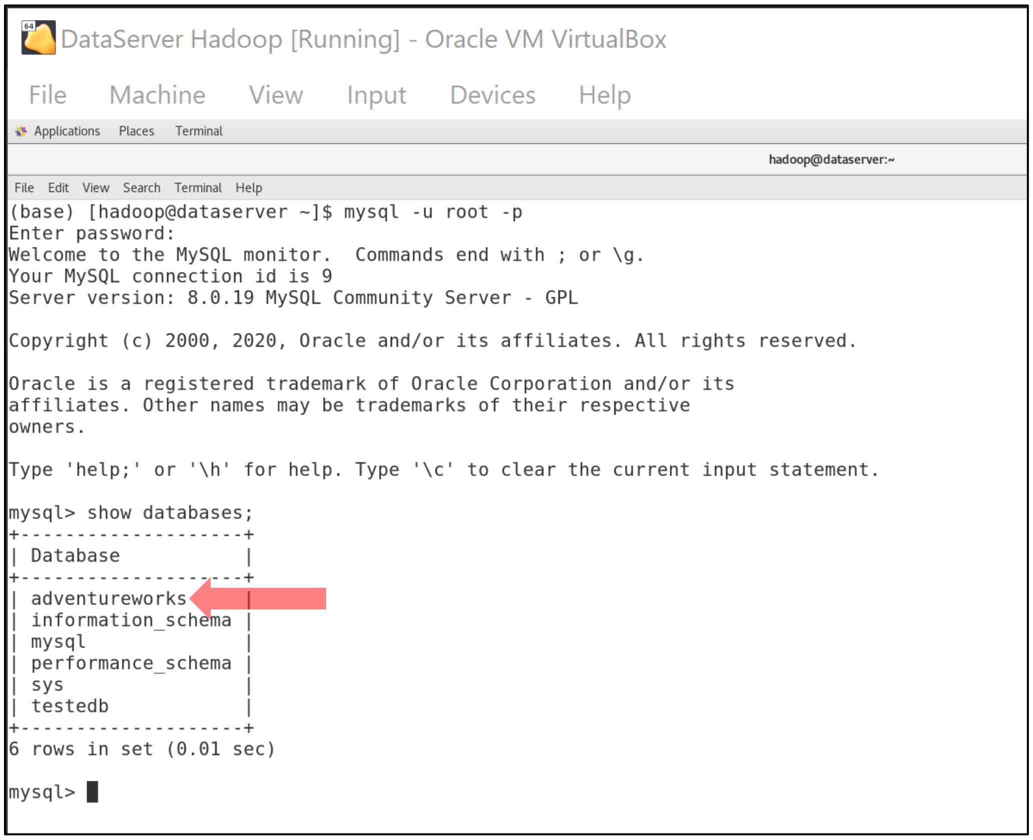
First checking if we have MySQL active:


DataServer Hadoop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal
hadoop@dataserver:~
File Edit View Search Terminal Help

```
(base) [hadoop@dataserver ~]$ sudo systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2020-06-07 16:01:31 -03; 2min 56s ago
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
  Process: 1307 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)
 Main PID: 1928 (mysqld)
   Status: "Server is operational"
    Tasks: 38
   CGroup: /system.slice/mysqld.service
           └─1928 /usr/sbin/mysqld

Jun 07 16:01:20 dataserver systemd[1]: Starting MySQL Server...
Jun 07 16:01:31 dataserver systemd[1]: Started MySQL Server.
(base) [hadoop@dataserver ~]$
```

Figure 5 - Status MySQL

After loading the database and all its tables, we will perform a query and view the databases that we have on MySQL:

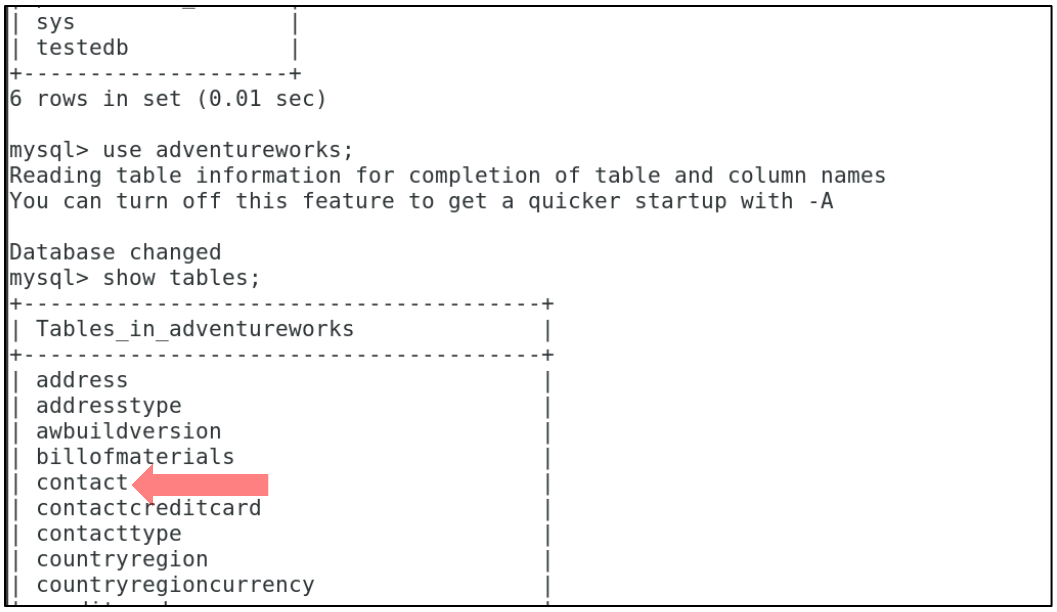

DataServer Hadoop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal
hadoop@dataserver:~
File Edit View Search Terminal Help

```
(base) [hadoop@dataserver ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| adventureworks     |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| testedb            |
+--------------------+
6 rows in set (0.01 sec)

mysql>
```

Figure 6 - Databases inside MySQL

As indicated by the red arrow in the figure above we have the loaded database and all its tables.

Viewing the tables, including the table that we will use in this project (red arrow in the image below):

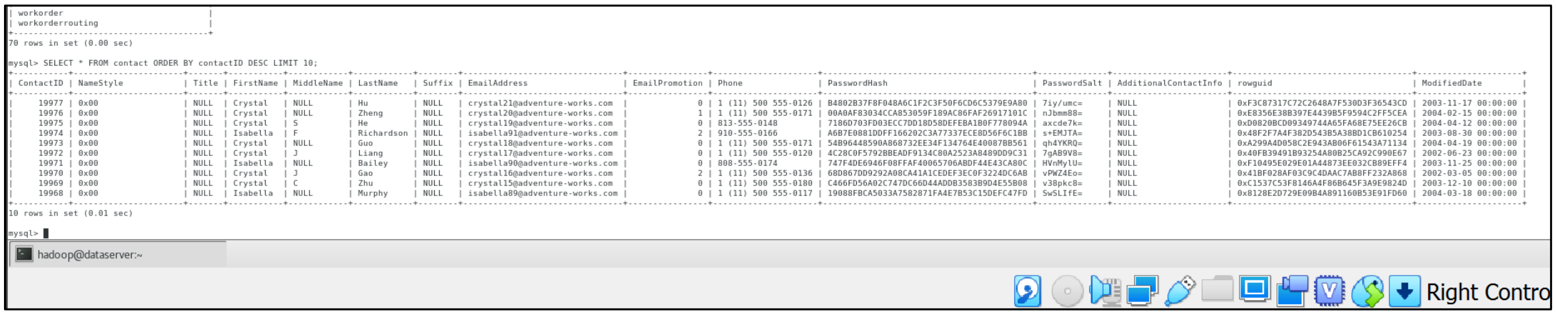


```
| sys                 |
| testedb             |
+---------------------+
6 rows in set (0.01 sec)

mysql> use adventureworks;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+--------------------------------------+
| Tables_in_adventureworks             |
+--------------------------------------+
| address                              |
| addresstype                          |
| awbuildversion                       |
| billofmaterials                      |
| contact                              |
| contactcreditcard                    |
| contacttype                          |
| countryregion                        |
| countryregioncurrency                |
```

Figure 7 – Tables within Database AdventureWorks

Following is a query performed on MySQL installed on the VM using the following SQL command to return all columns in the "contact" table limiting to the last 10 records:

SELECT * FROM contact ORDER BY contactID DESC LIMIT 10;



```
| workorder                            |
| workorderrouting                     |
+--------------------------------------+
70 rows in set (0.00 sec)

mysql> SELECT * FROM contact ORDER BY contactID DESC LIMIT 10;
+-----------+----------------------+-------+-----------+------------+-------------+--------+----------------------------------+----------------+---------------------+------------------------------------------+--------------+-----------------------+------------------------------------+---------------------+
| ContactID | NameStyle            | Title | FirstName | MiddleName | LastName    | Suffix | EmailAddress                     | EmailPromotion | Phone               | PasswordHash                             | PasswordSalt | AdditionalContactInfo | rowguid                            | ModifiedDate        |
+-----------+----------------------+-------+-----------+------------+-------------+--------+----------------------------------+----------------+---------------------+------------------------------------------+--------------+-----------------------+------------------------------------+---------------------+
|     19977 | 0x00                 | NULL  | Crystal   | NULL       | Hu          | NULL   | crystal21@adventure-works.com    |              0 | 1 (11) 500 555-0126 | B4802B37F8F048A6C1F2C3F50F6CD6C5379E9A80 | 7iy/umc=     | NULL                  | 0xF3C87317C72C2648A7F530D3F36543CD | 2003-11-17 00:00:00 |
|     19976 | 0x00                 | NULL  | Crystal   | NULL       | Zheng       | NULL   | crystal20@adventure-works.com    |              1 | 1 (11) 500 555-0171 | 00A0AF83034CCA853059F189AC86FAF26917101C | nJbmm88=     | NULL                  | 0xE8356E38B397E4439B5F9594C2FF5CEA | 2004-02-15 00:00:00 |
|     19975 | 0x00                 | NULL  | Crystal   | S          | He          | NULL   | crystal19@adventure-works.com    |              0 | 813-555-0148        | 7186D703FD03ECC7DD18D58DEFEBA1B0F778094A | axcde7k=     | NULL                  | 0xD0820BCD09349744A65FA68E75EE26CB | 2004-04-12 00:00:00 |
|     19974 | 0x00                 | NULL  | Isabella  | F          | Richardson  | NULL   | isabella91@adventure-works.com   |              2 | 910-555-0166        | A6B7E0881DDFF166202C3A77337ECE8D56F6C1BB | s+EMJTA=     | NULL                  | 0x48F2F7A4F382D543B5A38BD1CB610254 | 2003-08-30 00:00:00 |
|     19973 | 0x00                 | NULL  | Crystal   | NULL       | Guo         | NULL   | crystal18@adventure-works.com    |              0 | 1 (11) 500 555-0171 | 54B96448590A868732EE34F134764E40087BB561 | qh4YKRQ=     | NULL                  | 0xA299A4D058C2E943AB06F61543A71134 | 2004-04-19 00:00:00 |
|     19972 | 0x00                 | NULL  | Crystal   | J          | Liang       | NULL   | crystal17@adventure-works.com    |              0 | 1 (11) 500 555-0120 | 4C28C0F5792BBEADF9134C80A2523A8489DD9C31 | 7gAB9V8=     | NULL                  | 0x40FB39491B93254A80B25CA92C990E67 | 2002-06-23 00:00:00 |
|     19971 | 0x00                 | NULL  | Isabella  | NULL       | Bailey      | NULL   | isabella90@adventure-works.com   |              0 | 808-555-0174        | 747F4DE6946F08FFAF40065706ABDF44E43CA80C | HVnMylU=     | NULL                  | 0xF10495E029E01A44873EE032CB89EFF4 | 2003-11-25 00:00:00 |
|     19970 | 0x00                 | NULL  | Crystal   | J          | Gao         | NULL   | crystal16@adventure-works.com    |              2 | 1 (11) 500 555-0136 | 68D867DD9292A08CA41A1CEDEF3EC0F3224DC6AB | vPWZ4Eo=     | NULL                  | 0x41BF028AF03C9C4DAAC7AB8FF232A868 | 2002-03-05 00:00:00 |
|     19969 | 0x00                 | NULL  | Crystal   | C          | Zhu         | NULL   | crystal15@adventure-works.com    |              0 | 1 (11) 500 555-0180 | C466FD56A02C747DC66D44ADDB3583B9D4E55B08 | v38pkc8=     | NULL                  | 0xC1537C53F8146A4F86B645F3A9E9824D | 2003-12-10 00:00:00 |
|     19968 | 0x00                 | NULL  | Isabella  | NULL       | Murphy      | NULL   | isabella89@adventure-works.com   |              0 | 1 (11) 500 555-0117 | 19088FBCA5033A7582871FA4E7B53C15DEFC47FD | SwSLIfE=     | NULL                  | 0x8128E2D729E09B4A891160B53E91FD60 | 2004-03-18 00:00:00 |
+-----------+----------------------+-------+-----------+------------+-------------+--------+----------------------------------+----------------+---------------------+------------------------------------------+--------------+-----------------------+------------------------------------+---------------------+
10 rows in set (0.01 sec)

mysql> 
```

Figure 8 - Last records of table "contact"

After transferring and checking that everything is working perfectly with our database, we can define two premises:

- First: it is possible to observe that we have null data within the table, but I will not worry about treating this information as it is not the objective of this project;
- Second: we have 15 columns in the table, as shown above, to improve the operation I will limit the SCDs on Hive to the 7 main columns (ContactID, Title, FirstName, LastName, EmailAddress, Phone, ModifiedDate). So we developed the project without running away from the main concept of seeing SCDs happening as the source database is updated.

4.3 Loading Data on Hive with Apache Sqoop

Note in figure 4 that our next step is to load the "contact" table into Hive, forming a preparation table. In this step, I will load the data into my datalake inside HDFS without worrying about formatting or deleting columns. Just transfer a table from a source database into Hadoop.

To carry out the transfer, we have several platforms that use communication protocols and insert data according to specifications. In this project, I chose to use Apache Sqoop as it is an efficient tool for transactions between Hadoop and structured datastores.

Apache Sqoop[14] is basically an ETL tool that allows you to bring data from HDFS to a relational database, or to bring data to HDFS from a relational database.

To use Sqoop, I downloaded the file from the Apache Foundation website and configured it according to the documentation, but to perform ACID transactions (set of database transaction properties that allows the movement of databases and tables between platforms), it is necessary to adjust some variables in both Sqoop and Hive.

4.3.1 Troubleshooting

First we have to check if the libraries shared by both are compatible. Enter the path where hive libraries are installed: /hive/lib and identify the version of hive-common-X.X.X.jar. Copy this file into the path where the sqoop libraries are installed: /sqoop/lib if there is a different version in sqoop and delete the different version.

Then edit the environment variables by entering the path where Hive settings should be executed, because we are going to edit the settings so that we can force sqoop to read these settings and perform ACID transactions.

Inside /hive/conf folder, search for the file hive-site.xml (if it doesn't exist, make a copy of hive-default.xml) and modify the following settings:

- hive.support.concurrency –> true
- hive.enforce.bucketing –> true
- hive.exec.dynamic.partition.mode –> nonstrict
- hive.txn.manager –> org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
- hive.compactor.initiator.on –> true

[14] https://sqoop.apache.org/

- hive.compactor.worker.threads –> a positive number
- hive.auto.convert.join -> false

All these configurations were made according to the documentation available at: https://cwiki.apache.org/confluence/display/Hive/Hive+Transactions

That done, we can use sqoop to move the "contact" table from the adventureworks database present in MySQL into Hive, which is running on HDFS, with the following command:

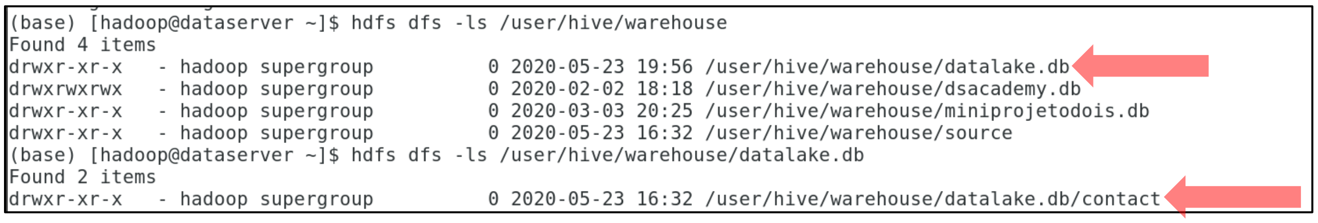
sqoop import --connect jdbc:mysql://localhost:3306/adventureworks?serverTimezone=UTC –username root --P –columns ContactID,Title,FirstName,LastName,EmailAddress,Phone,ModifiedDate --table contact --target-dir /user/hive/warehouse/source/contact --fields-terminated-by "," --hive-import --create-hive-table --hive-table datalake.contact

This instruction will create the "source" folder inside HDFS and insert the "contact" table from MySQL inside it. When the table is finished transporting, sqoop will import the "contact" table into the datalake on hive and clean the "source" directory created initially.

Let's look at each command in the instruction above:

- connect: JDBC connection between MySQL and Hadoop Server
- username: username responsible for the database
- P: requires password to connect to the database, continuing the process
- columns: columns of the source table that I want to transfer. Note that I have not used all of them
- table: table that will be accessed for the transfer process
- target-dir: destination directory on HDFS (Hadoop Distributed File System)
- fields-terminated-by: I may specify the column separator to be imported
- hive-import: instructions to import a table into the section
- hive-table: hive table where data will be inserted

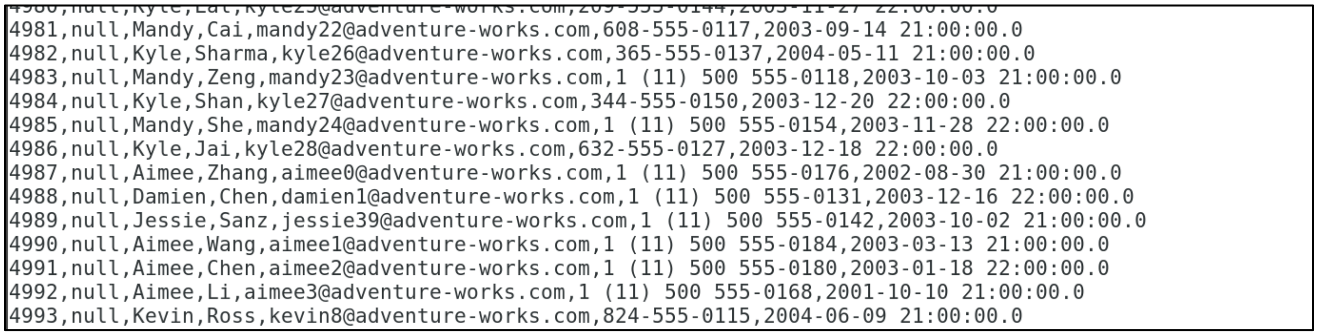
Accessing our cluster we can see that this transfer happened successfully:



```
(base) [hadoop@dataserver ~]$ hdfs dfs -ls /user/hive/warehouse
Found 4 items
drwxr-xr-x   - hadoop supergroup          0 2020-05-23 19:56 /user/hive/warehouse/datalake.db
drwxrwxrwx   - hadoop supergroup          0 2020-02-02 18:18 /user/hive/warehouse/dsacademy.db
drwxr-xr-x   - hadoop supergroup          0 2020-03-03 20:25 /user/hive/warehouse/miniprojetodois.db
drwxr-xr-x   - hadoop supergroup          0 2020-05-23 16:32 /user/hive/warehouse/source
(base) [hadoop@dataserver ~]$ hdfs dfs -ls /user/hive/warehouse/datalake.db
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2020-05-23 16:32 /user/hive/warehouse/datalake.db/contact
```

Figure 9 - HDFS with Transformation Table

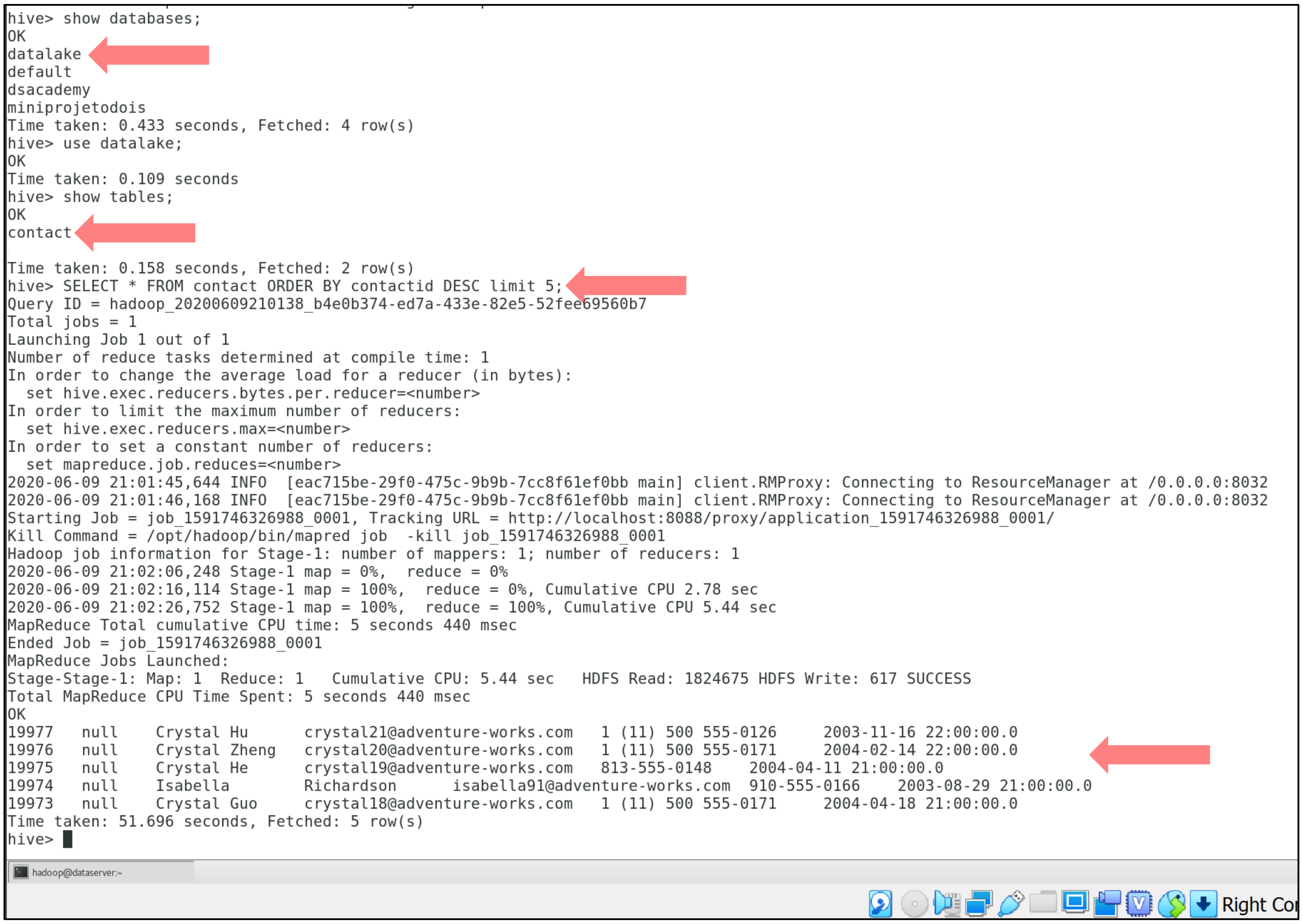
Within HDFS we can also access the data recorded there, as follows:



```
4980,null,Kyle,Lal,kyle25@adventure-works.com,209-555-0144,2003-11-27 22:00:00.0
4981,null,Mandy,Cai,mandy22@adventure-works.com,608-555-0117,2003-09-14 21:00:00.0
4982,null,Kyle,Sharma,kyle26@adventure-works.com,365-555-0137,2004-05-11 21:00:00.0
4983,null,Mandy,Zeng,mandy23@adventure-works.com,1 (11) 500 555-0118,2003-10-03 21:00:00.0
4984,null,Kyle,Shan,kyle27@adventure-works.com,344-555-0150,2003-12-20 22:00:00.0
4985,null,Mandy,She,mandy24@adventure-works.com,1 (11) 500 555-0154,2003-11-28 22:00:00.0
4986,null,Kyle,Jai,kyle28@adventure-works.com,632-555-0127,2003-12-18 22:00:00.0
4987,null,Aimee,Zhang,aimee0@adventure-works.com,1 (11) 500 555-0176,2002-08-30 21:00:00.0
4988,null,Damien,Chen,damien1@adventure-works.com,1 (11) 500 555-0131,2003-12-16 22:00:00.0
4989,null,Jessie,Sanz,jessie39@adventure-works.com,1 (11) 500 555-0142,2003-10-02 21:00:00.0
4990,null,Aimee,Wang,aimee1@adventure-works.com,1 (11) 500 555-0184,2003-03-13 21:00:00.0
4991,null,Aimee,Chen,aimee2@adventure-works.com,1 (11) 500 555-0180,2003-01-18 22:00:00.0
4992,null,Aimee,Li,aimee3@adventure-works.com,1 (11) 500 555-0168,2001-10-10 21:00:00.0
4993,null,Kevin,Ross,kevin8@adventure-works.com,824-555-0115,2004-06-09 21:00:00.0
```

Figure 10 – HDFS Records

Although it is possible to access and view the transformation table, we can view this table more clearly within Hive itself:



```
hive> show databases;
OK
datalake
default
dsacademy
miniprojetodois
Time taken: 0.433 seconds, Fetched: 4 row(s)
hive> use datalake;
OK
Time taken: 0.109 seconds
hive> show tables;
OK
contact

Time taken: 0.158 seconds, Fetched: 2 row(s)
hive> SELECT * FROM contact ORDER BY contactid DESC limit 5;
Query ID = hadoop_20200609210138_b4e0b374-ed7a-433e-82e5-52fee69560b7
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
2020-06-09 21:01:45,644 INFO  [eac715be-29f0-475c-9b9b-7cc8f61ef0bb main] client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2020-06-09 21:01:46,168 INFO  [eac715be-29f0-475c-9b9b-7cc8f61ef0bb main] client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
Starting Job = job_1591746326988_0001, Tracking URL = http://localhost:8088/proxy/application_1591746326988_0001/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1591746326988_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-06-09 21:02:06,248 Stage-1 map = 0%,  reduce = 0%
2020-06-09 21:02:16,114 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.78 sec
2020-06-09 21:02:26,752 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.44 sec
MapReduce Total cumulative CPU time: 5 seconds 440 msec
Ended Job = job_1591746326988_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.44 sec   HDFS Read: 1824675 HDFS Write: 617 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 440 msec
OK
19977   null    Crystal Hu      crystal21@adventure-works.com   1 (11) 500 555-0126     2003-11-16 22:00:00.0
19976   null    Crystal Zheng   crystal20@adventure-works.com   1 (11) 500 555-0171     2004-02-14 22:00:00.0
19975   null    Crystal He      crystal19@adventure-works.com   813-555-0148    2004-04-11 21:00:00.0
19974   null    Isabella        Richardson      isabella91@adventure-works.com  910-555-0166    2003-08-29 21:00:00.0
19973   null    Crystal Guo     crystal18@adventure-works.com   1 (11) 500 555-0171     2004-04-18 21:00:00.0
Time taken: 51.696 seconds, Fetched: 5 row(s)
hive> 
```

Figure 11 – Hive Records

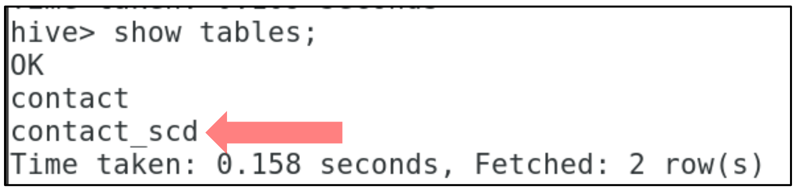
4.4 Creating SCD table on Hive

Our next step is to create the target table for our business problem, the table that will store Slow Changing Dimensions in Hive.

To create this table in Hive so that it is possible to carry out transactions between databases, it is mandatory that the table has transactional properties and that it is of the ORC (Optimized Row Columnar) type. The ORC format improves Hive's writing, reading and processing performance by providing an efficient way to store data.

```
CREATE TABLE datalake.contact_scd(
  ContactID int,
  Title string,
  FirstName string,
  LastName string,
  EmailAddress string,
  EmailAddress_OLD string,
  Phone string,
  Phone_OLD string,
  ModifiedDate date)
COMMENT 'This table holds the SCDs from Source RDBMS table named contact'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED as ORC
TBLPROPERTIES ("transactional"="true");
```

The command above creates our SCD table "contact_scd", inside the datalake, inserting columns "*** _ OLD" representing the record prior to the update thus making a Slow Changing Dimension Type 3.



```
hive> show tables;
OK
contact
contact_scd
Time taken: 0.158 seconds, Fetched: 2 row(s)
```

Figure 12 – Hive Records II

At that point we have an empty table and as we already load the preparation table (contact) with the columns we need, we can move the initial data into the “contact_scd” table and wait for new data to make adjustments to the dimensions as they happen.

To make it clear how SCDs work in practice during the design testing phase, I’ll manually enter some data instead of loading the source table from adventureworks.

```
INSERT INTO datalake.contact_scd VALUES
 (1,'Mr.','Gustavo','Achong','gustavo0@adventure-works.com','null','398-555-0132','null','2004-05-03'),
 (2,'Ms.','Catherine','Abel','catherine0@adventure-works.com','null','747-555-0171','null','2005-05-16'),
 (3,'Ms.','Kim','Abercrombie','kimmy2@yahoo-works.com','null','334','null','2005-05-07'),
 (4,'Sr.','Humberto','Acevedo','berto0@adventure-works.com','null','55-0127','null','2005-03-26'),
 (5,'Sra.','Pilar','Ackerman','pill@adventure-works.com','null','1 500 555-0132','null','2002-05-30'),
 (6,'Ms.','Frances','Adams','frn@sslv-works.com','null','991-555-0183','null','2001-09-26'),
 (7,'Ms.','Magret','Smith','margaret0@adv-works.com','null','959-555-0151','null','2000-01-22'),
 (8,'Ms.','Cala','Adams','carla0@adventure-works.com','null','107-555-0138','null','2001-01-11'),
 (9,'Mr.','Jay','Adans','jay1@gmail-works.com','null','158-555-0142','null','2000-11-02');
```

The information above was taken from the first lines of the source table but I made changes to some data, so when I import the original table (preparation table I imported from MySQL, which in turn came from the server of the company adventureworks) we will see Slow Changing Dimension going on.

ContactID	Title	FirstName	LastName	EmailAddress	EmailAddress_OLD	Phone	Phone_OLD	ModifiedDate
1	Mr.	Gustavo	Achong	gustavo0@adventure-works.com	null	398-555-0132	null	05/03/2004
2	Ms.	Catherine	Abel	catherine0@adventure-works.com	null	747-555-0171	null	05/16/2005
3	Ms.	Kim	Abercrombie	kimmy2@yahoo-works.com	null	334	null	05/07/2005
4	Sr.	Humberto	Acevedo	berto0@adventure-works.com	null	55-0127	null	03/26/2005
5	Sra.	Pilar	Ackerman	pill@adventure-works.com	null	1 500 555-0132	null	05/30/2002
6	Ms.	Frances	Adams	frn@sslv-works.com	null	991-555-0183	null	09/26/2001
7	Ms.	Magret	Smith	margaret0@adv-works.com	null	959-555-0151	null	01/22/2000
8	Ms.	Cala	Adams	carla0@adventure-works.com	null	107-555-0138	null	01/11/2001
9	Mr.	Jay	Adans	jay1@gmail-works.com	null	158-555-0142	null	11/02/2000

Table 5 - Table before Slow Changing Dimension

4.5 Slow Changing Dimension

We arrived at the critical point of the project, the moment when we are going to create instructions for SCDs to happen. There is no rule here, everything depends on the business problem, for our case we want to make comparisons between the source database and the target database that will store the SCDs in Hive.

To perform SCDs, Hive has the following instructions that assist and take the work to a higher level of efficiency during operations:

- INSERT
- UPDATE
- DELETE
- MERGE
- WHEN MATCHED / WHEN NOT MATCHED

The “WHEN MATCHED / WHEN NOT MATCHED” instruction is largely responsible for comparisons between database tables and it is from there that we take actions to adjust the dimensions of the Data Warehouse.

Here is the instruction I developed to perform a Slow Changing Dimension in our DW:

```
1->  MERGE INTO datalake.contact_scd AS TR
2->  USING datalake.contact AS SR
3->  ON TR.ContactID = SR.ContactID
4->  WHEN MATCHED AND hash(TR.Title, TR.FirstName, TR.LastName, TR.EmailAddress, TR.Phone) <>
         hash (SR.Title, SR.FirstName, SR.LastName, SR.EmailAddress, SR.Phone) THEN
5->  UPDATE SET
6->  Title = SR.Title, FirstName = SR.FirstName, LastName = SR.LastName,
     EmailAddress_OLD = TR.EmailAddress, EmailAddress = SR.EmailAddress,
     Phone_OLD = TR.Phone, Phone = SR.Phone,
     ModifiedDate = SR.ModifiedDate
7->  WHEN NOT MATCHED THEN
8->  INSERT VALUES (SR.ContactID, SR.Title, SR.FirstName, SR.LastName, SR.EmailAddress, 'null',
         SR.Phone, 'null', SR.ModifiedDate);
```

1-> Definition of the target table in which the SCD should be performed

2-> Definition of the table from the Source Database

3-> Definition of the column to be used as a comparison key between tables

4-> Instruction that will determine which comparisons should be taken into account when updating the table. The hash command is a smart way to make comparisons between multiple columns.

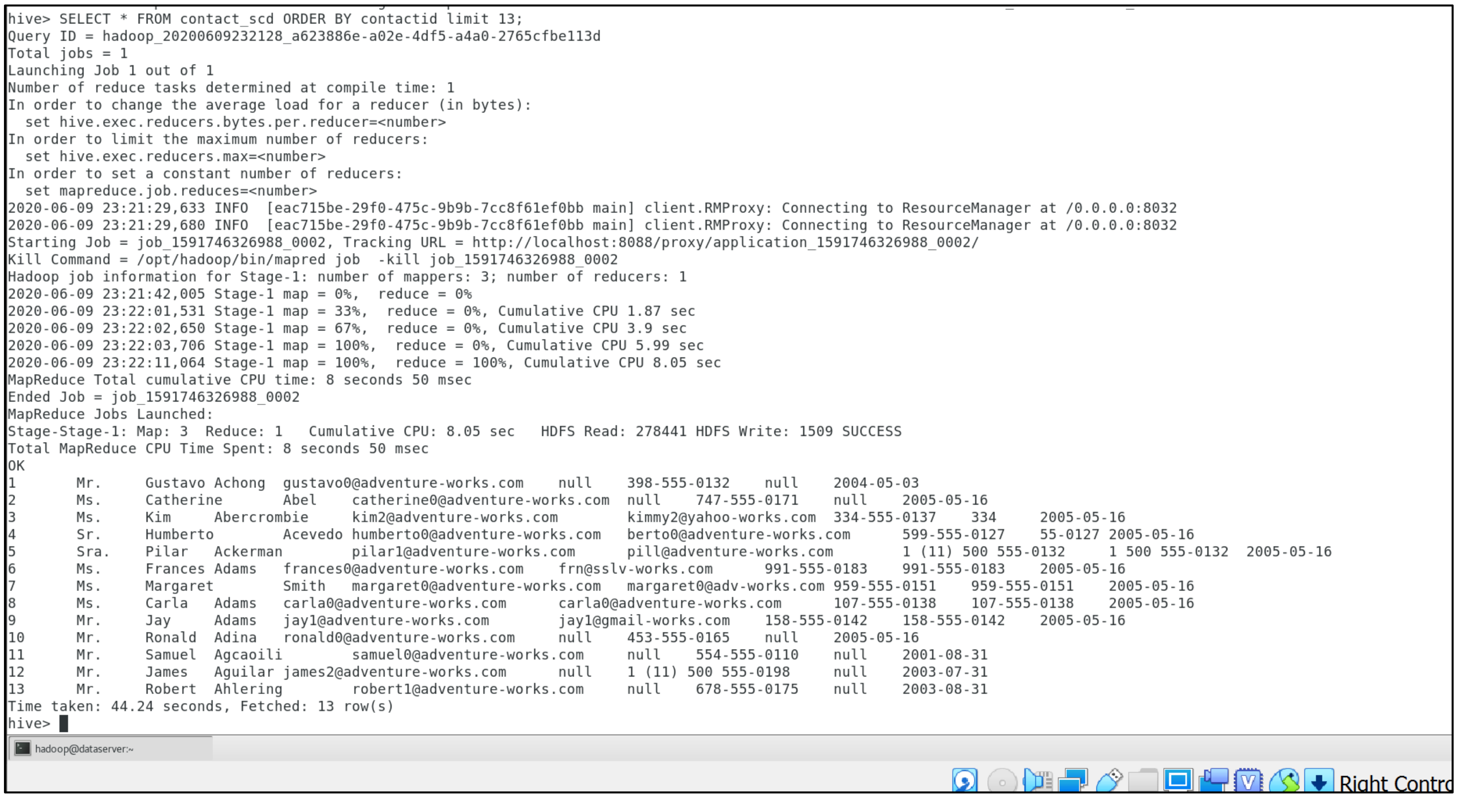
5-> Update instruction if the previous line is satisfied

6-> Information to be updated

7-> Instruction in case no record was found in the target table

8-> Insert new values as noted in the source table

With the insertion of the above commands in Hive we have a Slow Changing Dimension occurring in our data as we can see in the following table extracted from the datalake in Hive:



```
hive> SELECT * FROM contact_scd ORDER BY contactid limit 13;
Query ID = hadoop_20200609232128_a623886e-a02e-4df5-a4a0-2765cfbe113d
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
2020-06-09 23:21:29,633 INFO  [eac715be-29f0-475c-9b9b-7cc8f61ef0bb main] client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2020-06-09 23:21:29,680 INFO  [eac715be-29f0-475c-9b9b-7cc8f61ef0bb main] client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
Starting Job = job_1591746326988_0002, Tracking URL = http://localhost:8088/proxy/application_1591746326988_0002/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1591746326988_0002
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 1
2020-06-09 23:21:42,005 Stage-1 map = 0%,  reduce = 0%
2020-06-09 23:22:01,531 Stage-1 map = 33%,  reduce = 0%, Cumulative CPU 1.87 sec
2020-06-09 23:22:02,650 Stage-1 map = 67%,  reduce = 0%, Cumulative CPU 3.9 sec
2020-06-09 23:22:03,706 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.99 sec
2020-06-09 23:22:11,064 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.05 sec
MapReduce Total cumulative CPU time: 8 seconds 50 msec
Ended Job = job_1591746326988_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3  Reduce: 1   Cumulative CPU: 8.05 sec   HDFS Read: 278441 HDFS Write: 1509 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 50 msec
OK
1       Mr.     Gustavo Achong  gustavo0@adventure-works.com    null    398-555-0132    null    2004-05-03
2       Ms.     Catherine       Abel    catherine0@adventure-works.com  null    747-555-0171    null    2005-05-16
3       Ms.     Kim     Abercrombie     kim2@adventure-works.com        kimmy2@yahoo-works.com  334-555-0137    334     2005-05-16
4       Sr.     Humberto        Acevedo humberto0@adventure-works.com   berto0@adventure-works.com      599-555-0127    55-0127 2005-05-16
5       Sra.    Pilar   Ackerman        pilar1@adventure-works.com      pill@adventure-works.com        1 (11) 500 555-0132     1 500 555-0132  2005-05-16
6       Ms.     Frances Adams   frances0@adventure-works.com    frn@sslv-works.com      991-555-0183    991-555-0183    2005-05-16
7       Ms.     Margaret        Smith   margaret0@adventure-works.com   margaret0@adv-works.com 959-555-0151    959-555-0151    2005-05-16
8       Ms.     Carla   Adams   carla0@adventure-works.com      carla0@adventure-works.com      107-555-0138    107-555-0138    2005-05-16
9       Mr.     Jay     Adams   jay1@adventure-works.com        jay1@gmail-works.com    158-555-0142    158-555-0142    2005-05-16
10      Mr.     Ronald  Adina   ronald0@adventure-works.com     null    453-555-0165    null    2005-05-16
11      Mr.     Samuel  Agcaoili        samuel0@adventure-works.com     null    554-555-0110    null    2001-08-31
12      Mr.     James   Aguilar james2@adventure-works.com      null    1 (11) 500 555-0198     null    2003-07-31
13      Mr.     Robert  Ahlering        robert1@adventure-works.com     null    678-555-0175    null    2003-08-31
Time taken: 44.24 seconds, Fetched: 13 row(s)
hive> 
```

ContactID	Title	FirstName	LastName	EmailAddress	EmailAddress_OLD	Phone	Phone_OLD	ModifiedDate
1	Mr.	Gustavo	Achong	gustavo0@adventure-works.com	null	398-555-0132	null	05/03/2004
2	Ms.	Catherine	Abel	catherine0@adventure-works.com	null	747-555-0171	null	05/16/2005
3	Ms.	Kim	Abercrombie	kimmy2@yahoo-works.com	null	334	null	05/07/2005
4	Sr.	Humberto	Acevedo	berto0@adventure-works.com	null	55-0127	null	03/26/2005
5	Sra.	Pilar	Ackerman	pill@adventure-works.com	null	1 500 555-0132	null	05/30/2002
6	Ms.	Frances	Adams	frn@sslv-works.com	null	991-555-0183	null	09/26/2001
7	Ms.	Magret	Smith	margaret0@adv-works.com	null	959-555-0151	null	01/22/2000
8	Ms.	Cala	Adams	carla0@adventure-works.com	null	107-555-0138	null	01/11/2001
9	Mr.	Jay	Adans	jay1@gmail-works.com	null	158-555-0142	null	11/02/2000

ContactID	Title	FirstName	LastName	EmailAddress	EmailAddress_OLD	Phone	Phone_OLD	ModifiedDate
1	Mr.	Gustavo	Achong	gustavo0@adventure-works.com	null	398-555-0132	null	05/03/2004
2	Ms.	Catherine	Abel	catherine0@adventure-works.com	null	747-555-0171	null	05/16/2005
3	Ms.	Kim	Abercrombie	kim2@adventure-works.com	kimmy2@yahoo-works.com	334-555-0137	334	05/16/2005
4	Sr.	Humberto	Acevedo	humberto0@adventure-works.com	berto0@adventure-works.com	599-555-0127	55-0127	05/16/2005
5	Sra.	Pilar	Ackerman	pilar1@adventure-works.com	pill@adventure-works.com	1 (11) 500 555-0132	1 500 555-0132	05/16/2005
6	Ms.	Frances	Adams	frances0@adventure-works.com	frn@sslv-works.com	991-555-0183	991-555-0183	05/16/2005
7	Ms.	Margaret	Smith	margaret0@adventure-works.com	margaret0@adv-works.com	959-555-0151	959-555-0151	05/16/2005
8	Ms.	Carla	Adams	carla0@adventure-works.com	carla0@adventure-works.com	107-555-0138	107-555-0138	05/16/2005
9	Mr.	Jay	Adams	jay1@adventure-works.com	jay1@gmail-works.com	158-555-0142	158-555-0142	05/16/2005
10	Mr.	Ronald	Adina	ronald0@adventure-works.com	null	453-555-0165	null	05/16/2005
11	Mr.	Samuel	Agcaoili	samuel0@adventure-works.com	null	554-555-0110	null	08/31/2001
12	Mr.	James	Aguilar	james2@adventure-works.com	null	1 (11) 500 555-0198	null	07/31/2003
13	Mr.	Robert	Ahlering	robert1@adventure-works.com	null	678-555-0175	null	08/31/2003

Table 6 - Table After Slow Changing Dimension

We can see in Table 6 above our Slow Changing Dimension happening inside the Data Warehouse using Hive. All the information in the table was accessed through Hive, I just transported it to a more commercial viewing environment so that knowledge is consolidated.

Three things are happening in our SCD:

- First: looking, for example, at ‘ContactID 3’, we have a change in the “EmailAddress” and “Phone” record for that user. With this identification, the information that existed there becomes a previous version and is moved to the column “EmailAddress_OLD” and “Phone_OLD” according to the orange arrows in table 6. That way we keep a history of the last version before data has been changed.
- Second: based on the data version change, all new information for that user is inserted in the original columns, as can be seen in the green colors of table 6.
- Third: when comparing ContactID records does not return equal results, it means that the record does not exist in the table, so it must be inserted in a new line leaving the columns “*** _ OLD” without information just waiting for the moment when the table dimension changes, as indicated by the red color in table 6.

5. Final Considerations

The use of Apache Hadoop for a Big Data environment only brought advantages to the project. Using distributed storage and parallel processing helps to make the project more flexible and generate productivity without much bureaucracy with collected data. The most laborious and determining part is the initial cluster configurations, that is, adjusting all parameters takes time and attention.

After configuring the cluster, we installed Hadoop Ecosystem, emphasizing two main frameworks: Sqoop and Hive. Sqoop was used to move tables between databases and Hive to store structured data, or to structure unstructured data.

In this business problem, Slow Changing Dimension (SCD) was performed in a Data Warehouse with Hive for a database stored in MySQL. Initially this database was shared with the Virtual Machine, moved to MySQL installed on Linux and transported to a datalake in HDFS using an instruction from Sqoop where it besides moving the source table to HDFS, also inserted in Hive a table with only the columns under study, the columns that will undergo changes in their dimensions.

From the definition of this data and considering a Slow Changing Dimension Type 3 it was possible to build the table that will store the SCDs in Hive in order to store the version of the last change in a separate column.

With the table scheme created, the initial load was performed by manually entering some information to visualize the SCDs happening during the time. In order to generate updates in dimensions, HiveQL provides us with some INSERT, UPDATE, DELETE, MERGE, WHEN MATCHED / WHEN NOT MATCHED instructions, all of which are high-level and make the comparison work between the source table and the destination table flexible, allowing us to carry out rules of insertion.

With that it was possible to see a Slow Changing Dimension in practice in a Data Warehouse using Hive in a single table of the Source Database maintaining the versioning of previous data and updating the data according to the change made in the company AdventureWorks.

Following the same concept, we can apply more transformations and rules to the business problem, helping information management and data management that gradually change its dimensions.