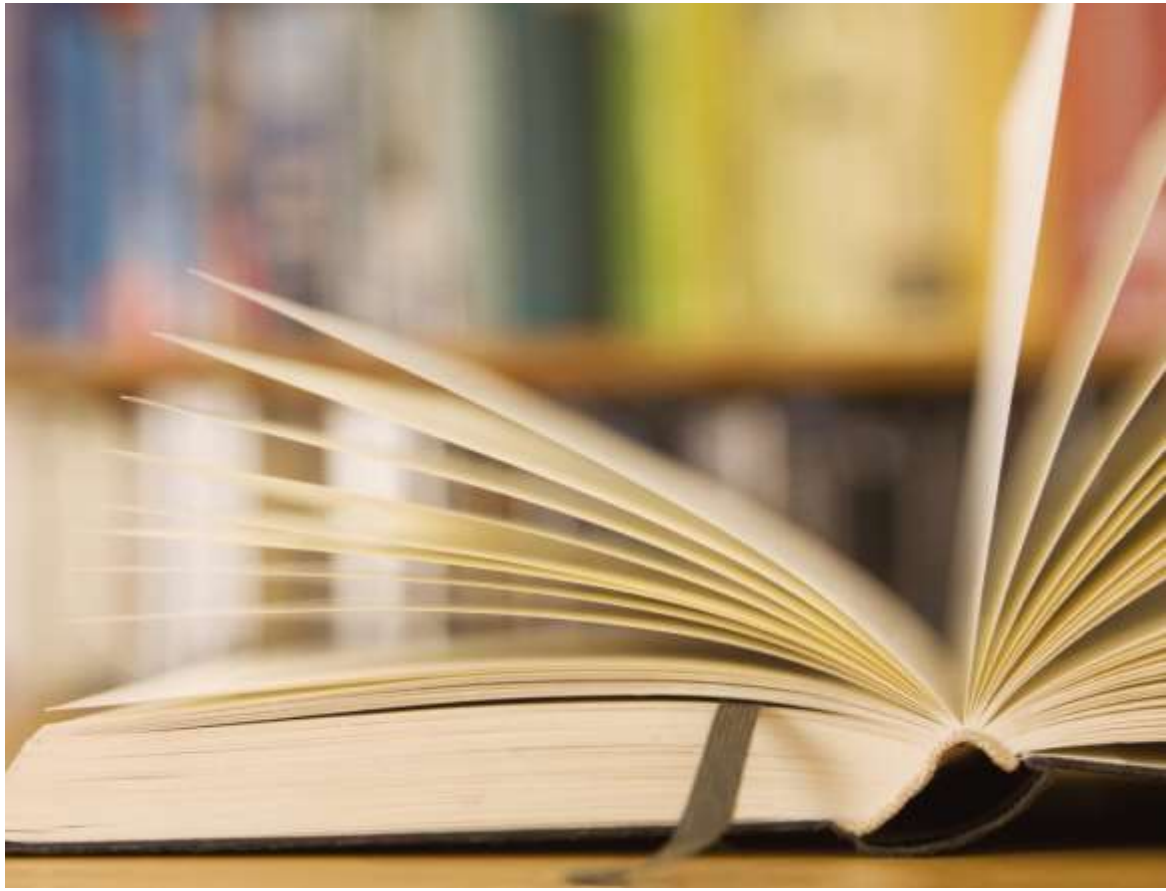
CSCI317 – Database Performance Tuning

Materialized View

17 August 2022



Problem statements …1/2

Considering the following queries over TPCHR benchmark database.

1. Find the names of parts and the names of supplier who supply parts. List the names of parts (P_NAME) and the names of suppliers (S_NAME).
2. Find the total number of parts supplied by each supplier. Consider only supplier who supply at least one part. List the names of suppliers (S_NAME) and the total number of parts by each supplier.
3. Find the names of parts, the names of suppliers who supply the parts, and available quantities of supplied parts. List the names of parts (P_NAME), the names of suppliers (S_NAME), and available quantities (PS_AVAILQTY).

Problem statements …2/2

4. Find the average supply costs (PS_SUPPLYCOST) of each part. List the average supply costs (PS_SUPPLYCOST) and part name (P_NAME).
5. For each part find a name of brand it belongs to. List a part name (P_NAME) and brand name (P_BRAND).

An objective of this task is to create the smallest number of materialized views that can be automatically used to speed up the processing of the queries specified above.

Initialization

Step 0: Initialize the environment

```
CONNECT AS SYSTEM
ALTER SYSTEM SET OPTIMIZER_MODE=ALL_ROWS;
ALTER SYSTEM SET QUERY_REWRITE_ENABLED=TRUE;
ALTER SYSTEM SET QUERY_REWRITE_INTEGRITY=ENFORCED;
GRANT CREATE MATERIALIZED VIEW TO TPCHR;
GRANT QUERY REWRITE TO TPCHR;
GRANT CREATE ANY TABLE TO TPCHR;
```

Realizing the queries …1/5

Step 1: Realizing the queries as SELECT statements.

1. Find the names of parts and the names of supplier who supply parts. List the names of parts (P_NAME) and the names of suppliers (S_NAME).

```
SELECT P_NAME, S_NAME
FROM PART JOIN PARTSUPP
    ON P_PARTKEY = PS_PARTKEY
    JOIN SUPPLIER
    ON S_SUPPKEY = PS_SUPPKEY;
```

Realizing the queries …2/5

2. Find the total number of parts supplied by each supplier. Consider only supplier who supply at least one part. List the names of suppliers (S_NAME) and the total number of parts by each supplier.

```
SELECT S_NAME, COUNT(*)
FROM SUPPLIER JOIN PARTSUPP
        ON S_SUPPKEY = PS_SUPPKEY
        GROUP BY S_NAME;
```

Realizing the queries …3/5

3. Find the names of parts, the names of suppliers who supply the parts, and available quantities of supplied parts. List the names of parts (P_NAME), the names of suppliers (S_NAME), and available quantities (PS_AVAILQTY).

```
SELECT P_NAME, S_NAME, PS_AVAILQTY
FROM PART JOIN PARTSUPP
        ON P_PARTKEY = PS_PARTKEY
        JOIN SUPPLIER
        ON S_SUPPKEY = PS_SUPPKEY;
```

Realizing the queries …4/5

4. Find the average supply costs (PS_SUPPLYCOST) of each part. List the average supply costs (PS_SUPPLYCOST) and part name (P_NAME).

```
SELECT P_NAME, AVG(PS_SUPPLYCOST)
FROM PART JOIN PARTSUPP
        ON P_PARTKEY = PS_PARTKEY
GROUP BY P_NAME;
```

Realizing the queries …5/5

5. For each part find a name of brand it belongs to. List a part name (P_NAME) and brand name (P_BRAND).

```
SELECT P_NAME, P_BRAND
FROM PART;
```

Find query processing plans …1/5

Step 2: Find the query processing plans for the SELECT statements

```
EXPLAIN PLAN FOR
SELECT P_NAME, S_NAME
FROM PART JOIN PARTSUPP
     ON P_PARTKEY = PS_PARTKEY
     JOIN SUPPLIER
     ON S_SUPPKEY = PS_SUPPKEY;
```

Find query processing plans …2/5

```
EXPLAIN PLAN FOR
SELECT S_NAME, COUNT(*)
FROM SUPPLIER JOIN PARTSUPP
        ON S_SUPPKEY = PS_SUPPKEY
GROUP BY S_NAME;
```

Find query processing plans …3/5

```
EXPLAIN PLAN FOR
SELECT P_NAME, S_NAME, PS_AVAILQTY
FROM PART JOIN PARTSUPP
        ON P_PARTKEY = PS_PARTKEY
        JOIN SUPPLIER
        ON S_SUPPKEY = PS_SUPPKEY;
```

Find query processing plans …4/5

```
EXPLAIN PLAN FOR
SELECT P_NAME, AVG(PS_SUPPLYCOST)
FROM PART JOIN PARTSUPP
        ON P_PARTKEY = PS_PARTKEY
GROUP BY P_NAME;
```

Find query processing plans …5/5

```
EXPLAIN PLAN FOR
SELECT P_NAME, P_BRAND
FROM PART;
```

Create a materialized view …1/

Step 3: Create and load data into the smallest number of materialised views that can be used to speed up the retrieval of the same information as SELECT statements implemented in Step 1.

```
CREATE MATERIALIZED VIEW PARTSUPP1
        ENABLE QUERY REWRITE AS (
SELECT PS_SUPPKEY,SUM(PS_AVAILQTY),COUNT(*)
FROM PARTSUPP
GROUP BY PS_SUPPKEY);
```

Create a materialized view …2/

```
CREATE MATERIALIZED VIEW PARTSUPP2
        ENABLE QUERY REWRITE AS (
SELECT PS_SUPPKEY,SUM(PS_AVAILQTY),COUNT(*)
FROM PARTSUPP
GROUP BY PS_SUPPKEY);
```

Create a materialized view …3/

```
CREATE MATERIALIZED VIEW PARTSUPP3
        ENABLE QUERY REWRITE AS (
SELECT PS_COMMENT,SUM(PS_AVAILQTY),COUNT(*)
FROM PARTSUPP
GROUP BY PS_COMMENT);
```

Create a materialized view …4/

```
CREATE MATERIALIZED VIEW V4_PARTSUPP
        ENABLE QUERY REWRITE AS (
SELECT PS_PARTKEY,PS_COMMENT,SUM(PS_AVAILQTY),COUNT(*)
FROM PARTSUPP
GROUP BY PS_PARTKEY,PS_COMMENT);
```

Find query processing plans …1/5

Step 4: Find the query processing plans for the SELECT statements. This time, it is expected that the query processing plans use the materialized view created in step 3.

```
EXPLAIN PLAN FOR
SELECT P_NAME, S_NAME
FROM PART JOIN PARTSUPP
     ON P_PARTKEY = PS_PARTKEY
     JOIN SUPPLIER
     ON S_SUPPKEY = PS_SUPPKEY;
```

Find query processing plans …2/5

```
EXPLAIN PLAN FOR
SELECT S_NAME, COUNT(*)
FROM SUPPLIER JOIN PARTSUPP
        ON S_SUPPKEY = PS_SUPPKEY
GROUP BY S_NAME;
```

Find query processing plans …3/5

```
EXPLAIN PLAN FOR
SELECT P_NAME, S_NAME, PS_AVAILQTY
FROM PART JOIN PARTSUPP
        ON P_PARTKEY = PS_PARTKEY
        JOIN SUPPLIER
        ON S_SUPPKEY = PS_SUPPKEY;
```

Find query processing plans …4/5

```
EXPLAIN PLAN FOR
SELECT P_NAME, AVG(PS_SUPPLYCOST)
FROM PART JOIN PARTSUPP
        ON P_PARTKEY = PS_PARTKEY
GROUP BY P_NAME;
```

Find query processing plans …5/5

```
EXPLAIN PLAN FOR
SELECT P_NAME, P_BRAND
FROM PART;
```