

**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**](#_mnfi9uht1mvk) **3**

[**Constraints**](#_hvhom9usdt1t) **3**

[**Indexes**](#_a4r1bfb7ckhk) **4**

[**Questions**](#_o52g7j7dxlk4) **6**

[Question 1](#_fyksuoltuuja) 6

[Query](#_jucno5sns50p) 6

[Result](#_6fd37268zpi) 7

[Execution Times](#_hlz4d4xkcarh) 7

[Execution Plan](#_bu2bylrfd5hj) 8

[Query using X tables](#_3m8w8e21vmt4) 8

[Comments](#_5t3vr55sodpd) 8

[Query using Y tables](#_v9y3q7l5c5r2) 8

[Comments](#_3s2e8rk7el04) 9

[Query using Z tables](#_m45w33tokrq2) 9

[Comments](#_uwxvdjahdjhu) 9

[Question 2](#_z1eu27utqtm4) 9

[Query](#_2olqm27x7x6m) 9

[Result](#_6jnivd29z7zq) 10

[Execution Times](#_ek7j1xyaw0cn) 10

[Execution Plans](#_y1fhbh23rdj9) 11

[Query using X tables](#_aru6o5l3dd6y) 11

[Comments](#_scah34xv2zzd) 11

[Query using Y tables](#_pfp70np13ylv) 12

[Comments](#_yvmhv7af8qua) 12

[Query using Z tables](#_ylfir7vlwjng) 12

[Comments](#_52dvzql8lrng) 13

[Question 3](#_k9z14orjgs73) 13

[Query](#_bd1vs4p2hs7s) 13

[Result](#_s0r0l54a0fky) 13

[Execution Times](#_1gsq2yiswsoa) 14

[Query A](#_rpc4ujynt5pn) 14

[Query B](#_3tevzgkpsuli) 14

[Execution Plan](#_u2bhkvv161es) 15

[Query A](#_r16hku1atyhq) 15

[Environment X](#_8sq1r34x9dzs) 15

[Comments](#_6ywa3xancclh) 15

[Environment Y](#_2g0k4ufs5qf3) 15

[Comments](#_h41c8ki8x5tj) 15

[Environment Z](#_2e5wvngvd3zz) 16

[Comments](#_5h46ko8tdys0) 16

[Query B](#_2toi54oo02jd) 16

[Environment X](#_noy00yyfycig) 16

[Comments](#_ezilj3jyctp) 17

[Environment Y](#_3nvy99tkwire) 17

[Comments](#_dvtg1q65aq00) 17

[Environment Z](#_t94vdooojlte) 18

[Comments](#_oousgztz4xrx) 18

[Question 4](#_m30ghe109d2h) 18

[Query](#_vf8a5lrdw8ug) 18

[Result](#_4cew8cm3f04d) 18

[Execution Times](#_cb0aph7ovgwv) 19

[Execution Plan](#_smlmy3v5mjtv) 20

[Query using X tables](#_jiu3oqbyzpc) 20

[Comments](#_rcrazge427b9) 20

[Query using Y tables](#_k4wuch9tat2x) 21

[Comments](#_gul9ko5lt0vo) 21

[Query using Z tables](#_2bwpzjpb8w8k) 22

[Comments](#_pwp2aor2huas) 22

[Question 5](#_xt4dvfjrbjae) 22

[Query](#_imonbr7f9tk8) 23

[Result](#_8lnu9iwthcf3) 23

[Execution Times](#_yrzgoqov2p8i) 23

[Execution Plans](#_wa1holi7577g) 23

[B-Tree](#_ncl8s325pu4l) 23

[Bitmap](#_hgi3ajid2lto) 25

[Question 6](#_heybadi9bp5c) 25

[Query](#_abfw78h0sr6x) 25

[Result](#_36d0hnlco7xi) 25

[Execution Times](#_lk4p7j3op6o5) 26

[Execution Plans](#_6873tii9jql9) 27

[Query using X tables](#_imn5f3y4ingf) 27

[Comments](#_ucgee2fw58bi) 27

[Query using Y tables](#_y0kvsljhpnth) 27

[Comments](#_7h7zpnz9tbu2) 27

[Query using Z tables](#_1yxyd3r4q9cl) 27

[Comments](#_m5hqro7oy95j) 28

[**Conclusion**](#_xuvcavhbyif) **28**

# 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 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](#_xt4dvfjrbjae).

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'; |

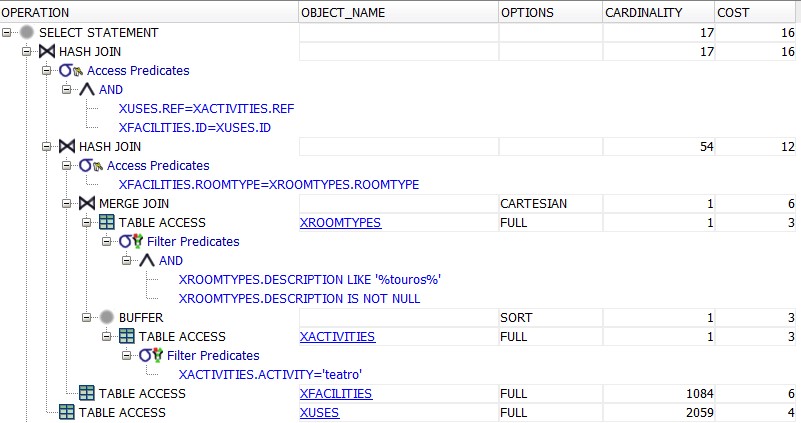
### Result

### Execution Times

|  |  |  |  |
| --- | --- | --- | --- |
| **Environment** | **X** | **Y** | **Z** |
|  | 0.031 | 0.022 | 0.019 |
|  | 0.039 | 0.019 | 0.018 |
| Time (s) | 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

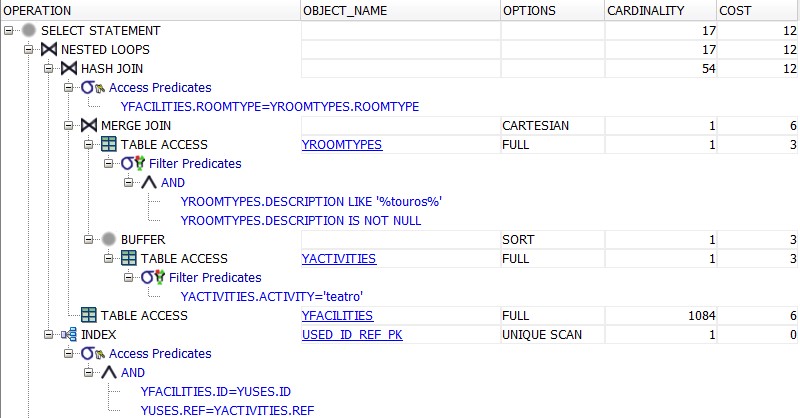
#### Query using X tables



###### 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 the half of the total cost of the query

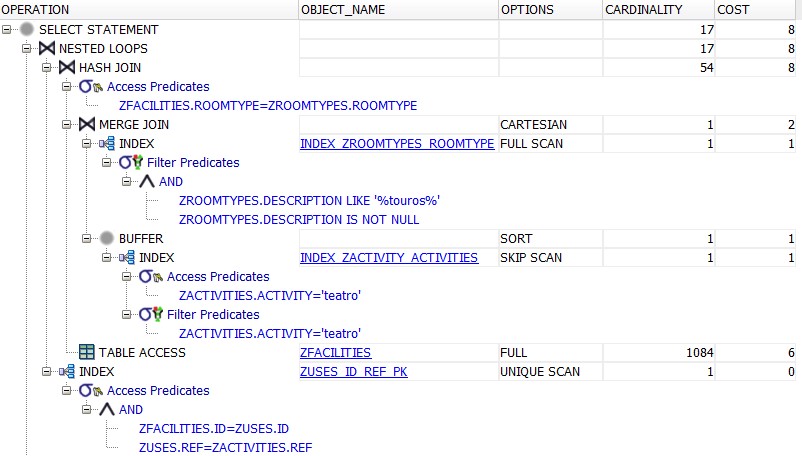
#### Query using Y tables



###### 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

### 

### Execution Times

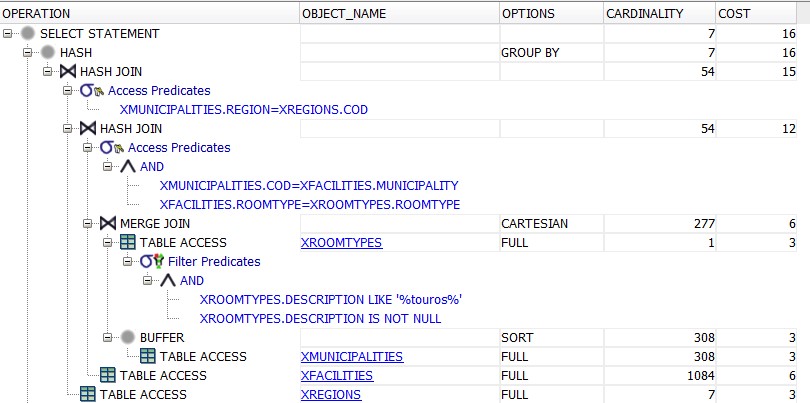
### 

|  |  |  |  |
| --- | --- | --- | --- |
| **Environment** | **X** | **Y** | **Z** |
|  | 0.031 | 0.022 | 0.020 |
|  | 0.031 | 0.023 | 0.019 |
| Time (s) | 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

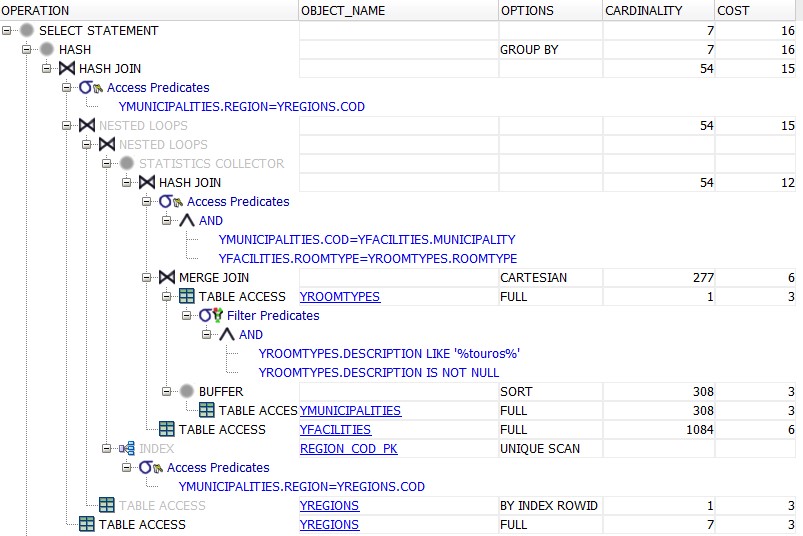
#### Query using X tables



###### 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.

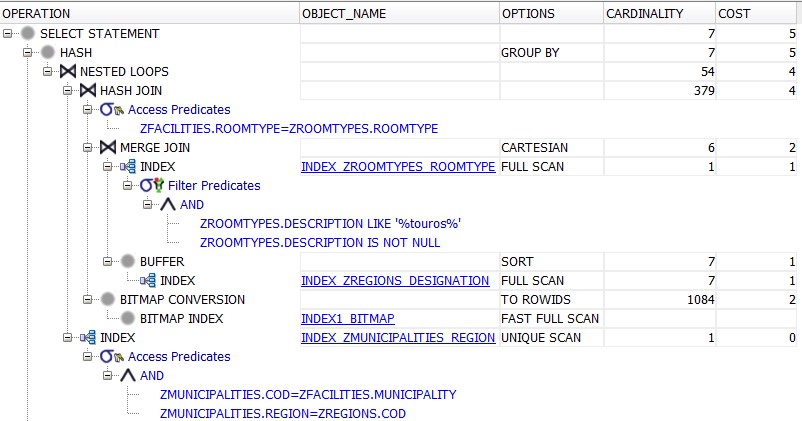
#### Query using Y tables



###### 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



###### 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’?*

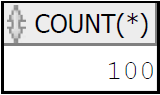
*a. Use not in.*

*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



### Execution Times

#### Query A

|  |  |  |  |
| --- | --- | --- | --- |
| **Environment** | **X** | **Y** | **Z** |
|  | 0.028 | 0.023 | 0.023 |
|  | 0.026 | 0.022 | 0.023 |
| Time (s) | 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

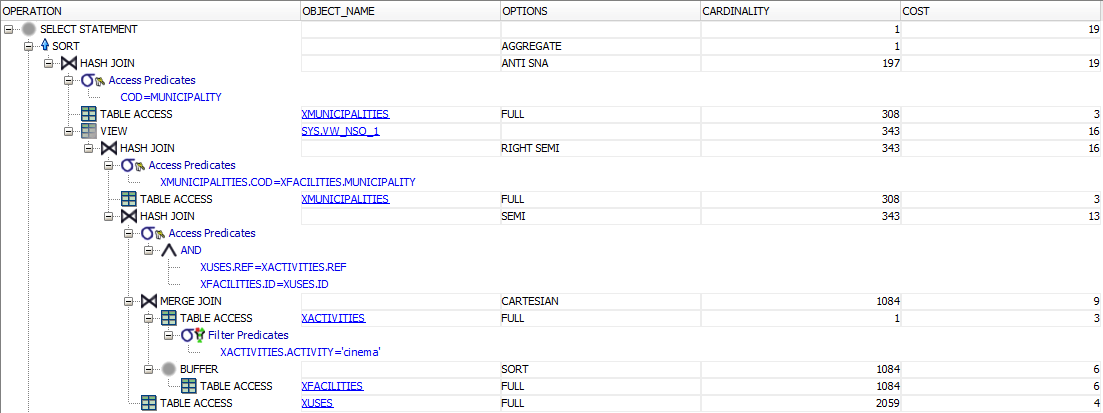
|  |  |  |  |
| --- | --- | --- | --- |
| **Environment** | **X** | **Y** | **Z** |
|  | 0.024 | 0.022 | 0.023 |
|  | 0.028 | 0.022 | 0.023 |
| Time (s) | 0.023 | 0.024 | 0.021 |
|  | 0.027 | 0.022 | 0.021 |
|  | 0.025 | 0.022 | 0.023 |
| Average Time (s) | 0.025 | 0.022 | 0.022 |

### 

### Execution Plan

#### Query A

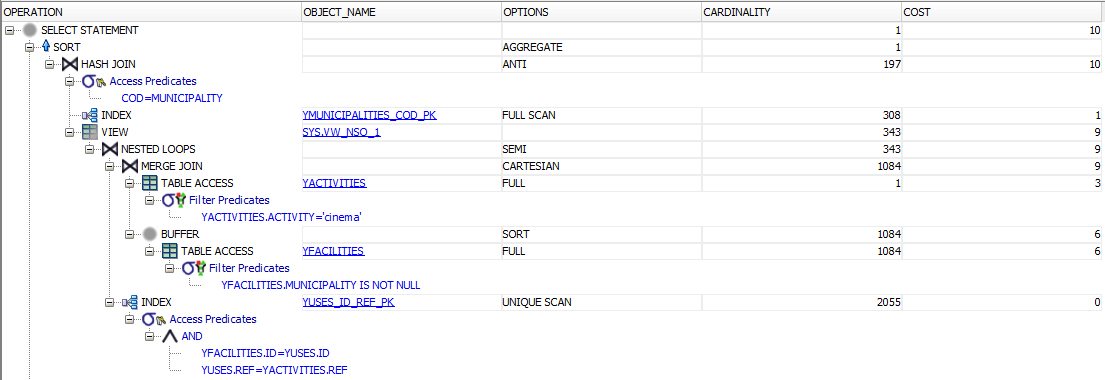
##### Environment X



###### 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).

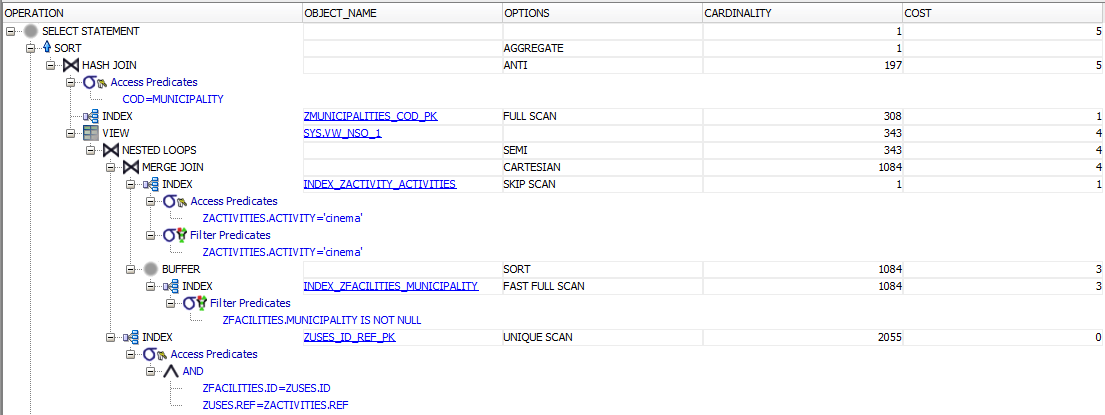
##### Environment Y



###### 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

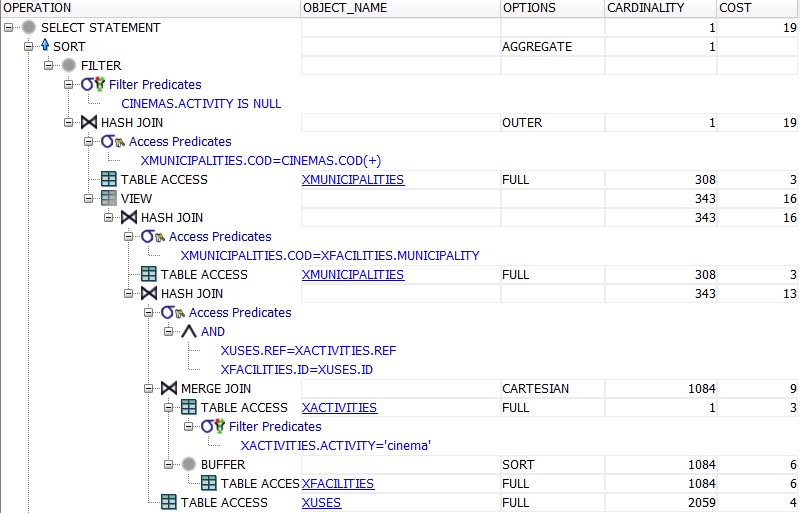


###### Comments

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 scan is performed. The index on the ACTIVITIES table allows for a skip scan to take place instead of a full search.

#### Query B

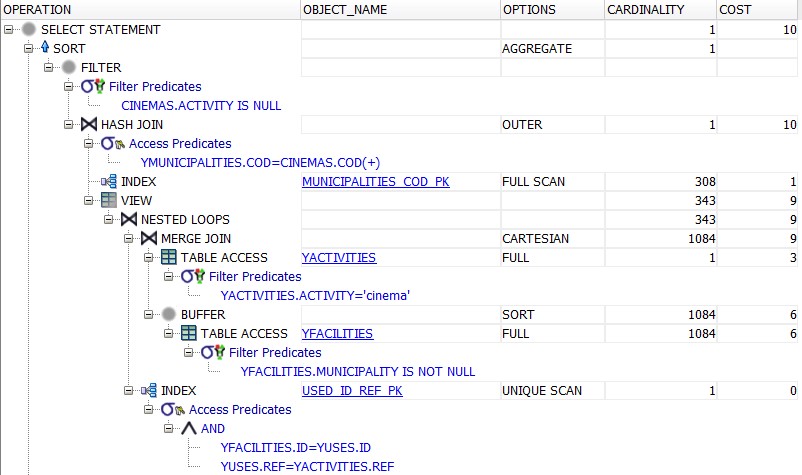
##### Environment X



###### Comments

In the environment X, the cost is very high and we can see that most of the cost is caused by the full access to the tables, especially to table XFACILITIES and XUSES (just they 2 occupies the of the total cost).

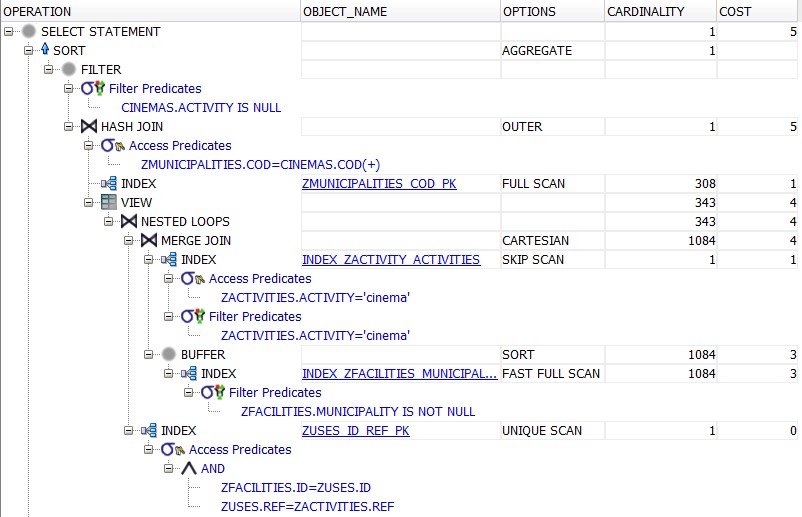
##### Environment Y



###### Comments

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

##### Environment Z



###### Comments

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

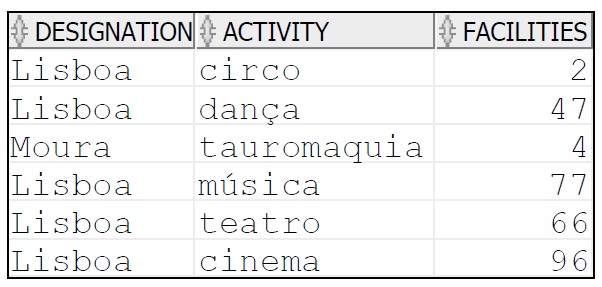
## Question 4

*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



### Execution Times

|  |  |  |  |
| --- | --- | --- | --- |
| **Environment** | **X** | **Y** | **Z** |
|  | 0.029 | 0.031 | 0.028 |
|  | 0.032 | 0.031 | 0.029 |
| Time (s) | 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

##### *Comments*

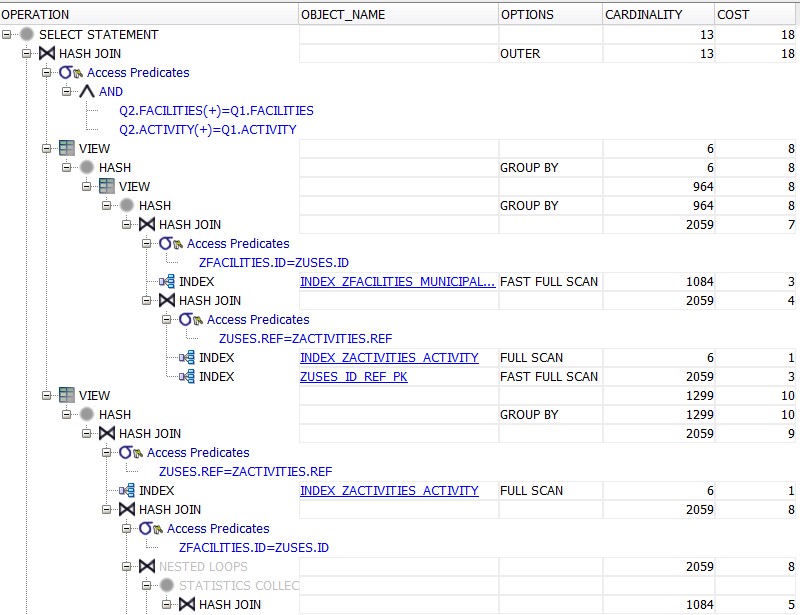
In this case, the query requested is a little bit tricky, so the solution proposed uses 3 views as support. Since 2 of the 3 views do very similar things, the cost is inevitably increased, but it’s also noticeable that with some indexes the cost of table access can be decreased, and so the total cost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

#### Query using Y tables

##### 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





##### 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



### Execution Times

|  |  |  |
| --- | --- | --- |
| **Index** | **B-TREE** | **BITMAP** |
|  | 0.026 | 0.024 |
|  | 0.024 | 0.028 |
| Time (s) | 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

##### 

##### Comment

#### Bitmap

##### 

##### Comment

## 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



### 

### Execution Times

|  |  |  |  |
| --- | --- | --- | --- |
| **Environment** | **X** | **Y** | **Z** |
|  | 0.026 | 0.022 | 0.019 |
|  | 0.034 | 0.021 | 0.020 |
| Time (s) | 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

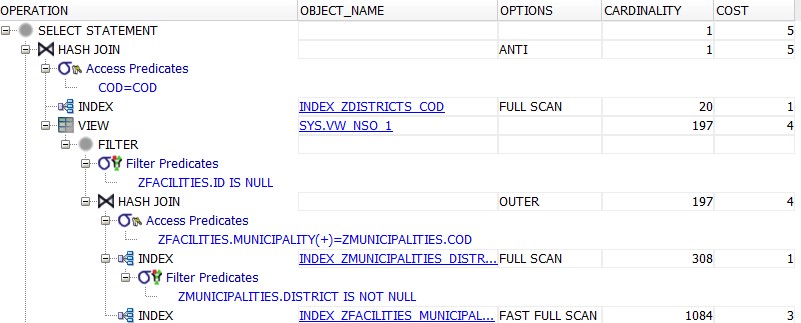
#### 

###### Comments

#### Query using Y tables

###### Comments

#### Query using Z tables



###### Comments

# 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.