University of Trieste

Data Management for Big Data Course

Academic Year 2022–2023

Data Warehouse case study

Davide Capone[*] Sandro Junior Della Rovere[†] Enrico Stefanel[‡]

Last update on 2023–06–21T19:07:34Z.

1 Introduction

The aim of this project is to study an efficient implementation of a suite of business oriented ad-hoc queries over the public TPC-H benchmark, which can be considered as a Big Data database, that has been implemented in Postgres.

1.1 TPC-H benchmark database

The TPC-H benchmark is a decision support benchmark that can be downloaded from the TPC official website. The data generator lets the user specify a *scale factor* in order to control the size of the resulted database. Our choices was to use a *scale factor* of 10, meaning that the overall database size is approximately 13 GB.

1.1.1 Database statistics

The benchmark is composed by eight tables:

- `CUSTOMER`, with 1 500 000 tuples (312 MB);
- `LINEITEM`, with 59 986 052 tuples (11 GB); the main attributes that are going to be used are:

[*]`davide.capone@studenti.units.it`
[†]`sandrojunior.dellarovere@studenti.units.it`
[‡]`enrico.stefanel@studenti.units.it`

    - l_extendedprice (1 351 462 distinct values, i.e. there is an average of 44 tuples with the same value, that range from 900.91 to 104 949.50),
    - l_discount (11 distinct values, i.e. there is an average of 5 453 277 tuples with the same value, that range from 0.00 to 0.10),
    - l_returnflag (which can assume values A→accepted, R→returned, N→not yet delivered; the percentage of tuples for each value is roughly the same),
    - l_commitdate,
    - l_receiptdate;

- NATION, with 25 tuples (24 kB);
- ORDERS, with 1 500 000 tuples (2481 kB); the main attributes that are going to be used are:
    - o_orderdate,
- PART, with 2 000 000 tuples (363 MB); the main attributes that are going to be used are:
    - p_type;
- PARTSUPP, with 8 000 000 tuples (1535 MB);
- REGION, with 5 tuples (24 kB);
- SUPPLIER, with 100 000 tuples (20 MB).

Other attributes have been used, but statistics about them have been omitted for lack of usefulness (e.g., keys of the tables, for which the cardinality is exactly the cardinality of the corresponding table).

2 Export/import revenue value

2.1 Naïve Implementation

```sql
WITH query1 AS (
SELECT
    EXTRACT (YEAR FROM o_orderdate) AS _year,
    EXTRACT (QUARTER FROM o_orderdate) AS _quarter,
    EXTRACT (MONTH FROM o_orderdate) AS _month,
    c_regionname,
    c_nationname,
    c_name,
    s_regionname,
    s_nationname,
    s_name,
    p_type,
    SUM(l_extendedprice * (1 - l_discount)) AS revenue
FROM lineitem_orders
    JOIN part ON l_partkey = p_partkey
    JOIN supplier_location ON (s_suppkey = l_suppkey)
    JOIN customer_location ON (c_custkey = o_custkey)
WHERE s_nationkey <> c_nationkey
GROUP BY
    _year,
    _quarter,
    _month,
    c_regionkey,
    c_regionname,
    c_nationkey,
    c_nationname,
    c_custkey,
    c_name,
    s_regionkey,
    s_regionname,
    s_nationkey,
    s_nationname,
    s_suppkey,
    s_name,
    p_type
)
SELECT * FROM query1;
```

3 Late delivery

It is asked to retrieve the number of orders where at least one “lineitem” has been received later than the committed date. The aggregation should be performed with the Month → Year roll-up, and the (Customer’s) Nation → Region roll-up.

Lorem ipsum. . .

4 Returned item loss

It is asked to retrieve the *revenue loss* for customers who might be having problems with the parts that are shipped to them, where a *revenue loss* is defined as `SUM(l_extendedprice*(1-l_disc` for all qualifying *lineitems*.

4.1 Naïve implementation

```sql
WITH lineitem_orders AS (
    SELECT
        o_orderkey,
        l_partkey,
        l_suppkey,
        o_orderdate,
        o_custkey,
        l_extendedprice,
        l_discount,
        l_returnflag,
        l_commitdate,
        l_receiptdate
    FROM lineitem JOIN orders ON (l_orderkey = o_orderkey)
),
query3 AS (
SELECT
    EXTRACT (YEAR FROM o_orderdate) AS _year,
    EXTRACT (QUARTER FROM o_orderdate) AS _quarter,
    EXTRACT (MONTH FROM o_orderdate) AS _month,
    c_name,
    SUM(l_extendedprice * (1 - l_discount)) AS returnloss
FROM
    lineitem_orders
    JOIN customer ON o_custkey = c_custkey
WHERE
    l_returnflag = 'R'
    -- AND c_name = 'Customer#000129976'
    -- AND EXTRACT (QUARTER FROM o_orderdate) = 1
GROUP BY
    _year,
    _quarter,
    _month,
    c_custkey,
    c_name
)
SELECT * FROM query3;
```