FEUP

Licenciatura em Engenharia Informática e Computação
Tecnologia de Sistemas de Gestão de Bases de Dados
2001/2002

Exame de Avaliação

12 de Julho de 2002

Resolução do Exame
(23 de Julho de 2002)

Observe por favor as seguintes instruções:

- Leia cuidadosamente o exame até ao fim por forma a escolher a sua estratégia.
- O exame tem a duração máxima de duas horas e meia (150 minutos).
- O exame é com consulta de todo o material próprio trazido para o efeito.
- Deve responder nos espaços fornecidos neste exame, podendo usar, em último recurso, o espaço das costas da folha.
- O exame tem 10 perguntas, com as pontuações indicadas, totalizando 100 pontos.

Problema	1	2	3	4	5	6	7	8	9	10	Total	NOTA
Máx. Pontos	5	15	15	5	10	10	10	10	10	10	100	–
Pontos												

João Correia Lopes

1. Armazenamento de Dados: Ficheiros e Índices [5 pontos]

Na *buffer pool* mantida por um gestor de buffers de um SGBD pode ocorrer um fenómeno conhecido como *sequential flooding*.

Diga em que consiste este fenómeno e dê um exemplo ilustrativo de como ele pode acontecer na prática, de acordo como uma dada política de troca de páginas.

Resposta:
Certas implementações de operadores relacionais (por exemplo a junção) requerem *scans* repetidos de uma relação. Considerando que há 10 *frames* disponíveis e que o ficheiro a ordenar tem 11 ou mais páginas, se for usada a política de troca de páginas LRU, cada *scan* resulta na leitura de todas as páginas do ficheiro. Esta situação é conhecida como *sequential flooding* e, neste caso, LRU é a pior estratégia.

2. Indexação e Ordenação [15 pontos]

Considere que se pretende ordenar um ficheiro com 4500 registos de 48 bytes cada um, guardados em páginas de 512 bytes. A chave de ordenação (procura) tem 4 bytes, os identificadores de registo (*rid*) têm 8 bytes e os identificadores de página têm 4 bytes.

a) Supondo que existe um índice em *B+ tree* para o campo de ordenação, calcule o custo de retirar os registos pela ordem desejada para o caso (1) em que o índice é agrupado e (2) o índice não é agrupado.

Resposta:
Cada página pode conter até $\lfloor 512/48 \rfloor = 10$ **registos. Para guardar o ficheiro são necessárias** $N = 4500/10 = 450$ **páginas.**

O custo de retirar os registos por ordem usando a B+ tree é igual ao custo de atravessar a árvore da raiz até à folha mais à esquerda (3 ou 4 I/O) mais o custo de retirar as páginas do conjunto sequência.

No caso (1) é necessário retirar todas as páginas de dados (450 I/O), o que dá um custo total de $3 + 450 = 453$ **I/O.**

No caso (2) é necessário percorrer (*scan*) todas as páginas de dados e para cada entrada de dados retirar, no pior caso, uma página de disco. São necessárias $\lfloor 4500 * 12/512 \rfloor = 106$ **páginas de dados, já que cada entrada tem 12 bytes. O custo, neste caso, é de** $3 + 106 + 4500 = 4609$ **I/O.**

b) Supondo que estão disponíveis 4 páginas de buffer, calcule o número de corridas ordenadas produzidas pelo 1º passo de um algoritmo de ordenação externa, calcule o número de passos necessários para ordenar o ficheiro e o custo total da ordenação em termos de I/O.

Resposta:
Temos $N = 450$ **e** $B = 4$**. O número de corridas é** $\lceil N/B \rceil = 113$**; cada corrida tem 4 páginas, excepto a última que tem 2 páginas. O número de passos** $P = log_{B-1}\lceil N/B \rceil + 1 = log_3 113 + 1 = 6$ **passos. O custo total é de** $2 * N * P = 5400$ **I/O.**

c) Calcule o número de registos do maior ficheiro que consegue ordenar com as 4 páginas de buffer em apenas 2 passos; refaça os cálculos para o caso de existirem 257 páginas de buffer.

Resposta:
Para juntar no 2º passo as $\lceil N/B \rceil$ **corridas do 1º passo temos** $B - 1 \geq \lceil N - 1 \rceil$**, ou seja,** $3 \geq \lceil N/4 \rceil$ **o que dá** $N = 12$ **páginas. O maior ficheiro que se pode ordenar com 4 páginas de buffer tem** $12 * 10 = 120$ **registos. No caso de o número de páginas disponível ser** $B = 257$ **temos** $256 \geq \lceil N/257 \rceil$ **ou seja,** $N = 65792$ **páginas,** 657920 **registos.**

3. Optimização de interrogações [15 pontos]

Considere o seguinte esquema de relação de empregados de uma dada empresa:

```
Emp(nome, cargo, cidade, dept)
```

em que os domínios são *strings* do mesmo tamanho, e a seguinte interrogação em SQL:

```
SELECT nome
FROM Emp
WHERE cargo="Capataz" AND dept="Obras";
```

Considere ainda que apenas 10% dos registos satisfaz a condição de selecção `cargo="Capataz"`, que apenas 10% dos registos satisfaz `dept="Obras"`, que 5% dos registos satisfaz a conjunção das duas condições, que a relação contém 10 000 páginas e que existem 10 páginas de buffer disponíveis.

a) Supondo a existência apenas de um índice aglomerado do tipo *B+ tree* no campo `cargo`,descreva o melhor plano para responder à pergunta e calcule o respectivo custo em I/O.

Resposta:
O melhor plano para responder à pergunta envolve uma procura no índice B+ tree para encontrar o 1º índice com `cargo = "Capataz"` **(custo 2 I/O) seguida de um *scan* do índice por forma a tirar as páginas que satisfazem a condição (10 000 x 10%). A selecção** `dept="Obras"` **e a projecção são efectuadas *on-the-fly*. Como o campo** `cargo` **tem 1/4 do tamanho do registo temos 2 500 páginas de dados do índice e o custo de percorrer o índice é** $2500 * 10\%$**. O custo total é, então, de** $2 + 2500 * 10\% + 10000 * 10\% = 1252$ **I/O.**

b) Supondo a existência apenas de um índice aglomerado do tipo *B+ tree* nos campos <`cargo,dept`>, descreva o melhor plano para responder à pergunta e calcule o respectivo custo em I/O.

Resposta:
Como os os dois campos do índice perfazem 1/2 do tamanho do registo, o número de páginas de dados é de $10000/2 = 5000$**. Como a chave de procura é suportada pelo índice** <`cargo,dept`>**, este pode ser usado tirando partido da maior selectividade da conjunção das condições, com um custo total de** $2 + 5000 * 5\% + 10000 * 5\% = 752$ **I/O.**

4. Limitações do Modelo Relacional [5 pontos]

Para resolver limitações do Modelo Relacional foram propostos novos modelos de dados, nomeadamente o Modelo Relacional-Objecto e o Modelo Orientado aos Objectos. Nos últimos 15 anos tem-se assistido a um aumento sistemático do número de novas instalações Relacional-Objecto e de migração de instalações existentes para este modelo, em contraste com o número reduzido de instalações de SGBDs Orientadas aos Objectos.

Identifique e descreva brevemente o que, na sua opinião, tem contribuído para esta situação.

Resposta:
O modelo relacional-objecto com a introdução de tipos de dados abstractos (ADT) resolve as mesmas limitações que são resolvidas pelo modelo orientado aos objectos e mantém o uso de uma linguagem de interrogação baseada em SQL. Ao contrário, nos SGBDs O-O não existe uma linguagem de interrogação com as características de declarativiadade e facilidade de utilização do SQL. O facto de poderem continuar a usar SQL (compatibilidade com esquemas relacionais SQL92), nomeadamente as mesmas interrogações e apenas as novas características do modelo quando necessário, leva as empresas a manterem os seus programadores e fazerem uma transição suave na migração das suas aplicações e em novas instalações. Nos modelos O-O, para além da inexistência de linguagem de interrogação, a evolução do esquema da base de dados é mais difícil.

5. SQL3, ADTs e Colecções [10 pontos]

Suponha que pretende guardar numa base de dados informação sobre os empregados de uma dada empresa.

Para cada empregado, identificado pelo número de contribuinte (`nctb`), deve guardar-se `anos` (os anos que o empregado trabalhou para a empresa) e `nome`. Há empregados "a prazo" e normais e o salário é calculado, de maneira diferente para os dois tipos, por um método `calcSalario` que aceita `anos` como parâmetro. Para cada empregado normal deve ser guardado o nome e a idade de cada filho. Para cada departamento, identificado pelo codigo (`cod`), deve guardar-se `nome` e `emps` (conjunto de empregados que trabalham no departamento).

Considere ainda válidas as seguintes restrições de integridade:

R1: Nenhum empregado a prazo trabalha na empresa mais do que 3 anos.

R2: Qualquer empregado normal ganha mais do que qualquer um dos empregados a prazo.

R3: Nenhum empregado com mais do que três filhos ganha menos do que 1000.

Complete o esquema relacional-objecto SQL3, por exemplo usando a notação apresentada nas aulas, considerando os requisitos enumerados para a aplicação referida, sem esquecer a primeira restrição de integridade (R1).

```
// Dominio para empregado Normal ou a Prazo
CREATE DOMAIN TipoDeEmp AS CHAR(1) DEFAULT 'N' CHECK (tipo IN ('N','P'));
// tipo para empregado
CREATE TYPE TipoEmp (
    nctb       CHAR(8) UNIQUE NOT NULL,
    nome       CHAR(52),
    anos       INTEGER,
    tipo       TipoDeEmp,
    filhos     LIST(REF(TipoFilho)),
```

```
    PUBLIC FUNCTION CalcSalario(anos: integer): REAL
);
// tipo para departamento
CREATE TYPE TipoDept (
    cod         INTEGER UNIQUE NOT NULL,
    nome        CHAR(52),
    emps        LIST(REF(TipoEmp))
);
// tipo para filho
CREATE TYPE TipoFilho (
    nome        CHAR(52),
    nasce       DATE
);
```

Resposta:

```
CREATE TABLE Empregados OF TipoEmp (
    nctb        PRIMARY KEY
    CHECK ( NOT ( tipo='P' AND anos>3))
);
CREATE TABLE Departamentos OF TipoDept (
    codigo      PRIMARY KEY
);
```

6. Módulos Persistentes em SQL3 [10 pontos]

Considere novamente a base de dados do problema 5.

Apresente o código SQL3 de um módulo persistente de servidor com a função e o procedimento seguintes:

```
maisUmAno(); // incrementa em um o número de anos dos empregados que têm
// mais do que dois filhos
//
quantosAPrazo(dep, sal): integer; // devolve o numero de empregados a prazo do
// departamento dep que ganham mais do que o salário sal
```

Resposta:

```
CREATE MODULE Emps
        LANGUAGE SQL;
    CREATE PROCEDURE maisUmAno()
    BEGIN
        UPDATE Empregados E
            SET anos = anos +1
            WHERE 2 < (SELECT count(*) FROM TABLE(E.filhos));
    END;
    CREATE FUNCTION quantosAPrazo(dep: INTEGER, sal: REAL): INTEGER
    BEGIN
        DECLARE c INTEGER;
        SELECT count(*) INTO c
        FROM Empregados E
        WHERE nctb IN (
            SELECT T.nctb
            FROM Departamentos D, TABLE (D.trabalhadores) T
            WHERE D.codigo = dep AND E.tipo = 'P' AND
                calcSalario(E.anos) > sal);
        RETURN c;
    END;
END MODULE;
```

7. Restrições de Integridade e Gatilhos [10 pontos]

Considere novamente a base de dados do problema 5.

a) Escreva uma asserção em SQL3 para impor a restrição R2.

Resposta:

```
CREATE ASSERTION a_r2
    AFTER
        INSERT ON Empregados,
        UPDATE OF anos ON Empregados,
        UPDATE OF tipo ON Empregados
    CHECK ( NOT EXISTS (
          SELECT *
          FROM Empregados E1
          WHERE E1.tipo= 'N' AND
              calcSalario(E1.anos) < ANY (
                  SELECT calcSalario(E2.anos)
                  FROM Empregados E2
                  WHERE E2.tipo = 'P')));
```

b) Escreva um ou mais gatilhos em SQL3 para impor a restrição R3 de forma incremental ao ser actualizada a informação sobre empregados.

Resposta:

```
CREATE TRIGGER g_r3
    AFTER
        INSERT ON Empregados,
        UPDATE OF anos ON Empregados,
        UPDATE OF filhos ON Empregados
    REFERENCING NEW AS n_t
    FOR EACH ROW
    WHEN ( calcSalario(n_t.anos) < 1 000 AND
        3 < ( SELECT count(*) FROM TABLE(n_t.filhos)))
    BEGIN
        ROLLBACK;
    END;
```

8. Estrutura Lógica de Documentos XML [10 pontos]

Considere o seguinte DTD para documentos XML:

```
<!DOCTYPE Empresa [
   <!ELEMENT EMP (DEPARTAMENTO*)>
   <!ELEMENT DEPARTAMENTO (LOCAL+, EMPREGADO*)>
   <!ATTLIST DEPARTAMENTO Cod ID #REQUIRED Nome CDATA>
   <!ELEMENT LOCAL EMPTY>
   <!ATTLIST LOCAL Cod ID #REQUIRED Nome CDATA>
   <!ELEMENT EMPREGADO (FILHO | FILHA)*>
   <!ATTLIST EMPREGADO Cod ID #REQUIRED Nome CDATA #REQUIRED Telefone CDATA
             Locais IDREFS #REQUIRED>
   <!ELEMENT FILHO (#PCDATA)>
   <!ELEMENT FILHA (#PCDATA)>
]>
```

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

```
<?XML VERSION="1.0" STANDALONE="no"?>
<!DOCTYPE Empresa SYSTEM "../DTDs/empresa.dtd">
<EMP>
   <DEPARTAMENTO Cod="D1" Nome="Vendas"> // 1
      <EMPREGADO Cod="E1" Nome="João Lopes" Telefone="123456789"> // A 2
      <EMPREGADO Cod="E2" Nome="Joaquim Lopes" Telefone="123456789"> // A 2
   </DEPARTAMENTO>
   <DEPARTAMENTO Cod="D2" Nome="Produção">
      <LOCAL Cod="L1" Nome="Porto"/>
      <EMPREGADO Cod="E1" Nome="João Lopes" Telefone="123456789" Local="L1"> // A 2 3 5
      <EMPREGADO Cod="E2" Nome="Joaquim Lopes"Telefone="123456789" Local="L1"> // A 2 3 5
      <LOCAL Cod="L2" Nome="Lisboa"> // B 4
      <EMPREGADO Cod="E3" Nome="João Lopes" Locais="L1 L2 E1">
           <FILHA>"Maria Lopes"</FILHA>
```

```
            <FILHO>"Pedro Lopes"</FILHO>
        </EMPREGADO>
    </DEPARTAMENTO>
    <DEPARTAMENTO Cod="D3">
        <LOCAL Cod="L3" Nome="Porto"></LOCAL>
    </DEPARTAMENTO>
</EMP>
```

Resposta:
O documento XML não é bem formado:

A/ <LOCAL> deve ser fechada por /> ou </LOCAL>

B/ <EMPREGADO> deve ser fechada por /> ou </EMPREGADO>

O documento XML viola o DTD nos seguintes pontos:

1/ <DEPARTAMENTO> tem de ter um <LOCAL>

2/ o atributo "Locais"é obrigatório

3/ repetição de ID

4/ <LOCAL> tem de estar antes de <EMPREGADO>

5/ o atributo é "Locais"e não "Local"

9. Transformação e apresentação de XML [10 pontos]

Considere novamente o DTD apresentado no problema 8 e instâncias de documentos XML de acordo com esse DTD.

Apresente um conjunto de regras de transformação XSLT que permitam passar para HTML, para ser mostrado num navegador Web, o nome de todos os departamentos e, para cada um, o nome dos empregados que têm filhos e o nome dos respectivos filhos.

Resposta:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="htpp://www.w3.org/XSL/TRANSFORM/1.0">
<xsl:template match="/">
    <HTML>
    <BODY>
    <xsl:for-each select="//DEPARTAMENTO">
        <xsl:value-of select="@Nome"/>
            <P>
            <xsl:if test="EMPREGADO/FILHO">
                <xsl:template match="EMPREGADO">
                    <xsl:value-of select="@Nome"/>
                    <xsl:for-each select="FILHO">
                        <xsl:value-of select="."/>
                    </xsl:for-each>
                </xsl:template>
                <BR>
            </xsl:for-each>
        </xsl:for-each>
    </xsl:for-each>
    </BODY>
    </HTML>
</xsl:template>
</xsl:stylesheet>
```

10. XML Schemas [10 pontos]

Para o DTD apresentado no problema 8 e considerando apenas os elementos `LOCAL` e `EMPREGADO`, apresente somente a parte correspondente de um XML Schema equivalente ao DTD (isto é, que permita as mesmas instâncias de documentos XML)

Resposta:

```
...
    <xsd:element name="LOCAL" minOccurs="1" maxOccurs="Unbound">
        <xsd:complexType>
            <xsd:attribute name="Cod" use="required" type="xsd:ID"/>
            <xsd:attribute name="Nome" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="EMPREGADO" minOccurs="0" maxOccurs="Unbound">
        <xsd:complexType>
            <xsd:choice minOccurs="0" maxOccurs="Unbound">
                <xsd:element name="FILHO" minOccurs="0" maxOccurs="Unbound"
                        type="xsd:string"/>
                <xsd:element name="FILHA" minOccurs="0" maxOccurs="Unbound"
                        type="xsd:string"/>
            <xsd:attribute name="Cod" use="required" type="xsd:ID"/>
            <xsd:attribute name="Nome" use="required" type="xsd:string"/>
            <xsd:attribute name="Telefone" type="xsd:string"/>
            <xsd:attribute name="Locais" use="required" type="xsd:IDREFS"/>
        </xsd:complexType>
    </xsd:element>
...
```

FIM.