Université Libre de Bruxelles

INFO-H419 - Data Warehouses

Implementation of the TPC-DI benchmark on SQL Server

Brahim Amssafi, Soumoy Astrid

December 24, 2020

Table Content

1 Introduction

In order to practice our knowledge and competences in the class *Data warehouses*, we have to do a project which is divided in two parts. Two benchmark will be implemented, a TPC-DS and a TPC-DI. To implement them, we choose to use SQL Server. In this part, the main focus will be on the TPC-DI benchmark.
First of all, we will give some details about what is a TPC-DI benchmark. Then, we will explained what is SQL Server Integration Services. We will continue with how we did our implementation. Finally, you will find the results of the benchmark followed by a discussion about them.

2 The TPC-DI benchmark

2.1 What is a benchmark

In general, the term benchmarking is when you want to compare things based on a standard. In computer science, a benchmarck is an evaluation of a software, hardware or computer. This process will measure the performance, based on conditions that you have pre-defined, to help you compare specifics products. The goal is to determine which product will be the best to execute a specific task. It's also a good way to resolve implementation problems.

2.2 Specifications of the TPC-DI

The TPC-DI is a benchmarck specialized for Data Integration (DI).

The objective of Data Integration is to extract and combined data that comes from multiple source. This data can have different format, so it wants to regroup all of them in the same data store. The Data Integration will unified the data format into a new data model. It will result by a coherent database system for internal and external users.

Data integration will typically support two kinds of integration : the history load and the incremental load. The difference between both is that the history load wants to load data one time and to keep them during a long time. It will allows user to compare data based on the data they had at the beginning of the implementation. The incremental load is the opposite, we wants to load data periodically for example every day, every week or every month. The purpose is to keep the data updated with the most recent data.

The role of the TPC-DI benchmarck is to merge the data together and transform this data. After this transformation to homogenize this data, it will be loaded into a data warehouse. The benchmarck will capture the diversity and the complexity of typical Data Integration tasks. The tasks can be to load a huge amount of data, to update data by incrementation, transform specific data types, etc. TPC-DI has also requirements that the vendors needs to respect. When an industry develop a tool and wants to show its efficiency, they have to implement their own TPC-DI benchmarck with specifications. They needs to proof that their benchmarck respect the requirements so that it is a fair way to compare the tools between vendors.

On the following figure, you can see the TPC-DI benchmarcked system. The staging area contains various data format such as CSV file or XML file. Their is two main parts in the

TPC-DI the source data model and the target data model. As we said, the target should be harmonized but the source data model can have multiple format.

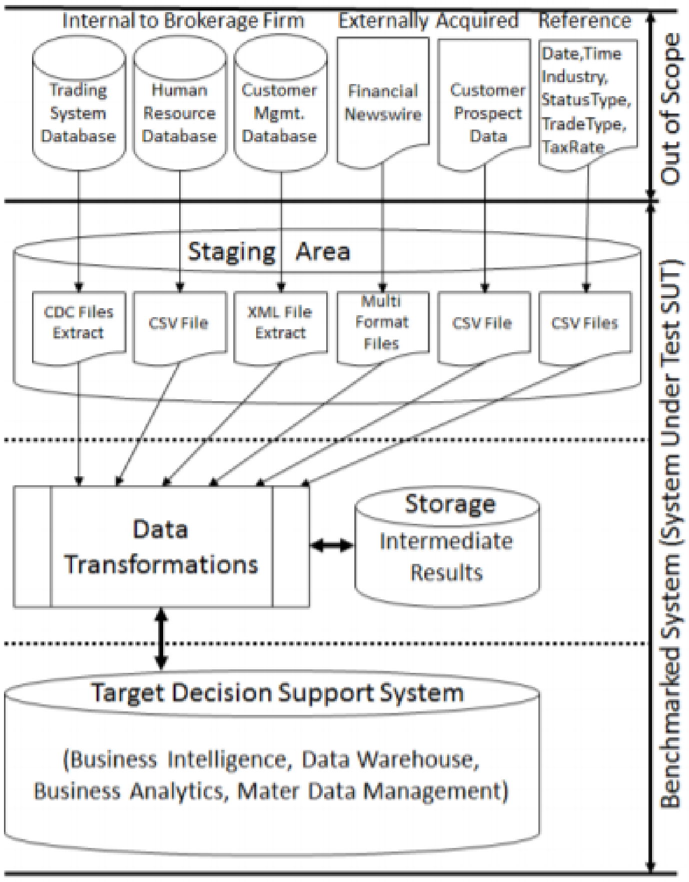


Figure 1: Benchmarked System and Workflow

Advantages of having a TPC-DI benchmarck :

- It is useful for the costumer that wants to compare Data integration tools. For example, if a company need one of this tool, but have multiple vendors, it can help them to evaluate those tools and there efficiency.

- TPC-DI is the first industry standard benchmark that allows hardware and software vendors to illustrate the performance of what they have developed in a comprehensive, competitive, fair and open way.

- It has an hybrid approach, it includes both synthetic and real world data.

3 SQL Server Integration Services

For the TPC-DS benchmarck, we used SQL Server Analysis Services (SSIS) but the TPC-DI benchmarck will be implemented on the SQL Server Analysis Services.
SQL Server Integration Services is a Microsoft tool developed to make data integration and data transformation solution. It is the tool we needed to use to test the Data integration for the database SQL Server. This tool has all the functionalities we talked about such as data transformation from different sources like xml or relational data. The advantage is that it is a recent tool and it contains some graphical tools that can be useful in benchmarking too.

Here you can see the main idea of the SSIS. You have source data, database, files and you send the into ETL (that is the previous name for TPC-DI). And the result is target data.

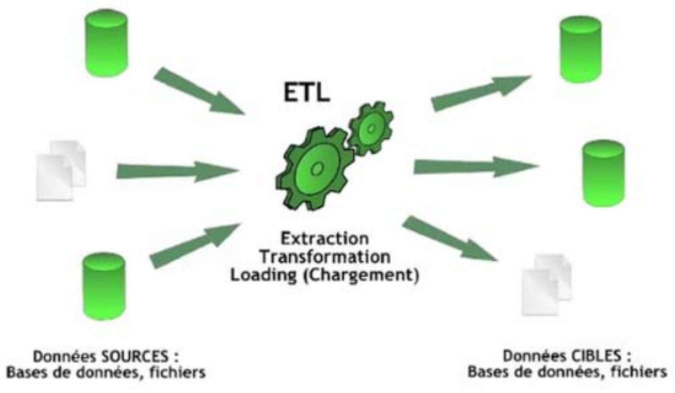


Figure 2: SSIS Schema

4 Implementation

In this section, we will explain step by step how we implement the TPC-DI Benchmark on SQL server.

4.1 Installation

To install the tool, the first thing to do is to download the TPC-DI zip file from the website [1]. Then, you need to extract the documents. Once the file unzipped, you just have to follow those few steps :

1. Since we choose to implement the benchmarck with SQL Sever, you need to download it, also from a website[2].

[1]`http://www.tpc.org`
[2]`https://www.microsoft.com/fr-fr/sql-server/sql-server-downloads`

2. Install the SSIS[3] because it is the Integration Services component required for Sql Server.
3. Install Microsoft Visual Studio 2005.
   (Or a later version).
4. Install Sql Server Management Studio[4].
   (Or a later version).

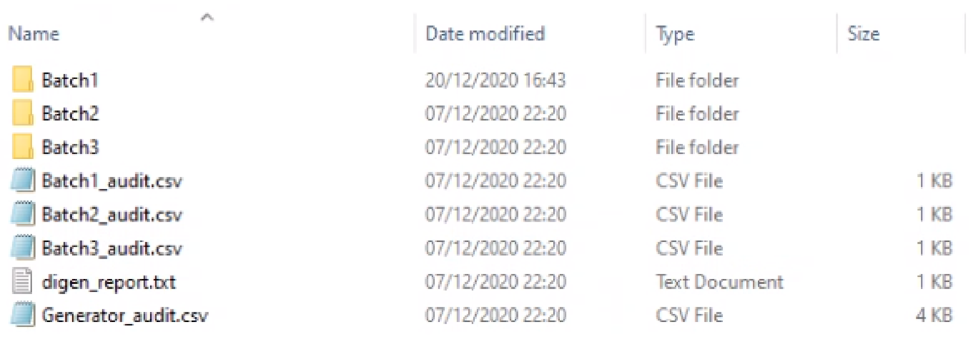
4.2 Generate data

In order to generate the data, we extract the TPC-DI zip file and inside it we used a specific command :

```
    java -jar DIGen.jar -o outputdir -s 3
```

- DIGEN.jar : is the jar executable that will generate the data.
- -s : this option is used to specified the scale factor. Here we put 3 as scale factor because we just need 3GB of data. We thought it was the more appropriate.
- -o : with this option, you give a filename (here "outputdir" for example) that will be used to create the output folder that will contain the data that we have generated.

We entered this command inside the main folder of the extract zip file of TPC-DI because it contains the jar executable. Once this command done, the folder created should look like this :



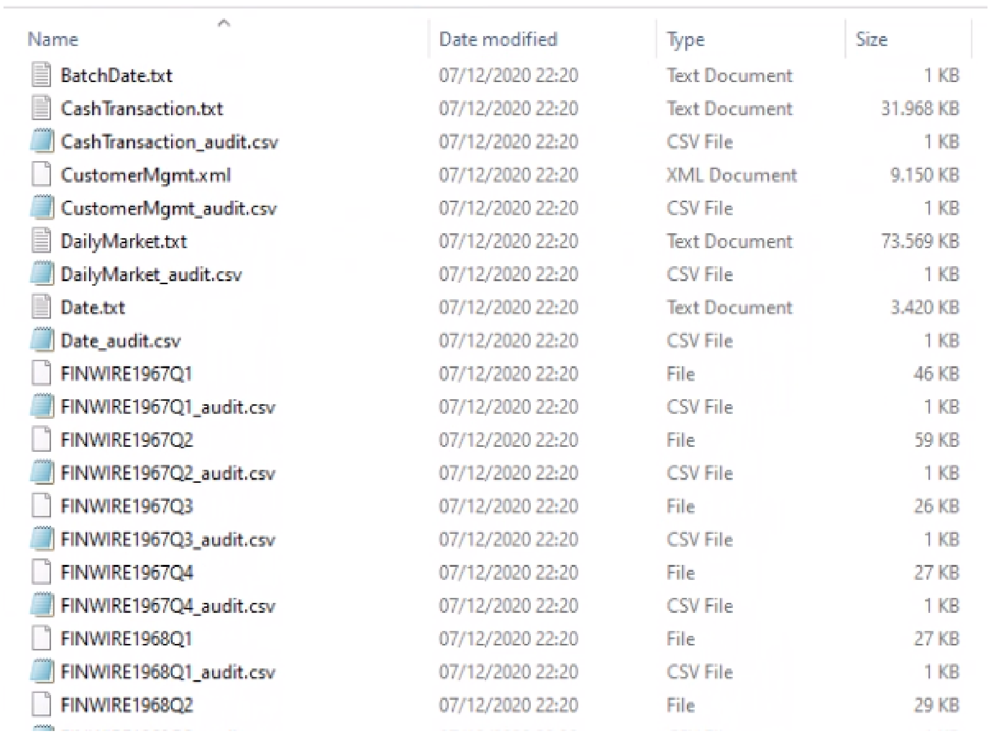
Name	Date modified	Type	Size
Batch1	20/12/2020 16:43	File folder	
Batch2	07/12/2020 22:20	File folder	
Batch3	07/12/2020 22:20	File folder	
Batch1_audit.csv	07/12/2020 22:20	CSV File	1 KB
Batch2_audit.csv	07/12/2020 22:20	CSV File	1 KB
Batch3_audit.csv	07/12/2020 22:20	CSV File	1 KB
digen_report.txt	07/12/2020 22:20	Text Document	1 KB
Generator_audit.csv	07/12/2020 22:20	CSV File	4 KB

Figure 3: Output directory generated

[3]https://docs.microsoft.com/en-us/sql/integration-services/install-windows/install-integration-services?view=sql-server-ver15

[4]https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15

As you can see, there is three Batchs generated. The first one *Batch1* contains all the data we need for the Historical load (the first part of the bencharck). On the following capture, you can have an idea of some of the files present in the **Batch1** folder.



Name	Date modified	Type	Size
BatchDate.txt	07/12/2020 22:20	Text Document	1 KB
CashTransaction.txt	07/12/2020 22:20	Text Document	31.968 KB
CashTransaction_audit.csv	07/12/2020 22:20	CSV File	1 KB
CustomerMgmt.xml	07/12/2020 22:20	XML Document	9.150 KB
CustomerMgmt_audit.csv	07/12/2020 22:20	CSV File	1 KB
DailyMarket.txt	07/12/2020 22:20	Text Document	73.569 KB
DailyMarket_audit.csv	07/12/2020 22:20	CSV File	1 KB
Date.txt	07/12/2020 22:20	Text Document	3.420 KB
Date_audit.csv	07/12/2020 22:20	CSV File	1 KB
FINWIRE1967Q1	07/12/2020 22:20	File	46 KB
FINWIRE1967Q1_audit.csv	07/12/2020 22:20	CSV File	1 KB
FINWIRE1967Q2	07/12/2020 22:20	File	59 KB
FINWIRE1967Q2_audit.csv	07/12/2020 22:20	CSV File	1 KB
FINWIRE1967Q3	07/12/2020 22:20	File	26 KB
FINWIRE1967Q3_audit.csv	07/12/2020 22:20	CSV File	1 KB
FINWIRE1967Q4	07/12/2020 22:20	File	27 KB
FINWIRE1967Q4_audit.csv	07/12/2020 22:20	CSV File	1 KB
FINWIRE1968Q1	07/12/2020 22:20	File	27 KB
FINWIRE1968Q1_audit.csv	07/12/2020 22:20	CSV File	1 KB
FINWIRE1968Q2	07/12/2020 22:20	File	29 KB

Figure 4: Batch1

The next step is to create a destination database. For the TPC-DS benchmark, there were a sql script in the zip folder that allows us to generate. Since here we didn't have this kind of script, we had to find one by ourself. We just had to open this script *createtable.sql* with Sql Server Management Studio, to create the destination database.

Now that we have everything we need, we wants to create a new project on Visual Studio. We choose the template **Integration Services project**.

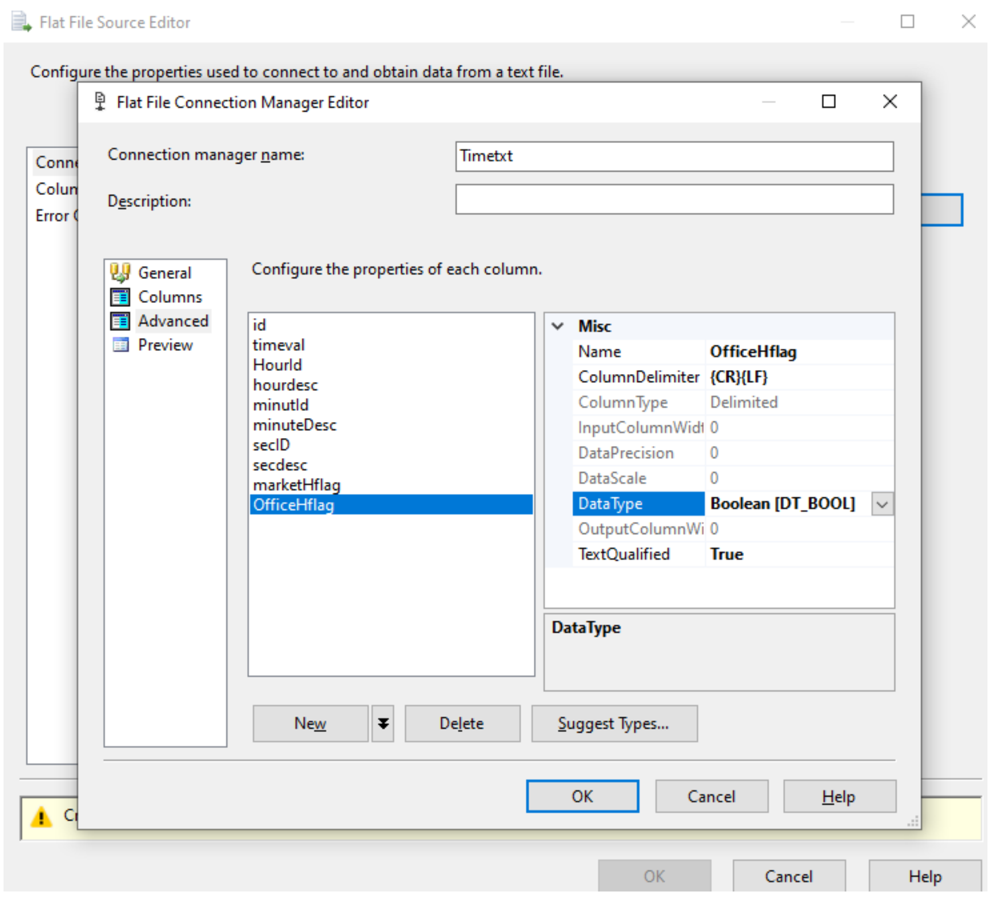
We can finally start the benchmarking !

4.3 Historical Load

With the Batch1 folder generated, we can make the historical load, which is one kind of TPC-DI benchmarking. Here are the following step to realize this benchmark :

STEP 1 : ETL Process for DimTime.
For this ETL process, we have to use the Flat File source and select the file *Time.txt*. Then, we need to uncheck the column *name* in the first data row. The reason is that we don't have this parameter in our source file. We can make this configuration in the Flat File Connection

Manager Editor by clicking on Advanced tab. This will allows you to configure the properties of each columns (*see picture below*), here we need to configure the columns Name -> Datatype. We can also change the delimiter if it is needed.



Configure the properties used to connect to and obtain data from a text file.

Connection manager name: Timetxt

Description:

Configure the properties of each column.

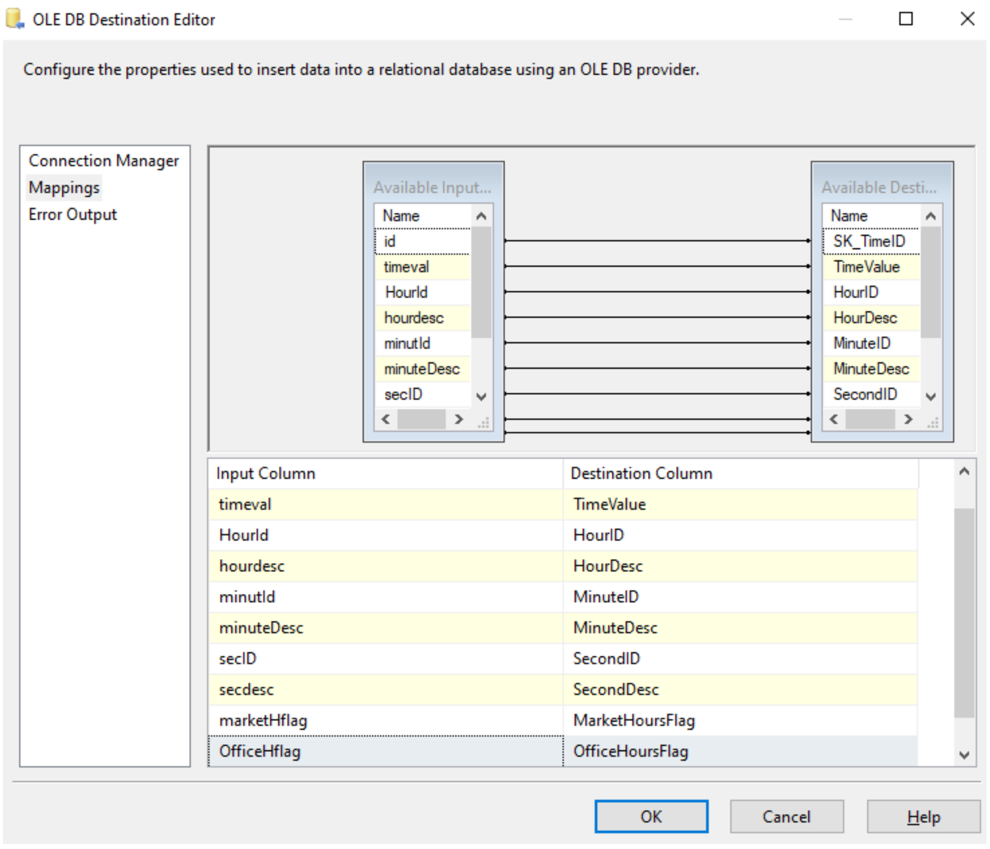
id
timeval
HourId
hourdesc
minutId
minuteDesc
secID
secdesc
marketHflag
OfficeHflag

Misc	
Name	OfficeHflag
ColumnDelimiter	{CR}{LF}
ColumnType	Delimited
InputColumnWidt	0
DataPrecision	0
DataScale	0
DataType	Boolean [DT_BOOL]
OutputColumnWi	0
TextQualified	True

DataType

Figure 5: Configuration of column

After that, we wants to link the source file with OLE DB destination and to do the mapping. It can be done via the OLE DB Destination Editor, on the tabs Connection Manager and Mapping as you can see on the figure 6.



Configure the properties used to insert data into a relational database using an OLE DB provider.

Input Column	Destination Column
timeval	TimeValue
HourId	HourID
hourdesc	HourDesc
minutId	MinuteID
minuteDesc	MinuteDesc
secID	SecondID
secdesc	SecondDesc
marketHflag	MarketHoursFlag
OfficeHflag	OfficeHoursFlag

Figure 6: Mapping

STEP 2 :
To continue this benchmarking, we need to repete the **STEP 1**, for all those tables : DimDate, StatusType, TaxRate, Industry and TradeType. Once all the configurations done, we are getting the results we can see on the image below. We are able to see the connections between all the text files and the corresponding databases with the number of rows.

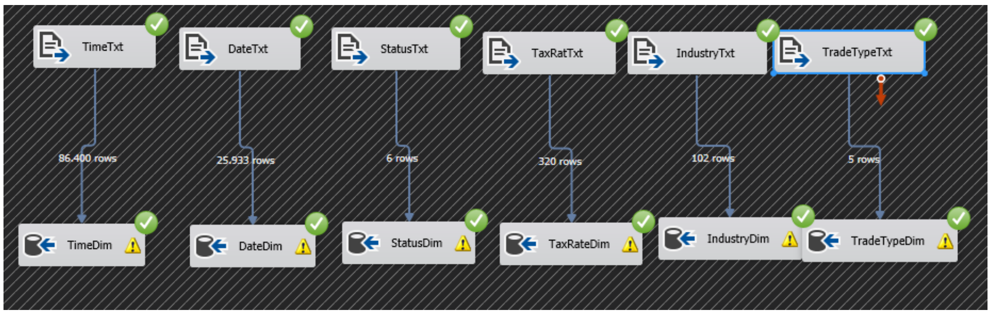


Figure 7: Results

STEP 3 : Load the rest of the tables

DimCompany, DimFinancial and DimSecurity

For those 3 table, the data is based on the FINWIRE files. To load all those files, we have to select new **Foreach loop Container** and specify the folder *Batch1* with files name **FINWIRE*Q***. Each line on FINWIRE files contains one of the Rectype: "Fin","REC", "CMP". The split of data will be based on the Rectype.

DimCompany is obtained from FINWIRE files with the selection RecType="CMP". The Status by matching Status with *ST_ID* from the *StatusType.txt* file and Industry by matching *IndustryID* with *IN_ID* from the *Industry.txt* file.

DimSecurity is obtained from FINWIRE files with the selection RecType="SEC". The Status also by matching Status with *ST_ID* from the *StatusType.txt* file and *SK_CompanyID* is obtained from the **DimCompany** table.

DimBroker HR.csv contains the data for **DimBroker** table. First, we want to ignore if the attribute *EmployeeJobCode* is not equal to 314. Then, we want to add the columns *batchID, isCurrent, endDate* and make a merge join with the column *minDate.* We will also use a lookup on the natural key. The reason is that we want to find when the natural key is already present or not to separate the data. Finally, we will union all data and link it to the DimBroker in the OLE DB destination. You can find the description of this steps on the following schema :

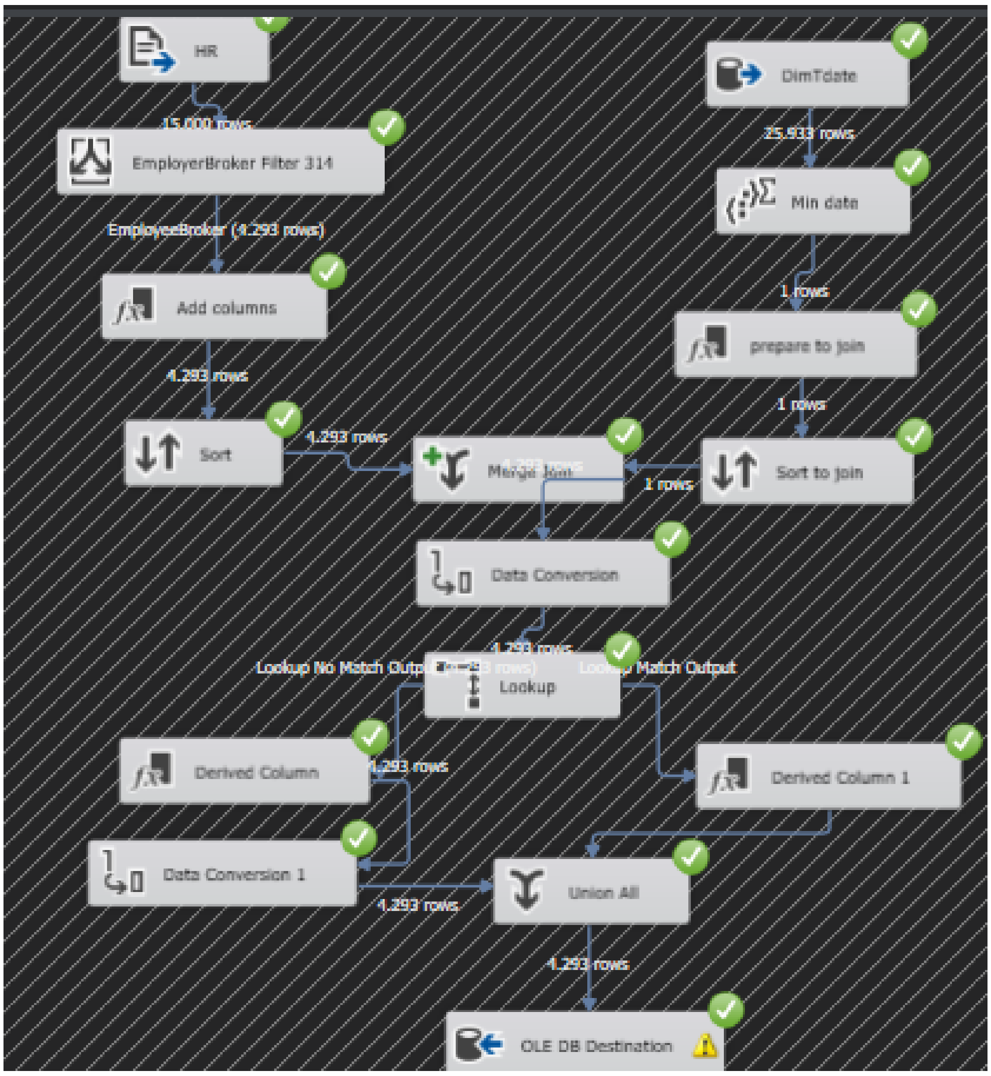


Figure 8: Results

DimCustomer, DimAcount Data for those 2 tables obtained from the data file *Customer-Mgmt.xml* ,The structure of this XML document is described by an XML Schema *Customer-Mgmt.xsd* given by TPC-DI documentation.The implement where where there is a new Account (Customer).

FactWatches We used *WatchHistory.txt* file for *FactWatches* and Surrogate keys from Dim-Customer, DimSecurity and DimDate.We implement the part where *W_ACTION = "ACTV"*.

Prospect Prospect table data is obtained from the Prospect and *DimTime*.IsCustomer is set to True or False depending on whether the customer record matches a current customer record in *DimCustomer*.

5 Results

Using the ETL process, we start with multi-types files (txt,xml,csv) and we got a Data Warehouse that combines the data of all those files. Using TPC-DI Benchmark was a good thing because it contains a huge data, different type of files and a complex transform of data.

6 Conclusion

This project was really interesting, it help us understand how a benchmark works. It is also a good way to put in practice what we have learnt during theoretical lessons. We found this benchmark very stimulating because it is important when we have to deal with a huge amount of data to have a relevant system that has the ability of treating them. In web development, it can happen very often that we need to merge data from a lot of different sources. It is rare to find a database already complete a full for the project we have to do. That is one of the reason data integration is something very important in computer engineering. A good unified data model always required and with a lot of data, the efficiency is the key.