Faculty of Engineering of the University of Porto

Project 1

Query Optimization

M.EIC028 - Database Technologies (TBD)

1MEIC02

Professor

Mariana Curado Malta

Student

Diogo Alexandre da Costa Melo Moreira da Fonte - up20204175@edu.fe.up.pt

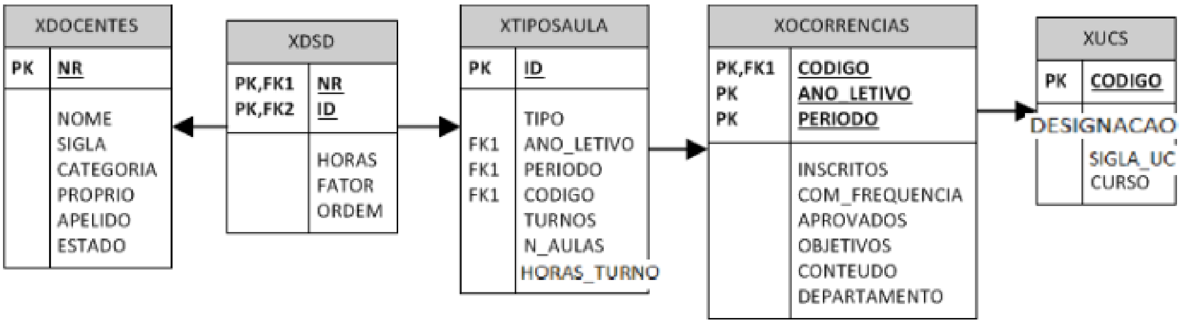
Nuno André Gomes França - up201807530@edu.fe.up.pt

1. Introduction

The purpose of this homework is to analyze several SQL execution plans in a test database. We also need to assess the impact of indexes comparing the statistics of the executions of different strategies for query organization and executed in three different experimentation environments.

The database model we are working with stores data about the distribution of teaching services in a faculty. There are:

- courses (table XUCS);
- occurrences of courses (table XOCORRENCIAS);
- types of classes that belong to occurrences (table XTIPOSAULA) and each occurrence may have one or more class types;
- teaching service distributions (table XDSD) that records, for each class type of occurrence, how many weekly hours are assigned to that professor. If a professor is teaching, in a single class, more than one course at the same time, for example, from different programs, the weight of that course may be less than 1 (attribute factor);
- list of professors (table XDOCENTES).



(figure 1 - Relation Model)

2. Experimentation Environments

We were asked to implement three different experimentation environments, to be able to compare the execution of different queries, analyzing the impact of the query organization and the influence of adding integrity constraints and indexes in our database tables. Below, we explain in more detail each experimentation environment.

2.1. Environment X

In this environment we only created the sql tables, without adding any index or integrity constraint. We created the tables copying the original tables provided.

Here it is the SQL script to create this environment X:

```sql
CREATE TABLE xdocentes AS SELECT * FROM GTD10.docentes;

CREATE TABLE xdsd AS SELECT * FROM GTD10.dsd;

CREATE TABLE xtiposaula AS SELECT * FROM GTD10.tiposaula;

CREATE TABLE xocorrencias AS SELECT * FROM GTD10.ocorrencias;

CREATE TABLE xucs AS SELECT * FROM GTD10.ucs;
```

2.2. Environment Y

In the environment Y, we added some integrity constraints, primary keys and foreign keys. The constraints are also represented in the relational model above, in the figure 1. We implemented some basic primary and foreign keys, but also some compound keys.

The added primary keys, either compound or not, can uniquely identify each record/row in each table, and the foreign keys are used to represent the relations between the tables, mentioned in the introduction section.

Here it is the SQL script to create this environment Y:

```sql
CREATE TABLE ydocentes AS SELECT * FROM GTD10.docentes;
ALTER TABLE ydocentes ADD PRIMARY KEY ("NR");

CREATE TABLE yucs AS SELECT * FROM GTD10.ucs;
ALTER TABLE yucs ADD PRIMARY KEY ("CODIGO");

CREATE TABLE yocorrencias AS SELECT * FROM GTD10.ocorrencias;
ALTER TABLE yocorrencias ADD CONSTRAINT YOCORRENCIAS_PK PRIMARY
KEY("CODIGO","ANO_LETIVO", "PERIODO");
ALTER TABLE yocorrencias ADD FOREIGN KEY ("CODIGO") REFERENCES
yucs("CODIGO");

CREATE TABLE ytiposaula AS SELECT * FROM GTD10.tiposaula;
ALTER TABLE ytiposaula ADD PRIMARY KEY ("ID");
ALTER TABLE ytiposaula ADD FOREIGN KEY ("CODIGO", "ANO_LETIVO", "PERIODO")
REFERENCES yocorrencias("CODIGO", "ANO_LETIVO", "PERIODO");

CREATE TABLE ydsd AS SELECT * FROM GTD10.dsd;
ALTER TABLE ydsd ADD CONSTRAINT YDSD_PK PRIMARY KEY("NR","ID");
ALTER TABLE ydsd ADD FOREIGN KEY ("NR") REFERENCES ydocentes("NR");
ALTER TABLE ydsd ADD FOREIGN KEY ("ID") REFERENCES ytiposaula("ID");
```

2.3. Environment Z

Finally, in our environment Z, in addition to adding the integrity constraints mentioned above, in the environment Y description, we added some indexes. Our choice of indexes was based on the analysis of the questions presented in the assignment and we created indexes in the fields that would be most useful to make our queries more efficient. Only one exception, in the table ZTIPOSAULA we tested two types of indexes in the fields "TIPO" and "ANO_LETIVO", as requested in question number five of the assignment.

Here it is the SQL script to create this environment Z:

```sql
CREATE TABLE zdocentes AS SELECT * FROM GTD10.docentes;
ALTER TABLE zdocentes ADD PRIMARY KEY ("NR");
CREATE INDEX IDX_NOME_DOCENTE ON zdocentes("NOME");

CREATE TABLE zucs AS SELECT * FROM GTD10.ucs;
ALTER TABLE zucs ADD PRIMARY KEY ("CODIGO");
CREATE INDEX IDX_DESIGNACAO_UC ON zucs("DESIGNACAO");
CREATE INDEX IDX_CURSO_UC ON zucs("CURSO");

CREATE TABLE zocorrencias AS SELECT * FROM GTD10.ocorrencias;
ALTER TABLE zocorrencias ADD CONSTRAINT ZOCORRENCIAS_PK PRIMARY
KEY("CODIGO","ANO_LETIVO", "PERIODO");
ALTER TABLE zocorrencias ADD FOREIGN KEY ("CODIGO") REFERENCES
zucs("CODIGO");
CREATE INDEX IDX_DEPARTAMENTO_OCORRENCIA ON zocorrencias("DEPARTAMENTO");

CREATE TABLE ztiposaula AS SELECT * FROM GTD10.tiposaula;
ALTER TABLE ztiposaula ADD PRIMARY KEY ("ID");
ALTER TABLE ztiposaula ADD FOREIGN KEY ("CODIGO", "ANO_LETIVO", "PERIODO")
REFERENCES zocorrencias("CODIGO", "ANO_LETIVO", "PERIODO");
CREATE INDEX BTREE_IDX_TIPO_AND_ANO_LETIVO_TIPOSAULA ON ztiposaula("TIPO",
"ANO_LETIVO"); --BTREE
--CREATE BITMAP INDEX BITMAP_IDX_TIPO_AND_ANO_LETIVO_TIPOSAULA ON
ztiposaula("TIPO", "ANO_LETIVO"); --BITMAP

CREATE TABLE zdsd AS SELECT * FROM GTD10.dsd;
ALTER TABLE zdsd ADD CONSTRAINT ZDSD_PK PRIMARY KEY("NR","ID");
ALTER TABLE zdsd ADD FOREIGN KEY ("NR") REFERENCES zdocentes("NR");
ALTER TABLE zdsd ADD FOREIGN KEY ("ID") REFERENCES ztiposaula("ID");
```

3. Questions Analysis

In this section, we are going to approach each question of the assignment, analyzing our queries, the retrieved results and the comparison between the executions in the several environments.

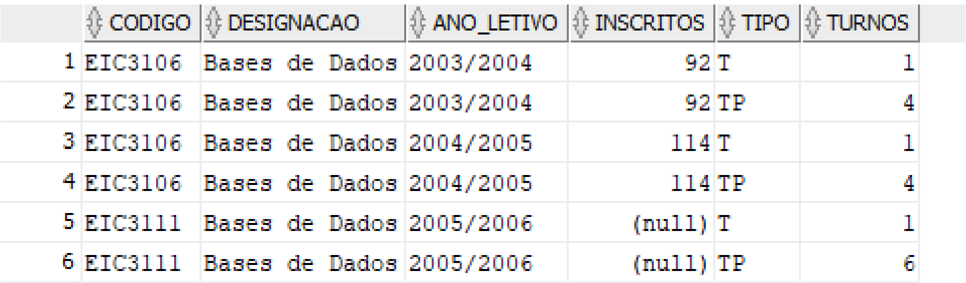
3.1. Question 1

Description: Selection and Join - Show the codigo, designacao, ano_letivo, inscritos, tipo, and turnos for the course 'Bases de Dados' of the program 275.

SQL Query:

```
SELECT ucs.codigo, ucs.designacao, ocorrencias.ano_letivo,
ocorrencias.inscritos, tiposaula.tipo, tiposaula.turnos
FROM tiposaula
LEFT JOIN ocorrencias ON tiposaula.codigo = ocorrencias.codigo AND
tiposaula.ano_letivo = ocorrencias.ano_letivo AND tiposaula.periodo =
ocorrencias.periodo
LEFT JOIN ucs ON ocorrencias.codigo = ucs.codigo
WHERE ucs.curso = 275 AND ucs.designacao = 'Bases de Dados'
```

Retrieved Results:



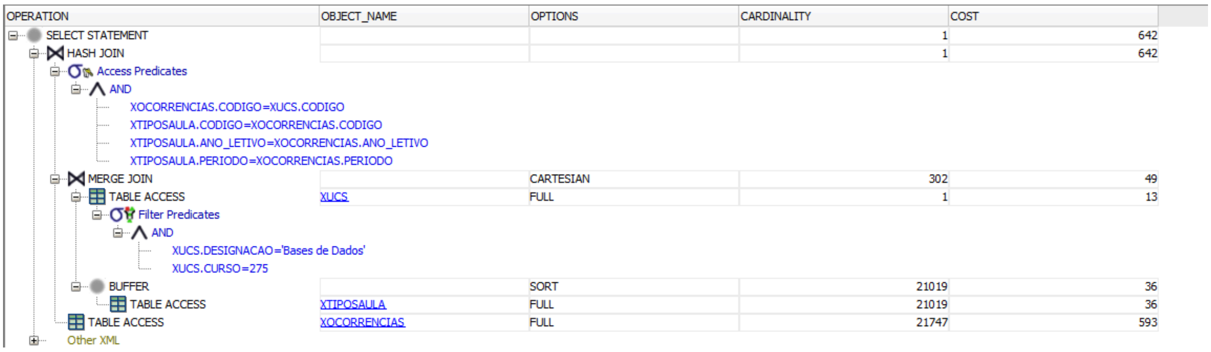
	CODIGO	DESIGNACAO	ANO_LETIVO	INSCRITOS	TIPO	TURNOS
1	EIC3106	Bases de Dados	2003/2004	92	T	1
2	EIC3106	Bases de Dados	2003/2004	92	TP	4
3	EIC3106	Bases de Dados	2004/2005	114	T	1
4	EIC3106	Bases de Dados	2004/2005	114	TP	4
5	EIC3111	Bases de Dados	2005/2006	(null)	T	1
6	EIC3111	Bases de Dados	2005/2006	(null)	TP	6

(figure 2 - Retrieved Results for Question 1)

Execution Data:

Environment X:

Query Execution Time: 0,032 seconds

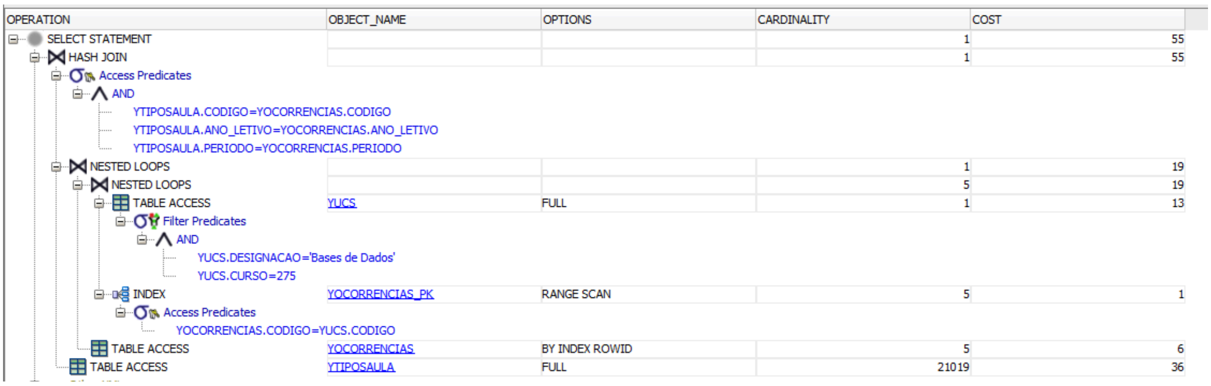


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	642
HASH JOIN			1	642
Access Predicates				
AND				
XOCORRENCIAS.CODIGO=XUCS.CODIGO				
XTIPOSAULA.CODIGO=XOCORRENCIAS.CODIGO				
XTIPOSAULA.ANO_LETIVO=XOCORRENCIAS.ANO_LETIVO				
XTIPOSAULA.PERIODO=XOCORRENCIAS.PERIODO				
MERGE JOIN		CARTESIAN	302	49
TABLE ACCESS	XUCS	FULL	1	13
Filter Predicates				
AND				
XUCS.DESIGNACAO='Bases de Dados'				
XUCS.CURSO=275				
BUFFER		SORT	21019	36
TABLE ACCESS	XTIPOSAULA	FULL	21019	36
TABLE ACCESS	XOCORRENCIAS	FULL	21747	593
Other XML				

(figure 3 - Explain Plan of Environment X for Question 1)

Environment Y:

Query Execution Time: 0,031 seconds

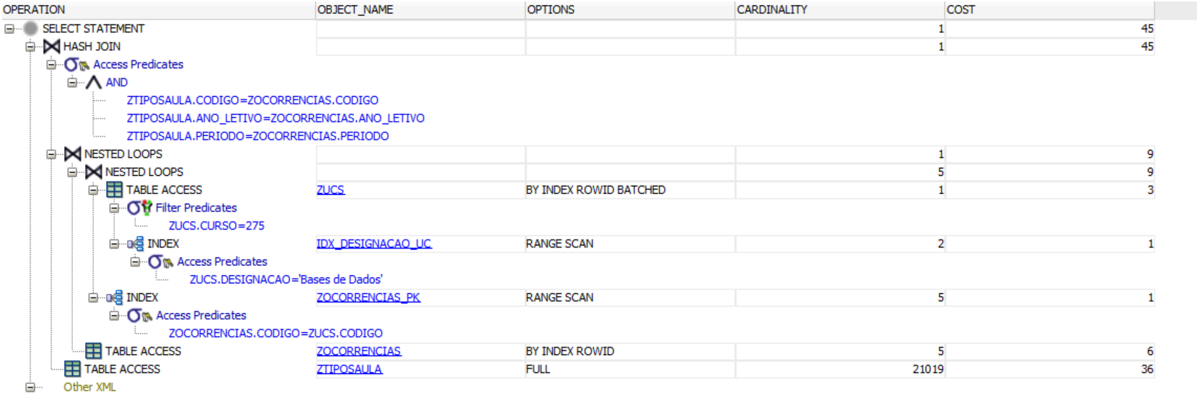


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	55
HASH JOIN			1	55
Access Predicates				
AND				
YTIPOSAULA.CODIGO=YOCORRENCIAS.CODIGO				
YTIPOSAULA.ANO_LETIVO=YOCORRENCIAS.ANO_LETIVO				
YTIPOSAULA.PERIODO=YOCORRENCIAS.PERIODO				
NESTED LOOPS			1	19
NESTED LOOPS			5	19
TABLE ACCESS	YUCS	FULL	1	13
Filter Predicates				
AND				
YUCS.DESIGNACAO='Bases de Dados'				
YUCS.CURSO=275				
INDEX	YOCORRENCIAS_PK	RANGE SCAN	5	1
Access Predicates				
YOCORRENCIAS.CODIGO=YUCS.CODIGO				
TABLE ACCESS	YOCORRENCIAS	BY INDEX ROWID	5	6
TABLE ACCESS	YTIPOSAULA	FULL	21019	36

(figure 4 - Explain Plan of Environment Y for Question 1)

Environment Z:

Query Execution Time: 0,041 seconds



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	45
HASH JOIN			1	45
Access Predicates				
AND				
ZTIPOSAULA.CODIGO=ZOCORRENCIAS.CODIGO				
ZTIPOSAULA.ANO_LETIVO=ZOCORRENCIAS.ANO_LETIVO				
ZTIPOSAULA.PERIODO=ZOCORRENCIAS.PERIODO				
NESTED LOOPS			1	9
NESTED LOOPS			5	9
TABLE ACCESS	ZUCS	BY INDEX ROWID BATCHED	1	3
Filter Predicates				
ZUCS.CURSO=275				
INDEX	IDX_DESIGNACAO_UC	RANGE SCAN	2	1
Access Predicates				
ZUCS.DESIGNACAO='Bases de Dados'				
INDEX	ZOCORRENCIAS_PK	RANGE SCAN	5	1
Access Predicates				
ZOCORRENCIAS.CODIGO=ZUCS.CODIGO				
TABLE ACCESS	ZOCORRENCIAS	BY INDEX ROWID	5	6
TABLE ACCESS	ZTIPOSAULA	FULL	21019	36
Other XML				

(figure 5 - Explain Plan of Environment Z for Question 1)

Comparison:

Analyzing the three environments explain plans, we can conclude that the addition of integrity constraints (primary keys (PK) and foreign keys (FK)) is essential for query performance. We can see that in the first environment (X) it needed a *Merge Join*, and in the second and third Nested Loops were used to link the rows using the propers PK and FK.

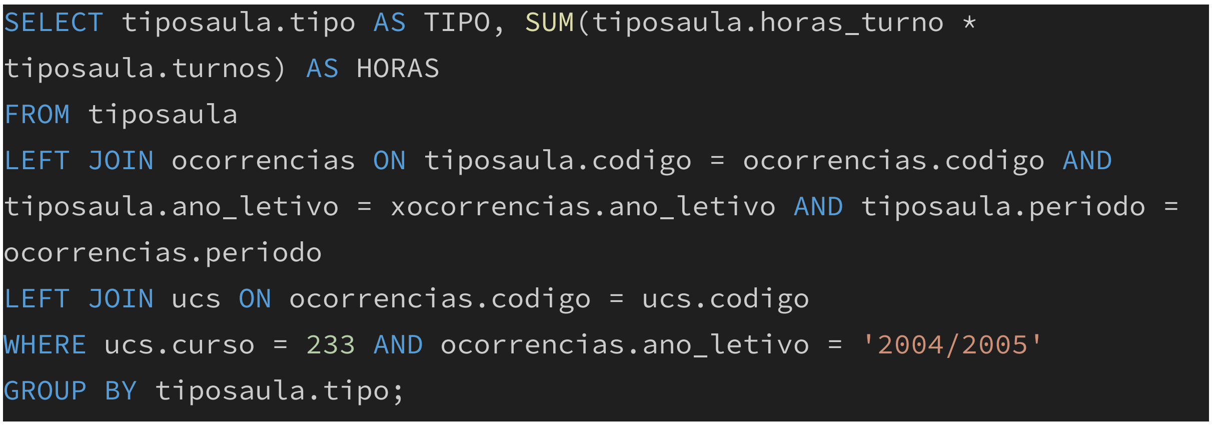
The indexes didn't make a lot of difference, but were useful to select and join the wanted record/row, and improved a bit the cost of the query.

About the time execution, it was not very conclusive because the execution times collected were a lot similar. We noticed that it has grown with the addition of integrity constraints and with the indexes, but we think that it is a good trade between query execution time versus query execution cost.

3.2. Question 2

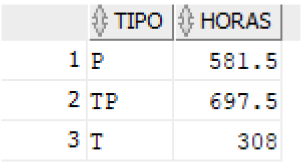
Description: Aggregation - How many class hours of each type did the program 233 planned in 2004/2005?

SQL Query:



```
SELECT tiposaula.tipo AS TIPO, SUM(tiposaula.horas_turno *
tiposaula.turnos) AS HORAS
FROM tiposaula
LEFT JOIN ocorrencias ON tiposaula.codigo = ocorrencias.codigo AND
tiposaula.ano_letivo = xocorrencias.ano_letivo AND tiposaula.periodo =
ocorrencias.periodo
LEFT JOIN ucs ON ocorrencias.codigo = ucs.codigo
WHERE ucs.curso = 233 AND ocorrencias.ano_letivo = '2004/2005'
GROUP BY tiposaula.tipo;
```

Retrieved Results:



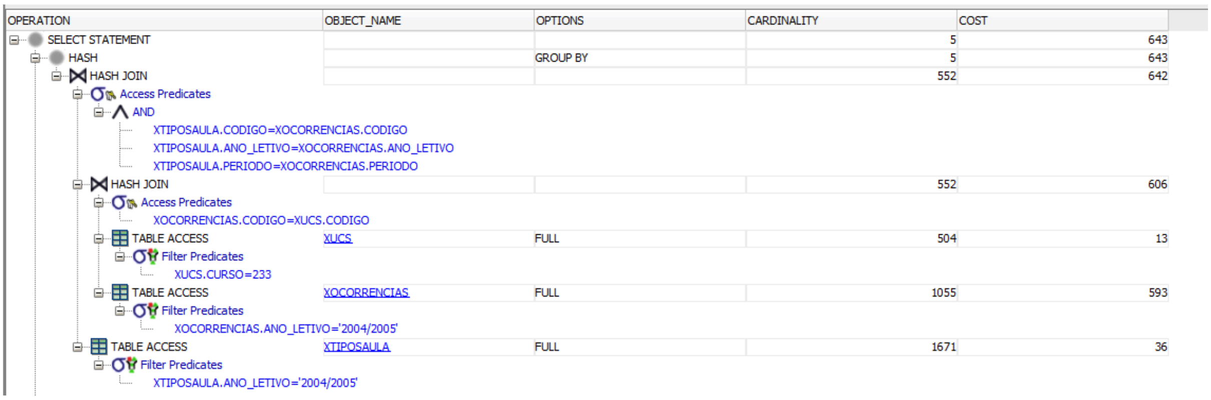
	TIPO	HORAS
1	P	581.5
2	TP	697.5
3	T	308

(figure 6 - Retrieved Results for Question 2)

Execution Data:

Environment X:

Query Execution Time: 0,039 seconds

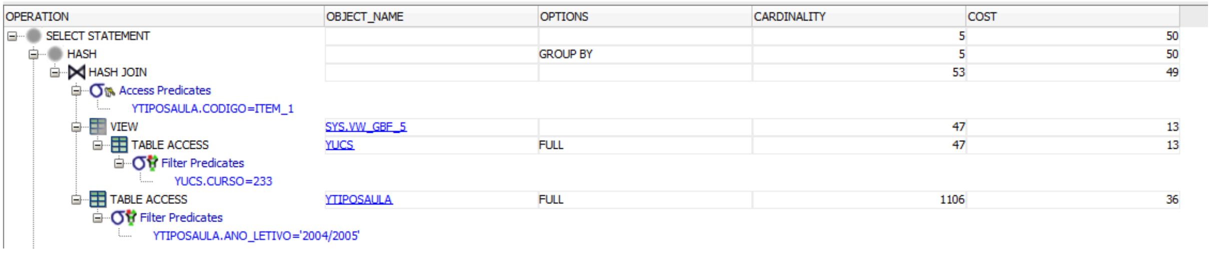


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	643
HASH		GROUP BY	5	643
HASH JOIN			552	642
Access Predicates				
AND				
XTIPOSAULA.CODIGO=XOCORRENCIAS.CODIGO				
XTIPOSAULA.ANO_LETIVO=XOCORRENCIAS.ANO_LETIVO				
XTIPOSAULA.PERIODO=XOCORRENCIAS.PERIODO				
HASH JOIN			552	606
Access Predicates				
XOCORRENCIAS.CODIGO=XUCS.CODIGO				
TABLE ACCESS	XUCS	FULL	504	13
Filter Predicates				
XUCS.CURSO=233				
TABLE ACCESS	XOCORRENCIAS	FULL	1055	593
Filter Predicates				
XOCORRENCIAS.ANO_LETIVO='2004/2005'				
TABLE ACCESS	XTIPOSAULA	FULL	1671	36
Filter Predicates				
XTIPOSAULA.ANO_LETIVO='2004/2005'				

(figure 7 - Explain Plan of Environment X for Question 2)

Environment Y:

Query Execution Time: 0,032 seconds

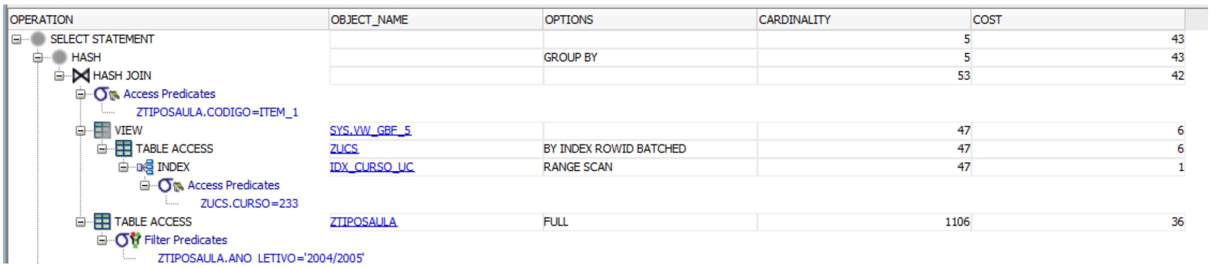


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	50
HASH		GROUP BY	5	50
HASH JOIN			53	49
Access Predicates				
YTIPOSAULA.CODIGO=ITEM_1				
VIEW	SYS.VW_GBF_5		47	13
TABLE ACCESS	YUCS	FULL	47	13
Filter Predicates				
YUCS.CURSO=233				
TABLE ACCESS	YTIPOSAULA	FULL	1106	36
Filter Predicates				
YTIPOSAULA.ANO_LETIVO='2004/2005'				

(figure 8 - Explain Plan of Environment Y for Question 2)

Environment Z:

Query Execution Time: 0,029 seconds



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	43
HASH		GROUP BY	5	43
HASH JOIN			53	42
Access Predicates				
ZTIPOSAULA.CODIGO=ITEM_1				
VIEW	SYS.VW_GBF_5		47	6
TABLE ACCESS	ZUCS	BY INDEX ROWID BATCHED	47	6
INDEX	IDX_CURSO_UC	RANGE SCAN	47	1
Access Predicates				
ZUCS.CURSO=233				
TABLE ACCESS	ZTIPOSAULA	FULL	1106	36
Filter Predicates				
ZTIPOSAULA.ANO LETIVO='2004/2005'				

(figure 9 - Explain Plan of Environment Z for Question 2)

Comparison:

Comparing the three environments explain plans, we noticed that the implementation of integrity constraints (PK and FK) is crucial for query performance. We can see that in the first environment (X) it needed two Hash Joins, and in the second and third only one. With the proper integrity constraints we can see that in the environments Y and Z it wasn't needed to pass through the table occurrences. It was possible to link directly the table ucs through the attribute codigo in the table tiposaula.

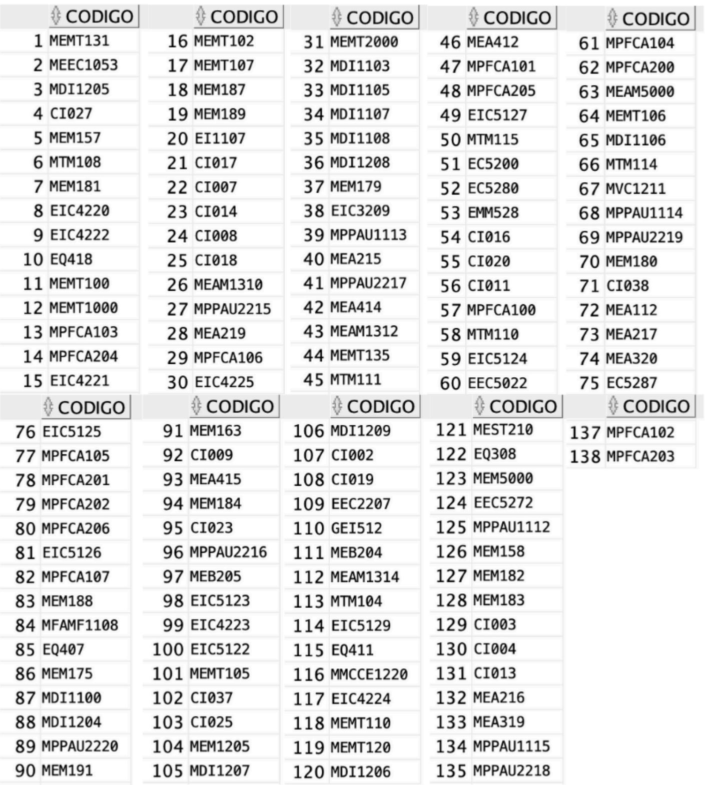
The indexes didn’t make a lot of difference, but were useful to select and join the wanted record/row, and improved a bit the cost of the query.

In this question, the execution times were not very conclusive because the times collected were a lot similar. Despite that, we noticed that it has decreased with the addition of integrity constraints and with the indexes, but we think that it is a good trade between query execution time versus query execution cost.

3.3. Question 3

Description: Negation - Which courses (show the code) did have occurrences planned but did not get service assigned in the year 2003/2004?

Retrieved Results for both approaches:



	CODIGO
1	MEMT131
2	MEEC1053
3	MDI1205
4	CI027
5	MEM157
6	MTM108
7	MEM181
8	EIC4220
9	EIC4222
10	EQ418
11	MEMT100
12	MEMT1000
13	MPFCA103
14	MPFCA204
15	EIC4221
16	MEMT102
17	MEMT107
18	MEM187
19	MEM189
20	EI1107
21	CI017
22	CI007
23	CI014
24	CI008
25	CI018
26	MEAM1310
27	MPPAU2215
28	MEA219
29	MPFCA106
30	EIC4225
31	MEMT2000
32	MDI1103
33	MDI1105
34	MDI1107
35	MDI1108
36	MDI1208
37	MEM179
38	EIC3209
39	MPPAU1113
40	MEA215
41	MPPAU2217
42	MEA414
43	MEAM1312
44	MEMT135
45	MTM111
46	MEA412
47	MPFCA101
48	MPFCA205
49	EIC5127
50	MTM115
51	EC5200
52	EC5280
53	EMM528
54	CI016
55	CI020
56	CI011
57	MPFCA100
58	MTM110
59	EIC5124
60	EEC5022
61	MPFCA104
62	MPFCA200
63	MEAM5000
64	MEMT106
65	MDI1106
66	MTM114
67	MVC1211
68	MPPAU1114
69	MPPAU2219
70	MEM180
71	CI038
72	MEA112
73	MEA217
74	MEA320
75	EC5287
76	EIC5125
77	MPFCA105
78	MPFCA201
79	MPFCA202
80	MPFCA206
81	EIC5126
82	MPFCA107
83	MEM188
84	MFAMF1108
85	EQ407
86	MEM175
87	MDI1100
88	MDI1204
89	MPPAU2220
90	MEM191
91	MEM163
92	CI009
93	MEA415
94	MEM184
95	CI023
96	MPPAU2216
97	MEB205
98	EIC5123
99	EIC4223
100	EIC5122
101	MEMT105
102	CI037
103	CI025
104	MEM1205
105	MDI1207
106	MDI1209
107	CI002
108	CI019
109	EEC2207
110	GEI512
111	MEB204
112	MEAM1314
113	MTM104
114	EIC5129
115	EQ411
116	MMCCE1220
117	EIC4224
118	MEMT110
119	MEMT120
120	MDI1206
121	MEST210
122	EQ308
123	MEM5000
124	EEC5272
125	MPPAU1112
126	MEM158
127	MEM182
128	MEM183
129	CI003
130	CI004
131	CI013
132	MEA216
133	MEA319
134	MPPAU1115
135	MPPAU2218
137	MPFCA102
138	MPFCA203

(figure 10 - Retrieved Results for Question 3)

a) **Use not in.**

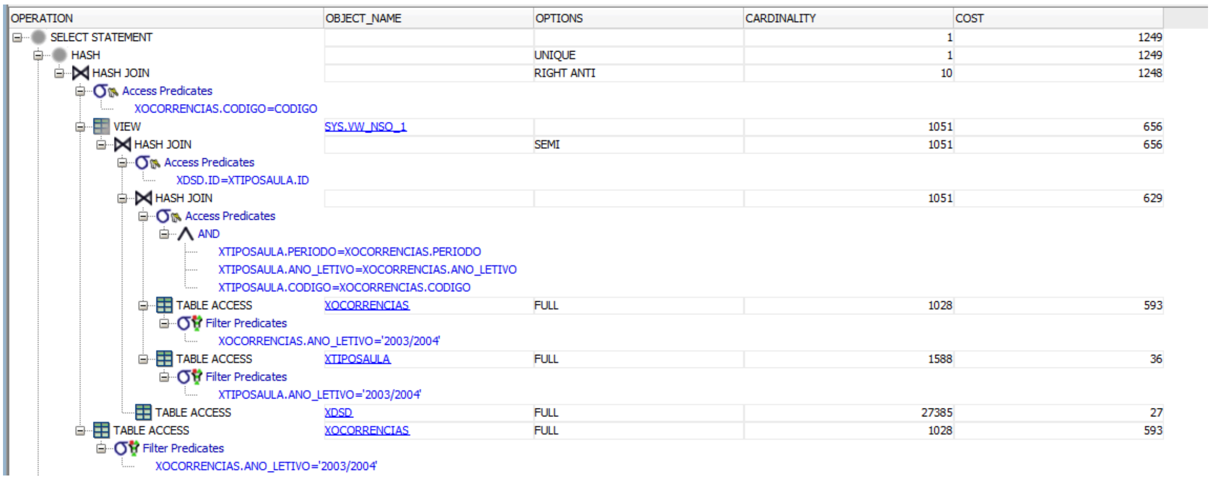
SQL Query:

```
SELECT DISTINCT ocorrencias.codigo
FROM ocorrencias
WHERE ocorrencias.ano_letivo = '2003/2004'
AND ocorrencias.codigo NOT IN (
    SELECT DISTINCT ocorrencias.codigo
    FROM dsd
    LEFT JOIN tiposaula ON dsd.ID = tiposaula.ID
    LEFT JOIN ocorrencias ON tiposaula.codigo = ocorrencias.codigo AND
tiposaula.ano_letivo = ocorrencias.ano_letivo AND tiposaula.periodo =
ocorrencias.periodo
    WHERE ocorrencias.ano_letivo = '2003/2004'
)
```

Execution Data:

Environment X:

Query Execution Time: 0,102 seconds

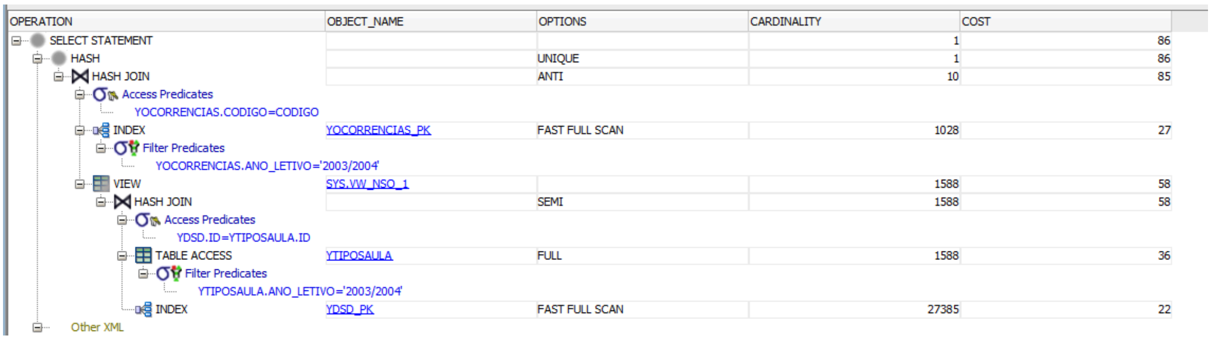


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1249
HASH		UNIQUE	1	1249
HASH JOIN		RIGHT ANTI	10	1248
Access Predicates				
XOCORRENCIAS.CODIGO=CODIGO				
VIEW	SYS.VW_NSO_1		1051	656
HASH JOIN		SEMI	1051	656
Access Predicates				
XDSD.ID=XTIPOSAULA.ID				
HASH JOIN			1051	629
Access Predicates				
AND				
XTIPOSAULA.PERIODO=XOCORRENCIAS.PERIODO				
XTIPOSAULA.ANO_LETIVO=XOCORRENCIAS.ANO_LETIVO				
XTIPOSAULA.CODIGO=XOCORRENCIAS.CODIGO				
TABLE ACCESS	XOCORRENCIAS	FULL	1028	593
Filter Predicates				
XOCORRENCIAS.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XTIPOSAULA	FULL	1588	36
Filter Predicates				
XTIPOSAULA.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XDSD	FULL	27385	27
TABLE ACCESS	XOCORRENCIAS	FULL	1028	593
Filter Predicates				
XOCORRENCIAS.ANO_LETIVO='2003/2004'				

(figure 11 - Explain Plan of Environment X for Question 3)

Environment Y:

Query Execution Time: 0,087 seconds

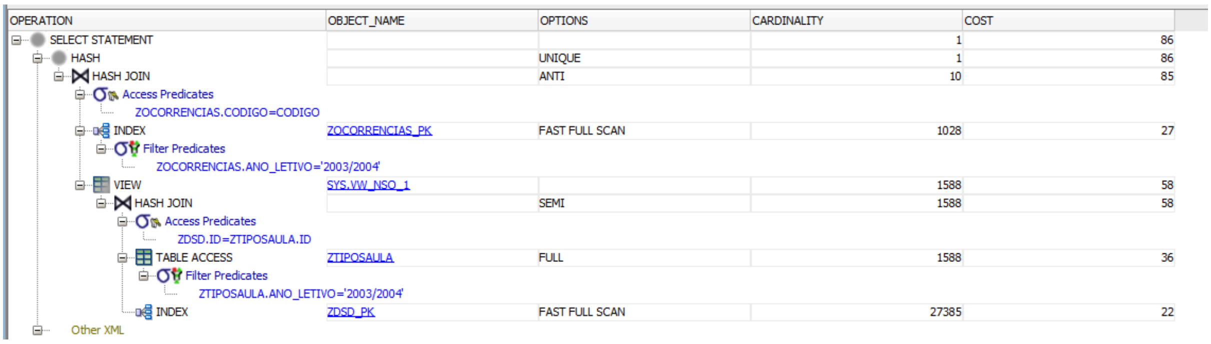


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	86
HASH		UNIQUE	1	86
HASH JOIN		ANTI	10	85
Access Predicates				
YOCORRENCIAS.CODIGO=CODIGO				
INDEX	YOCORRENCIAS_PK	FAST FULL SCAN	1028	27
Filter Predicates				
YOCORRENCIAS.ANO_LETIVO='2003/2004'				
VIEW	SYS.VW_NSO_1		1588	58
HASH JOIN		SEMI	1588	58
Access Predicates				
YDSD.ID=YTIPOSAULA.ID				
TABLE ACCESS	YTIPOSAULA	FULL	1588	36
Filter Predicates				
YTIPOSAULA.ANO_LETIVO='2003/2004'				
INDEX	YDSD_PK	FAST FULL SCAN	27385	22
Other XML				

(figure 12 - Explain Plan of Environment Y for Question 3)

Environment Z:

Query Execution Time: 0,084 seconds



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	86
HASH		UNIQUE	1	86
HASH JOIN		ANTI	10	85
Access Predicates				
ZOCORRENCIAS.CODIGO=CODIGO				
INDEX	ZOCORRENCIAS_PK	FAST FULL SCAN	1028	27
Filter Predicates				
ZOCORRENCIAS.ANO_LETIVO='2003/2004'				
VIEW	SYS.VW_NSO_1		1588	58
HASH JOIN		SEMI	1588	58
Access Predicates				
ZDSD.ID=ZTIPOSAULA.ID				
TABLE ACCESS	ZTIPOSAULA	FULL	1588	36
Filter Predicates				
ZTIPOSAULA.ANO_LETIVO='2003/2004'				
INDEX	ZDSD_PK	FAST FULL SCAN	27385	22
Other XML				

(figure 13 - Explain Plan of Environment Z for Question 3)

Comparison:

Comparing the execution data of the environments, we noticed a great improvement with the addition of integrity constraints (PK and FK). The cost of queries executed in the environments Y and Z had a cost 1400% smaller than the cost of the query in the environment X.

This time, the indexes didn't make a big difference, the cost remained the same as the cost in the environment Y.

The execution times didn´t oscillate much, but it followed a decreasing trend.

b) Use external join, and it is null.

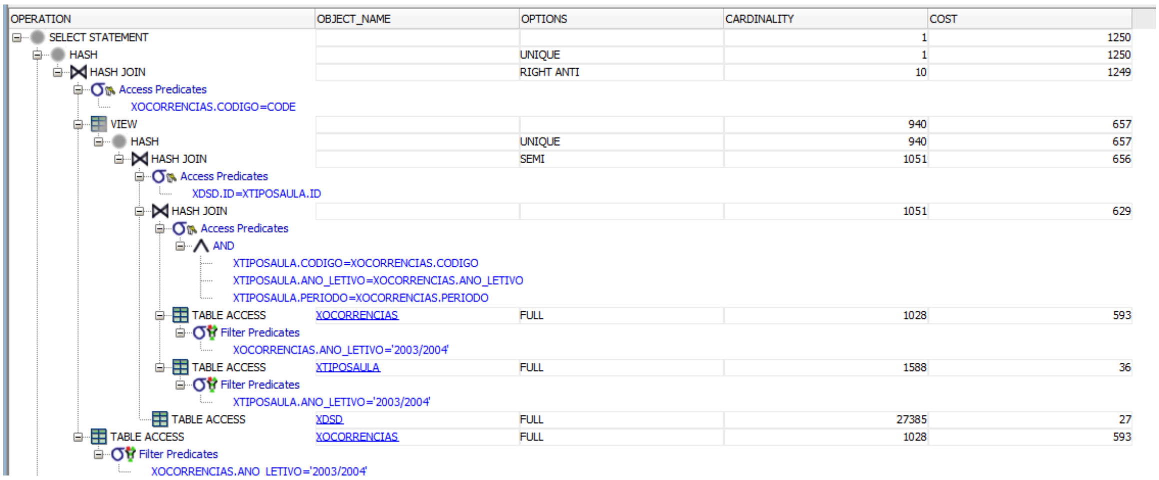
SQL Query:

```sql
SELECT DISTINCT ucs.codigo
FROM ocorrencias
LEFT JOIN (
    SELECT DISTINCT ocorrencias.codigo AS code
    FROM dsd
    LEFT JOIN tiposaula ON dsd.ID = tiposaula.ID
    LEFT JOIN ocorrencias ON tiposaula.codigo = ocorrencias.codigo AND
tiposaula.ano_letivo = ocorrencias.ano_letivo AND tiposaula.periodo =
ocorrencias.periodo
    WHERE ocorrencias.ano_letivo = '2003/2004'
) ON ocorrencias.codigo = code
WHERE ocorrencias.ano_letivo = '2003/2004' and code IS NULL
```

Execution Data:

Environment X:

Query Execution Time: 0,098 seconds

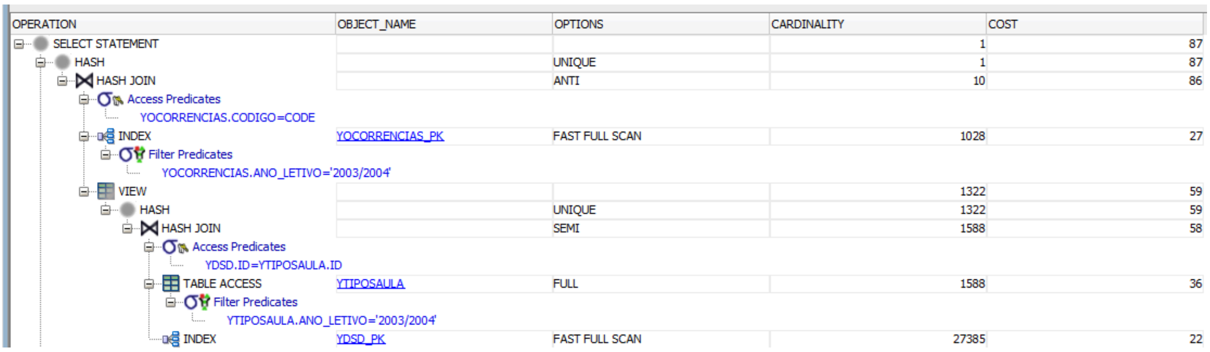


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1250
HASH		UNIQUE	1	1250
HASH JOIN		RIGHT ANTI	10	1249
Access Predicates				
XOCORRENCIAS.CODIGO=CODE				
VIEW			940	657
HASH		UNIQUE	940	657
HASH JOIN		SEMI	1051	656
Access Predicates				
XDSD.ID=XTIPOSAULA.ID				
HASH JOIN			1051	629
Access Predicates				
AND				
XTIPOSAULA.CODIGO=XOCORRENCIAS.CODIGO				
XTIPOSAULA.ANO_LETIVO=XOCORRENCIAS.ANO_LETIVO				
XTIPOSAULA.PERIODO=XOCORRENCIAS.PERIODO				
TABLE ACCESS	XOCORRENCIAS	FULL	1028	593
Filter Predicates				
XOCORRENCIAS.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XTIPOSAULA	FULL	1588	36
Filter Predicates				
XTIPOSAULA.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XDSD	FULL	27385	27
TABLE ACCESS	XOCORRENCIAS	FULL	1028	593
Filter Predicates				
XOCORRENCIAS.ANO_LETIVO='2003/2004'				

(figure 15 - Explain Plan of Environment X for Question 3)

Environment Y:

Query Execution Time: 0,083 seconds

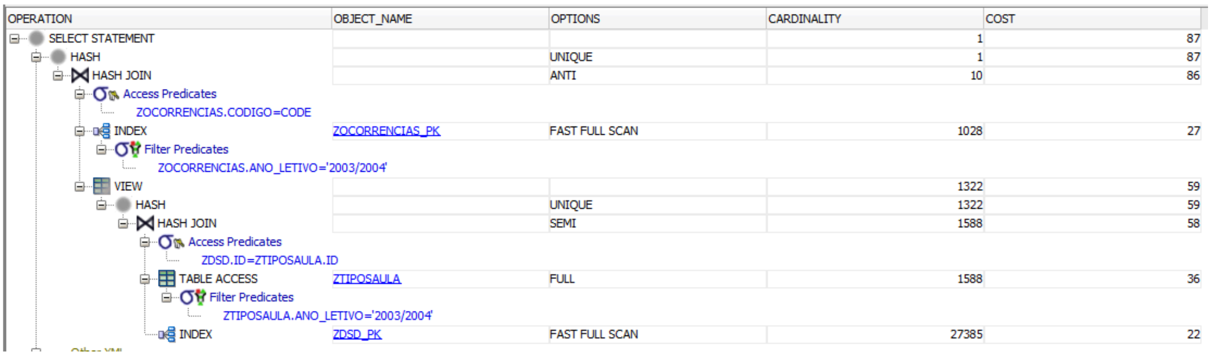


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	87
HASH		UNIQUE	1	87
HASH JOIN		ANTI	10	86
Access Predicates				
YOCORRENCIAS.CODIGO=CODE				
INDEX	YOCORRENCIAS_PK	FAST FULL SCAN	1028	27
Filter Predicates				
YOCORRENCIAS.ANO_LETIVO='2003/2004'				
VIEW			1322	59
HASH		UNIQUE	1322	59
HASH JOIN		SEMI	1588	58
Access Predicates				
YDSD.ID=YTIPOSAULA.ID				
TABLE ACCESS	YTIPOSAULA	FULL	1588	36
Filter Predicates				
YTIPOSAULA.ANO_LETIVO='2003/2004'				
INDEX	YDSD_PK	FAST FULL SCAN	27385	22

(figure 16 - Explain Plan of Environment Y for Question 3)

Environment Z:

Query Execution Time: 0,080 seconds



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	87
HASH		UNIQUE	1	87
HASH JOIN		ANTI	10	86
Access Predicates				
ZOCORRENCIAS.CODIGO=CODE				
INDEX	ZOCORRENCIAS_PK	FAST FULL SCAN	1028	27
Filter Predicates				
ZOCORRENCIAS.ANO_LETIVO='2003/2004'				
VIEW			1322	59
HASH		UNIQUE	1322	59
HASH JOIN		SEMI	1588	58
Access Predicates				
ZDSD.ID=ZTIPOSAULA.ID				
TABLE ACCESS	ZTIPOSAULA	FULL	1588	36
Filter Predicates				
ZTIPOSAULA.ANO_LETIVO='2003/2004'				
INDEX	ZDSD_PK	FAST FULL SCAN	27385	22

(figure 17 - Explain Plan of Environment Z for Question 3)

Comparison:

Comparing the execution data of the three environments, we noticed a great improvement with the addition of integrity constraints (PK and FK). The cost of queries executed in the environments Y and Z had a cost 1400% smaller than the cost of the query in the environment X.

This time, the indexes didn't make a big difference, the cost remained the same as the cost in the environment Y.

The execution times oscillated a bit, but it isn’t a good parameter to retrieve more conclusions for this comparison.

Comparison of the Two Approaches:

Using the *Not In* clause with a subquery (Approach A) or using the *External Join* and checking if the *join* result is *null* (Approach B) didn't make any big difference in the execution times and costs.

Since both use a previously calculated *view*, either to check the *Not In* or to *Join*, they are going to accumulate that execution/calculation cost.

The major difference remains in the *where* conditions of the queries. In approach A the query only needs to check the entire *view* (previously calculated) for each record/row, but in approach B it only needs to check if the *join* to the other table was successful or not (check if the Id of the other table was filled). We think that the little execution time difference, between the two approaches, is caused by this.

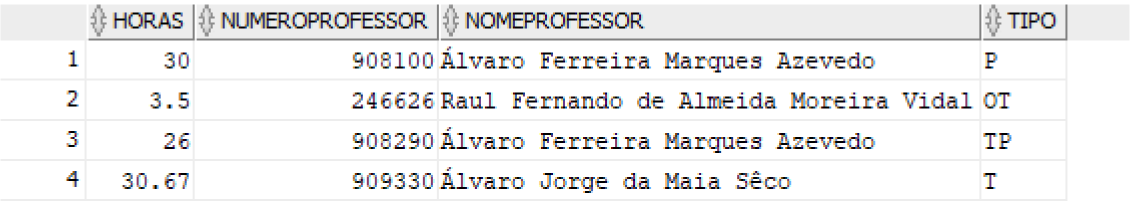
3.4. Question 4

Description: Who was the professor with the most class hours for each class type in 2003/2004? Show the number and name of the professor, the kind of class, and the total number of class hours times the factor.

SQL Query:

```
SELECT MAX(Horas) AS Horas, MAX(NumeroProfessor) AS NumeroProfessor,
MAX(NomeProfessor) AS NomeProfessor, Tipo
FROM (
    SELECT SUM(dsd.horas) * dsd.fator AS Horas, tiposaula.tipo AS Tipo,
docentes.NR AS NumeroProfessor, docentes.nome AS NomeProfessor
    FROM dsd
    LEFT JOIN tiposaula ON dsd.ID = tiposaula.ID
    LEFT JOIN docentes ON dsd.NR = docentes.NR
    WHERE tiposaula.ano_letivo = '2003/2004' AND dsd.horas IS NOT NULL AND
dsd.fator IS NOT NULL
    GROUP BY dsd.fator, tiposaula.tipo, docentes.NR, docentes.nome
    ORDER BY Horas DESC
)
GROUP BY Tipo
```

Retrieved Results:



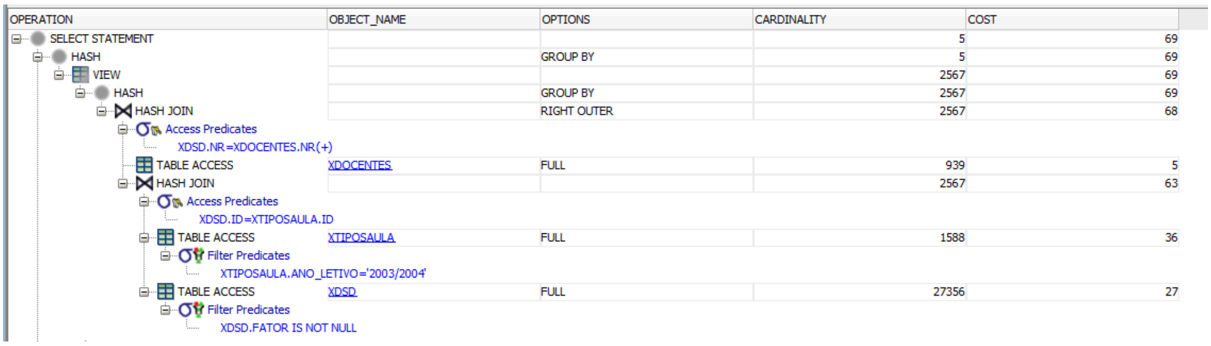
	HORAS	NUMEROPROFESSOR	NOMEPROFESSOR	TIPO
1	30	908100	Álvaro Ferreira Marques Azevedo	P
2	3.5	246626	Raul Fernando de Almeida Moreira Vidal	OT
3	26	908290	Álvaro Ferreira Marques Azevedo	TP
4	30.67	909330	Álvaro Jorge da Maia Sêco	T

(figure 18 - Retrieved Results for Question 4)

Execution Data:

Environment X:

Query Execution Time: 0,035 seconds

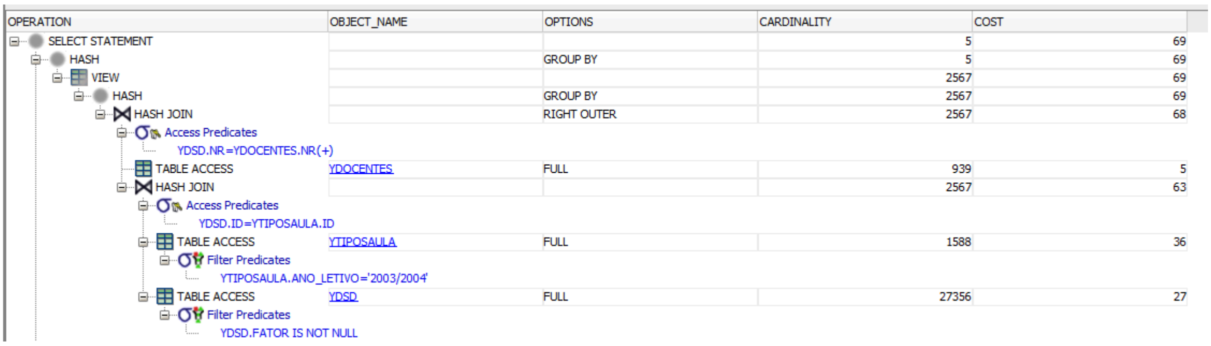


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	69
HASH		GROUP BY	5	69
VIEW			2567	69
HASH		GROUP BY	2567	69
HASH JOIN		RIGHT OUTER	2567	68
Access Predicates				
XDSD.NR=XDOCENTES.NR(+)				
TABLE ACCESS	XDOCENTES	FULL	939	5
HASH JOIN			2567	63
Access Predicates				
XDSD.ID=XTIPOSAULA.ID				
TABLE ACCESS	XTIPOSAULA	FULL	1588	36
Filter Predicates				
XTIPOSAULA.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XDSD	FULL	27356	27
Filter Predicates				
XDSD.FATOR IS NOT NULL				

(figure 19 - Explain Plan of Environment X for Question 4)

Environment Y:

Query Execution Time: 0,032 seconds

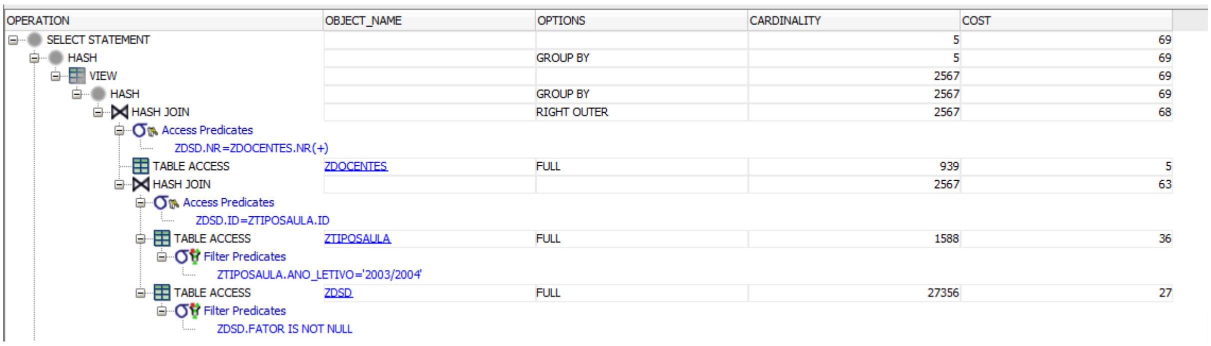


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	69
HASH		GROUP BY	5	69
VIEW			2567	69
HASH		GROUP BY	2567	69
HASH JOIN		RIGHT OUTER	2567	68
Access Predicates				
YDSD.NR=YDOCENTES.NR(+)				
TABLE ACCESS	YDOCENTES	FULL	939	5
HASH JOIN			2567	63
Access Predicates				
YDSD.ID=YTIPOSAULA.ID				
TABLE ACCESS	YTIPOSAULA	FULL	1588	36
Filter Predicates				
YTIPOSAULA.ANO_LETIVO='2003/2004'				
TABLE ACCESS	YDSD	FULL	27356	27
Filter Predicates				
YDSD.FATOR IS NOT NULL				

(figure 20 - Explain Plan of Environment Y for Question 4)

Environment Z:

Query Execution Time: 0,038 seconds



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	69
HASH		GROUP BY	5	69
VIEW			2567	69
HASH		GROUP BY	2567	69
HASH JOIN		RIGHT OUTER	2567	68
Access Predicates				
ZDSD.NR=ZDOCENTES.NR(+)				
TABLE ACCESS	ZDOCENTES	FULL	939	5
HASH JOIN			2567	63
Access Predicates				
ZDSD.ID=ZTIPOSAULA.ID				
TABLE ACCESS	ZTIPOSAULA	FULL	1588	36
Filter Predicates				
ZTIPOSAULA.ANO_LETIVO='2003/2004'				
TABLE ACCESS	ZDSD	FULL	27356	27
Filter Predicates				
ZDSD.FATOR IS NOT NULL				

(figure 21 - Explain Plan of Environment Z for Question 4)

Comparison:

This comparison of execution environments was not conclusive at all. The cost of the query was the same in the three environments and even the operations executed for the query completion were the same, as we can see in the figures 19, 20 and 21.

The only difference we noticed was a little oscillation in the execution times, but it wasn't significant.

3.5. Question 5

Description: Compare the execution plans (only the environment Z) and the index sizes for the query, giving the course code, the academic year, the period, and the number of hours of the type 'OT' in the academic years of 2002/2003 and 2003/2004.

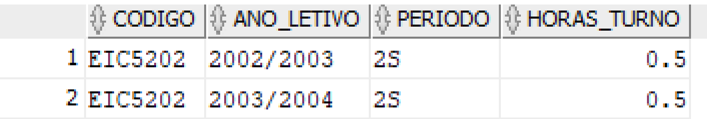
a) With a B-tree index on the type and academic year columns of the ZTIPOSAULA table.

b) With a bitmap index on the type and academic year columns of the ZTIPOSAULA table.

SQL Query:

```sql
SELECT ztiposaula.codigo, ztiposaula.ano_letivo, ztiposaula.periodo,
ztiposaula.horas_turno
FROM ztiposaula
WHERE ztiposaula.tipo = 'OT' AND (ztiposaula.ano_letivo = '2002/2003' OR
ztiposaula.ano_letivo = '2003/2004')
```

Retrieved Results:



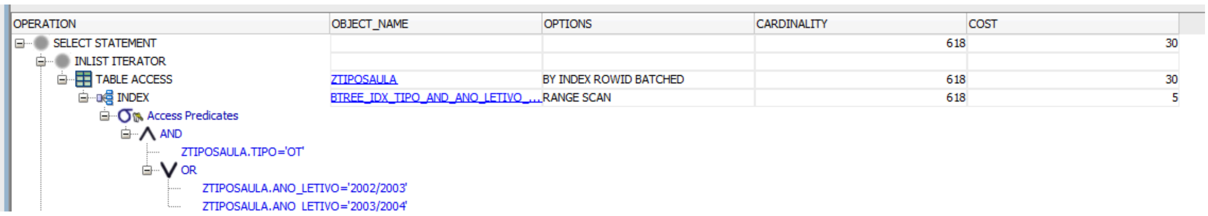
	CODIGO	ANO_LETIVO	PERIODO	HORAS_TURNO
1	EIC5202	2002/2003	2S	0.5
2	EIC5202	2003/2004	2S	0.5

(figure 22 - Retrieved Results for Question 5)

Execution Data:

Index Approach A:

Query Execution Time: 0,002 seconds

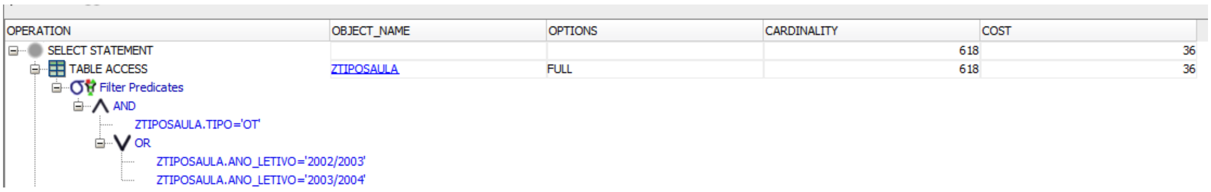


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			618	30
INLIST ITERATOR				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID BATCHED	618	30
INDEX	BTREE_IDX_TIPO_AND_ANO_LETIVO_...	RANGE SCAN	618	5
Access Predicates				
AND				
ZTIPOSAULA.TIPO='OT'				
OR				
ZTIPOSAULA.ANO_LETIVO='2002/2003'				
ZTIPOSAULA.ANO_LETIVO='2003/2004'				

(figure 23 - Explain Plan of Approach A for Question 5)

Index Approach B:

Query Execution Time: 0,005 seconds



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			618	36
TABLE ACCESS	ZTIPOSAULA	FULL	618	36
Filter Predicates				
AND				
ZTIPOSAULA.TIPO='OT'				
OR				
ZTIPOSAULA.ANO_LETIVO='2002/2003'				
ZTIPOSAULA.ANO_LETIVO='2003/2004'				

(figure 24 - Explain Plan of Approach B for Question 5)

Comparison:

A B-tree index is a balanced tree structure commonly used in databases to organize and search for keys in sorted order, making it efficient for range queries and disk-based systems. On the other hand, a Bitmap index represents a set of keys using a bitmap, providing space-efficient storage and fast set operations, particularly suitable for low cardinality data. The choice between them depends on factors such as the nature of the dataset and the types of queries to be performed.

Analyzing the data from the two executions (A (B-tree index) and B (Bitmap index)) we concluded that the B-tree index in approach A was a bit more efficient than the Bitmap index in approach B. The time in approach A was more than half of the time in B, and the cost for the B-tree index was 20% smaller than the cost for the Bitmap index.

With a cardinality greater than 600, and with the analysis we have done above, we think that a B-tree index is a better choice, in this case.

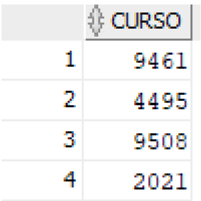
3.6. Question 6

Description: Select the programs (curso) with classes with all the existing types.

SQL Query:

```
SELECT Curso
FROM (
    SELECT COUNT(DISTINCT tiposaula.tipo) AS NumeroTipoAulas, ucs.curso AS Curso
    FROM tiposaula
    LEFT JOIN ocorrencias ON tiposaula.codigo = ocorrencias.codigo AND tiposaula.ano_letivo = ocorrencias.ano_letivo AND tiposaula.periodo = ocorrencias.periodo
    LEFT JOIN ucs ON ocorrencias.codigo = ucs.codigo
    GROUP BY ucs.curso
)
WHERE NumeroTipoAulas = (
    SELECT COUNT (DISTINCT tiposaula.tipo)
    FROM tiposaula
)
```

Retrieved Results:



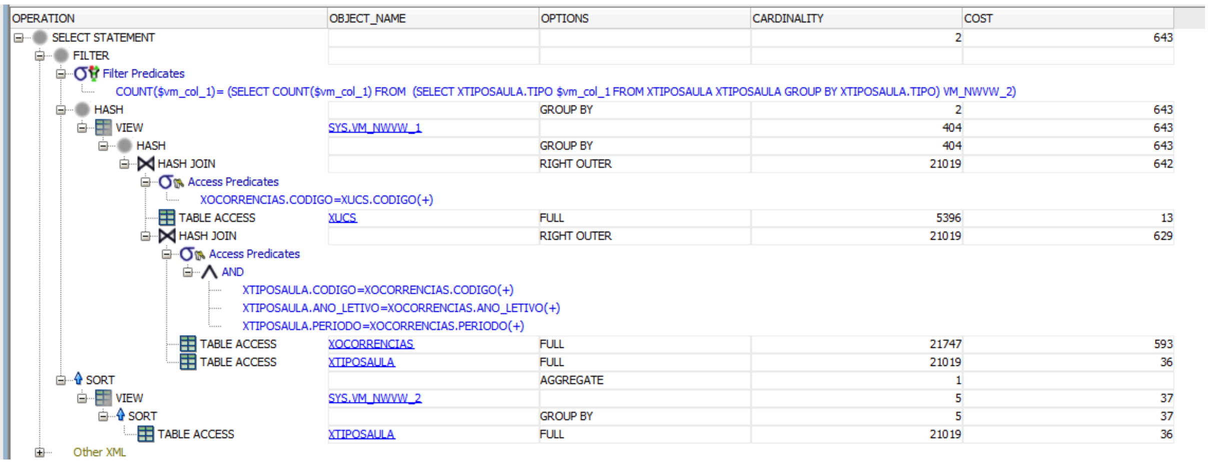
	CURSO
1	9461
2	4495
3	9508
4	2021

(figure 25 - Retrieved Results for Question 6)

Execution Data:

Environment X:

Query Execution Time: 0,046 seconds

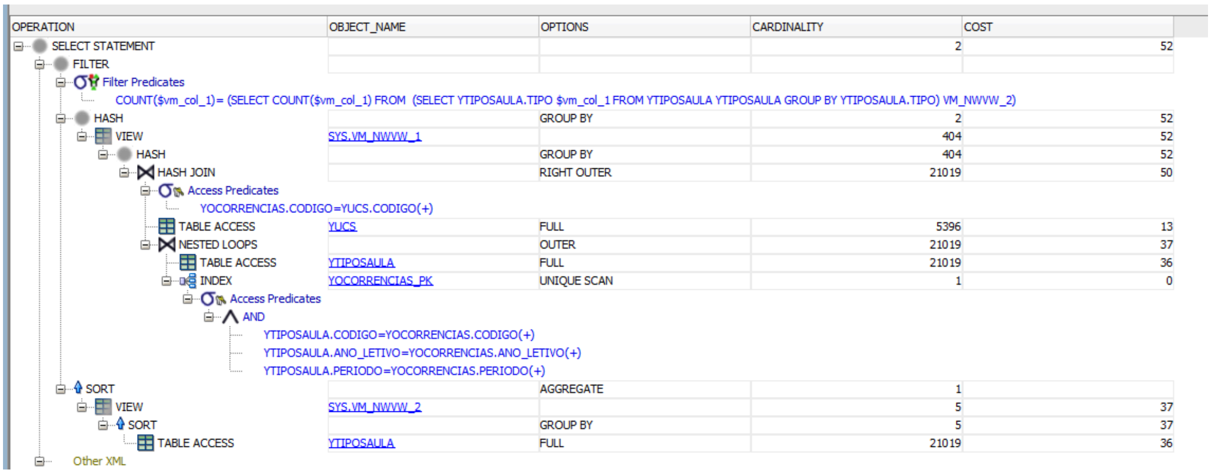


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	643
FILTER				
Filter Predicates				
COUNT($vm_col_1)= (SELECT COUNT($vm_col_1) FROM (SELECT XTIPOSAULA.TIPO $vm_col_1 FROM XTIPOSAULA XTIPOSAULA GROUP BY XTIPOSAULA.TIPO) VM_NWVW_2)				
HASH		GROUP BY	2	643
VIEW	SYS.VM_NWVW_1		404	643
HASH		GROUP BY	404	643
HASH JOIN		RIGHT OUTER	21019	642
Access Predicates				
XOCORRENCIAS.CODIGO=XUCS.CODIGO(+)				
TABLE ACCESS	XUCS	FULL	5396	13
HASH JOIN		RIGHT OUTER	21019	629
Access Predicates				
AND				
XTIPOSAULA.CODIGO=XOCORRENCIAS.CODIGO(+)				
XTIPOSAULA.ANO_LETIVO=XOCORRENCIAS.ANO_LETIVO(+)				
XTIPOSAULA.PERIODO=XOCORRENCIAS.PERIODO(+)				
TABLE ACCESS	XOCORRENCIAS	FULL	21747	593
TABLE ACCESS	XTIPOSAULA	FULL	21019	36
SORT		AGGREGATE	1	
VIEW	SYS.VM_NWVW_2		5	37
SORT		GROUP BY	5	37
TABLE ACCESS	XTIPOSAULA	FULL	21019	36
Other XML				

(figure 26 - Explain Plan of Environment X for Question 6)

Environment Y:

Query Execution Time: 0,055 seconds

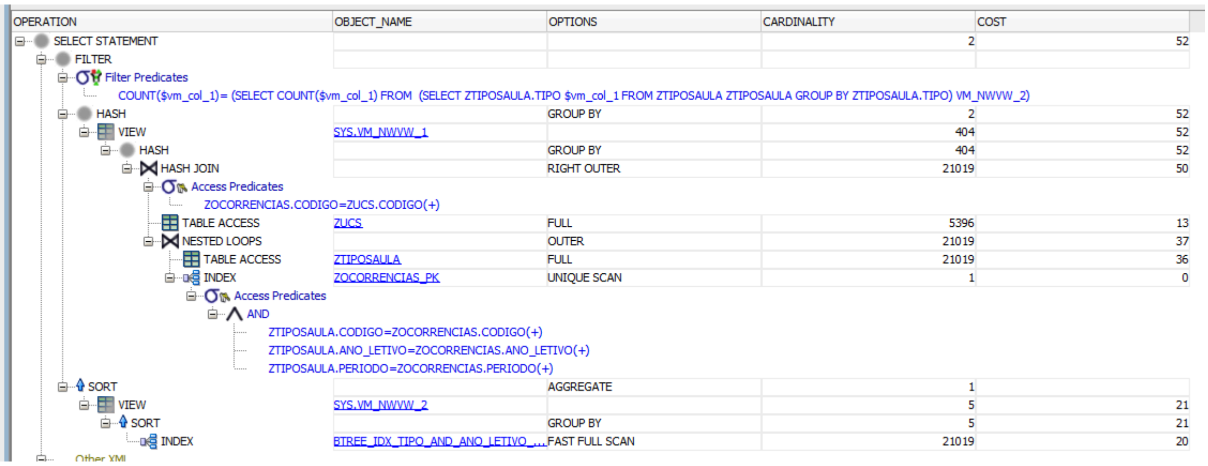


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	52
FILTER				
Filter Predicates				
COUNT($vm_col_1)= (SELECT COUNT($vm_col_1) FROM (SELECT YTIPOSAULA.TIPO $vm_col_1 FROM YTIPOSAULA YTIPOSAULA GROUP BY YTIPOSAULA.TIPO) VM_NWVW_2)				
HASH		GROUP BY	2	52
VIEW	SYS.VM_NWVW_1		404	52
HASH		GROUP BY	404	52
HASH JOIN		RIGHT OUTER	21019	50
Access Predicates				
YOCORRENCIAS.CODIGO=YUCS.CODIGO(+)				
TABLE ACCESS	YUCS	FULL	5396	13
NESTED LOOPS		OUTER	21019	37
TABLE ACCESS	YTIPOSAULA	FULL	21019	36
INDEX	YOCORRENCIAS_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
YTIPOSAULA.CODIGO=YOCORRENCIAS.CODIGO(+)				
YTIPOSAULA.ANO_LETIVO=YOCORRENCIAS.ANO_LETIVO(+)				
YTIPOSAULA.PERIODO=YOCORRENCIAS.PERIODO(+)				
SORT		AGGREGATE	1	
VIEW	SYS.VM_NWVW_2		5	37
SORT		GROUP BY	5	37
TABLE ACCESS	YTIPOSAULA	FULL	21019	36
Other XML				

(figure 27 - Explain Plan of Environment Y for Question 6)

Environment Z:

Query Execution Time: 0,051 seconds



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	52
FILTER				
Filter Predicates				
COUNT($vm_col_1)= (SELECT COUNT($vm_col_1) FROM (SELECT ZTIPOSAULA.TIPO $vm_col_1 FROM ZTIPOSAULA ZTIPOSAULA GROUP BY ZTIPOSAULA.TIPO) VM_NWVW_2)				
HASH		GROUP BY	2	52
VIEW	SYS.VM_NWVW_1		404	52
HASH		GROUP BY	404	52
HASH JOIN		RIGHT OUTER	21019	50
Access Predicates				
ZOCORRENCIAS.CODIGO=ZUCS.CODIGO(+)				
TABLE ACCESS	ZUCS	FULL	5396	13
NESTED LOOPS		OUTER	21019	37
TABLE ACCESS	ZTIPOSAULA	FULL	21019	36
INDEX	ZOCORRENCIAS_PK	UNIQUE SCAN	1	0
Access Predicates				
AND				
ZTIPOSAULA.CODIGO=ZOCORRENCIAS.CODIGO(+)				
ZTIPOSAULA.ANO_LETIVO=ZOCORRENCIAS.ANO_LETIVO(+)				
ZTIPOSAULA.PERIODO=ZOCORRENCIAS.PERIODO(+)				
SORT		AGGREGATE	1	
VIEW	SYS.VM_NWVW_2		5	21
SORT		GROUP BY	5	21
INDEX	BTREE_IDX_TIPO_AND_ANO_LETIVO_...	FAST FULL SCAN	21019	20
Other XML				

(figure 28 - Explain Plan of Environment Z for Question 6)

Comparison:

Analyzing the three environments explain plans, we can conclude that the addition of integrity constraints (primary keys (PK) and foreign keys (FK)) is essential for query performance, because the cost has reduced tremendously from the environment X to the environment Y.

We also noticed that the index applied in the table tiposaula has helped with the reduction of the cost too, although it was a very tiny improvement compared to the one made from the integrity constraints.

About the time execution, it was not very conclusive because the execution times collected were a lot similar, we only noticed a little oscillation in the execution times, but it wasn’t significant.

4. Conclusions

Through comprehensive analysis of results, costs, and execution times across different query scenarios and execution environments, several key learning lessons and takeaways have emerged.

The study on query optimization revealed the importance of effective query organization, indexing, and query restructuring in improving database performance. By aligning query structures with the data model and using appropriate indexing techniques, execution times were significantly improved. Environmental factors, such as system resources and concurrent workloads, also influence execution statistics. Indexing plays a crucial role in query performance, reducing execution times but requiring a balance between utilization and maintenance costs.