DI- FCT/UNL **11 de Janeiro de 2010**

Sistemas de Bases de Dados

Exame de Época Normal, 2009/10

Duração: 3 horas (sem consulta)

Grupo 1

Considere parte duma base de dados de um banco, onde se regista informação sobre (algumas) operações dos clientes. Essa base de dados inclui as seguintes tabelas (onde em cada uma delas apenas se apresentam parte dos atributos, e os atributos que constituem a chave primária estão sublinhados):

clientes({CodCl,NomeC,Sx,CatRendimentos,HabLiterarias...}) contas({NumConta, codCliente,CodSuc,...})
movimentos({NumMov,NumConta,Tipo,Valor,...}) sucursais(CodSuc,NomeSuc,Morada,...)

Para cada uma destas tabelas existe um índice (non-clustered) de árvore B+ sobre o(s) atributo(s) da chave primária.

Tendo em conta o sistema de gestão de bases de dados usado, tipicamente cabem num bloco 20 tuplos da tabela de clientes ou da tabela sucursais, ou 30 tuplos da tabela contas ou 50 tuplos da tabela movimentos. Para os exercícios, assuma que o sistema que suporta esta base de dados permite ter em memória apenas 100 blocos.

Sabemos ainda que num dado momento a tabela clientes tem 100.000 tuplos, a de sucursais 10 tuplos, a de contas 300.000 tuplos e a de movimentos 5.000.000 tuplos.

Nota: Sempre que, neste grupo, 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 dois planos para execução da seguinte pergunta SQL (que retorna os vários movimentos de contas da cliente "Maria"), justificando qual deles deverá ter um menor custo na base de dados em causa.

```
select movimentos.*
from clientes natural inner join contas natural inner join movimentos
where NomeC = "Maria"
```

1 b) Considere que o sistema apenas implementa para junções o algoritmo de "nested loop join", eventualmente usando índices quando disponíveis.
Para cada uma das junções abaixo indique qual lhe parece ser a melhor ordem para fazer as junções e, em cada operação, qual deveria ser a "inner table" e qual deveria ser a "outer table" no "nested loop":

- contas $\bowtie$ sucursais
- clientes $\bowtie$ contas $\bowtie$ movimentos

1 c) Apresente um exemplo de pergunta só sobre a relação de clientes, em que o uso de índices de árvore B+ apenas tornaria a execução dessa pergunta menos eficiente.

1 d) Suponha agora que o sistema lhe permitia definir índices de bitmap. Apresente um exemplo duma pergunta em que a existência dum índice de bitmap tornaria muito mais eficiente a execução, e *apresente um outro* em que não faça qualquer sentido usar índices de bitmap.

1 e) Dê um exemplo duma pergunta envolvendo uma junção, em que o algoritmo de "merge-join" lhe pareça o mais adequado para a junção.

1 f) Por vezes pode acontecer que o melhor plano de execução para uma pergunta contenha sub-planos que não sejam os melhores para a sub-pergunta correspondente (se esta fosse feita isoladamente). Dê um exemplo de uma pergunta em que tal aconteça, dizendo qual o melhor plano para a pergunta completa, qual a sub-pergunta e qual seria o melhor plano para a sub-pergunta se esta fosse feita isoladamente.

1 g) Apresente um escalonamento de operações SQL em duas transacções concorrentes em que em que o resultado seja diferente consoante se esteja em modo de isolamento Read uncommited, Read commited ou Serializable, mostrando qual o resultado para cada um dos 3 modos de isolamento.

1 h) Alguns sistemas de bases de dados, como por exemplo o Oracle, usam protocolos optimistas multi-versão de controlo de concorrência que garantem *"snapshot isolation"*. No entanto, o *"snapshot isolation"* não garante que as transacções sejam sempre serializáveis (e.g. o Oracle prevê mesmo um erro de assinalando que não pode serializar as transacções).
Apresente um escalonamento de operações SQL em duas transacções concorrentes em que se possa verificar isso mesmo, i.e. em que o usando um protocolo optimista multi-versão com *"snapshot isolation"* não se consigam serializar as transacções. (Por outras palavras, um escalonamento em que e.g. o Oracle apresentasse o tal erro).

1 i) Considere agora que a base de dados é distribuída sendo que as relações de contas e movimentos se encontram fragmentadas, com os tuplos correspondentes às contas de cada sucursal apenas localizados num servidor localizado na sucursal, e em que as relações de clientes e sucursais se encontram apenas num servidor (e.g. na sede do banco).
Apresente uma boa estratégia para processamento da pergunta: *Para cada cliente, qual o seu nome, categoria de rendimentos, habilitações literárias e total de movimentos de contas.*

Grupo 2

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

2 a) *"A maior parte dos sistemas de gestão de bases de dados trata de forma diferente o armazenamento de atributos que guardem valores com pequena dimensão, do armazenamento de atributos que guardem valores com grande dimensão."*

Apresente pelo menos duas vantagens para que o façam.

2 c) *"O fase de optimização é normalmente essencial para a execução de perguntas em SQL. Mas muitos dos sistemas de gestão de bases de dados têm formas de curto-circuitar a fase de optimização, até porque há situações em que essa fase não traz grandes vantagens, e pode mesmo ser desvantajosa."*

Caracterize essas situações e mostre um exemplo concreto (de base de dados e pergunta) em que seria preferível não recorrer ao optimizador.

2 b) *"Os protocolos para controlo de acesso concorrente a bases de dados baseados em locks são usualmente chamados de pessimistas"*.

Em que é que os protocolos baseados em locks são pessimistas? Por oposição, o que seria um protocolo optimista. Conhece algum método optimista de controlo concorrente a bases de dados?

.