

**Database Technologies**

QUERY OPTIMIZATION – TEACHING SERVICE

Integrated Master in Informatics and Computer Engineering

2019/2020

Bernardo Ramos - up201505092

Daniela João -[up201505982](mailto:up201505982@fe.up.pt)

Luís Mouta - up201808916

Patrícia Janeiro - up201605946

**Table of Contents:**

[**1.Introduction**](#_d7nu25ikvneu) **3**

[**2.Restrictions and Indexes**](#_o254dto7xno5) **4**

[2.1 Restrictions](#_9kn1yufxmk7o) 4

[2.2 Indexes](#_721p135lmmkp) 4

[**3.Questions**](#_pigao9bfrvoy) **4**

[3.1 Question 1](#_tyhtp0wkuskr) 4

[3.1.1 SQL query](#_l5sor5ro7nq5) 4

[3.1.2 Answer](#_ltuku7i9i6rm) 5

[3.1.3 Execution plan Analysis](#_snvvvbj1g7lq) 5

[X-Environment](#_5km7s1yi6y2i) 5

[Y-Environment](#_ah6kebgjat1c) 6

[Z-Environment](#_4htnens3m8bv) 6

[3.1.4 Execution Times](#_74dtau6s9tcb) 6

[3.2 Question 2](#_7l4e58nf1k5q) 7

[3.2.1 SQL query](#_uab0mlfy9asy) 7

[3.2.2 Answer](#_f88tnzs5deen) 7

[3.2.3 Execution plan Analysis](#_ksk1fmc03q0k) 7

[X-Environment](#_5j620sfz3ovb) 7

[Y-Environment](#_tyo9ujpp34c8) 8

[Z-Environment](#_7cuewz4jez9l) 8

[3.2.4 Execution Times](#_ycytoab3ht8d) 8

[3.3 Question 3](#_3xcspiuqx2vh) 9

[3.3.1 SQL query](#_nw6bd3hk63i8) 9

[a)](#_qtpimlg3w5fz) 9

[b)](#_1ucinm2pdis) 9

[3.3.2 Answer](#_r1370agj1kfz) 10

[3.3.3 Execution plan Analysis](#_mul5yzjo8mc1) 10

[X-Environment](#_m6dy3gz2fqu9) 10

[Y-Environment](#_flxawpliwy8o) 11

[Z-Environment](#_kgn40i1qif63) 11

[3.3.4 Execution Times](#_j4ghvgcf6m4z) 11

[3.4 Question 4](#_p7gmgf1zo24x) 12

[3.4.1 SQL query](#_68lzrrrrnrx) 12

[3.4.2 Answer](#_bzan7dgo4vtz) 12

[3.4.3 Execution plan Analysis](#_el0950fwxcy7) 12

[X-Environment](#_ngr2ozsjcnz) 12

[Y-Environment](#_sdz3vot8c3lu) 12

[Z-Environment](#_b140gipqfax4) 12

[3.4.4 Execution Times](#_ynfnyulxdcpn) 12

[3.5 Question 5](#_hxv3gg96wyo0) 12

[3.5.1 SQL query](#_9ld9lojoqilu) 13

[3.5.2 Answer](#_afahkmfkppff) 13

[3.5.3 Execution plan Analysis](#_26hdwfmur0ws) 13

[A) B-tree](#_gamefi4xii60) 13

[B) Bitmap](#_q6iiwaw51hx) 14

[3.6 Question 6](#_23a90qors5vz) 15

[3.6.1 SQL query](#_u35jlt7aw8wc) 16

[3.6.2 Answer](#_w9c6m9mbv26) 16

[3.6.3 Execution plan Analysis](#_j26ji4hyszsf) 16

[X-Environment](#_qpyqhdpk05wa) 16

[Y-Environment](#_9htcv9hw8qxr) 17

[Z-Environment](#_h6gnhimorc) 17

[3.6.4 Execution Times](#_167k165o2lm) 17

# 

# **1.Introduction**

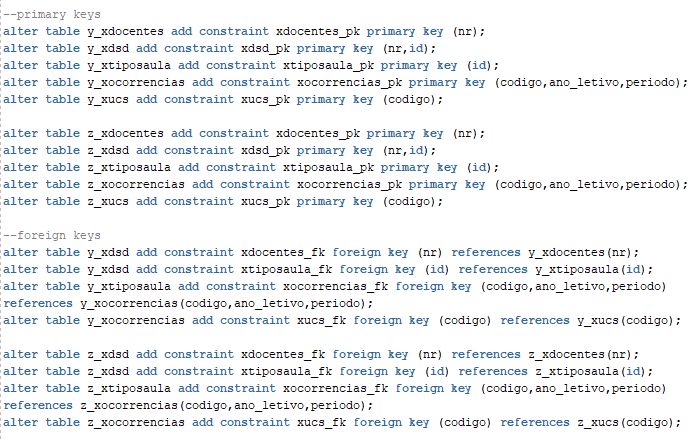
This project has been developed in the scope of the Database Technologies course unit from the Master in Informatics and Computing Engineering in FEUP. Its purpose it’s to analyse several SQL execution plans in a test database and assess the impact of indexes, statistics and the use of different strategies for query organization and optimization. In this report we will analyze 6 different question when it comes to their execution plans, estimated effort and comparison of execution times in three different environments (3 different types of tables).

Regarding the different types of tables, we had to create: X-tables, where no indexes or integrity constraints were created; Y-tables with standard integrity constraints and Z-tables, with standard integrity constraints as well and some extra indexes we might find convenient.

# **2. Restrictions and Indexes**

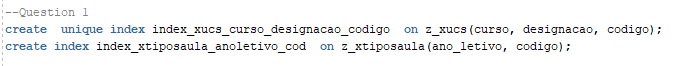
## **2.2 Restrictions**

The restriction imposed on the Y and Z tables were the ones presented on the assignment paper (primary and foreign keys). The following code represents the script that creates this restrictions.



## **2.2 Indexes**

As requested in the assignment paper for the Z\_Environment, we created the indexes presented in the following scripts to optimize the queries of each question.



**Justification:**

The first index (*index\_xucs\_curso\_designacao\_codigo*) on the *Z\_XUCS* table for the attributes *curso, designacao* and *codigo* is of major importance and it is used in several queries since most queries are about a specific *curso*. In this specific query we need to access the table *Z\_XUCS* through the attribute *codigo* and *curso* and project the attribute *designacao*. With this index we avoid the access to the table, we only need the index.

The second index (*index\_xtiposaula\_anoletivo\_cod*) on the *Z\_XTIPOSAULA* table for the attributes *ano\_letivo* and *codigo* is very important since we need to access the table through the *ano\_letivo* and *codigo* attributes.



**Justification:**

The first index follows the justification presented for the same index in question 1.

The second index (*index\_xtiposaula\_anoletivo*) on the *Z\_XTIPOSAULA* table for the attribute *ano\_letivo* is relevant since the access to the table is done through the attribute *ano\_letivo*, given that the query is done for a specific *ano\_letivo*.



**Justification:**

This index is very important in lowering the cost of the query, seeing as the JOIN clause is done in the *codigo* attribute. It also allows to access the *tipo* through the index, instead of performing a table access.

The (*index\_xucs\_curso\_designacao\_codigo*) on the *Z\_XUCS* table is also used in this query in order to access the column *curso* while avoiding a table access.

# **3.Questions**

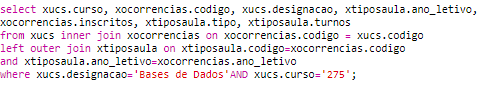
## **3.1 Question 1**

Selection and join.

Show the *código, designacao, ano\_letivo, inscritos, tipo*, and *turnos* for the course

*‘Bases de Dados’* of the program 275.

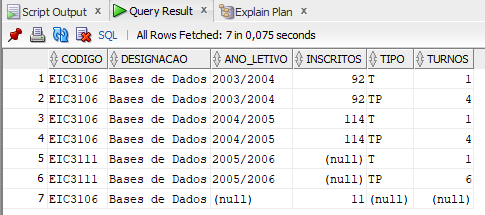
### **3.1.1 SQL query**



### 

### 

### **3.1.2 Answer**



**Fig.1: First Query’s Results**

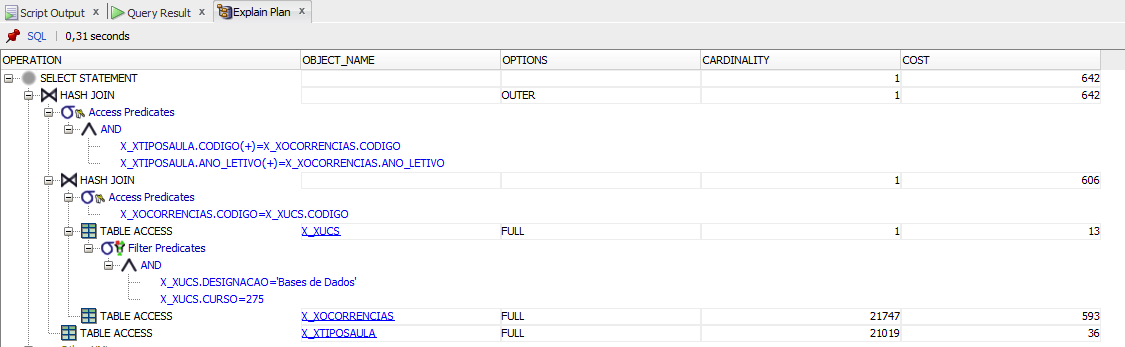
**Comments:**

After analysing the data we realized that even though there were entries on the *XOCORRENCIAS* table for the unit course EIC3106 for *ano\_lectivo*=’2002/2003’, there were no entries for that same year on the *XTIPOSAULA* table. Since there were students enrolled in the course that year, even though there was no registry of classes, the tuple of that year is presented using the keywords *OUTER JOIN*.

### 

### **3.1.3 Execution plan Analysis**

#### **X-Environment**

**Fig.2: Execution plan for query 1 on the X tables.**

**Comments:**

In this execution plan we realize that the first operation is a full search on the *X\_XUCS* table taking into account the filters presented in the *WHERE* clause. After this search, the results are the tuples that respected the previous clause ( designacao =’Bases de Dados’ and curso=275). Then, a full search is done on the *X\_XOCORRENCIAS*, only selecting the tuples in which the ‘código’ is the same as the ‘código’ from the tuples resulting from the previous step. After both searches, an hash join carries out the joinment of results. Finally, a full search on the *X\_XTIPOSAULA* is done to find out the tuples in which the *‘codigo’* and *‘ano\_letivo’* are the same as the tuples in the *X\_XOCORRENCIAS* selected in the previous selection. After this another *HASH JOIN* is done.

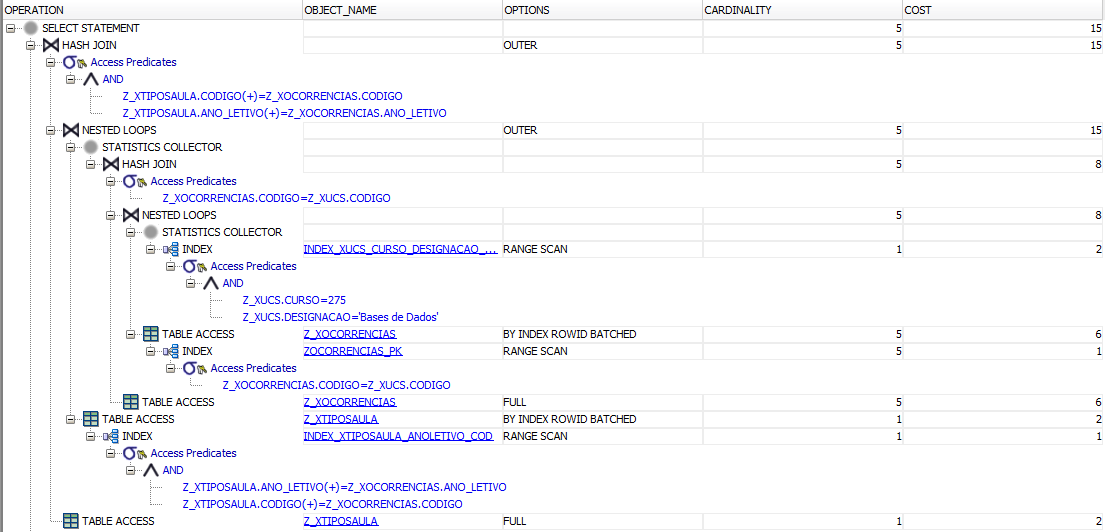
#### **Y-Environment**

**Fig.3: Execution plan for query 1 on the Y tables.**

**Comments:**

With the creation of primary and foreign keys on the Y\_Tables we immediately realize that is no longer need a full search on the *Y\_XOCORRENCIAS* table. Now, after the selection of the tuples in the *Y\_XUCS* table that were filtered through the *WHERE* clause, the tuples to be selected in the *Y\_XOCORRENCIAS* are accessed through the primary key.

#### **Z-Environment**



**Fig.4: Execution plan for query 1 on the Z tables.**

**Comments:**

After analyzing this execution plan we verify that the creation of the b-tree index on the *X\_ZUCS* table in the attributes *curso, designacao* and *codigo* - (*index\_xucs\_curso\_designacao\_codigo*), the access to the *Z\_XUCS* is avoided. With the creation of the b-tree index on the table *Z\_XTIPOSAULA* with the attribute ano\_letivo and código -index\_xtiposaula\_anoletivo\_cod, a full search on the *Z\_XTIPOSAULA* table is also avoided.

### **3.1.4 Execution Times**

|  |  |
| --- | --- |
| **Environment** | **Time (s)** |
| X | 0,097 |
| Y | 0.085 |
| Z | 0.08 |

**Tab. 1:Execution time of query 1 on tables X, Y and Z.**

## **3.2 Question 2**

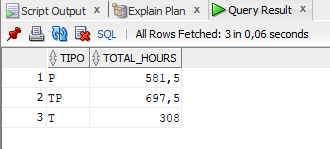
Aggregation

How many class hours of each type did the program 233 got in year 2004/2005?

### **3.2.1 SQL query**

### 

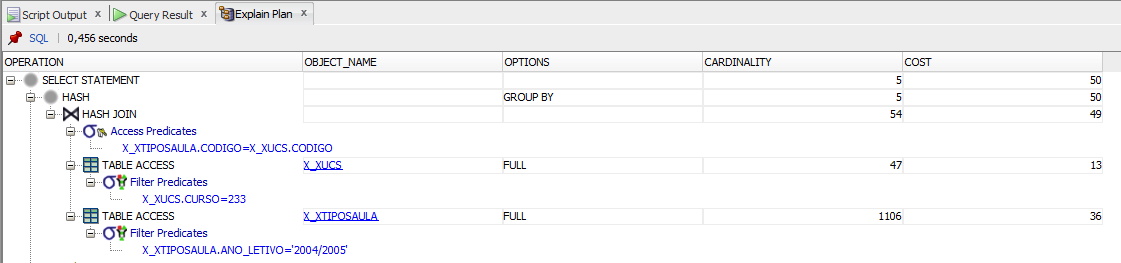
### **3.2.2 Answer**



**Fig.5: Second Query’s Results**

### **3.2.3 Execution plan Analysis**

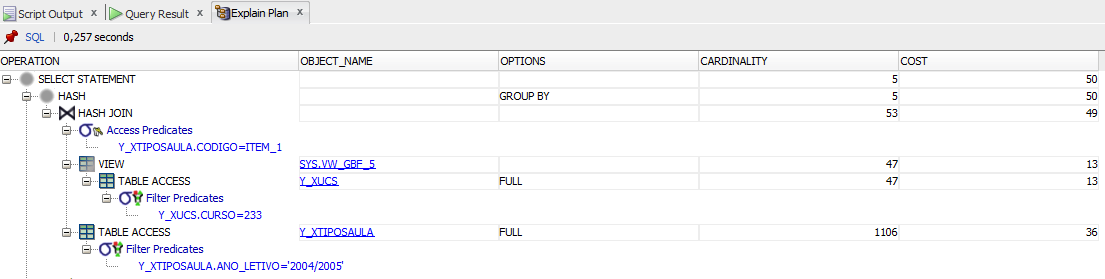
#### **X-Environment**

**Fig.6: Execution plan for query 2 on the X tables.**

**Comments:**

After analysing this execution plan, we realize a full search on the X\_XUCS and X\_XTIPOSAULA tables and the respective application of the filters (*curso* and *ano\_letivo*). After this, a joinment in done through an HASH JOIN.

#### **Y-Environment**

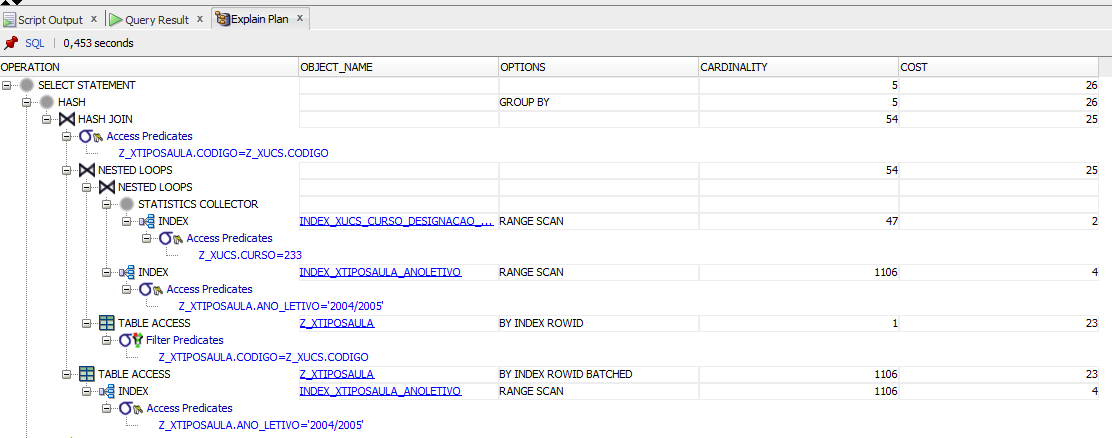


**Fig.7: Execution plan for query 2 on the Y tables.**

**Comments:**

This plan execution is similar to the one from X\_ENVIRONMENT, since the created keys in the Y-Tables aren’t related to any of the attributes used to access the tables in this query. (*Y\_XUCS.CURSO* and *Y\_XTIPOSAULA.ANO\_LETIVO*).

#### **Z-Environment**



**Fig.7: Execution plan for query 2 on the Y tables.**

**Comments:** With the creation of theb-tree index on the table X\_ZUCS in the attributes *curso, designacao* and *codigo* - (*index\_xucs\_curso\_designacao\_codigo*) and the b-tree index on the table Z\_XTIPOSAULA with the attribute *ano\_letivo* - *index\_xtiposaula\_anoletivo* a full search on both tables is avoided. We also verify a significant reduction in cost from the first two execution plans on this query to this last one. ( 50 to 26)

### **3.2.4 Execution Times**

|  |  |
| --- | --- |
| **Environment** | **Time (s)** |
| X | 0,154 |
| Y | 0,083 |
| Z | 0,076 |

**Tab. 2:Execution time of query 2 on tables X, Y and Z.**

## **3.3 Question 3**

Negation

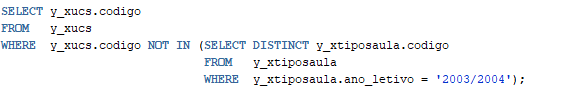
Which courses (show the code) did not have service assigned in year 2003/2004?

a. Use not in.

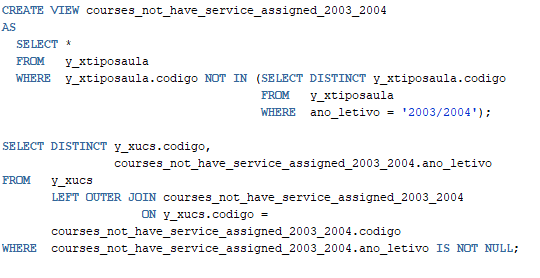
b. Use external join and is not null.

### **3.3.1 SQL query**

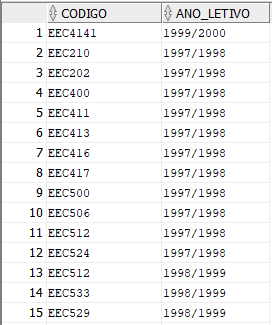
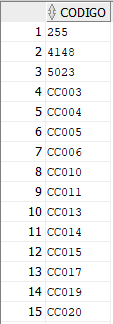
#### **a)**



#### **b)**



### **3.3.2 Answer**

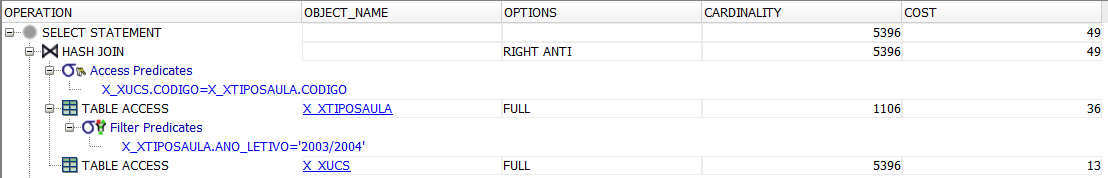


**Fig.6: a) and b) Search Query’s Results**

### **3.3.3 Execution plan Analysis**

#### **X-Environment**

* a)

****

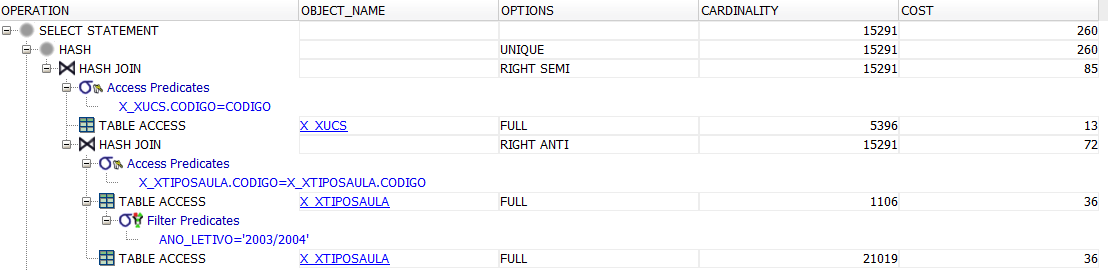
**Fig.7: Execution plan for query 3-a) on the X tables.**

**Comments:**

In this execution plan , a search is initially made for the x\_xtiposAula table , applying the ano\_letivo filter = ‘2003/2004’.

After having this initial search in memory , the x\_xucs table is scanned, selecting only codes that are not in the initial table

* b)



**Fig.8: Execution plan for query 3-b) on the X tables.**

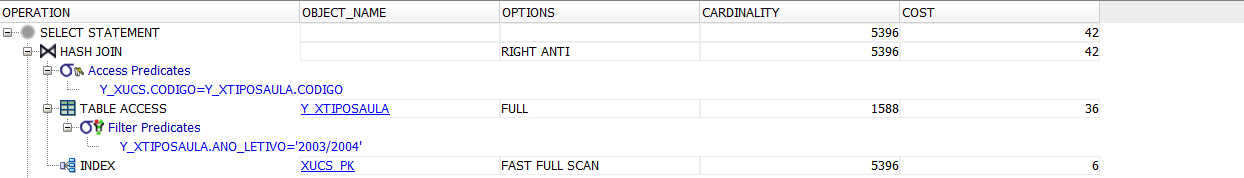
**Comments:**

Initially a complete search is made to the x\_tiposAula table applying the filter of the ano\_letivo = ‘2003/2004’.

Then a join Right Anti is done, saving the results of the x\_tiposAula table in memory and then scanning the table again looking for the codes that are not in that table in memory.

#### **Y-Environment**

* a)

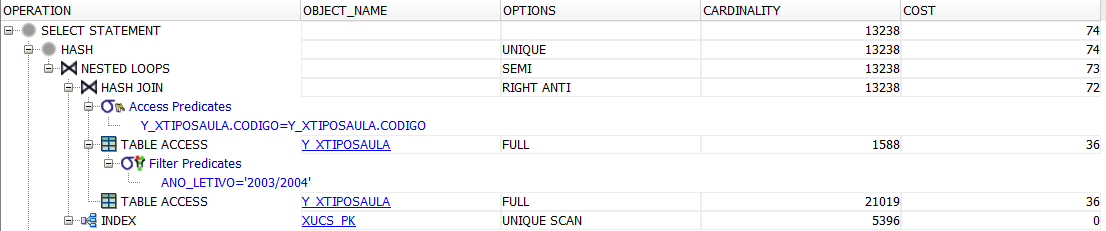
****

**Fig.9: Execution plan for query 3-a) on the Y tables.**

**Comments:**

The only visible difference is that it was not necessary to scan the table completely, making the access to the y\_ucs table no longer necessary, it was only done through the index (code).

* b)



**Fig.10: Execution plan for query 3-b) on the Y tables.**

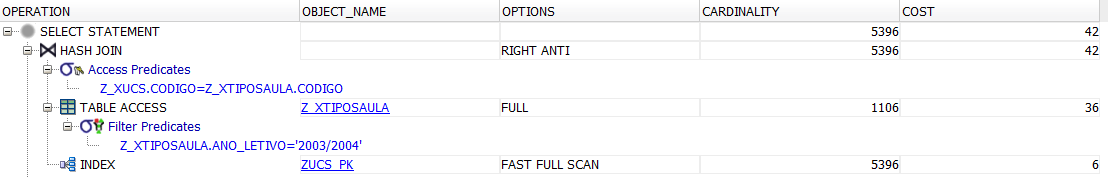
**Comments:**

The only change compared to the previous execution plan is that there is no longer a complete scan of the Y\_xucs table.

Access to the Y\_xucs table is no longer necessary and is only done by accessing the index.

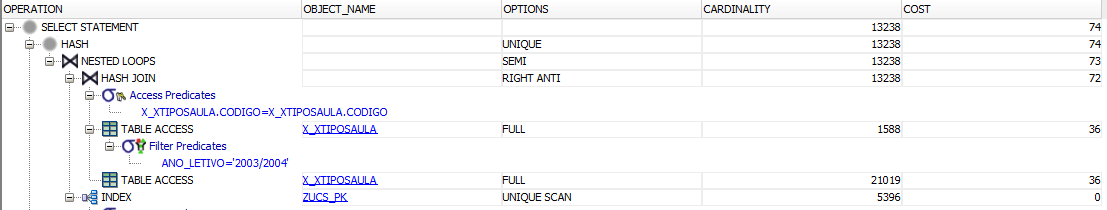
#### **Z-Environment**

* a)



**Fig.11: Execution plan for query 3-a) on the Z tables.**

* b)

****

**Fig.12: Execution plan for query 3-b) on the Z tables.**

**Comments:**

There is no impact of the chosen indexes on the cost of the queries.

### **3.3.4 Execution Times**

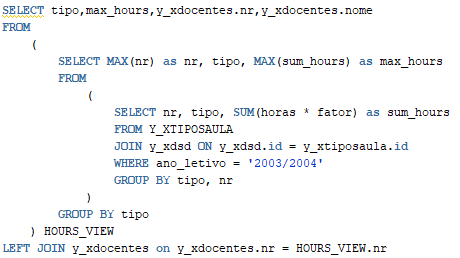
|  |  |  |
| --- | --- | --- |
| **Query 3** | **Environment** | **Time (s)** |
| Query A | X | 0.023 |
| Y | 0.022 |
| Z | 0.02 |
| Query B | X | 0.015 + 0.044 = 0.059 |
| Y | 0.011 + 0,042 = 0,053 |
| Z | 0.010 + 0.040 = 0.050 |

**Tab. 3:Execution time of query 3 on tables X, Y and Z**

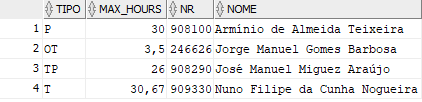
## **3.4 Question 4**

Who is the professor with more class hours for each type of class, in the academic year 2003/2004? Show the number and name of the professor, the type of class and the total of class hours times the factor.

### **3.4.1 SQL query**



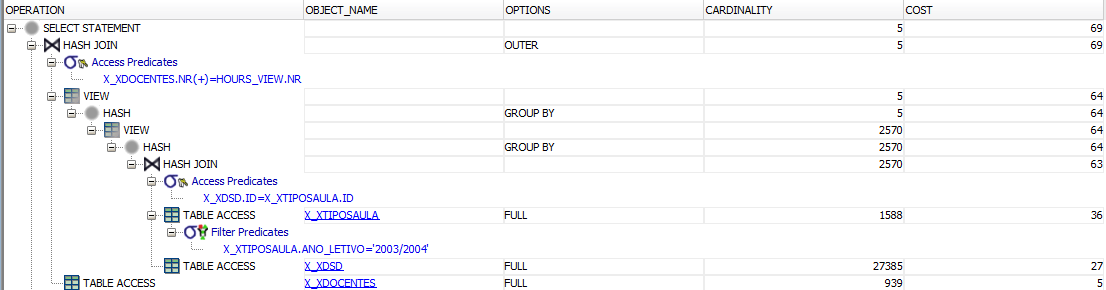
### **3.4.2 Answer**



**Fig.13: Fourth Query Result**

### **3.4.3 Execution plan Analysis**

#### **X-Environment**

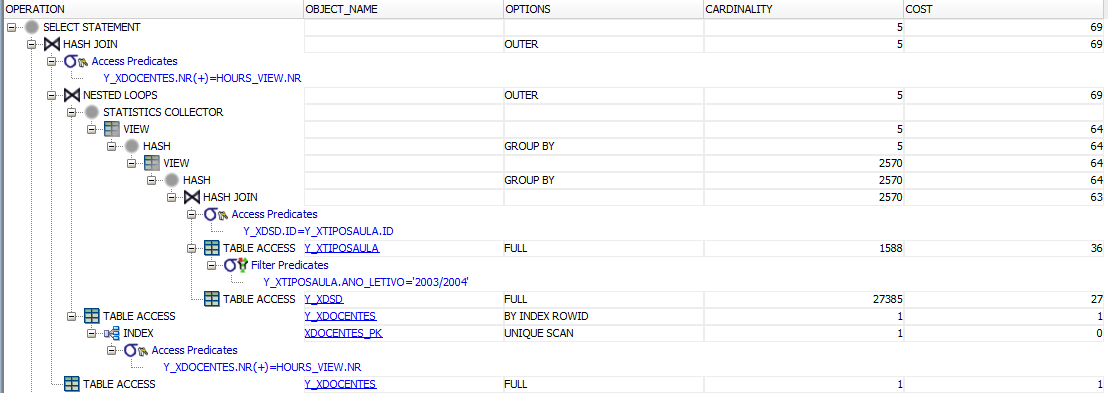


**Fig.14: Execution plan for query 4 on the X tables.**

**Comments:**

In this query first an access to the table X\_TIPOSAULA joined with X\_XDSD and X\_XDOCENTES is performed, then it is filtered by ANO\_LETIVO=’2003/2004’ , following a view is created with the result of the sum of the lectures hours grouped by the class type which then we select the max sum of hours of each type of class.

#### **Y-Environment**

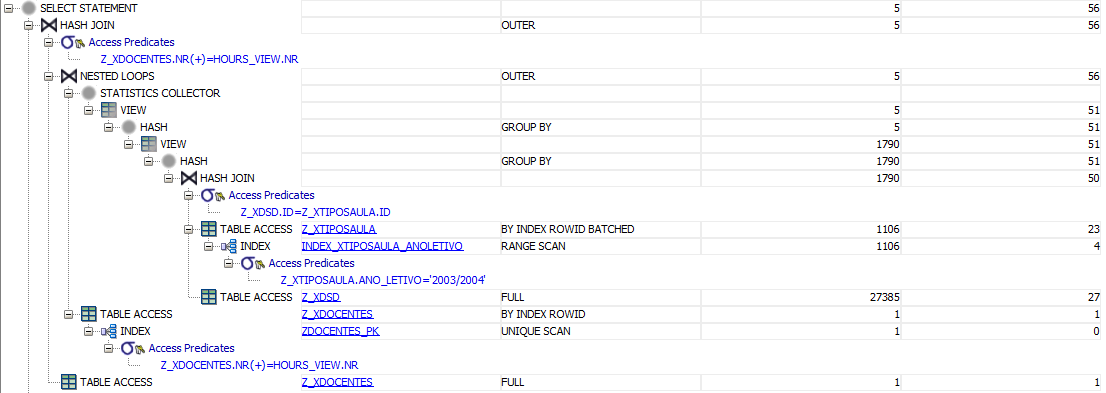


**Fig.15: Execution plan for query 4 on the Y tables.**

**Comments:**

Both the X and Y environment present the same total cost (69) being that the query itself is already optimized.

#### **Z-Environment**



**Fig.16: Execution plan for query 4 on the Z tables.**

**Comments:**

On the Z environment there was a reduction in cost to 56, this was possible by the use of the index present on XTIPOSAULA\_ANOLETIVO which reduced the cost of the first access on Z\_XTIPOSAULA from 36 to 27.

### **3.4.4 Execution Times**

|  |  |  |
| --- | --- | --- |
| **Query 4** | **Environment** | **Time (s)** |
| Query A | X | 0,055 |
| Y | 0,036 |
| Z | 0,029 |

**Tab. 4:Execution time of query 4 on tables X, Y and Z**

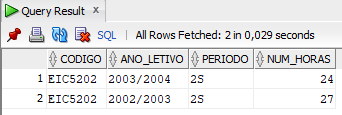
## **3.5 Question 5**

Compare the execution plans and the index sizes for the query giving the course code, the academic year, the period, and number of hours of the type ‘OT’ in the academic years of 2002/2003 and 2003/2004.

### **3.5.1 SQL query**

### 

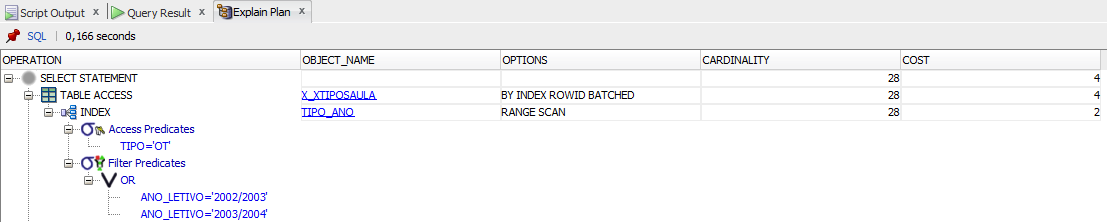
### **3.5.2 Answer**



**Fig.17 : 5th Query Result**

### **3.5.3 Execution plan Analysis**

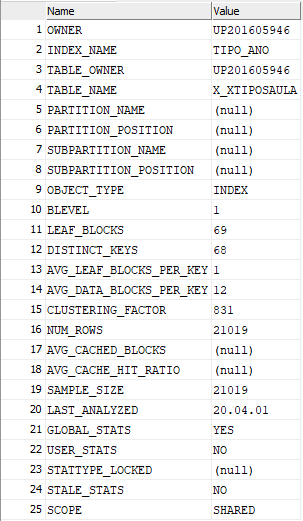
#### **A) B-tree**



**Fig.18: Execution plan for query 5 a) (b-tree)**

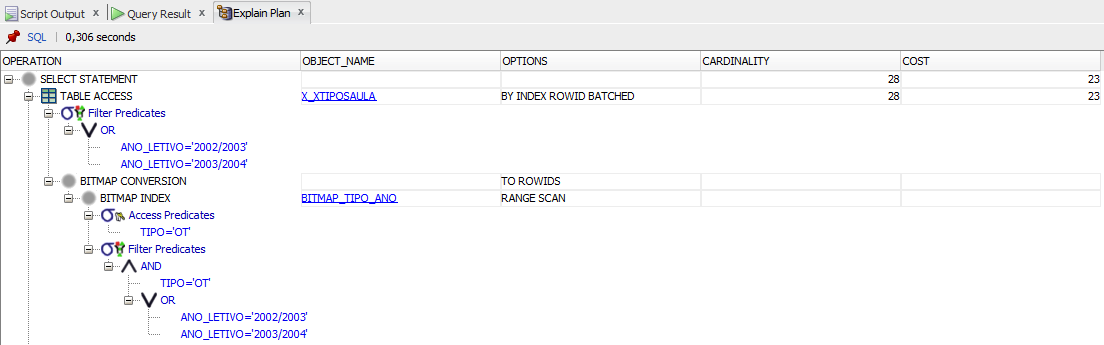
**Comments:**

As we can see, the access to the *tipo* and *ano\_letivo* attributes, used in the WHERE clause, is done using the index *(tipo\_ano)*. There is still the need to perform a table access by index rowid because some of the attributes fetched are not part of the created B-tree index.



**Fig.19: Index statistics for Query 5 a) (b-tree)**

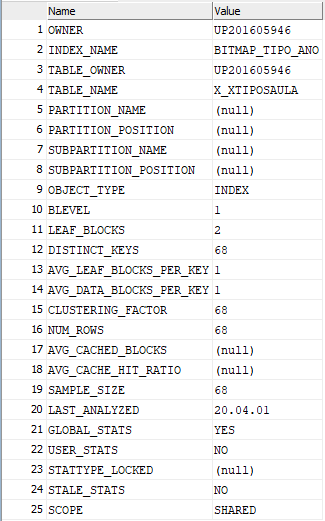
#### **B) Bitmap**



**Fig.20: Execution plan for query 5 b) (bitmap)**

**Comments:**

The way of accessing the attributes is similar to the one on the execution with the B-tree index, using an index access and a table access by index rowid, however, the query has a higher cost when executed with a bitmap index.



**Fig.21: Index statistics for query 5 b) (bitmap)**

### **3.5.4 Execution Times**

|  |  |
| --- | --- |
| **Environment** | **Time (s)** |
| A | 0.034 |
| B | 0.026 |

**Tab.5 :Execution time of query 5 with indexes a) and b).**

## **3.6 Question 6**

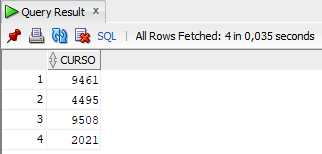
Select the programs (curso) that have classes with all the types.

### **3.6.1 SQL query**

### 

**Fig.22: Question 6 Sql Query**

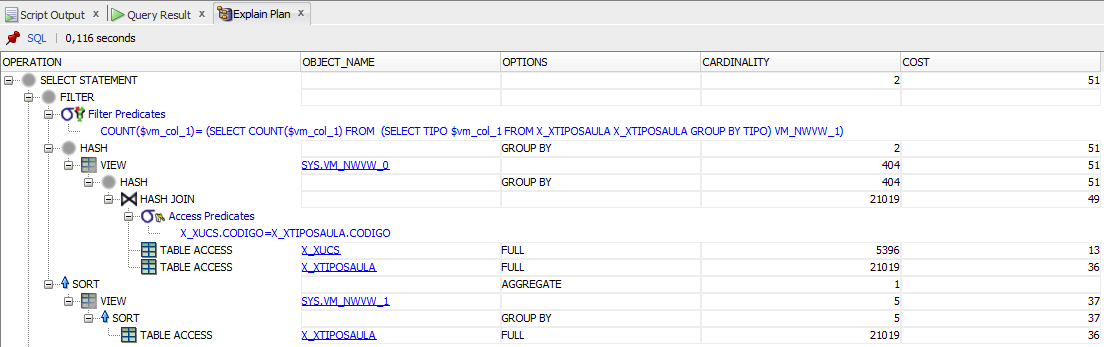
### **3.6.2 Answer**



**Fig.23 : Question 6 SQL Query Results**

### **3.6.3 Execution plan Analysis**

#### **X-Environment**



**Fig.24: Execution plan for query 6 on the X tables.**

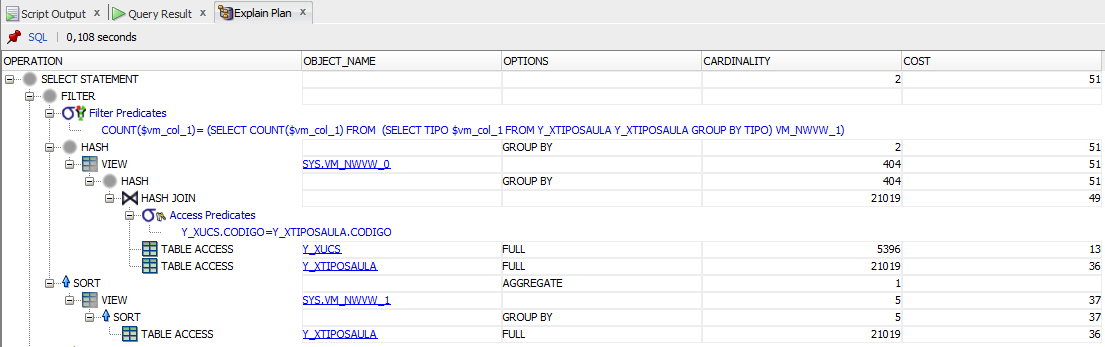
**Comments:**

In this execution plan, we can observe that the first actions performed are two full table accesses on the tables X\_XUCS and X\_XTIPOSAULA, in order to perform the JOIN operation between them. This is followed by a GROUP BY expression on the *curso* and *tipo* attributes, the result of this operation is stored on a view (SYS.VM\_NWWW\_0).

There is then a new table access to the X\_XTIPOSAULA table followed by a GROUP BY expression on the *tipo* attribute, which corresponds to the query that gives us the number of distinct types that exist on the column *tipo*. This is also stored in a view (SYS.VM\_NWWW\_1).

In the end, the results are filtered according to the HAVING clause.

#### **Y-Environment**

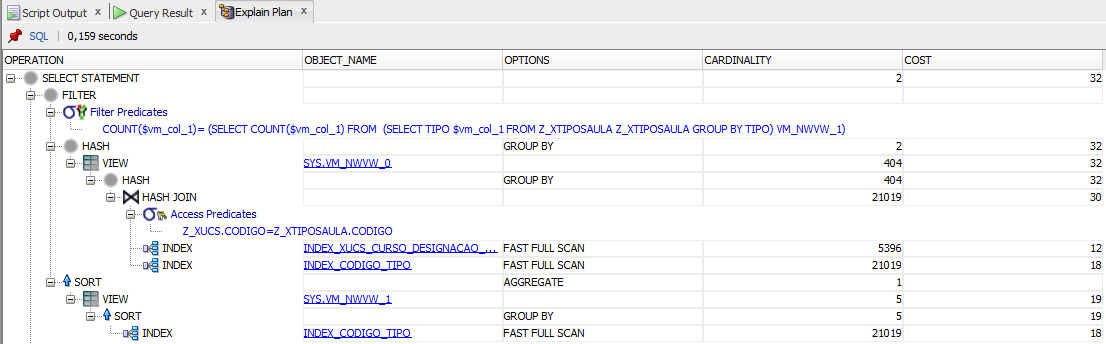


**Fig.25: Execution plan for query 6 on the Y tables.**

**Comments:**

As we can see, the execution plan on the Y tables is the same as the one we obtained on the X tables, therefore there are no differences in cost or performance when we use only the primary and foreign key constraints.

#### **Z-Environment**



**Fig.26: Execution plan for query 6 on the Z tables.**

**Comments:**

The execution in the Z environment is similar to the ones in the X and Y environments, except for the fact that the accesses to the attributes are made through the indexes. Using the *(index\_codigo\_tipo)* and *(index\_xucs\_curso\_designacao\_codigo)* indexes we avoid performing three table accesses and reduce the cost of executing the query from 51 to 32.

### **3.6.4 Execution Times**

|  |  |
| --- | --- |
| **Environment** | **Time (s)** |
| X | 0,0534 |
| Y | 0,0492 |
| Z | 0,0434 |

**Tab. 6:Execution time of query 6 on tables X, Y and Z.**