Sistemas de Bases de Dados

Exame de recurso (com consulta limitada: 2 folhas identificadas)
Duração: 3 horas

N.º:		**Nome:**	

Grupo 1 (5 valores)

1 a) Para que servem os meios físicos de armazenamento terciário (ou *offline*)?

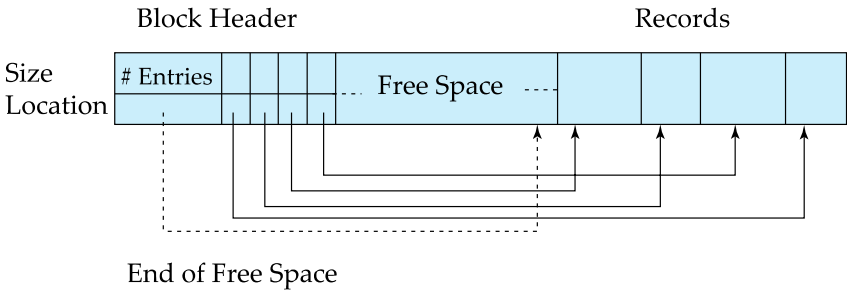
Essencialmente para *backup* e para guardar grandes volumes de dados.

1 b) Quais as vantagens dum sistema RAID?

Aumento de fiabilidade, via redundância.

Melhoria de desempenho, graças a paralelismo.

1 c) Considere a estrutura de *slotted page* (figura abaixo).



I. Para que serve esta estrutura?

Para guardar registos de tamanho variável.

II. Que atualizações concretas são feitas pelo SGBD nesta estrutura aquando da remoção dum registo?

Na remoção dum registo, este é marcado como removido na respetiva entrada do cabeçalho, e os outros registos serão encostados à direita, atualizando-se também as suas localizações nas respetivas entradas, tudo dentro do mesmo bloco.

1 d) Mostre como seria a organização de dados das seguintes tabelas em *clustering* multitabela:

turmas(*turma, sala*):

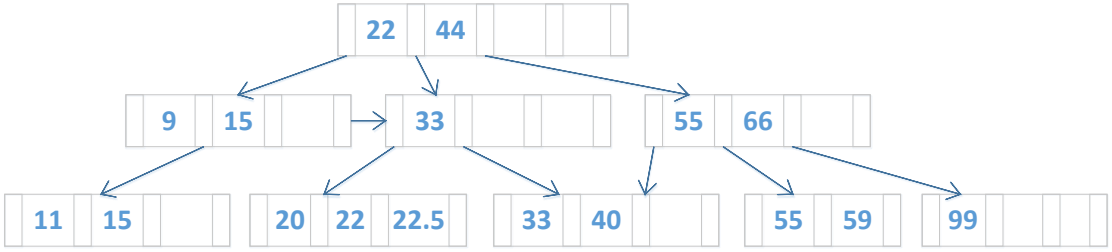
turma	1	2
sala	azul	verde

alunos(*nome, turma, nota*):

nome	ana	ze	ivo	eva	rui
turma	1	1	2	1	2
nota	MB	B	S	MB	B

1	azul
ana	MB
ze	B
eva	MB
2	verde
ivo	S
rui	B

1 e) Indique na figura e em texto quais os erros do seguinte índice *B+* sem duplicados (onde se excluem os apontadores para os registos), construído com as regras habituais (com *n*=4):



1- Raiz com 5 apontadores
2- Falta apontador (e respetivo nó) à esquerda de '9'
3- Falta apontador (e respetivo nó) à direita de '15', no nó interior
4- O valor '15' não pode aparecer neste nó (teria de ser >= 9 e < 15)
5- Nó interior subpreenchido
6- O valor '20' não pode aparecer neste nó (teria de ser >= 22 e < 33)
7- Apontador devia ser para outro nó (valores de 44 a 55)
8- Nó folha curto (3 apontadores)
9- Nó folha subpreenchido
10- Apontador indevido entre nós intermédios
11- Faltam os apontadores ligando as folhas

N.º:	

1 f) Mostre o índice fruto da instrução `create bitmap index i on aluno(nota)` aplicado à seguinte instância da tabela **aluno**, fisicamente ordenada por Id:

Id	1	2	3	4	5	6	7	8
sexo	M	M	F	F	M	F	M	F
nota	MB	B	B	B	S	MB	MB	B

```
<MB>:   1000 0110
<B> :   0111 0001
<S> :   0000 1000
```

Grupo 2 (5 valores)

2 a) Considere que queremos ordenar uma relação com os seguintes (13) registos (com apenas uma coluna: um inteiro) em disco: (13, 4, 10, 41, 10, 16, 75, 99, 9, 8, 13, 8, 22). Seja o *blocking factor f*=2 e a capacidade da memória *M*=3 blocos. Aplicando o algoritmo de *sort-merge* (assuma que *buffer blocks* b_b=1), **mostre o conteúdo do primeiro *run*** (onde se percebam os respetivos blocos). **Diga ainda com quantos outros *runs* se juntará este no primeiro *merge pass*** (a sua resposta deve deixar claro quantos *runs* se juntam, i.e. se inclui o próprio ou não).

4	10	10	13	16	41

Este *run* (com *M*=3 blocos de *f*=2 registos) juntar-se-á outro no primeiro passo. Isto é, serão juntados *M*-1=2 *runs*.

2 b) Para a junção de 2 tabelas, quais as condições necessárias para que se possa aplicar o algoritmo *indexed nested-loop join* ?

1- Tratar-se duma junção natural ou *equi-join* (i.e. com condições de igualdade)
2- Haver (ou ser feito) um índice sobre os atributos de junção na relação interior

2 c) Considere a tabela *temp*(*id*, *city*, *ano*, *mês*, *dia*, *hora*, *val*) de valores de temperatura, criada em SQL da seguinte forma:

```
create table temp(
      id INTEGER NOT NULL primary key,
      City INTEGER NOT NULL references cidades(id),
      Ano INTEGER NOT NULL,
      Mes INTEGER NOT NULL check(mes between 1 and 12),
      Dia INTEGER NOT NULL check(dia between 1 and 31),
      Hora INTEGER NOT NULL check(hora between 0 and 23),
      Val INTEGER NOT NULL,
      unique(cidade,ano,mes,dia,hora));
```

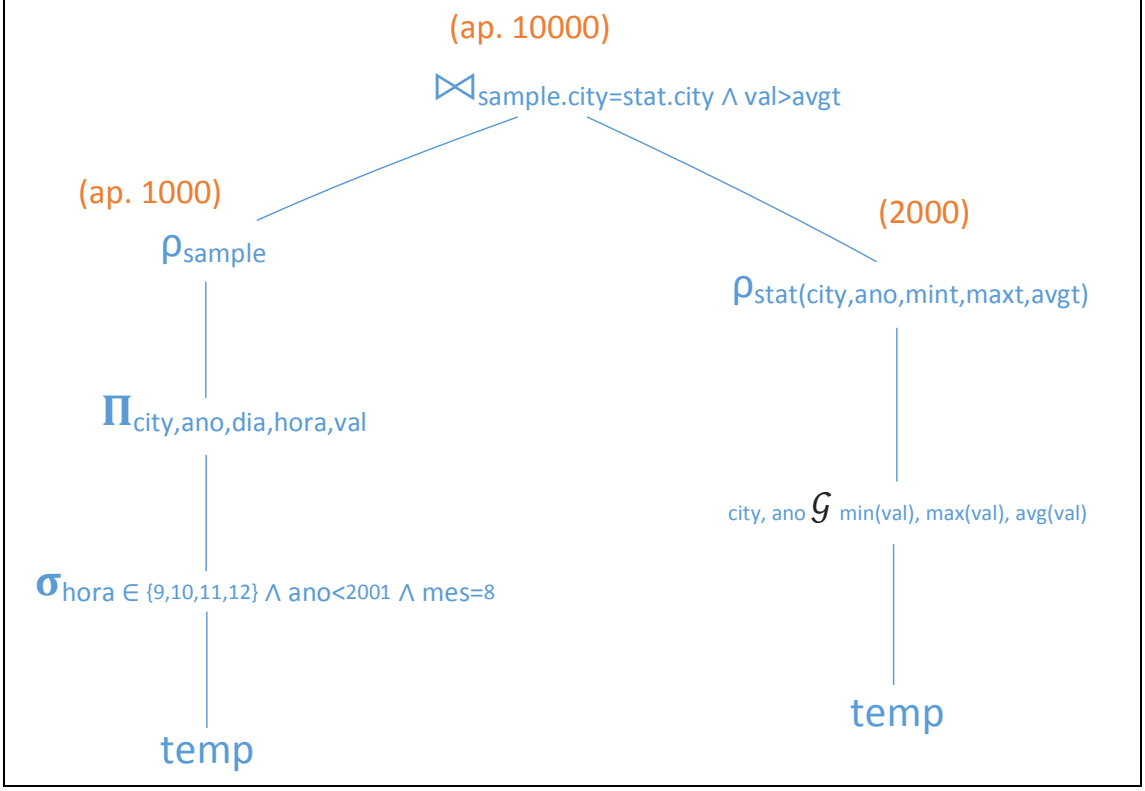
Foram registados 750 mil (7.5e5) valores de temperatura em 100 cidades europeias entre os anos 1999 e 2018, inclusive (há várias medições em falta, no espaço e no tempo, por diferentes motivos). Considere que neste sistema um inteiro ocupa 8 bytes, um real ocupa 16 bytes, e que são usados blocos de 10000 bytes (i.e. 10^4 bytes).
Imagine a seguinte consulta SQL:

```
with stat as
 (select city, ano, min(val) as mint, max(val) as maxt, avg(val) as avgt
  from temp group by city, ano),
 sample as (select city, ano, dia, hora, val from temp
            where hora in (9,10,11,12) and ano<2001 and mes=8)
(select * from sample inner join stat
   on (sample.city=stat.city and val > avgt);
```

Desenhe a árvore da expressão em álgebra relacional correspondente à conversão natural desta consulta, e indique para cada nó o respetivo valor estimado de tuplos.

n(stat) = 100*20 = 2000
n(sample) = 750000/(24/4)/(20/2)/12 = 750000/6/10/12 = 75000/72 > 1000

n(join) = n(sample)*20/2 = 10000



N.º:	

2 d) Considere uma seleção $\sigma_{A \geq v}(r)$, havendo um índice primário sobre o atributo *A* da relação r.

I. Como deverá ser a melhor maneira de obter todos os tuplos resultantes da seleção?
Usar o índice para encontrar o primeiro tuplo com $A \geq v$, e percorrer sequencialmente o ficheiro a partir daí.

II. Em que seria diferente o procedimento se a condição fosse $A \leq v$?
Não vale a pena usar o índice; basta percorrer sequencialmente o ficheiro da relação e ir devolvendo os tuplos até encontrar o primeiro com $A > v$.

Grupo 3 (6 valores)

3 a) Qual o mecanismo de base dum SGBD para poder garantir as seguintes propriedades duma transação? I.e. que operações ou estruturas de dados as possibilitam?

Atomicidade	*Rollback*; capacidade de abortar transações.
Durabilidade	Registos de *log*.

3 b) No protocolo de *locking* de 2 fases com conversões de *locks*, em que é que se caracteriza a 2.ª fase? I.e. o que é que aí se pode fazer?

Podem-se libertar *locks* partilhados (*shared* – S) e exclusivos (X), e converter *locks* X em S (*downgrade*).

3 c) Para cada um dos 4 escalonamentos abaixo (onde *sel*, *upd* e *ins* representam, respetivamente, *select*, *update*, *insert*, sobre tabelas com colunas *a* e *b*), indique (com 'X') se é ou não serializável de conflito. Em caso afirmativo, indique também uma serialização que lhe seja equivalente (de conflito).

Nota: Nesta alínea cada opção certa vale 0,3 valores, e cada opção errada vale –0,3 valores (negativo, portanto). Uma resposta afirmativa correta mas com serialização incorreta será cotada a 50% (i.e. 0,15 valores positivos).

	Não serializável	**Serializável**	**Serialização**
S1:	X		
S2:		X	T2, T1, T3
S3:		X	T1,T2
S4:	X		

S1

T1	T2
	upd s set a=10
upd t set a=9	
	sel * from t
	sel * from s
sel * from t	
sel * from s	

S2

T1	T2	T3
sel * from t		
sel * from s		
	sel * from t	
	upd v set a=0	
		upd v set b=0
upd t set a=0		
		sel * from v
upd s set a=0		
		upd s set a=0

S3

T1	T2
sel * from s;	
	sel * from t where a=1;
	ins into s(a,b) values (1,1);
upd t set b=1 where a=0;	
sel * from t;	
	sel count(*) from s;

S4

T1	T2	T3	T4
sel * from t			
			sel * from s
	upd t set a=0		
		sel * from s	
		sel * from t	
			upd s set a=0
upd s set b=0			
		upd t set a=1	

N.º:	

3 d) Seja a tabela ***t****(a, b, c)* já com os tuplos (1,1,1) e (2,2,2), previamente criada com:

```
create table t(a int primary key, b int, c int);
```

Considere as seguintes transações executando em *Snapshot Isolation* :

T1	T2
	Insert into t values (3,3,3);
	Update t set b=b-1, c=0 where a+c <> 4;
Select * from t;	
Update t set b=3, c=c+1 where b=2;	
	Commit;
Commit;	

I. O que verá o utilizador como resultado da instrução '*select * from t*' dentro de T1?

{(1,1,1), (2,2,2)}

II. Pode o escalonamento concluir com sucesso? Se não, explique porquê; se sim, mostre então o que seria o conteúdo final da tabela *t*.

Sim: {(1,0,0), (2,3,3), (3,2,0)}

3 e) O fenómeno *Unrepeatable Read* será possível no modo *Read Commited* do SQL? Se não, explique porquê; se sim, apresente um escalonamento concreto com operações sobre a tabela ***t*** da alínea anterior, onde tal se verifique.

Sim, é possível. Por exemplo:

T1	T2
Select * from t;	
	Update t set c = 0;
	Commit;
Select * from t;	

3 f) Considere agora o protocolo baseado em *timestamps* para o seguinte escalonamento das transações T1, T2, T3, e T4, com *timestamps* 1, 2, 3, e 4, respetivamente:

T1	T2	T3	T4
	Read(A)		
			Read(B)
		Write(D)	
Read(B)			
Write(A)			
	Write(C)		
		Write(A)	
	Write(B)		
			Read(A)
			Write(C)

Com este protocolo, haverá transações a abortar? Se sim, quais, onde, e devido a quê?

Sim:
1.º - T1, após Write(A), devido a Read(A) anterior de T2;
2.º - T2, após Write(B), devido a Read(B) anterior de T4;

Grupo 4 (4 valores)

4 a) Considere o modelo simplificado de *log* dado nas aulas, e o seguinte escalonamento executado em modo *read committed*, onde inicialmente A, B, e C têm o valor 0.

T1		B ← A+1	Commit		
T2	A ← 2			C ← 3*B	A ← C-A

I. Escreva a sequência de registos de *log* referentes a este escalonamento.

<T2 start>
<T2, A, 0,2>
<T1 start>
<T1, B, 0,1>
<T1 commit>
<T2, C, 0,3>
<T2, A, 2,1>

II. Apresente agora a sequência adicional de registos fruto do abortar da transação T2.

<T2, A, 2>
<T2, C, 0>
<T2, A, 0>
<T2 abort>

N.º:	

4 b) Complete (à direita) o registo de *log* abaixo, após uma falha do sistema. I.e. acrescente os registos associados à sua recuperação.

<T4, X, 3,4> <checkpoint {T3,T4}> <T4, Y, 0,1> <T5 start> <T4 commit> <T5, X, 4,5> <checkpoint {T3,T5}> <T3, D, 2,1> <T6 start> <T6, F, 9,4> <T3, D,2> <T12 start> <T12, B, 3,2> <T3 abort> <T5, D, 2,5> <T23 start> <T23, O2, operation-begin> <T23, B, 2,5> <T5, C, 2,4> <T23, O2, operation-end, (B,-3)> <T23, A, 3,4> <T12 commit> <T23, O4, operation-begin> <T23, B, 5,4> <T23, O4, operation-end, (B,+1)> <T23, B, 5> <T23, O4, operation-abort> <T5, O3, operation-begin> <T5, B, 5,0>	<T5, B, 5> <T23, A, 3> <T23, B, 5,2> <T23, O2, operation-abort> <T5, C, 2> <T23 abort> <T5, D, 2> <T6, F, 9> <T6 abort> <T5, X, 4> <T5 abort>

4 c) Para uma tabela ***aluno****(id, nome, escola, nota)*, dê um exemplo (baseando-se nos seus atributos) de fragmentação horizontal, e escreva uma consulta SQL que, nesse caso, poderia beneficiar de processamento paralelo.

Um fragmento (conjunto de registos) para cada *escola* (valor), armazenado no *site* dessa escola.

select escola, max(nota) from aluno group by escola

4 d) Considere uma base de dados distribuída por 13 sites, e 3 itens de dados, A, B, C. cada um replicado em 5 sites diferentes. Com um protocolo de gestão de *locks* baseado em maioria, poderá aqui haver *deadlocks*? Justifique.

Sim, por exemplo com 3 transações T1, T2, T3 a querer aceder a A, onde T1 e T2 obtêm 2 *locks* cada, e T3 obtém 1. Ficariam assim as 5 réplicas *locked*, sem nenhuma transação conseguir ter a maioria, impossibilitando o progresso.