Database Technologies (TBD)
2021/2022

Database Query Optimization
Teaching Service

Henrique Reis Sendim Rodrigues (UP201606462)
João Carlos Machado Rocha Pires (UP201806079)
Mariana Catarina Pereira Soares (UP201605775)

U. PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Contents

1. Problem description

Our problem consists on a database with information regarding the distribution of teaching service in a faculty.

There are courses (table XUCS), described by a code (codigo), a designation (designacao), an acronym (sigla_uc) and a program (curso). Courses have occurrences in several years. Each occurrence is recorded by a row in the table XOCORRENCIAS, with information on the course code (codigo), academic year (ano_letivo), period of classes (periodo, that may be A-annual, 1S- first semester, 1T- first trimester, etc.), number of enrolled students (inscritos), students with distributed assessment (com_frequencia), number of approved (aprovados), course goals (objetivos) and content (conteudo), and department in charge (departamento).

Each occurrence may have one or more class types (T-theoretic, P-practical, L-laboratory, TP- theoretic/practical, OT- tutorial guidance). Each class type for an occurrence is recorded on table XTIPOSAULA with the number of similar classes (turnos), the number of week hours for each class (horas_turno), and in some cases the number of weekly classes (n_aulas).

The table XDSD records the teaching service distribution, in each semester, for each professor. More specifically, it records, for each class type of an 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, in the perspective of the professor, may be less than 1 and recorded in attribute fator. Otherwise, the attribute fator will be 1. From the program perspective, the attribute fator is ignored. The attribute ordem enables listing the set of professors of a specific course occurrence in a specific order.

The professors are recorded in the table XDOCENTES with a number (nr), a name (nome), an acronym (sigla), a category code (categoria), a given name (proprio), a family name (apelido), and a status (estado: A-ativo, NA-não ativo, R-reformado).

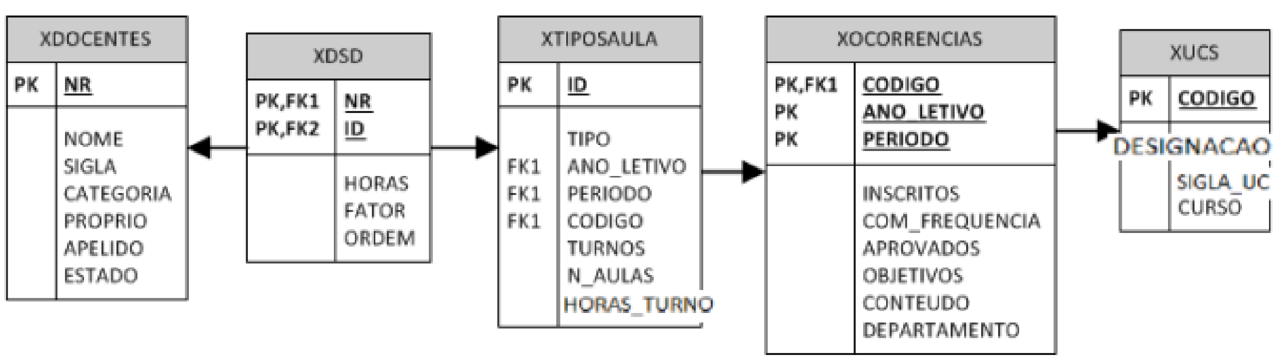


Figure 1.1: Relational model for the case Teaching Service.

2. Setup

In the setup phase, we've copied the tables XDOCENTES, XDSD, XOCORRENCIAS, XTIPOSAULA and XUCS three times, each one with the prefixes 'X', 'Y' and 'Z'.

- X – no indexes and no integrity constraints.
- Y – with the standard integrity constraints (primary keys and foreign keys).
- Z – with the standard integrity constraints and extra indexes (the extra indexes were created as needed, which means they'll be justified in each question and their use is cumulative, starting from question 1 and ending up in question 6).

The following SQL code represents the copy process, followed by the creation of the standard integrity constraints for both 'Y' and 'Z' prefixed tables.

```sql
/* COPY TABLES - X, Y, Z */

CREATE TABLE XDOCENTES AS SELECT * FROM GTD10.xdocentes;
CREATE TABLE YDOCENTES AS SELECT * FROM GTD10.xdocentes;
CREATE TABLE ZDOCENTES AS SELECT * FROM GTD10.xdocentes;

CREATE TABLE XDSD AS SELECT * FROM GTD10.xdsd;
CREATE TABLE YDSD AS SELECT * FROM GTD10.xdsd;
CREATE TABLE ZDSD AS SELECT * FROM GTD10.xdsd;

CREATE TABLE XOCORRENCIAS AS SELECT * FROM GTD10.xocorrencias;
CREATE TABLE YOCORRENCIAS AS SELECT * FROM GTD10.xocorrencias;
CREATE TABLE ZOCORRENCIAS AS SELECT * FROM GTD10.xocorrencias;

CREATE TABLE XTIPOSAULA AS SELECT * FROM GTD10.xtiposaula;
CREATE TABLE YTIPOSAULA AS SELECT * FROM GTD10.xtiposaula;
CREATE TABLE ZTIPOSAULA AS SELECT * FROM GTD10.xtiposaula;

CREATE TABLE XUCS AS SELECT * FROM GTD10.xucs;
CREATE TABLE YUCS AS SELECT * FROM GTD10.xucs;
CREATE TABLE ZUCS AS SELECT * FROM GTD10.xucs;

/* CREATE PRIMARY AND FOREIGN KEY CONSTRAINTS Y */

ALTER TABLE YDOCENTES ADD CONSTRAINT YDOCENTES_PK PRIMARY KEY (NR);
ALTER TABLE YDSD ADD CONSTRAINT YDSD_PK PRIMARY KEY (NR, ID);
ALTER TABLE YOCORRENCIAS ADD CONSTRAINT YOCORRENCIAS_PK PRIMARY KEY (CODIGO, ANO_LETIVO,
    PERIODO);
ALTER TABLE YTIPOSAULA ADD CONSTRAINT YTIPOSAULA_PK PRIMARY KEY (ID);
ALTER TABLE YUCS ADD CONSTRAINT YUCS_PK PRIMARY KEY (CODIGO);

ALTER TABLE YDSD ADD CONSTRAINT YDSD_FK1 FOREIGN KEY (NR) REFERENCES YDOCENTES(NR);
```

```
ALTER TABLE YDSD ADD CONSTRAINT YDSD_FK2 FOREIGN KEY (ID) REFERENCES YTIPOSAULA(ID);
ALTER TABLE YOCORRENCIAS ADD CONSTRAINT YOCORRENCIAS_FK FOREIGN KEY (CODIGO) REFERENCES
    YUCS(CODIGO);
ALTER TABLE YTIPOSAULA ADD CONSTRAINT YTIPOSAULA_FK FOREIGN KEY (ANO_LETIVO, PERIODO,
    CODIGO) REFERENCES YOCORRENCIAS(ANO_LETIVO, PERIODO, CODIGO);

/* CREATE PRIMARY AND FOREIGN KEY CONSTRAINTS Z */

ALTER TABLE ZDOCENTES ADD CONSTRAINT ZDOCENTES_PK PRIMARY KEY (NR);
ALTER TABLE ZDSD ADD CONSTRAINT ZDSD_PK PRIMARY KEY (NR, ID);
ALTER TABLE ZOCORRENCIAS ADD CONSTRAINT ZOCORRENCIAS_PK PRIMARY KEY (CODIGO, ANO_LETIVO,
     PERIODO);
ALTER TABLE ZTIPOSAULA ADD CONSTRAINT ZTIPOSAULA_PK PRIMARY KEY (ID);
ALTER TABLE ZUCS ADD CONSTRAINT ZUCS_PK PRIMARY KEY (CODIGO);

ALTER TABLE ZDSD ADD CONSTRAINT ZDSD_FK1 FOREIGN KEY (NR) REFERENCES ZDOCENTES(NR);
ALTER TABLE ZDSD ADD CONSTRAINT ZDSD_FK2 FOREIGN KEY (ID) REFERENCES ZTIPOSAULA(ID);
ALTER TABLE ZOCORRENCIAS ADD CONSTRAINT ZOCORRENCIAS_FK FOREIGN KEY (CODIGO) REFERENCES
    ZUCS(CODIGO);
ALTER TABLE ZTIPOSAULA ADD CONSTRAINT ZTIPOSAULA_FK FOREIGN KEY (ANO_LETIVO, PERIODO,
    CODIGO) REFERENCES ZOCORRENCIAS(ANO_LETIVO, PERIODO, CODIGO);
```

3. Questions

In this chapter, each subsection will present the SQL query and the corresponding answer, as well as the execution plans and time for each one of the six raised questions. Each subsection starts with the question itself.

Note 1: It will only be presented the SQL query for the 'X' prefixed tables execution plan. The queries for the 'Y' and 'Z' prefixed tables execution plans are similar, with the difference of the prefix. The same happens to the result, which is the same for all, as expected.

Note 2: The execution times were calculated using the average of the obtained times for three consecutive executions on the same machine.

3.1 Question 1

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

3.1.1 SQL query

The following SQL query selects the 'codigo', 'designacao', 'ano_letivo', 'inscritos', 'tipo' and 'turnos' fields from the join between the tables 'xucs', 'xocorrencias' and 'xtiposaula', with the constraint of the 'designacao' being equal to 'Bases de Dados' and 'curso' equal to 275.

```
SELECT xucs.codigo, xucs.designacao, xocorrencias.ano_letivo, xocorrencias.inscritos,
    xtiposaula.tipo, xtiposaula.turnos
```

```
FROM xucs JOIN xocorrencias ON xucs.codigo=xocorrencias.codigo JOIN xtiposaula ON
    xocorrencias.ano_letivo=xtiposaula.ano_letivo and xocorrencias.codigo=xtiposaula.
    codigo and xocorrencias.periodo=xtiposaula.periodo
WHERE   xucs.designacao = 'Bases de Dados' and
        xucs.curso = 275;
```

3.1.2 Answer

The result for this first question is the following:

	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 3.1: Question 1 - Answer.

3.1.3 Execution Plans

Starting with the tables prefixed with 'X', the following picture shows the execution plan.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	642
HASH JOIN			1	642
Access Predicates				
AND				
XOCORRENCIAS.ANO_LETIVO=XTIPOSAULA.ANO_LETIVO				
XOCORRENCIAS.CODIGO=XTIPOSAULA.CODIGO				
XOCORRENCIAS.PERIODO=XTIPOSAULA.PERIODO				
XUCS.CODIGO=XOCORRENCIAS.CODIGO				
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

Figure 3.2: Question 1 - X Execution Plan.

As we can see, the operation performed was to join the tables was a Semi Merge Join with the operation total cost being 642.

Moving on to the tables prefixed with 'Y', the following picture shows the execution plan, this time with a Nested Loops join and a total cost of 55.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	55
HASH JOIN			1	55
Access Predicates				
AND				
YOCORRENCIAS.ANO_LETIVO=YTIPOSAULA.ANO_LETIVO				
YOCORRENCIAS.CODIGO=YTIPOSAULA.CODIGO				
YOCORRENCIAS.PERIODO=YTIPOSAULA.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				
YUCS.CODIGO=YOCORRENCIAS.CODIGO				
TABLE ACCESS	YOCORRENCIAS	BY INDEX ROWID	5	6
TABLE ACCESS	YTIPOSAULA	FULL	21019	36

Figure 3.3: Question 1 - Y Execution Plan.

Last but not least, the execution plan for the tables prefixed with 'Z', similar to the one above, but with a lower cost since the following index was introduced, avoiding the 'and' operator between the two contraints:

```
CREATE INDEX ZUCS_DESIGNACAO ON ZUCS(DESIGNACAO);
```

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

Figure 3.4: Question 1 - Z Execution Plan.

The reason behind the creation of this index was the fact that the 'designacao' field contains 3285 unique values, which might give us a performance advantage by having an index on it.

3.1.4 Execution Time

The execution times were 0,050 sec. for the first execution plan ('X') and 0,022 sec. for the last two ('Y' and 'Z').

3.2 Question 2

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

Note 3: For this question, we assumed that it was supposed to obtain the weekly hours for each type of class of the program 233, year 2004/2005, calculating by multiplying 'turnos' by 'horas_turno'.

3.2.1 SQL query

The following SQL query selects the 'tipo' and 'class_hours' (multiplication of 'turnos' by 'horas_-turno') from the join of 'xucs' and 'xtiposaula' tables, having as constraints the 'ano_letivo' being equal to '2004/2005' and 'curso' equal to 233, with all this grouped by the type of class.

```sql
SELECT xtiposaula.tipo, sum(xtiposaula.turnos*xtiposaula.horas_turno) as "CLASS_HOURS"
FROM xucs JOIN xtiposaula ON xucs.codigo=xtiposaula.codigo
WHERE xtiposaula.ano_letivo = '2004/2005' and xucs.curso = 233
GROUP BY xtiposaula.tipo;
```

3.2.2 Answer

The following picture shows the answer for this question.

	TIPO	CLASS_HOURS
1	P	581,5
2	TP	697,5
3	T	308

Figure 3.5: Question 2 - Answer.

3.2.3 Execution Plans

The execution plans for this question use all Hash Joins. The difference is in the one for the 'Z' prefixed tables, since we created the following index, reducing the cost of the first two executions from 50 to 43.

```sql
CREATE INDEX ZUCS_CURSO ON ZUCS(CURSO);
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	50
HASH		GROUP BY	5	50
HASH JOIN			54	49
Access Predicates				
XUCS.CODIGO=XTIPOSAULA.CODIGO				
TABLE ACCESS	XUCS	FULL	47	13
Filter Predicates				
XUCS.CURSO=233				
TABLE ACCESS	XTIPOSAULA	FULL	1106	36
Filter Predicates				
XTIPOSAULA.ANO_LETIVO='2004/2005'				

Figure 3.6: Question 2 - X Execution Plan.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	50
HASH		GROUP BY	5	50
HASH JOIN			53	49
Access Predicates				
ITEM_1=YTIPOSAULA.CODIGO				
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 3.7: Question 2 - Y Execution Plan.

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

Figure 3.8: Question 2 - Z Execution Plan.

3.2.4 Execution Time

Regarding the execution time, it was the same for all the execution plans: 0,020 sec.

3.3 Question 3

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

Note 4: This question raised some ambiguity inside the group, so we have decided to assume it was referring to something in particular: we assumed that it was intended to get the 'codigo' of the UCS that didn't had any 'inscritos' in the 'ano_letivo' equal to '2003/2004'. We also considered to include UCS with 'inscritos', but without 'com_frequencia'. However, this last option was discarded since there was no way to check when we were in the presence of a non-occurrence or of an occurrence without a single student having enough frequency.

3.3.1 a. Use not in.

3.3.1.1 SQL query

For the first part of the question, this SQL query select the 'codigo' from 'xocorrencias' where the same 'codigo' is not in the list of 'codigo' obtained from a sub-query where the 'inscritos' field is not null and the 'ano_letivo' is '2003/2004'.

```sql
SELECT CODIGO
FROM XOCORRENCIAS
WHERE CODIGO NOT IN (SELECT CODIGO FROM XOCORRENCIAS WHERE INSCRITOS IS NOT NULL AND
    ANO_LETIVO = '2003/2004') AND ANO_LETIVO = '2003/2004';
```

3.3.1.2 Answer

The answer for this first part of the question was different of the answer obtained for the second part. Having searched for the reason why that happened, we found that the 'NOT IN' in Oracle SQL has a problem with 'NULL' values, which makes it produce different results.

	CODIGO
1	MEM170
2	EIC4221
3	MEM157
4	EIC4222
5	EIC4220
6	MEM181
7	MEM189
8	EIC4225
9	EMG5206
10	MEM187
11	MEM132
12	EI1107
13	EIC5127
14	MEM164
15	MEA317
16	MEM179
17	MTM117
18	MTM111
19	MEM1208
20	MFAMF1106
21	MFAMF1207
22	EEC5225
23	MEA212
24	EIC5124
25	MEAM5000
26	EMM528
27	MEA402
28	MEEC1089
29	MEM111
30	EIC5126

	CODIGO
31	EIC5125
32	MTM114
33	EC5287
34	MEM1209
35	MEM180
36	EIC5122
37	EIC5123
38	MEM184
39	EIC4223
40	MFAMF1204
41	EI1208
42	MEM163
43	MEM153
44	MEM191
45	MMI1203
46	MEM173
47	MFAMF1108
48	MEM188
49	MEEC2105
50	MMI1104
51	MTM104
52	EIC4224
53	MEM1117
54	EIC5129
55	MMCCE1200
56	MEM183
57	MEM167
58	MEM158
59	EI1103
60	MEM182

Figure 3.9: Question 3a. - Answer.

3.3.1.3 Execution Plans

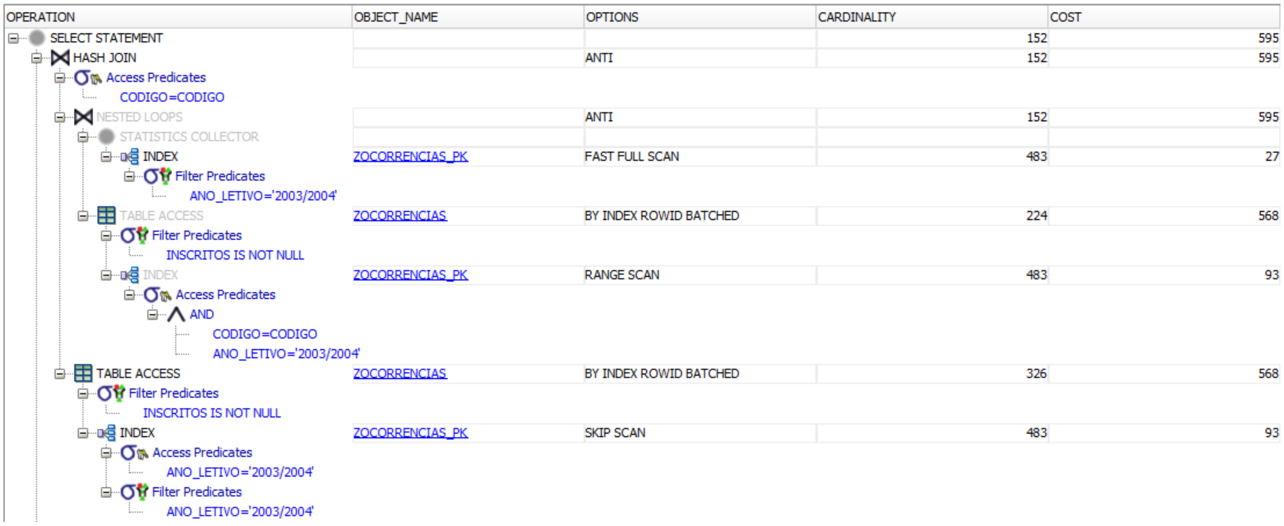
For this question, having the two indexes previously defined, we didn't found the need to create new indexes. Also, for all the execution plans, a Hash Join was verified. The cost for the first was 1185, 595 for the second and third.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			152	1185
HASH JOIN		ANTI	152	1185
Access Predicates				
CODIGO=CODIGO				
TABLE ACCESS	XOCORRENCIAS	FULL	483	593
Filter Predicates				
ANO_LETIVO='2003/2004'				
TABLE ACCESS	XOCORRENCIAS	FULL	326	593
Filter Predicates				
AND				
ANO_LETIVO='2003/2004'				
INSCRITOS IS NOT NULL				

Figure 3.10: Question 3a. - X Execution Plan.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			152	595
HASH JOIN		ANTI	152	595
Access Predicates				
CODIGO=CODIGO				
NESTED LOOPS		ANTI	152	595
STATISTICS COLLECTOR				
INDEX	YOCORRENCIAS_PK	FAST FULL SCAN	483	27
Filter Predicates				
ANO_LETIVO='2003/2004'				
TABLE ACCESS	YOCORRENCIAS	BY INDEX ROWID BATCHED	224	568
Filter Predicates				
INSCRITOS IS NOT NULL				
INDEX	YOCORRENCIAS_PK	RANGE SCAN	483	93
Access Predicates				
AND				
CODIGO=CODIGO				
ANO_LETIVO='2003/2004'				
TABLE ACCESS	YOCORRENCIAS	BY INDEX ROWID BATCHED	326	568
Filter Predicates				
INSCRITOS IS NOT NULL				
INDEX	YOCORRENCIAS_PK	SKIP SCAN	483	93
Access Predicates				
ANO_LETIVO='2003/2004'				
Filter Predicates				
ANO_LETIVO='2003/2004'				

Figure 3.11: Question 3a. - Y Execution Plan.



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			152	595
HASH JOIN		ANTI	152	595
Access Predicates				
CODIGO=CODIGO				
NESTED LOOPS		ANTI	152	595
STATISTICS COLLECTOR				
INDEX	ZOCORRENCIAS_PK	FAST FULL SCAN	483	27
Filter Predicates				
ANO_LETIVO='2003/2004'				
TABLE ACCESS	ZOCORRENCIAS	BY INDEX ROWID BATCHED	224	568
Filter Predicates				
INSCRITOS IS NOT NULL				
INDEX	ZOCORRENCIAS_PK	RANGE SCAN	483	93
Access Predicates				
AND				
CODIGO=CODIGO				
ANO_LETIVO='2003/2004'				
TABLE ACCESS	ZOCORRENCIAS	BY INDEX ROWID BATCHED	326	568
Filter Predicates				
INSCRITOS IS NOT NULL				
INDEX	ZOCORRENCIAS_PK	SKIP SCAN	483	93
Access Predicates				
ANO_LETIVO='2003/2004'				
Filter Predicates				
ANO_LETIVO='2003/2004'				

Figure 3.12: Question 3a. - Z Execution Plan.

3.3.1.4 Execution Time

Regarding the execution times, they were similar to all the execution plans: 0,046 sec.

3.3.2 b. Use external join and is null.

3.3.2.1 SQL query

For this second part of the question, instead of the use of 'NOT IN', the 'codigo' field is selected from the join between the tables 'xucs' and 'xocorrencias' considering all the entries where 'inscritos' is 'NULL' and for the 'ano_letivo' equal to '2003/2004'.

```sql
SELECT XUCS.CODIGO
FROM XUCS JOIN XOCORRENCIAS ON XUCS.CODIGO = XOCORRENCIAS.CODIGO
WHERE XOCORRENCIAS.INSCRITOS IS NULL AND XOCORRENCIAS.ANO_LETIVO = '2003/2004';
```

3.3.2.2 Answer

The answer, as mentioned above, was different.

	CODIGO
1	EC5287
2	EEC5225
3	EI1103
4	EI1107
5	EI1208
6	EIC4220
7	EIC4221
8	EIC4222
9	EIC4223
10	EIC4224
11	EIC4225
12	EIC5122
13	EIC5123
14	EIC5124
15	EIC5125
16	EIC5126
17	EIC5127
18	EIC5129
19	EM446
20	EMG5206
21	EMM528
22	GEI2414
23	MEA111
24	MEA212
25	MEA317
26	MEA402
27	MEAM5000
28	MEEC1089
29	MEEC2105
30	MEM111
31	MEM1117
32	MEM1208
33	MEM1209

	CODIGO
34	MEM132
35	MEM153
36	MEM157
37	MEM158
38	MEM163
39	MEM164
40	MEM167
41	MEM170
42	MEM173
43	MEM175
44	MEM179
45	MEM180
46	MEM181
47	MEM182
48	MEM183
49	MEM184
50	MEM187
51	MEM188
52	MEM189
53	MEM191
54	MFAMF...
55	MFAMF...
56	MFAMF...
57	MFAMF...
58	MGI1204
59	MMCCE...
60	MMI1104
61	MMI1203
62	MRSC1104
63	MTM104
64	MTM111
65	MTM114
66	MTM117

Figure 3.13: Question 3a. - Answer.

3.3.2.3 Execution Plans

Regarding the execution plans, for the first case, a Hash Join was used (cost = 606), while for the remaining ones it was a Table Access by RowID (cost = 568).

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			157	606
HASH JOIN			157	606
Access Predicates				
XUCS.CODIGO=XOCORRENCIAS.CODIGO				
TABLE ACCESS	XOCORRENCIAS	FULL	157	593
Filter Predicates				
AND				
XOCORRENCIAS.INSCRITOS IS NULL				
XOCORRENCIAS.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XUCS	FULL	5396	13

Figure 3.14: Question 3b. - X Execution Plan.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			157	568
TABLE ACCESS	YOCORRENCIAS	BY INDEX ROWID BATCHED	157	568
Filter Predicates				
YOCORRENCIAS.INSCRITOS IS NULL				
INDEX	YOCORRENCIAS_PK	SKIP SCAN	483	93
Access Predicates				
YOCORRENCIAS.ANO_LETIVO='2003/2004'				
Filter Predicates				
YOCORRENCIAS.ANO_LETIVO='2003/2004'				

Figure 3.15: Question 3b. - Y Execution Plan.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			157	568
TABLE ACCESS	ZOCORRENCIAS	BY INDEX ROWID BATCHED	157	568
Filter Predicates				
ZOCORRENCIAS.INSCRITOS IS NULL				
INDEX	ZOCORRENCIAS_PK	SKIP SCAN	483	93
Access Predicates				
ZOCORRENCIAS.ANO_LETIVO='2003/2004'				
Filter Predicates				
ZOCORRENCIAS.ANO_LETIVO='2003/2004'				

Figure 3.16: Question 3b. - Z Execution Plan.

3.3.2.4 Execution Time

Again, as for the first part of the question, the execution times were similar: 0,046 sec. ('X') and 0,042 sec. ('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

3.4.3 Execution Plans

3.4.4 Execution Time

3.5 Question 5

Compare the execution plans (just the environment Z) 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.

Note 5: This question raised some ambiguity inside the group, so we have decided to assume it was referring to something in particular: we assumed that it was intended to compare the execution plan of the query without indexes with the query using each type of indexes. We also assumed that it was supposed to compare the index size between the indexes of the type and academic year columns.

3.5.0.1 SQL query

Note 6: For this question, we assumed that it was supposed to obtain the weekly hours for the type of class equal to 'OT' of the year 2002/2003 or 2003/2004, calculating by multiplying 'turnos' by 'horas_turno'.

The following SQL query selects the 'codigo', 'ano_letivo', 'periodo' and 'num_horas' (multiplication of 'turnos' by 'horas_turno') from the join of 'zucs' and 'ztiposaula' tables, having as constraints the 'tipo' equal to 'OT' and the 'ano_letivo' equal to 2002/2003 or 2003/2004, with all this grouped by course code, academic year and period.

```
SELECT ZUCS.CURSO, ZTIPOSAULA.ANO_LETIVO, ZTIPOSAULA.PERIODO, SUM(ZTIPOSAULA.TURNOS*
    ZTIPOSAULA.HORAS_TURNO) AS "NUM_HORAS"
FROM ZUCS JOIN ZTIPOSAULA ON ZUCS.CODIGO=ZTIPOSAULA.CODIGO
WHERE (ZTIPOSAULA.ANO_LETIVO = '2002/2003' OR ZTIPOSAULA.ANO_LETIVO = '2003/2004')
AND ZTIPOSAULA.TIPO = 'OT'
GROUP BY ZUCS.CURSO, ZTIPOSAULA.ANO_LETIVO, ZTIPOSAULA.PERIODO;
```

3.5.0.2 Answer

The following picture shows the answer for this question.

	CURSO	ANO_LETIVO	PERIODO	NUM_HORAS
1	275	2002/2003	2S	27
2	275	2003/2004	2S	24

Figure 3.17: Question 5. - Answer.

3.5.0.3 Execution Plans

The following picture shows the execution plan for the above query.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			443	50
HASH		GROUP BY	443	50
HASH JOIN			443	49
Access Predicates				
ZUCS.CODIGO=ZTIPOSAULA.CODIGO				
NESTED LOOPS			443	49
NESTED LOOPS				
STATISTICS COLLECTOR				
TABLE ACCESS	ZTIPOSAULA	FULL	443	36
Filter Predicates				
AND				
ZTIPOSAULA.TIPO='OT'				
OR				
ZTIPOSAULA.ANO_LETIVO='2002/2003'				
ZTIPOSAULA.ANO_LETIVO='2003/2004'				
INDEX	ZUCS_PK	UNIQUE SCAN		
Access Predicates				
ZUCS.CODIGO=ZTIPOSAULA.CODIGO				
TABLE ACCESS	ZUCS	BY INDEX ROWID	1	13
TABLE ACCESS	ZUCS	FULL	5396	13

Figure 3.18: Question 5. - Z Execution Plan.

As we can see, the operation total cost was 50.

3.5.0.4 Execution Time

The execution time was 0,027 sec.

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

3.5.1.1 Execution Plans

Before running the query presented in question 5, we created the following B-tree indexes.

```sql
CREATE INDEX ZTIPOSAULA_TIPO ON ZTIPOSAULA(TIPO);
CREATE INDEX ZTIPOSAULA_ANOLETIVO ON ZTIPOSAULA(ANO_LETIVO);
```

Note 7: By default, Oracle Database creates B-tree indexes.

The following picture shows the execution plan for the query.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			28	19
HASH		GROUP BY	28	19
HASH JOIN			28	18
Access Predicates				
ZUCS.CODIGO=ZTIPOSAULA.CODIGO				
NESTED LOOPS			28	18
NESTED LOOPS				
STATISTICS COLLECTOR				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID BATCHED	28	5
Filter Predicates				
OR				
ZTIPOSAULA.ANO_LETIVO='2002/2003'				
ZTIPOSAULA.ANO_LETIVO='2003/2004'				
INDEX	ZTIPOSAULA_TIPO	RANGE SCAN	193	1
Access Predicates				
ZTIPOSAULA.TIPO='OT'				
INDEX	ZUCS_PK	UNIQUE SCAN		
Access Predicates				
ZUCS.CODIGO=ZTIPOSAULA.CODIGO				
TABLE ACCESS	ZUCS	BY INDEX ROWID	1	13
TABLE ACCESS	ZUCS	FULL	5396	13

Figure 3.19: Question 5a. - Z Execution Plan.

As we can see, by using a index in type column, the operation total cost decreased from 50 to 19.

Note 8: By default, Oracle Database chooses if it should use the indexes or not in order to minimize costs. In this case, it choose not to use the index 'ZTIPOSAULA_ANOLETIVO'. If we force to use both indexes, using the hint '/*+INDEX(ZTIPOSAULA ZTIPOSAULA_ANOLETIVO)*/' immediately after the select instruction, we verified that the cost goes up to 75.

To get the size of the index used in the column type we used the following query.

```sql
select sum(bytes)/1024/1024 as "Index Size (MB)" from user_segments where segment_name='
    ZTIPOSAULA_TIPO';
```

As a result of that query, we obtained the following size to the 'ZTIPOSAULA_TIPO' index.

	Index Size (MB)
1	0,375

Figure 3.20: Index Size Query - Answer.

To get the size of the index used in the column academic year we used the following query.

```sql
select sum(bytes)/1024/1024 as "Index Size (MB)" from user_segments where segment_name='
    ZTIPOSAULA_ANOLETIVO';
```

As a result of that query, we obtained the following size to the 'ZTIPOSAULA_ANOLETIVO' index.

	Index Size (MB)
1	0,5625

Figure 3.21: Index Size Query - Answer.

We can see that the index size created in the academic year column is greater than the one created in the type column. This could be the reason so that the 'ZTIPOSAULA_ANOLETIVO' index is not used by the oracle optimizer.

3.5.1.2 Execution Time

The execution time was 0,024 sec.

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

3.5.2.1 Execution Plans

Before running the query presented in question 5, we created the following bitmap indexes.

```
CREATE BITMAP INDEX BM_ZTIPOSAULA_TIPO ON ZTIPOSAULA(TIPO);
CREATE BITMAP INDEX BM_ZTIPOSAULA_ANOLETIVO ON ZTIPOSAULA(ANO_LETIVO);
```

The following picture shows the execution plan for the query.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			28	22
HASH		GROUP BY	28	22
HASH JOIN			28	21
Access Predicates				
ZUCS.CODIGO=ZTIPOSAULA.CODIGO				
NESTED LOOPS			28	21
NESTED LOOPS				
STATISTICS COLLECTOR				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID BATCHED	28	8
BITMAP CONVERSION		TO ROWIDS		
BITMAP AND				
BITMAP INDEX	BM_ZTIPOSAULA_TIPO	SINGLE VALUE		
Access Predicates				
ZTIPOSAULA.TIPO='OT'				
BITMAP OR				
BITMAP INDE	BM_ZTIPOSAULA_ANOLETIVO	SINGLE VALUE		
Access Predicates				
ZTIPOSAULA.ANO_LETIVO='2002/2003'				
BITMAP INDE	BM_ZTIPOSAULA_ANOLETIVO	SINGLE VALUE		
Access Predicates				
ZTIPOSAULA.ANO_LETIVO='2003/2004'				
INDEX	ZUCS_PK	UNIQUE SCAN		
Access Predicates				
ZUCS.CODIGO=ZTIPOSAULA.CODIGO				
TABLE ACCESS	ZUCS	BY INDEX ROWID	1	13
TABLE ACCESS	ZUCS	FULL	5396	13

Figure 3.22: Question 5b. - Z Execution Plan.

As we can see, by using bitmap indexes in the mentioned columns in the question, the total cost decreased from 50 to 22.

To get the index size of the type column we used the following query.

```sql
select sum(bytes)/1024/1024 as "Index Size (MB)" from user_segments
where segment_name='BM_ZTIPOSAULA_TIPO';
```

As a result of that query, we obtained the following size to the 'BM_ZTIPOSAULA_TIPO' index.

	Index Size (MB)
1	0,0625

Figure 3.23: Index Size Query - Answer.

To get the index size of the academic year column we used the following query.

```sql
select sum(bytes)/1024/1024 as "Index Size (MB)" from user_segments where segment_name='
    BM_ZTIPOSAULA_ANOLETIVO';
```

As a result of that query, we obtained the following size to the 'BM_ZTIPOSAULA_ANOLETIVO' index.

	Index Size (MB)
1	0,0625

Figure 3.24: Index Size Query - Answer.

The size of each index turned out to be the same and smaller than in the previous ones.

3.5.2.2 Execution Time

The execution time was 0,024 sec.

3.6 Question 6

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

Note 9: For this question, we found ourselves trying to understand which of the following 3 interpretations was the correct one for this question:

- A. Get the UCs with classes from all types.

- B. Get the courses with classes from all types, i.e. the UCs belonging to a course may not have classes from all types, but all the classes, for all the UCs of that course, would have (or not) the overall class type representatives.
- C. Get the name(s) of the course(s) with at least 1 UC with all the class types.

We've decided that the interpretation B. was the one desired, so we moved one with that one in what respects to the analysis of the answer and the execution plan and time. However, before proceeding with that option, the following two SQL queries represent the ones that would retrieve the results for the interpretations A. and C.

```
/*A*/
select codigo
from (select codigo, tipo
      from xtiposaula
      group by codigo, tipo)
group by codigo
having count(codigo) = (select count(*)
                        from (select distinct tipo
                              from xtiposaula));
```

```
/*C*/
create view aux_codigo as
select codigo
from (select codigo, tipo
      from xtiposaula
      group by codigo, tipo)
group by codigo
having count(codigo) = (select count(*)
                        from (select distinct tipo
                              from xtiposaula));

select distinct curso
from aux_codigo join xucs on xucs.codigo = aux_codigo.codigo;
```

3.6.1 SQL query

The following SQL query, for the interpretation B., selects the 'curso' field from a sub-query that gets the list of courses and the corresponding class types, grouping then by course and having as constraint the fact that the course carnality (in this case the number of different class types) should be the same as the number of different course types (5) obtained by a nested sub-query.

```
/*B*/
select curso
from (select curso, tipo
      from xtiposaula join xucs on xtiposaula.codigo = xucs.codigo
      group by curso, tipo)
group by curso
having count(curso) = (select count(*)
                       from (select distinct tipo
                             from xtiposaula));
```

3.6.2 Answer

The answer for this question is the one that follows:

	CURSO
1	9461
2	4495
3	9508
4	2021

Figure 3.25: Question 6. - Answer.

3.6.3 Execution Plans

Considering the indexes created until now, we didn't find the need to create a new one for this question in particular.

That said, the following three pictures represent the three different execution plans for the 'X', 'Y' and 'Z' prefixed tables.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	51
FILTER				
Filter Predicates				
COUNT($vm_col_1)= (SELECT COUNT(*) FROM (SELECT DISTINCT TIPO TIPO FROM XTIPOSAULA XTIPOSAULA) from$_subquery$_005)				
HASH		GROUP BY	2	51
VIEW	SYS.VM_NWVW_0		404	51
HASH		GROUP BY	404	51
HASH JOIN			21019	49
Access Predicates				
XTIPOSAULA.CODIGO=XUCS.CODIGO				
TABLE ACCESS	XUCS	FULL	5396	13
TABLE ACCESS	XTIPOSAULA	FULL	21019	36
SORT		AGGREGATE	1	
VIEW			5	37
SORT		UNIQUE	5	37
TABLE ACCESS	XTIPOSAULA	FULL	21019	36

Figure 3.26: Question 6 - X Execution Plan.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	51
FILTER				
Filter Predicates				
COUNT($vm_col_1)= (SELECT COUNT(*) FROM (SELECT DISTINCT TIPO TIPO FROM YTIPOSAULA YTIPOSAULA) from$_subquery$_005)				
HASH		GROUP BY	2	51
VIEW	SYS.VM_NWVW_0		404	51
HASH		GROUP BY	404	51
HASH JOIN			21019	49
Access Predicates				
YTIPOSAULA.CODIGO=YUCS.CODIGO				
TABLE ACCESS	YUCS	FULL	5396	13
TABLE ACCESS	YTIPOSAULA	FULL	21019	36
SORT		AGGREGATE	1	
VIEW			5	37
SORT		UNIQUE	5	37
TABLE ACCESS	YTIPOSAULA	FULL	21019	36

Figure 3.27: Question 6 - Y Execution Plan.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	51
FILTER				
Filter Predicates				
COUNT($vm_col_1)= (SELECT COUNT(*) FROM (SELECT DISTINCT TIPO TIPO FROM ZTIPOSAULA ZTIPOSAULA) from$_subquery$_005)				
HASH		GROUP BY	2	51
VIEW	SYS.VM_NWVW_0		404	51
HASH		GROUP BY	404	51
HASH JOIN			21019	49
Access Predicates				
ZTIPOSAULA.CODIGO=ZUCS.CODIGO				
TABLE ACCESS	ZUCS	FULL	5396	13
TABLE ACCESS	ZTIPOSAULA	FULL	21019	36
SORT		AGGREGATE	1	
VIEW			5	37
SORT		UNIQUE	5	37
TABLE ACCESS	ZTIPOSAULA	FULL	21019	36

Figure 3.28: Question 6 - Z Execution Plan.

As we can see, the Hash Join was used for the three cases, and the cost was also the same for all: 51.

3.6.4 Execution Time

Regarding the execution times, they were 0,032, 0,030 and 0,029 sec. for the 'X', 'Y' and 'Z' prefixed tables, respectively.