Sistemas de Bases de Dados

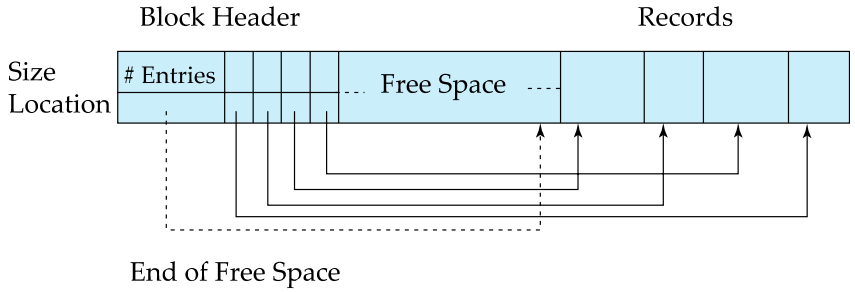
Exame de recurso (com consulta limitada: 3 folhas identificadas)
Duração: 3 horas

N.º:		**Nome:**	

Grupo 1 (5 valores)

1 a) Indique 2 meios físicos de armazenamento terciário (ou *offline*).

1 b) Considere a estrutura de *slotted page* (figura abaixo) para guardar vários registos de tamanho variável. Como sabe, os registos são mantidos contíguos no final do bloco para melhor se aproveitar o espaço livre. Assim, 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. Esta deslocação dos registos pode ter implicação na atualização de apontadores para os mesmos (e.g. em índices). O que é que os SGBDs costumam fazer para evitar esse problema?



1 c) Quais as vantagens e desvantagens dum índice esparso?

1 d) Qual a diferença entre um índice *B+-tree* e uma organização de ficheiro de dados em *B+-tree*?

1 e) Qual a diferença entre *B+-tree* e *B-tree* relativamente a um nó não terminal? Que implicações tal pode ter na eficiência de pesquisa num índice com essa estrutura?

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

Id	1	2	3	4	5	6	7	8
sexo	M	M	F	F	M	F	M	F
nota	OK	REP	REP	OK	REP	OK	OK	OK

N.º:	

Grupo 2 (5 valores)

2 a) Considere que queremos ordenar uma relação com os seguintes (33) registos (com apenas uma coluna: um inteiro) em disco: (45,34,4,78,3,14,17,52,65,35,56,9,3,23,18,8,1,33,77,88,55,44,22,11,2, 12,13,76,87,98,32,21,19). Seja o *blocking factor* f=2 e a capacidade da memória M=4 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).

2 b) O algoritmo básico de *nested-loop join* (iteração para todos os pares de tuplos) para 2 relações *r1* e *r2* (em disco) que não cabem em memória, requer no pior caso (M=2: 1 bloco de cada relação) $n_{r1}+b_{r1}$ *seeks* e $n_{r1}*b_{r2} + b_{r1}$ transferências de blocos (onde n representa o número de tuplos, e b o número de blocos da respetiva relação).

I. Pelas expressões, qual conclui ser a relação exterior e qual a interior? Justifique também ambas as expressões (de *seeks* e de transferências).

II. Se a menor relação couber em memória (melhor caso), bastam 2 *seeks* e $b_{r1}+b_{r2}$ transferências de blocos. Justifique.

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 um milhão (10^6) de valores de temperatura em 100 cidades europeias entre os anos 1979 e 2018, inclusive (há várias medições em falta, no espaço e no tempo, por diferentes motivos). Considere que neste sistema um inteiro ocupa 10 bytes, um real ocupa 40 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),
 hot as (select city, ano, dia, hora, val from temp
            where hora in (14,15,16) and ano<1981 and mes=7)
(select * from hot inner join stat
   on (hot.city=stat.city and val > avgt);
```

I. 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.º:	

II. Se for aplicado o algoritmo *block nested-loop join* sem recorrer a índices, com M=10, qual deverá ser a relação interior? Porquê?

III. Sem fazer contas, discuta sobre se o *Merge-Join* seria uma opção.

IV. Sem fazer contas, discuta sobre se o *Hash-Join* seria uma opção.

Grupo 3 (5 valores)

3 a) 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:	☐	☐	_____
S2:	☐	☐	_____
S3:	☐	☐	_____
S4:	☐	☐	_____

S1

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

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**	**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			

S4

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;	

N.º:	

3 b) Considere a tabela ***alunos****(id, nota)* já com os tuplos (1,18) e (2,13), previamente criada com:

```
create table alunos(id int primary key,
                    nota int check(nota between 0 and 20));
```

Apresente um escalonamento de transações com operações apenas sobre esta tabela, que seja serializável de vista, mas não de conflito:

3 c) O protocolo de *locking* de 2 fases garante escalonamentos serializáveis de conflito? Se sim, diga como seria dada a ordem de serialização. Se não, dê um exemplo.

3 d) Voltando à tabela ***alunos****(id, nota)* da alínea *b* (com tuplos <1,18> e <2,13>), considere as seguintes transações executando em *Snapshot Isolation* :

T1	**T2**
Insert into alunos values (3,12);	
Update alunos set nota=nota-1 where mod(id, 2) = 1;	
	Select * from alunos;
	Update alunos set nota=nota+1 where mod(nota, 2) = 1;
Commit;	
	Commit;

I. Alguma transação abortará ou bloqueará? Se sim, qual e onde/quando?

II. O que verá o utilizador como resultado da instrução '*select * from alunos*' dentro de T2?

III. Qual o conteúdo da tabela *alunos* após a conclusão de todo o escalonamento?

3 e) Como sabe, o uso de *locks* pode levar a *deadlocks*.

I. Indique uma possível estratégia de controlo de concorrência para evitar *deadlocks* (sem abortar transações).

II. Indique agora uma estratégia baseada nos *timestamps* das transações para quebrar *deadlocks* (abortando transações).

III. Num esquema baseado em *timeout*, há um limite de tempo para a obtenção dum *lock* — caso seja ultrapassado, a transação aborta e recomeça. Quais as vantagens e inconvenientes deste esquema?

N.º:	

Grupo 4 (5 valores)

4 a) Considere o modelo simplificado de *log* dado nas aulas, e o seguinte escalonamento executado em modo *read committed*, onde inicialmente X, Y, e Z têm o valor 1.

T1	X ← 3			Z ← 2*Y	X ← Z+X
T2		Y ← X+2	Commit		

I. Escreva a sequência de registos de *log* referentes a este escalonamento.	**II.** Apresente agora a sequência adicional de registos fruto do abortar da transação T1.

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, Z, 3,4> <checkpoint {T3,T5}> <T3, D, 2,1> <T6 start> <T6, F, 9,4> <T3, D,2> <T1 start> <T1, B, 3,2> <T3 abort> <T5, D, 2,5> <T2 start> <T2, O1, operation-begin> <T2, B, 2,5> <T5, C, 2,4> <T2, O1, operation-end, (B,-3)> <T2, A, 3,4> <T1 commit> <T2, O4, operation-begin> <T2, B, 5,4> <T2, O4, operation-end, (B,+1)> <T2, B, 5> <T2, O4, operation-abort> <T5, O3, operation-begin> <T5, B, 5,0>	

4 c) Qual o mecanismo base para permitir *fuzzy checkpointing*?
(Não é necessário descrever o processo)

4 d) Para uma tabela ***clientes****(id, nome, cidade, saldo)*, dê um exemplo (baseando-se nos seus atributos) de fragmentação vertical e escreva uma consulta SQL que, nesse caso, poderia beneficiar de processamento paralelo.

4 e) No protocolo de '*commit* em 2 fases' em sistemas distribuídos, o que tem de fazer o coordenador *C* duma transação *T*, e o que tem de acontecer para se realizar o *commit*?

4 f) Qual o interesse da estratégia de *semijoin*? I.e. para que serve?

N.º:	

4 g) Considere as *Queries LDAP* e como são especificadas.

I. De que formas se pode especificar uma query LDAP (i.e. como é que pode ser passada ao sistema)?

II. O que deve e o que pode constar numa tal *query*?

III. Que operações de álgebra relacional permitem tais *queries*?

Exame de SBD

- duração: 3 horas
- pode sair a partir das 10:30 (não antes)
- pode responder a lápis
- pode ter consigo 3 folhas agrafadas de consulta, identificadas com nome e número
- tenha o seu documento de identificação (e.g. cartão de estudante) visível ao seu lado
- não são permitidos equipamentos eletrónicos
- não pergunte pelas horas, por favor