

Faculdade de Engenharia da Universidade do Porto

Licenciatura em Engenharia Informática e Computação



Universidade do Porto

Faculdade de Engenharia

FEUP

Tecnologias de Bases de Dados

Optimização de Interrogações

Dezembro 2004

Nelson Rodrigues

nelson@fe.up.pt

Introdução

Este relatório serve para documentar a execução do Trabalho Prático nº3 da disciplina de Tecnologias de Bases de Dados – Optimização de Interrogações.

O objectivo é analisar tanto o plano de execução como o tempo que demoram executar certas interrogações sobre uma base de dados relativamente grande. Pretende-se ainda desenvolver várias estratégias de formulação de perguntas e verificar como as diferentes estratégias se reflectem ao nível do desempenho da interrogação.

Exercício 1

Após a primeira execução de cada questão sem a utilização de restrições de integridade ou índices foram criadas as restrições usando os seguintes comandos:

```
ALTER TABLE lics ADD PRIMARY KEY (codigo);

ALTER TABLE cands ADD PRIMARY KEY (bi, curso, ano_lectivo);
ALTER TABLE cands ADD FOREIGN KEY (curso) REFERENCES lics;

ALTER TABLE alus ADD PRIMARY KEY (numero, bi, a_lect_matricula);
ALTER TABLE alus ADD FOREIGN KEY (bi, curso, a_lect_matricula)
REFERENCES cands;
```

- a) Qual o número dos alunos que obtiveram uma média final superior à de candidatura?

Formulação SQL:

```
SELECT numero
FROM alus INNER JOIN cands ON (alus.bi = cands.bi AND
    alus.curso = cands.curso AND
    alus.a_lect_matricula = cands.ano_lectivo)
WHERE med_final > media;
```

Resposta:

NUMERO
980503066
980503004
980503027
960506011
960506008

Tempo de Execução:

- Sem restrições: 00:00:00.03
- Com restrições: 00:00:00.03

Plano de Execução:

▪ Sem restrições:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=27 Card=461 Bytes= 22589)
1	0	HASH JOIN (Cost=27 Card=461 Bytes=22589)
2	1	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=276420)
3	1	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=13 Card=14566 Bytes=276754)

▪ Com restrições:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=27 Card=461 Bytes= 18440)
1	0	HASH JOIN (Cost=27 Card=461 Bytes=18440)
2	1	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=230350)
3	1	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=13 Card=14566 Bytes=218490)

- b) Qual a lista do número dos alunos especiais, que terminaram o curso com sigla EIC em menos de cinco anos, e quantos anos demoraram.

Formulação SQL:

```
SELECT numero, (a_lect_conclusao - a_lect_matricula + 1) AS duracao
FROM alus INNER JOIN lics ON (alus.curso = lics.codigo)
WHERE sigla = 'EIC' AND (a_lect_conclusao - a_lect_matricula + 1 <
5);
```

Resposta:

NUMERO	DURACAO
970509050	3
980509039	3

Tempo de Execução:

- Sem restrições: 00:00:00.02
- Com restrições: 00:00:00.02

Plano de Execução:

▪ Sem restrições:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=15 Card=42 Bytes=1 218)
1	0	HASH JOIN (Cost=15 Card=42 Bytes=1218)
2	1	TABLE ACCESS (FULL) OF 'LICS' (TABLE) (Cost=2 Card=1 Bytes=8)
3	1	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=13 Card=461 Bytes=9681)

▪ Com restrições:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=15 Card=42 Bytes=1 218)
1	0	HASH JOIN (Cost=15 Card=42 Bytes=1218)
2	1	TABLE ACCESS (FULL) OF 'LICS' (TABLE) (Cost=2 Card=1 Bytes=8)
3	1	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=13 Card=461 Bytes=9681)

Comentário às duas questões anteriores:

Como foi possível observar quer pelos tempos de execução quer pelos planos de execução apresentados, a existência de índices decorrentes da existência de chaves primárias não alterou em nada nenhum destes factores. Este facto pode-se facilmente atribuir à própria natureza das interrogações que implicam análises completas das tabelas.

Exercício 2

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

Formulação SQL:

```
SELECT min(media), sigla, ano_lectivo
FROM alus INNER JOIN (cands INNER JOIN lics ON curso = codigo)
ON alus.bi = cands.bi AND alus.curso = cands.curso AND
alus.a_lect_matricula = cands.ano_lectivo
WHERE media IS NOT NULL
GROUP BY ano_lectivo, curso, sigla
ORDER BY ano_lectivo desc;
```

Resposta:

MIN(MEDIA)	SIGLA	ANO_LECTIVO
12,75	EC	2002
13,45	EEC	2002
13,78	EIC	2002
...		
12	EM	1993
62,7	EEC	1991
5,5	EEC	1989

Tempo de Execução:

- Sem restrições: 00:00:00.02
- Com restrições: 00:00:00.02

Plano de Execução:

- Sem restrições:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=31 Card=1876 Bytes=82544)
1	0	SORT (GROUP BY) (Cost=31 Card=1876 Bytes=82544)
2	1	HASH JOIN (Cost=29 Card=2031 Bytes=89364)
3	2	TABLE ACCESS (FULL) OF 'LICS' (TABLE) (Cost=2 Card=11 Bytes=88)
4	2	HASH JOIN (Cost=27 Card=2031 Bytes=73116)
5	4	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=14 Card=3211 Bytes=61009)
6	4	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=156638)

- Com restrições:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=30 Card=1876 Bytes=69412)
1	0	SORT (GROUP BY) (Cost=30 Card=1876 Bytes=69412)
2	1	HASH JOIN (Cost=29 Card=2031 Bytes=75147)
3	2	MERGE JOIN (Cost=16 Card=3211 Bytes=73853)
4	3	TABLE ACCESS (BY INDEX ROWID) OF 'LICS' (TABLE) (Cost=1 Card=11 Bytes=88)
5	4	INDEX (FULL SCAN) OF 'SYS_C0034517' (INDEX (UNIQUE)) (Cost=1 Card=11)
6	3	SORT (JOIN) (Cost=15 Card=3211 Bytes=48165)
7	6	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=14 Card=3211 Bytes=48165)
8	2	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=128996)

- b) Quais os cursos que têm alunos excluídos em todos os anos de 1995 a 2002? Indique o código e o nome do curso.

Formulação SQL:

```
SELECT sigla, nome
FROM lics
WHERE codigo NOT IN
  (SELECT codigo
   FROM lics, anos
   WHERE ano BETWEEN 1995 AND 2002
        AND (codigo, ano) NOT IN
          (SELECT curso, ano_lectivo
           FROM cands
           WHERE resultado = 'E'))
);
```

Resposta:

SIGLA	NOME
EC	Engenharia Civil
EEC	Engenharia Electrotécnica e de Computadores

Tempo de Execução:

- Sem restrições: 00:00:00.02
- Com restrições: 00:00:00.02

Plano de Execução:

- Sem restrições:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=22 Card=10 Bytes=490)
1	0	HASH JOIN (ANTI) (Cost=22 Card=10 Bytes=490)
2	1	TABLE ACCESS (FULL) OF 'LICS' (TABLE) (Cost=2 Card=11 Bytes=396)
3	1	VIEW OF 'VW_NSO_1' (VIEW) (Cost=19 Card=1 Bytes=13)
4	3	HASH JOIN (ANTI) (Cost=19 Card=1 Bytes=17)
5	4	MERGE JOIN (CARTESIAN) (Cost=5 Card=86 Bytes=602)
6	5	TABLE ACCESS (FULL) OF 'ANOS' (TABLE) (Cost=2 Card=8 Bytes=24)
7	5	BUFFER (SORT) (Cost=3 Card=11 Bytes=44)
8	7	TABLE ACCESS (FULL) OF 'LICS' (TABLE) (Cost=0 Card=11 Bytes=44)
9	4	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=14 Card=548 Bytes=5480)

- Com restrições:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=20 Card=10 Bytes=4 90)
1	0	MERGE JOIN (ANTI) (Cost=20 Card=10 Bytes=490)
2	1	TABLE ACCESS (BY INDEX ROWID) OF 'LICS' (TABLE) (Cost=1 Card=11 Bytes=396)
3	2	INDEX (FULL SCAN) OF 'SYS_C0034517' (INDEX (UNIQUE)) (Cost=1 Card=11)
4	1	SORT (UNIQUE) (Cost=19 Card=1 Bytes=13)
5	4	VIEW OF 'VW_NSO_1' (VIEW) (Cost=18 Card=1 Bytes=13)
6	5	HASH JOIN (ANTI) (Cost=18 Card=1 Bytes=14)
7	6	MERGE JOIN (CARTESIAN) (Cost=4 Card=86 Bytes=602)
8	7	TABLE ACCESS (FULL) OF 'ANOS' (TABLE) (Cost=2 Card=8 Bytes=24)
9	7	BUFFER (SORT) (Cost=2 Card=11 Bytes= 44)
10	9	INDEX (FULL SCAN) OF 'SYS_C0034517' (INDEX (UNIQUE)) (Cost=1 Card=11 Bytes=44)
11	6	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=14 Card=4855 Bytes=33985)

- c) Qual é a melhor média final do curso com maior número de candidaturas?
Indique a sigla do curso e a média.

Formulação SQL:

```
SELECT sigla, MAX(med_final) AS media
FROM alus INNER JOIN lics ON curso = codigo
WHERE curso = (
  SELECT curso
  FROM candS
  GROUP BY curso
  HAVING COUNT(curso) = (
    SELECT MAX(COUNT(curso)) AS contagem
    FROM candS
    GROUP BY curso
  )
)
GROUP BY curso, sigla;
```

Resposta:

SIGLA	MEDIA
EC	17,25

Tempo de Execução:

- Sem restrições: 00:00:00.02
- Com restrições: 00:00:00.02

Plano de Execução:

- Sem restrições:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=32 Card=86 Bytes=1 290)
1	0	SORT (GROUP BY) (Cost=32 Card=86 Bytes=1290)
2	1	HASH JOIN (Cost=15 Card=838 Bytes=12570)
3	2	TABLE ACCESS (FULL) OF 'LICS' (TABLE) (Cost=2 Card=11 Bytes=88)
4	2	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=838 Bytes=5866)

5	4	FILTER
6	5	SORT (GROUP BY) (Cost=16 Card=1 Bytes=4)
7	6	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=13 Card=14566 Bytes=58264)
8	5	SORT (AGGREGATE) (Cost=16 Card=1 Bytes=4)
9	8	SORT (GROUP BY) (Cost=16 Card=1 Bytes=4)
10	9	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=13 Card=14566 Bytes=58264)

▪ Com restrições:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=32 Card=86 Bytes=1118)
1	0	SORT (GROUP BY) (Cost=32 Card=86 Bytes=1118)
2	1	MERGE JOIN (Cost=15 Card=838 Bytes=10894)
3	2	TABLE ACCESS (BY INDEX ROWID) OF 'LICS' (TABLE) (Cost=1 Card=11 Bytes=88)
4	3	INDEX (FULL SCAN) OF 'SYS_C0034517' (INDEX (UNIQUE)) (Cost=1 Card=11)
5	2	SORT (JOIN) (Cost=14 Card=838 Bytes=4190)
6	5	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=838 Bytes=4190)
7	6	FILTER
8	7	SORT (GROUP BY) (Cost=16 Card=1 Bytes=3)
9	8	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=13 Card=14566 Bytes=43698)
10	7	SORT (AGGREGATE) (Cost=16 Card=1 Bytes=3)
11	10	SORT (GROUP BY) (Cost=16 Card=1 Bytes=3)
12	11	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=13 Card=14566 Bytes=43698)

Comentário às perguntas anteriores:

Como foi possível determinar pela observação dos planos de execução, alguns dos “TABLE ACCESS” são substituídos por acessos do tipo “INDEX”, no entanto esta modificação não se reflecte nos tempos de execução que se mantêm exactamente iguais.

Exercício 3

1. Estratégia da Contagem:

Formulação SQL:

```
SELECT ano_lectivo, sigla, nome
FROM lics,
( SELECT ano_lectivo, curso, COUNT(bi) c
  FROM cans
  WHERE resultado='C'
  GROUP BY ano_lectivo, curso ) x
WHERE (ano_lectivo, x.curso, c) IN
( SELECT a_lect_matricula, curso, COUNT(bi)
  FROM alus
  GROUP BY a_lect_matricula, curso )
AND curso = codigo;
```

Resposta:

ANO_LLECTIVO	SIGLA	NOME
1996	EMG	Engenharia de Minas e Geoambiente
1980	EM	Engenharia Mecânica
1977	EM	Engenharia Mecânica
1974	EM	Engenharia Mecânica

2001	EMI	Engenharia de Minas
1984	EMM	Engenharia Metalúrgica e de Materiais
1983	EMM	Engenharia Metalúrgica e de Materiais
1982	EMM	Engenharia Metalúrgica e de Materiais
1972	EC	Engenharia Civil
1970	EC	Engenharia Civil
1981	EMT	Engenharia Metalúrgica

Tempo de Execução: 00:00:00.05

Plano de Execução:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=34 Card=8 Bytes=912)
1	0	HASH JOIN (Cost=34 Card=8 Bytes=912)
2	1	HASH JOIN (Cost=31 Card=8 Bytes=624)
3	2	VIEW OF 'VW_NSO_1' (VIEW) (Cost=14 Card=242 Bytes=9438)
4	3	SORT (GROUP BY) (Cost=14 Card=242 Bytes=1936)
5	4	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=73712)
6	2	VIEW (Cost=16 Card=249 Bytes=9711)
7	6	SORT (GROUP BY) (Cost=16 Card=249 Bytes=2490)
8	7	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=14 Card=13400 Bytes=134000)
9	1	TABLE ACCESS (FULL) OF 'LICS' (TABLE) (Cost=2 Card=11 Bytes=396)

2. Estratégia da Dupla Negação:

Formulação SQL:

```

SELECT DISTINCT ano_lectivo, sigla, nome
FROM cans X, lics
WHERE X.curso = lics.codigo AND
NOT EXISTS(
  SELECT *
  FROM cans Y
  WHERE Y.curso = X.curso AND
        Y.ano_lectivo = X.ano_lectivo AND
        (Y.bi, Y.curso, Y.ano_lectivo) NOT IN (
          SELECT Z.bi, Z.curso, Z.ano_lectivo
          FROM cans Z, alus
          WHERE Z.ano_lectivo = alus.a_lect_matricula AND
                Z.bi = alus.bi AND
                Z.curso = alus.curso
        )
);

```

Resposta:

ANO_LECTIVO	SIGLA	NOME
1996	EMG	Engenharia de Minas e Geoambiente
1980	EM	Engenharia Mecânica
1977	EM	Engenharia Mecânica
1974	EM	Engenharia Mecânica

2001	EMI	Engenharia de Minas
1984	EMM	Engenharia Metalúrgica e de Materiais
1983	EMM	Engenharia Metalúrgica e de Materiais
1982	EMM	Engenharia Metalúrgica e de Materiais
1972	EC	Engenharia Civil
1970	EC	Engenharia Civil
1981	EMT	Engenharia Metalúrgica

Tempo de Execução: 00:00:00.07

Plano de Execução:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=61 Card=1876 Bytes =97552)
1	0	SORT (UNIQUE) (Cost=61 Card=1876 Bytes=97552)
2	1	HASH JOIN (Cost=59 Card=14525 Bytes=755300)
3	2	TABLE ACCESS (FULL) OF 'LICS' (TABLE) (Cost=2 Card=11 Bytes=396)
4	2	HASH JOIN (ANTI) (Cost=56 Card=14525 Bytes=232400)
5	4	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=13 Card=14566 Bytes =116528)
6	4	VIEW OF 'VW_SQ_2' (VIEW) (Cost=41 Card=14565 Bytes=116520)
7	6	HASH JOIN (ANTI) (Cost=41 Card=14565 Bytes=742815)
8	7	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=13 Card=14566 Bytes=247622)
9	7	VIEW OF 'VW_NSO_1' (VIEW) (Cost=27 Card=9214 Bytes=313276)
10	9	HASH JOIN (Cost=27 Card=9214 Bytes=313276)
11	10	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=156638)
12	10	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=13 Card=14566 Bytes=247622)

Comentário:

Nesta pergunta as duas estratégias revelaram-se bastante diferentes em termos de plano de execução, sendo que a estratégia da contagem tem menos operações, tendo-se revelado marginalmente mais rápido.

Exercício 4

Formulação SQL:

```
SELECT *
FROM cans
WHERE resultado <> 'C' OR resultado <> 'E';
```

a) Sem índice em Resultado:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=14 Card=12948 Bytes=207168)
1	0	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=14 Card=12948 Bytes=207168)

b) Com índice em Resultado:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=14 Card=12948 Bytes=207168)
1	0	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=14 Card=12948 Bytes=207168)

c) Com índice Bitmap em Resultado:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=14 Card=12948 Bytes=207168)
1	0	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=14 Card=12948 Bytes=207168)

Comentário:

Como se pode observar pelos planos de execução apresentados, a criação de índices no campo resultado não altera em nada execução da pergunta. Tal facto pode ser interpretado como consequência da estruturação da pergunta: qualquer coisa diferente de 'C' ou 'E'. Caso se tivesse feito uma pergunta do tipo: igual a 'S' os índices já teriam tido efeito nos planos de execução da pergunta.

Exercício 5**Formulação SQL:**

a) Sub pergunta constante:

```
SELECT COUNT(*)
FROM cands
WHERE resultado='C'
AND (bi, ano_lectivo, curso) NOT IN (
    SELECT bi, a_lect_matricula, curso
    FROM alus
);
```

b) Sub pergunta variável:

```
SELECT count(*)
FROM cands
WHERE resultado='C'
AND (bi, ano_lectivo, curso) NOT IN (
    SELECT bi, a_lect_matricula, curso
    FROM alus
    WHERE a_lect_matricula=ano_lectivo
    AND alus.curso=cands.curso
);
```

Planos de Execução:

a) Sub pergunta constante:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=31491 Card=1 Bytes=15)
1	0	SORT (AGGREGATE)
2	1	FILTER
3	2	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=14 Card=4855 Bytes=72825)
4	2	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=13 Card=1 Bytes=14)

b) Sub pergunta variável:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=27 Card=1 Bytes=29)
1	0	SORT (AGGREGATE)
2	1	HASH JOIN (ANTI) (Cost=27 Card=4855 Bytes=140795)
3	2	TABLE ACCESS (FULL) OF 'CANDS' (TABLE) (Cost=14 Card=4855 Bytes=72825)
4	2	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=128996)

Comentário:

Nesta pergunta podemos observar uma diferença subtil entre os dois planos de execução, mas que traz grandes vantagens em termos de tempos de execução, enquanto que a sub pergunta constante usa uma operação “FILTER” no passo 2, a sub pergunta variável usa uma operação de “ANTI JOIN” muito mais eficiente.

Exercício 6**Resposta 1:**

Plano de execução:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=29 Card=1 Bytes=65)
1	0	HASH JOIN (Cost=29 Card=1 Bytes=65)
2	1	VIEW OF 'VW_SQ_1' (VIEW) (Cost=14 Card=12 Bytes=312)
3	2	SORT (GROUP BY) (Cost=14 Card=12 Bytes=180)
4	3	VIEW (Cost=14 Card=94 Bytes=1410)
5	4	SORT (GROUP BY) (Cost=14 Card=94 Bytes=658)
6	5	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=64498)
7	1	VIEW (Cost=14 Card=94 Bytes=3666)
8	7	SORT (GROUP BY) (Cost=14 Card=94 Bytes=658)
9	8	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=64498)

Resposta 2:

Plano de execução:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=29 Card=1 Bytes=65)
1	0	HASH JOIN (Cost=29 Card=1 Bytes=65)
2	1	VIEW OF 'VW_SQ_1' (VIEW) (Cost=14 Card=12 Bytes=312)
3	2	SORT (GROUP BY) (Cost=14 Card=12 Bytes=180)
4	3	VIEW OF 'MEDIA' (VIEW) (Cost=14 Card=94 Bytes=1410)
5	4	SORT (GROUP BY) (Cost=14 Card=94 Bytes=658)
6	5	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=64498)
7	1	VIEW OF 'MEDIA' (VIEW) (Cost=14 Card=94 Bytes=3666)
8	7	SORT (GROUP BY) (Cost=14 Card=94 Bytes=658)
9	8	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=64498)

Resposta 3:

Plano de execução:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=29 Card=1 Bytes=65)
1	0	HASH JOIN (Cost=29 Card=1 Bytes=65)
2	1	VIEW (Cost=14 Card=12 Bytes=312)
3	2	SORT (GROUP BY) (Cost=14 Card=12 Bytes=180)
4	3	VIEW OF 'MEDIA' (VIEW) (Cost=14 Card=94 Bytes=1410)
5	4	SORT (GROUP BY) (Cost=14 Card=94 Bytes=658)
6	5	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=64498)
7	1	VIEW OF 'MEDIA' (VIEW) (Cost=14 Card=94 Bytes=3666)
8	7	SORT (GROUP BY) (Cost=14 Card=94 Bytes=658)
9	8	TABLE ACCESS (FULL) OF 'ALUS' (TABLE) (Cost=12 Card=9214 Bytes=64498)

Resposta 4:

Plano de execução:

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=6 Card=2 Bytes=130)
1	0	HASH JOIN (SEMI) (Cost=6 Card=2 Bytes=130)
2	1	TABLE ACCESS (FULL) OF 'ZMEDIA' (TABLE) (Cost=2 Card=76 Bytes=2964)
3	1	VIEW OF 'VW_NSO_1' (VIEW) (Cost= 3 Card=76 Bytes=1976)
4	3	SORT (GROUP BY) (Cost=3 Card=76 Bytes=1976)
5	4	TABLE ACCESS (FULL) OF 'ZMEDIA' (TABLE) (Cost=2 Card=76 Bytes=19 76)

Comentário:

Analisando as várias estratégias para conseguir a resposta pretendida podemos verificar que a melhor alternativa é a resposta 4, porque “despeja” o resultado de uma pergunta intermédia numa tabela que depois interroga facilmente. As respostas 2 e 3 adoptam uma estratégia semelhante mas colocam o resultado da pergunta intermédia numa vista que depois tem de ser calculada na altura da pergunta principal, a única vantagem desta estratégia em relação à primeira pergunta é facilitar a leitura da pergunta. A resposta 1 coloca a pergunta intermédia directamente na pergunta principal, o optimizador chama a si a responsabilidade de criar uma vista temporária para proceder à execução desta sub pergunta.