DI- FCT NOVA

Sistemas de Bases de Dados

2º teste, (modelo)

Duração: 1 hora + 30 minutos (sem consulta)

Grupo 1

Considere uma base de dados para registar consultas feitas num hospital e receitas de fármacos feitas em cada consulta. 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):

fármacos({NumF,NomeF,...}) consultas({NumCons, codMed,NomePaciente, Data,...})
receitas(NumCons,NumF,quantidade,...) médicos({CodMed,Nome,Sx,Especialidade,...})

Para cada uma das tabelas há um índice 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 fármacos ou da tabela médicos, ou 30 da tabela consultas, ou 40 da tabela receitas. 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 fármacos tem 500.000 tuplos, a de médicos 50 tuplos, a de consultas 10.000 tuplos e a de receitas 50.000 tuplos.

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 dois planos para execução da seguinte pergunta SQL (que retorna os vários fármacos alguma vez receitados pelo médico João), justificando qual deles deverá ter um menor custo na base de dados em causa.

```
select distinct NomeF
from fármacos natural inner join receitas natural inner join consultas
                natural inner join médicos
where Nome = "João"
```

1 b) Considere que o sistema apenas implementa para junções o algoritmo de *(block) nested loop join*, eventualmente usando índices quando disponíveis. Para a junção das tabelas de **1a** 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*.

1 c) Dê um exemplo duma pergunta de junção em que o algoritmo de *merge-join* lhe pareça o mais adequado.

1 d) Apresente um exemplo de pergunta SQL e plano de execução em que não seja possível usar exclusivamente *pipelining* na avaliação de expressões intermédias.

1 e) Qual o melhor plano para a execução da seguinte pergunta SQL?

```
select * from consultas A
where NomePaciente = "Maria" and
        exists (select Nome from médicos B where A.CodMed = B.CodMed)
```

Pode assumir que estão definidas chaves estrangeiras na bases de dados, devendo nesse caso explicitar que chave(s) estrangeira(s) está a assumir que existe(m).

Grupo 2

Nota: Dê respostas **breves**.

2 a) Assuma que cada um dos tuplos abaixo ocupa um bloco de memória, e que a memória do sistema apenas permite que sejam lidos 3 blocos de cada vez. Apresente os vários *runs* criados pelo algoritmo *external sort-merge* para a ordenação pelo 1º atributo dos tuplos: (C,...); (A,...); (T,...), (J,...); (B,...); (S,...); (F,...); (G,...); (H,...); (E,...); (R,...); (D,...)

2 b) Em certas perguntas de junção não é possível usar o algoritmo *hash partitioned join* para processamento paralelo da pergunta, podendo-se usar, em alternativa o *fragment-and-replicate join* ou o *asymmetric fragment-and-replicate join*. Para que tipo de perguntas isso se passa? Para esse tipo de perguntas, em que situações é preferível usar o *fragment-and-replicate join* e em que situações é preferível o *asymmetric fragment-and-replicate join*?

2 c) Indique duas situações distintas nas quais a declaração de chaves estrangeiras pode permitir que o otimizador do SGBD obtenha melhores planos de execução.