Data Warehouses

INFO-H419

Part 2 : TPC-DI benchmark on Impala

BAKKALI Yahya (000445166)
FALLAHI Amirmohammad (000460073)
HAUWAERT Maxime (000461714)
LIBERT Alexandre (000435755)

December 2021

Université Libre de Bruxelles (ULB)

Contents

1 Introduction

The ongoing demand for modernisation in nowadays societies, have been obliged the data warehouse tools to supply high efficiency. In order to provide high efficiency, the **data integration** process of these tools can play an effective role.

Following the first part of the project, in this part also **TPC** tools have been used to verify efficiency of **Impala**. For this part of project, **TPC-DI** benchmark have been exploited; TPC-DI is a benchmark for data integration. In this paper the details of the TPC-DI experiment, performed on Impala, will be explained.

2 Data Integration & ELT

Data integration is set of processes in which data from several sources will be combined in order to create a connected, single view of data. **ETL** is a 3-step operation in order to Extract, Transform and Load data into the data warehouse. In fact ETL process can be used in order to integrate data. During ETL process the is collected from an initial system, then it will be converted into analyzable format, and placed into a data warehouse.

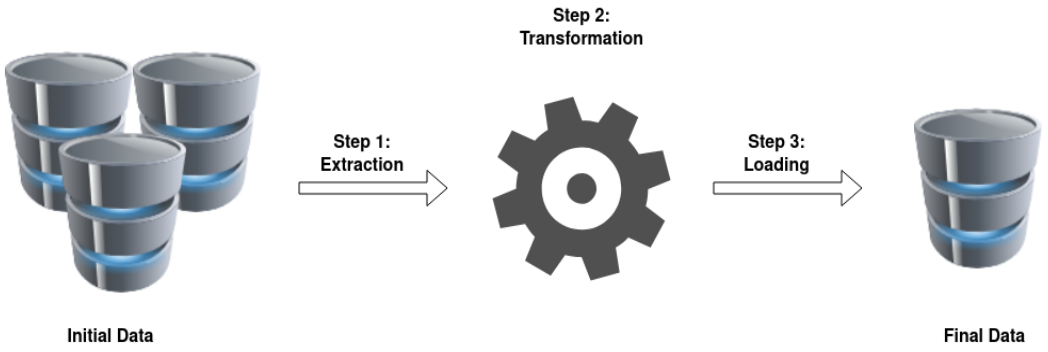


Figure 1: ETL process

The ETL steps can be defined as follow:

1. **Extraction:** In this step data will be extracted from different sources and will be stored in a "staging area" in order to avoid disorder in initial sources. This step validate the data and ensure that the data is accurate. At this stage, unnecessary data is removed, duplicates are identified and deleted. The data type is also examined and modified.
2. **Transformation:** Data extracted from data sources are raw and usually not ready for use and analysis. Raw data at this stage must be cleared and converted to the required format. This step is key step of ETL as it transform raw data to valuable and usable data. Second phase of data validation is also checked at this stage. If it is necessary to merge columns or separate columns and convert them into several columns, this will be done in this step.

3. **Loading:** Loading data to the data warehouse is the last step in the ETL process. Usually a large amount of data must be loaded in a short time (overnight) in the data warehouse, so it is very important to pay attention to performance optimisation. The data loading process may also fail and stop during execution. The recovery operation must be performed exactly from the stop point and the necessary actions must be taken to prevent non-integration and duplication or loss of data. At this stage, the necessary strategies to deal with such events must be planned.

3 Hardware specification

In fact, the same computer (*Dell xps-15-9560*) as the first part has been used in order to perform the TPC-DI benchmark experimentation. The following table shows the hardware specification of this computer.

Operating system	Ubuntu 20.04.3 LTS
System type	64 bits
CPU	Intel®Core™i7-7700HQ
CPU frequency	2.8 GHz up to 3.8 GHz
RAM capacity	16 GB
RAM type	DDR4
RAM speed	2,400 MT/s
Hard disk	PM981 NVMe Samsung
Hard disk capacity	256 GB

Table 1: Device information

4 Software Specification

As in the first part of the project, Impala structure and functionality have been described in details, in this paper these kinds of description have been ignored. So in this section, the TPC-DI tool will be verified.

4.1 TPC-DI

In this first part of the project, TPC-DS experiment has been performed on Impala; In this part, the TPC-DI experiment will be verified.

TPC-DI is a benchmarking test for technologies that transform and integrate data between systems. The benchmark procedure prepares data for utilization in a Data Warehouse by arranging a specified size of data. Data from an On-Line Transaction Processing (OTLP) system is transformed, together with data from other sources, and fed into a Data Warehouse in the benchmark model. The benchmark tests a wide range of system components related with data integration contexts, which would include:

- The loading and processing of enormous amounts of data
- A combination of transformation types like error checking, data type conversion, etc.

- Historical loading and cumulative updates of a target Data Warehouse using the transformed data
- Several data sources in various forms
- Other useful tests

The operations of the TPC-DI are represented as follows:

- TPC supplied code is used to produce source data. The data is transferred as flat files, similar to what many extraction programs provide.
- The data transformation process starts with the reading of the Source Data by the System Under Test (SUT).
- The transformations ensure that the Source Data is accurate and that the data is appropriately structured before being loaded into a Data Warehouse.
- When all Source Data has been transformed and is available in the Data Warehouse, the process is complete.

4.1.1 TPC-DI experimentation method

In order to illustrate better the experimentation method, the photo of official documentation of TPC-DI has been used. In fact the figure 2 illustrates the TPC-DI benchmark modelisation. Let's verify this model:

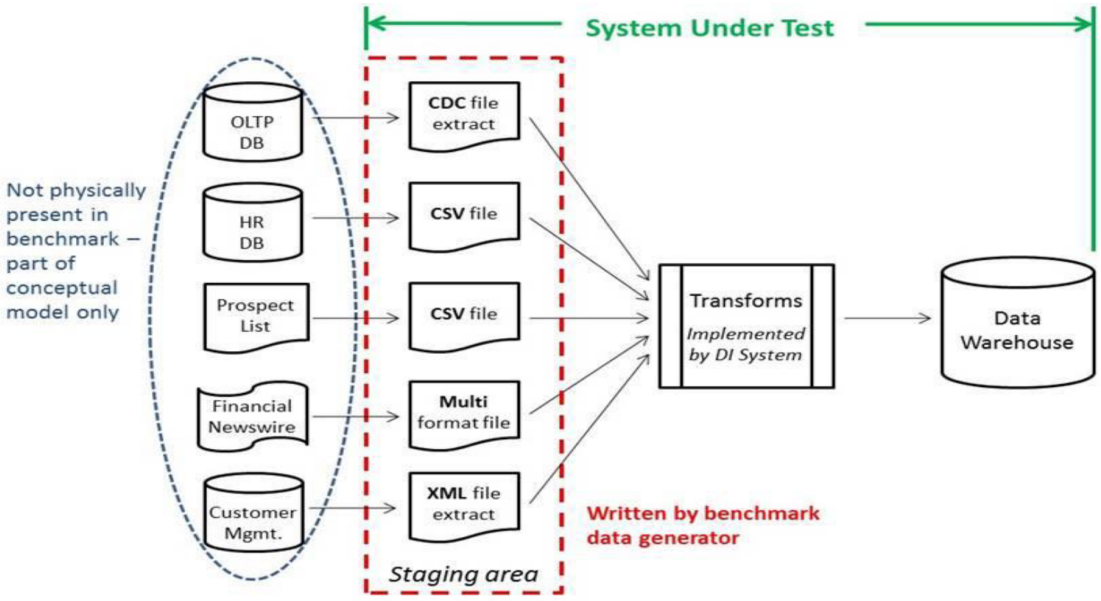


Figure 2: TPC-DI benchmark model

As the figure shows, first the data is extracted from source databases and added to staging area, then the transformation begins on extracted data and finally the transformed data will be loaded on data warehouse.

The procedure of extracting files from data sources to the staging area is not tested by the TPC-DI because it uses its generative scripts to produce these files automatically.

As it has been explained in the previous sections, the key phase of ETL process is transformation. In this phase, unnecessary columns will be removed, different kinds of definition for same data will be homogenised, the units will be unified and in general converting data from character representations to data types that comply with the Data Warehouse standard.

Developing a data warehouse for commercial or scientific exploitation, consists of two main phases:

- **Historical load:** Which is simply the record of already existed data, for instance the transactions of previous years.
- **Incremental update:** Which is the technique of loading data progressively. The destination database only receives new and modified data. Non-changing data will be left alone.

In this project, only *historical load* phase has been verified.

Now let's take a look at the figure which has been provided by TPC-DI official documentation for the destination data model.

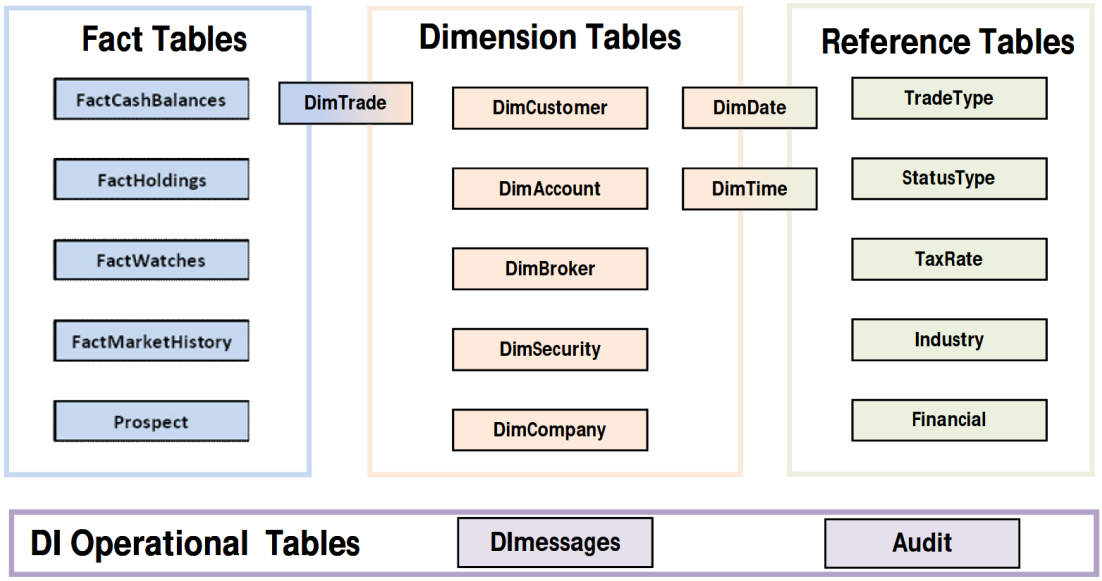


Figure 3: Destination Data Model

A dimensional data warehouse is where the TPC-DI workload ends up. Dimensional models are widely used in the industry and are meant to provide quick answers to a number of business concerns. There are numerous methods to create a data warehouse, however this format gives a well-understood structure in the benchmark Data Warehouse while also allowing the TPC-DI workload to apply a diverse set of data transformations.

According to TPC-DI official documentation, in a dimensional model, dimension tables defines the business

entities of interest and fact tables provide measurement data about what happened, such as the price and volume of a transaction.

4.2 Apache Impala vs Apache Hive

Hadoop, main component, is HDFS (Hadoop Distributed File System) and MapReduce. MapReduce involves breaking into small batches a large volumes of data, and then process them separately In the TPC_DS project we saw how to install and run Impala.

4.2.1 MPP vs MapReduce

MPP and MapReduce are used in different scenarios. MPP is used on expensive, specialized hardware tuned for CPU, storage and network performance. MapReduce and Hadoop find themselves deployed to clusters of commodity servers that in turn use commodity disks. The commodity nature of typical Hadoop hardware (and the free nature of Hadoop software) means that clusters can grow as data volumes do, whereas MPP products are bound by the cost of, and finite hardware in, the appliance and the relative high cost of the software. [1]

4.2.2 Impala vs Hive Queries

Hive queries translate to MapReduce jobs. Impala queries implement memory bound MPP jobs that are much faster than disk based MapReduce. Unlike Hive, Impala doesn't rely on data transformations or moving data to process queries. Impala also avoids startup overhead by starting daemon processes at boot time. This makes Impala more readily available to process queries than Hive since Hive has to start new processes for every query you run. Since Hive is MapReduce based, it adds a level of fault-tolerance that Impala can't match. While Impala makes querying a lot faster, it loses the added advantage of fault-tolerance provided by Hadoop MapReduce jobs. [2]

4.2.3 Read Only Schema

Hive is "Schema on READ only". So, functions like the update, modifications, etc. don't work with this. Because the Hive query in a typical cluster runs on multiple Data Nodes. So it is not possible to update and modify data across multiple nodes.(Hive versions below 0.13)[3]

4.2.4 Impala no Update

As explained, the idea of read-only in Hive also applies to Impala. That' s why, in this project, we could not perform any data update in the tables.

[1] https://www.zdnet.com/article/mapreduce-and-mpp-two-sides-of-the-big-data-coin/
[2] https://www.stackchief.com/blog/Hive%20vs%20Impalal
[3] https://www.guru99.com/introduction-hive.html

4.3 Pentaho Data Integration

Pentaho is a piece of software which allows everyone to perform business intelligence tasks on large quantities of data. It covers the common areas of ETL, reporting, OLAP/analysis and data mining. The software is developed since 2004 entirely in Java which allows cross-platform support for the popular operating systems, at the beginning by Pentaho Corporation, then Pentaho was acquired by Hitachi Data Systems in 2015. [4]

4.3.1 Introduction to Pentaho

Pentaho Data Integration[5](PDI) is a software of Pentaho which performs data integration. It is an ETL solution which means that it can extract, transform and load data from various sources. Pentaho also provides a desktop application called Spoon, which allows anyone to perform these tasks with an intuitive graphical interface(the PDI desktop is called Spoon).

4.3.2 Use cases

Pentaho Data Integration(PDI) is suitable for several use cases[6]:

- Having to insert large quantities of data into databases while using the full power of computation of the available pieces of hardware.
- Having to populate a data warehouse without having problems with creating surrogate keys as well as slowly changing dimensions.
- Having to clean or correct data with simple or complex computations.
- Having to migrate data from one database to another.
- Having to deal with data arriving in real-time.
- Having to perform several Hadoop functions such scheduling and executing jobs, designing Hadoop MapReduce.
- Having to prototype Relational OLAP schemas.

4.3.3 PDI structure

The Data Integration allows the use of two basic document types:

- Transformations : used to describe the ETL data flows, e.g. reading input from a source file, then transforming these data and loading them into a target location(database, file, ...).
- Jobs : used to coordinate ETL activities such as defining the order of execution of the transformations flow and dependencies, or checking for execution conditions, before execution e.g. I/O availability...

[4] https://www.hitachivantara.com/en-us/company.html
[5] https://help.hitachivantara.com/Documentation/Pentaho/5.2/0J0/0C0/000
[6] https://help.hitachivantara.com/Documentation/Pentaho/5.2/0J0/0C0/000

PDI transformation use two main components:

- Steps : They are the building blocks, each step is designed to perfom a specific task, e.g. input/output file or database, filter rows... and more than 140 others
- Hops : they are the data pathways between steps

4.3.4 Connect Cloudera Impala to Pentaho

Impala offers connectors "Impala JDBC Connector 2.6.X for Cloudera Enterprise" to allow connection between the database and Pentaho DI, it can be found in their official site [7]. Pentaho can not insert efficiently on Impala database, it suggests to use Hadoop cluster instead.

5 Implementation

In this section, details of implementation and experimentation will be seen. This section shows how the input of the experimentation has been provided and also explains how the situation of the experimentation has been facilitated for users. It will also verify different level of experimentation.

As in the first part of the project, the handling of Impala was explained in depth, we will ignore it here.

5.1 Preparing TPC-DI environment

Once the TCP-DI tool was downloaded from TPC official website[8], the experimentation environment can be prepared. In this subsection, the preparation for experiment environment will be verified.

5.1.1 Source data preparation

The TPC-DI tool gives the possibility to create the source data; The source data will be generated by launching the following commands after extracting the file `tpc-di-tool.zip` and going to Tools directory.

```
$ mv PDGF pdgf
$ java -jar DIGen.jar
```

5.1.2 Data warehouse preparation

In order to provide the data warehouse, just the file `create_table_fk.sql` has been launched by following command:

```
$ bin/impala-shell.sh -f <path_to_the_create_tables_fk.sql>/create_tables_fk.sql
```

[7] https://www.cloudera.com/downloads/connectors/impala/jdbc/2-6-15.html
[8] http://tpc.org/tpc_documents_current_versions/current_specifications5.asp

The result is as follows:

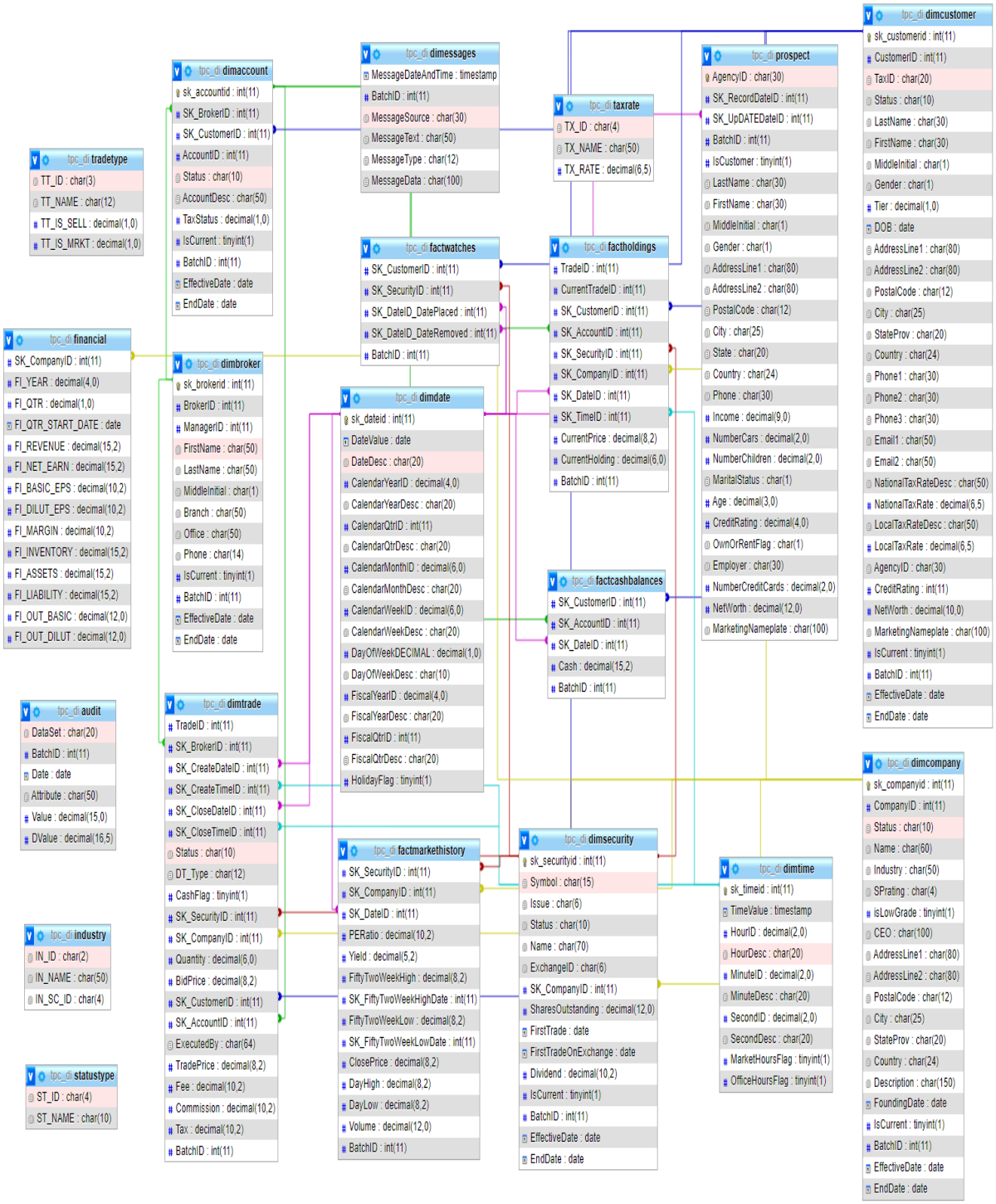


Figure 4: Data warehouse

5.1.3 Preparing Pentaho Data Integrator

After downloading Pentaho from the official website [9], by launching the file `spoon.sh` the Pentaho graphical user interface will be opened and will be ready to begin the experiment.

5.2 Historical Load

In this section, the ETL process of each table will be shown in details. There are tables which have been loaded easily, but there are also tables that their historical load depends on other tables. The complete process for historical load and the result of this process can be seen below:

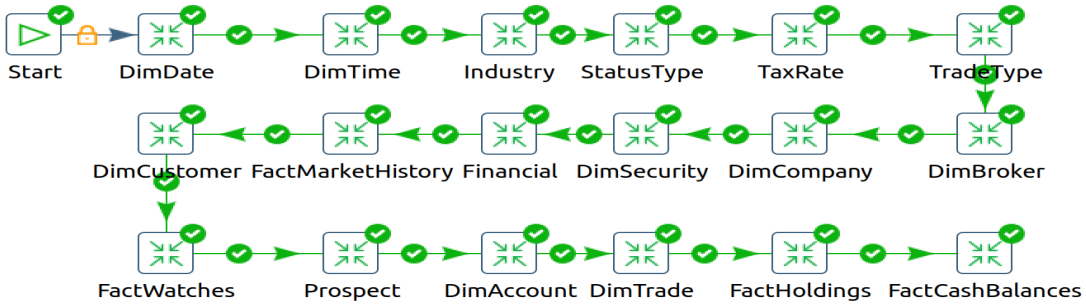
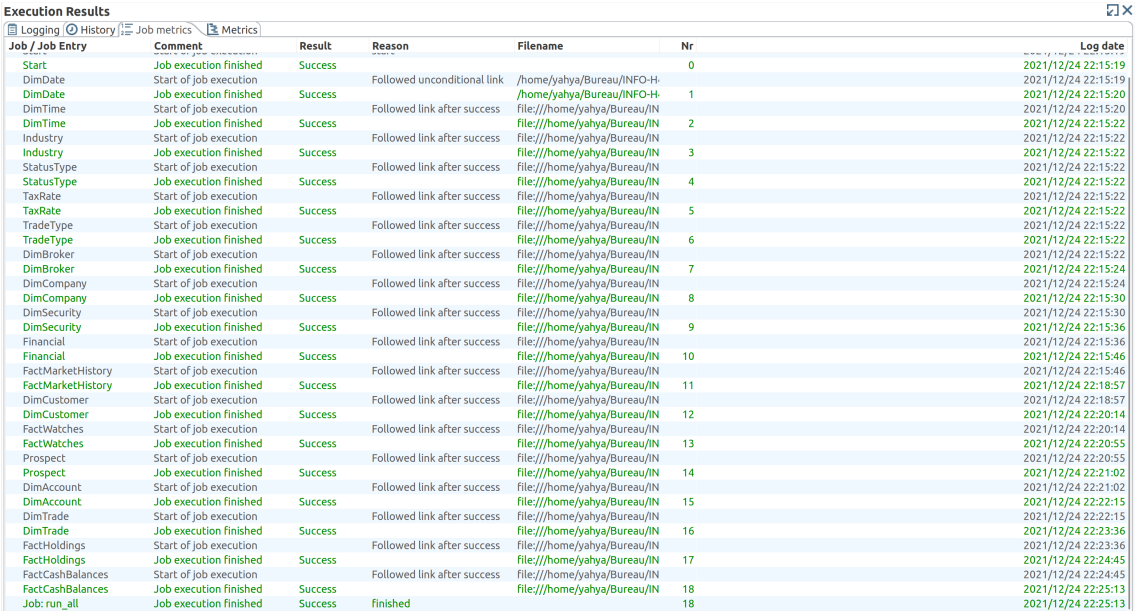


Figure 5: Historical load process


Execution Results

Logging | History | Job metrics | Metrics

Job / Job Entry	Comment	Result	Reason	Filename	Nr	Log date
Start	Job execution finished	Success			0	2021/12/24 22:15:19
DimDate	Start of job execution		Followed unconditional link	/home/yahya/Bureau/INFO-H		2021/12/24 22:15:19
DimDate	Job execution finished	Success		/home/yahya/Bureau/INFO-H	1	2021/12/24 22:15:20
DimTime	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:15:20
DimTime	Job execution finished	Success		file:///home/yahya/Bureau/IN	2	2021/12/24 22:15:22
Industry	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:15:22
Industry	Job execution finished	Success		file:///home/yahya/Bureau/IN	3	2021/12/24 22:15:22
StatusType	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:15:22
StatusType	Job execution finished	Success		file:///home/yahya/Bureau/IN	4	2021/12/24 22:15:22
TaxRate	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:15:22
TaxRate	Job execution finished	Success		file:///home/yahya/Bureau/IN	5	2021/12/24 22:15:22
TradeType	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:15:22
TradeType	Job execution finished	Success		file:///home/yahya/Bureau/IN	6	2021/12/24 22:15:22
DimBroker	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:15:22
DimBroker	Job execution finished	Success		file:///home/yahya/Bureau/IN	7	2021/12/24 22:15:24
DimCompany	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:15:24
DimCompany	Job execution finished	Success		file:///home/yahya/Bureau/IN	8	2021/12/24 22:15:30
DimSecurity	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:15:30
DimSecurity	Job execution finished	Success		file:///home/yahya/Bureau/IN	9	2021/12/24 22:15:36
Financial	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:15:36
Financial	Job execution finished	Success		file:///home/yahya/Bureau/IN	10	2021/12/24 22:15:46
FactMarketHistory	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:15:46
FactMarketHistory	Job execution finished	Success		file:///home/yahya/Bureau/IN	11	2021/12/24 22:18:57
DimCustomer	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:18:57
DimCustomer	Job execution finished	Success		file:///home/yahya/Bureau/IN	12	2021/12/24 22:20:14
FactWatches	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:20:14
FactWatches	Job execution finished	Success		file:///home/yahya/Bureau/IN	13	2021/12/24 22:20:55
Prospect	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:20:55
Prospect	Job execution finished	Success		file:///home/yahya/Bureau/IN	14	2021/12/24 22:21:02
DimAccount	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:21:02
DimAccount	Job execution finished	Success		file:///home/yahya/Bureau/IN	15	2021/12/24 22:22:15
DimTrade	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:22:15
DimTrade	Job execution finished	Success		file:///home/yahya/Bureau/IN	16	2021/12/24 22:23:36
FactHoldings	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:23:36
FactHoldings	Job execution finished	Success		file:///home/yahya/Bureau/IN	17	2021/12/24 22:24:45
FactCashBalances	Start of job execution		Followed link after success	file:///home/yahya/Bureau/IN		2021/12/24 22:24:45
FactCashBalances	Job execution finished	Success		file:///home/yahya/Bureau/IN	18	2021/12/24 22:25:13
Job: run_all	Job execution finished	Success	finished		18	2021/12/24 22:25:13

Figure 6: Historical load process result

[9] https://sourceforge.net/projects/pentaho/files/latest/download

5.2.1 Type read & load

These tables are static; The data will be read from the file and will be loaded in the destination without any modifications. Below is an example for implementation of these kinds of table:

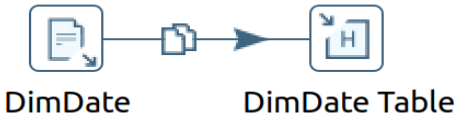


Figure 7: Implementation of tables type read & load

The following tables are the tables of type read & load:

DimTime	DimDate	StatusType
TaxRate	Industry	TradeType

5.2.2 DimBroker

Data of this table have been extracted from `HR.csv` file; From this file, those employees who are brokers (EmployeeJobCode = 314) will have data copied to the Broker table.

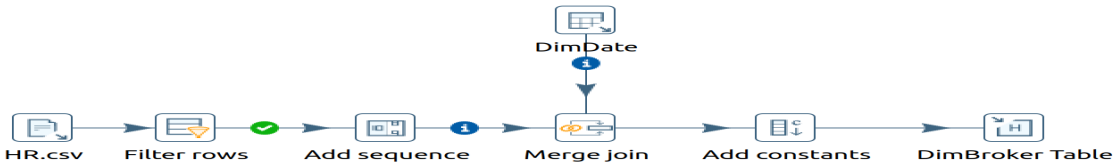


Figure 8: Implementation of DimBroker table

5.2.3 DimCompany

The `FINWIRE` files are used to extract DimCompany data. The records of type CMP are used to process all `FINWIREyyyyQq` files in ascending year and quarter sequence. Notice that in its implementation also some connection with StatusType and Industry tables have been done.

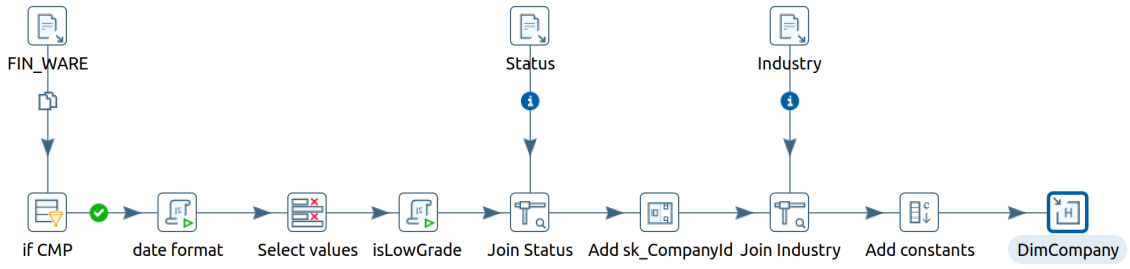


Figure 9: Implementation of DimCompany table

5.2.4 DimSecurity

The `FINWIRE` files are used to extract DimCompany data. The records of type CMP are used to process all `FINWIREyyyyQq` files in ascending year and quarter sequence, and records of type SEC are used. It is linked with DimCompany and StatusType.

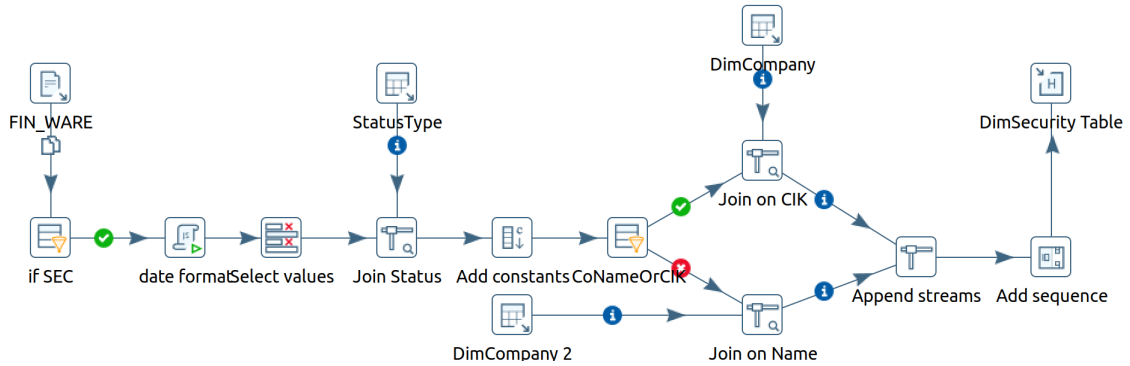


Figure 10: Implementation of DimSecurity table

5.2.5 Financial

The `FINWIRE` files are used to extract DimCompany data. The records of type CMP are used to process all `FINWIREyyyyQq` files in ascending year and quarter sequence, and records of type FIN are used. There is a connection with DimCompany in its implementation.

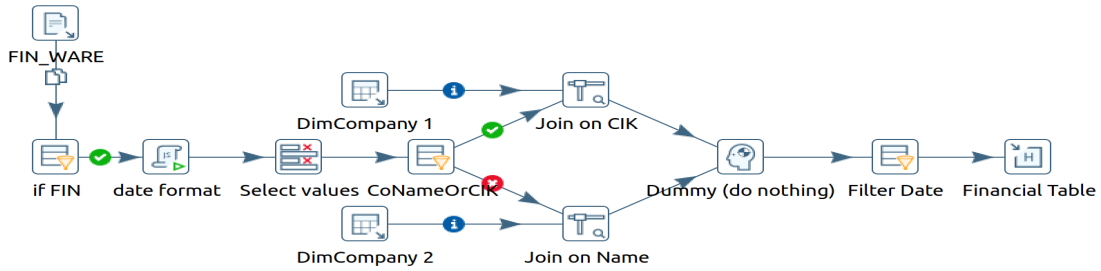


Figure 11: Implementation of Financial table

5.2.6 FactMarketHistory

The data for this table is extracted from thr file `DailyMarket.txt`. The challenging aspect for this table was grouping the entries by date and security ID while keeping the date of highest value and lowest value for both of them. As following figure shows, this problem has been handled.

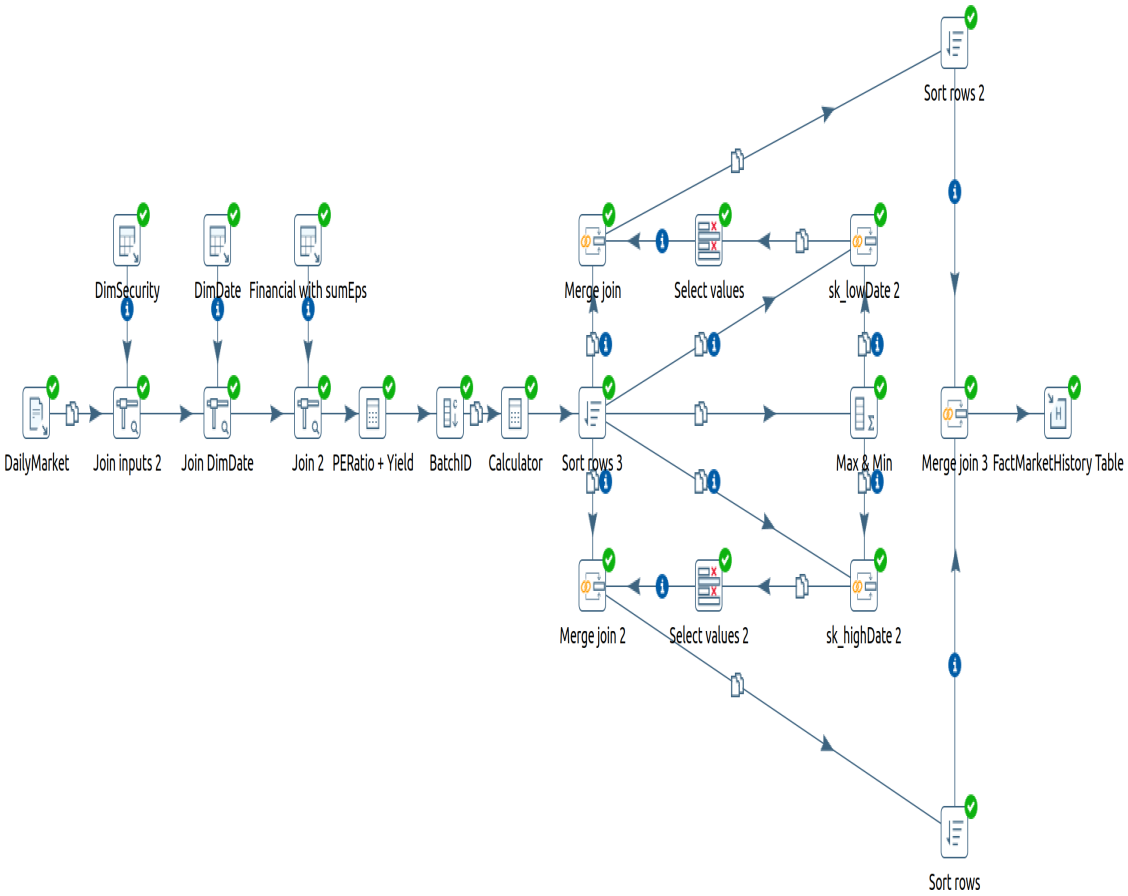


Figure 12: Implementation of Financial table

5.2.7 DimCustomer

The data of this table is extracted by using the file `CustomerMgmt.xml`. Customer data is kept as a sub-part of the Action element. Every action will have a Customer associated with it, but not all actions will necessitate changing the DimCustomer table.

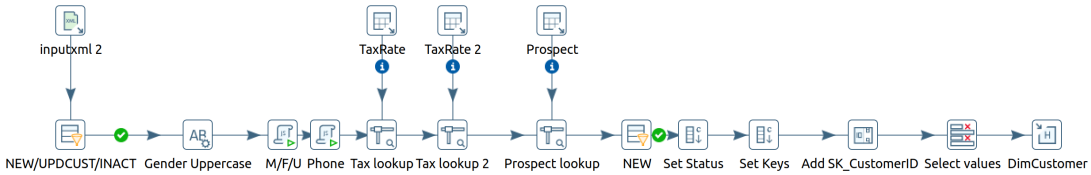


Figure 13: Implementation of DimCustomer table

5.2.8 FactWatches

Data for FactWatches extracted from the `WatchHistory.txt` file. For the Customer, Security, and Date dimension references, surrogate keys must be acquired. The date keys show when the watch was put on and taken off.

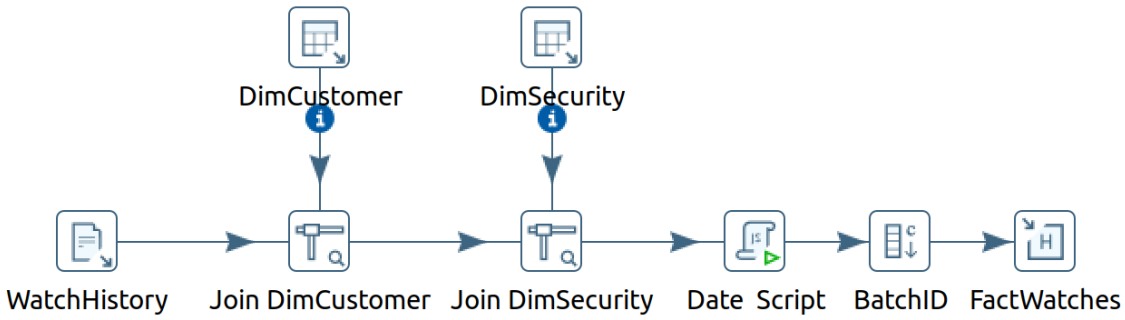


Figure 14: Implementation of FactWatches table

5.2.9 Prospect

Prospect data have been extracted from `Prospect.csv`. The only matter to handle for this table was to verify if a customer exists (or does not).

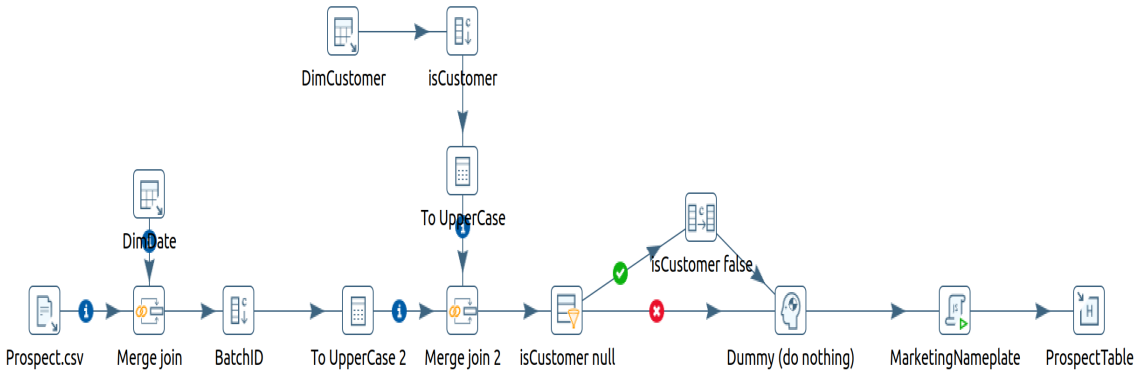


Figure 15: Implementation of Prospect table

5.2.10 DimAccount

DimAccount data is also based on `CustomerMgmt.xml` like DimCustomer data; The major difference with DimCustomer is that DimAccount must be linked to the relevant broker.

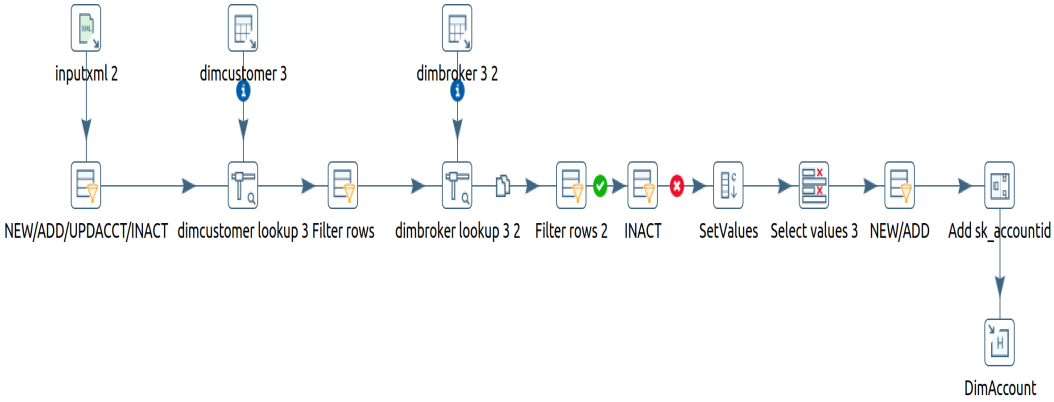


Figure 16: Implementation of DimAccount table

5.2.11 DimTrade

DimTrade data is extracted form `TradeHistory.txt` and `Trade.txt`. On the basis of the T_ID field, the incoming files may be logically linked. A new DimTrade record is created if a T_ID encountered does not match a TradeID from the DimTrade table. The DimTrade record is updated when a T_ID is found that matches one of the TradeIDs in the DimTrade database.

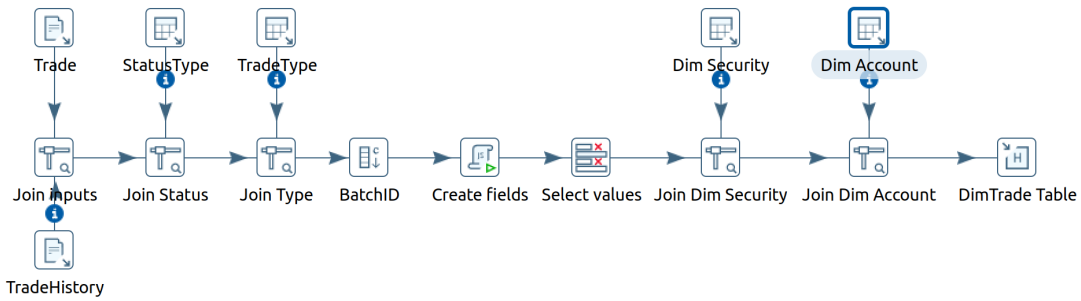


Figure 17: Implementation of DimTrade table

5.2.12 FactHoldings

The `HoldingHistory.txt` file and the DimTrade table provide data for FactHoldings. The number and price values represent a security's holdings following the most recent exchange.

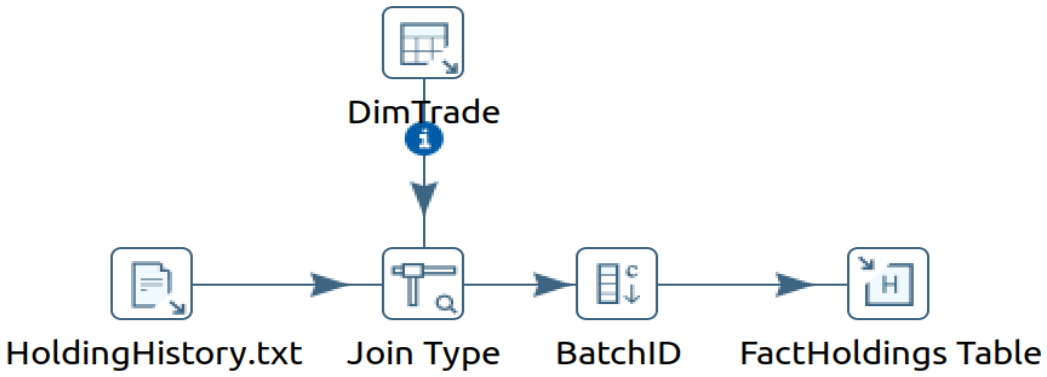


Figure 18: Implementation of FactHoldings table

5.2.13 FactCashBalances

FactCashBalances data is extracted from the file `CashTransaction.txt`. The net effect of all cash transactions for a specific account on a given day is summed, and only one record is created per account with changes on that day.

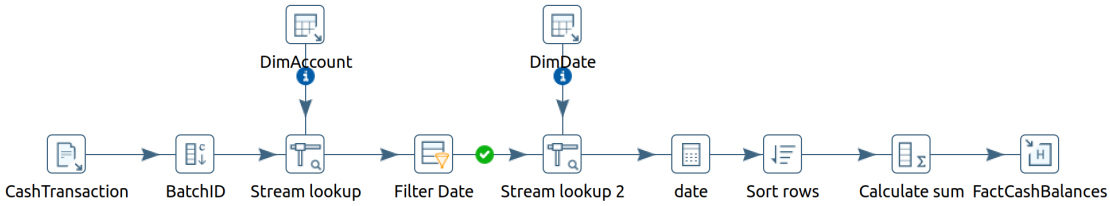


Figure 19: Implementation of FactCashBalances table

5.3 Automatic Auditing

After all ETL tasks have been completed and the database is populated. We should perform automated audit queries to verify the accuracy of all transformations. And check the accuracy and consistency of the data.

Prof. Esteban Zimanyi provided us with 4 scripts via email to audit our results. since Impala syntax and dataypes are limited and strict, it was not easy to adapted them. Thus, all 4 scripts failed.

6 Results and benchmarks

In this section, the different benchmarks done as well as their results and some analysis will be seen. It has been decided to set the size of the data-set to 1 GB. The table shows the execution time that an ETL process takes for each table and the figure illustrates the time differences.

raw	Table	Execution time
1	Industry	0.0s
2	StatusType	0.0s
3	TaxRate	0.0s
4	TradeType	0.1s
5	DimDate	0.9s
6	DimTime	2.2s
7	DimBroker	2.5s
8	Prospect	3.4s
9	DimCompany	5.9s
10	DimSecurity	6.4s
11	Financial	14.1s
12	FactCashBalances	26s
13	FactWatches	43.2s
14	DimAccount	78s
15	DimTrade	81s
16	DimCustomer	88s
17	FactHoldings	202s
18	FactMarketHistory	210s

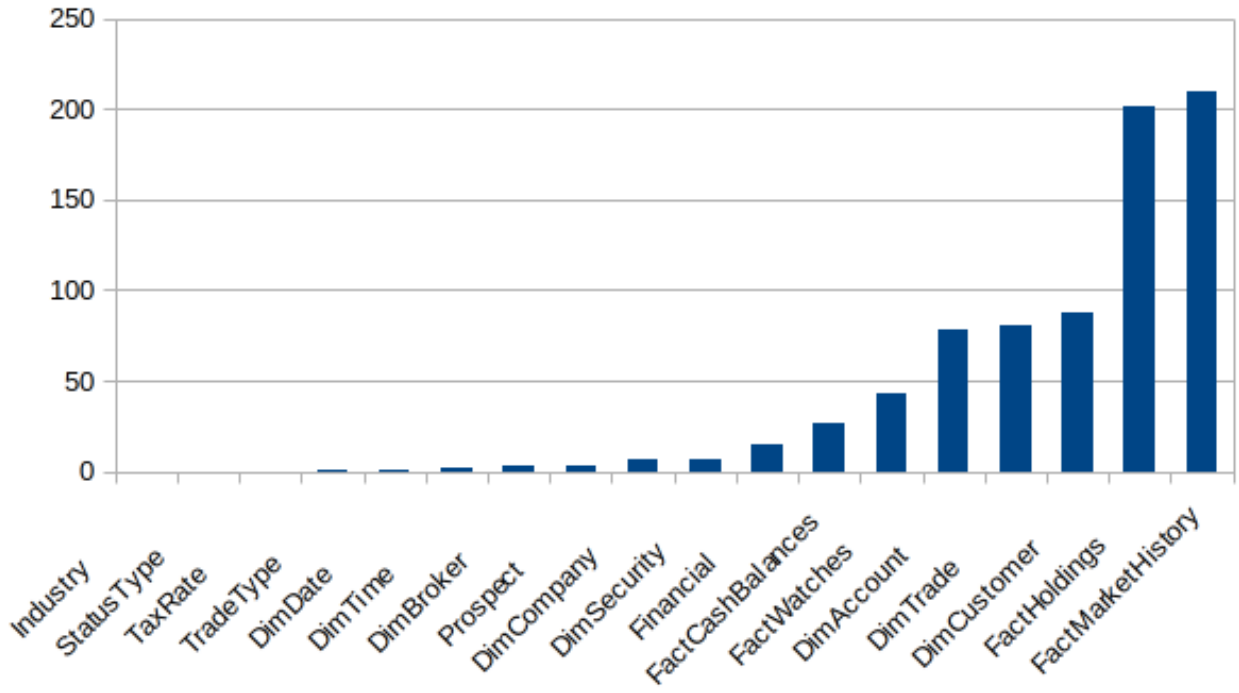


Figure 20: Execution time in seconds for each table ETL process

Note that for dimCustomer and dimAccount, only insertions have been performed. The obtained results show that it takes 10 min to perform the historical load. Thus, taking into account the updates or the modification that should be done could increase a lot the time needed.

7 Conclusion

In this project, *TPC-DI* tools have been used in order to get a data integration benchmark of Apache Impala. In this report, the necessary steps and critical points of experimentation described so that anyone can be able to reproduce the obtained results.

A challenging thing was that the documentation for TPC-DI which had not a good quality (the last update was for 2014).

It is also noteworthy to mention that Pentaho provides an excellent services for data integration and it is really simple to work with this tool.

The main issue was the Read-Only aspect of Impala. Since we were not able to perform any Update of the Inserted Data, we could not implement some part of TPC-DI and could not perform the audit Part. However Impala is suitable for large data warehouses, as it is capable of holding terabytes of data.

References

[1] *Transaction Processing Performance Council (TPC) - TPC BENCHMARK™ DI Standard Specification — Version 1.1.0, November 2014*