Master in Informatics and Computing Engineering

Query optimization - Teaching Service

Database Technologies

Academic Year - 2021/2022
up201800175, Juliane Marubayashi
up201806791, Ricardo Carvalho
up201806534, Rodrigo Reis
Oporto, April 2022

Index

1 Summary

The goal of the first project of the curricular unit of Tecnologias de Base de Dados is to develop a project in which we test several queries in different environments:

- Environment X - no indexes and no integrity constraints;
- Environment Y - with the standard integrity constraints (primary keys and foreign keys);
- Environment Z - with the standard integrity constraints and with extra indexes.

The project consists in a database related to teaching service, and for each query we will analyze the execution time in each environment, as well as the query result, the execution plan and some analysis comparing each environment and execution time.

2 Indexes

2.1 Query 1

```sql
CREATE INDEX TIPOSAULA_IDX ON ZTIPOSAULA(CODIGO, ANO_LETIVO, PERIODO);
```

```sql
CREATE INDEX UCS_IDX ON ZUCS(DESIGNACAO, CURSO);
```

2.2 Query 2

```sql
CREATE INDEX UCS_IDX ON ZUCS(CODIGO, CURSO);
```

2.3 Query 3

```sql
CREATE INDEX TIPOSAULA_IDX ON ZTIPOSAULA(ANO_LETIVO);
```

2.4 Query 4

```sql
CREATE INDEX TIPOSAULA_IDX ON ZTIPOSAULA(ANO_LETIVO);
```

2.5 Query 5

In this query it was tested two types of indexes (i.e b-trees, bitmap).

Listing 1: b-tree index

```sql
CREATE INDEX TIPOSAULA_IDX ON ZTIPOSAULA(ANO_LETIVO, TIPO);
```

Listing 2: bitmap index

```sql
CREATE BITMAP INDEX TIPOSAULA_IDX ON ZTIPOSAULA(ANO_LETIVO, TIPO);
```

2.6 Query 6

```sql
CREATE INDEX TIPOSAULA_IDX ON ZTIPOSAULA(CODIGO, TIPO);
```

```sql
CREATE INDEX UCS_IDX ON ZUCS(CODIGO, CURSO);
```

3 Questions

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.

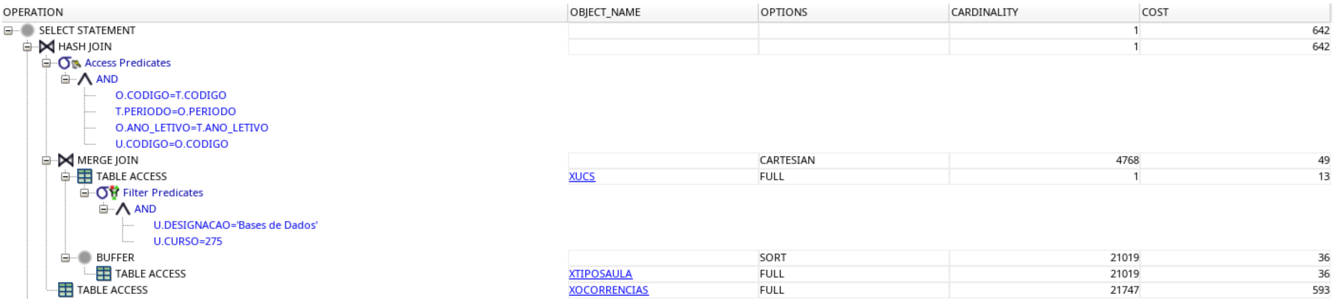
3.1.1 SQL query

```sql
SELECT u.codigo, u.designacao, o.ano_letivo, o.inscritos, t.tipo, t.turnos
FROM xucs u
JOIN xocorrencias o on u.codigo=o.codigo
JOIN xtiposaula t on o.codigo=t.codigo and t.periodo= o.periodo and
    o.ano_letivo=t.ano_letivo
WHERE u.designacao='Bases de Dados' and u.curso=275;
```

3.1.2 Execution time

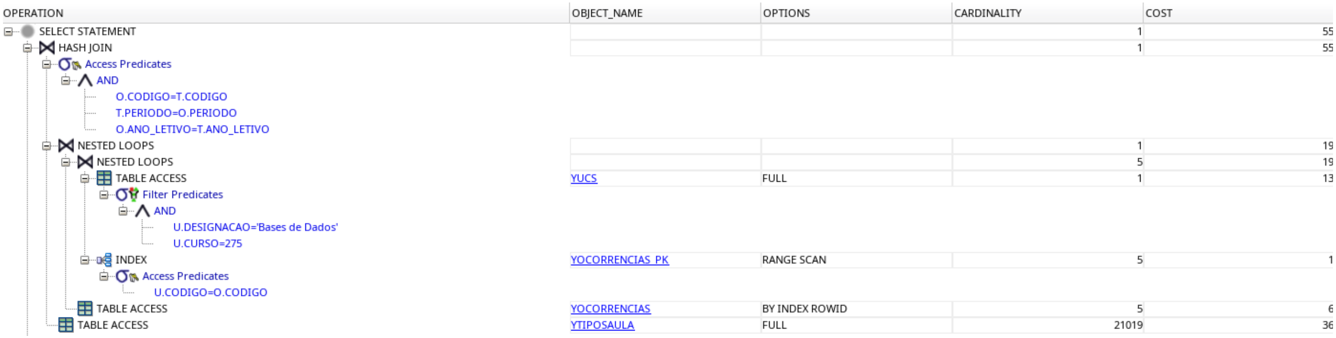
Table X	Table Y	Table Z
0.053	0.035	0.022

3.1.3 Execution plan



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	642
HASH JOIN			1	642
Access Predicates				
AND				
O.CODIGO=T.CODIGO				
T.PERIODO=O.PERIODO				
O.ANO_LETIVO=T.ANO_LETIVO				
U.CODIGO=O.CODIGO				
MERGE JOIN		CARTESIAN	4768	49
TABLE ACCESS	XUCS	FULL	1	13
Filter Predicates				
AND				
U.DESIGNACAO='Bases de Dados'				
U.CURSO=275				
BUFFER		SORT	21019	36
TABLE ACCESS	XTIPOSAULA	FULL	21019	36
TABLE ACCESS	XOCORRENCIAS	FULL	21747	593

Figure 3.1: Execution plan of query 1 for table X



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	55
HASH JOIN			1	55
Access Predicates				
AND				
O.CODIGO=T.CODIGO				
T.PERIODO=O.PERIODO				
O.ANO_LETIVO=T.ANO_LETIVO				
NESTED LOOPS			1	19
NESTED LOOPS			5	19
TABLE ACCESS	YUCS	FULL	1	13
Filter Predicates				
AND				
U.DESIGNACAO='Bases de Dados'				
U.CURSO=275				
INDEX	YOCORRENCIAS_PK	RANGE SCAN	5	1
Access Predicates				
U.CODIGO=O.CODIGO				
TABLE ACCESS	YOCORRENCIAS	BY INDEX ROWID	5	6
TABLE ACCESS	YTIPOSAULA	FULL	21019	36

Figure 3.2: Execution plan of query 1 for table Y

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	10
HASH JOIN			1	10
Access Predicates				
AND				
O.CODIGO=T.CODIGO				
T.PERIODO=O.PERIODO				
O.ANO_LETIVO=T.ANO_LETIVO				
NESTED LOOPS			1	10
NESTED LOOPS			1	10
STATISTICS COLLECTOR				
HASH JOIN			1	8
Access Predicates				
U.CODIGO=O.CODIGO				
NESTED LOOPS			1	8
STATISTICS COLLECTOR				
TABLE ACCESS	ZUCS	BY INDEX ROWID BATCHED	1	2
INDEX	UCS_IDX	RANGE SCAN	1	1
Access Predicates				
AND				
U.DESIGNACAO='Bases de Dados'				
U.CURSO=275				
TABLE ACCESS	ZOCORRENCIAS	BY INDEX ROWID BATCHED	5	6
INDEX	ZOCORRENCIAS_PK	RANGE SCAN	5	1
Access Predicates				
U.CODIGO=O.CODIGO				
TABLE ACCESS	ZOCORRENCIAS	FULL	5	6
INDEX	TIPOSAULA_IDX	RANGE SCAN	1	1
Access Predicates				
AND				
O.CODIGO=T.CODIGO				
O.ANO_LETIVO=T.ANO_LETIVO				
T.PERIODO=O.PERIODO				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID	1	2
TABLE ACCESS	ZTIPOSAULA	FULL	1	2

Figure 3.3: Execution plan of query 1 for table Z

3.1.4 Query Result

	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 3.4: Query 1 result

3.1.5 Analysis

The X tables performed `FULL TABLE ACCESSES` in totality, which condemned the performance of the operation. The Y tables still performed `FULL TABLE ACCESSES`, but adding primary and foreign keys allowed the optimizer to apply `RANGE SCAN`, since primary keys are indexed by default and replaced the `FULL TABLE ACCESS` in the `YOCORRENCIAS` by an `ACCESS BY INDEX ROWID`.

In Z tables, the addition of indexes to some foreign keys, which are not indexed by default, and to the columns referenced in the `WHERE` statement (i.e `DESIGNACAO` and `CURSO`), allowed the Oracle Optimizer to use `RANGE SCAN` and perform `INDEX ACCESS` instead of `TABLE ACCESS`.

3.2 Question 2

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

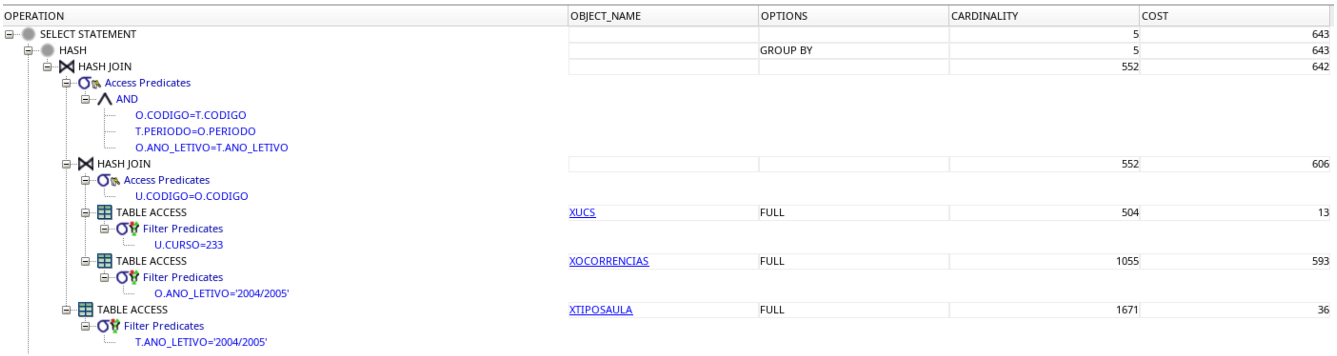
3.2.1 SQL query

```
SELECT t.tipo, SUM(t.horas_turno * COALESCE(t.n_aulas,1) * COALESCE(t.turnos,1))
    as horas
FROM xucs u
JOIN xocorrencias o on u.codigo=o.codigo
JOIN xtiposaula t on o.codigo=t.codigo and t.periodo=o.periodo and
    o.ano_letivo=t.ano_letivo
WHERE u.curso=233 and o.ano_letivo='2004/2005'
GROUP BY t.tipo;
```

3.2.2 Execution time

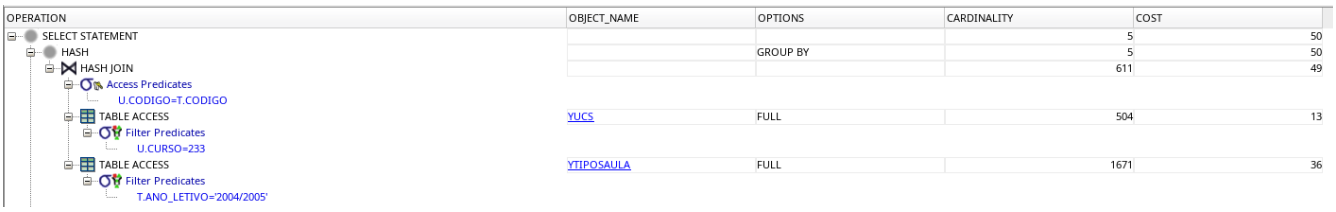
Table X	Table Y	Table Z
0.093	0.081	0.030

3.2.3 Execution plan



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	643
HASH		GROUP BY	5	643
HASH JOIN			552	642
Access Predicates				
AND				
O.CODIGO=T.CODIGO				
T.PERIODO=O.PERIODO				
O.ANO_LETIVO=T.ANO_LETIVO				
HASH JOIN			552	606
Access Predicates				
U.CODIGO=O.CODIGO				
TABLE ACCESS	XUCS	FULL	504	13
Filter Predicates				
U.CURSO=233				
TABLE ACCESS	XOCORRENCIAS	FULL	1055	593
Filter Predicates				
O.ANO_LETIVO='2004/2005'				
TABLE ACCESS	XTIPOSAULA	FULL	1671	36
Filter Predicates				
T.ANO_LETIVO='2004/2005'				

Figure 3.5: Execution plan of query 2 for table X



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	50
HASH		GROUP BY	5	50
HASH JOIN			611	49
Access Predicates				
U.CODIGO=T.CODIGO				
TABLE ACCESS	YUCS	FULL	504	13
Filter Predicates				
U.CURSO=233				
TABLE ACCESS	YTIPOSAULA	FULL	1671	36
Filter Predicates				
T.ANO_LETIVO='2004/2005'				

Figure 3.6: Execution plan of query 2 for table Y

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	41
HASH		GROUP BY	5	41
HASH JOIN			57	40
Access Predicates				
U.CODIGO=T.CODIGO				
NESTED LOOPS			57	40
NESTED LOOPS				
STATISTICS COLLECTOR				
INDEX	UCS_IDX	FAST FULL SCAN	47	7
Filter Predicates				
U.CURSO=233				
INDEX	CODIGO_ANOLETIVO_IDX	RANGE SCAN	1671	5
Access Predicates				
AND				
U.CODIGO=T.CODIGO				
T.ANO_LETIVO='2004/2005'				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID	1	33
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID BATCHED	1671	33
INDEX	ANOLETIVO_IDX	RANGE SCAN	1671	5
Access Predicates				
T.ANO_LETIVO='2004/2005'				

Figure 3.7: Execution plan of query 2 for table Z

3.2.4 Query Result

	TIPO	HORAS
1	P	747.5
2	TP	967.5
3	T	580

Figure 3.8: Query 2 result

3.2.5 Analysis

We tested this query with and without the use of `COALESCE`, giving different results in each case. We chose to use COALESCE because there were null values in the columns `N_AULAS` and `TURNOS`. As we can see, the environment Y performs better than environment X due to the fact that the columns that represent primary keys are indexed.
With the creation of an extra index in the column `CURSOS`, that was neither primary key nor foreign key, and was a column used in the `WHERE` clause, the environment Z performed better than environment Y and X.
We can also see that the cost and cardinality for environment Z is much better than the cost and cardinality for the other two environments.

3.3 Question 3

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

3.3.1 SQL query for point a)

```
SELECT UNIQUE(u.codigo)
FROM xucs u
JOIN xocorrencias o ON u.codigo=o.codigo AND o.ano_letivo='2003/2004'
WHERE u.codigo NOT IN (
   SELECT UNIQUE(codigo)
   FROM xtiposaula
   JOIN xdsd ON xdsd.id=xtiposaula.id
   WHERE ano_letivo='2003/2004'
);
```

3.3.2 SQL query for point b)

```
CREATE VIEW has_service AS
SELECT UNIQUE(t.codigo)
FROM xtiposaula t
JOIN xdsd d ON d.id=t.id
WHERE t.ano_letivo='2003/2004';

SELECT UNIQUE(u.codigo)
FROM xucs u
JOIN xocorrencias o ON o.codigo=u.codigo
LEFT OUTER JOIN has_service hs ON u.codigo=hs.codigo
WHERE hs.codigo IS NULL AND o.ano_letivo='2003/2004';

DROP VIEW has_service;
```

3.3.3 Execution time

Table X	Table Y	Table Z
0.124	0.099	0.033

Table 3.1: Execution time for point a)

Table X	Table Y	Table Z
0.063	0.076	0.055

Table 3.2: Execution time for point b)

3.3.4 Execution plan for point a)

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			941	670
HASH		UNIQUE	941	670
HASH JOIN		RIGHT ANTI	941	669
Access Predicates				
U.CODIGO=CODIGO				
VIEW	SYS.VW_NSO_1		1588	63
HASH JOIN		SEMI	1588	63
Access Predicates				
XDSD.ID=XTIPOSAULA.ID				
TABLE ACCESS	XTIPOSAULA	FULL	1588	36
Filter Predicates				
XTIPOSAULA.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XDSD	FULL	27385	27
HASH JOIN		RIGHT SEMI	941	606
Access Predicates				
U.CODIGO=O.CODIGO				
TABLE ACCESS	XOCORRENCIAS	FULL	1028	593
Filter Predicates				
O.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XUCS	FULL	5396	13

Figure 3.9: Execution plan of query 3.a) for table X

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	86
HASH		UNIQUE	1	86
HASH JOIN		ANTI	10	85
Access Predicates				
O.CODIGO=CODIGO				
INDEX	YOCORRENCIAS_PK	FAST FULL SCAN	1028	27
Filter Predicates				
O.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

Figure 3.10: Execution plan of query 3.a) for table Y

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	81
HASH		UNIQUE	1	81
HASH JOIN		ANTI	10	80
Access Predicates				
O.CODIGO=CODIGO				
INDEX	ZOCORRENCIAS_PK	FAST FULL SCAN	1028	27
Filter Predicates				
O.ANO_LETIVO='2003/2004'				
VIEW	SYS.VW_NSO_1		1588	53
HASH JOIN		SEMI	1588	53
Access Predicates				
ZDSD.ID=ZTIPOSAULA.ID				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID BATCHED	1588	31
INDEX	TIPOSAULA_IDX	RANGE SCAN	1588	5
Access Predicates				
ZTIPOSAULA.ANO_LETIVO='2003/2004'				
INDEX	ZDSD_PK	FAST FULL SCAN	27385	22

Figure 3.11: Execution plan of query 3.a) for table Z

3.3.5 Execution plan for point b)

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			941	671
HASH		UNIQUE	941	671
HASH JOIN		RIGHT ANTI	941	670
Access Predicates				
U.CODIGO=HS.CODIGO				
VIEW	HAS_SERVICE		1322	64
HASH		UNIQUE	1322	64
HASH JOIN		SEMI	1588	63
Access Predicates				
D.ID=T.ID				
TABLE ACCESS	XTIPOSAULA	FULL	1588	36
Filter Predicates				
T.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XDSD	FULL	27385	27
HASH JOIN		RIGHT SEMI	941	606
Access Predicates				
O.CODIGO=U.CODIGO				
TABLE ACCESS	XOCORRENCIAS	FULL	1028	593
Filter Predicates				
O.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XUCS	FULL	5396	13

Figure 3.12: Execution plan of query 3.b) for table X

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	87
HASH		UNIQUE	1	87
HASH JOIN		ANTI	10	86
Access Predicates				
O.CODIGO=HS.CODIGO				
INDEX	YOCORRENCIAS_PK	FAST FULL SCAN	1028	27
Filter Predicates				
O.ANO_LETIVO='2003/2004'				
VIEW	HAS_SERVICE		1322	59
HASH		UNIQUE	1322	59
HASH JOIN		SEMI	1588	58
Access Predicates				
D.ID=T.ID				
TABLE ACCESS	YTIPOSAULA	FULL	1588	36
Filter Predicates				
T.ANO_LETIVO='2003/2004'				
INDEX	YDSD_PK	FAST FULL SCAN	27385	22

Figure 3.13: Execution plan of query 3.b) for table Y

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	82
HASH		UNIQUE	1	82
HASH JOIN		ANTI	10	81
Access Predicates				
O.CODIGO=HS.CODIGO				
INDEX	ZOCORRENCIAS_PK	FAST FULL SCAN	1028	27
Filter Predicates				
O.ANO_LETIVO='2003/2004'				
VIEW	HAS_SERVICE		1322	54
HASH		UNIQUE	1322	54
HASH JOIN		SEMI	1588	53
Access Predicates				
D.ID=T.ID				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID BATCHED	1588	31
INDEX	TIPOSAULA_IDX	RANGE SCAN	1588	5
Access Predicates				
T.ANO_LETIVO='2003/2004'				
INDEX	ZDSD_PK	FAST FULL SCAN	27385	22

Figure 3.14: Execution plan of query 3.b) for table Z

3.3.6 Query Result

	CODIGO		CODIGO		CODIGO		CODIGO
1	MEMT1000	36	MEAM1312	71	MEMT106	106	EQ411
2	MEMT100	37	MEMT135	72	EC5287	107	MDI1207
3	EQ418	38	MPPAU1113	73	MDI1106	108	MDI1209
4	MTM108	39	EIC3209	74	MPPAU2219	109	MEB204
5	MEMT131	40	MEM179	75	MPFCA105	110	MMCCE1220
6	MEEC1053	41	MEA215	76	MPFCA107	111	EEC2207
7	MEM157	42	MEA414	77	MPFCA201	112	EIC4224
8	MEM181	43	MDI1107	78	MPFCA202	113	EIC5129
9	MDI1205	44	MDI1208	79	MPFCA206	114	CI019
10	MPFCA103	45	MDI1108	80	EIC5125	115	CI002
11	MPFCA204	46	MPPAU2217	81	EIC5126	116	CI025
12	EIC4220	47	MPFCA101	82	CI038	117	CI037
13	EIC4221	48	MPFCA205	83	MEB205	118	MEB105
14	EIC4222	49	EIC5127	84	EQ407	119	EQ308
15	CI027	50	MTM115	85	MDI1204	120	MPPAU2218
16	MEMT107	51	EMM528	86	MDI1100	121	MPPAU1112
17	MEMT102	52	MTM110	87	MFAMF1108	122	EEC5272
18	MEAM1310	53	MEAM5000	88	MPPAU2220	123	MEM5000
19	MPPAU2215	54	EC5280	89	MPPAU2216	124	MEM158
20	MEM187	55	MPFCA100	90	MEM163	125	MEM182
21	MEM189	56	MPFCA104	91	MEM175	126	MEM183
22	MEA219	57	MPFCA200	92	MEM184	127	MEA216
23	EI1107	58	EC5200	93	MEM188	128	MEA319
24	MPFCA106	59	EEC5022	94	MEM191	129	MEST210
25	EIC4225	60	EIC5124	95	MEA415	130	MEMT110
26	CI014	61	CI020	96	EIC4223	131	MDI1206
27	CI018	62	CI016	97	EIC5122	132	MEMT120
28	CI007	63	CI011	98	EIC5123	133	MPPAU1115
29	CI017	64	MTM114	99	CI023	134	MPFCA102
30	CI008	65	MPPAU1114	100	CI009	135	MPFCA203
31	MEA412	66	MEM180	101	MEM1205	136	CI003
32	MTM111	67	MVC1211	102	GEI512	137	CI004
33	MDI1105	68	MEA112	103	MEMT105	138	CI013
34	MDI1103	69	MEA217	104	MTM104		
35	MEMT2000	70	MEA320	105	MEAM1314		

Figure 3.15: Query 3 result

The result is the same for the approaches taken either in point a) and b).

3.3.7 Analysis

- Point A

Let's first analyse the table X. The high cost of this query is a consequence of `XOCORRENCIAS FULL TABLE ACCESS`, which costs 670. Although the table `XOCORRENCIAS` has a similar number of rows when compared to `XTIPOSAULA` there's a high cost difference between the two. This cost variance, however, is well explained by the number of blocks in each table: `XOCORRENCIAS` contains 2181, while `XTIPOSAULA` contains only 126.

In table Y, by adding primary and foreign keys, the optimizer uses the `FAST FULL SCAN`, which behaves like a `FULL TABLE SCAN`, but it's faster once allows multiblock reading. Yet, the conditions to use `FAST FULL SCAN` matches our scenario: the index contains all the query columns and this column is not null. This is the main optimization that contributes to a query total cost of 86 over 670.

The table Z, slightly improves the query performance when compared to the tables Y, due to `RANGE SCANS`, achieved by adding indexes to the columns in the `WHERE` clause.

- Point B

The point B has a similar cost values when compared to the point A. In both points the optimizer performs an anti-join to obtain the courses that did not get a service.

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

```
CREATE VIEW docente_horas AS
SELECT doc.nr, SUM(d.horas*d.fator) as sum_horas, t.tipo, doc.nome
FROM xdocentes doc
JOIN xdsd d ON doc.nr=d.nr
JOIN xtiposaula t ON d.id=t.id
WHERE t.ano_letivo='2003/2004'
GROUP BY doc.nome,doc.nr,t.tipo;


CREATE VIEW max_horas_tipo AS
SELECT MAX(sum_horas) as sum_horas, dh.tipo
FROM docente_horas dh
GROUP BY dh.tipo;

SELECT dh.nr,dh.nome,dh.tipo,mht.sum_horas
FROM max_horas_tipo mht
JOIN docente_horas dh ON dh.sum_horas=mht.sum_horas AND dh.tipo=mht.tipo;

DROP VIEW docente_horas;
DROP VIEW max_horas_tipo;
```

3.4.2 Execution time

Table X	Table Y	Table Z
0.311	0.397	0.088

3.4.3 Execution plan

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			46	139
HASH JOIN			46	139
Access Predicates				
AND				
DH.SUM_HORAS=MHT.SUM_HORAS				
DH.TIPO=MHT.TIPO				
VIEW	MAX_HORAS_TIPO		5	69
HASH		GROUP BY	5	69
VIEW	DOCENTE_HORAS		2570	69
HASH		GROUP BY	2570	69
HASH JOIN			2570	68
Access Predicates				
DOC.NR=D.NR				
TABLE ACCESS	XDOCENTES	FULL	939	5
HASH JOIN			2570	63
Access Predicates				
D.ID=T.ID				
TABLE ACCESS	XTIPOSAULA	FULL	1588	36
Filter Predicates				
T.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XDSD	FULL	27385	27
VIEW	DOCENTE_HORAS		2570	69
HASH		GROUP BY	2570	69
HASH JOIN			2570	68
Access Predicates				
DOC.NR=D.NR				
TABLE ACCESS	XDOCENTES	FULL	939	5
HASH JOIN			2570	63
Access Predicates				
D.ID=T.ID				
TABLE ACCESS	XTIPOSAULA	FULL	1588	36
Filter Predicates				
T.ANO_LETIVO='2003/2004'				
TABLE ACCESS	XDSD	FULL	27385	27

Figure 3.16: Execution plan of query 4 for table X

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			46	139
HASH JOIN			46	139
Access Predicates				
AND				
DH.SUM_HORAS=MHT.SUM_HORAS				
DH.TIPO=MHT.TIPO				
VIEW	MAX_HORAS_TIPO		5	69
HASH		GROUP BY	5	69
VIEW	DOCENTE_HORAS		2570	69
HASH		GROUP BY	2570	69
HASH JOIN			2570	68
Access Predicates				
DOC.NR=D.NR				
TABLE ACCESS	YDOCENTES	FULL	939	5
HASH JOIN			2570	63
Access Predicates				
D.ID=T.ID				
TABLE ACCESS	YTIPOSAULA	FULL	1588	36
Filter Predicates				
T.ANO_LETIVO='2003/2004'				
TABLE ACCESS	YDSD	FULL	27385	27
VIEW	DOCENTE_HORAS		2570	69
HASH		GROUP BY	2570	69
HASH JOIN			2570	68
Access Predicates				
DOC.NR=D.NR				
TABLE ACCESS	YDOCENTES	FULL	939	5
HASH JOIN			2570	63
Access Predicates				
D.ID=T.ID				
TABLE ACCESS	YTIPOSAULA	FULL	1588	36
Filter Predicates				
T.ANO_LETIVO='2003/2004'				
TABLE ACCESS	YDSD	FULL	27385	27

Figure 3.17: Execution plan of query 4 for table Y

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			46	127
HASH JOIN			46	127
Access Predicates				
AND				
DH.SUM_HORAS=MHT.SUM_HORAS				
DH.TIPO=MHT.TIPO				
VIEW	MAX_HORAS_TIPO		5	63
HASH		GROUP BY	5	63
VIEW	DOCENTE_HORAS		2570	63
HASH		GROUP BY	2570	63
HASH JOIN			2570	62
Access Predicates				
DOC.NR=D.NR				
INDEX	DOC_IDX	FAST FULL SCAN	939	4
HASH JOIN			2570	58
Access Predicates				
D.ID=T.ID				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID BATCHED	1588	31
INDEX	TIPOSAULA_IDX	RANGE SCAN	1588	5
Access Predicates				
T.ANO_LETIVO='2003/2004'				
TABLE ACCESS	ZDSD	FULL	27385	27
VIEW	DOCENTE_HORAS		2570	63
HASH		GROUP BY	2570	63
HASH JOIN			2570	62
Access Predicates				
DOC.NR=D.NR				
INDEX	DOC_IDX	FAST FULL SCAN	939	4
HASH JOIN			2570	58
Access Predicates				
D.ID=T.ID				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID BATCHED	1588	31
INDEX	TIPOSAULA_IDX	RANGE SCAN	1588	5
Access Predicates				
T.ANO_LETIVO='2003/2004'				
TABLE ACCESS	ZDSD	FULL	27385	27

Figure 3.18: Execution plan of query 4 for table Z

3.4.4 Query Result

	NR	NOME	TIPO	SUM_HORAS
1	208187	António Almerindo Pinheiro Vieira	P	30
2	207638	Fernando Francisco Machado Veloso Gomes	T	30.67
3	249564	Cecília do Carmo Ferreira da Silva	TP	26
4	210006	João Carlos Pascoal de Faria	OT	3.5

Figure 3.19: Query 4 result

3.4.5 Analysis

In the execution plan of both environments (X and Y), we can see that the cost is exactly the same, because the indexes (i.e primary and foreign keys) doesn't contains all the columns in the select statement, which makes the use of indexes an inefficient choice. We can see that the execution time in Y is worse than in X, which proves that the constraints don't do us much use in this query. The addition of an index on the table ZTIPOSAULA and ZDSD improved the performance by performing `RANGE SCAN`, since it was column used in the `WHERE` clause.

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.

3.5.1 SQL query

```
SELECT u.codigo, u.curso, o.ano_letivo, o.periodo, SUM(t.horas_turno *
    COALESCE(n_aulas,1) * COALESCE(t.turnos,1))
FROM zucs u
JOIN zocorrencias o on u.codigo=o.codigo
JOIN ztiposaula t on o.codigo=t.codigo and t.periodo= o.periodo and
    o.ano_letivo=t.ano_letivo
JOIN zdsd d on d.id=t.id
WHERE t.tipo='OT' AND (t.ano_letivo='2002/2003' OR t.ano_letivo='2003/2004')
GROUP BY u.curso, o.ano_letivo, o.periodo, u.codigo;
```

3.5.2 Execution time

Table Z (b-tree)	Table Z (bitmap)
0.037	0.033

The execution time between b-trees and bitmaps, doesn't represents a significant difference, thus we reach in a inconclusive result regarding which one of the queries is faster.

3.5.3 Execution plan

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			28	50
HASH		GROUP BY	28	50
HASH JOIN			28	49
Access Predicates				
U.CODIGO=T.CODIGO				
NESTED LOOPS			28	49
NESTED LOOPS				
STATISTICS COLLECTOR				
TABLE ACCESS	YTIPOSAULA	FULL	28	36
Filter Predicates				
AND				
T.TIPO='OT'				
OR				
T.ANO_LETIVO='2002/2003'				
T.ANO_LETIVO='2003/2004'				
INDEX	YUCS_PK	UNIQUE SCAN		
Access Predicates				
U.CODIGO=T.CODIGO				
TABLE ACCESS	YUCS	BY INDEX ROWID	1	13
TABLE ACCESS	YUCS	FULL	5396	13

Figure 3.20: Execution plan of query 5 for table Y

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			28	13
HASH		GROUP BY	28	13
HASH JOIN			28	12
Access Predicates				
U.CODIGO=T.CODIGO				
NESTED LOOPS			28	12
STATISTICS COLLECTOR				
INLIST ITERATOR				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID BATCHED	28	5
INDEX	TIPOSAULA_IDX	RANGE SCAN	28	3
Access Predicates				
AND				
OR				
T.ANO_LETIVO='2002/2003'				
T.ANO_LETIVO='2003/2004'				
T.TIPO='OT'				
INDEX	UCS_IDX	RANGE SCAN	1	7
Access Predicates				
U.CODIGO=T.CODIGO				
INDEX	UCS_IDX	FAST FULL SCAN	5396	7

Figure 3.21: Execution plan of query 5 for table Z with b-tree index

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			28	15
HASH		GROUP BY	28	15
HASH JOIN			28	14
Access Predicates				
U.CODIGO=T.CODIGO				
NESTED LOOPS			28	14
STATISTICS COLLECTOR				
INLIST ITERATOR				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID BATCHED	28	7
BITMAP CONVERSION		TO ROWIDS		
BITMAP INDEX	TIPOSAULA_IDX	SINGLE VALUE		
Access Predicates				
AND				
OR				
T.ANO_LETIVO='2002/2003'				
T.ANO_LETIVO='2003/2004'				
T.TIPO='OT'				
INDEX	UCS_IDX	RANGE SCAN	1	7
Access Predicates				
U.CODIGO=T.CODIGO				
INDEX	UCS_IDX	FAST FULL SCAN	5396	7

Figure 3.22: Execution plan of query 5 for table Z with bitmap index

3.5.4 Query Result

	CODIGO	CURSO	ANO_LETIVO	PERIODO	HORAS_TOTAIS
1	EIC5202	275	2002/2003	2S	27
2	EIC5202	275	2003/2004	2S	24

Figure 3.23: Query 5 result

3.5.5 Analysis

We are able to measure the number of MB for each index applying the following query:

```
SELECT SUM(bytes)/1024/1024 AS "Index Size (MB)"
FROM user_segments
WHERE segment_name='TIPOSAULA_IDX';
```

In the results presented, it was computed that the B-tree occupies 0.625 (MB), whereas the bitmap indexing occupies 0.0625 (MB). The storage size found in the bitmap index is justified by the small number of images and low *degree of cardinality* [1] of the columns ANO_LETIVO (19) and TIPO (5).

The query that uses the B-tree is a little more efficient than the Bitmap, but the B-trees occupies 10 times more space than the Bitmap indexing. Thus choosing the right indexing is a trade-off: it might be better for a critical system with hard deadlines [2] to use B-trees, while systems that without tight constraints might prefer to save space with a cost of loosing a little of performance.

The indexes were added to the columns TIPOS and ANO_LETIVO, since they were filtered in WHERE clause. Other columns in the GROUP BY statement already had indexes, since the majority is a primary key. The exception is the CURSO column, which is not a primary key, but adding this column as an index increased the query cost.

3.6 Question 6

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

3.6.1 SQL query

```
-- Calculates the number of types.
SELECT UNIQUE(tipo) FROM xtiposaula;

-- Get's the elements that have all the types.
SELECT COUNT(UNIQUE(t.tipo)) AS cursos,u.curso
FROM xucs u
JOIN xocorrencias o ON u.codigo=o.codigo
```

```
JOIN xtiposaula t ON o.codigo=t.codigo AND t.periodo= o.periodo AND
    o.ano_letivo=t.ano_letivo
GROUP BY u.curso
HAVING COUNT(UNIQUE(t.tipo))=5;
```

3.6.2 Execution time

Table X	Table Y	Table Z
0.055	0.039	0.03

The execution time between tables differs slightly, but this time variation probably could be seem better in larger tables.

3.6.3 Execution plan

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	643
HASH		GROUP BY	2	643
Filter Predicates				
COUNT($vm_col_1)=5				
VIEW	SYS.VM_NWVW_1		404	643
HASH		GROUP BY	404	643
HASH JOIN			21019	642
Access Predicates				
U.CODIGO=O.CODIGO				
TABLE ACCESS	XUCS	FULL	5396	13
HASH JOIN			21019	629
Access Predicates				
AND				
O.CODIGO=T.CODIGO				
T.PERIODO=O.PERIODO				
O.ANO_LETIVO=T.ANO_LETIVO				
TABLE ACCESS	XOCORRENCIAS	FULL	21747	593
TABLE ACCESS	XTIPOSAULA	FULL	21019	36

Figure 3.24: Execution plan of query 6 for table X

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	51
HASH		GROUP BY	2	51
Filter Predicates				
COUNT($vm_col_1)=5				
VIEW	SYS.VM_NWVW_1		404	51
HASH		GROUP BY	404	51
HASH JOIN			21019	49
Access Predicates				
U.CODIGO=T.CODIGO				
TABLE ACCESS	YUCS	FULL	5396	13
TABLE ACCESS	YTIPOSAULA	FULL	21019	36

Figure 3.25: Execution plan of query 6 for table Y

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	27
HASH		GROUP BY	2	27
Filter Predicates				
COUNT($vm_col_1)=5				
VIEW	SYS.VM_NWVW_1		404	27
HASH		GROUP BY	404	27
HASH JOIN			21019	25
Access Predicates				
U.CODIGO=T.CODIGO				
INDEX	UCS_IDX	FAST FULL SCAN	5396	7
INDEX	TIPOSAULA_IDX	FAST FULL SCAN	21019	18

Figure 3.26: Execution plan of query 6 for table Z

3.6.4 Query result

	CURSOS	CURSO
1	5	9461
2	5	4495
3	5	9508
4	5	2021

Figure 3.27: Query 6 result

3.6.5 Analysis

By making use of primary and foreign keys, the optimizer can notice that joining three tables is equivalent to merge YTIPOSAULA and YUCS, since the select doesn't contains any columns from YOCORRENCIAS, which improves the performance significantly. However, it means that the query could be made in the following way:

```
-- Calculates the number of types.
SELECT UNIQUE(tipo) FROM xtiposaula;

-- Get's the elements that have all the types.
SELECT COUNT(UNIQUE(t.tipo)) AS cursos, u.curso
FROM xucs u
JOIN xtiposaula t ON u.codigo=t.codigo
GROUP BY u.curso
HAVING COUNT(UNIQUE(t.tipo))=5;
```

This reduced version of the query allowed us to create the right indexes for Z tables. Firstly, without the analysis of the table Y, the following indexes were created:

```
CREATE INDEX TIPOSAULA_IDX ON ZTIPOSAULA(CODIGO, TIPO, PERIODO, ANO_LETIVO);
```

```
CREATE INDEX UCS_IDX ON ZUCS(CODIGO, CURSO);
```

Since the CODIGO, PERIODO and ANO_LETIVO are foreign keys, it might considered that adding it would improve the processing, when it actually adds cost when performing the match U.CODIGO = T.CODIGO.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	38
HASH		GROUP BY	2	38
Filter Predicates				
COUNT($vm_col_1)=5				
VIEW	SYS.VM_NWVW_1		404	38
HASH		GROUP BY	404	38
HASH JOIN			21019	36
Access Predicates				
U.CODIGO=T.CODIGO				
INDEX	UCS_IDX	FAST FULL SCAN	5396	7
INDEX	TIPOSAULA_IDX	FAST FULL SCAN	21019	29

Figure 3.28: Execution plan of query 6 for table Z with other possibility of indexes

This is a understandable result, since the B-tree is not only clustered by the CODIGO, but also by PERIODO and ANO_LETIVO, which are not used.

Thus, by analysing the Oracle Optimizer output, we were able to choose the right indexes for this question referenced in the section 2, which had a cost of 27, while the naive approach referenced above has a cost of 38.

References

[1] Glossary. URL: https://docs.oracle.com/cd/A97630_01/server.920/a96520/glossary.htm#433146. Accessed in: 06/04/2021.

[2] Soft vs. Hard Deadlines. URL: https://help.novoed.com/hc/en-us/articles/213421746-Soft-vs-Hard-Deadlines. Accessed in: 06/04/2021.