FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

# QUERY OPTIMIZATION
# CULTURAL FACILITIES

Database Technologies 2020/2021


**Group G:**

André Mori - up201700493@edu.fe.up.pt
Diogo Silva - up201405742@edu.fe.up.pt
Jiahao Zhang - up202010263@edu.fe.up.pt
Pasit Khantigul - up202010272@edu.fe.up.pt

# Index

# Introduction

This project was developed in the scope of the Database Technologies course from the Master in Informatics and Computing Engineering at FEUP. The goal of the project was to develop different SQL execution plans in a database and analyse the impact of using indexes and different approaches.
To achieve the goal, three different table environments were created: X (no indexes or constraints); Y (standard integrity constraints); Z (standard integrity constraints and indexes).
**Note:** On the ***sql/*** folder delivered, there are three different files: *create.sql* to setup the environment, *indexes.sql* to create the indexes and *queries.sql* with the queries for each question.

# Constraints

The constraints added to the Y and Z tables are primary and foreign keys. The code below represents the script which creates these constraints. Primary keys follow a name convention of *<TABLE_NAME>_<COLUMN>_PK*, while foreign keys follow *<TABLE_NAME>_<FOREIGN_TABLE_NAME>_FK*.

```
-- Primary Keys
ALTER TABLE "YREGIONS" ADD CONSTRAINT YREGIONS_COD_PK PRIMARY KEY("COD");
ALTER TABLE "YDISTRICTS" ADD CONSTRAINT YDISTRICTS_COD_PK PRIMARY KEY("COD");
ALTER TABLE "YMUNICIPALITIES" ADD CONSTRAINT YMUNICIPALITIES_COD_PK PRIMARY KEY("COD");
ALTER TABLE "YFACILITIES" ADD CONSTRAINT YFACILITIES_ID_PK PRIMARY KEY("ID");
ALTER TABLE "YROOMTYPES" ADD CONSTRAINT YROOMTYPES_ROOMTYPES_PK PRIMARY KEY("ROOMTYPE");
ALTER TABLE "YACTIVITIES" ADD CONSTRAINT YACTIVITIES_REF_PK PRIMARY KEY("REF");
ALTER TABLE "YUSES" ADD CONSTRAINT YUSES_ID_REF_PK PRIMARY KEY("ID","REF");
ALTER TABLE "ZREGIONS" ADD CONSTRAINT ZREGIONS_COD_PK PRIMARY KEY("COD");
ALTER TABLE "ZDISTRICTS" ADD CONSTRAINT ZDISTRICTS_COD_PK PRIMARY KEY("COD");
ALTER TABLE "ZMUNICIPALITIES" ADD CONSTRAINT ZMUNICIPALITIES_COD_PK PRIMARY KEY("COD");
ALTER TABLE "ZFACILITIES" ADD CONSTRAINT ZFACILITIES_ID_PK PRIMARY KEY("ID");
ALTER TABLE "ZROOMTYPES" ADD CONSTRAINT ZROOMTYPES_ROOMTYPES_PK PRIMARY KEY("ROOMTYPE");
ALTER TABLE "ZACTIVITIES" ADD CONSTRAINT ZACTIVITIES_REF_PK PRIMARY KEY("REF");
ALTER TABLE "ZUSES" ADD CONSTRAINT ZUSES_ID_REF_PK PRIMARY KEY("ID","REF");

-- Foreign Keys
ALTER TABLE "YDISTRICTS" ADD CONSTRAINT YDISTRICTS_REGION_FK FOREIGN KEY("REGION")
REFERENCES "YREGIONS"("COD");
ALTER TABLE "YMUNICIPALITIES" ADD CONSTRAINT YMUNICIPALITIES_REGION_FK FOREIGN
KEY("REGION") REFERENCES "YREGIONS"("COD");
ALTER TABLE "YMUNICIPALITIES" ADD CONSTRAINT YMUNICIPALITIES_DISTRICT_FK FOREIGN
KEY("DISTRICT") REFERENCES "YDISTRICTS"("COD");
ALTER TABLE "YFACILITIES" ADD CONSTRAINT YFACILITIES_MUNICIPALITY_FK FOREIGN
KEY("MUNICIPALITY") REFERENCES "YMUNICIPALITIES"("COD");
```

```
ALTER TABLE "YFACILITIES" ADD CONSTRAINT YFACILITIES_ROOMTYPE_FK FOREIGN KEY("ROOMTYPE")
REFERENCES "YROOMTYPES"("ROOMTYPE");
ALTER TABLE "YUSES" ADD CONSTRAINT YUSES_ID_FK FOREIGN KEY("ID") REFERENCES
"YFACILITIES"("ID");
ALTER TABLE "YUSES" ADD CONSTRAINT YUSES_REF_FK FOREIGN KEY("REF") REFERENCES
"YACTIVITIES"("REF");
ALTER TABLE "ZDISTRICTS" ADD CONSTRAINT ZDISTRICTS_REGION_FK FOREIGN KEY("REGION")
REFERENCES "ZREGIONS"("COD");
ALTER TABLE "ZMUNICIPALITIES" ADD CONSTRAINT ZMUNICIPALITIES_REGION_FK FOREIGN
KEY("REGION") REFERENCES "ZREGIONS"("COD");
ALTER TABLE "ZMUNICIPALITIES" ADD CONSTRAINT ZMUNICIPALITIES_DISTRICT_FK FOREIGN
KEY("DISTRICT") REFERENCES "ZDISTRICTS"("COD");
ALTER TABLE "ZFACILITIES" ADD CONSTRAINT ZFACILITIES_MUNICIPALITY_FK FOREIGN
KEY("MUNICIPALITY") REFERENCES "ZMUNICIPALITIES"("COD");
ALTER TABLE "ZFACILITIES" ADD CONSTRAINT ZFACILITIES_ROOMTYPE_FK FOREIGN KEY("ROOMTYPE")
REFERENCES "ZROOMTYPES"("ROOMTYPE");
ALTER TABLE "ZUSES" ADD CONSTRAINT ZUSES_ID_FK FOREIGN KEY("ID") REFERENCES
"ZFACILITIES"("ID");
ALTER TABLE "ZUSES" ADD CONSTRAINT ZUSES_REF_FK FOREIGN KEY("REF") REFERENCES
"ZACTIVITIES"("REF");
```

# Indexes

Here we show the additional created indexes used on the Z environment to improve performance.

## Query 1

```
CREATE UNIQUE INDEX INDEX_ZROOMTYPES_ROOMTYPE ON ZROOMTYPE (ROOMTYPE
ASC,DESCRIPTION ASC)
CREATE UNIQUE INDEX INDEX_ZACTIVITY_ACTIVITIES ON ZACTIVITIES (REF ASC, ACTIVITY
ASC);
```

The first index is used to access the table *ROOMTYPE* through the attribute description and project it to the attribute *room_type* and so using this index, we avoid the access to the full table.
The second index has the same functionality as the first index but it's on the table *ACTIVITIES*, since on the query 1 we need to access both the activity attribute and the ref attribute.

## Query 2

```
CREATE UNIQUE INDEX INDEX_ZROOMTYPES_ROOMTYPE ON ZROOMTYPES (ROOMTYPE ASC,
DESCRIPTION ASC);
CREATE UNIQUE INDEX INDEX_ZREGIONS_DESIGNATION ON ZREGIONS (COD ASC, DESIGNATION
ASC);
CREATE UNIQUE INDEX INDEX_ZMUNICIPALITIES_REGION ON ZMUNICIPALITIES (COD ASC,
REGION ASC);
CREATE BITMAP INDEX INDEX_ZFACILITY_FACILITIES ON ZFACILITIES (MUNICIPALITY
ASC,ROOMTYPE ASC);
```

Here we use again the same index used on the 1st query on the table *ZROOMTYPE* because we need to access the table to filter the attribute description (in the WHERE condition) and we also need the *roomtype* key to do the join.
The other second and third indexes are used mostly for the join and since they involve primary keys, we can use the unique index.
The last one is the most important one because it involves 2 attributes that are not primary keys, so it's better to use a bitmap instead of a unique index, and doing the test we can see that even if the cardinality is not that low, the cost of the query is lower using the bitmap instead of b-tree index.

## Query 3

```
CREATE UNIQUE INDEX INDEX_ZFACILITIES_MUNICIPALITY ON ZFACILITIES (ID,
MUNICIPALITY);
CREATE UNIQUE INDEX INDEX_ZACTIVITY_ACTIVITIES ON ZACTIVITIES (REF, ACTIVITY);
```

The indexes used for both queries 3_A and 3_B shown here are very simple : the index on *ZFACILITIES* is used mostly to avoid full access on the table for the join, and the index on *ZACTIVITIES* is used since we have a WHERE clause that involves the attribute *ACTIVITY*.

## Query 4

```
CREATE UNIQUE INDEX INDEX_ZFACILITIES_MUNICIPALITY ON ZFACILITIES (ID ASC,
MUNICIPALITY ASC);
CREATE UNIQUE INDEX INDEX_ZACTIVITIES_ACTIVITY ON ZACTIVITIES (REF ASC, ACTIVITY
ASC);
CREATE UNIQUE INDEX INDEX_ZMUNICIPALITIES_DESIGNATION ON ZMUNICIPALITIES (COD
ASC, DESIGNATION ASC);
```

These 3 indexes are the same used in the last 3 queries because some access to the tables are made on attributes and not on primary keys.

Here it is not convenient to use bit-map because the attributes involved in the same table are very less and we are using the primary key as one of the attributes of the indexes, which decreases the efficiency of the bit-map indexes.

## Query 5

```
---- Index for 5 a)
CREATE INDEX INDEX_B_TREE ON ZFACILITIES (ROOMTYPE, MUNICIPALITY);

---- Index for 5 b)
CREATE BITMAP INDEX INDEX_BITMAP ON ZFACILITIES (ROOMTYPE, MUNICIPALITY);
```

These indexes used for question 5 were instructed by the assignment, using *roomtype* and *municipality* columns on the *Facilities* table. For question 5 a) a B-Tree index is added and for 5 B) a Bitmap index is added. More details are presented at the Question 5 section here.

## Query 6

```
CREATE UNIQUE INDEX INDEX_ZMUNICIPALITIES_DISTRICT ON ZMUNICIPALITIES (COD
ASC,DISTRICT ASC);
CREATE UNIQUE INDEX INDEX_ZFACILITIES_MUNICIPALITY ON ZFACILITIES (ID ASC,
MUNICIPALITY ASC);
CREATE UNIQUE INDEX INDEX_ZDISTRICTS_COD ON ZDISTRICTS (COD ASC, DESIGNATION
ASC);
```

Most of the indexes created here are used to avoid full access to the table for the join. Here we use unique B-tree indexes and not bitmap indexes because the indexes we are creating involve primary keys and have a high cardinality, which may decrease the efficiency of the bitmap indexes.

# Questions

## Question 1

*Which are the facilities where the room type description contains 'touros' and have 'teatro' as one of their activities? Show the id, name, description and activity.*

### Query

```
SELECT id, name, description, activity
```

```
FROM xfacilities NATURAL JOIN xroomtypes NATURAL JOIN xuses
NATURAL JOIN xactivities
WHERE description LIKE '%touros%' AND activity = 'teatro';
```

## Result

| ID | NAME | DESCRIPTION | ACTIVITY |
|---|---|---|---|
| 916 | COLISEU JOSÉ RONDÃO DE ALMEIDA–EX PRAÇA DE TOIROS | Praça de touros multiusos | teatro |
| 940 | ARENA DE ÉVORA – EX PRAÇA DE TOIROS | Praça de touros multiusos | teatro |
| 957 | COLISEU DE REDONDO – EX PRAÇA DE TOIROS | Praça de touros multiusos | teatro |

## Execution Times

| **Environment** | **X** | **Y** | **Z** |
|---|---|---|---|
| Time (s) | 0.031 | 0.022 | 0.019 |
| | 0.039 | 0.019 | 0.018 |
| | 0.030 | 0.024 | 0.019 |
| | 0.055 | 0.031 | 0.019 |
| | 0.025 | 0.021 | 0.019 |
| Average Time (s) | 0.036 | 0.023 | 0.019 |

## Execution Plan

### Query using X tables

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 17 | 16 |
| HASH JOIN | | | 17 | 16 |
| Access Predicates | | | | |
| AND | | | | |
| XUSES.REF=XACTIVITIES.REF | | | | |
| XFACILITIES.ID=XUSES.ID | | | | |
| HASH JOIN | | | 54 | 12 |
| Access Predicates | | | | |
| XFACILITIES.ROOMTYPE=XROOMTYPES.ROOMTYPE | | | | |
| MERGE JOIN | | CARTESIAN | 1 | 6 |
| TABLE ACCESS | XROOMTYPES | FULL | 1 | 3 |
| Filter Predicates | | | | |
| AND | | | | |
| XROOMTYPES.DESCRIPTION LIKE '%touros%' | | | | |
| XROOMTYPES.DESCRIPTION IS NOT NULL | | | | |
| BUFFER | | SORT | 1 | 3 |
| TABLE ACCESS | XACTIVITIES | FULL | 1 | 3 |
| Filter Predicates | | | | |
| XACTIVITIES.ACTIVITY='teatro' | | | | |
| TABLE ACCESS | XFACILITIES | FULL | 1084 | 6 |
| TABLE ACCESS | XUSES | FULL | 2059 | 4 |

*Comments*

The cost of the query for the environment X is mostly dominated by the table access on the tables *XUSES* and *XFACILITIES*, which has a lot of info. In fact we can see that the cost for the full access to the 2 tables are 10 and it's more than half of the total cost of the query.

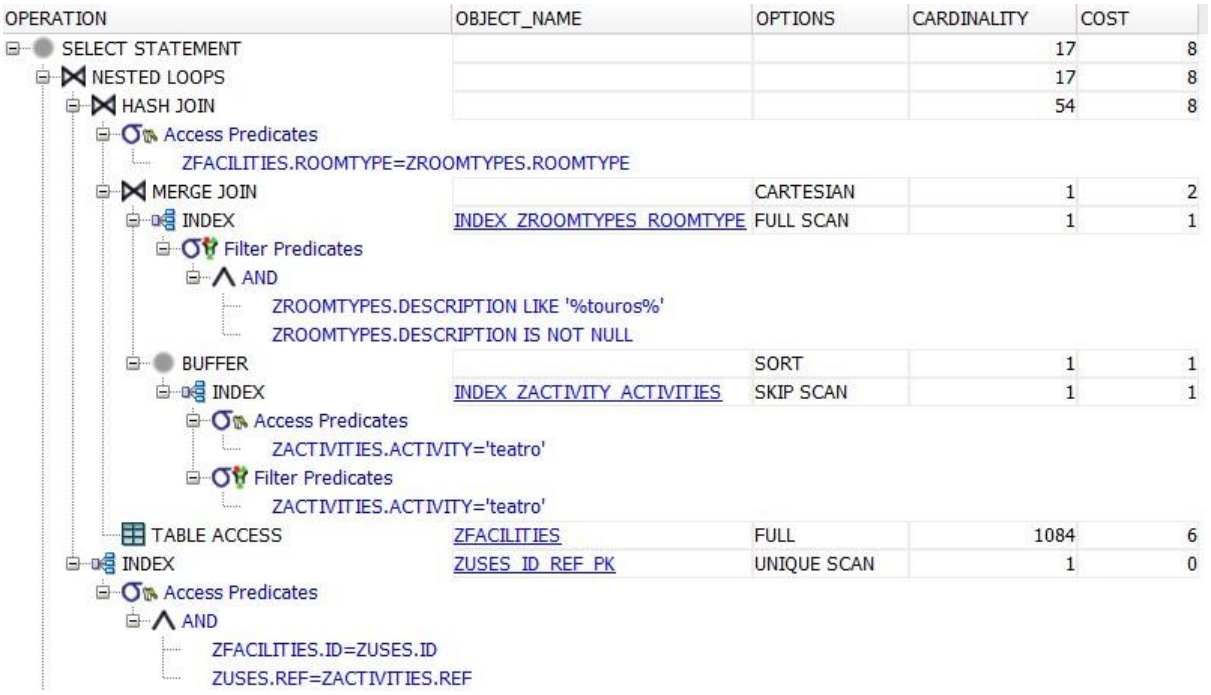
### Query using Y tables

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 17 | 12 |
| NESTED LOOPS | | | 17 | 12 |
| HASH JOIN | | | 54 | 12 |
| Access Predicates | | | | |
| YFACILITIES.ROOMTYPE=YROOMTYPES.ROOMTYPE | | | | |
| MERGE JOIN | | CARTESIAN | 1 | 6 |
| TABLE ACCESS | YROOMTYPES | FULL | 1 | 3 |
| Filter Predicates | | | | |
| AND | | | | |
| YROOMTYPES.DESCRIPTION LIKE '%touros%' | | | | |
| YROOMTYPES.DESCRIPTION IS NOT NULL | | | | |
| BUFFER | | SORT | 1 | 3 |
| TABLE ACCESS | YACTIVITIES | FULL | 1 | 3 |
| Filter Predicates | | | | |
| YACTIVITIES.ACTIVITY='teatro' | | | | |
| TABLE ACCESS | YFACILITIES | FULL | 1084 | 6 |
| INDEX | USED_ID_REF_PK | UNIQUE SCAN | 1 | 0 |
| Access Predicates | | | | |
| AND | | | | |
| YFACILITIES.ID=YUSES.ID | | | | |
| YUSES.REF=YACTIVITIES.REF | | | | |

*Comments*

Here, the cost is significantly reduced due to the primary key index on table *YFACILITIES* and *YUSES* because the join is done with the primary keys, but we still can see that half of the cost is still dominated by the access on table *YFACILITIES*, *YACTIVITIES* and *YROOMTYPES* because we are not accessing the tables using the primary key but by their attributes

Query using Z tables




*Comments*

In the environment Z, we added the 2 indexes described on the INDEX section, and we can see that the cost is reduced by ⅓ by adding the 2 indexes on *ZROOMTYPES* and *ZACTIVITY*, which each one index decreases the final cost of the query by 2 values.

## Question 2

*How many facilities with 'touros' in the room type description are there in each region?*

### Query

```
SELECT xregions.designation, count(*)
FROM xfacilities NATURAL JOIN xroomtypes INNER JOIN
xmunicipalities
ON xmunicipalities.cod = xfacilities.municipality
INNER JOIN xregions ON xmunicipalities.region = xregions.cod
```

```
WHERE description  LIKE '%touros%'
GROUP BY xregions.designation;
```

Result

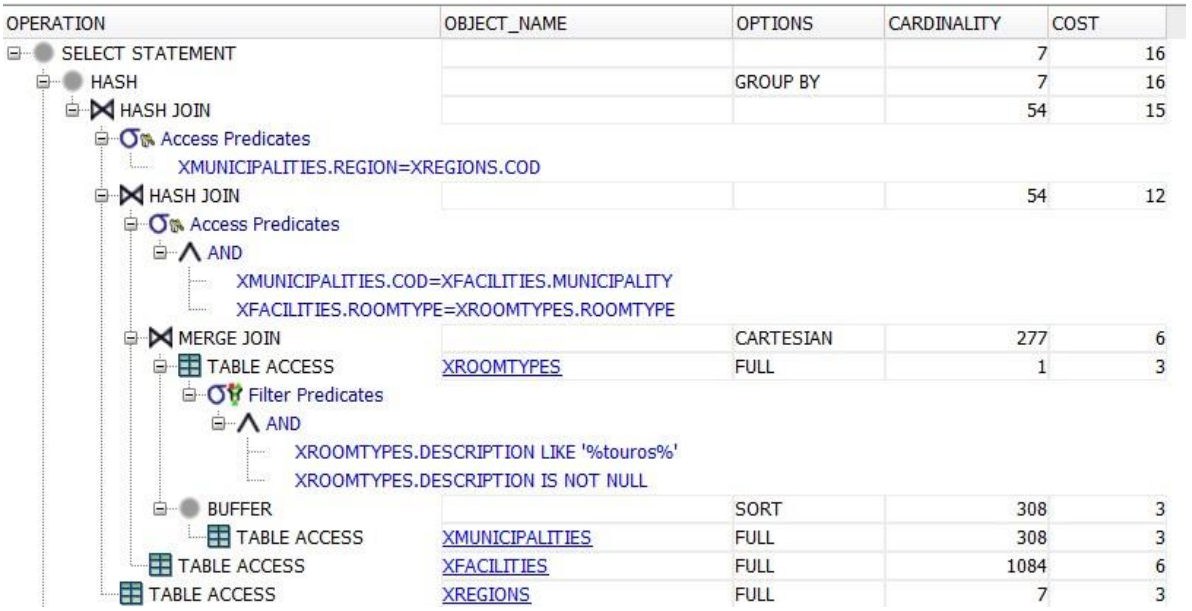
| DESIGNATION | COUNT(*) |
|---|---|
| Lisboa | 6 |
| Norte | 3 |
| Centro | 11 |
| Alentejo | 43 |
| Algarve | 1 |

Execution Times

| **Environment** | **X** | **Y** | **Z** |
|---|---|---|---|
| Time (s) | 0.031 | 0.022 | 0.020 |
| | 0.031 | 0.023 | 0.019 |
| | 0.033 | 0.021 | 0.020 |
| | 0.023 | 0.022 | 0.024 |
| | 0.025 | 0.023 | 0.019 |
| Average Time (s) | 0.028 | 0.022 | 0.020 |

## Execution Plans

### Query using X tables

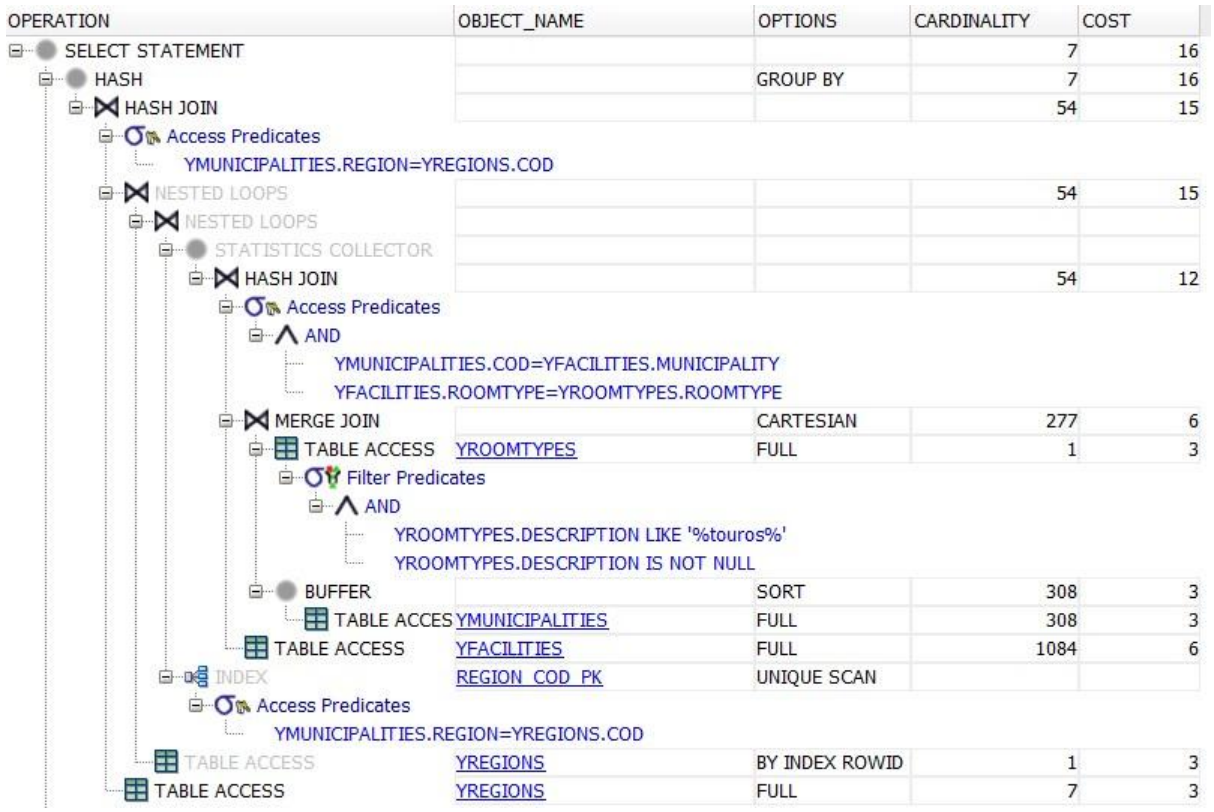


| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 7 | 16 |
| HASH | | GROUP BY | 7 | 16 |
| HASH JOIN | | | 54 | 15 |
| Access Predicates | | | | |
| XMUNICIPALITIES.REGION=XREGIONS.COD | | | | |
| HASH JOIN | | | 54 | 12 |
| Access Predicates | | | | |
| AND | | | | |
| XMUNICIPALITIES.COD=XFACILITIES.MUNICIPALITY | | | | |
| XFACILITIES.ROOMTYPE=XROOMTYPES.ROOMTYPE | | | | |
| MERGE JOIN | | CARTESIAN | 277 | 6 |
| TABLE ACCESS | XROOMTYPES | FULL | 1 | 3 |
| Filter Predicates | | | | |
| AND | | | | |
| XROOMTYPES.DESCRIPTION LIKE '%touros%' | | | | |
| XROOMTYPES.DESCRIPTION IS NOT NULL | | | | |
| BUFFER | | SORT | 308 | 3 |
| TABLE ACCESS | XMUNICIPALITIES | FULL | 308 | 3 |
| TABLE ACCESS | XFACILITIES | FULL | 1084 | 6 |
| TABLE ACCESS | XREGIONS | FULL | 7 | 3 |

*Comments*

In this execution plan the first operation is a full search on the table *XROOMTYPES* while filtering only tuples in which the description of the roomtypes contains ‘touros’. We can see here that the cost also depends on the full accesses done within the tables: *XMUNICIPALITIES*, *XFACILITIES*, *XREGIONS*.

Query using Y tables

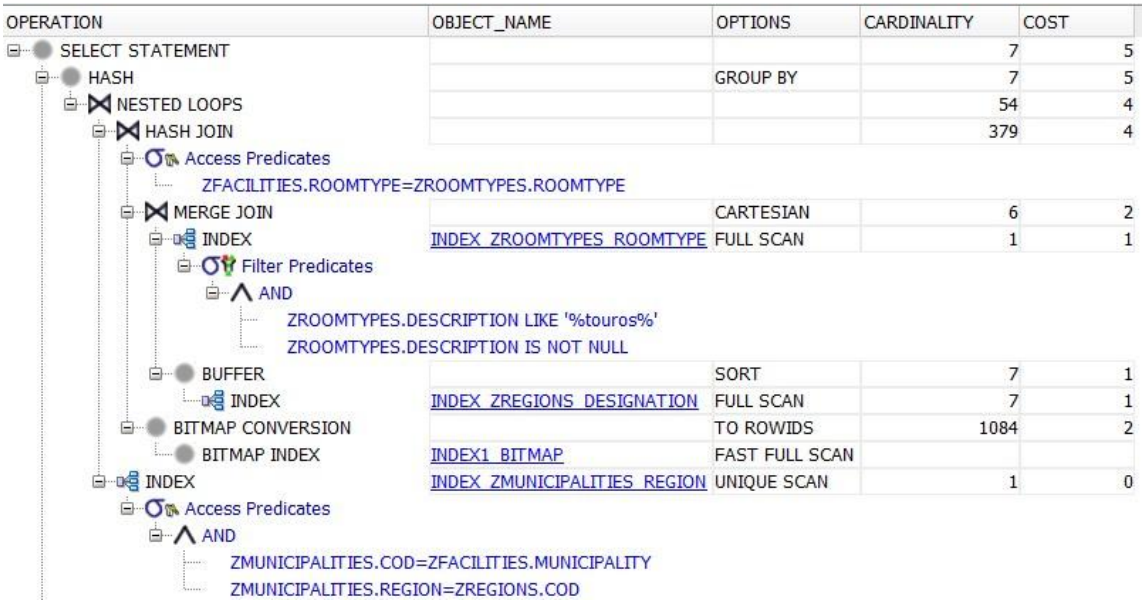


| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 7 | 16 |
| HASH | | GROUP BY | 7 | 16 |
| HASH JOIN | | | 54 | 15 |
| Access Predicates | | | | |
| YMUNICIPALITIES.REGION=YREGIONS.COD | | | | |
| NESTED LOOPS | | | 54 | 15 |
| NESTED LOOPS | | | | |
| STATISTICS COLLECTOR | | | | |
| HASH JOIN | | | 54 | 12 |
| Access Predicates | | | | |
| AND | | | | |
| YMUNICIPALITIES.COD=YFACILITIES.MUNICIPALITY | | | | |
| YFACILITIES.ROOMTYPE=YROOMTYPES.ROOMTYPE | | | | |
| MERGE JOIN | | CARTESIAN | 277 | 6 |
| TABLE ACCESS | YROOMTYPES | FULL | 1 | 3 |
| Filter Predicates | | | | |
| AND | | | | |
| YROOMTYPES.DESCRIPTION LIKE '%touros%' | | | | |
| YROOMTYPES.DESCRIPTION IS NOT NULL | | | | |
| BUFFER | | SORT | 308 | 3 |
| TABLE ACCESS | YMUNICIPALITIES | FULL | 308 | 3 |
| TABLE ACCESS | YFACILITIES | FULL | 1084 | 6 |
| INDEX | REGION_COD_PK | UNIQUE SCAN | | |
| Access Predicates | | | | |
| YMUNICIPALITIES.REGION=YREGIONS.COD | | | | |
| TABLE ACCESS | YREGIONS | BY INDEX ROWID | 1 | 3 |
| TABLE ACCESS | YREGIONS | FULL | 7 | 3 |

*Comments*

In this case, even by adding the primary key constraint in the table *YREGIONS*, a full search is still needed since the query is selecting the ‘designation’ of the regions, which is not a primary key. For what concerns the other tables, they are all fully accessed. Therefore the cost of the execution plan X and Y are the same.

- Query using Z tables



| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 7 | 5 |
| HASH | | GROUP BY | 7 | 5 |
| NESTED LOOPS | | | 54 | 4 |
| HASH JOIN | | | 379 | 4 |
| Access Predicates | | | | |
| ZFACILITIES.ROOMTYPE=ZROOMTYPES.ROOMTYPE | | | | |
| MERGE JOIN | | CARTESIAN | 6 | 2 |
| INDEX | INDEX_ZROOMTYPES_ROOMTYPE | FULL SCAN | 1 | 1 |
| Filter Predicates | | | | |
| AND | | | | |
| ZROOMTYPES.DESCRIPTION LIKE '%touros%' | | | | |
| ZROOMTYPES.DESCRIPTION IS NOT NULL | | | | |
| BUFFER | | SORT | 7 | 1 |
| INDEX | INDEX_ZREGIONS_DESIGNATION | FULL SCAN | 7 | 1 |
| BITMAP CONVERSION | | TO ROWIDS | 1084 | 2 |
| BITMAP INDEX | INDEX1_BITMAP | FAST FULL SCAN | | |
| INDEX | INDEX_ZMUNICIPALITIES_REGION | UNIQUE SCAN | 1 | 0 |
| Access Predicates | | | | |
| AND | | | | |
| ZMUNICIPALITIES.COD=ZFACILITIES.MUNICIPALITY | | | | |
| ZMUNICIPALITIES.REGION=ZREGIONS.COD | | | | |

*Comments*

In the last environment, we can see that the cost has been significantly decreased due to the indexes used (shown in the indexes paragraph). The reason why one of the indexes is a bit-map is because the attributes where we apply the bit-map indexes are not primary keys. Even if the cardinality is not as shown at lecture less than 1/100, we saw that it still has a good effort on decreasing the cost of the query.

## Question 3

*How many municipalities do not have any facility with an activity of 'cinema'?*

*a. Use not in.*

*b. Use external join and is null.*

### Query

```
---- a)
SELECT count(*) FROM
xmunicipalities
WHERE cod NOT IN
(SELECT municipality
FROM xfacilities NATURAL JOIN xuses NATURAL JOIN xactivities
INNER JOIN
xmunicipalities ON xmunicipalities.cod = xfacilities.municipality
WHERE activity = 'cinema');

---- b)
SELECT count(*)
FROM xmunicipalities LEFT OUTER JOIN (SELECT municipality as cod,
activity
FROM xfacilities NATURAL JOIN xuses NATURAL JOIN xactivities
INNER JOIN
xmunicipalities ON xmunicipalities.cod = xfacilities.municipality
WHERE activity = 'cinema') cinemas
ON xmunicipalities.cod = cinemas.cod
WHERE activity IS NULL;
```

### Result

| COUNT(*) |
| --- |
| 100 |

## Execution Times

### Query A

| Environment | X | Y | Z |
|---|---|---|---|
| Time (s) | 0.028 | 0.023 | 0.023 |
| | 0.026 | 0.022 | 0.023 |
| | 0.028 | 0.024 | 0.023 |
| | 0.025 | 0.027 | 0.021 |
| | 0.027 | 0.021 | 0.020 |
| Average Time (s) | 0.027 | 0.023 | 0.022 |

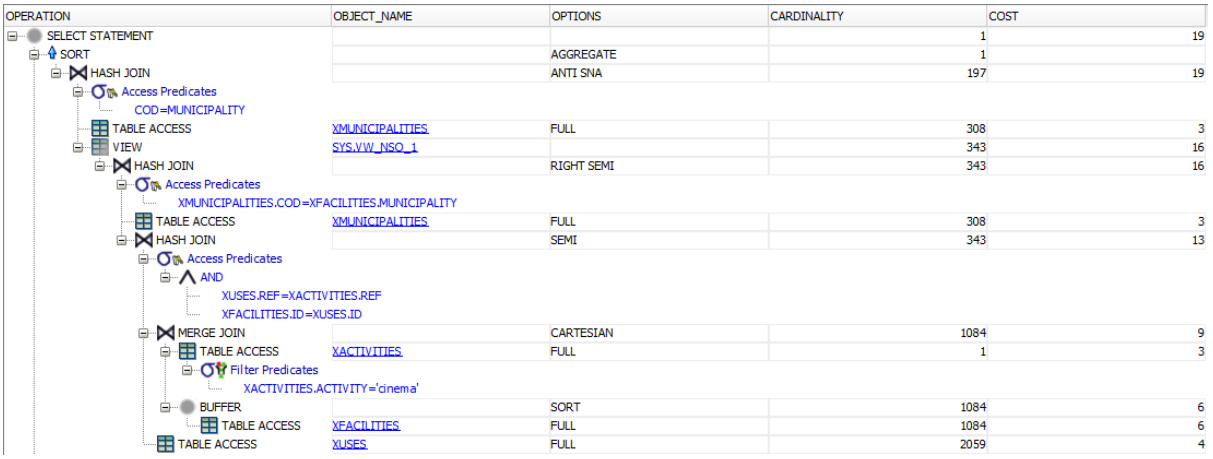
### Query B

| Environment | X | Y | Z |
|---|---|---|---|
| Time (s) | 0.024 | 0.022 | 0.023 |
| | 0.028 | 0.022 | 0.023 |
| | 0.023 | 0.024 | 0.021 |
| | 0.027 | 0.022 | 0.021 |
| | 0.025 | 0.022 | 0.023 |
| Average Time (s) | 0.025 | 0.022 | 0.022 |

# Execution Plan

## Query A

### Environment X

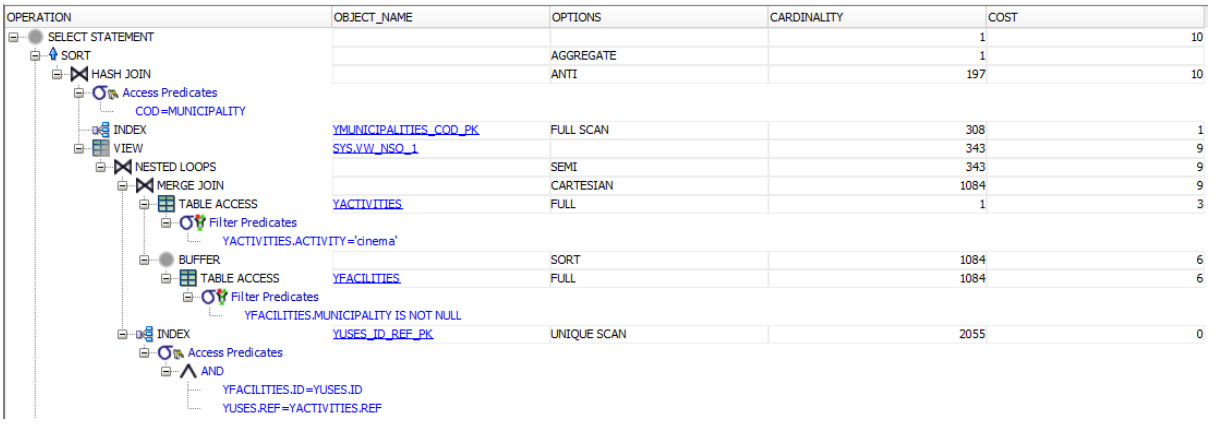


| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 1 | 19 |
| SORT | | AGGREGATE | 1 | |
| HASH JOIN | | ANTI SNA | 197 | 19 |
| Access Predicates | | | | |
| COD=MUNICIPALITY | | | | |
| TABLE ACCESS | XMUNICIPALITIES | FULL | 308 | 3 |
| VIEW | SYS.VW_NSO_1 | | 343 | 16 |
| HASH JOIN | | RIGHT SEMI | 343 | 16 |
| Access Predicates | | | | |
| XMUNICIPALITIES.COD=XFACILITIES.MUNICIPALITY | | | | |
| TABLE ACCESS | XMUNICIPALITIES | FULL | 308 | 3 |
| HASH JOIN | | SEMI | 343 | 13 |
| Access Predicates | | | | |
| AND | | | | |
| XUSES.REF=XACTIVITIES.REF | | | | |
| XFACILITIES.ID=XUSES.ID | | | | |
| MERGE JOIN | | CARTESIAN | 1084 | 9 |
| TABLE ACCESS | XACTIVITIES | FULL | 1 | 3 |
| Filter Predicates | | | | |
| XACTIVITIES.ACTIVITY='cinema' | | | | |
| BUFFER | | SORT | 1084 | 6 |
| TABLE ACCESS | XFACILITIES | FULL | 1084 | 6 |
| TABLE ACCESS | XUSES | FULL | 2059 | 4 |

*Comments*

In this environment the high cost is due to the fact that a full search is used 5 times (including twice on the table *MUNICIPALITIES*).

### Environment Y

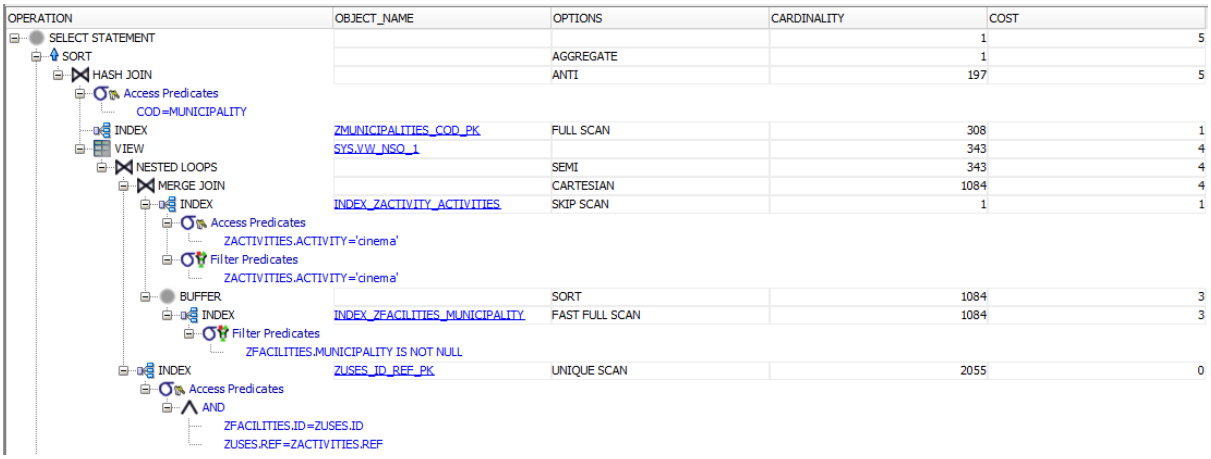


| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 1 | 10 |
| SORT | | AGGREGATE | 1 | |
| HASH JOIN | | ANTI | 197 | 10 |
| Access Predicates | | | | |
| COD=MUNICIPALITY | | | | |
| INDEX | YMUNICIPALITIES_COD_PK | FULL SCAN | 308 | 1 |
| VIEW | SYS.VW_NSO_1 | | 343 | 9 |
| NESTED LOOPS | | SEMI | 343 | 9 |
| MERGE JOIN | | CARTESIAN | 1084 | 9 |
| TABLE ACCESS | YACTIVITIES | FULL | 1 | 3 |
| Filter Predicates | | | | |
| YACTIVITIES.ACTIVITY='cinema' | | | | |
| BUFFER | | SORT | 1084 | 6 |
| TABLE ACCESS | YFACILITIES | FULL | 1084 | 6 |
| Filter Predicates | | | | |
| YFACILITIES.MUNICIPALITY IS NOT NULL | | | | |
| INDEX | YUSES_ID_REF_PK | UNIQUE SCAN | 2055 | 0 |
| Access Predicates | | | | |
| AND | | | | |
| YFACILITIES.ID=YUSES.ID | | | | |
| YUSES.REF=YACTIVITIES.REF | | | | |

*Comments*

The biggest change here is the number of full searches that is reduced in more than half. The search on the *USES* table turns into an unique scan, due to the fact that we now have primary and foreign keys, and there is no longer needed two full searches on the table *MUNICIPALITIES*. Instead only a full scan is used.

### Environment Z



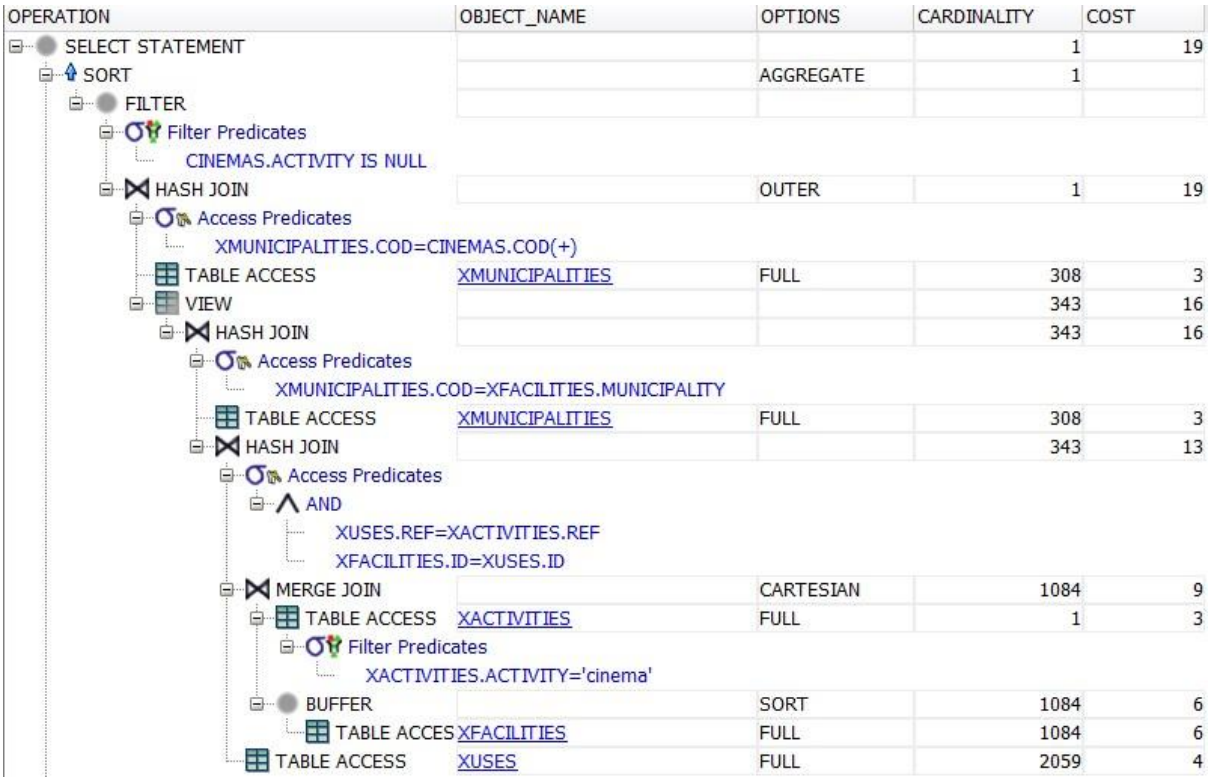
| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 1 | 5 |
| SORT | | AGGREGATE | 1 | |
| HASH JOIN | | ANTI | 197 | 5 |
| Access Predicates | | | | |
| COD=MUNICIPALITY | | | | |
| INDEX | ZMUNICIPALITIES_COD_PK | FULL SCAN | 308 | 1 |
| VIEW | SYS.VW_NSO_1 | | 343 | 4 |
| NESTED LOOPS | | SEMI | 343 | 4 |
| MERGE JOIN | | CARTESIAN | 1084 | 4 |
| INDEX | INDEX_ZACTIVITY_ACTIVITIES | SKIP SCAN | 1 | 1 |
| Access Predicates | | | | |
| ZACTIVITIES.ACTIVITY='cinema' | | | | |
| Filter Predicates | | | | |
| ZACTIVITIES.ACTIVITY='cinema' | | | | |
| BUFFER | | SORT | 1084 | 3 |
| INDEX | INDEX_ZFACILITIES_MUNICIPALITY | FAST FULL SCAN | 1084 | 3 |
| Filter Predicates | | | | |
| ZFACILITIES.MUNICIPALITY IS NOT NULL | | | | |
| INDEX | ZUSES_ID_REF_PK | UNIQUE SCAN | 2055 | 0 |
| Access Predicates | | | | |
| AND | | | | |
| ZFACILITIES.ID=ZUSES.ID | | | | |
| ZUSES.REF=ZACTIVITIES.REF | | | | |

*Comments*

With the introduction of the two indexes we now have reduced the cost by half. The index on the *FACILITIES* table makes it so that a full search is no longer needed and only a fast full search is performed. The index on the *ACTIVITIES* table allows for a skip scan to take place instead of a full search.

## Query B

### Environment X

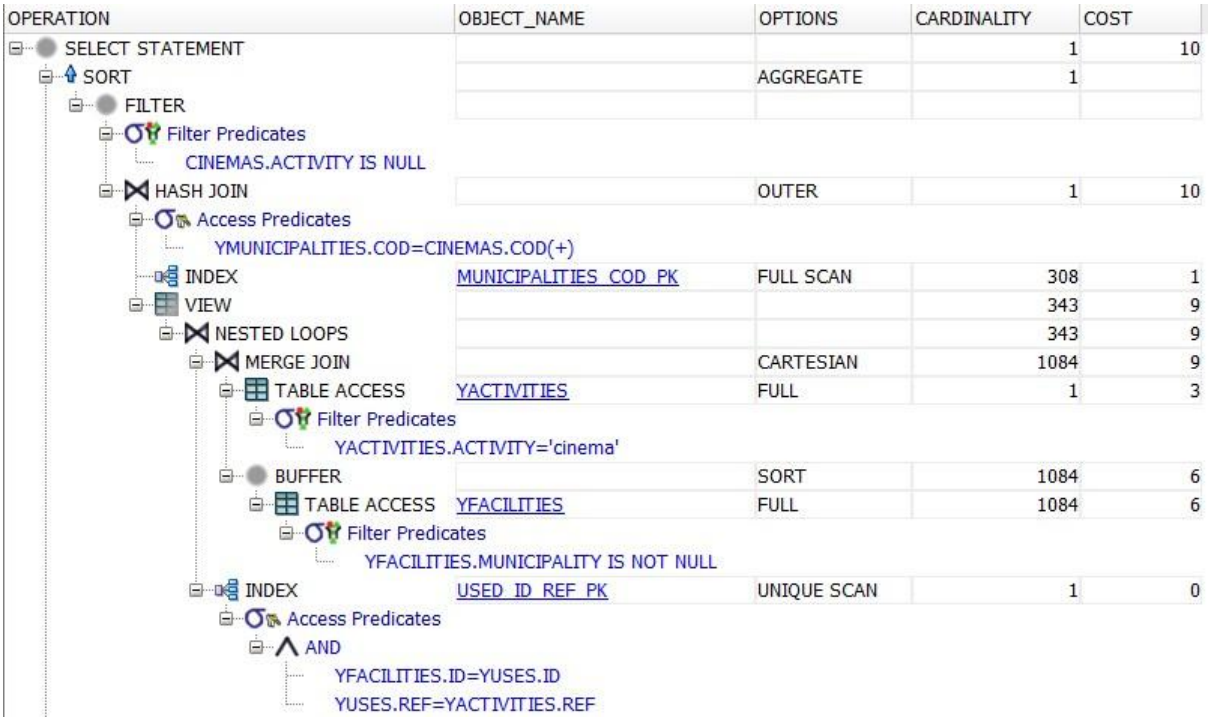


| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 1 | 19 |
| SORT | | AGGREGATE | 1 | |
| FILTER | | | | |
| Filter Predicates | | | | |
| CINEMAS.ACTIVITY IS NULL | | | | |
| HASH JOIN | | OUTER | 1 | 19 |
| Access Predicates | | | | |
| XMUNICIPALITIES.COD=CINEMAS.COD(+) | | | | |
| TABLE ACCESS | XMUNICIPALITIES | FULL | 308 | 3 |
| VIEW | | | 343 | 16 |
| HASH JOIN | | | 343 | 16 |
| Access Predicates | | | | |
| XMUNICIPALITIES.COD=XFACILITIES.MUNICIPALITY | | | | |
| TABLE ACCESS | XMUNICIPALITIES | FULL | 308 | 3 |
| HASH JOIN | | | 343 | 13 |
| Access Predicates | | | | |
| AND | | | | |
| XUSES.REF=XACTIVITIES.REF | | | | |
| XFACILITIES.ID=XUSES.ID | | | | |
| MERGE JOIN | | CARTESIAN | 1084 | 9 |
| TABLE ACCESS | XACTIVITIES | FULL | 1 | 3 |
| Filter Predicates | | | | |
| XACTIVITIES.ACTIVITY='cinema' | | | | |
| BUFFER | | SORT | 1084 | 6 |
| TABLE ACCESS | XFACILITIES | FULL | 1084 | 6 |
| TABLE ACCESS | XUSES | FULL | 2059 | 4 |

*Comments*

In the environment X, the cost is very high and we can see that this is mostly due to the amount of full access that are performed to the tables, especially to the table *XFACILITIES* and *XUSES* (just these two use up $2/3$of the total cost).

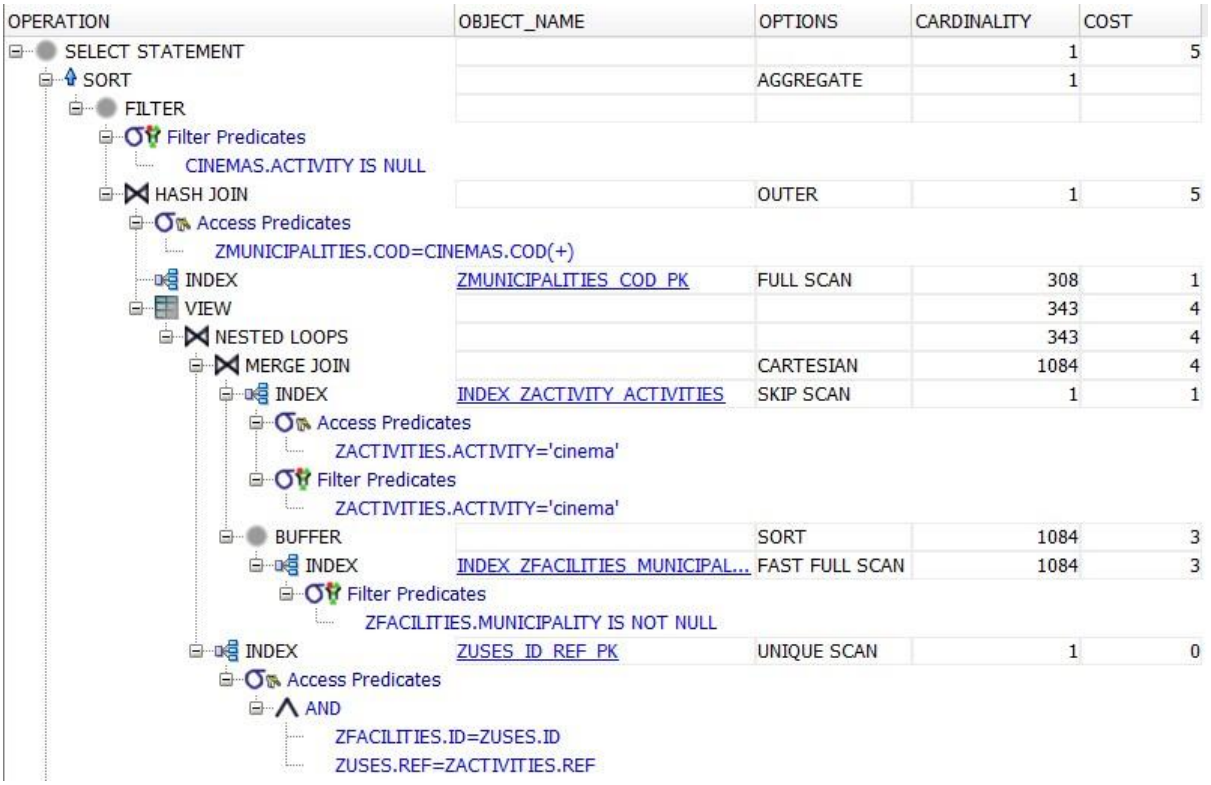
Environment Y




*Comments*

Here the cost is decreased because we are using indexes on the primary keys on table *YUSES*, *YMUNICIPALITIES* but we can still notice that there are 2 tables with the full access that can be optimized, especially the table *YFACILITIES*, where one access has a cost of 6, which is more than the half of the total cost.

Environment Z



| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 1 | 5 |
| SORT | | AGGREGATE | 1 | |
| FILTER | | | | |
| Filter Predicates | | | | |
| CINEMAS.ACTIVITY IS NULL | | | | |
| HASH JOIN | | OUTER | 1 | 5 |
| Access Predicates | | | | |
| ZMUNICIPALITIES.COD=CINEMAS.COD(+) | | | | |
| INDEX | ZMUNICIPALITIES_COD_PK | FULL SCAN | 308 | 1 |
| VIEW | | | 343 | 4 |
| NESTED LOOPS | | | 343 | 4 |
| MERGE JOIN | | CARTESIAN | 1084 | 4 |
| INDEX | INDEX_ZACTIVITY_ACTIVITIES | SKIP SCAN | 1 | 1 |
| Access Predicates | | | | |
| ZACTIVITIES.ACTIVITY='cinema' | | | | |
| Filter Predicates | | | | |
| ZACTIVITIES.ACTIVITY='cinema' | | | | |
| BUFFER | | SORT | 1084 | 3 |
| INDEX | INDEX_ZFACILITIES_MUNICIPAL... | FAST FULL SCAN | 1084 | 3 |
| Filter Predicates | | | | |
| ZFACILITIES.MUNICIPALITY IS NOT NULL | | | | |
| INDEX | ZUSES_ID_REF_PK | UNIQUE SCAN | 1 | 0 |
| Access Predicates | | | | |
| AND | | | | |
| ZFACILITIES.ID=ZUSES.ID | | | | |
| ZUSES.REF=ZACTIVITIES.REF | | | | |

*Comments*

In this last environment, we added an index on the *ZFACILITIES* and another on the *ZACTIVITIES* , where both decreased almost by half their cost to access the table (comparing the execution to the previous Y environment).

## Question 4

*Which is the municipality with more facilities engaged in each of the six kinds of activities? Show the activity, the municipality name and the corresponding number of facilities.*

### Query

```
SELECT q2.designation, q1.activity, q1.facilities
FROM (SELECT activity, max(facilities) as facilities
FROM (SELECT municipality, activity, count(*) as facilities
FROM xfacilities NATURAL JOIN xuses NATURAL JOIN xactivities
GROUP BY municipality, activity)
GROUP BY activity) q1 LEFT JOIN (SELECT designation, activity,
count(*) as facilities
FROM xmunicipalities INNER JOIN xfacilities NATURAL JOIN xuses
```

```
NATURAL JOIN xactivities
ON xmunicipalities.cod = xfacilities.municipality
GROUP BY designation, activity) q2
ON q2.activity = q1.activity AND q2.facilities = q1.facilities;
```

Result

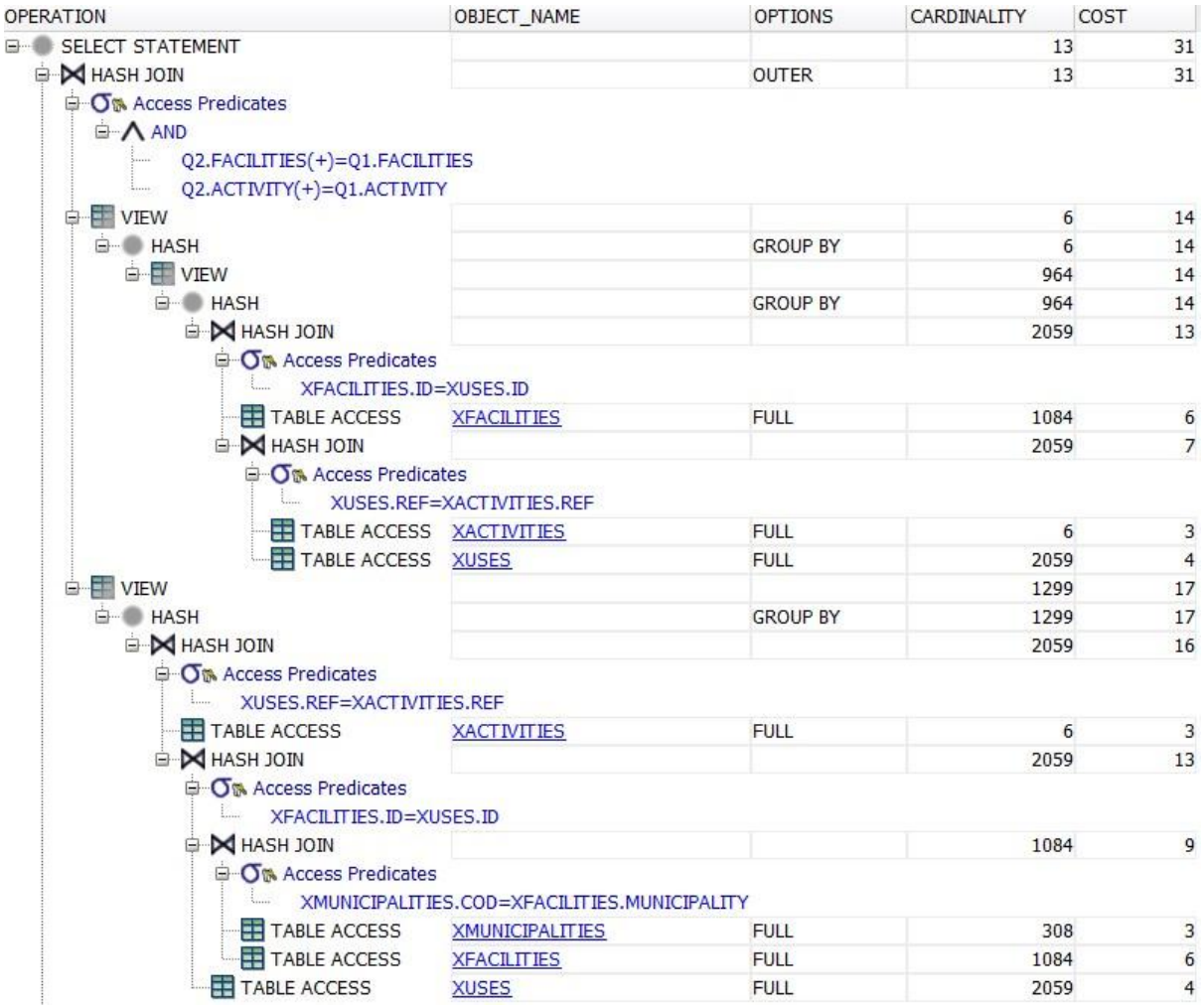
| DESIGNATION | ACTIVITY | FACILITIES |
|---|---|---|
| Lisboa | circo | 2 |
| Lisboa | dança | 47 |
| Moura | tauromaquia | 4 |
| Lisboa | música | 77 |
| Lisboa | teatro | 66 |
| Lisboa | cinema | 96 |

Execution Times

| **Environment** | **X** | **Y** | **Z** |
|---|---|---|---|
| Time (s) | 0.029 | 0.031 | 0.028 |
| | 0.032 | 0.031 | 0.029 |
| | 0.030 | 0.027 | 0.025 |
| | 0.025 | 0.027 | 0.029 |
| | 0.034 | 0.030 | 0.026 |
| Average Time | 0.030 | 0.029 | 0.0274 |

## Execution Plan

### Query using X tables

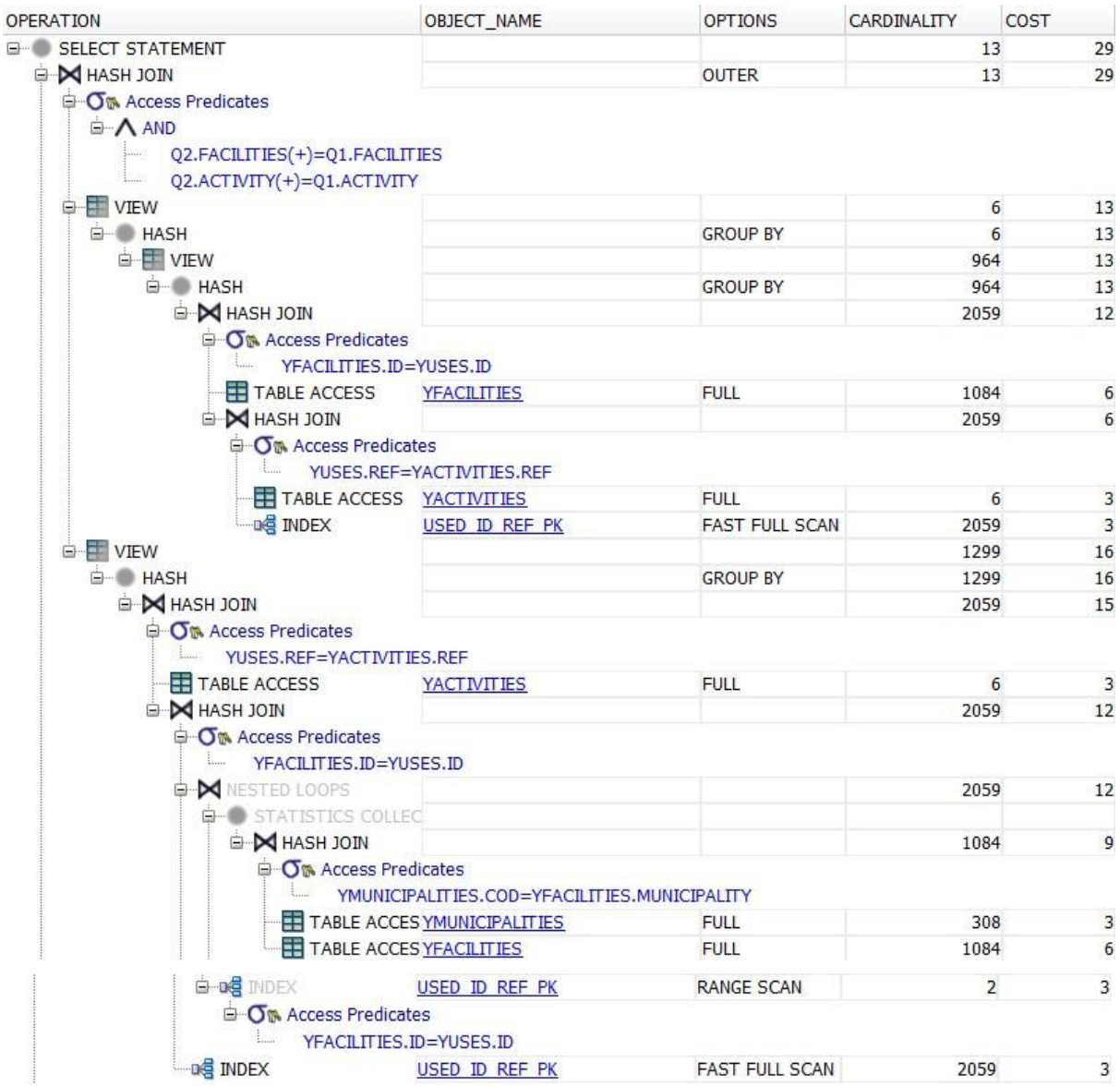


| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 13 | 31 |
| HASH JOIN | | OUTER | 13 | 31 |
| Access Predicates | | | | |
| AND | | | | |
| Q2.FACILITIES(+)=Q1.FACILITIES | | | | |
| Q2.ACTIVITY(+)=Q1.ACTIVITY | | | | |
| VIEW | | | 6 | 14 |
| HASH | | GROUP BY | 6 | 14 |
| VIEW | | | 964 | 14 |
| HASH | | GROUP BY | 964 | 14 |
| HASH JOIN | | | 2059 | 13 |
| Access Predicates | | | | |
| XFACILITIES.ID=XUSES.ID | | | | |
| TABLE ACCESS | XFACILITIES | FULL | 1084 | 6 |
| HASH JOIN | | | 2059 | 7 |
| Access Predicates | | | | |
| XUSES.REF=XACTIVITIES.REF | | | | |
| TABLE ACCESS | XACTIVITIES | FULL | 6 | 3 |
| TABLE ACCESS | XUSES | FULL | 2059 | 4 |
| VIEW | | | 1299 | 17 |
| HASH | | GROUP BY | 1299 | 17 |
| HASH JOIN | | | 2059 | 16 |
| Access Predicates | | | | |
| XUSES.REF=XACTIVITIES.REF | | | | |
| TABLE ACCESS | XACTIVITIES | FULL | 6 | 3 |
| HASH JOIN | | | 2059 | 13 |
| Access Predicates | | | | |
| XFACILITIES.ID=XUSES.ID | | | | |
| HASH JOIN | | | 1084 | 9 |
| Access Predicates | | | | |
| XMUNICIPALITIES.COD=XFACILITIES.MUNICIPALITY | | | | |
| TABLE ACCESS | XMUNICIPALITIES | FULL | 308 | 3 |
| TABLE ACCESS | XFACILITIES | FULL | 1084 | 6 |
| TABLE ACCESS | XUSES | FULL | 2059 | 4 |

*Comments*

In this case, the query requested is a little bit tricky, so the solution proposed uses 3 views as support. Since 2 of the 3 views do very similar things, the cost is inevitably increased, but it's also noticeable that with some indexes the cost of table access can be decreased, and so the total cost of the query.

Query using Y tables

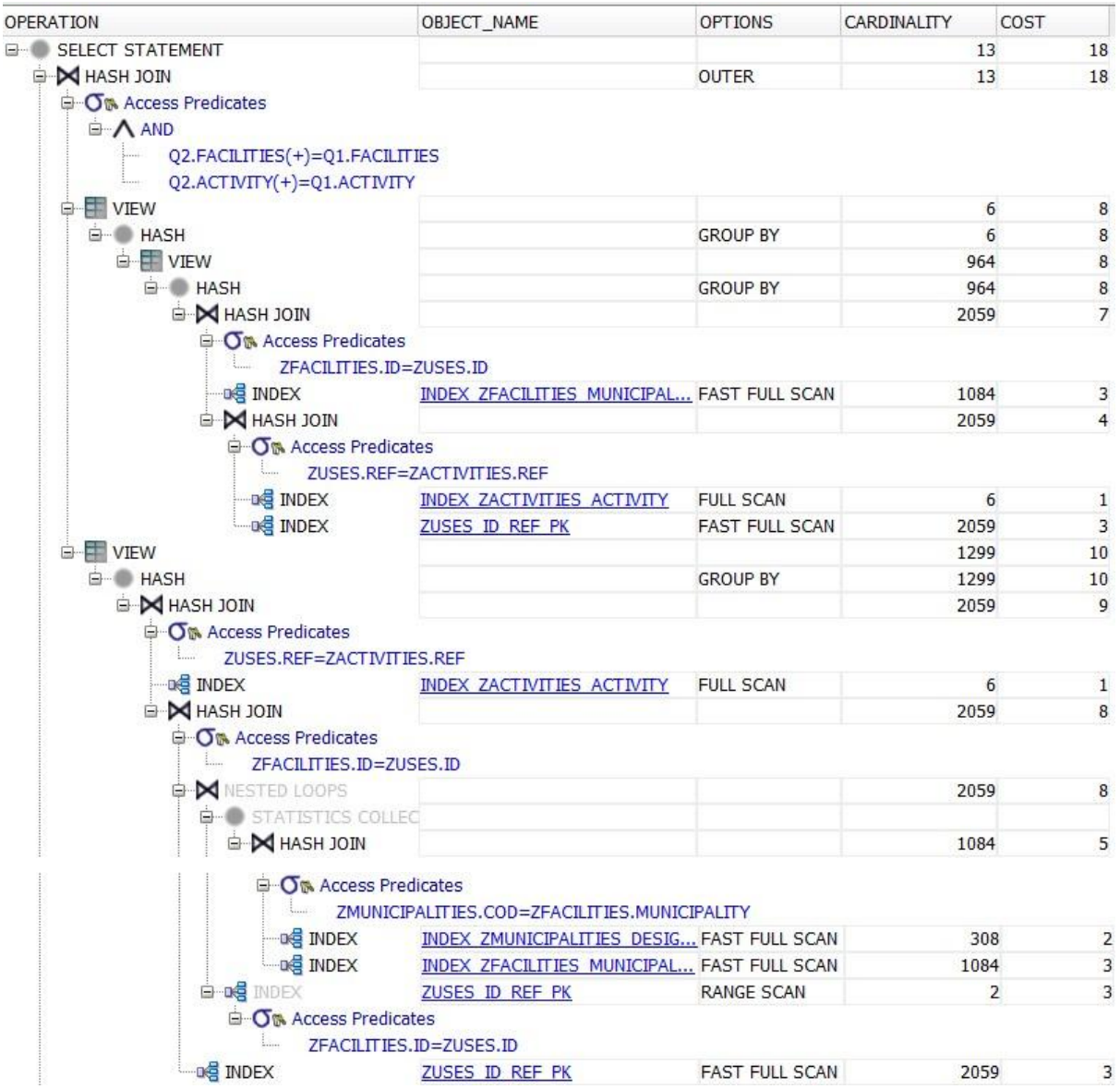


| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 13 | 29 |
| HASH JOIN | | OUTER | 13 | 29 |
| Access Predicates | | | | |
| AND | | | | |
| Q2.FACILITIES(+)=Q1.FACILITIES | | | | |
| Q2.ACTIVITY(+)=Q1.ACTIVITY | | | | |
| VIEW | | | 6 | 13 |
| HASH | | GROUP BY | 6 | 13 |
| VIEW | | | 964 | 13 |
| HASH | | GROUP BY | 964 | 13 |
| HASH JOIN | | | 2059 | 12 |
| Access Predicates | | | | |
| YFACILITIES.ID=YUSES.ID | | | | |
| TABLE ACCESS | YFACILITIES | FULL | 1084 | 6 |
| HASH JOIN | | | 2059 | 6 |
| Access Predicates | | | | |
| YUSES.REF=YACTIVITIES.REF | | | | |
| TABLE ACCESS | YACTIVITIES | FULL | 6 | 3 |
| INDEX | USED_ID_REF_PK | FAST FULL SCAN | 2059 | 3 |
| VIEW | | | 1299 | 16 |
| HASH | | GROUP BY | 1299 | 16 |
| HASH JOIN | | | 2059 | 15 |
| Access Predicates | | | | |
| YUSES.REF=YACTIVITIES.REF | | | | |
| TABLE ACCESS | YACTIVITIES | FULL | 6 | 3 |
| HASH JOIN | | | 2059 | 12 |
| Access Predicates | | | | |
| YFACILITIES.ID=YUSES.ID | | | | |
| NESTED LOOPS | | | 2059 | 12 |
| STATISTICS COLLEC | | | | |
| HASH JOIN | | | 1084 | 9 |
| Access Predicates | | | | |
| YMUNICIPALITIES.COD=YFACILITIES.MUNICIPALITY | | | | |
| TABLE ACCES | YMUNICIPALITIES | FULL | 308 | 3 |
| TABLE ACCES | YFACILITIES | FULL | 1084 | 6 |
| INDEX | USED_ID_REF_PK | RANGE SCAN | 2 | 3 |
| Access Predicates | | | | |
| YFACILITIES.ID=YUSES.ID | | | | |
| INDEX | USED_ID_REF_PK | FAST FULL SCAN | 2059 | 3 |

Comments

In the environment Y, we can see that the cost is a bit less with the introduction of indexes on the primary key like on table *YUSED*, but we see that there are still a lot of full access to the tables like on *YFACILITIES* and *YMUNICIPALITIES* that can be optimized by using appropriate indexes as shown in the environment *Z*

Query using Z tables



| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 13 | 18 |
| HASH JOIN | | OUTER | 13 | 18 |
| Access Predicates | | | | |
| AND | | | | |
| Q2.FACILITIES(+)=Q1.FACILITIES | | | | |
| Q2.ACTIVITY(+)=Q1.ACTIVITY | | | | |
| VIEW | | | 6 | 8 |
| HASH | | GROUP BY | 6 | 8 |
| VIEW | | | 964 | 8 |
| HASH | | GROUP BY | 964 | 8 |
| HASH JOIN | | | 2059 | 7 |
| Access Predicates | | | | |
| ZFACILITIES.ID=ZUSES.ID | | | | |
| INDEX | INDEX_ZFACILITIES_MUNICIPAL... | FAST FULL SCAN | 1084 | 3 |
| HASH JOIN | | | 2059 | 4 |
| Access Predicates | | | | |
| ZUSES.REF=ZACTIVITIES.REF | | | | |
| INDEX | INDEX_ZACTIVITIES_ACTIVITY | FULL SCAN | 6 | 1 |
| INDEX | ZUSES_ID_REF_PK | FAST FULL SCAN | 2059 | 3 |
| VIEW | | | 1299 | 10 |
| HASH | | GROUP BY | 1299 | 10 |
| HASH JOIN | | | 2059 | 9 |
| Access Predicates | | | | |
| ZUSES.REF=ZACTIVITIES.REF | | | | |
| INDEX | INDEX_ZACTIVITIES_ACTIVITY | FULL SCAN | 6 | 1 |
| HASH JOIN | | | 2059 | 8 |
| Access Predicates | | | | |
| ZFACILITIES.ID=ZUSES.ID | | | | |
| NESTED LOOPS | | | 2059 | 8 |
| STATISTICS COLLEC | | | | |
| HASH JOIN | | | 1084 | 5 |
| Access Predicates | | | | |
| ZMUNICIPALITIES.COD=ZFACILITIES.MUNICIPALITY | | | | |
| INDEX | INDEX_ZMUNICIPALITIES_DESIG... | FAST FULL SCAN | 308 | 2 |
| INDEX | INDEX_ZFACILITIES_MUNICIPAL... | FAST FULL SCAN | 1084 | 3 |
| INDEX | ZUSES_ID_REF_PK | RANGE SCAN | 2 | 3 |
| Access Predicates | | | | |
| ZFACILITIES.ID=ZUSES.ID | | | | |
| INDEX | ZUSES_ID_REF_PK | FAST FULL SCAN | 2059 | 3 |

Comments

Here we can see that the cost has been decreased by ⅔ comparing it to the cost of the environment Y, and that could be done only by introducing indexes on table *ZMUNICIPALITIES* and *ZFACILITIES*.
The total cost, compared to the other queries, is still very high since the query is a bit tricky but reducing almost to half the cost from environment X to environment Z using adequate indexes is a very good result.

## Question 5

*Compare the execution plans (just the Z environment) for the query giving the municipality designation, the facility name, and the room type description, where the*

*room type of the facility includes 'touros' and the municipality is part of the Porto district.*

*a. With a B-tree index on the room type and municipality columns of the facilities table;*

*b. With a bitmap index on the room type and municipality columns of the facilities table.*

## Query

```
SELECT zmunicipalities.designation, zfacilities.name,
zroomtypes.description
FROM zfacilities NATURAL JOIN zroomtypes INNER JOIN
zmunicipalities On
zmunicipalities.cod = zfacilities.municipality INNER JOIN
zdistricts ON
zdistricts.cod = zmunicipalities.district
WHERE zdistricts.designation = 'Porto' AND zroomtypes.description
LIKE '%touros%';
```

## Result

| DESIGNATION | NAME | DESCRIPTION |
|---|---|---|
| Póvoa de Varzim | PRAÇA DE TOIROS DA PÓVOA DE VARZIM | Praça de touros |

## Execution Times

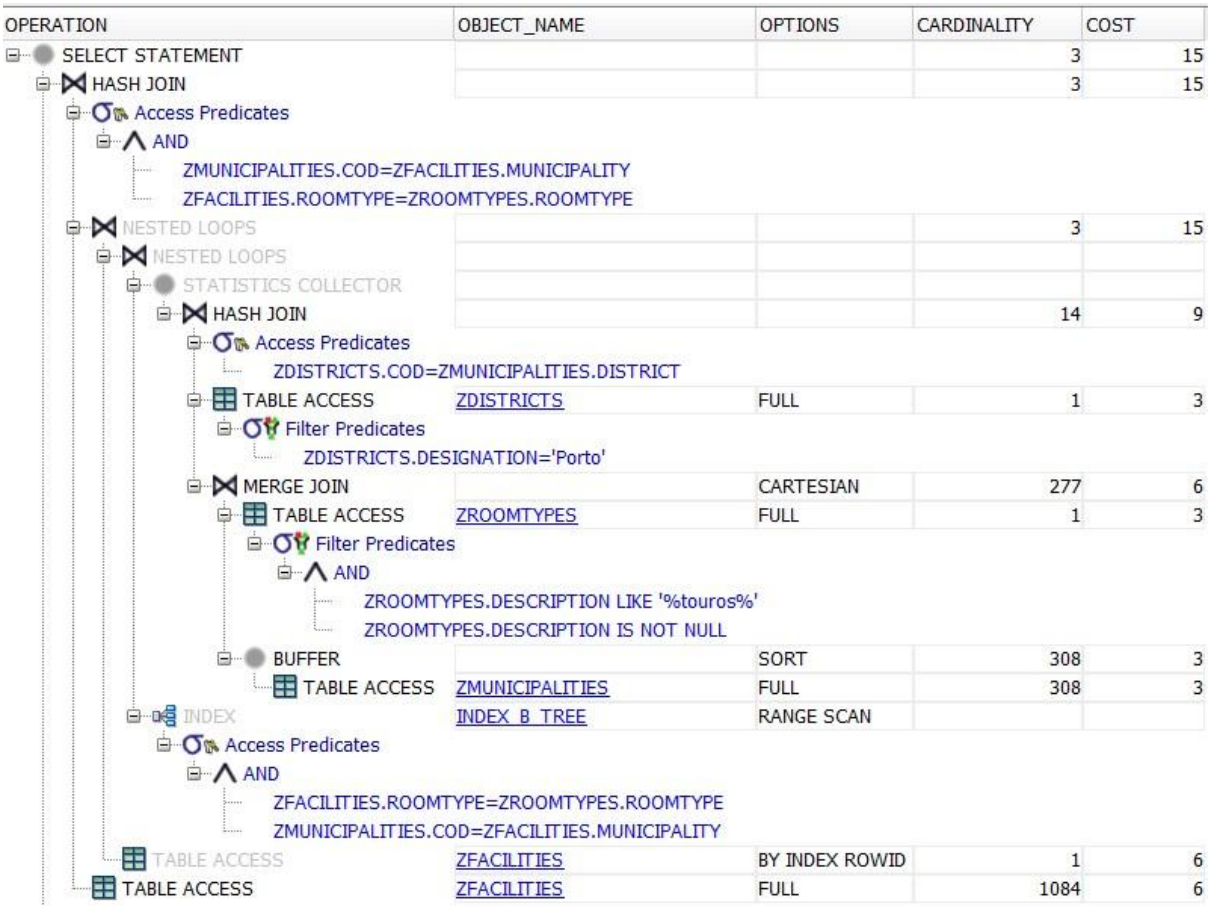
| **Index** | **B-TREE** | **BITMAP** |
|---|---|---|
| Time (s) | 0.026 | 0.024 |
| | 0.024 | 0.028 |
| | 0.025 | 0.025 |
| | 0.026 | 0.027 |
| | 0.023 | 0.029 |
| Average Time | 0.025 | 0.027 |

## Execution Plans

The main difference between both execution plans below is that, even though both indexes use the same columns/table and have the same cost, the Bitmap index is

not even used. That must be because of the cardinality of the *Facilities* table which is very high, making the B-Tree more effective.

B-Tree

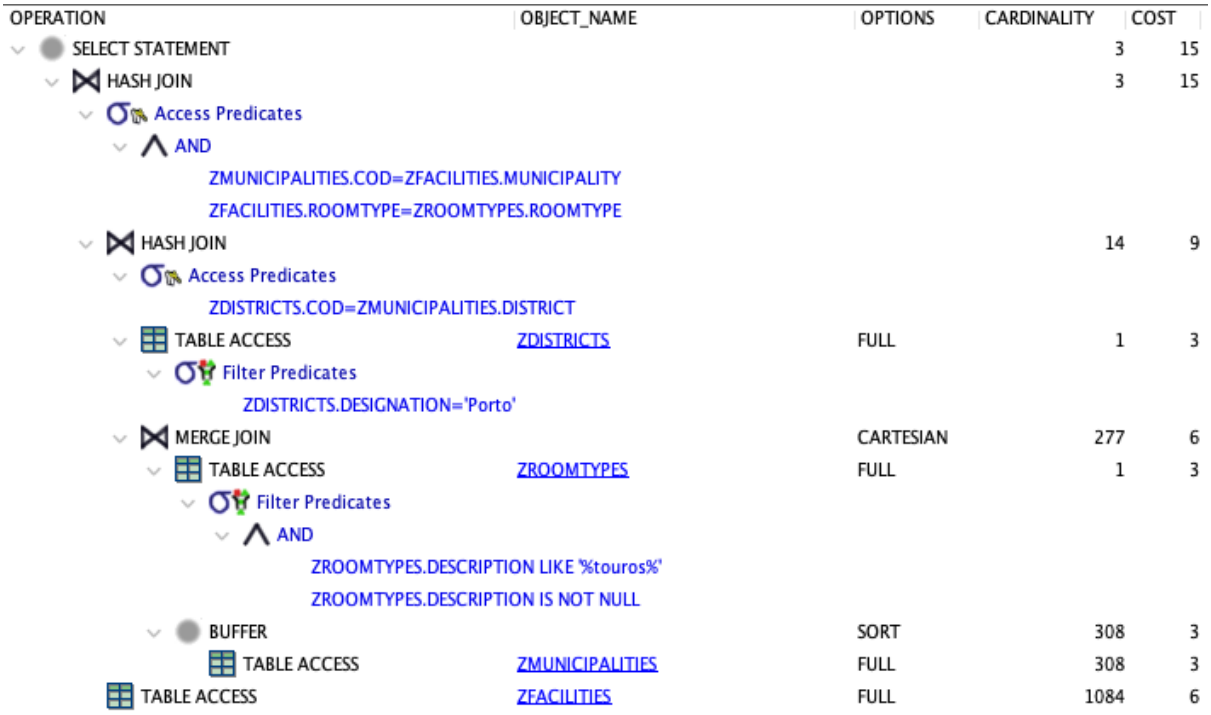


| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 3 | 15 |
| HASH JOIN | | | 3 | 15 |
| Access Predicates | | | | |
| AND | | | | |
| ZMUNICIPALITIES.COD=ZFACILITIES.MUNICIPALITY | | | | |
| ZFACILITIES.ROOMTYPE=ZROOMTYPES.ROOMTYPE | | | | |
| NESTED LOOPS | | | 3 | 15 |
| NESTED LOOPS | | | | |
| STATISTICS COLLECTOR | | | | |
| HASH JOIN | | | 14 | 9 |
| Access Predicates | | | | |
| ZDISTRICTS.COD=ZMUNICIPALITIES.DISTRICT | | | | |
| TABLE ACCESS | ZDISTRICTS | FULL | 1 | 3 |
| Filter Predicates | | | | |
| ZDISTRICTS.DESIGNATION='Porto' | | | | |
| MERGE JOIN | | CARTESIAN | 277 | 6 |
| TABLE ACCESS | ZROOMTYPES | FULL | 1 | 3 |
| Filter Predicates | | | | |
| AND | | | | |
| ZROOMTYPES.DESCRIPTION LIKE '%touros%' | | | | |
| ZROOMTYPES.DESCRIPTION IS NOT NULL | | | | |
| BUFFER | | SORT | 308 | 3 |
| TABLE ACCESS | ZMUNICIPALITIES | FULL | 308 | 3 |
| INDEX | INDEX_B_TREE | RANGE SCAN | | |
| Access Predicates | | | | |
| AND | | | | |
| ZFACILITIES.ROOMTYPE=ZROOMTYPES.ROOMTYPE | | | | |
| ZMUNICIPALITIES.COD=ZFACILITIES.MUNICIPALITY | | | | |
| TABLE ACCESS | ZFACILITIES | BY INDEX ROWID | 1 | 6 |
| TABLE ACCESS | ZFACILITIES | FULL | 1084 | 6 |

Comment

As we can see, the access to the *ROOMTYPE* and *MUNICIPALITY* on the join clause is done using the index. The only problem here is that it is doing it twice probably because some of the attributes are not part of the B-tree index.

Bitmap



| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 3 | 15 |
| HASH JOIN | | | 3 | 15 |
| Access Predicates | | | | |
| AND | | | | |
| ZMUNICIPALITIES.COD=ZFACILITIES.MUNICIPALITY | | | | |
| ZFACILITIES.ROOMTYPE=ZROOMTYPES.ROOMTYPE | | | | |
| HASH JOIN | | | 14 | 9 |
| Access Predicates | | | | |
| ZDISTRICTS.COD=ZMUNICIPALITIES.DISTRICT | | | | |
| TABLE ACCESS | ZDISTRICTS | FULL | 1 | 3 |
| Filter Predicates | | | | |
| ZDISTRICTS.DESIGNATION='Porto' | | | | |
| MERGE JOIN | | CARTESIAN | 277 | 6 |
| TABLE ACCESS | ZROOMTYPES | FULL | 1 | 3 |
| Filter Predicates | | | | |
| AND | | | | |
| ZROOMTYPES.DESCRIPTION LIKE '%touros%' | | | | |
| ZROOMTYPES.DESCRIPTION IS NOT NULL | | | | |
| BUFFER | | SORT | 308 | 3 |
| TABLE ACCESS | ZMUNICIPALITIES | FULL | 308 | 3 |
| TABLE ACCESS | ZFACILITIES | FULL | 1084 | 6 |

Comment

Using the bitmap index we can see that it is not used and the cost remains the same.

## Question 6

*Which are the codes and designations of the districts with facilities in all the municipalities?*

### Query

```
SELECT cod, designation FROM xdistricts WHERE cod NOT IN
(
SELECT xdistricts.cod
FROM xmunicipalities INNER JOIN xdistricts ON
xmunicipalities.district = xdistricts.cod
LEFT OUTER JOIN xfacilities on xfacilities.municipality =
xmunicipalities.cod
WHERE id is null
);
```

Result

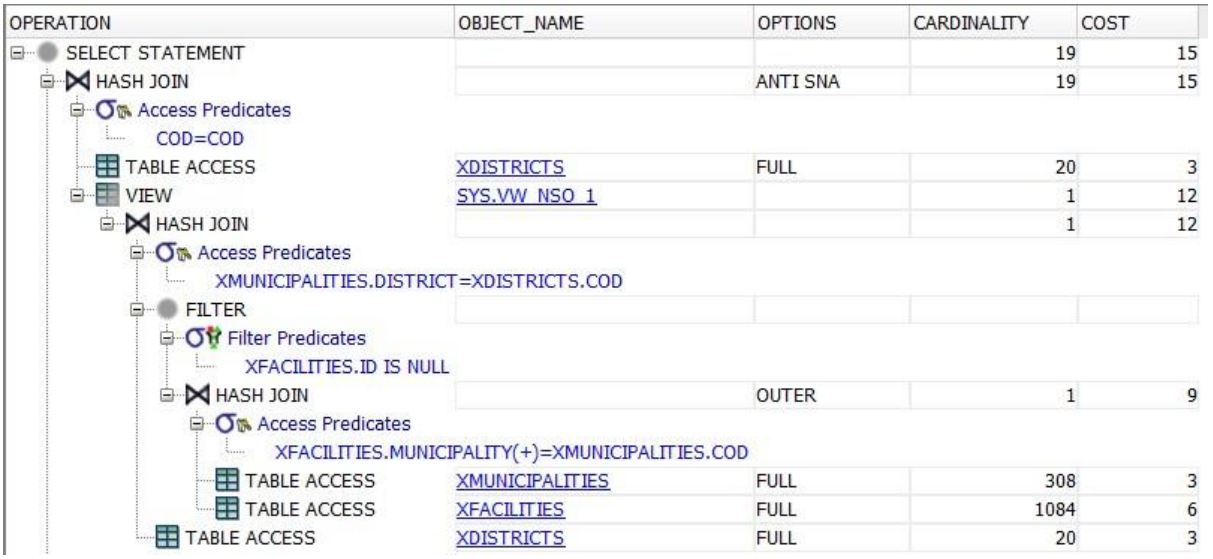
| COD | DESIGNATION |
|---|---|
| 15 | Setúbal |
| 7 | Évora |
| 11 | Lisboa |
| 12 | Portalegre |

Execution Times

| **Environment** | **X** | **Y** | **Z** |
|---|---|---|---|
| Time (s) | 0.026 | 0.022 | 0.019 |
| | 0.034 | 0.021 | 0.020 |
| | 0.035 | 0.020 | 0.018 |
| | 0.031 | 0.022 | 0.020 |
| | 0.028 | 0.035 | 0.019 |
| Average Time | 0.031 | 0.024 | 0.019 |

## Execution Plans

### Query using X tables

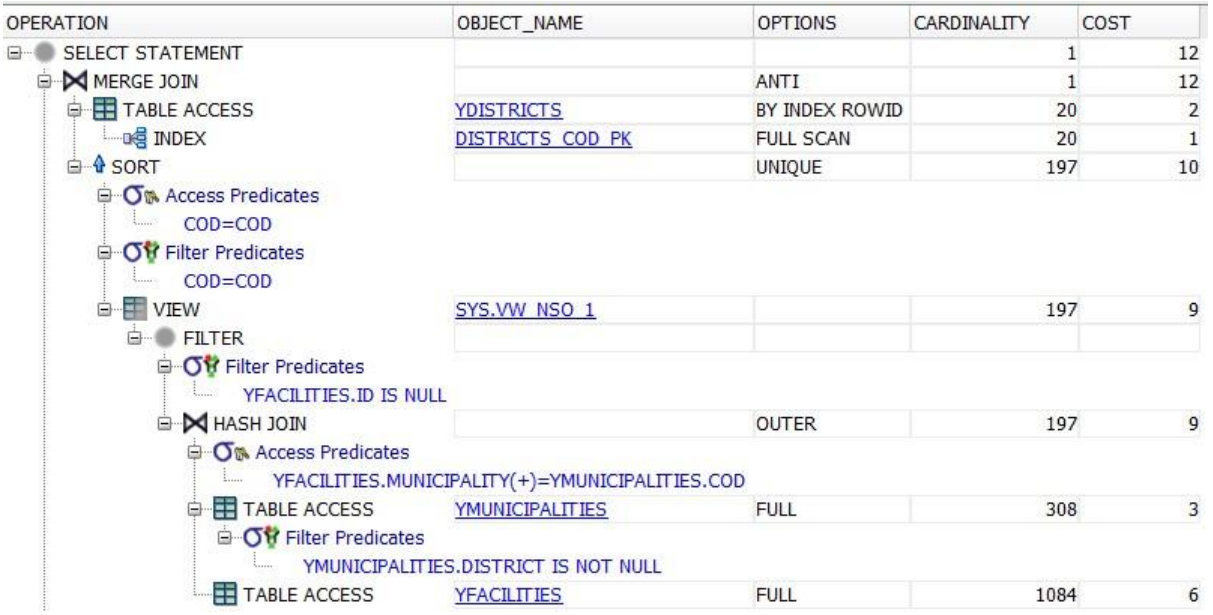


| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 19 | 15 |
| HASH JOIN | | ANTI SNA | 19 | 15 |
| Access Predicates | | | | |
| COD=COD | | | | |
| TABLE ACCESS | XDISTRICTS | FULL | 20 | 3 |
| VIEW | SYS.VW_NSO_1 | | 1 | 12 |
| HASH JOIN | | | 1 | 12 |
| Access Predicates | | | | |
| XMUNICIPALITIES.DISTRICT=XDISTRICTS.COD | | | | |
| FILTER | | | | |
| Filter Predicates | | | | |
| XFACILITIES.ID IS NULL | | | | |
| HASH JOIN | | OUTER | 1 | 9 |
| Access Predicates | | | | |
| XFACILITIES.MUNICIPALITY(+)=XMUNICIPALITIES.COD | | | | |
| TABLE ACCESS | XMUNICIPALITIES | FULL | 308 | 3 |
| TABLE ACCESS | XFACILITIES | FULL | 1084 | 6 |
| TABLE ACCESS | XDISTRICTS | FULL | 20 | 3 |

*Comments*

As in previous queries, in the environment X, tables are fully accessed. First, a full search of *XMUNICIPALITIES* and *XFACILITIES* are done. Then only the district codes of tuples containing *null* values in the *Facilities' ID* field are selected. Using *NOT IN* operator, these results are the ones that are not considered during the full access on the table *XDISTRICTS*.

Query using Y tables



| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 1 | 12 |
| MERGE JOIN | | ANTI | 1 | 12 |
| TABLE ACCESS | YDISTRICTS | BY INDEX ROWID | 20 | 2 |
| INDEX | DISTRICTS_COD_PK | FULL SCAN | 20 | 1 |
| SORT | | UNIQUE | 197 | 10 |
| Access Predicates | | | | |
| COD=COD | | | | |
| Filter Predicates | | | | |
| COD=COD | | | | |
| VIEW | SYS.VW_NSO_1 | | 197 | 9 |
| FILTER | | | | |
| Filter Predicates | | | | |
| YFACILITIES.ID IS NULL | | | | |
| HASH JOIN | | OUTER | 197 | 9 |
| Access Predicates | | | | |
| YFACILITIES.MUNICIPALITY(+)=YMUNICIPALITIES.COD | | | | |
| TABLE ACCESS | YMUNICIPALITIES | FULL | 308 | 3 |
| Filter Predicates | | | | |
| YMUNICIPALITIES.DISTRICT IS NOT NULL | | | | |
| TABLE ACCESS | YFACILITIES | FULL | 1084 | 6 |

*Comments*

In the environment Y for this query the cost is reduced due to the introduction of the primary key in the *YDISTRICTS* table, named ‘cod’. The reduction of the cost is not significant, since full accesses are still done on the tables *YMUNICIPALITIES* and *YFACILITIES*. To improve the search on these tables we should add specific indexes.

Query using Z tables

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 1 | 5 |
| HASH JOIN | | ANTI | 1 | 5 |
| Access Predicates | | | | |
| COD=COD | | | | |
| INDEX | INDEX_ZDISTRICTS_COD | FULL SCAN | 20 | 1 |
| VIEW | SYS.VW_NSO_1 | | 197 | 4 |
| FILTER | | | | |
| Filter Predicates | | | | |
| ZFACILITIES.ID IS NULL | | | | |
| HASH JOIN | | OUTER | 197 | 4 |
| Access Predicates | | | | |
| ZFACILITIES.MUNICIPALITY(+)=ZMUNICIPALITIES.COD | | | | |
| INDEX | INDEX_ZMUNICIPALITIES_DISTR... | FULL SCAN | 308 | 1 |
| Filter Predicates | | | | |
| ZMUNICIPALITIES.DISTRICT IS NOT NULL | | | | |
| INDEX | INDEX_ZFACILITIES_MUNICIPAL... | FAST FULL SCAN | 1084 | 3 |

*Comments*

In the last environment the cost is definitely improved since it is reduced by ⅓ compared to the cost from environment X and almost ⅓ compared to the cost from the environment Y. From this, it was our understanding that the cost is reduced especially when appropriate indexes are introduced, in this case, on the table *ZFACILITIES* and *ZMUNICIPALITIS*.

# Conclusion

With the development of the project, the group learned more about the impact of indexes on queries and how it can benefit the overall performance. Still, the indexes must be carefully selected, especially regarding its type (B-Tree or Bitmap), since it may not be effective in some circumstances and just create additional costs.