



Universidade do Porto
Faculdade de Engenharia
FEUP

QUERY OPTIMIZATION TEACHING SERVICE

Tecnologias de Bases de Dados
4º ano do Mestrado Integrado em Engenharia Informática e
Computação

Elementos do grupo F:

Catarina Ramos - up201406219 - up201406219@fe.up.pt
Inês Gomes - up201405778 - up201405778@fe.up.pt
Mário Fernandes - up201201705 - up201201705@fe.up.pt

9 de Abril de 2018

Questions

0. Justification of the constraints (Y) and the extra indexes (Z).

a. Y constraints

Para o conjunto Y temos como únicas restrições as restrições de integridade, ou seja, as primary keys e foreign keys para a construção da BD normalizada.

YDOCENTES tem como única constraint a sua primary key (nr).

YDSD é um caso especial de tabelas muitos para muitos. Tem duas foreign keys correspondentes às duas primary keys das duas tabelas que esta tabela associa (YDOCENTES e YTIPOAULA).

YTIPOSAULA tem a sua primary key (ID) e tem 3 foreign keys correspondentes às primary keys da tabela YOCORRENCIAS o que indicia uma relação 1 para muitos.

YOCORRENCIA é constituída por uma primary key de 3 elementos (codigo,ano_letivo,periodo) sendo uma delas uma foreign key para a tabela YUCS.

YUCS tem como única constraint a sua primary key (codigo).

b. Z extra indexes

Para o conjunto Z temos colocamos as restrições de Y mais dois novos índices:

```
CREATE INDEX ZTIPOSAULA_COD_ANO_IND ON ZTIPOSAULA(CODIGO, ANO_LETIVO)
CREATE INDEX ZUCS_CODCURS_IND ON ZUCS(CODIGO, CURSO)
```

O primeiro (ZTIPOSAULA_COD_ANO_IND) do tipo B-Tree junta o código de uma disciplina com o seu ano letivo na tabela ZTIPOSAULA. Embora, em conjunto com o período, estes atributos sejam primary key, são várias as queries que necessitam de saber o código e ano letivo sem o período, não dando uso ao index original da primary key.

O segundo (ZUCS_CODCURS_IND) também do tipo B-Tree junta o código de uma disciplina com o seu curso na tabela ZUCS. Esta tabela tem como primary key o código, mas não o curso. Como existem algumas cláusulas WHERE com seleção, não só pelo código, mas pelo curso também, este índice é otimizador.

1. Selection and join.

Show the *codigo*, *designacao*, *ano_letivo*, *inscritos*, *tipo*, and *turnos* for the course (*designacao*) '*Bases de Dados*' of the program (*curso*) 275.

a. SQL query;

```
SELECT XUCS.CODIGO, DESIGNACAO, XOCORRENCIAS.ANO_LETIVO, INSCRITOS, TIPO,
TURNOS
FROM XUCS
JOIN XOCORRENCIAS ON XUCS.CODIGO=XOCORRENCIAS.CODIGO
JOIN XTIPOSAULA ON XTIPOSAULA.CODIGO=XOCORRENCIAS.CODIGO
AND XTIPOSAULA.ANO_LETIVO=XOCORRENCIAS.ANO_LETIVO
AND XTIPOSAULA.PERIODO = XOCORRENCIAS.PERIODO
```

WHERE CURSO=275 and DESIGNACAO='Bases de Dados';

b. Answer;

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

c. Analysis of the three execution plans in the three environments and of the corresponding estimated effort;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	642
HASH JOIN			1	642
Access Predicates				
AND				
XTIPOSULA.CODIGO=XOCORRENCIAS.CODIGO				
XTIPOSULA.ANO_LETIVO=XOCORRENCIAS.ANO_LETIVO				
XTIPOSULA.PERIODO=XOCORRENCIAS.PERIODO				
XUCS.CODIGO=XOCORRENCIAS.CODIGO				
MERGE JOIN		CARTESIAN	4810	49
TABLE ACCESS	XUCS	FULL	1	13
Filter Predicates				
AND				
XUCS.DESIGNACAO='Bases de Dados'				
XUCS.CURSO=275				
BUFFER		SORT	21206	36
TABLE ACCESS	XTIPOSULA	FULL	21206	36
TABLE ACCESS	XOCORRENCIAS	FULL	21747	593

Figura 1: execution plan correspondente às tabelas 'X' da query da pergunta 1

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	55
HASH JOIN			5	55
Access Predicates				
AND				
YTIPOSULA.CODIGO=YOCORRENCIAS.CODIGO				
YTIPOSULA.ANO_LETIVO=YOCORRENCIAS.ANO_LETIVO				
YTIPOSULA.PERIODO=YOCORRENCIAS.PERIODO				
NESTED LOOPS			5	19
NESTED LOOPS			5	19
TABLE ACCESS	YUCS	FULL	1	13
Filter Predicates				
AND				
YUCS.DESIGNACAO='Bases de Dados'				
YUCS.CURSO=275				
INDEX	YOCORRENCIAS_PRIMARY_KEY	RANGE SCAN	5	1
Access Predicates				
YUCS.CODIGO=YOCORRENCIAS.CODIGO				
TABLE ACCESS	YOCORRENCIAS	BY INDEX ROWID	24	6
TABLE ACCESS	YTIPOSULA	FULL	21206	36

Figura 2: execution plan correspondente às tabelas 'Y' da query da pergunta 1

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			8	30
HASH JOIN			8	30
Access Predicates				
ZTIPOSAULA.CODIGO=ZOCORRENCIAS.CODIGO AND ZTIPOSAULA.ANO_LETIVO=ZOCORRENCIAS.ANO_LETIVO AND ZTIPOSAULA.PERIODO=ZOCORRENCIAS.PERI				
NESTED LOOPS			8	30
NESTED LOOPS			8	30
STATISTICS COLLECTOR				
HASH JOIN			8	19
Access Predicates				
ZUCS.CODIGO=ZOCORRENCIAS.CODIGO				
NESTED LOOPS			8	19
STATISTICS COLLECTOR				
TABLE ACCESS	ZUCS	FULL	1	13
Filter Predicates				
ZUCS.DESIGNACAO='Bases de Dados' AND ZUCS.CURSO=275				
TABLE ACCESS	ZOCORRENCIAS	BY INDEX ROWID BATCHED	36	6
INDEX	ZOCORRENCIAS_PRIMARY_KEY	RANGE SCAN	5	1
Access Predicates				
ZUCS.CODIGO=ZOCORRENCIAS.CODIGO				
TABLE ACCESS	ZOCORRENCIAS	FULL	36	6
INDEX	ZTIPOSAULA_COD_ANO_IND	RANGE SCAN	1	1
Access Predicates				
ZTIPOSAULA.CODIGO=ZOCORRENCIAS.CODIGO AND ZTIPOSAULA.ANO_LETIVO=ZOCORRENCIAS.ANO_LETIVO				
TABLE ACCESS	ZTIPOSAULA	BY INDEX ROWID	1	2
Filter Predicates				
ZTIPOSAULA.PERIODO=ZOCORRENCIAS.PERIODO				
TABLE ACCESS	ZTIPOSAULA	FULL	1	2

Figura 3: execution plan correspondente às tabelas 'Z' da query da pergunta 1

Na figura 1 é possível ver que o custo de execução da query sem índices é de 642, passando para 55 na figura 2 para o caso dos índices colocados nas primary e foreign keys. Também existe diferença no tipo de operações usadas. Enquanto sem índices é executado **hash join** seguido de um **merge join**, na figura 2 é executado um hash join seguido de dois **nested loops**. É de notar que na figura 1 a pesquisa é do tipo **full**, ou seja, é percorrido cada elemento da tabela um a um até chegar à solução. Esta é a opção mais custosa, o que explica um custo tão elevado.

Já na figura 2, neste caso de **nested loops**, para cada linha da tabela exterior o Oracle procura as linhas da tabela interior que satisfazem a condição de junção. De seguida combina os dados em pares de linhas que satisfazem a junção e devolve-as como resultado. Assim, o facto de existir um índice no **nested loop interior** otimiza a execução, porque executa o segundo passo em menos tempo. É apenas usado o index Y_OCORRENCIAS_PRIMARY_KEY.

Na figura 3, temos os mesmos índices da figura 2 mais os dois especificados na questão 0. Esse índice extra (ZTIPOSAULA_COD_AND_IND), é aplicado diminuindo o custo em quase metade. Este índice altera a ordem das operações executadas. Embora exista na mesma um **hash join** seguido de 2 **nested loops**, aqui dentro haverá um novo **hash join** com um **nested loop** para possibilitar o seu uso. Embora haja mais operações, o custo é muito menor. É também usado o índice ZOCORRENCIAS_PRIMARY_KEY.

- d. If the execution time is measurable, a comparison of the execution times in the three environments.

	X	Y	Z
Time (s)	0.08	0.03	0.022

A diferença de tempos não é significativa.

2. Aggregation.

How many class hours of each type (*tipo*) did the program (*curso*) 233 got in year (*ano letivo*) 2004/2005?

a. SQL query;

```
SELECT TIPO, SUM(N_AULAS * HORAS_TURNO) AS HOURS
FROM XUCS
JOIN XOCORRENCIAS ON XUCS.CODIGO = XOCORRENCIAS.CODIGO
JOIN XTIPOSAULA ON XOCORRENCIAS.CODIGO = XTIPOSAULA.CODIGO
      AND XOCORRENCIAS.ANO_LETIVO=XTIPOSAULA.ANO_LETIVO
      AND XOCORRENCIAS.PERIODO = XTIPOSAULA.PERIODO
WHERE XUCS.CURSO=233 AND XOCORRENCIAS.ANO_LETIVO='2004/2005'
GROUP BY TIPO;
```

b. Answer;

TIPO	HOURS
P	102,5
T	369
TP	299,5

c. Analysis of the three execution plans in the three environments and of the corresponding estimated effort;

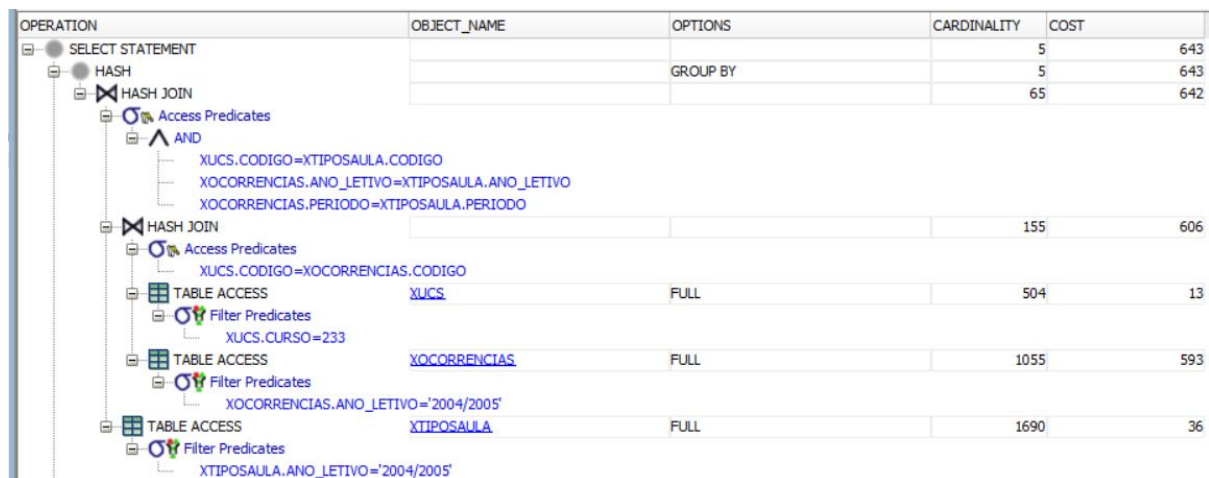


Figura 4: execution plan correspondente às tabelas 'X' da query da pergunta 2

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	50
HASH		GROUP BY	5	50
NESTED LOOPS			232	49
HASH JOIN			612	49
Access Predicates	YUCS.CODIGO=YTIPOSAULA.CODIGO			
TABLE ACCESS	YUCS	FULL	504	13
Filter Predicates	YUCS.CURSO=233			
TABLE ACCESS	YTIPOSAULA	FULL	1690	36
Filter Predicates	YTIPOSAULA.ANO_LETIVO='2004/2005'			
INDEX	YOCORRENCIAS_PRIMARY_KEY	UNIQUE SCAN	1	0
Access Predicates				
AND	YUCS.CODIGO=YOCORRENCIAS.CODIGO YOCORRENCIAS.ANO_LETIVO='2004/2005' YOCORRENCIAS.PERIODO=YTIPOSAULA.PERIODO			

Figura 5: execution plan correspondente às tabelas 'Y' da query da pergunta 2

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	44
HASH		GROUP BY	5	44
HASH JOIN			564	43
Access Predicates	ZUCS.CODIGO=ZOCORRENCIAS.CODIGO			
INDEX	ZUCS_CODCURS_IND	FAST FULL SCAN	504	7
Filter Predicates	ZUCS.CURSO=233			
NESTED LOOPS			1077	36
TABLE ACCESS	ZTIPOSAULA	FULL	1690	36
Filter Predicates	ZTIPOSAULA.ANO_LETIVO='2004/2005'			
INDEX	ZOCORRENCIAS_PRIMARY_KEY	UNIQUE SCAN	1	0
Access Predicates				
AND	ZOCORRENCIAS.CODIGO=ZTIPOSAULA.CODIGO ZOCORRENCIAS.ANO_LETIVO='2004/2005' ZOCORRENCIAS.PERIODO=ZTIPOSAULA.PERIODO			

Figura 6: execution plan correspondente às tabelas 'Z' da query da pergunta 2

Na figura 4 é possível verificar que o custo de execução sem qualquer índice é de 643, que é reduzido para 50 na figura 5 com apenas os índices primários e secundários. Tal como já foi mencionado anteriormente, as figuras relativas às tabelas X usam maioritariamente as opções **full** que obrigam a percorrer a tabela elemento a elemento, causando uma cardinalidade superior e um custo maior.

Neste caso, a figura 5 tem como ordem de execução **nested loop hash join** (diferente da figura 4). Aqui é usado o índice YOCORRENCIAS_PRIMARY_KEY para filtrar o join, sendo o grande diminuidor de custo.

Já para a figura 6 existe uma redução de custo diminuta, relativa apenas ao acréscimo do índice ZUCS_CODCURS_IND que associa o curso ao código da cadeira, possibilitando um **fast full scan**. O acréscimo deste índice é suficiente para inverter a ordem de execução que passa a ser **hash join nested loop**.

- d. If the execution time is measurable, a comparison of the execution times in the three environments.

	X	Y	Z
Time (s)	0.038	0.032	0.029

A diferença de tempos não é significativa.

3. Negation.

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

a. Use not in.

i. SQL query;

```
SELECT CODIGO
FROM XUCS
WHERE CODIGO NOT IN
(
  SELECT CODIGO
  FROM XOCORRENCIAS
  WHERE ANO_LETIVO='2003/2004'
);
```

ii. Answer;

Existem 4378 cursos que não foram lecionados no ano letivo 2003/2004.

iii. Analysis of the three execution plans in the three environments and of the corresponding estimated effort;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5396	606
HASH JOIN		RIGHT ANTI	5396	606
Access Predicates		CODIGO=CODIGO		
TABLE ACCESS	XOCORRENCIAS	FULL	1028	593
Filter Predicates		ANO_LETIVO='2003/2004'		
TABLE ACCESS	XUCS	FULL	5396	13

Figura 7: execution plan correspondente às tabelas 'X' da query da pergunta 3 alínea a

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5396	33
HASH JOIN		RIGHT ANTI	5396	33
Access Predicates		CODIGO=CODIGO		
INDEX	XOCORRENCIAS_PRIMARY_KEY	FAST FULL SCAN	1028	27
Filter Predicates		ANO_LETIVO='2003/2004'		
INDEX	XUCS_PRIMARY_KEY	FAST FULL SCAN	5396	6

Figura 8: execution plan correspondente às tabelas 'Y' da query da pergunta 3 alínea a

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5396	33
HASH JOIN		RIGHT ANTI	5396	33
Access Predicates		CODIGO=CODIGO		
INDEX	XOCORRENCIAS_PRIMARY_KEY	FAST FULL SCAN	1028	27
Filter Predicates		ANO_LETIVO='2003/2004'		
INDEX	XUCS_PRIMARY_KEY	FAST FULL SCAN	5396	6

Figura 9: execution plan correspondente às tabelas 'Z' da query da pergunta 3 alínea a

Na figura 7 é possível verificar um custo de execução de 606 que é reduzido para 33 na figura 8. Neste caso, ambas as execuções usam **hash join** no entanto a segunda usa dois índices, YOCORRENCIAS_PRIMARY_KEY e YUCS_PRIMARY_KEY. Estes dois índices permitem o uso da opção **fast full scan** nas igualdades presentes no select interior e no próprio not in. A figura 8 é igual à figura 7 visto que nenhum dos índices adicionados é usado neste contexto.

- iv. If the execution time is measurable, a comparison of the execution times in the three environments.

	X	Y	Z
Time (s)	0.033	0.027	0.027

- b. Use external join and is not null.

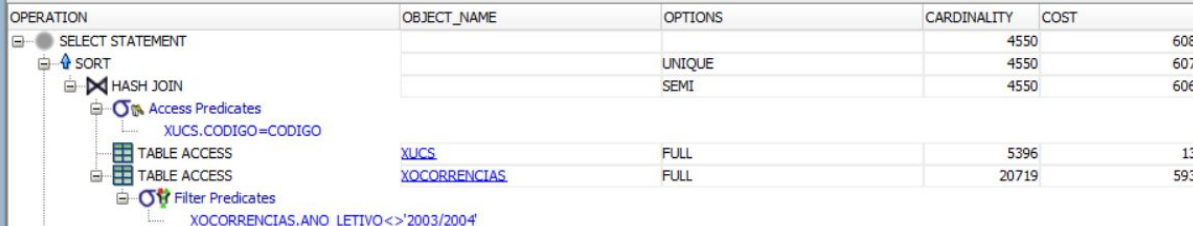
- i. SQL query;

```
SELECT DISTINCT XUCS.CODIGO
FROM XUCS
LEFT OUTER JOIN
(
    SELECT CODIGO, ANO_LETIVO
    FROM XOCORRENCIAS
    WHERE XOCORRENCIAS.ANO_LETIVO != '2003/2004'
) TEMP2
ON XUCS.CODIGO = TEMP2.CODIGO
WHERE ANO_LETIVO IS NOT NULL
ORDER BY CODIGO ASC;
```

- ii. Answer;

Igual a 3 a) ii.

- iii. Analysis of the three execution plans in the three environments and of the corresponding estimated effort;



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			4550	608
SORT		UNIQUE	4550	607
HASH JOIN		SEMI	4550	606
Access Predicates				
XUCS.CODIGO=CODIGO				
TABLE ACCESS	XUCS	FULL	5396	13
TABLE ACCESS	XOCORRENCIAS	FULL	20719	593
Filter Predicates				
XOCORRENCIAS.ANO_LETIVO <> '2003/2004'				

Figura 10: execution plan correspondente às tabelas 'X' da query da pergunta 3 alínea b



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			4550	30
INDEX	YOCORRENCIAS_PRIMARY_KEY	FAST FULL SCAN	20719	27
Filter Predicates				
YOCORRENCIAS.ANO_LETIVO <> '2003/2004'				

Figura 11: execution plan correspondente às tabelas 'Y' da query da pergunta 3 alínea b

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			4550	30
SORT		UNIQUE	4550	28
INDEX	ZOCORRENCIAS_PRIMARY_KEY	FAST FULL SCAN	20719	27
Filter Predicates	ZOCORRENCIAS.ANO_LETIVO<>'2003/2004'			

Figura 12: execution plan correspondente às tabelas 'Z' da query da pergunta 3 alínea b

Na figura 10 é possível verificar um custo de execução de 608, reduzido para 30 na figura 11. Neste caso, o uso do índice Y_OCORRENCIAS_PRIMARY_KEY é suficiente para mudar as operações executadas. No primeiro é executado um **sort** seguido de **hash join** que executam em opção **full**. No caso da figura 11 é apenas usado um **sort** que ordena pelo index na opção **fast full scan**.

A figura 12 é igual à figura 11 visto que nenhum dos índices adicionados é usado neste contexto.

Em comparação com a outra negação (3.a) para as tabelas Z, esta é mais eficiente porque apenas utiliza ordenação. Na query anterior é necessário fazer um **hash join** devido às duas comparações e usar índices nas duas. Neste caso só existe um índice.

- iv. If the execution time is measurable, a comparison of the execution times in the three environments.

	X	Y	Z
Time (s)	0.048	0.031	0.031

A diferença de tempos diverge um pouco entre as tabelas X e Y/Z.

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.

- a. SQL query;

```

SELECT TIPO, MAX_HORAS, XDOCENTES.NR, XDOCENTES.NOME
FROM
( --Gets the max hours per type of class
  SELECT MAX(NR) AS NR, TIPO, MAX(HORAS_TOTAIS) AS MAX_HORAS
  FROM
  ( --Selects the total hours per professor per type of class
    SELECT NR, TIPO, SUM(HORAS * FATOR) AS HORAS_TOTAIS
    FROM XTIPOSAULA
    JOIN XDSD ON XDSD.ID = XTIPOSAULA.ID
    WHERE ANO_LETIVO = '2003/2004'
    GROUP BY TIPO, NR
  )
)
GROUP BY TIPO

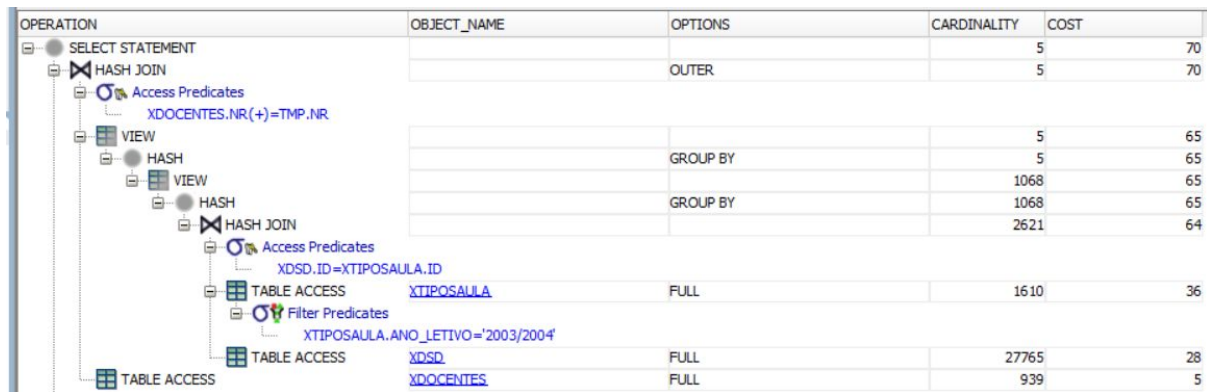
```

) TMP
LEFT JOIN XDOCENTES ON XDOCENTES.NR = TMP.NR;

b. Answer;

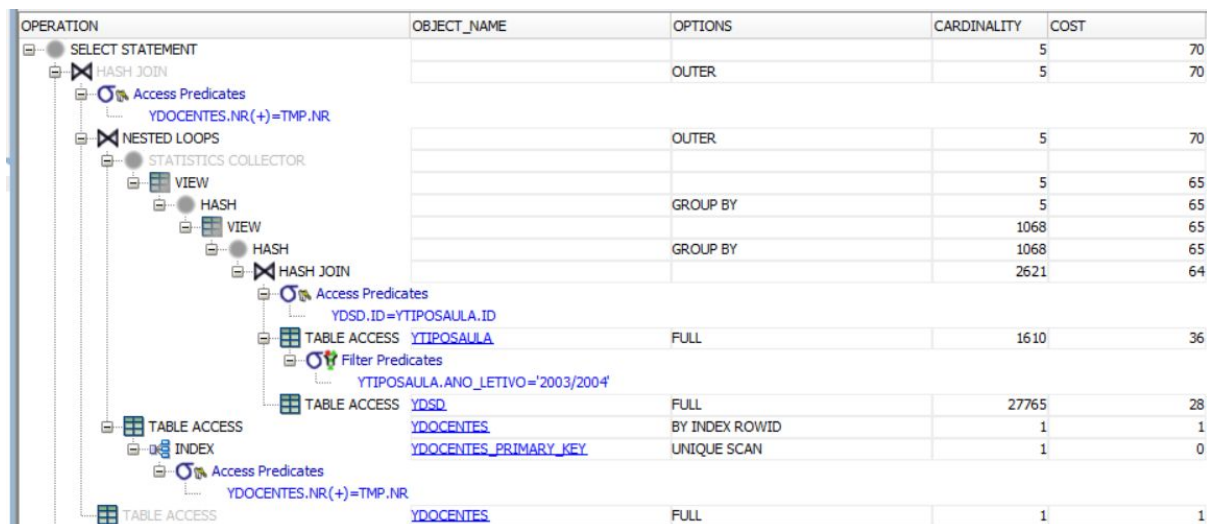
TIPO	MAX_HORAS	NR	NOME
OT	3,5	246626	Jorge Manuel Gomes Barbosa
P	30	908100	Armínio de Almeida Teixeira
TP	26	908290	José Manuel Miguez Araújo
T	30,67	909330	Nuno Filipe da Cunha Nogueira

c. Analysis of the three execution plans in the three environments and of the corresponding estimated effort;



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	70
HASH JOIN		OUTER	5	70
Access Predicates XDOCENTES.NR(+) = TMP.NR				
VIEW			5	65
HASH		GROUP BY	5	65
VIEW			1068	65
HASH		GROUP BY	1068	65
HASH JOIN			2621	64
Access Predicates XDSO.ID = XTIPOSAULA.ID				
TABLE ACCESS XTIPOSAULA		FULL	1610	36
Filter Predicates XTIPOSAULA.ANO_LETIVO = '2003/2004'				
TABLE ACCESS XDSO		FULL	27765	28
TABLE ACCESS XDOCENTES		FULL	939	5

Figura 13: execution plan correspondente às tabelas 'X' da query da pergunta 4



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	70
HASH JOIN		OUTER	5	70
Access Predicates YDOCENTES.NR(+) = TMP.NR				
NESTED LOOPS				
STATISTICS COLLECTOR				
VIEW			5	65
HASH		GROUP BY	5	65
VIEW			1068	65
HASH		GROUP BY	1068	65
HASH JOIN			2621	64
Access Predicates YDSO.ID = YTIPOSAULA.ID				
TABLE ACCESS YTIPOSAULA		FULL	1610	36
Filter Predicates YTIPOSAULA.ANO_LETIVO = '2003/2004'				
TABLE ACCESS YDSO		FULL	27765	28
TABLE ACCESS YDOCENTES		BY INDEX ROWID	1	1
INDEX YDOCENTES_PRIMARY_KEY		UNIQUE SCAN	1	0
Access Predicates YDOCENTES.NR(+) = TMP.NR				
TABLE ACCESS YDOCENTES		FULL	1	1

Figura 14: execution plan correspondente às tabelas 'Y' da query da pergunta 4

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	70
HASH JOIN		OUTER	5	70
Access Predicates				
ZDOCENTES.NR(+) = TMP.NR				
NESTED LOOPS		OUTER	5	70
STATISTICS COLLECTOR				
VIEW			5	65
HASH		GROUP BY	5	65
VIEW			1068	65
HASH		GROUP BY	1068	65
HASH JOIN			2621	64
Access Predicates				
ZDSD.ID = ZTIPOSAULA.ID				
TABLE ACCESS	ZTIPOSAULA	FULL	1610	36
Filter Predicates				
ZTIPOSAULA.ANO_LETIVO = '2003/2004'				
TABLE ACCESS	ZDSD	FULL	27765	28
TABLE ACCESS	ZDOCENTES	BY INDEX ROWID	1	1
INDEX	ZDOCENTES_PRIMARY_KEY	UNIQUE SCAN	1	0
Access Predicates				
ZDOCENTES.NR(+) = TMP.NR				
TABLE ACCESS	ZDOCENTES	FULL	1	1

Figura 15: execution plan correspondente às tabelas 'Z' da query da pergunta 4

Neste exercício a própria query já está otimizada, pelo que o custo é 70 para as 3 figuras. No entanto, existe uma diferença nas diferentes execução devido ao uso de índices nas figuras 14 e 15. Nestas 2 últimas figuras, não há diferença do plano de execução visto que usam o mesmo índice YDOCENTES_PRIMARY_KEY. Foi estudada a criação de um índice para o ano letivo, no entanto, embora o índice fosse usado, o custo apenas diferia em 3 e a cardinalidade era a mesma. Também já temos dois índices sobre este atributo, pelo que não seria racional acrescentar mais um índice. Um índice na seleção do **hash join** seria, teoricamente, a maneira mais eficiente de encarar o problema.

- d. If the execution time is measurable, a comparison of the execution times in the three environments.

	X	Y	Z
Time (s)	0.038	0.033	0.033

A diferença de tempos não é significativa.

5. Compare the execution plans and the index sizes for the query giving the course code (*codigo*), the academic year (*ano_letivo*), the period(*periodo*), and number of hours of the '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 XTIPOSAULA table;

- i. SQL query;

```
SELECT XUCS.CODIGO, XOCORRENCIAS.ANO_LETIVO, XOCORRENCIAS.PERIODO,
SUM(XDSD.HORAS) AS HORAS
```

```

FROM XUCS
  JOIN XOCORRENCIAS ON XUCS.CODIGO = XOCORRENCIAS.CODIGO
  JOIN XTIPOSAULA ON XTIPOSAULA.CODIGO=XOCORRENCIAS.CODIGO
                  AND XTIPOSAULA.ANO_LETIVO=XOCORRENCIAS.ANO_LETIVO
                  AND XTIPOSAULA.PERIODO = XOCORRENCIAS.PERIODO
  JOIN XDSD ON XTIPOSAULA.ID = XDSD.ID
WHERE XTIPOSAULA.TIPO = 'OT'
      AND (XOCORRENCIAS.ANO_LETIVO = '2002/2003' OR XOCORRENCIAS.ANO_LETIVO =
'2003/2004')
GROUP BY XUCS.CODIGO, XOCORRENCIAS.ANO_LETIVO, XOCORRENCIAS.PERIODO;

```

ii. Answer;

CODIGO	ANO_LETIVO	PE	HORAS
EIC5202	2002/2003	2S	25.5
EIC5202	2003/2004	2S	22

iii. Analysis of the three execution plans in the three environments and of the corresponding estimated effort;

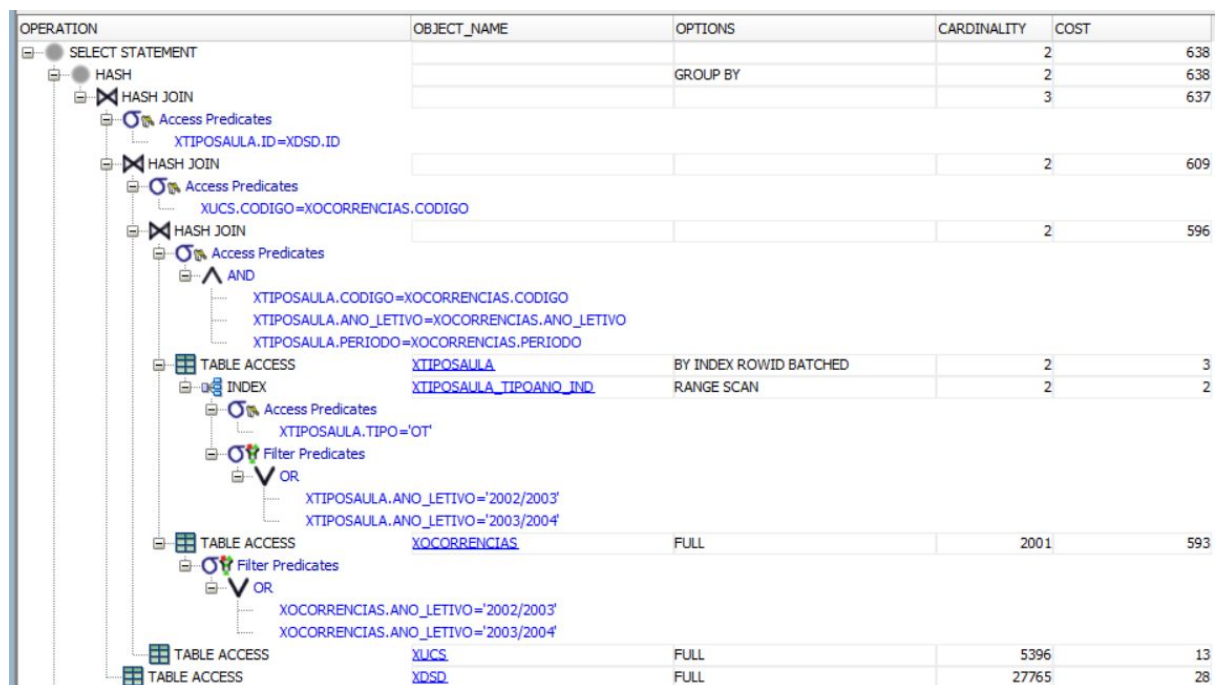


Figura 16: execution plan correspondente às tabelas 'X' da query da pergunta 5 alínea a

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			358	65
HASH		GROUP BY	358	65
HASH JOIN			1012	64
Access Predicates				
YTIPOSAULA.ID=YDSD.ID				
NESTED LOOPS			622	36
TABLE ACCESS	YTIPOSAULA	FULL	622	36
Filter Predicates				
AND				
YTIPOSAULA.TIPO='OT'				
OR				
YTIPOSAULA.ANO_LETIVO='2002/2003'				
YTIPOSAULA.ANO_LETIVO='2003/2004'				
INDEX	YOCORRENCIAS_PRIMARY_KEY	UNIQUE SCAN	1	0
Access Predicates				
AND				
YTIPOSAULA.CODIGO=YOCORRENCIAS.CODIGO				
YTIPOSAULA.ANO_LETIVO=YOCORRENCIAS.ANO_LETIVO				
YTIPOSAULA.PERIODO=YOCORRENCIAS.PERIODO				
Filter Predicates				
OR				
YOCORRENCIAS.ANO_LETIVO='2002/2003'				
YOCORRENCIAS.ANO_LETIVO='2003/2004'				
TABLE ACCESS	YDSD	FULL	27765	28

Figura 17: execution plan correspondente às tabelas 'Y' da query da pergunta 5 alínea a

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	65
HASH		GROUP BY	1	65
NESTED LOOPS			1	65
VIEW	SYS.VW_GBC_9		1	65
HASH JOIN		GROUP BY	1	65
Access Predicates			3	64
ZTIPOSAULA.ID=ZDSD.ID				
TABLE ACCESS	ZTIPOSAULA	FULL	2	36
Filter Predicates				
AND				
ZTIPOSAULA.TIPO='OT'				
OR				
ZTIPOSAULA.ANO_LETIVO='2002/2003'				
ZTIPOSAULA.ANO_LETIVO='2003/2004'				
TABLE ACCESS	ZDSD	FULL	27765	28
Access Predicates				
AND				
ITEM_3=ZOCORRENCIAS.CODIGO				
ITEM_2=ZOCORRENCIAS.ANO_LETIVO				
ITEM_1=ZOCORRENCIAS.PERIODO				
Filter Predicates				
OR				
ZOCORRENCIAS.ANO_LETIVO='2002/2003'				
ZOCORRENCIAS.ANO_LETIVO='2003/2004'				
INDEX	ZOCORRENCIAS_PRIMARY_KEY	UNIQUE SCAN	1	0

Figura 18: execution plan correspondente às tabelas 'Z' da query da pergunta 5 alínea a

Na figura 16 é possível ver que o custo de execução da query sem índices é de 638. Na figura 17 e 18 são em ambos 65, embora com cardinalidades diferentes. Como na tabela X não existem primary keys, fazer joins entre as tabelas fica bastante custoso, chegando a existir 3 **hash joins**. Graças ao index b-tree criado, identificar a combinação entre o tipo e o ano_letivo da tabela tiposaula fica extremamente menos custoso face a uma pesquisa sem index, visto que o custo total do acesso à tabela ficaria por 36 em vez do atual de apenas custo 3.

Comparativamente aos casos retratados nas figuras 17 e 18, o custo é claramente menor que na tabela X sendo que dois **hash joins** são substituídos por 1 **nested loop**, fazendo ao todo apenas 2 pesquisas **FULL** em vez de 3.

- iv. If the execution time is measurable, a comparison of the execution times in the three environments.

	X	Y	Z
Time (s)	0.042	0.03	0.03

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

i. SQL query;

Igual à query 5 a) i.

ii. Answer;

Igual à query 5 a) ii.

iii. Analysis of the three execution plans in the three environments and of the corresponding estimated effort;

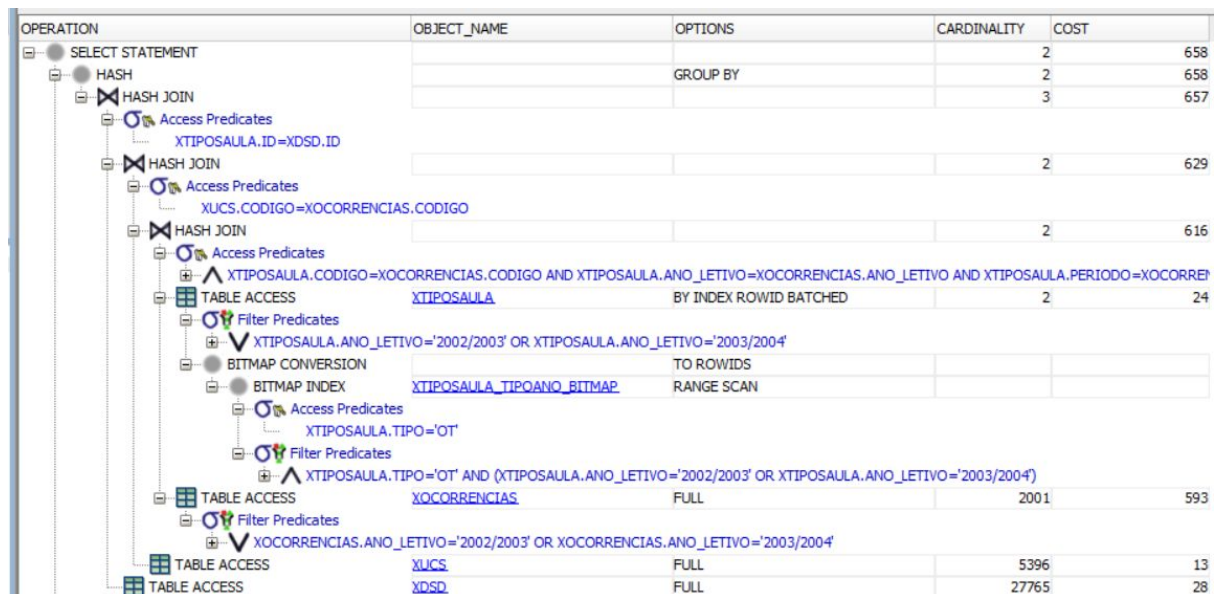


Figura 19: execution plan correspondente às tabelas 'X' da query da pergunta 5 alínea b

A figura 19 retrata um caso semelhante ao da figura 16: custo bastante alto face aos casos da figura 17 e 18, 3 **hash joins** e um acesso a uma tabela por index. Neste caso o index usado é um **bitmap** e o custo torna-se maior face ao index em **b-tree**. O acesso à tabela tiposaula também é mais custoso, por isso o uso deste index é desaconselhado.

iv. If the execution time is measurable, a comparison of the execution times in the three environments.

	X	Y	Z
Time (s)	0.04	0.03	0.03

Conclusão deste exercício

Para poucas combinações entre parâmetros (como é o caso de relativamente poucos anos letivos e apenas 5 tipos de aulas), os indexes em b-tree são claramente melhor, visto que não só tornam o acesso mais rápido como também têm um impacto pequeno na inserção ou modificação de dados.

Se fosse utilizado na tabela Z o index em b-tree, o custo da query iria para quase metade do atual. Isto é bastante bom, mas como até o parâmetro ano_letivo já tem 3 outros índices associados e como a diminuição de custo é de apenas 30, o grupo achou desnecessário (e possivelmente prejudicial consoante a escalabilidade da base de dados) o uso de um index adicional.

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

a. SQL query;

```
SELECT CURSO
FROM
( --Count the number of types of classes per course
  SELECT CURSO, COUNT(TIPO) AS N_TIPO
  FROM
  (
    --Type of classes per course
    SELECT XUCS.CURSO, TIPO
    FROM XUCS
    JOIN XTIPOSAULA ON XUCS.CODIGO = XTIPOSAULA.CODIGO
    GROUP BY XUCS.CURSO, TIPO
    ORDER BY XUCS.CURSO
  )
  GROUP BY CURSO
)
WHERE N_TIPO =
( --Number of types of classes
  SELECT COUNT(TIPO) AS N_TIPO
  FROM
  (
    SELECT TIPO
    FROM XTIPOSAULA
    GROUP BY TIPO
  )
);
```

b. Answer;

CURSO
9508

2021
9461
4495

c. Analysis of the three execution plans in the three environments and of the corresponding estimated effort;

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	51
FILTER				
Filter Predicates	COUNT(\$vm_col_1) = (SELECT COUNT(\$vm_col_1) FROM (SELECT TIPO \$vm_col_1 FROM XTIPOSAULA XTIPOSAULA GROUP BY TIPO) VM_NWWW_1)			
HASH		GROUP BY	2	51
VIEW	SYS.VM_NWWW_0		206	51
HASH		GROUP BY	206	51
HASH JOIN			21206	49
Access Predicates	XUCS.CODIGO=XTIPOSAULA.CODIGO			
TABLE ACCESS	XUCS	FULL	5396	13
TABLE ACCESS	XTIPOSAULA	FULL	21206	36
SORT		AGGREGATE	1	
VIEW	SYS.VM_NWWW_1		5	37
SORT		GROUP BY	5	37
TABLE ACCESS	XTIPOSAULA	FULL	21206	36

Figura 20: execution plan correspondente às tabelas 'X' da query da pergunta 6

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	51
FILTER				
Filter Predicates	COUNT(\$vm_col_1) = (SELECT COUNT(\$vm_col_1) FROM (SELECT TIPO \$vm_col_1 FROM YTIPOSAULA YTIPOSAULA GROUP BY TIPO) VM_NWWW_1)			
HASH		GROUP BY	2	51
VIEW	SYS.VM_NWWW_0		206	51
HASH		GROUP BY	206	51
HASH JOIN			21206	49
Access Predicates	YUCS.CODIGO=YTIPOSAULA.CODIGO			
TABLE ACCESS	YUCS	FULL	5396	13
TABLE ACCESS	YTIPOSAULA	FULL	21206	36
SORT		AGGREGATE	1	
VIEW	SYS.VM_NWWW_1		5	37
SORT		GROUP BY	5	37
TABLE ACCESS	YTIPOSAULA	FULL	21206	36

Figura 21: execution plan correspondente às tabelas 'Y' da query da pergunta 6

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	45
FILTER				
Filter Predicates	COUNT(\$vm_col_1) = (SELECT COUNT(\$vm_col_1) FROM (SELECT TIPO \$vm_col_1 FROM ZTIPOSAULA ZTIPOSAULA GROUP BY TIPO) VM_NWWW_1)			
HASH		GROUP BY	2	45
VIEW	SYS.VM_NWWW_0		404	45
HASH		GROUP BY	404	45
HASH JOIN			21206	43
Access Predicates	ZUCS.CODIGO=ZTIPOSAULA.CODIGO			
INDEX	ZUCS.CODCURS_IND	FAST FULL SCAN	5396	7
TABLE ACCESS	ZTIPOSAULA	FULL	21206	36
SORT		AGGREGATE	1	
VIEW	SYS.VM_NWWW_1		5	37
SORT		GROUP BY	5	37
TABLE ACCESS	ZTIPOSAULA	FULL	21206	36

Figura 22: execution plan correspondente às tabelas 'Z' da query da pergunta 6

Nesta query consegue-se concluir que o uso de primary e foreign keys é inútil e apenas índices a conseguem otimizar. Isto deve-se ao facto de ter de ser feito um acesso do tipo **full** a todas as tabelas em causa. Por causa disto, a única otimização aqui presente é a do index que conjuga o código e o curso da tabela ucs (ZUCS_CODCURS_IND). Em todas as 3 figuras 20, 21 e 22 é feito um **hash join**.

- d. If the execution time is measurable, a comparison of the execution times in the three environments.

	X	Y	Z
Time (s)	0.05	0.05	0.047

Conclusão

No que diz respeito a otimização de queries, em caso geral, o mais importante é definir as primary e foreign keys das tabelas. Com o recurso a índices e a diferentes maneiras de fazer uma query, como as que foram exploradas neste trabalho, é possível obter um plano de execução o máximo otimizado. Ainda assim, conseguiu-se ver em alguns exercícios que o tempo de uma query a tabelas já com primary e foreign keys pode ser reduzido para menos de metade.

Uma das otimizações verificadas foi relativa à formulação das queries. É possível verificar que diferentes formulações como as agregações, negação, joins e seleções produzem planos de execução diferentes, com custos e cardinalidades diferentes.

Relativamente à diferenciação entre índices, chegou-se à conclusão que os índices b-tree são os mais indicados para parâmetros que podem assumir um baixo número de valores. Os índices bitmap são bastante mais custosos, nomeadamente em inserções ou modificações de uma tabela, mas para parâmetros que podem assumir muitos valores consegue-se obter uma melhor otimização.