e-Fólio A

U.C. 21103

Sistemas de Gestão de Bases de Dados

2017-2018

Resolução e Critérios de Correção

INSTRUÇÕES

- O e-fólio é constituído por 6 alíneas com cotação de 0,5 valores cada. A cotação global é de 3 valores.
- O e-fólio deve ser entregue num único ficheiro PDF, não zipado, com fundo branco, com perguntas numeradas e sem necessidade de rodar o texto para o ler. Penalização de 1 a 3 valores.
- Não são aceites e-fólios manuscritos, i.e. tem penalização de 100%.
- O nome do ficheiro deve seguir a normal "eFolioA" + <nº estudante> + <nome estudante com o máximo de 3 palavras>
- Durante a realização do e-fólio, os estudantes devem concentrar-se na resolução do seu trabalho individual, não sendo permitida a colocação de perguntas ao professor ou entre colegas.
- A interpretação das perguntas também faz parte da sua resolução, se encontrar alguma ambiguidade deve indicar claramente como foi resolvida.
- A legibilidade, a objetividade e a clareza nas respostas serão valorizadas, pelo que, a falta destas qualidades serão penalizadas.

Vetor Cotações
1 2 3, 4 5 6 pergunta
5 5 5, 5 5 5 décimas

Critérios de correção gerais: todas as respostas devem ser justificadas, incluir imagens e exemplos com vista a clarificar os argumentos expostos.

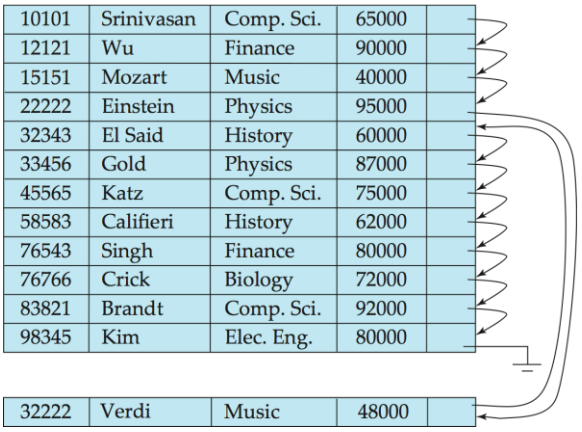
1) Relativo ao Cap.10 – Storage and File Structure
Existem várias formas de organização de registos em ficheiros. Quais as formas que conhece? Exemplifique.

Resposta:
Formas de organização de registos em ficheiros:

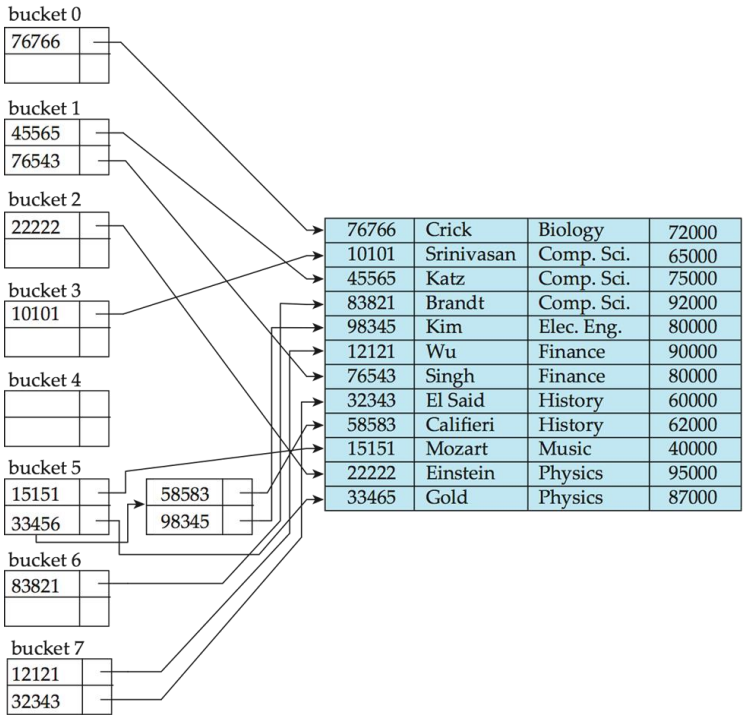
- *Heap* (ou pilha),
- sequenciais,
- *hashing* e
- agrupados (*clustered*).

Ficheiros 'heap': os registos podem ser armazenados em qualquer parte do ficheiro, desde que exista espaço para tal. Os registos não são ordenados.

Ficheiros sequenciais: os registos são armazenados em ordem sequencial da chave de busca.



Ficheiros ‘hash’: os registos são armazenados em “buckets” indexados por uma função ‘hash’.



Ficheiros agrupados: registos de várias tabelas diferentes podem ser armazenados no mesmo ficheiro; pode ser utilizado, quando uma junção é efetuada com frequência.

department

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Physics	Watson	70000

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

multitable clustering of *department* and *instructor*

Comp. Sci.	Taylor	100000
45564	Katz	75000
10101	Srinivasan	65000
83821	Brandt	92000
Physics	Watson	70000
33456	Gold	87000

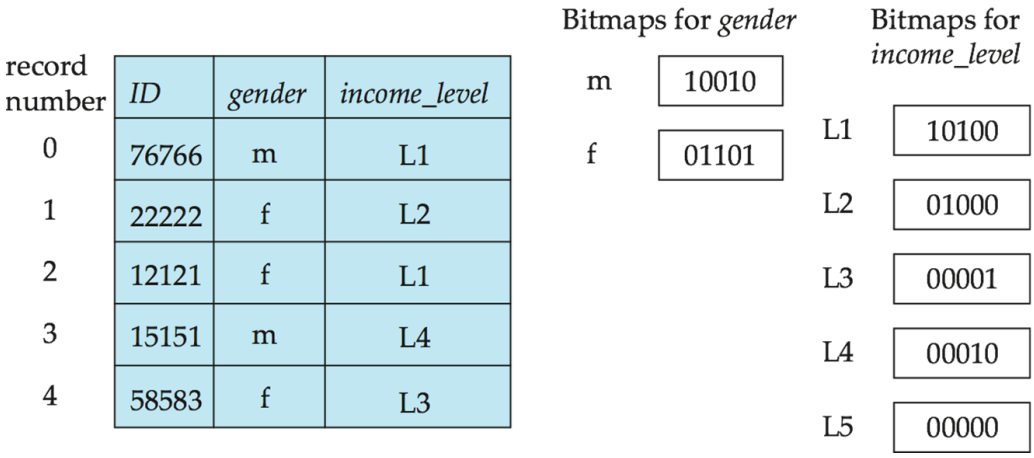
Os SGBD DB2 e PostgreSQL permitem apenas ficheiros ‘heap’. Oracle usa ficheiros ‘heap’ por omissão.

Nota: figuras retiradas de http://www.db-book.com/

Critério de correção:
- 5 décimas para as 4 formas de organização
- erros, omissões, redundâncias ou indentação desadequada: -20% a -100%

2) Relativo ao Cap. 11 - Indexing and Hashing
Discuta as vantagens da utilização de índices mapa de bits.

Resposta:
Na sua forma mais simples, num índice de mapa de bits existe um mapa de bits para cada valor X do atributo.
- um mapa de bits tem tantos bits quanto registros;
- num mapa de bits para um valor X, o bit tem valor 1 se o registo tiver X e 0 no caso contrário.



Índices de bitmap são úteis para consultas em vários atributos e não são particularmente úteis para consultas de atributo único.

Os mapas de *bits* são eficientes para representar chaves com poucos valores únicos, e para representar valores nulos, o que não pode ser feito com índices arborescentes.

As consultas booleanas são respondidas usando operações de bitmap: Intersecção (e), União (ou) e Negação (não).

Exemplo: Homens com nível de renda L1: 10010 E 10100 = 10000

Fornecem uma forma eficiente de realizar consultas booleanas e são compactos relativamente aos índices arborescentes.

Nota: figuras retiradas de http://www.db-book.com/

Critério de correção:
- 2 décimas, para definição e exemplo de mapa de índices
- 3 décimas, 2 vantagens relativamente aos índices arborescentes
- erros, omissões, redundâncias ou indentação desadequada: -20% a -100%

3) Relativo ao Cap. 12 - Query Processing
Escreva a seguinte consulta em álgebra relacional e desenhe os respectivos planos de execução. Tente desenhar mais do que um plano para a consulta.

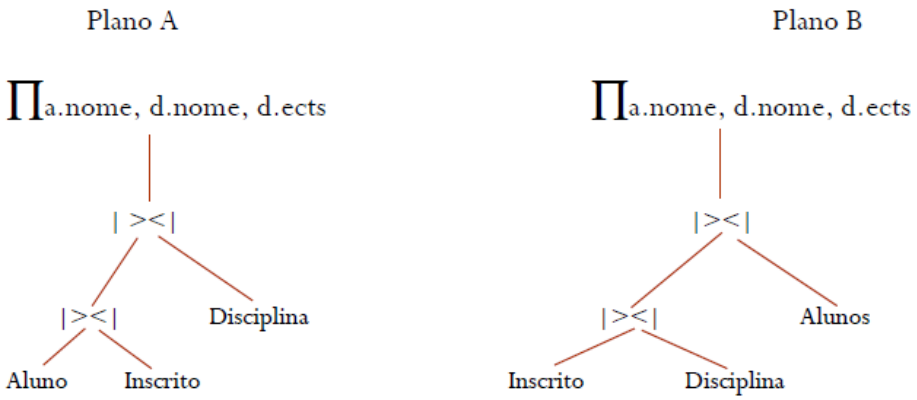
```
select a.nome, d.nome, d.ects
from aluno a, inscrito i, disciplina d
where  a.id=i.aluno_id
and      i.disc_id=d.id.
```

Resposta:

Álgebra relacional do Plano A:

πaluno.nome, ((aluno |><| inscrito) |><| disciplina)
disciplina.nome, aluno_id disc_id
disciplina.ects

Planos alternativos A e B:



Critério de correção:
- 3 décimas, álgebra relacional
- 2 décimas, planos alternativos
- erros, omissões, redundâncias ou indentação desadequada: -20% a -100%

4) Relativo ao Cap. 13 - Query Optimization (F Gouveia 7.5)
Que informação precisa para otimizar a seguinte consulta? Estime valores aleatórios para a informação que pretende e desenhe um plano de execução.

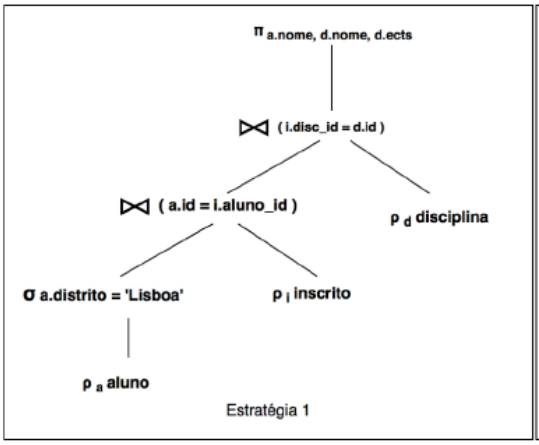
```
select a.nome, d.nome, d.ects
from aluno a, inscrito i, disciplina d
where a.id=i.aluno_id
and     i.disc_id=d.id
and    a.distrito=’lisboa’
```

Resposta:

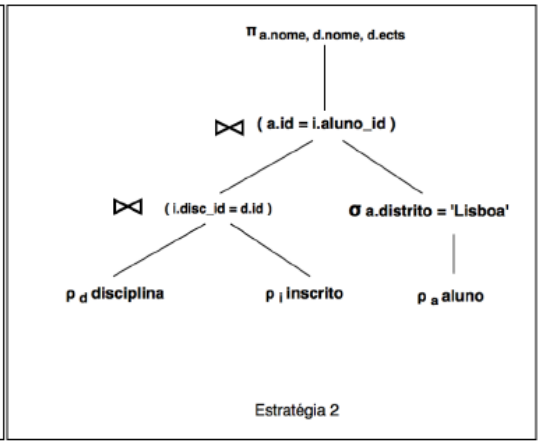
Informação necessária para otimizar a consulta:

- número de tuplos/linhas das tabelas Nr;
- número de valores distintos das colunas envolvidas nas junções V(A,r);
- a existência de histogramas com a distribuição dos dados;
- a existência de um índice em distrito e valores da seleção distrito=’lisboa’;

Sendo possíveis os seguintes 2 planos:



Estratégia 1



Estratégia 2

Considerando os seguinte valores para Nr e V(A,r):

Tabela r	**Nr tuplos**
disciplina	150
inscrito	10.000
aluno	1.000
aluno_Lisboa	400

A dimensão estimada na junção: min (Nr*Ns/V(A,r), Nr*Ns/V(A,s))

		N	V(id-aluno)
s	aluno_Lisboa	400	400
r	inscrito	10.000	1.000
j1	Nr*Ns/V(A,s)	10.000	
j2	Nr*Ns/V(A,r)	4.000	
	Custo = min(j1,j2)	4.000	

		N	V(id-disc)
s	disciplina	150	150
r	inscrito	10.000	150
j1	Nr*Ns/V(A,s)	10.000	
j2	Nr*Ns/V(A,r)	10.000	
	Custo = min(j1,j2)	10.000	

		N	V(id-disc)
s	aluno_Lisboa |><| inscrito	2.000	120
r	disciplina	150	150
j1	Nr*Ns/V(A,s)	2.500	
j2	Nr*Ns/V(A,r)	2.000	
	dimensão = min(j1,j2)	2.000	
	Dimensão Plano 1	6.000	

		N	V(id-aluno)
s	disciplina |><| inscrito	10.000	1.000
r	aluno_Lisboa	400	400
j1	Nr*Ns/V(A,s)	4.000	
j2	Nr*Ns/V(A,r)	10.000	
	dimensão = min(j1,j2)	4.000	
	Dimensão Plano 2	14.000	

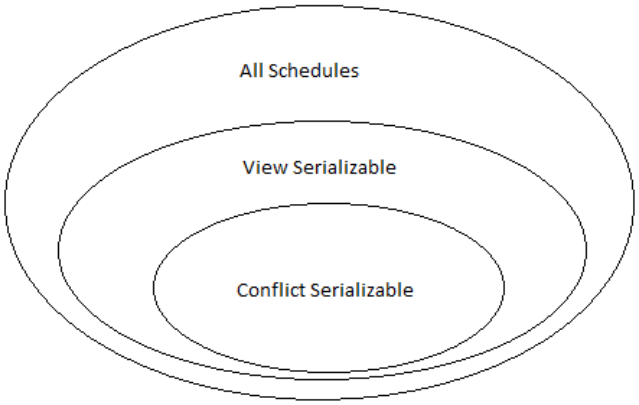
O Plano 1 apresenta uma menor dimensão estimada nas junções.

Critério de correção:
- 3 décimas, informação necessária
- 2 décimas, escolha do plano
- erros, omissões, redundâncias ou indentação desadequada: -20% a -100%

5) Relativo ao Cap. 14 – Transactions
Visto que todos os escalonamentos seriável a conflitos são também seriáveis a vistas, porque é que se dá mais enfase aos escalonamentos seriável a conflitos?

Resposta:

Na realidade os escalonamentos seriável a conflitos estão contidos nos escalonamentos seriáveis a vistas.



A maior parte dos protocolos de controlo de concorrência usados na prática são baseados nos escalonamentos seriável a conflitos.

A forma mais geral dos escalonamentos seriáveis a vistas são computacionalmente muito dispendiosos de testar, com valor de N!, sendo N o número de transações, e só ocorrem em condições particulares do controlo de concorrência.

Critério de correção:
- 2 décimas, explicar os escalonamentos seriáveis
- 3 décimas, explicação para que o ‘a conflitos’ seja mais utilizado
- erros, omissões, redundâncias ou indentação desadequada: -20% a -100%

6) Relativo ao Cap. 14 – Transactions
O escalonamento R1(x), R2(x), R3(y), W2(x), R1(y), W1(y), R3(x), W3(z) é seriável a vistas ou a conflitos? Porquê?

Resposta:

6.a) Definição de serialização a conflitos e a vistas.

Serialização a conflitos: um escalonamento é seriável por conflitos se o seu grafo de precedência não contiver ciclos.

Na construção do grafo de precedências as arestas são montadas a partir das observações das transações que participa, da escala sendo duas transação T_i e T_j haverá uma aresta: $T_i \rightarrow T_j$ se forem observadas as seguintes condições:

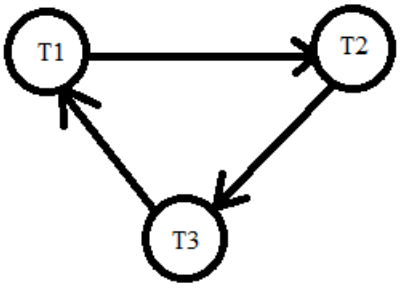
- T_i executa *write*(q) antes de T_j executar *read*(q)
- T_i executa *read*(q) antes de T_j executar *write*(q)
- T_i executa *write*(q) antes de T_j executar *write*(q)

Serialização por vistas: Diz-se que "Ti lê x de Tj", se Wj(x) for a última operação de escrita antes de Ri(x). Dois escalamentos são equivalentes a vistas, se a relação "lê de" em ambas for a mesma [Feliz Gouveia 2014].
Dois escalonamentos S1 e S2 são equivalentes se:
(1) Todos os elementos A têm o mesmo valor inicial
Se S1: $r_i(A)$ lê o valor inicial, então S2: $r_i(A)$ também lê o mesmo valor inicial
(2) Um elemento A tem a mesma vista nos dois escalonamentos
Se S1: $w_j(A) \Rightarrow r_i(A)$ então em S2: $w_j(A) \Rightarrow r_i(A)$; têm a mesma vista
(3) Todos os elementos A têm a mesma vista final
Se S1: T_i termina $w_i(A)$, então S2: T_i também termina com $w_i(A)$; os valores finais são os mesmos.

6.b) Escalonamento seriável a conflitos

Divisão por recurso/item: Item x: R1, R2, W2, R3 Item y: R3, R1, W1 Item z: W3	As precedências são as seguintes: R1 -> W2, W2 -> R3 R3 -> W1

Graficamente obtemos um grafo cíclico, pelo que não é seriável a conflitos.



6.c) Para analisar se o escalonamento é seriável a vistas, devemos criar um ou mais escalonamentos em série, e de seguida verificar se cumpre a regra "Ti lê x de Tj".

Com e transações temos 3!=6 escalonamentos em série. Contudo, vamos testar os primeiros dois: T1,T2,T3 e T3,T2,T1.

O escalonamento concorrente é o dado no problema. Criámos dois escalonamentos em série:

- Série-1 (S1): verificamos que R1(y) e R3(y) no escalonamento concorrente são iniciados com o mesmo valor, o que não acontece em S1 visto que existe um W1(y). Não é serializável a vistas, porque T1 e T3 não conseguem ter uma mesma relação "lê de".

- Série-2 (S2): no escalonamento S2 verificamos que R2(x) e R1(x) são iniciados com o mesmo valor, o que não acontece com S2 visto que existe um W2(x). Não é serializável a vistas, porque T2 e T1 não conseguem ter uma mesma relação "lê de".

T1	**T2**	**T3**
R(x)		
	R(x)	
		R(y)
	W(x)	
R(y)		
W(y)		
		R(x)
		W(z)
	concorrente	

T1	**T2**	**T3**
R(x)		
R(y)		
W(y)		
	R(x)	
	W(x)	
		R(y)
		R(x)
		W(z)
	série 1	

T1	**T2**	**T3**
		R(y)
		R(x)
		W(z)
	R(x)	
	W(x)	
R(x)		
R(y)		
W(y)		
	série 2	

Conclusão: o escalonamento não é serializável por conflito e não é serializável por vistas.

Critério de correção:
- 1 décimas, explicar os escalonamentos seriáveis
- 2 décimas, detalhes 'a conflitos'
- 2 décimas, detalhes 'a vistas'
- erros, omissões, redundâncias ou indentação desadequada: -20% a -100%