Faculdade de Engenharia da Universidade do Porto

Relatório TBDA

1.º Projeto

Otimização de Interrogações - Candidatos

Turma 1

João Augusto dos Santos Lima, up201605314@fe.up.pt

Susana Maria de Sousa Lima, up201603634@fe.up.pt

Índice

Introdução **4**

Justificação das restrições **4**

Chaves Primárias 4

Chaves Estrangeiras 5

Justificação dos índices **5**

Método **7**

Perguntas e respostas **7**

Pergunta 1 - Seleção e Junção 7

1.1. Formulação SQL 7

1.2. Resultados 8

1.3. Planos de execução 9

1.3.1. Ambiente X 9

1.3.2. Ambiente Y 9

1.3.3. Ambiente Z 9

1.4. Tempos de execução 9

1.5. Conclusões 10

Pergunta 2 - Agregação 10

2.1. Formulação SQL 10

2.2. Resultados 11

2.3. Planos de execução 11

2.3.1. Ambiente X 11

2.3.2. Ambiente Y 12

2.3.3. Ambiente Z 12

2.4. Tempos de execução 12

2.5. Conclusões 12

Pergunta 3 13

3.1. Formulação SQL 13

3.1.1. Subpergunta Constante 13

3.1.2. Subpergunta Variável 13

3.2. Resultados 14

3.3. Planos de execução 14

3.3.1 Ambiente X 14

3.3.1.1. Subpergunta Constante 14

3.3.1.2. Subpergunta Variável 15

3.3.2 Ambiente Y 15

3.3.2.1. Subpergunta Constante 15

3.3.2.2. Subpergunta Variável 15

3.3.3 Ambiente Z 16
3.3.3.1. Subpergunta Constante 16
3.3.3.2. Subpergunta Variável 16
3.4. Tempos de execução 16
3.4.2. Subpergunta Constante 16
3.4.2. Subpergunta Variável 16
3.5. Conclusões 17
Pergunta 4 17
4.1. Formulação SQL 17
4.1.1. Versão 1 17
4.1.2. Versão 2 18
4.2. Resultados 19
4.3. Planos de execução 19
4.3.1. Ambiente X 19
4.3.1.1. Versão 1 19
4.3.1.2. Versão 2 20
4.3.2. Ambiente Y 20
4.3.2.1. Versão 1 20
4.3.2.2. Versão 2 20
4.3.3. Ambiente Z 21
4.3.3.1. Versão 1 21
4.3.3.2. Versão 2 21
4.4. Tempos de execução 21
4.4.1. Versão 1 21
4.4.2. Versão 2 22
4.5. Conclusões 22
Pergunta 5 22
5.1. Formulação SQL 22
5.2. Resultados 23
5.3. Planos de execução 23
5.3.1. Ambiente X 23
5.3.2. Ambiente Y 23
5.3.3. Ambiente Z árvore B 23
5.3.4. Ambiente Z bitmap 23
5.4. Tempos de execução 23
5.5. Conclusões 24
Pergunta 6 24
6.1. Formulação SQL 24
6.1.1. Contagem 24
6.1.2. Dupla negação 25
6.2. Resultados 26
6.3. Planos de execução 26

6.3.1. Ambiente X 26
6.3.1.1. Contagem 26
6.3.1.2. Dupla negação 27
6.3.2. Ambiente Y 27
6.3.2.1. Contagem 27
6.3.2.2. Dupla negação 28
6.3.3. Ambiente Z 28
6.3.3.1. Contagem 28
6.3.3.2. Dupla negação 29
6.4. Tempos de execução 29
6.4.1. Contagem 29
6.4.2. Dupla negação 29
6.5. Conclusões 30
Conclusões Finais **31**

Introdução

O projeto consiste na análise de planos de execução de diferentes interrogações *SQL* a uma base de dados e na avaliação do impacto da existência de diferentes índices e estatísticas e do recurso a diversas estratégias de estruturação de perguntas. Para tal, foram criados 3 ambientes de execução, um sem índices nem restrições de integridade, ambiente X, um com as restrições de integridade standard (chave primária e chave estrangeira), ambiente Y, e por fim, um com as restrições de integridade *standard* e índices adicionais, ambiente Z.

Justificação das restrições

As restrições de integridade (chaves primárias e chaves estrangeiras) foram adicionadas aos ambientes Y e Z de acordo com o especificado no enunciado do projeto, nomeadamente:

- Chave primária em *code*, tabela *programs*;
- Chave primária em *id*, *program*, *year*, tabela *candidates*;
- Chave primária em *nr* , tabela *students*;
- Chave estrangeira em *program*, tabela *candidates*, referindo *code*, tabela *programs*;
- Chave estrangeira em *program*, tabela *students*, referindo *code*, tabela *programs*;
- Chave estrangeira em *id, program, enroll_year*, tabela *students*, referindo *id, program, year*, tabela *candidates*.

Chaves Primárias

```
alter table yprograms add constraint yprograms_code_pk primary key
(code);
alter table zprograms add constraint zprograms_code_pk primary key
(code);

alter table ycandidates add constraint ycandidates_id_program_year_pk
primary key (id, program, year);
alter table zcandidates add constraint zcandidates_id_program_year_pk
primary key (id, program, year);
```

```
alter table ystudents add constraint ystudents_nr_pk primary key (nr);
alter table zstudents add constraint zstudents_nr_pk primary key (nr);
```

Chaves Estrangeiras

```
alter table ycandidates add constraint ycandidates_program_fk foreign
key (program) references yprograms(code);
alter table zcandidates add constraint zcandidates_program_fk foreign
key (program) references zprograms(code);

alter table ystudents add constraint ystudents_program_fk foreign key
(program) references yprograms(code);
alter table zstudents add constraint zstudents_program_fk foreign key
(program) references zprograms(code);

alter table ystudents add constraint ystudents_id_program_year_fk
foreign key (id, program, enroll_year) references ycandidates(id,
program, year);
alter table zstudents add constraint zstudents_id_program_year_fk
foreign key (id, program, enroll_year) references zcandidates(id,
program, year);
```

Justificação dos índices

De modo a melhorar a eficiência e a diminuir o tempo de execução das interrogações feitas, foram adicionados seis índices ao ambiente Z. Os índices foram criados com base nas interrogações especificadas e nos atributos (e conjuntos de atributos) mais frequentes nas mesmas. É de referir que foram escolhidos índices *B-tree* quando todos os campos do índice possuem cardinalidade alta. Assim, foram criados os seguintes índices:

- ***students_program_status_conclusion_enroll***: índice composto do tipo *B-tree* na diferença entre as colunas *conclusion_year* e *enroll_year* e nas colunas *program*, *status* da tabela *students*. Este índice foi adicionado com o objetivo de otimizar a *query* da primeira pergunta. A ordem dos campos deste índice corresponde à ordem do seu acesso pela *query* em questão.
- ***students_id_program_enroll_year***: índice composto do tipo *B-tree* nas colunas *id*, *program* e *enroll_year* da tabela *students*. Este índice foi adicionado com o intuito de

melhorar a eficiência das *queries* da pergunta 3 e da pergunta 6, uma vez que ambas utilizam estes atributos em conjunto nas suas pesquisas.

- ***students_avg_conclusion_program***: índice do tipo *B-tree* nas colunas *final_average, conclusion_year, program* da tabela *students*. Este índice foi adicionado com o objetivo de otimizar as *queries* da quarta pergunta, uma vez que estas colunas são frequentemente selecionadas em conjunto durante a pesquisa.
- ***candidates_result***: índice do tipo *bitmap* na coluna *result* da tabela *candidates*. Este índice foi criado como resposta à pergunta 5, onde é pedida a comparação entre o índice *B-tree* e o índice *bitmap* nesta coluna. Verificou-se que o *bitmap* revelou melhores resultados, pelo que foi mantido.
- ***candidates_program_year_result***: índice composto do tipo *bitmap* nas colunas *result, program, year* da tabela *candidates.* Este índice foi adicionado com o propósito de otimizar as *queries* da pergunta 6, uma vez que usam este conjunto de atributos. A coluna *result* apresenta uma baixa cardinalidade, pelo que se torna uma boa candidata para um índice do tipo *bitmap.* As restantes colunas são acedidas com frequências após uma pesquisa por resultado.
- ***students_program_enroll***: índice composto do tipo *B-tree* nas colunas *program* e *enroll_year* da tabela *students*. Este índice foi adicionado para melhorar a eficiência da pergunta seis, em conjunto com o índice anterior.

```
create index students_program_status_conclusion_enroll on zstudents
(conclusion_year - enroll_year, status, program);

create index students_id_program_enroll_year on zstudents (id,
program,enroll_year);

create index students_avg_conclusion_program on zstudents
(final_average, conclusion_year, program);

create bitmap index candidates_result on zcandidates (result);

create bitmap index candidates_program_year_result on zcandidates
(result, program, year);

create index students_program_enroll on zstudents (program,
enroll_year);
```

Método

Para cada uma das interrogações apresentadas no enunciado, foi comparado o tempo e o custo da execução em cada um dos ambientes definidos (X,Y e Z). De modo a obter o tempo de execução, cada *query* foi executada 15 vezes, calculando, no fim, a média dos tempos obtidos.

Perguntas e respostas

Pergunta 1 - Seleção e Junção

Qual a lista dos números dos alunos especiais, que terminaram o curso com sigla EIC em menos de cinco anos, e quantos anos demoraram.

1.1. Formulação SQL

```
select s.nr, s.conclusion_year - s.enroll_year as years
from  xstudents s join xprograms p on s.program = p.code
where p.acronym = 'EIC'
and  s.conclusion_year - s.enroll_year < 5
and s.status = 'C'
order by s.nr
```

1.2. Resultados

	NR	YEARS
1	940509003	4
2	940509004	4
3	940509008	4
4	940509010	4
5	940509011	4
6	940509012	4
7	940509013	4
8	940509014	4
9	940509015	4
10	940509017	4
11	940509018	4
12	940509022	4
13	940509024	4
14	940509025	4
15	940509027	4
16	940509033	4
17	940509039	4
18	940509040	4
19	940509041	4
20	940509042	4
21	940509043	4

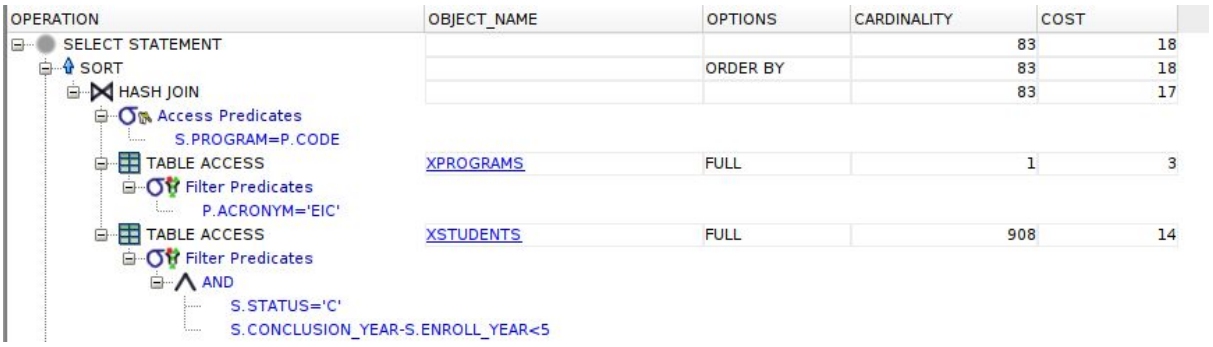
	NR	YEARS
22	940509045	4
23	940509049	4
24	950509002	4
25	950509003	4
26	950509006	4
27	950509009	4
28	950509011	4
29	950509012	4
30	950509013	4
31	950509014	4
32	950509025	4
33	950509029	4
34	950509031	4
35	950509032	4
36	950509033	4
37	950509034	4
38	950509038	4
39	950509039	4
40	950509042	4
41	950509045	4
42	950509052	4

	NR	YEARS
43	960509001	4
44	960509002	4
45	960509003	4
46	960509004	4
47	960509007	4
48	960509008	4
49	960509010	4
50	960509011	4
51	960509012	4
52	960509013	4
53	960509014	4
54	960509015	4
55	960509016	4
56	960509019	4
57	960509021	4
58	960509024	4
59	960509025	4
60	960509032	4
61	960509035	4
62	960509039	4
63	960509040	4

	NR	YEARS
64	960509041	4
65	960509042	4
66	960509043	4
67	960509045	4
68	960509046	4
69	960509054	4
70	970509008	4
71	970509050	2
72	980509039	2

1.3. Planos de execução

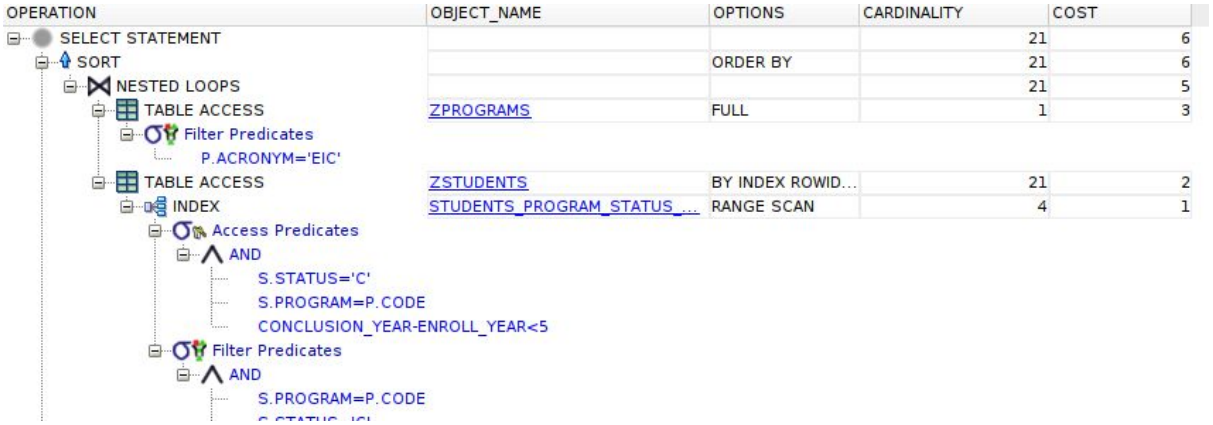
1.3.1. Ambiente X



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			83	18
SORT		ORDER BY	83	18
HASH JOIN			83	17
Access Predicates				
S.PROGRAM=P.CODE				
TABLE ACCESS	XPROGRAMS	FULL	1	3
Filter Predicates				
P.ACRONYM='EIC'				
TABLE ACCESS	XSTUDENTS	FULL	908	14
Filter Predicates				
AND				
S.STATUS='C'				
S.CONCLUSION_YEAR-S.ENROLL_YEAR<5				

1.3.2. Ambiente Y

1.3.3. Ambiente Z



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			21	6
SORT		ORDER BY	21	6
NESTED LOOPS			21	5
TABLE ACCESS	ZPROGRAMS	FULL	1	3
Filter Predicates				
P.ACRONYM='EIC'				
TABLE ACCESS	ZSTUDENTS	BY INDEX ROWID...	21	2
INDEX	STUDENTS_PROGRAM_STATUS_...	RANGE SCAN	4	1
Access Predicates				
AND				
S.STATUS='C'				
S.PROGRAM=P.CODE				
CONCLUSION_YEAR-ENROLL_YEAR<5				
Filter Predicates				
AND				
S.PROGRAM=P.CODE				

1.4. Tempos de execução

Ambiente	X	Y	Z
Tempo (ms)	34.5	31.9	29.0

1.5. Conclusões

Query simples cujo o problema predominante existente são as comparações usadas para obtermos o resultado desejado. Estas são as principais causas do custo e da cardinalidade da *query*.

Em termos dos ambientes X e Y não existem diferenças notáveis visto que as restrições aplicadas não afetam a *query* mas só o *join*. É possível, por conseguinte, ver uma pequena melhoria no tempo de execução do *join* entre as tabelas *students* e *programs*.

O uso de índice ***`students_program_status_conclusion_enroll`*** melhora bastante o desempenho da *query*. O custo da *query* foi reduzido substancialmente no ambiente Z, comparativamente aos ambientes X e Y.

Resumindo, como os tempos de execução melhoram relativamente aos ambientes anteriores o uso das restrições e dos índices não só melhoram a execução da *query*, como também baixam a cardinalidade e o custo de execução.

Pergunta 2 - Agregação

Qual a média mínima de candidatura em cada curso, em cada ano, dos alunos matriculados? Nem todas as candidaturas têm a média preenchida.

2.1. Formulação SQL

```
select  c.program, c.year, min(c.average) as minAverage
from xcandidates c
where c.average is not null
and c.result = 'C'
group by c.program, c.year
order by c.program, c.year
```

2.2. Resultados

	PROGRAM	YEAR	MINAVERAGE
1	233	1986	15
2	233	1998	10
3	233	2000	13.28
4	233	2001	10
5	233	2002	12.75
6	255	1989	5.5
7	255	1990	6.4
8	255	1991	12.54
9	255	1994	13
10	255	1995	12
11	255	1996	14.08
12	255	1997	13
13	255	1998	11
14	255	1999	14.8
15	255	2000	11.43
16	255	2001	11
17	255	2002	13.43
18	275	1998	10
19	275	2000	11.5
20	275	2001	10.2
21	275	2002	13.78
22	304	1993	12

	PROGRAM	YEAR	MINAVERAGE
23	304	1994	13
24	304	1998	12
25	304	2000	10.9
26	304	2001	10.2
27	304	2002	12.23
28	315	1997	12.83
29	315	1998	12
30	315	2000	10.65
31	315	2001	11.15
32	315	2002	10.88
33	318	1991	7.5
34	331	1996	12
35	331	1997	15.88
36	331	1998	11
37	331	2000	10.38
38	331	2001	14.2
39	331	2002	11
40	433	1998	15
41	433	2000	14.45

	PROGRAM	YEAR	MINAVERAGE
42	433	2001	14.6
43	433	2002	14.38
44	649	1998	12
45	649	2000	10.75
46	649	2001	10.7
47	649	2002	11.3
48	1093	2002	12.68

2.3. Planos de execução

2.3.1. Ambiente X

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			249	16
SORT		GROUP BY	249	16
TABLE ACCESS	XCANDIDATES	FULL	2954	15
Filter Predicates				
AND				
AVERAGE IS NOT NULL				
C.RESULT='C'				

2.3.2. Ambiente Y

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			249	16
SORT		GROUP BY	249	16
TABLE ACCESS	YCANDIDATES	FULL	1070	15
Filter Predicates				
AND				
AVERAGE IS NOT NULL				
C.RESULT='C'				

2.3.3. Ambiente Z

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			263	16
SORT		GROUP BY	263	16
TABLE ACCESS	ZCANDIDATES	FULL	1070	15
Filter Predicates				
AND				
AVERAGE IS NOT NULL				
C.RESULT='C'				

2.4. Tempos de execução

Ambiente	X	Y	Z
Tempo (ms)	33.1	34.1	31.5

2.5. Conclusões

O objetivo desta *query* é usar a propriedade de agrupamento de dados do *SQL* para poder obter um valor mínimo. Consequentemente, o custo da *query* está presente no acesso à tabela.

Em termos de custo, nos 3 ambientes não existe diferença visto o plano de execução não ser afetado. No entanto, as restrições aplicadas à tabela de *candidates* diminuem o valor da cardinalidade. Isto acontece porque os valores que estamos a agrupar estão presentes na chave primária da tabela.

Concluindo, a cardinalidade desta *query* é reduzida através do uso de chaves primárias mas o seu custo e o tempo de execução (diferença considerada insignificante) não são afetadas. O otimizador não aplicou índices visto que a *query* só acede a atributos da chave primária.

Pergunta 3

Considere a questão de saber quantos candidatos aceites não se matricularam nesse ano lectivo. Compare uma formulação que use uma subpergunta constante com a equivalente que use uma subpergunta variável (sugestão: usar EXISTS).

3.1. Formulação SQL

3.1.1. Subpergunta Constante

```
select c.year, count(*) as notEnrolled
from xcandidates c
where result = 'C' and
(c.id, c.program, c.year) NOT IN
(
   select s.id, s.program, s.enroll_year
   from xstudents s
)
group by c.year
order by c.year
```

3.1.2. Subpergunta Variável

```
select c.year, count(*) as notEnrolled
from xcandidates c
where result = 'C' and
NOT EXISTS
(
   select s.enroll_year
   from xstudents s
   where c.id = s.id and c.year = s.enroll_year and c.program =
   s.program
)
group by c.year
order by c.year
```

3.2. Resultados

	YEAR	NOTENROLLED
1	1972	2
2	1973	2
3	1974	3
4	1975	8
5	1976	14
6	1977	3
7	1978	8
8	1979	12
9	1980	10
10	1981	31
11	1982	21
12	1983	16
13	1984	26
14	1985	29
15	1986	46
16	1987	60

	YEAR	NOTENROLLED
17	1988	96
18	1989	145
19	1990	143
20	1991	222
21	1992	232
22	1993	296
23	1994	264
24	1995	333
25	1996	354
26	1997	370
27	1998	344
28	1999	340
29	2000	300
30	2001	216
31	2002	241

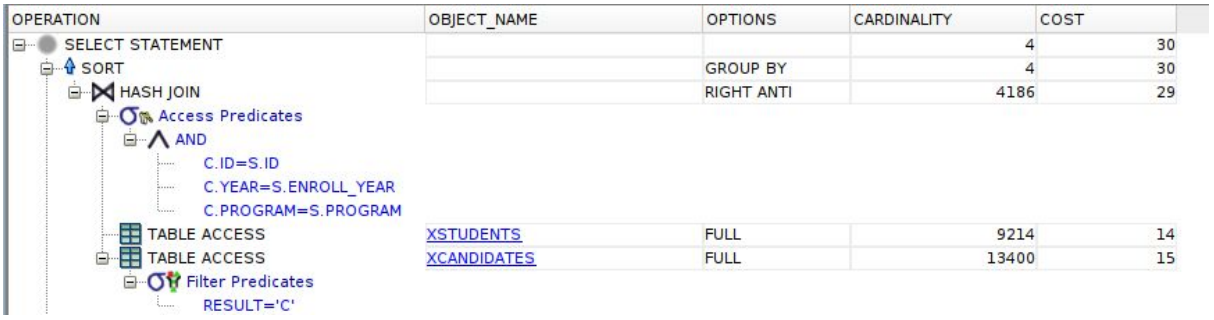
3.3. Planos de execução

3.3.1 Ambiente X

3.3.1.1. Subpergunta Constante

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	33
SORT		GROUP BY	1	33
MERGE JOIN		ANTI NA	4186	32
SORT		JOIN	13400	16
TABLE ACCESS	XCANDIDATES	FULL	13400	15
Filter Predicates				
RESULT='C'				
SORT		UNIQUE	9214	15
Access Predicates				
AND				
INTERNAL_FUNCTION(C.ID)=INTERNAL_FUNCTION(S.ID)				
INTERNAL_FUNCTION(C.PROGRAM)=INTERNAL_FUNCTION(S.PROGRAM)				
INTERNAL_FUNCTION(C.YEAR)=INTERNAL_FUNCTION(S.ENROLL_YEAR)				
Filter Predicates				
AND				
INTERNAL_FUNCTION(C.YEAR)=INTERNAL_FUNCTION(S.ENROLL_YEAR)				
INTERNAL_FUNCTION(C.PROGRAM)=INTERNAL_FUNCTION(S.PROGRAM)				
INTERNAL_FUNCTION(C.ID)=INTERNAL_FUNCTION(S.ID)				
TABLE ACCESS	XSTUDENTS	FULL	9214	14

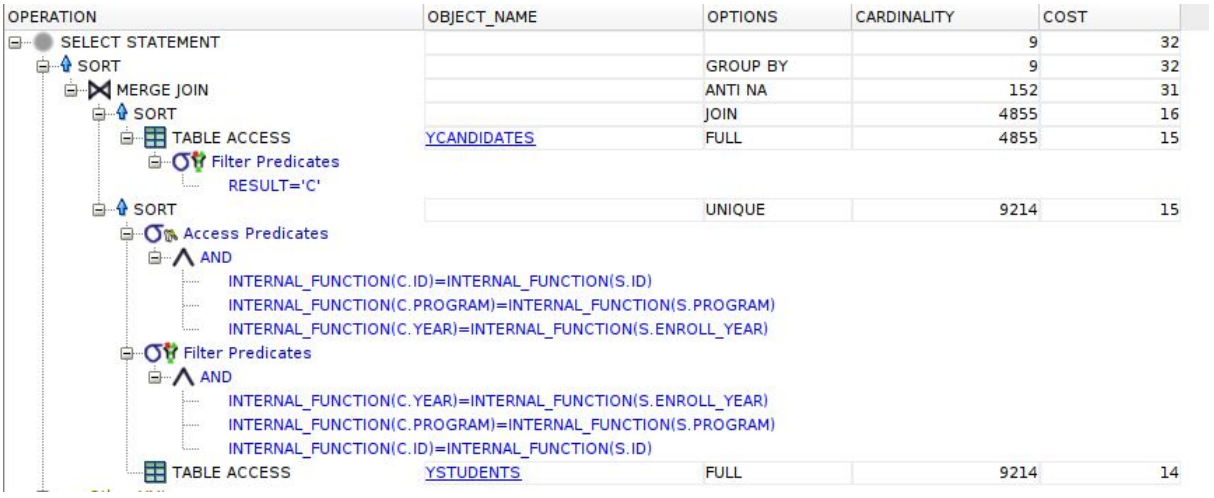
3.3.1.2. Subpergunta Variável



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			4	30
SORT		GROUP BY	4	30
HASH JOIN		RIGHT ANTI	4186	29
Access Predicates				
AND				
C.ID=S.ID				
C.YEAR=S.ENROLL_YEAR				
C.PROGRAM=S.PROGRAM				
TABLE ACCESS	XSTUDENTS	FULL	9214	14
TABLE ACCESS	XCANDIDATES	FULL	13400	15
Filter Predicates				
RESULT='C'				

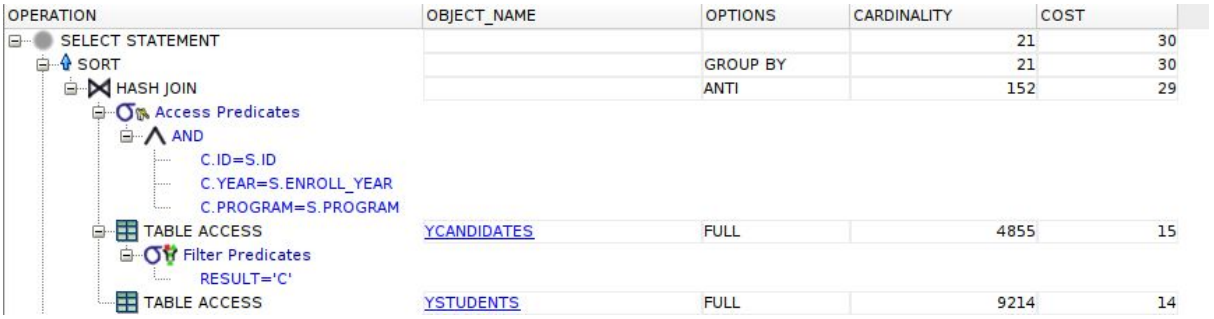
3.3.2 Ambiente Y

3.3.2.1. Subpergunta Constante



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			9	32
SORT		GROUP BY	9	32
MERGE JOIN		ANTI NA	152	31
SORT		JOIN	4855	16
TABLE ACCESS	YCANDIDATES	FULL	4855	15
Filter Predicates				
RESULT='C'				
SORT		UNIQUE	9214	15
Access Predicates				
AND				
INTERNAL_FUNCTION(C.ID)=INTERNAL_FUNCTION(S.ID)				
INTERNAL_FUNCTION(C.PROGRAM)=INTERNAL_FUNCTION(S.PROGRAM)				
INTERNAL_FUNCTION(C.YEAR)=INTERNAL_FUNCTION(S.ENROLL_YEAR)				
Filter Predicates				
AND				
INTERNAL_FUNCTION(C.YEAR)=INTERNAL_FUNCTION(S.ENROLL_YEAR)				
INTERNAL_FUNCTION(C.PROGRAM)=INTERNAL_FUNCTION(S.PROGRAM)				
INTERNAL_FUNCTION(C.ID)=INTERNAL_FUNCTION(S.ID)				
TABLE ACCESS	YSTUDENTS	FULL	9214	14

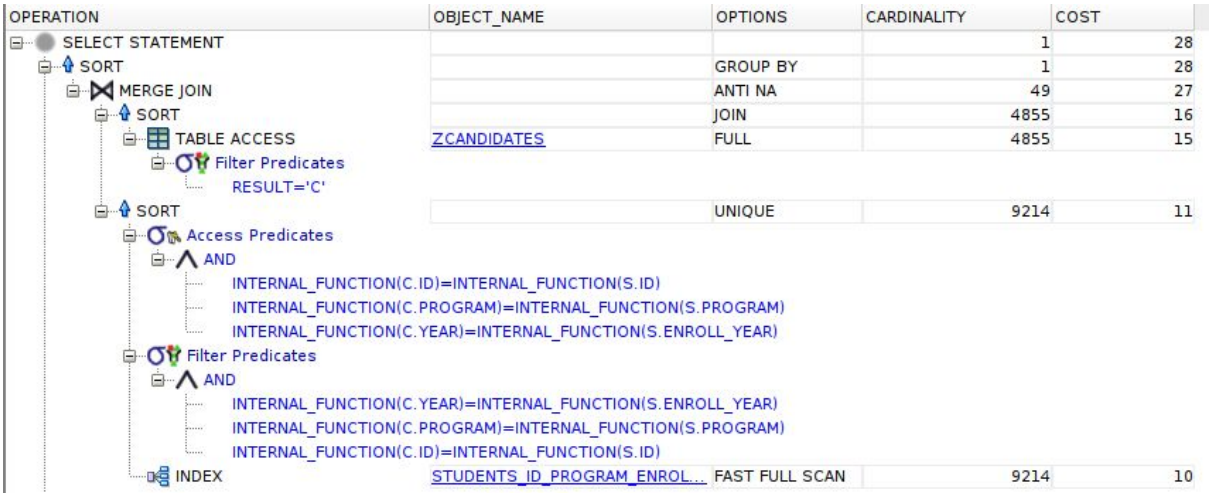
3.3.2.2. Subpergunta Variável



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			21	30
SORT		GROUP BY	21	30
HASH JOIN		ANTI	152	29
Access Predicates				
AND				
C.ID=S.ID				
C.YEAR=S.ENROLL_YEAR				
C.PROGRAM=S.PROGRAM				
TABLE ACCESS	YCANDIDATES	FULL	4855	15
Filter Predicates				
RESULT='C'				
TABLE ACCESS	YSTUDENTS	FULL	9214	14

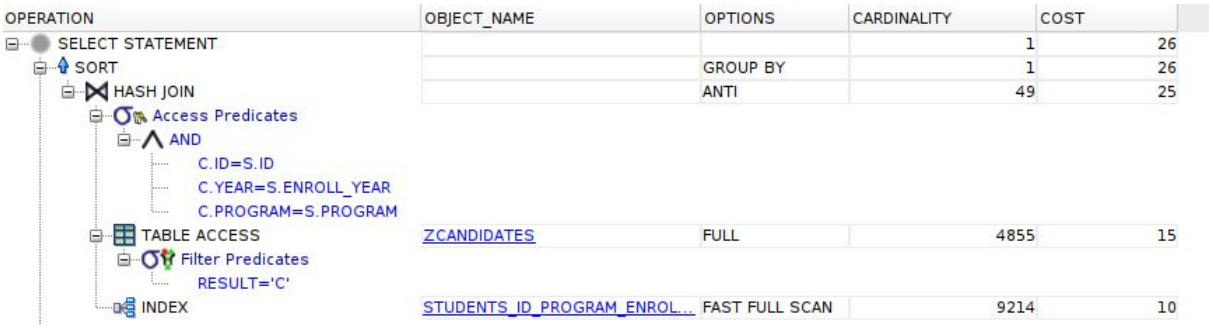
3.3.3 Ambiente Z

3.3.3.1. Subpergunta Constante



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	28
SORT		GROUP BY	1	28
MERGE JOIN		ANTI NA	49	27
SORT		JOIN	4855	16
TABLE ACCESS	ZCANDIDATES	FULL	4855	15
Filter Predicates				
RESULT='C'				
SORT		UNIQUE	9214	11
Access Predicates				
AND				
INTERNAL_FUNCTION(C.ID)=INTERNAL_FUNCTION(S.ID)				
INTERNAL_FUNCTION(C.PROGRAM)=INTERNAL_FUNCTION(S.PROGRAM)				
INTERNAL_FUNCTION(C.YEAR)=INTERNAL_FUNCTION(S.ENROLL_YEAR)				
Filter Predicates				
AND				
INTERNAL_FUNCTION(C.YEAR)=INTERNAL_FUNCTION(S.ENROLL_YEAR)				
INTERNAL_FUNCTION(C.PROGRAM)=INTERNAL_FUNCTION(S.PROGRAM)				
INTERNAL_FUNCTION(C.ID)=INTERNAL_FUNCTION(S.ID)				
INDEX	STUDENTS_ID_PROGRAM_ENROL...	FAST FULL SCAN	9214	10

3.3.3.2. Subpergunta Variável



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	26
SORT		GROUP BY	1	26
HASH JOIN		ANTI	49	25
Access Predicates				
AND				
C.ID=S.ID				
C.YEAR=S.ENROLL_YEAR				
C.PROGRAM=S.PROGRAM				
TABLE ACCESS	ZCANDIDATES	FULL	4855	15
Filter Predicates				
RESULT='C'				
INDEX	STUDENTS_ID_PROGRAM_ENROL...	FAST FULL SCAN	9214	10

3.4. Tempos de execução

3.4.2. Subpergunta Constante

Ambiente	X	Y	Z
Tempo (ms)	55.1	54.0	44.5

3.4.2. Subpergunta Variável

Ambiente	X	Y	Z
Tempo (ms)	38.4	43.1	36.7

3.5. Conclusões

O objetivo desta pergunta consiste em comparar as diferenças entre uma *query* com uma subpergunta constante e uma subpergunta variável. Para o constante foi usado o ***not in*** e para a variável foi usado ***not exists***.

Em termos de performance, o efeito mais notável verifica-se no ambiente Z, sendo que a adição de restrições de integridade *standard* não revela um impacto significativo em termos de custo ou plano de execução. O uso do índice ***students_id_program_enroll_year*** (índice composto do tipo *B-tree* nas colunas *id*, *program* e *enroll_year* da tabela *students*) nesta situação melhora a performance da *query*. Em vez de um *full scan* na tabela de *students* é feito um *fast full scan,* melhorando o custo desse passo e consequentemente melhorando os tempos de execução em ambas as *queries* em análise.

Comparando as 2 *queries* feitas, o uso de *exists* diminui a cardinalidade e melhora o tempo de execução da *query*.

Deste modo, conclui-se que a subpergunta variável em conjunto com os índices adicionados (no ambiente Z) constituiu uma solução mais eficiente do problema.

Pergunta 4

Estude as tentativas de resposta à questão "Qual o curso do aluno com a melhor média de conclusão em cada ano lectivo" apresentadas abaixo. Comente-as.

4.1. Formulação SQL

4.1.1. Versão 1

```
with aux as
(
   select s.conclusion_year, s.program, max(s.final_average) as maxAvg
   from xstudents s
   where s.final_average is not null
   group by s.conclusion_year, s.program
   order by s.conclusion_year
)

select q1.conclusion_year, q1.program, q2.result
from aux q1,
(
```

```
   select temp.conclusion_year, max(temp.maxAvg) as result
   from aux temp
  group by temp.conclusion_year
  order by temp.conclusion_year
) q2
where q1.conclusion_year = q2.conclusion_year and
q1.maxAvg = q2.result
```

4.1.2. Versão 2

```
Select distinct s1.conclusion_year,p.code,s2.max_average as result
From xprograms p, xstudents s1,
(
   select s.conclusion_year,max(s.final_average) as max_average
   from xstudents s
   where s.status = 'C'
   group by s.conclusion_year
) s2
Where p.code = s1.program
   and s1.conclusion_year = s2.conclusion_year
   and s1.final_average = s2.max_average
Order by s1.conclusion_year
```

4.2. Resultados

	CONCLUSION_YEAR	PROGRAM	RESULT
1	1988	304	11
2	1990	233	12
3	1992	255	18
4	1993	331	18
5	1994	255	18
6	1994	331	18
7	1994	433	18
8	1995	304	18
9	1996	233	17
10	1996	255	17
11	1996	433	17
12	1997	255	17
13	1997	304	17
14	1997	331	17
15	1998	255	19
16	1999	304	18
17	2000	304	18
18	2001	233	17.25

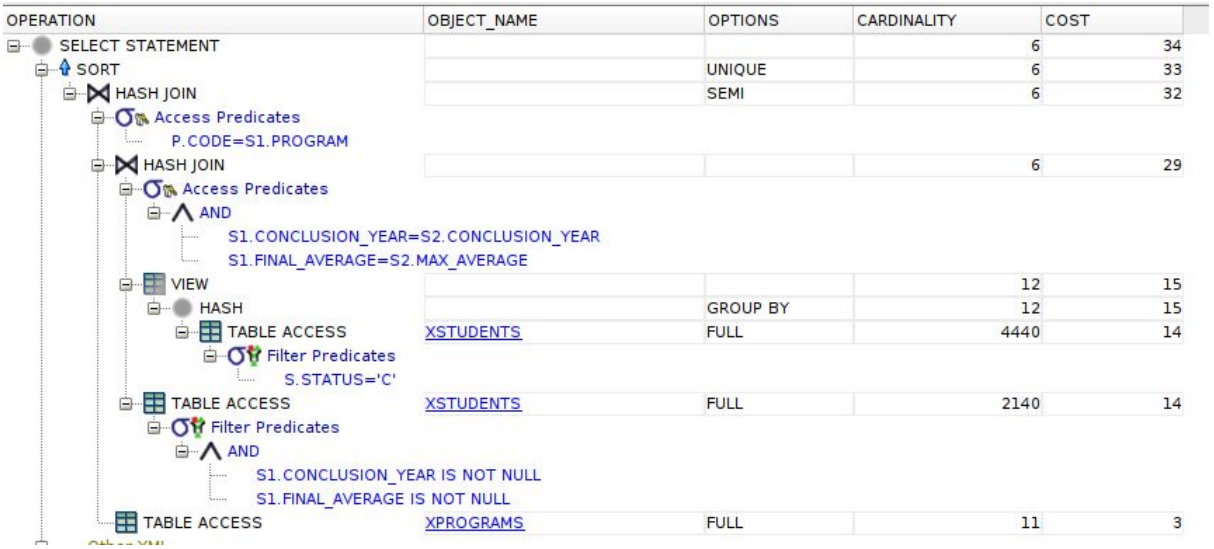
4.3. Planos de execução

4.3.1. Ambiente X

4.3.1.1. Versão 1

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	20
TEMP TABLE TRANSFORMATION				
LOAD AS SELECT	SYS_TEMP_0FD9D9FA8_446CC2CF	(CURSOR DURAT...		
SORT		GROUP BY	94	15
TABLE ACCESS	XSTUDENTS	FULL	4440	14
Filter Predicates				
S.FINAL_AVERAGE IS NOT NULL				
HASH JOIN			1	5
Access Predicates				
AND				
Q1.CONCLUSION_YEAR=Q2.CONCLUSION_YEAR				
Q1.MAXAVG=Q2.RESULT				
VIEW			12	3
SORT		GROUP BY	12	3
VIEW			94	2
TABLE ACCESS	SYS.SYS_TEMP_0FD9D9FA8_446...	FULL	94	2
VIEW			94	2
TABLE ACCESS	SYS.SYS_TEMP_0FD9D9FA8_446...	FULL	94	2

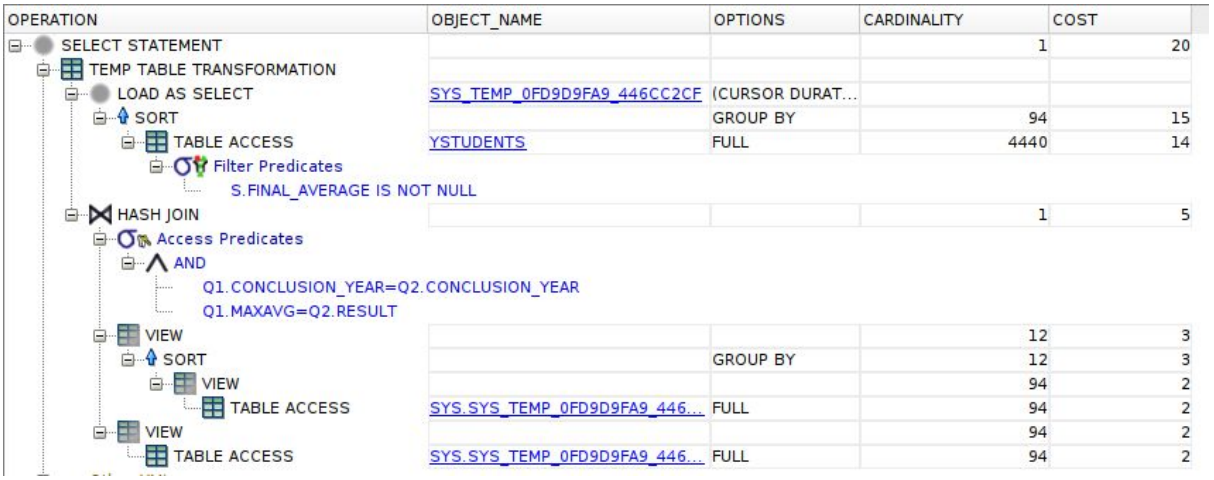
4.3.1.2. Versão 2



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			6	34
SORT		UNIQUE	6	33
HASH JOIN		SEMI	6	32
Access Predicates				
P.CODE=S1.PROGRAM				
HASH JOIN			6	29
Access Predicates				
AND				
S1.CONCLUSION_YEAR=S2.CONCLUSION_YEAR				
S1.FINAL_AVERAGE=S2.MAX_AVERAGE				
VIEW			12	15
HASH		GROUP BY	12	15
TABLE ACCESS	XSTUDENTS	FULL	4440	14
Filter Predicates				
S.STATUS='C'				
TABLE ACCESS	XSTUDENTS	FULL	2140	14
Filter Predicates				
AND				
S1.CONCLUSION_YEAR IS NOT NULL				
S1.FINAL_AVERAGE IS NOT NULL				
TABLE ACCESS	XPROGRAMS	FULL	11	3

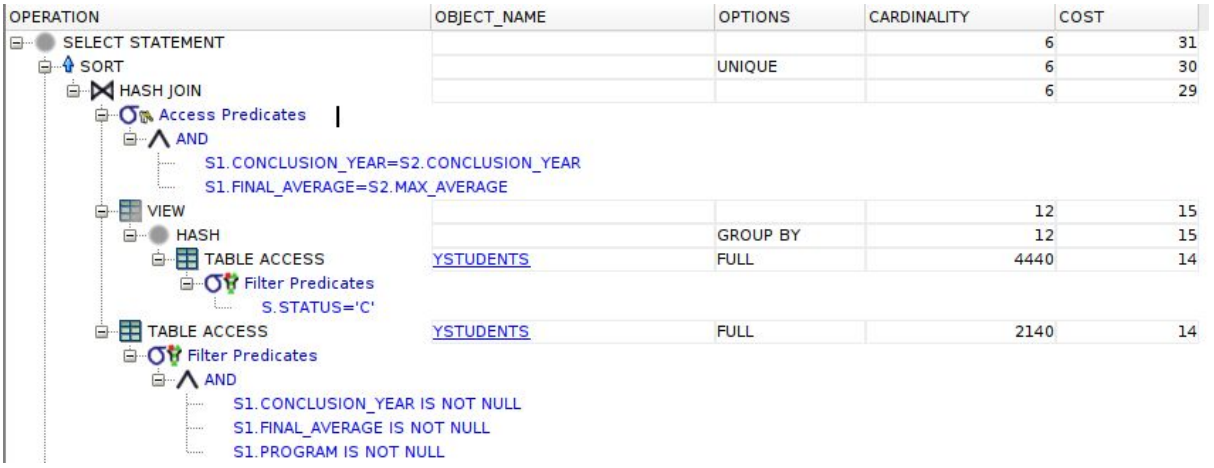
4.3.2. Ambiente Y

4.3.2.1. Versão 1



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	20
TEMP TABLE TRANSFORMATION				
LOAD AS SELECT	SYS_TEMP_0FD9D9FA9_446CC2CF	(CURSOR DURAT...		
SORT		GROUP BY	94	15
TABLE ACCESS	YSTUDENTS	FULL	4440	14
Filter Predicates				
S.FINAL_AVERAGE IS NOT NULL				
HASH JOIN			1	5
Access Predicates				
AND				
Q1.CONCLUSION_YEAR=Q2.CONCLUSION_YEAR				
Q1.MAXAVG=Q2.RESULT				
VIEW			12	3
SORT		GROUP BY	12	3
VIEW			94	2
TABLE ACCESS	SYS.SYS_TEMP_0FD9D9FA9_446...	FULL	94	2
VIEW			94	2
TABLE ACCESS	SYS.SYS_TEMP_0FD9D9FA9_446...	FULL	94	2

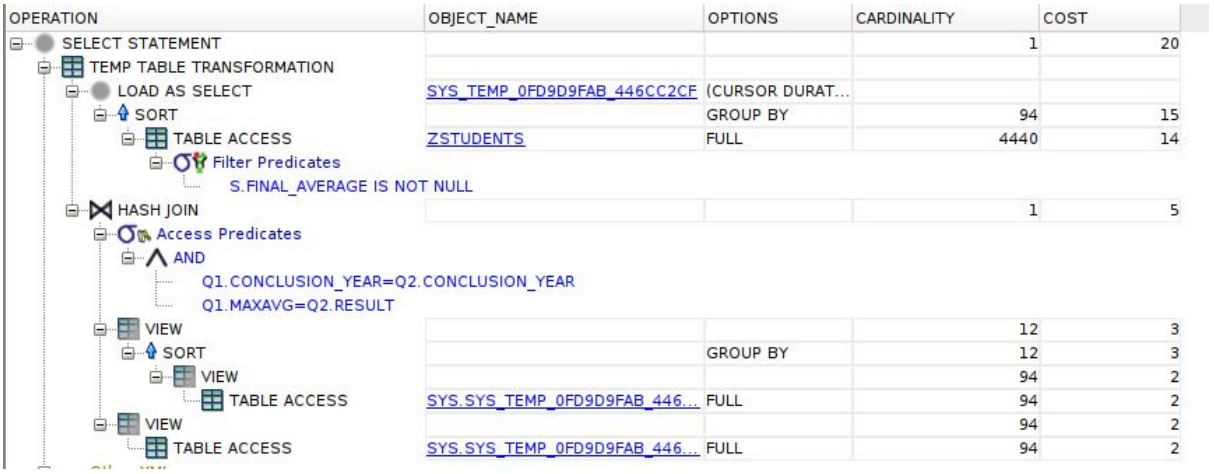
4.3.2.2. Versão 2



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			6	31
SORT		UNIQUE	6	30
HASH JOIN			6	29
Access Predicates				
AND				
S1.CONCLUSION_YEAR=S2.CONCLUSION_YEAR				
S1.FINAL_AVERAGE=S2.MAX_AVERAGE				
VIEW			12	15
HASH		GROUP BY	12	15
TABLE ACCESS	YSTUDENTS	FULL	4440	14
Filter Predicates				
S.STATUS='C'				
TABLE ACCESS	YSTUDENTS	FULL	2140	14
Filter Predicates				
AND				
S1.CONCLUSION_YEAR IS NOT NULL				
S1.FINAL_AVERAGE IS NOT NULL				
S1.PROGRAM IS NOT NULL				

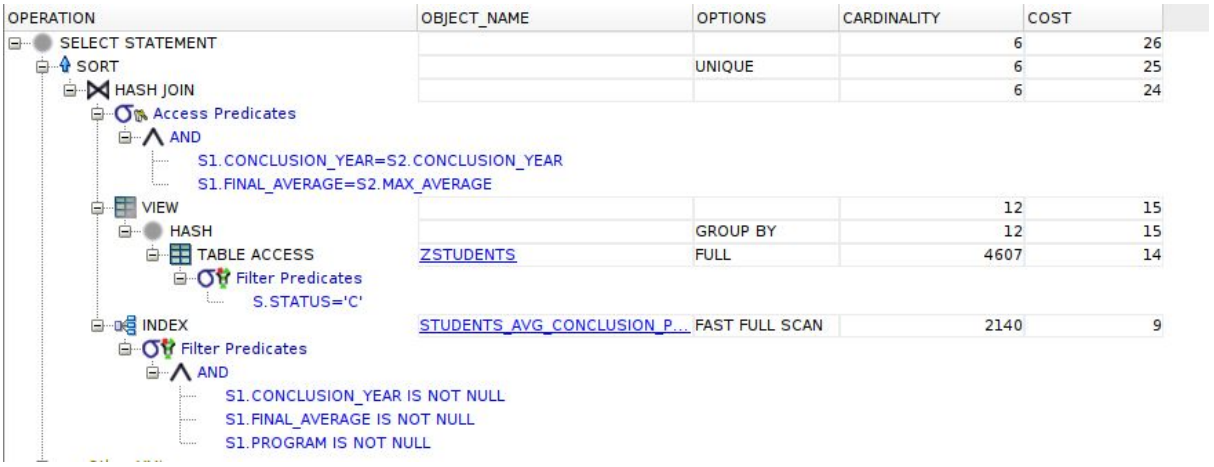
4.3.3. Ambiente Z

4.3.3.1. Versão 1



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	20
TEMP TABLE TRANSFORMATION				
LOAD AS SELECT	SYS_TEMP_0FD9D9FAB_446CC2CF	(CURSOR DURAT...		
SORT		GROUP BY	94	15
TABLE ACCESS	ZSTUDENTS	FULL	4440	14
Filter Predicates				
S.FINAL_AVERAGE IS NOT NULL				
HASH JOIN			1	5
Access Predicates				
AND				
Q1.CONCLUSION_YEAR=Q2.CONCLUSION_YEAR				
Q1.MAXAVG=Q2.RESULT				
VIEW			12	3
SORT		GROUP BY	12	3
VIEW			94	2
TABLE ACCESS	SYS.SYS_TEMP_0FD9D9FAB_446...	FULL	94	2
VIEW			94	2
TABLE ACCESS	SYS.SYS_TEMP_0FD9D9FAB_446...	FULL	94	2

4.3.3.2. Versão 2



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			6	26
SORT		UNIQUE	6	25
HASH JOIN			6	24
Access Predicates				
AND				
S1.CONCLUSION_YEAR=S2.CONCLUSION_YEAR				
S1.FINAL_AVERAGE=S2.MAX_AVERAGE				
VIEW			12	15
HASH		GROUP BY	12	15
TABLE ACCESS	ZSTUDENTS	FULL	4607	14
Filter Predicates				
S.STATUS='C'				
INDEX	STUDENTS_AVG_CONCLUSION_P...	FAST FULL SCAN	2140	9
Filter Predicates				
AND				
S1.CONCLUSION_YEAR IS NOT NULL				
S1.FINAL_AVERAGE IS NOT NULL				
S1.PROGRAM IS NOT NULL				

4.4. Tempos de execução

4.4.1. Versão 1

Ambiente	X	Y	Z
Tempo (ms)	37.3	37.7	34.8

4.4.2. Versão 2

Ambiente	X	Y	Z
Tempo (ms)	35.4	32.6	32.7

4.5. Conclusões

Nesta questão foram aplicados dois métodos diferente de resolução do problema, um que cria uma tabela temporária (versão 1) e outro que usa *subqueries* (versão 2).

Comparando as 2 *queries* em termos de custo e cardinalidade a versão 1 mostra melhores resultados relativamente a versão 2, nos três ambientes em análise.

O uso de restrições de integridade *standard* afeta de forma mais significativa a versão 2 do problema visto que ser retirado um acesso à tabela *programs*. Também o uso de índices tem mais impacto na versão 2 do programa passando a usar um *fast full scan* em vez de *full scan* na tabela de *students*.

Os tempos de execução são melhorados no ambiente Y e ainda mais no Z para a versão 2, no entanto para a 1 a performance só é melhorada no ambiente Z (talvez devido a fatores externos e/ou à quantidade de dados analisados).

Deste modo, a análise das duas versões permite concluir que o uso de índices e restrições tem mais efeito dependendo do tipo de *query* que fazemos para o mesmo resultado final.

Pergunta 5

Compare os planos de execução da pesquisa "Quantos candidatos tiveram como resultado algo diferente de "C" ou "E", usando, no contexto Z

a. Com índice árvore-B em Resultado;

b. Com índice bitmap em Resultado.

5.1. Formulação SQL

```
select count(*) as nrCandidates
from xcandidates c
where c.result != 'C'
and c.result != 'E'
```

5.2. Resultados

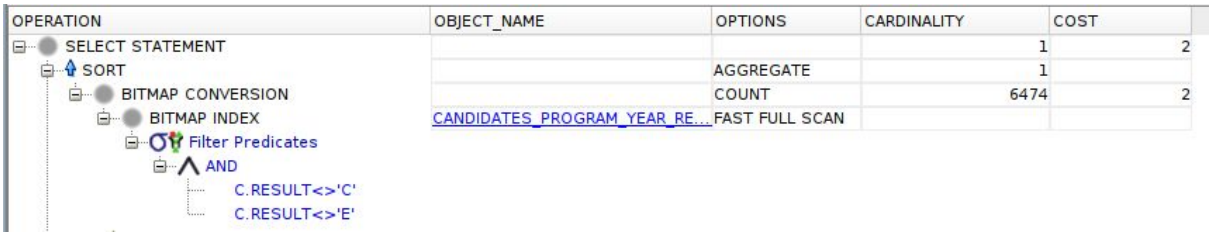
	NRCANDIDATES
1	618

5.3. Planos de execução

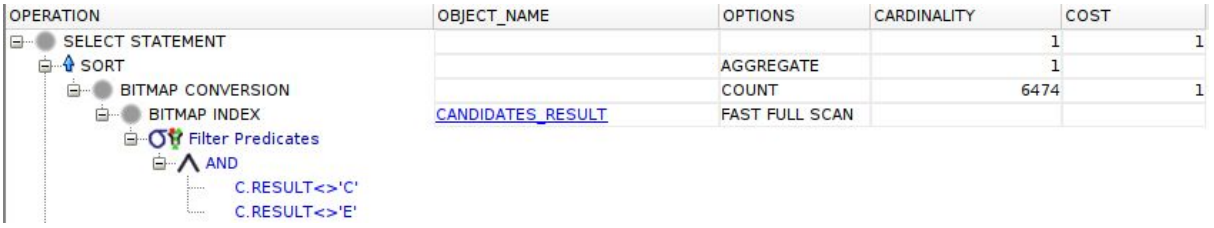
5.3.1. Ambiente X

5.3.2. Ambiente Y

5.3.3. Ambiente Z árvore B



5.3.4. Ambiente Z *bitmap*



5.4. Tempos de execução

Ambiente	X	Y	Z árvore B	Z bitmap
Tempo (ms)	27.7	29.2	26.9	26.4

5.5. Conclusões

A *query* que responde a esta pergunta é bastante simples, o que se revela no seu plano de execução nos ambientes X e Y, que são feitos sem nenhuma otimização. Uma vez que a *query* não utiliza nenhuma das chaves primárias definidas, as restrições de integridade *standard* presentes no ambiente Y não revelam um impacto significativo no plano de execução.

Por outro lado, a adição de um índice na coluna *result* revela-se bastante eficiente. No caso do índice do tipo *B-tree* o custo foi reduzido para 2, enquanto no caso do índice *bitmap* este passou a ser 1. Conclui-se que o índice *bitmap* tem um impacto mais significativo no plano de execução, pelo facto de a coluna onde é aplicado ter uma cardinalidade bastante baixa (apenas três valores possíveis).

Relativamente aos tempos de execução obtidos, nota-se de facto uma pequena melhoria nos ambientes Z em relação ao ambientes X e Y. O tempo obtido no ambiente Y é superior ao obtido no X, no entanto, sendo que não existem implicações das restrições de integridade *standard,* esta diferença não é considerada significativa (talvez devido a fatores externos e/ou à quantidade de dados analisados).

Pergunta 6

A pergunta "Há, em algum ano algum curso (ano_lectivo, sigla e nome) que tenha todas as candidaturas aceites transformadas em matrículas, nesse mesmo ano?" é de natureza universal. Compare do ponto de vista temporal e de plano de execução as estratégias da dupla negação e da contagem.

6.1. Formulação SQL

6.1.1. Contagem

```
select candidates.year, p.acronym, p.designation
from
(
   select s.enroll_year, s.program, count(*) nr
   from xstudents s
   group by s.enroll_year, s.program
) students,
(
   select c.year, c.program, count(*) nr
```

```
   from xcandidates c
   where c.result = 'C'
   group by c.year, c.program
) candidates,
xprograms p
where students.enroll_year = candidates.year
and students.program = candidates.program
and students.nr = candidates.nr
and p.code = students.program
order by candidates.year, candidates.program
```

6.1.2. Dupla negação

```
select candidates.year, p.acronym, p.designation
from xcandidates candidates, xprograms p
where p.code = candidates.program and
candidates.result = 'C' and
(program, year) not in
(
   select c.program, c.year
   from xcandidates c
   where c.result = 'C'
   and not exists
   (
         select s.enroll_year, s.program
         from xstudents s
         where s.enroll_year = c.year
         and s.program = c.program
         and s.id = c.id
   )
)
group by candidates.year, p.acronym, p.designation
order by candidates.year, p.acronym
```

6.2. Resultados

	YEAR	ACRONYM	DESIGNATION
1	1970	EC	Engenharia Civil
2	1972	EC	Engenharia Civil
3	1974	EM	Engenharia Mecânica
4	1977	EM	Engenharia Mecânica
5	1980	EM	Engenharia Mecânica
6	1981	EMT	Engenharia Metalúrgica
7	1982	EMM	Engenharia Metalúrgica e de Materiais
8	1983	EMM	Engenharia Metalúrgica e de Materiais
9	1984	EMM	Engenharia Metalúrgica e de Materiais
10	1996	EMG	Engenharia de Minas e Geoambiente
11	2001	EMI	Engenharia de Minas

6.3. Planos de execução

6.3.1. Ambiente X

6.3.1.1. Contagem

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			8	36
SORT		ORDER BY	8	36
HASH JOIN			8	35
Access Predicates				
P.CODE=STUDENTS.PROGRAM				
HASH JOIN			8	32
Access Predicates				
AND				
STUDENTS.ENROLL_YEAR=CANDIDATES.YEAR				
STUDENTS.PROGRAM=CANDIDATES.PROGRAM				
STUDENTS.NR=CANDIDATES.NR				
VIEW			242	15
HASH		GROUP BY	242	15
TABLE ACCESS	XSTUDENTS	FULL	9214	14
VIEW			249	16
HASH		GROUP BY	249	16
TABLE ACCESS	XCANDIDATES	FULL	13400	15
Filter Predicates				
C.RESULT='C'				
TABLE ACCESS	XPROGRAMS	FULL	11	3

6.3.1.2. Dupla negação

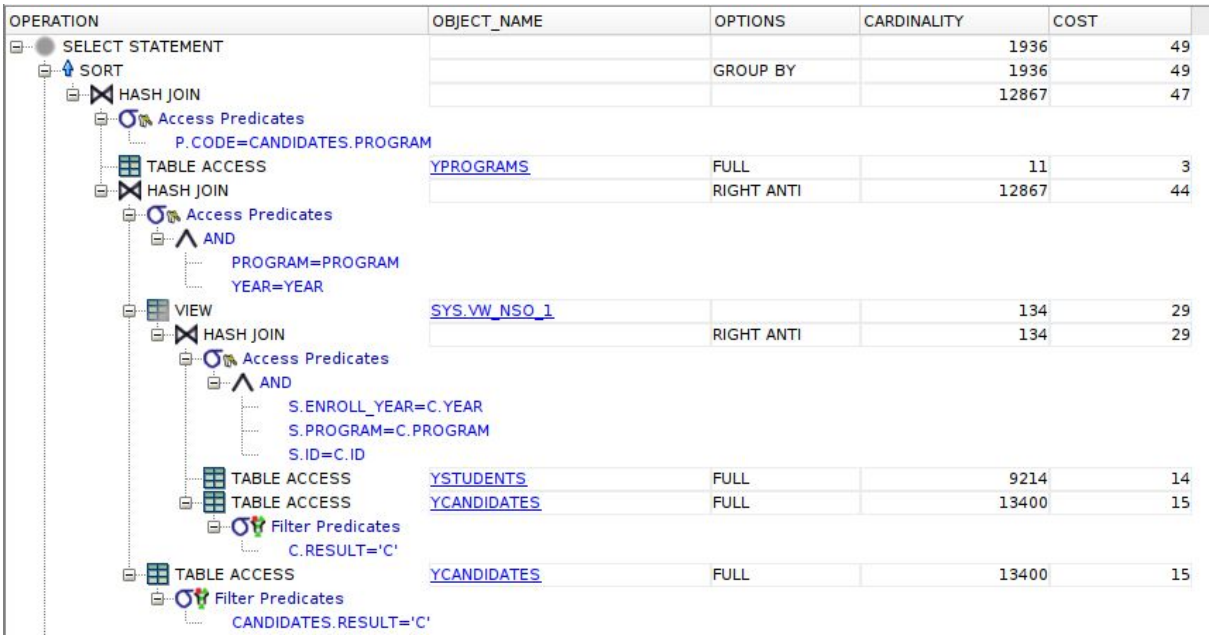
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1936	49
SORT		GROUP BY	1936	49
HASH JOIN			12867	47
Access Predicates				
P.CODE=CANDIDATES.PROGRAM				
TABLE ACCESS	XPROGRAMS	FULL	11	3
HASH JOIN		RIGHT ANTI	12867	44
Access Predicates				
AND				
PROGRAM=PROGRAM				
YEAR=YEAR				
VIEW	SYS.VW_NSO_1		134	29
HASH JOIN		RIGHT ANTI	134	29
Access Predicates				
AND				
S.ENROLL_YEAR=C.YEAR				
S.PROGRAM=C.PROGRAM				
S.ID=C.ID				
TABLE ACCESS	XSTUDENTS	FULL	9214	14
TABLE ACCESS	XCANDIDATES	FULL	13400	15
Filter Predicates				
C.RESULT='C'				
TABLE ACCESS	XCANDIDATES	FULL	13400	15
Filter Predicates				
CANDIDATES.RESULT='C'				

6.3.2. Ambiente Y

6.3.2.1. Contagem

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	34
SORT		ORDER BY	2	34
NESTED LOOPS			2	33
NESTED LOOPS			2	33
HASH JOIN			2	31
Access Predicates				
AND				
STUDENTS.ENROLL_YEAR=CANDIDATES.YEAR				
STUDENTS.PROGRAM=CANDIDATES.PROGRAM				
STUDENTS.NR=CANDIDATES.NR				
VIEW			242	15
HASH		GROUP BY	242	15
TABLE ACCESS	YSTUDENTS	FULL	9214	14
VIEW			249	16
HASH		GROUP BY	249	16
TABLE ACCESS	YCANDIDATES	FULL	4855	15
Filter Predicates				
C.RESULT='C'				
INDEX	YPROGRAMS_CODE_PK	UNIQUE SCAN	1	0
Access Predicates				
P.CODE=STUDENTS.PROGRAM				
TABLE ACCESS	YPROGRAMS	BY INDEX ROWID	1	1

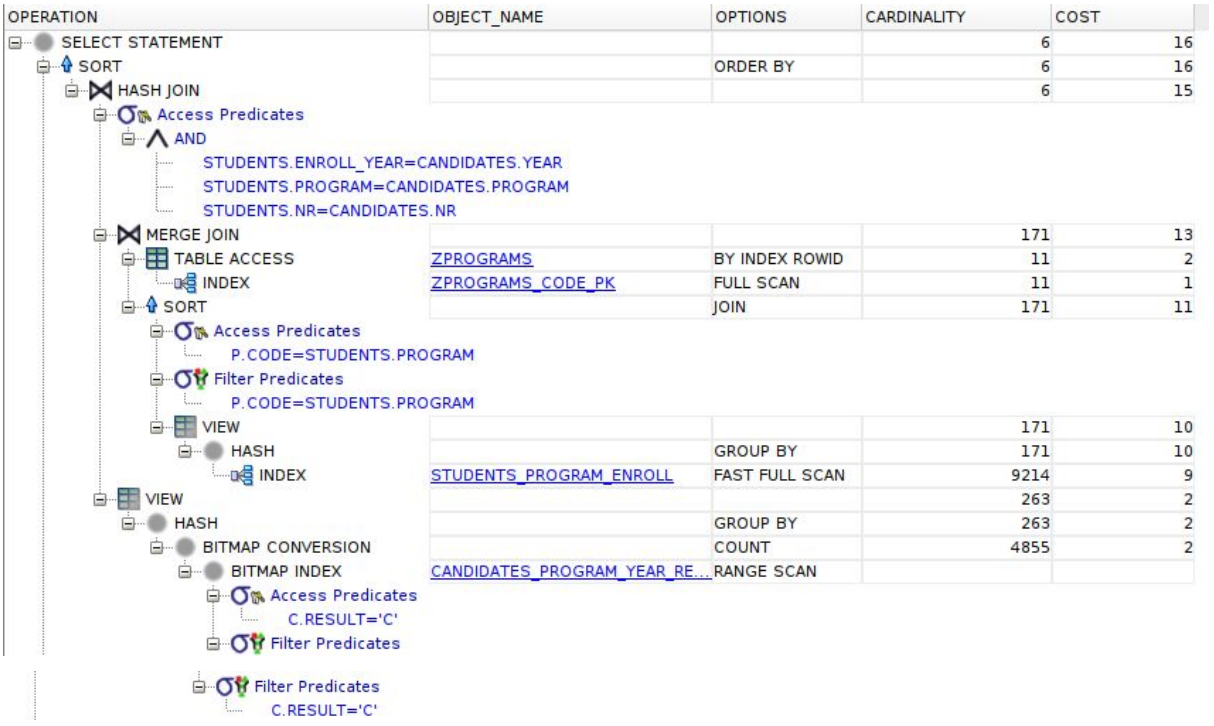
6.3.2.2. Dupla negação



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1936	49
SORT		GROUP BY	1936	49
HASH JOIN			12867	47
Access Predicates				
P.CODE=CANDIDATES.PROGRAM				
TABLE ACCESS	YPROGRAMS	FULL	11	3
HASH JOIN		RIGHT ANTI	12867	44
Access Predicates				
AND				
PROGRAM=PROGRAM				
YEAR=YEAR				
VIEW	SYS.VW_NSO_1		134	29
HASH JOIN		RIGHT ANTI	134	29
Access Predicates				
AND				
S.ENROLL_YEAR=C.YEAR				
S.PROGRAM=C.PROGRAM				
S.ID=C.ID				
TABLE ACCESS	YSTUDENTS	FULL	9214	14
TABLE ACCESS	YCANDIDATES	FULL	13400	15
Filter Predicates				
C.RESULT='C'				
TABLE ACCESS	YCANDIDATES	FULL	13400	15
Filter Predicates				
CANDIDATES.RESULT='C'				

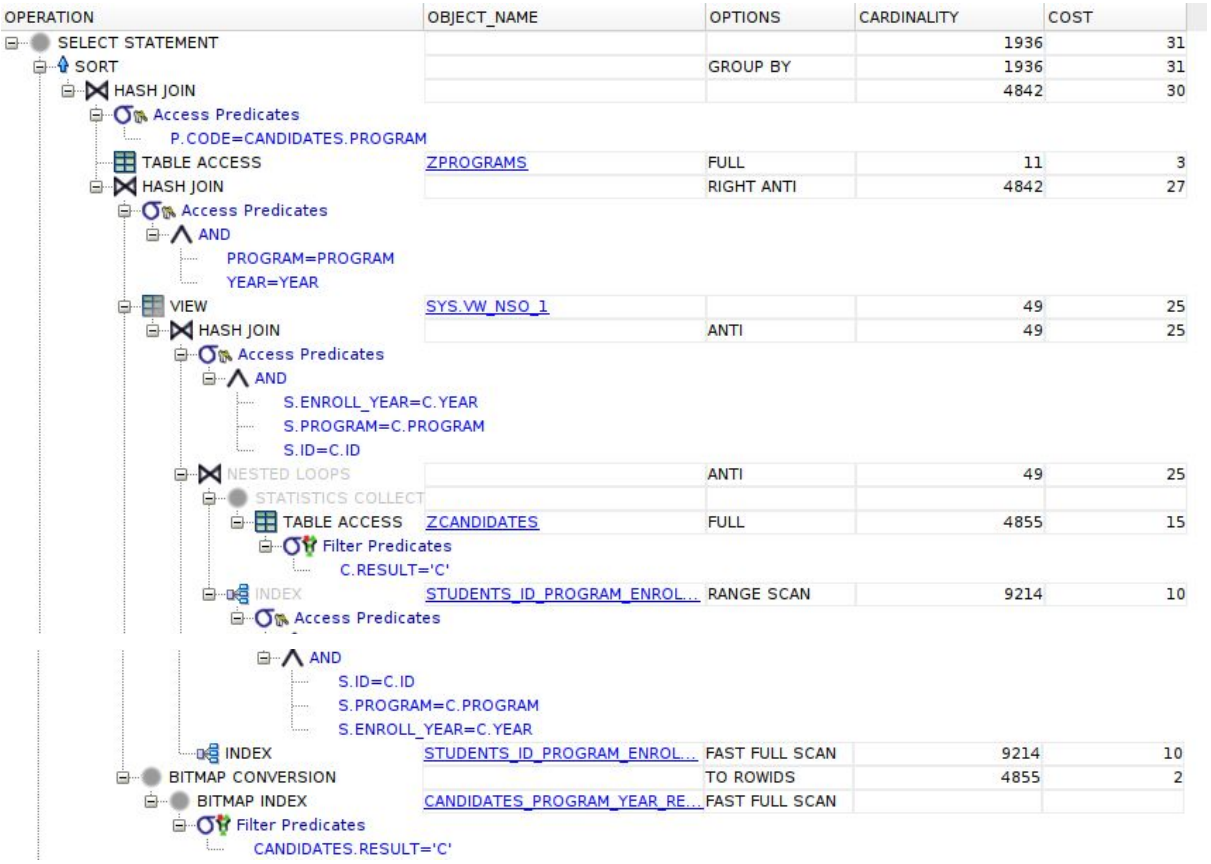
6.3.3. Ambiente Z

6.3.3.1. Contagem



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			6	16
SORT		ORDER BY	6	16
HASH JOIN			6	15
Access Predicates				
AND				
STUDENTS.ENROLL_YEAR=CANDIDATES.YEAR				
STUDENTS.PROGRAM=CANDIDATES.PROGRAM				
STUDENTS.NR=CANDIDATES.NR				
MERGE JOIN			171	13
TABLE ACCESS	ZPROGRAMS	BY INDEX ROWID	11	2
INDEX	ZPROGRAMS_CODE_PK	FULL SCAN	11	1
SORT		JOIN	171	11
Access Predicates				
P.CODE=STUDENTS.PROGRAM				
Filter Predicates				
P.CODE=STUDENTS.PROGRAM				
VIEW			171	10
HASH		GROUP BY	171	10
INDEX	STUDENTS_PROGRAM_ENROLL	FAST FULL SCAN	9214	9
VIEW			263	2
HASH		GROUP BY	263	2
BITMAP CONVERSION		COUNT	4855	2
BITMAP INDEX	CANDIDATES_PROGRAM_YEAR_RE...	RANGE SCAN		
Access Predicates				
C.RESULT='C'				
Filter Predicates				
Filter Predicates				
C.RESULT='C'				

6.3.3.2. Dupla negação



6.4. Tempos de execução

6.4.1. Contagem

Ambiente	X	Y	Z
Tempo (ms)	35.8	36.6	32.0

6.4.2. Dupla negação

Ambiente	X	Y	Z
Tempo (ms)	42.6	42.4	41.2

6.5. Conclusões

De modo a responder a esta pergunta, como especificado no enunciado, foram aplicadas duas estratégias de resolução: **contagem** e **dupla negação**.

Por um lado, a estratégia de **contagem** representa uma abordagem mais simples de resolução o que transparece no seu plano de execução, uma vez que essencialmente apenas compara os valores de dois contadores. A adição de restrições de integridade *standard* não revela um impacto significativo em termos de custo ou plano de execução neste método. No entanto, no ambiente Z é notória uma redução do custo de execução para 2, revelando-se também uma pequena redução do tempo de execução. Por conseguinte, verifica-se que adição de dois índices, um índice composto do tipo *B-tree* nas colunas *program* e *enroll_year* da tabela *students* (***students_program_enroll)*** usado pelo otimizador na *subquery* que conta o número de estudantes inscritos e outro índice composto do tipo *bitmap* nas colunas *result*, *program*, *year* da tabela *candidates* (***candidates_program_year_result)*** usado pelo otimizador na *subquery* que conta o número de candidatos, permite uma maior eficiência no ambiente Z.

Por outro lado, a estratégia de **dupla negação** revela-se mais complexa, quer ao nível da *query* implementada quer ao nível do plano de execução obtido. Do mesmo modo, a adição de restrições de integridade *standard* no ambiente Y não revela melhorias significativas relativamente ao ambiente X (em relação ao custo, plano e tempo de execução). No ambiente Z o custo de execução foi reduzido para 2 (e revela-se também uma redução do tempo de execução, possivelmente não muito significativa) face aos ambiente X e Y. Neste ambiente o otimizador recorre a dois dos índices adicionados, ***students_id_program_enroll_year*** (índice composto do tipo B-tree nas colunas id, program e enroll_year da tabela students) e ***candidates_program_year_result*** utilizado também na estratégia de contagem.

Nesta situação, conclui-se ser mais vantajosa a estratégia de **contagem** uma vez que constitui uma abordagem mais simples e intuitiva de resolução do problema e revelou, em todos os ambientes, um plano de execução mais simples e tempos de execução significativamente inferiores comparativamente à estratégia de **dupla negação**.

Conclusões Finais

Este trabalho permite concluir que a adição de índices pode ter um impacto muito significativo nos planos e custo de execução das interrogações feitas a uma base de dados. Deste modo, é muito importante uma escolha ponderada destes mesmos índices uma vez que também eles acarretam um custo e podem ou não impactar de forma positiva a eficiência e desempenho da base de dados.

Por outro lado, a adição de restrições de integridade *standard* (chaves primárias e estrangeiras), embora tenha um papel fundamental na implementação e manutenção da base de dados, não revela (em geral) um impacto significativo nos planos e custo de execução das interrogações.