SparkSQL TPC-DI Benchmarking

Data Warehouses (INFO-H419)

Cools Arnaud (499956)
Dubois Alexandre (501050)
Rocca Valerio (589084)
Salazar Maria (587449)

December 24, 2023

Contents

1 Introduction

1.1 Introduction to TPC-DI

The Transaction Processing Performance Council (**TPC**) is a non-profit organization founded in 1988 that defines and develops industry-standard benchmarks for evaluating the performance of various database and transaction processing systems. [3]

TPC-DI (TPC-Data Integration) stands as the inaugural industry benchmark dedicated to evaluating data integration processes. The term "data integration" has emerged as a comprehensive replacement for ETL (Extract, Transform, Load), encompassing the extraction and consolidation of data from varied sources, its transformation into a unified model, and subsequent loading into a designated data store. This multifaceted process extends across diverse scenarios such as business intelligence, analytics, data warehousing, application synchronization, data migrations, master data management, and other related domains [2].

1.2 Introduction to Spark SQL

Spark SQL [1] is an Apache Spark module for working with structured data. The main advantages of this technology are:

- a native integration with Spark, which allows to easily combine it with other Spark components such as Spark Streaming and MLib;
- a uniform way to access various data sources, including Hive, Avro, Parquet, and JSON.
- the possibility to exploit parallel computing by leveraging the Spark engine, which allows Spark SQL to handle large-scale data with speed;
- a sophisticated query optimiser, Catalyst, which can optimize SQL queries with different techniques, including predicate pushdown, constant folding, and join reordering.

It is essential to underline that while it allows the integration of SQL in the Spark environment, Spark SQL is not a Database Management System (DBMS).

1.3 Objective and scope

This project aims to implement the TPC-DI benchmark on Spark SQL and analyze the results. The chosen programming language is Python.

Due to time constraints, the team was unable to fully implement the benchmark.

1. The code partially implements the **Automated Audit Phase** and its components (Batch Validation Query, Data Visibility Queries, and insertion of messages in the `DImessages` table); however, it remains mostly commented as it is non-functional. As a consequence, the team resorted to manual **time** measurement using Python's `time` library.
2. Both the Historical Load and Incremental Phases exhibit **time-consuming executions**. This can be attributed to the extensive use of loops and `Pandas` data frames, which prove inefficient. Consequently, the team was not able to time the execution.

Another limitation stems from Spark SQL's design choices. The framework lacks a built-in operation for **updating** table-stored values. This deficiency becomes apparent in the Incremental Updates phase when transforming the `Prospect` table. To address this, the team opted to simulate the update instead of executing it.

1.4 Tools

In Table 1, we summarize the main tools and technologies utilized.

Tool	**Version**	**Description**
TPC-DI tools	3.2.0	.zip file by TPC containing the official data and query generation tools.
Apache Spark	3.5	Unified engine for large-scale data analytics. It contains Spark SQL as a built-in library.
Hadoop	2.8	Collection of open-source software utilities. Necessary for some utilities.
Python	3.11	High-level, general-purpose programming language used to run the scripts.
Java	8	Java was used to generate the data.
Jupyter Notebook	7.0.6	Environment used to produce Python notebooks.
GitHub	-	GitHub Desktop was used to share code files as well as images conveniently with the team members.
Desktop PC	-	Computer used to perform the analysis. It is equipped with: **processor**: Intel(R) Core(TM) i5-2400 GPU 3.10GHz; **RAM**: 8045MB; **operating system**: Debian GNU/Linux 12

Table 1: Tool Versions and Descriptions

2 TPC-DI Benchmark Overview

This section provides a concise and practical examination of the TPC-DI benchmark.

2.1 Data Model

The TPC-DI benchmark's data model portrays a retail brokerage, merging OLTP data with information from supplementary sources to construct the data warehouse. Figure 1 illustrates the conceptual model of TPC-DI:

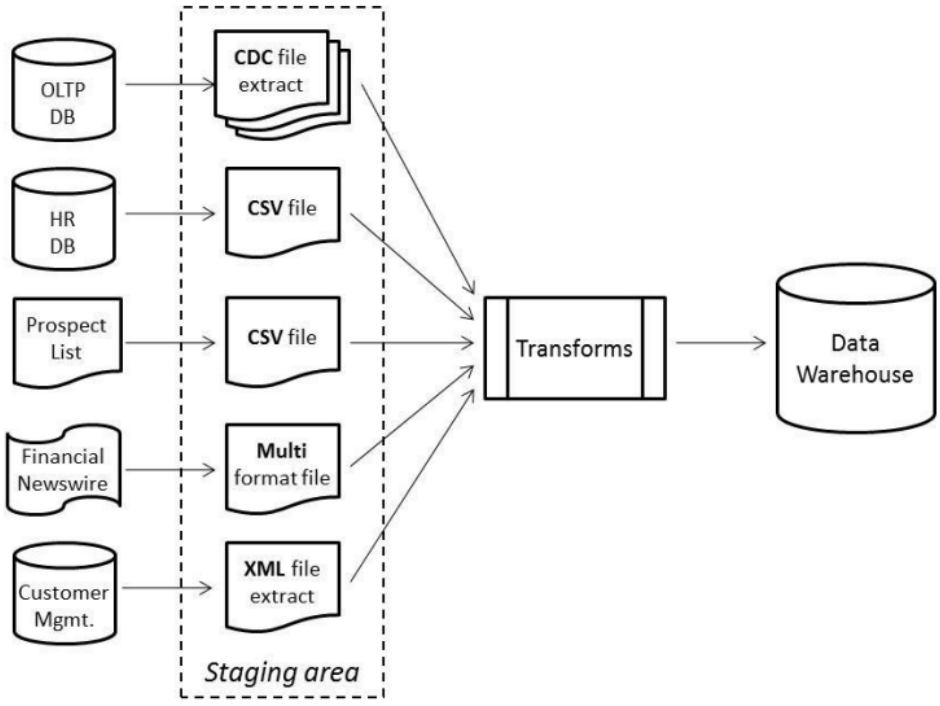

Figure 1: Graphical representation of the TPC-DI conceptual model

The subsequent section offers a brief elucidation of the input data and its content:

- **OLTP Database**: Information on customers, trades, account balances, market data, etc.
- **HR Database**: Table containing employee data.
- **Prospect List**: Demographic data for potential customers.
- **Financial Newswire**: Financial information about companies.
- **Customer Management**: New and updated customer and account information.

While integrating data from diverse source systems is a commonplace requirement in real-world systems, from a benchmark logistics perspective, it poses an intractable problem. Consequently, the mentioned data is purely conceptual, and TPC-DI generates files that can be directly transformed and loadedfor example, creating a single .csv file instead of the HR Database.

2.2 Operations

This section aims to encapsulate the principal TPC-DI operations:

1. **Preparation Phase**. This phase, unmeasured in time, establishes the benchmark performance environment and includes:
   (a) **Data Generation**. File generation and subsequent copying to the Staging Area.
   (b) **Data Warehouse Creation**. Formation of the Data Warehouse and its tables.
   (c) **Data Preparation**. Possible preparation and configuration of data, contingent on the employed DBMS.
2. **Historical Load Phase**. Simulates the initial workload of a Data Warehouse by migrating all relevant existing data in the DBMS to the new system.
3. **Incremental Update Phase**. Mimicks the periodic workload of a Data Warehouse; this phase is repeated twice.
4. **Automated Audit Phase**. At the start of the aforementioned phases, information is collected by executing the so-called **Batch Validation Query**. The data, alongside other stored statistics, is then used in the Automated Audit Phase for validation tests and the creation of a simple report. This phase is not timed.

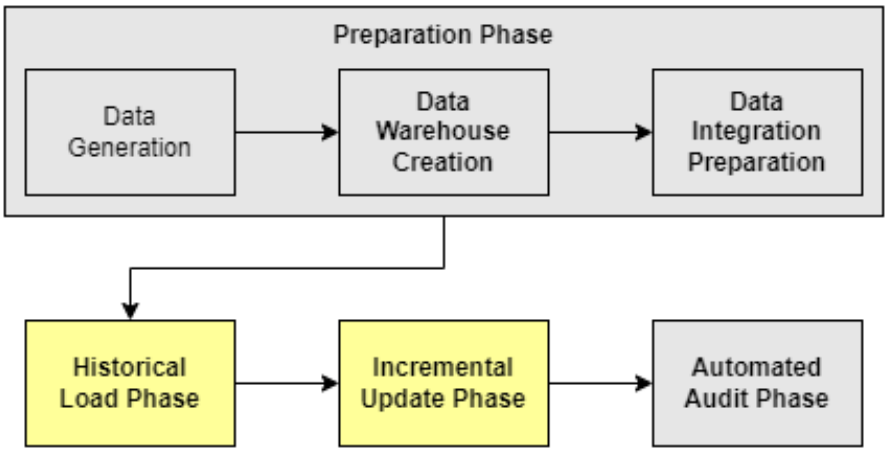

Figure 2: Tactical workflow of TPC-DI's operations (yellow: timed; grey: not timed).

Finally, it is also important to highlight the role of the **Data Visibility Queries**. They are used to verify the visibility of the data written to the Data Warehouse. They also collect data which is stored and examined in the Automated Audit Phase. Those queries have to be executed regularly, from the start of the Incremental Update 1 to the conclusion of the Incremental Update 2.

2.3 Data Warehouse

The culmination of the Data Integration process is a Dimensional Data Warehouse. The term 'Dimensional' pertains to the organization of data into easily comprehensible and analyzable structures, involving facts (measurable metrics) and dimensions (descriptive attributes). In this context, dimension tables represent the business entities of interest, while fact tables record the descriptions of occurrences (e.g., price and volume of a transaction).

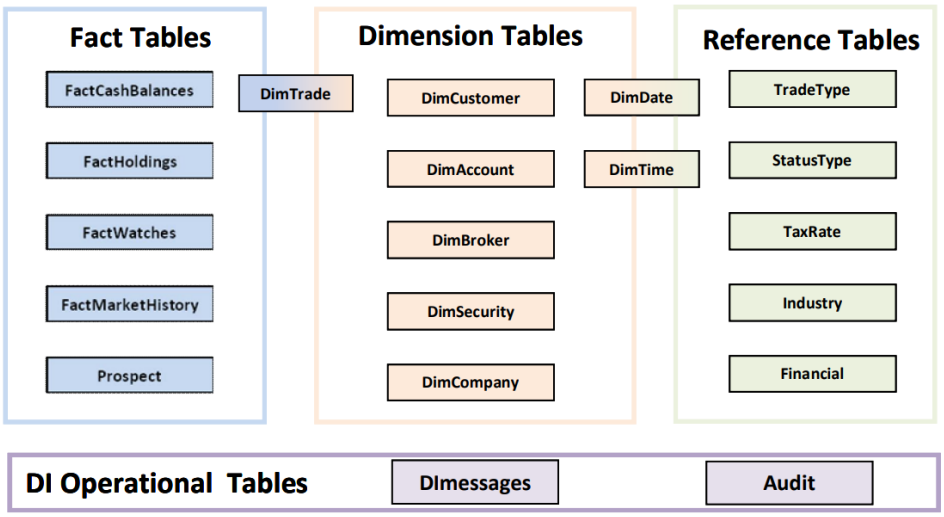

Figure 3: Graphical representation of the Data Warehouse's schema

2.4 Performance measure

Finally, follows the performance measure of the test.

$$TPC\_DI\_RPS = Trunc(GeoMean(T_H, Min(T_{I1}, T_{I2})))$$

- $T_H = R_H/E_H$ is the throughput of the Historical Load, where R_H is the total number of rows of Source Data in the Historical Load data set, while E_H is the elapsed time for the Historical Load, expressed in seconds.
- $T_{I1} = R_{I1}/Max(E_H, 1800)$ and $T_{I2} = R_{I2}/Max(E_H, 1800)$ are the throughput of the two Incremental Updates, where R_{I1} and R_{I2} are the total number of rows of Source Data in the Incremental Update data sets, while E_{I1} and E_{I2} are the elapsed time for the Incremental Updates, expressed in seconds.
- $Trunc$ the whole number portion of the argument.
- $GeoMean$ is the geometric mean of the arguments.

3 Implementation

3.1 Data generation

Data at each scale factor can be generated by the following command in **Bash**, over the toolkit folder:

```
java -jar DIGen.jar sf <SF>
```

where SF was changed for each scale factor and saved in the main folder.

To correctly run the code, generated data has to be stored in the following directory.

```
path-to-working-directory/data/scale-factor
```

3.2 Other files

To reproduce the results, a user needs also other files. All of them are available on the GitHub folder at this link, alongside their description.

References

[1] Apache Spark. *Apache Spark SQL Documentation*. Accessed on November 13, 2023. URL: https://spark.apache.org/sql/.

[2] Caufield B., Jacobsen H., Poess M., Rabl T. *TPC-DI: the first industry benchmark for data integration*. Accessed on November 16, 2023. URL: https://dl.acm.org/doi/10.14778/2733004.2733009.

[3] TPC. *TPC - Transaction Processing Performance Council*. Accessed on November 13, 2023. URL: https://www.tpc.org/.