DI- FCT/UNL **20 de julho de 2016**

Sistemas de Bases de Dados
Exame de Época Especial, 2015/16
Duração: 3 horas (sem consulta)

Grupo 1

Considere parte duma base de dados para realização de umas eleições legislativas (onde os atributos que constituem a chave primária estão sublinhados):

eleitor({NumBI,Nome,Sx,Eleit,HabLiterarias,Idade...}) partido({Sigla,Nome,Sede,Líder,Logo...})
votos({RefVot,Mesa,Freg,Conc,Distrito,Sigla}) candidato(NumBI,Sigla,Distrito,Ordem)

Para cada uma destas tabelas existe um índice clustered de árvore B+ sobre o(s) atributo(s) da chave primária. Assuma que no SGBD escolhido um nó da árvore B^+ pode conter cerca de 100 chaves de pesquisa e sabe-se que o tempo de um seek é dez vezes superior ao da transferência de um bloco (t_S = 10 * t_T), e a memória comporta apenas 100 blocos de 4KB.

Os registos de todas as tabelas têm dimensão variável, sendo que, em média, um registo da tabela de eleitores ocupa 1KB, um registo de partidos ocupa 20KB um registo da tabela de votos e de candidatos ocupa 100 bytes.

Sabemos ainda que num dado momento a tabela de eleitores tem 10.000.000 tuplos, a de votos 5.000.000 tuplos, a de partidos 15 tuplos e a de candidatos 3.000 tuplos. Pode assumir que existem 20 distritos em Portugal.

Nota: Neste grupo, sempre que se solicitarem exemplos, estes devem ser **exclusivamente** sobre esta base de dados. Além disso, **todas** as respostas deverão conter uma **breve justificação**.

1 a) Apresente o plano de execução que considere mais eficiente para responder à seguinte pergunta SQL (que retorna os nomes ordenados de todas as candidatas).

```
select eleitores.nome
from eleitores natural inner join candidato
where Sx = 'F'
order by eleitores.nome
```

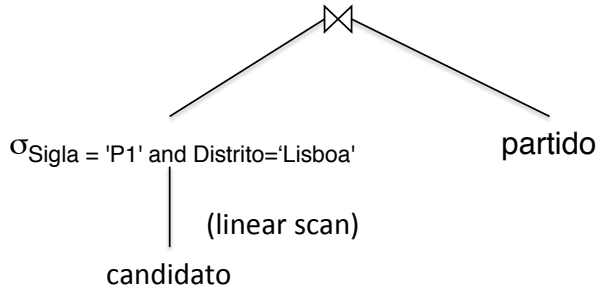
1 b) Poder-se-ia usar o mecanismo de *"slotted pages"* para armazenar todos os tuplos desta base de dados? Porquê?

1 c) Considere agora a seguinte pergunta SQL:

```
select *
from eleitores natural inner join candidatos
where HabLiterarias = 'Licenciado' and Nome = 'José Silva' and Sx = 'M'
```

Imagine que podia criar índices *non-clustered* de árvore B+ sobre qualquer dos atributos (individualmente ou multi-atributo) envolvidos nesta pergunta.
Indique se para algum dos índices o seu uso tornaria a execução da pergunta acima mais eficiente do que só com os índices *clustered* para as chaves (fornecendo um exemplo), e se existe algum índice que não tornaria mais eficiente a execução (basta um exemplo).

1 d) Compare o custo da execução do seguinte plano na figura quando a operação de junção é implementada por "*block nested loop join*" versus "*index nested loop join*", indicando explicitamente qual a relação interior e exterior para cada operação de junção de forma a minimizar o tempo de execução.



Página intencionalmente deixada em branco

1 e) Em geral o algoritmo *"merge-join"* não permite o uso de pipelining entre a junção e outras operações a montante. Mas há situações em que tal não causaria problema nenhum.

Apresente um exemplo de plano de execução de uma pergunta sobre a base de dados acima, que envolva pelo menos uma junção e uma seleção a montante da junção, em que o algoritmo da junção seja o *"merge-join"* e em que fosse possível usar pipelining para a junção.

1 f) Apresente um escalonamento de **operações SQL sobre a base de dados acima**, em duas transações concorrentes, que provoque um *deadlock*.

1 g) Considere a seguinte lista de eventos envolvendo três transações (as únicas a executar no sistema). Apresente o registo de log de acordo com o algoritmo estudado sabendo que no ínicio a idade é sempre 0.

	T1	**T2**	**T3**	**sistema**
1	begin transaction UPDATE eleitor SET idade = 19 WHERE NumBI = 5;			
2		begin transaction UPDATE eleitor SET idade = 37 WHERE NumBI = 2;		
			begin transaction UPDATE eleitor SET idade = 79 WHERE NumBI = 3;	
3	UPDATE eleitor SET idade = 18 WHERE NumBI = 5;			
4	ROLLBACK;			
5				**CHECKPOINT**
6			COMMIT**;**	
7				**CRASH!**
8				**RECOVER**

1 h) Alguns sistemas de bases de dados, como por exemplo o Oracle, usam protocolos optimistas multi-versão de controlo de concorrência, e onde a granularidade dos locks é ao nível dos tuplos, que garantem *"snapshot isolation"*. Apesar de garantir alguma forma de isolamento, este tipo de protocolos não garante que a execução concorrente seja sempre equivalente a uma das execuções sequenciais das transações.
Apresente um escalonamento de **operações SQL** em duas transações concorrentes em que se possa verificar isso mesmo. I.e. um escalonamento em que, usando por exemplo o Oracle, apesar de ambas as transações terminarem com sucesso o resultado seja diferente do que seria se se executasse primeiro uma qualquer delas até ao fim e só depois a outra.

Grupo 2

Nota: A resposta a cada uma das alíneas deste grupo **não pode em caso algum exceder uma página**.

2 a) *Porque é que, em geral, em sistemas de bases de dados não faz sentido usar a técnica de "static hashing" para organização de dados ?*

2 b) *Que problema existente em bases de dados distribuídas é endereçado pelo protocolo 2-phase commit?*

Na sua resposta refira qual (ou quais) das propriedades ACID é/são mais relevante(s) nesse problema. Mencione ainda porque é que em bases de dados centralizadas não faz sentido sequer falar neste protocolo.

2 c) *"Normalmente nas transações em bases de dados, a propriedade da consistência apenas requer que a base de dados esteja num estado consistente no final da transação, e não ao longo de toda a transação".*

Em que situações seria necessário ou vantajoso exigir consistência ao longo de toda uma transação e porquê?
E que operações sobre bases de dados não seriam possíveis se não fosse como descrito acima? Dê um exemplo concreto de base de dados e operação sobre ela que não seria possível se a consistência fosse sempre verificada ao longo de toda a transação.