Licenciatura em Engenharia Informática e Computação
Tecnologias de Sistemas de Gestão de Bases de Dados
2000/2001

Exame de Avaliação

29 de Junho de 2001

NOME: Resolução do Exame _____

1. Armazenamento de dados: ficheiros e índices [10 pontos]

Considere uma relação guardada em ficheiro de registos por ordem aleatória (*heap file*) para a qual existe apenas um índice não aglomerado num campo *nota*.

a) Diga se para responder a uma interrogação que pretende os registos com $nota > 10$ o uso deste índice é a melhor escolha e justifique devidamente a sua resposta.

Resposta:
Não é a melhor escolha. Como o índice é disperso cada entrada de dados que verifique a condição pode conter um rid aponta para uma página diferente, levando a tantos I/O quantas as entradas que verificam a interrogação de intervalo.

Melhor seria fazer um scan porque assim é garantido que cada página só é lida uma vez.

2. Ordenação externa [35 pontos]

Considere um disco com tempo médio de procura (*seek time*) de 10ms, *rotational delay* de 5ms e tempo de transferência de 1ms para uma página de 4k. O custo de ler ou escrever uma página é, então, de 16ms. No entanto, se as páginas forem lidas ou escritas em sequência, o tempo total será igual ao tempo para encontrar a primeira página (10ms+5ms) mais um 1ms por página para transferir os dados.

Considere que tem 320 páginas de buffer e pretende ordenar um ficheiro com 10 000 000 de páginas.

a) Explique porque é que não é boa ideia usar as 320 páginas como suporte de memória virtual e escolher um algoritmo de ordenação para memória primária, por exemplo Quicksort.

Resposta:
O Quicksort é eficiente em memória primária; neste caso o número de páginas é demasiado elevado para caber em memória e por isso a utilização de memória virtual (suportada em disco) levaria a muitos I/O de páginas.

Melhor seria utilizar um algoritmo de ordenação externa, pensado para este caso.

b) Considerando apenas o custo em I/O de páginas, calcule o custo de de obter corridas ordenadas de 320 páginas cada, no primeiro passo de um algoritmo de ordenação externa.

Resposta:
O número de corridas de 320 páginas cada é de $\lceil 10\ 000\ 000/320 \rceil = 31\ 250$.

O custo em I/O (não considerando CPU) por corrida é de 2*(10+5+1*320) = 670 ms.

O custo total do passo 0 é então de 31 250 * 670 = 20 937 500ms.

c) Avalie o custo de completar a ordenação através de junções de 319-vias (*319-way merge*).

Resposta:

Serão necessários mais dois passos de junção. O 1º passo produz ⌈31 250/319⌉ = 98 corridas ordenadas que podem ser combinadas no passo seguinte.

Em cada passo, cada página é lida e escrita individualmente com um custo de 16ms.

O custo da junção é, assim, 2*(2*16) * 10 000 000 = 640 000 000ms.

d) Avalie o custo de completar a ordenação através de junções de 256-vias, com 256 buffers de entrada de 1 página e um buffer de saída de 64 páginas.

Resposta:

Continuam a ser necessários dois passos. O 1º passo produz ⌈31 250/256⌉ = 123 corridas ordenadas.

Cada página é lida e escrita em cada passo, mas com custo diferente devido ao acesso em bloco.

Para ler temos 2*16*10 000 000 = 320 000 000ms.

As páginas são escrita em blocos de 64 páginas. O custo por bloco é de 10+5+1*64 = 79 ms, o número de blocos é 10 000 000/64 = 156 250 e o custo por passo é de 156 250 * 79 = 12 343 750ms. O custo total de escrita é de 2 * 12 343 750 = 24 687 500ms.

O custo total é 320 000 000 + 24 687 500 = 344 687 500ms (95.7h).

3. Optimização de interrogações [15 pontos]

Considere o seguinte esquema relacional:

```
Fornecedor(id: integer, nome: char(40), cidade: char(20))
Peças(id: integer, nome: char(40), preço: real)
Fornece(f-id: integer, p-id: integer)
```

e a seguinte interrrogação em SQL:

```
SELECT F.nome, P.nome
FROM Fornecedores F, Peças P, Fornece N
WHERE F.id=N.f-id AND P.id=N.p-id AND F.cidade='Porto' AND P.preço<1000;
```

a) Diga que informação acerca destas relações o optimizador deve possuir para poder seleccionar um bom plano de execução para a interrogação dada.

Resposta:

O optimizador precisa de saber que índices existem e de que tipo são (B+-tree, hash) nos campos `F.id`, `N.f-id`, `P.id`, `N.p-id`, `F.cidade`, `P.preço`. Para além disso, precisa saber estatísticas ácerca da instância da BD sobre valores mais elevados e mais baixos dos índices e distribuição entre campos.

b) Explique que índices poderiam ajudar na execução desta interrogação.

Resposta:

Um índice ordenado aglomerado em `P.preço` ajudaria a interrogação de intervalo.

Uma B+-tree em `F.id`, `N.f-id`, `P.id` ou `N.p-id` poderia ser usada num *sort-merge* só em índices.

c) Diga de que maneira poderiam ser afectados os planos produzidos se fosse adicionado `DISTINCT` à interrogação.

Resposta:

Para suportar `DISTINCT` é necessário ordenar os resultados (a menos que já estejam ordenados) e fazer um scan para detectar ocorrências múltiplas. O optimizador teria de ter isso em conta ao determinar os planos.

4. Limitações do Modelo Relacional e ODMG [10 pontos]

Aplicações de suporte a CAD, CAM, CASE, multimédia, publicação digital e GIS têm colocado a nu limitações do Modelo Relacional e levaram no passado recente à introdução de outros modelos.

a) Enumere e caracterize as principais fraquezas do Modelo Relacional e refira as melhorias propostas noutros modelos de dados, mais adaptados a estas aplicações.

Resposta:

- fraca representação das entidades do mundo real, levando a joins.
- homogeneidade da estrutura de dados horizontal (todos os tuplos têm os mesmos atributos) e vertical (os valores das colunas pertencem ao mesmo domínio) — fosso semântico; modelos RO e OO propõem NF^2, BLOBs.
- suporte pobre de restrições de integridade e de empresa (só entidade + integridade referencial) o que leva a ter parte da semântica do problema nos programas.
- sobrecarga semântica entidade/associação em tabelas; foram propostos modelos semânticos e modelos OO e RO.
- linguagem com número de operações limitado e não extensível; foram propostos ADTs.
- não trata recursão (e.g. fecho transitivo); proposta no SQL3.
- para fazer aplicações é necessário usar dois sistemas de tipos diferentes — desaptação de impedâncias; modelos OO e RO resolvem este problema.

5. SQL3, ADTs e Colecções [20 pontos]

Suponha que tem uma base de dados com informação relativa a uma árvore genealógica, com a seguinte informação de pessoas e respectivos casamentos. A pessoa é identificada pelo seu bi e guarda-se informação acerca do nome, data de nascimento, identificação da mãe, identificação do pai, sexo e morada (rua e cidade). Dos casamentos guarda-se a identificação das duas pessoas, a data em que se realizou, a data do divórcio, a identidade de todos os filhos e, de forma redundante, o número de filhos. Consideram-se válidas as seguintes restrições de integridade:

R1: O bi de pessoa é chave candidata.

R2: O bi do marido em conjunto com o bi da mulher formam uma chave candidata de casamento.

R3: O sexo das pessoas é `'M'` ou `'F'`

R4: A data do divórcio é maior que a data do casamento.

R5: A idade de um filho é sempre menor que a dos pais.

R6: Para qualquer casamento, o número de filhos é igual ao número de pessoas com a mãe igual à mulher do casamento e o pai igual ao marido do casamento.

a) Considere que tem ao seu dispor um SGBD relacional-objecto, como por exemplo o Oracle 8i ou o PostgreSQL, com a possibilidade de definir, por exemplo, tipos abstractos, tipos colecção, tipos referência e tabelas encaixadas. Apresente um esquema relacional-objecto (através dos respectivos comandos SQL) para a árvore genealógica apresentada, sem esquecer as restrições de integridade R1 a R4.

Resposta:

```
// usando a notação usada nas Teóricas
CREATE TYPE Pessoa AS Object (
    bi          CHAR(8) UNIQUE NOT NULL,
    nome        CHAR(30)
    sexo        CHAR(1) CHECK (sexo IN ('F', 'M')),
    data_nasc   DATE,
```

```
    bi_mae      REF Pessoa NOT NULL,
    bi_pai      REF Pessoa,
    morada      ROW (rua CHAR(30), cidade CHAR(20))
);
CREATE TABLE Pessoas OF Pessoa (
    bi          PRIMARY KEY
);
CREATE TABLE Casamentos (
    marido      REF Pessoa NOT NULL,
    mulher      REF Pessoa NOT NULL,
    data_ini    DATE NOT NULL,
    data_fim    DATE,
    filhos      LIST (REF Pessoa),
    #filhos     INTEGER DEFAULT 0,
    CONSTRAINT ck_datas CHECK (data_fim > data_ini),
    CONSTRAINT pk_casamento PRIMARY KEY (marido,mulher)
);
```

6. Módulos Persistentes em SQL3 [10 pontos]

Para a base de dados do problema 5, por forma a simplificar as answers, considere o seguinte esquema relacional:

```
Pessoa (bi: CHAR(8), nome, data_nascimento, bi_mae, bi_pai, sexo)
Casamento (bi_marido, bi_mulher, data_inicio, data_fim, #filhos)
```

a) Apresente um módulo persistente de servidor com os seguintes procedimentos:

```
numeroFilhos(bi_pessoa) // devolve o número de filhos da pessoa
casadaCom(bi_mulher) // devolve o bi do marido actual da pessoa
numeroIrmaosDe(bi_pessoa) // devolve o número de irmãos da pessoa
```

Resposta:

```
CREATE MODULE genealogia
        LANGUAGE SQL;
    CREATE PROCEDURE numeroFilhos(pessoa CHAR(8), OUT nfilhos INTEGER) AS
    BEGIN
        SELECT count(*) INTO nfilhos
        FROM Pessoas
        WHERE bi_pai = pessoa OR bi_mae = pessoa;
    END;
    CREATE PROCEDURE casadaCom(pessoa CHAR(8), OUT marido CHAR(8)) AS
    BEGIN
        SELECT bi_marido INTO marido
        FROM Casamentos
        WHERE bi_mulher = pessoa AND data_fim IS NULL;
    END;
    CREATE PROCEDURE numeroIrmaosDe(pessoa CHAR(8), OUT n_irmaos INTEGER) AS
    BEGIN
        DECLARE pai, mae: CHAR(8);
        SELECT bi_pai INTO pai
        FROM Pessoas
        WHERE bi = pessoa;
        SELECT bi_mae INTO mae
        FROM Pessoas
```

```
        WHERE bi = pessoa;
        SELECT count(*) INTO n_irmaos
        FROM Pessoas
        WHERE bi_pai = pai AND bi_mae = mae AND bi <> pessoa;
    END;
END;
```

7. Restrições de Integridade e Gatilhos [20 pontos]

Considere novamente a base de dados e o respectivo esquema relacional apresentado no problema 6.

a) Escreva uma asserção em SQL3 para impôr a restrição R5.

Resposta:

```
CREATE ASSERTION idade_filhos_R5
    AFTER
        INSERT ON Pessoas,
        UPDATE OF bi_pai, bi_mae ON Pessoas,
    CHECK ( NOT EXISTS
        ( SELECT * FROM Pessoas AS P1
          WHERE data_nasc <=
                SELECT data_nasc FROM Pessoas AS P2
                WHERE P2.bi=P1.bi_pai
          OR data_nasc <=
                SELECT data_nasc FROM Pessoas AS P3
                WHERE P3.bi=P1.bi_mae
        ) );
```

b) Escreva um ou mais gatilhos em SQL para impor a restrição R6 de forma incremental: ao inserir, eliminar ou actualizar uma linha da tabela `Pessoa`, deve ser actualizada, se existir, a instância respectiva da tabela `Casamentos`.

Resposta:

```
CREATE TRIGGER filhos_R6_1
    AFTER
        UPDATE OF bi_pai, bi_mae ON Pessoas,
    FOR EACH ROW
    WHEN ( NEW.bi_pai <> OLD.bi_pai OR NEW.bi_mae <> OLD.bi_mae )
    BEGIN
        UPDATE Casamentos SET #filhos = #filhos -1
          WHERE bi_marido = OLD.bi_pai AND bi_mulher = OLD.bi_mae;
        UPDATE Casamentos SET #filhos = #filhos +1
          WHERE bi_marido = NEW.bi_pai AND bi_mulher = NEW.bi_mae;
END;
CREATE TRIGGER filhos_R6_2
    AFTER
        INSERT ON Pessoas,
    FOR EACH ROW
    WHEN ( TRUE )
    BEGIN
        UPDATE Casamentos SET #filhos = #filhos +1
          WHERE bi_marido = NEW.bi_pai AND bi_mulher = NEW.bi_mae;
END;
CREATE TRIGGER filhos_R6_3
    AFTER
        DELETE ON Pessoas,
```

```
    FOR EACH ROW
    WHEN ( TRUE )
    BEGIN
        UPDATE Casamentos SET #filhos = #filhos -1
          WHERE bi_marido = OLD.bi_pai AND bi_mulher = OLD.bi_mae;
END;
```

8. Estrutura Lógica de Documentos XML [10 pontos]

Considere o seguinte DTD para documentos XML:

```
<!DOCTYPE Univ [
    <!ELEMENT UNIVERSIDADE (DEPARTAMENTO+)>
    <!ELEMENT DEPARTAMENTO (DOCENTE*)>
    <!ATTLIST DEPARTAMENTO Cod ID #REQUIRED
                  Nome CDATA #REQUIRED>
    <!ELEMENT DOCENTE (LOCAL, CARGO+)>
    <!ATTLIST DOCENTE Cod ID #REQUIRED
                  Nome CDATA #REQUIRED
                  Tipo (noite | dia ) "dia"
                  Departamentos IDREFS>
    <!ELEMENT LOCAL (#PCDATA)>
    <!ELEMENT CARGO (#PCDATA)>
  ]>
```

a) Verifique se o documento XML apresentado de seguida está conforme com o DTD apresentado (isto é, se é válido) e, no caso de não estar, assinale os pontos onde isso se verifica.

```
<?XML VERSION="1.0" STANDALONE="no"?>
<!DOCTYPE Univ SYSTEM "../DTDs/Univ.dtd">
    <UNIVERSIDADE>
      <DEPARTAMENTO Cod="D1" Nome="DEEC">
        <DOCENTE Cod="JCL" Nome="jlopes" Departamentos="D1">
          <LOCAL>Manteigas</LOCAL>
          <CARGO>Prof. Auxiliar</CARGO>
          <CARGO>Prof. Agregado</CARGO>
        </DOCENTE>
        <DOCENTE Cod="JCL2" Nome="jlopes" Tipo="noite">
          <LOCAL>Porto</LOCAL>
        </DOCENTE> // 1. Falta o CARGO
      </DEPARTAMENTO>
      <DEPARTAMENTO Cod="D2" Nome="DEQ">
        <DOCENTE Cod="JPF" Nome="jpascoal" Tipo="dia" Departamentos="D1">
          <LOCAL>Porto</LOCAL>
          <CARGO>Prof. Auxiliar</CARGO>
          <LOCAL>Manteigas</LOCAL> // 2. LOCAL está a mais
          <CARGO>Prof. Agregado</CARGO>
        </DOCENTE>
      </DEPARTAMENTO>
      <DEPARTAMENTO Cod="D3" Nome="DEC">
        <DOCENTE Sigla="JCA" Departamentos="JCL JPF"> // 3. Cod="JCA"
          <CARGO>Prof. Auxiliar</CARGO>
```

```
            <CARGO>Prof. Agregado</CARGO>
          </DOCENTE> // 4. Falta atributo Nome e o elemento LOCAL
        </DEPARTAMENTO>
        <DEPARTAMENTO Cod="D3" Nome="DEC"> // 5. D3 duplicado
        </DEPARTAMENTO>
      </UNIVERSIDADE>
```

9. Transformação e apresentação de XML [10 pontos]

Considere novamente o DTD apresentado no problema 8.

a) Apresente um conjunto de regras de transformação XSLT que permitam passar para HTML para ser mostrado num navegador Web, os nomes dos docentes constantes de documentos XML de acordo com este DTD.

Resposta:

```
<?XML VERSION="1.0"?>
<xsl:stylesheet xml:xsl="htpp://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
    <HTML>
    <BODY>
    <xsl:for-each select="DOCENTE">
        <xsl:value-of select="@Nome">
        <BR>
    </xsl:for-each>
    </BODY>
    </HTML>
</xsl:template>
</xsl:stylesheet>
```

10. Sistemas de Apoio à Decisão [10 pontos]

Considere a seguinte tabela de factos e uma das tabelas de dimensões:

```
Vendas(produto, cor, dia, quantidade, preco)
Produtos(id, categoria, preco)
```

Suponha que é submetida a seguinte interrogação:

```
SELECT produto, dia, SUM(quantidade*preco)
FROM Vendas, Produtos
GROUP BY categoria, dia;
```

a) Apresente uma interrogação em SQL que, aplicada de seguida, efectue uma operação de "roller-up".

Resposta:

```
SELECT produto, dia, SUM(quantidade*preco)
FROM Vendas, Produtos
GROUP BY dia;
```

b) Apresente um exemplo em SQL de aplicação de uma das operações de "pivoting", por exemplo, "slicing".

Resposta:

```
SELECT produto, dia, SUM(quantidade*preco)
FROM Vendas, Produtos
WHERE preco = 1000
GROUP BY categoria, dia;
```

FIM.