The case study: a Benchmark for decision support

For the case study of the course we shall a benchmark for decision support available at site www.tpc.org

The TPC Benchmark™H (TPC-H) is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications.

The queries and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation.

This benchmark illustrates decision support systems that:

- Examine large volumes of data;
- Execute queries with a high degree of complexity;
- Give answers to critical business questions.

TPC Benchmark™ H is comprised of a set of business queries designed to exercise system functionalities in a manner representative of complex business analysis applications.

These queries have been given a realistic context, portraying the activity of a wholesale supplier.

TPC-H does not represent the activity of any particular business segment, but rather any industry which must manage sell, or distribute a product worldwide (e.g., car rental, food distribution, parts, suppliers, etc.).

It includes:

- a logical schema;
- a set of queries;
- a scalable set of data.

TPC-H Schema and queries

The following diagram reports the logical schema (SF is Scale Factor to parametrically resize the number of rows).

These selected queries provide answers to the following classes of business analysis:

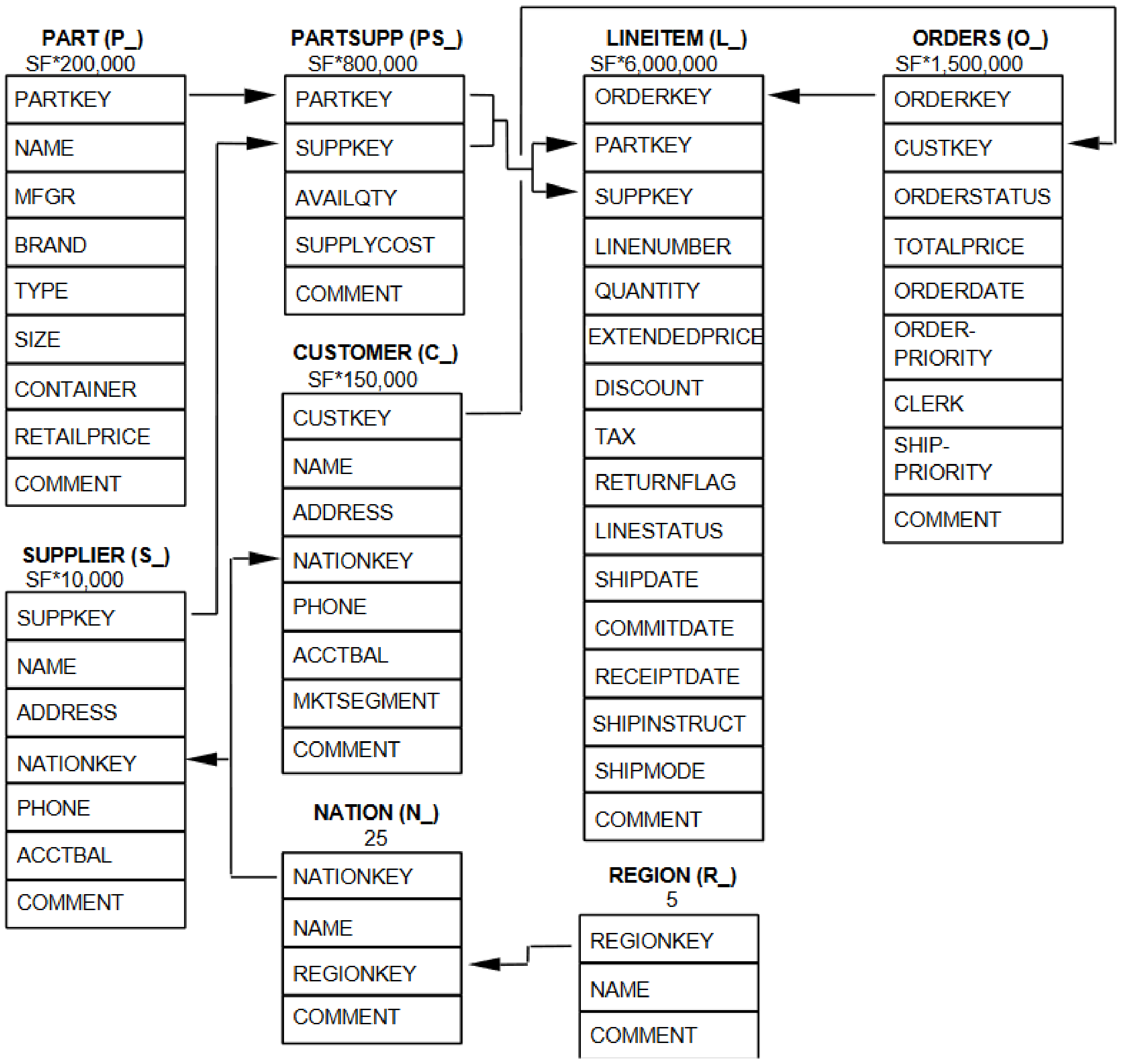
- Pricing and promotions;
- Supply and demand management;
- Profit and revenue management;
- Customer satisfaction study;
- Market share study;
- Shipping management.

The queries that have been selected exhibit the following characteristics:

- They have a high degree of complexity;
- They use a variety of access;
- They are of an ad hoc nature;
- They examine a large percentage of the available data;

- They all differ from each other;
- They contain query parameters that change across query executions.

Even though the available set of queries is not immediately used for the realization of the student project, it can be consulted as a relevant example of the database usage.



Benchmark Download

The benchmark can be downloaded at site www.tpc.org. The zip of the download folder is also available in the folder of the materials for the course in the TEAMS channel.

The complete documentation of the benchmark https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v3.0.1.pdf The pdf of documentation is also available in the folder of the materials for the course in the TEAMS channel.

The documentation contains:

- A complete specification of the logical schema (relational schemata and data types).
- A set of predefined parametric queries.
- The package for creating the set of data can be obtained at site

https://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp

The package allows to generate data parametric with respect to a Scale Factor**. For the course project a SF of 10 is required.**

Metrics for the data set with Scale Factor 1

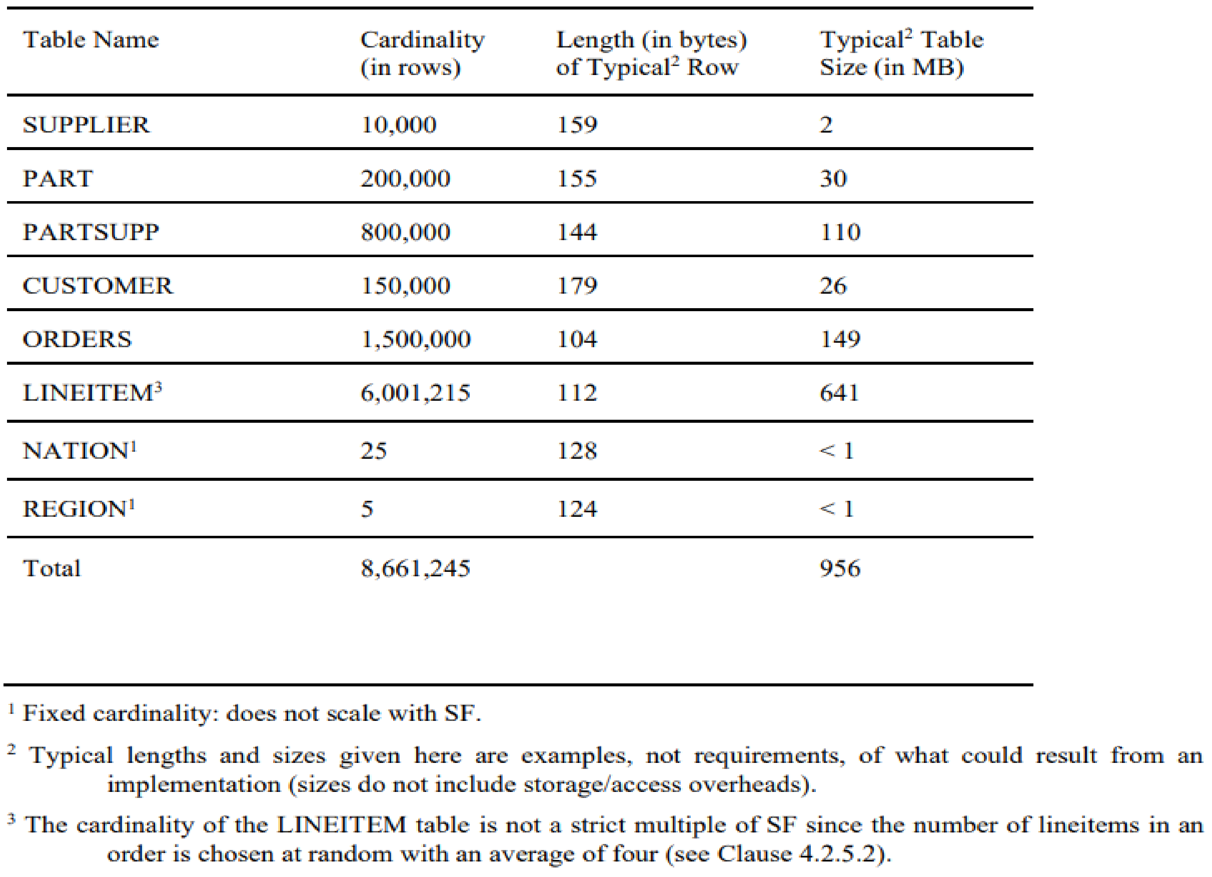


Table Name	Cardinality (in rows)	Length (in bytes) of Typical[2] Row	Typical[2] Table Size (in MB)
SUPPLIER	10,000	159	2
PART	200,000	155	30
PARTSUPP	800,000	144	110
CUSTOMER	150,000	179	26
ORDERS	1,500,000	104	149
LINEITEM[3]	6,001,215	112	641
NATION[1]	25	128	< 1
REGION[1]	5	124	< 1
Total	8,661,245		956

[1] Fixed cardinality: does not scale with SF.

[2] Typical lengths and sizes given here are examples, not requirements, of what could result from an implementation (sizes do not include storage/access overheads).

[3] The cardinality of the LINEITEM table is not a strict multiple of SF since the number of lineitems in an order is chosen at random with an average of four (see Clause 4.2.5.2).

Metrics of supported scale factors (SF 10 should be used for the project)

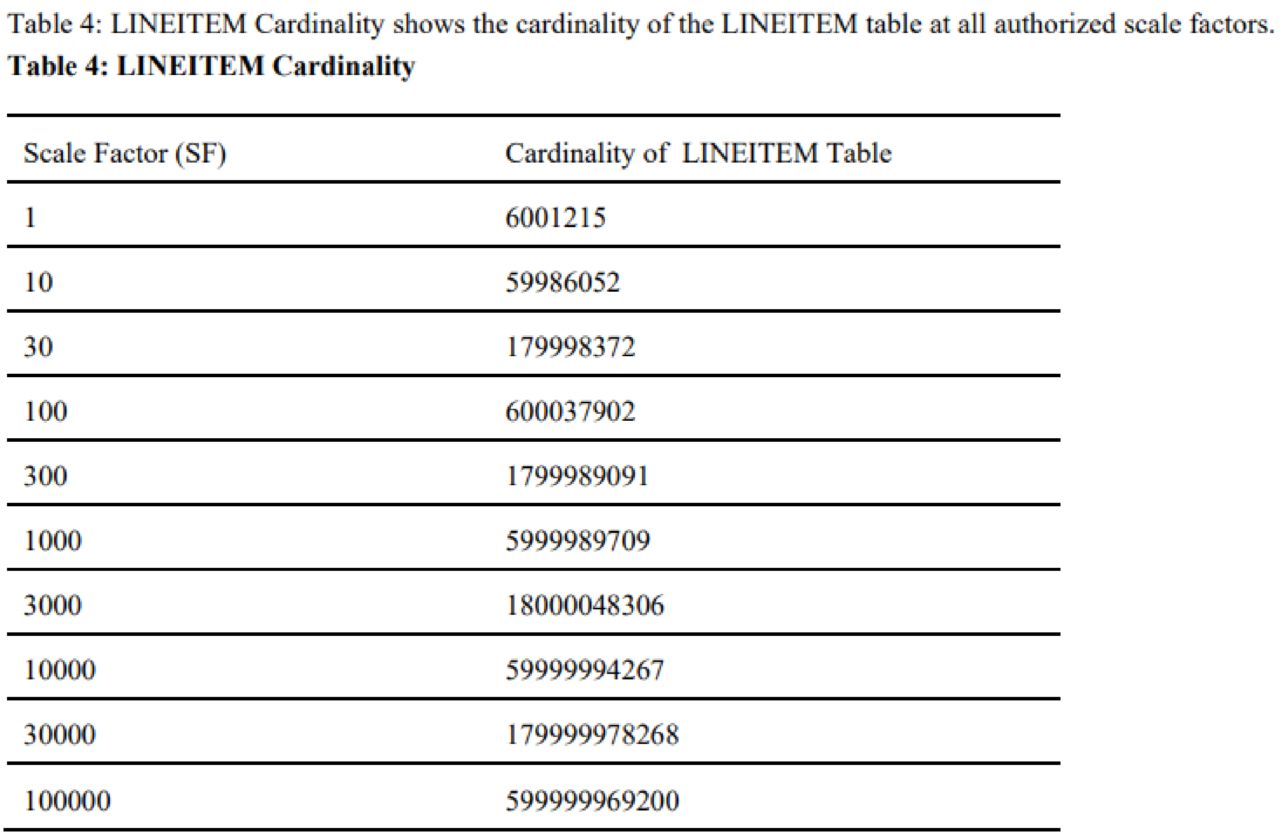


Table 4: LINEITEM Cardinality shows the cardinality of the LINEITEM table at all authorized scale factors.

Table 4: LINEITEM Cardinality

Scale Factor (SF)	Cardinality of LINEITEM Table
1	6001215
10	59986052
30	179998372
100	600037902
300	1799989091
1000	5999989709
3000	18000048306
10000	59999994267
30000	179999978268
100000	599999969200

Project requirements

The project is assigned to groups. A group can consist of up to three students. Groups consisting of only one person are discouraged but admitted.

The DBMS chosen for the implementation is PostgreSQL (no different DBMS is admitted).

The group follows these preliminary steps:

1. Download the dbgen package.
2. Implement the DB in the DBMS PostgreSQL following the specification provided by the documentation (structure and datatypes fixed by the logical schema)
3. Populate the DB using data generated by dbgen (csv files)
4. Choose a set of query schemata for the optimization exercise (the query schemata can be found in the following section)
5. Write the SQL code the query schemata (before optimization)
6. Collect the execution t for time (before optimization)
7. Points 5-6 must be reported in the documentation.

After populating the DB the groups collect some statistics which will be reported in the project documentation. For each table:

- Number of rows
- Table size
- Number of distinct values for each attribute which will be used for querying
- MinValue and MaxValue for each attribute which will be used for querying.

Query schemata

In the following are reported three query schemata. Each group freely chooses the schemata. A group of only one person can choose only the first schema; the groups of two persons must choose the first schema and between the second and the third; the groups of three persons must consider all the three schemata

Each schema requires SQL queries for roll-up and slice operations. As for roll-up, the queries should support all the roll-up combinatorics suggested by the schema. As for slicing, the optimization activity consider slicing by a parametric value (the value cannot be considered as fixed for the sake of optimization).

Query Schema 1 Export/import revenue value.

Aggregation of the export/import of revenue of lineitems between two different nations (E,I) where E is the nation of the lineitem supplier and I the nations of the lineitem customer (export means that the supplier is in the nation E and import means is in the nation I).

The revenue is obtained by l_extendedprice * (1 - l_discount) of the considered lineitems

The aggregations should be performed with the following roll-up

Month → Quarter → Year

Type

Nation → Region

The slicing is over Type and Exporting nation.

Query Schema 2 Late delivery.

Number of orders where at least one lineitem has been received later than the committed date.

The aggregations should be performed with the following roll-up

Month → Year

Nation → Region (Customer)

The same query can be issued with the following slicing

A specific Month

A specific Type of lineitem

Query Schema 3 Returned item loss

Returned item loss.

The query gives the revenue loss for customers who might be having problems with the parts that are shipped to them. Revenue lost is defined as sum(l_extendedprice*(1-l_discount)) for all qualifying lineitems.

The aggregations should be performed with the following roll-up

Month → Quarter → Year

Customer

The query can be issued with the following slicing (combined)

Name of a customer

A specific quarter

Optimization

The optimization should consider the following strategies:

1. Indexing on tables (mandatory)
2. Materialization of views (mandatory)
3. Indexing on materialized views (optional)
4. Horizontal partitioning (optional).

The final optimization must fulfil the space constraint that the amount of space for indexes and materialized views must be less than 1,5 times the size of the database.

Documentation

- **The documentation of the project includes:**
- **SQL definition of the tables**
- **Statistics of the data**
- **Definition of the set of queries**
- **Query cost before optimization**
- **Design of indexes**
- **Space cost of indexing**
- **Query cost of queries with indexes**
- **Design of materialization**
- **Query cost of queries with materialized views (without indexing)**
- **Query cost of queries with materialized views and indexing**
- **Fragmentation (optional)**
- **Recap of the final optimization strategy compared with the initial query cost.**