TÉCNICO LISBOA

Bases de Dados 2015/2016

# Enunciado do Projeto – Parte 2

*Este enunciado corresponde à segunda parte do projeto da disciplina Bases de Dados que consiste na implementação de um bloco de notas estruturado sobre uma Base de Dados já existente.*

## 1 Criação das Tabelas

O Apêndice A apresenta um script para criação das tabelas do bloco de notas estruturado. Utilize este script para responder as perguntas da próxima seção. Considere que, para além do seu grupo, existem outros a desenvolver outras aplicações sobre as mesmas tabelas. Deverá portanto desenvolver as interrogações SQL para que a aplicação continue a funcionar independentemente de haver alterações às tabelas, tanto ao nível dos registos como ao nível da inserção de novos campos.

## 2 Trabalho a Desenvolver

O trabalho a desenvolver para a primeira parte do projeto consiste nos seguintes itens:

- **Consultas em SQL:** Escreva num ficheiro com um script em SQL as interrogações abaixo indicadas:

  (a) Quais são os utilizadores que falharam o login mais vezes do que tiveram sucesso?

  (b) Quais são os registos que aparecem em todas as páginas de um utilizador?

  (c) Quais os utilizadores que tem o maior número médio de registos por página?

  (d) Quais os utilizadores que, em todas as suas páginas, têm registos de todos os tipos de registos que criaram?

- **Restrições de integridade:** Defina a seguinte restrição de integridade, recorrendo aos mecanismos mais apropriados para o efeito, e que estejam disponíveis no sistema MySQL:

  (a) Todo o valor de *contador_sequencia* existente na relação *sequencia* existe numa e uma vez no universo das relações *tipo_registo, pagina, campo, registo e valor*.

- **Desenvolvimento da aplicação:** Crie um conjunto de páginas em PHP e HTML simples que permita ao utilizador:

  (a) Inserir uma nova página

(b) Inserir um novo tipo de registo

(c) Inserir novos campos para um tipo de registo

(d) Retirar uma página

(e) Retirar um tipo de registo

(f) Retirar um campo de um tipo de registo

(g) Inserir um registo e os respectivos valores dos campos associados

(h) Ver uma página com os registos nela contido

- **Formas Normais:** Considere a relação *utilizador*, na qual existem duas chaves candidatas (*userid* e *email*).

  (a) Em que forma normal se encontra a relação *utilizador*?

  (b) Considere que existe um trigger que garante que é sempre verdade que:

  nome, password, questao2, resposta2, questao1, resposta1 → email

  - Em que forma normal se encontra agora a relação *utilizador*?
  - Caso esta não se encontre na BCNF, proponha uma decomposição (sem perdas de informação) da mesma de forma a que todas as relações obtidas estejam na BCNF.

- **Índices:** Suponha que as seguintes interrogações são muito frequentes no sistema:

  (a) Devolver a média do número de registos por página de um utilizador,

  (b) Ver o nome dos registos associados à página de um utilizador,

  Indique que tipo de índice(s), sobre que atributo(s) e sobre que tabela(s) faria sentido criar de modo a acelerar a execução destas interrogações. Crie o(s) índice(s) em SQL e mostre como a execução de cada interrogação beneficiou da sua existência.

- **Transações:** Considere que há vários programas a aceder simultaneamente à base de dados. Considere ainda que cada registo e os valores dos seus campos têm que ser inseridos de forma atómica. Reescreva o seu código PHP de modo a garantir esta nova funcionalidade.

- **Data warehouse:**

  (a) Crie na base de dados o esquema de uma estrela com informação de número de tentativas de login tendo como dimensões: *d_utilizador(email, nome, país, categoria)* e *d_tempo(dia, mes, ano)*. Escreva as instruções SQL necessárias para carregar o esquema em estrela a partir das tabelas existentes.

(b) Considerando o esquema da estrela criado em (a), escreva a interrogação em MySQL para obter a média de tentativas de login para todos os utilizadores de Portugal, em cada categoria, com rollup por ano e mês.

# 3 Relatório

O projeto será avaliado a partir do relatório entregue pelos alunos. O relatório deverá conter todas as respostas aos itens da seção anterior. A Tabela 1 indica o que deve constar em cada seção do relatório e a respectiva cotação.

| **Seção** | **Cotação (valores)** |
|---|---|
| 1. Relatório | 0,5 |
| 2. Consultas SQL | 1,5 |
| 3. Restrições de integridade | 1,0 |
| 4. HTML/PHP | 3,0 |
| 5. Formas Normais | 1,0 |
| 6. Índices | 1,0 |
| 7. Transações | 1,0 |
| 8. Data warehouse | 1,0 |

Tabela 1: Conteúdo do relatório.

O relatório deverá começar com uma folha de rosto com a indicação "Projeto de Bases de Dados, Parte 2", o nome e número dos alunos, o número do grupo e o turno a que o grupo pertence.

O relatório deverá ser entregue em duas versões:

(a) Versão digital em formato PDF a entregar via Fénix até às 12:00 do dia 11 de dezembro de 2015. Não deixe para a última hora.

(b) Versão em papel a entregar na primeira aula de laboratório seguinte, ao docente dos laboratórios. Não encaderne, apenas agrafe as folhas.

# A Script de criação da Base de Dados

```
CREATE TABLE IF NOT EXISTS utilizador (
    userid INT NOT NULL  AUTO_INCREMENT,
    email VARCHAR(255) NOT NULL,
    nome VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    questao1 VARCHAR(255) NOT NULL,
    resposta1 VARCHAR(255) NOT NULL,
    questao2 VARCHAR(255),
    resposta2 VARCHAR(255),
    pais VARCHAR(45) NOT NULL,
    categoria VARCHAR(45) NOT NULL,
PRIMARY KEY (userid),
UNIQUE INDEX email_UNIQUE (email)
);

CREATE TABLE IF NOT EXISTS  login (
    contador_login INT NOT NULL AUTO_INCREMENT,
    userid INT NOT NULL,
    sucesso TINYINT(1) NOT NULL,
    moment TIMESTAMP NOT NULL,
PRIMARY KEY (contador_login),
FOREIGN KEY (userid) REFERENCES utilizador (userid) ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS sequencia (
    contador_sequencia INT NOT NULL AUTO_INCREMENT,
    moment TIMESTAMP NOT NULL,
    userid INT NOT NULL,
PRIMARY KEY (contador_sequencia),
FOREIGN KEY (userid) REFERENCES utilizador (userid) ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS  tipo_registo (
    userid INT NOT NULL,
    typecnt INT NOT NULL,
    nome MEDIUMTEXT NOT NULL,
    ativo TINYINT(1) NOT NULL,
```

```
    idseq INT NULL,
    ptypecnt INT,
PRIMARY KEY (userid, typecnt),
FOREIGN KEY (userid) REFERENCES utilizador (userid) ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (userid, ptypecnt) REFERENCES tipo_registo (userid, typecnt),
FOREIGN KEY (idseq) REFERENCES  sequencia (contador_sequencia)
);

CREATE TABLE IF NOT EXISTS registo (
    userid INT NOT NULL,
    typecounter INT NOT NULL,
    regcounter INT NOT NULL,
    nome VARCHAR(1024),
    ativo TINYINT(1),
    idseq INT NULL,
    pregcounter INT,
PRIMARY KEY (userid, regcounter, typecounter),
FOREIGN KEY (idseq) REFERENCES  sequencia (contador_sequencia),
FOREIGN KEY (userid , typecounter) REFERENCES tipo_registo (userid ,
typecnt),
FOREIGN KEY (userid, pregcounter, typecounter)
REFERENCES registo (userid, regcounter, typecounter)
);

CREATE TABLE IF NOT EXISTS pagina (
    userid INT NOT NULL,
    pagecounter INT NOT NULL,
    nome VARCHAR(1024) NOT NULL,
    idseq INT NOT NULL,
    ativa TINYINT(1) NOT NULL,
    ppagecounter INT NULL,
PRIMARY KEY (userid, pagecounter) ,
FOREIGN KEY (userid) REFERENCES utilizador (userid) ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (idseq) REFERENCES  sequencia (contador_sequencia),
FOREIGN KEY (userid, ppagecounter) REFERENCES pagina (userid, pagecounter)
);

CREATE TABLE IF NOT EXISTS campo (
    userid INT NOT NULL,
    typecnt INT NOT NULL,
    campocnt INT NOT NULL,
```

```
    idseq INT NOT NULL,
    ativo TINYINT(1) NOT NULL,
    nome VARCHAR(1024) NOT NULL,
    pcampocnt INT,
PRIMARY KEY (userid, typecnt, campocnt) ,
FOREIGN KEY (userid , typecnt) REFERENCES tipo_registo (userid , typecnt)
ON DELETE CASCADE  ON UPDATE CASCADE,
FOREIGN KEY (idseq) REFERENCES  sequencia (contador_sequencia),
FOREIGN KEY (userid, typecnt, pcampocnt)
REFERENCES  campo  (userid, typecnt, campocnt)
);

CREATE TABLE IF NOT EXISTS valor (
    userid INT NOT NULL,
    typeid INT NOT NULL,
    regid INT NOT NULL,
    campoid INT NOT NULL,
    valor LONGTEXT NULL,
    idseq INT NOT NULL,
    ativo TINYINT(1) NOT NULL,
    pcampoid INT,
PRIMARY KEY (userid, regid, typeid, campoid) ,
FOREIGN KEY (userid,typeid, campoid)
REFERENCES campo (userid, typecnt , campocnt)
ON DELETE CASCADE  ON UPDATE CASCADE,
FOREIGN KEY (userid, regid, typeid)
REFERENCES registo (userid , regcounter,typecounter)
ON DELETE CASCADE  ON UPDATE CASCADE,
FOREIGN KEY (idseq) REFERENCES  sequencia (contador_sequencia),
FOREIGN KEY (userid, regid, typeid, pcampoid)
REFERENCES valor (userid, regid, typeid, campoid)
 );

CREATE TABLE IF NOT EXISTS reg_pag(
    idregpag INT NOT NULL  AUTO_INCREMENT,
    userid INT NOT NULL,
    pageid INT NOT NULL,
    typeid INT NOT NULL,
    regid INT NOT NULL,
    idseq INT NOT NULL,
    ativa TINYINT(1) NOT NULL,
    pidregpag INT,
PRIMARY KEY (idregpag),
```

```
FOREIGN KEY (userid ,pageid) REFERENCES pagina (userid ,pagecounter)
ON DELETE CASCADE  ON UPDATE CASCADE,
FOREIGN KEY (userid , regid , typeid)
REFERENCES registo (userid , regcounter , typecounter)
ON DELETE CASCADE  ON UPDATE CASCADE,
FOREIGN KEY (idseq) REFERENCES  sequencia (contador_sequencia),
FOREIGN KEY (pidregpag) REFERENCES  reg_pag (idregpag)
 );
```

## A.1 Observações:

Neste modelo, o histórico de cada elemento (página, tipo de registo, registo, campo e valor) reside na mesma tabela que os próprios elementos, sendo diferenciado por ter o valor "idseq" de valor mais elevado. Para faciliar as queries, o modelo disponibiliza às aplicações a flag "ativo/ativa" que estas poderão eventualmente usar (e manter) para identificar as versões correntes dos referidos elementos. Além disso, cada registo na tabela sequencia é referido por apenas um registo entre todas as tabelas dos elementos.

A inserção de um novo elemento implica inserir um novo registo na tabela sequencia e outro na tabela correspondente ao elemento em causa. Este último poderá ter a flag ativa a true, facilitando as queries das leituras futuras.

A alteração de um elemento existente implica:

- Ler o registo existente desse elemento na tabela.
- Inserir um novo registo na tabela com o valor antigo e com a flag "ativo/ativa" a false, e
- Atualizar o nome/valor do registo original do elemento na tabela, mantendo a flag "ativo/ativa" e com um novo idseq.

Exemplo, considere que a tabela de registos tem o seguinte conteúdo:

| **userid** | **typecounter** | **regcounter** | **nome** | **ativo** | **idseq** | **pregcounter** |
|---|---|---|---|---|---|---|
| 10 | 11 | 1 | Registo1 | True | 100 | NULL |

A alteração do nome "Registo1" para "Registo2":

| **userid** | **typecounter** | **regcounter** | **nome** | **ativo** | **idseq** | **pregcounter** |
|---|---|---|---|---|---|---|
| 10 | 11 | 1 | Registo2 | True | 101 | 2 |
| 10 | 11 | 2 | Registo1 | False | 100 | NULL |

A inserção de um novo registo "NovoRegisto" do mesmo tipo:

| **userid** | **typecounter** | **regcounter** | **nome** | **ativo** | **idseq** | **pregcounter** |
|---|---|---|---|---|---|---|
| 10 | 11 | 1 | Registo2 | True | 101 | 2 |
| 10 | 11 | 2 | Registo1 | False | 100 | NULL |
| 10 | 11 | 3 | NovoRegisto | True | 102 | NULL |

A alteração do nome "Registo2" para "Registo3":

| **userid** | **typecounter** | **regcounter** | **nome** | **ativo** | **idseq** | **pregcounter** |
|---|---|---|---|---|---|---|
| 10 | 11 | 1 | Registo3 | True | 103 | 4 |
| 10 | 11 | 2 | Registo1 | False | 100 | NULL |
| 10 | 11 | 3 | NovoRegisto | True | 102 | NULL |
| 10 | 11 | 4 | Registo2 | False | 101 | 2 |