Projeto de Bases de Dados

LEIC-A 2017/2018

Grupo 30

Parte 3

Relatório da Aplicação Web

Turno BD2251795L09 - Sexta-Feira 12:30
Professor Paulo Carreira

83467	Francisco Neves	24 horas
83468	Francisco Catarrinho	24 horas
83550	Pedro Santos	24 horas

1 Criação da base de dados

De seguida é apresentado o conjunto de instruções necessário para criar o esquema da base de dados no SGBD **PostgreSQL**. As instruções aqui presentes encontram-se no ficheiro **schema.sql**.
Para a definição das variáveis e respetivos tamanhos foram consultados os standards internacionais.

Rebuild schema, dropping existing tables.

```
DROP TABLE IF EXISTS reposicao;
DROP TABLE IF EXISTS evento_reposicao;
DROP TABLE IF EXISTS planograma;
DROP TABLE IF EXISTS prateleira;
DROP TABLE IF EXISTS corredor;
DROP TABLE IF EXISTS fornece_sec;
DROP TABLE IF EXISTS produto;
DROP TABLE IF EXISTS fornecedor;
DROP TABLE IF EXISTS constituida;
DROP TABLE IF EXISTS super_categoria;
DROP TABLE IF EXISTS categoria_simples;
DROP TABLE IF EXISTS categoria;
```

Create database schema.

Table structure for 'categoria'.

```
CREATE TABLE categoria (
    nome char(70) NOT NULL,
    PRIMARY KEY (nome)
);
```

Table structure for 'categoria_simples'.

```
CREATE TABLE categoria_simples (
    nome char(70) NOT NULL,
    PRIMARY KEY (nome),
    FOREIGN KEY (nome) REFERENCES categoria(nome) ON DELETE CASCADE
);
```

Table structure for 'super_categoria'.

```
CREATE TABLE super_categoria (
    nome char(70) NOT NULL,
    PRIMARY KEY (nome),
    FOREIGN KEY (nome) REFERENCES categoria(nome) ON DELETE CASCADE
);
```

Table structure for 'constituida'.

```
CREATE TABLE constituida (
    super_categoria char(70)   NOT NULL,
    categoria       char(70)   NOT NULL,
    PRIMARY KEY (super_categoria, categoria),
    FOREIGN KEY (super_categoria) REFERENCES super_categoria(nome) ON DELETE CASCADE,
    FOREIGN KEY (categoria) REFERENCES categoria(nome) ON DELETE CASCADE,
    CONSTRAINT supercategoria_dif_categoria CHECK (super_categoria != categoria)
);
```

Table structure for 'fornecedor'.

```
CREATE TABLE fornecedor (
    nif   char(9)  NOT NULL,
    nome  char(70) NOT NULL,
    PRIMARY KEY (nif),
    CONSTRAINT is_nif CHECK (check_nif(nif))
);
```

Table structure for 'produto'.

```
CREATE TABLE produto (
    ean            char(13)  NOT NULL,
    design         char(400) NOT NULL,
    categoria      char(70)  NOT NULL,
    forn_primario  char(70)  NOT NULL,
    data           date      NOT NULL,
    PRIMARY KEY (ean),
    FOREIGN KEY (categoria) REFERENCES categoria(nome),
    FOREIGN KEY (forn_primario) REFERENCES fornecedor(nif),
    CONSTRAINT is_ean CHECK (check_ean(ean))
);
```

Table structure for 'fornece_sec'.

```
CREATE TABLE fornece_sec (
    nif char(9)  NOT NULL,
    ean char(13) NOT NULL,
    PRIMARY KEY (nif, ean),
    FOREIGN KEY (nif) REFERENCES fornecedor(nif),
    FOREIGN KEY (ean) REFERENCES produto(ean) ON DELETE CASCADE,
    CONSTRAINT not_in_forn_prim CHECK (NOT forn_prim_sec(ean, nif)),
    CONSTRAINT is_nif CHECK (check_nif(nif)),
    CONSTRAINT is_ean CHECK (check_ean(ean))
);
```

Table structure for 'corredor'.

```
CREATE TABLE corredor (
    nro     integer NOT NULL,
    largura real    NOT NULL,
    PRIMARY KEY (nro)
);
```

Table structure for 'prateleira'.

```
CREATE TABLE prateleira (
    nro    integer  NOT NULL,
    lado   char(20) NOT NULL,
    altura char(20) NOT NULL,
    PRIMARY KEY (nro, lado, altura),
    FOREIGN KEY (nro) REFERENCES corredor(nro)
);
```

Table structure for 'planograma'.

```
CREATE TABLE planograma (
    ean      char(13) NOT NULL,
    nro      integer  NOT NULL,
    lado     char(20) NOT NULL,
    altura   char(20) NOT NULL,
    face     integer  NOT NULL,
    unidades integer  NOT NULL,
    loc      integer  NOT NULL,
    PRIMARY KEY (ean, nro, lado, altura),
    FOREIGN KEY (ean) REFERENCES produto(ean),
    FOREIGN KEY (nro, lado, altura) REFERENCES prateleira(nro, lado, altura),
    CONSTRAINT is_ean CHECK (check_ean(ean))
);
```

Table structure for 'evento_reposicao'.

```
CREATE TABLE evento_reposicao (
    operador char(70)  NOT NULL,
    instante timestamp NOT NULL CHECK (instante <= localtimestamp),
    PRIMARY KEY (operador, instante)
);
```

Table structure for 'reposicao'.

```
CREATE TABLE reposicao (
    ean       char(13)   NOT NULL,
    nro       integer    NOT NULL,
    lado      char(20)   NOT NULL,
    altura    char(20)   NOT NULL,
    operador  char(70)   NOT NULL,
    instante  timestamp  NOT NULL,
    unidades  integer    NOT NULL,
    PRIMARY KEY (ean, nro, lado, altura, operador, instante),
    FOREIGN KEY (ean, nro, lado, altura) REFERENCES planograma(ean, nro, lado, altura),
    FOREIGN KEY (operador, instante)     REFERENCES evento_reposicao(operador, instante),
    CONSTRAINT is_ean CHECK (check_ean(ean))
);
```

2 SQL

a) Qual o nome do fornecedor que forneceu o maior número de categorias? Note que pode ser mais do que um fornecedor.

```
WITH counts AS (
    SELECT COUNT(categoria), nif FROM (
        SELECT nif, ean, categoria FROM (
            SELECT forn_primario AS nif, ean FROM produto
            UNION
            SELECT nif, ean FROM fornece_sec
        ) AS f
        JOIN produto USING (ean)
    ) AS c
    GROUP BY nif
)

SELECT nome FROM fornecedor
JOIN counts USING (nif)
JOIN (
    SELECT MAX(count) AS count FROM counts
) AS max USING (count);
```

b) Quais os fornecedores primários (nome e nif) que forneceram produtos de todas as categorias simples?

```
SELECT DISTINCT nome, nif FROM fornecedor JOIN (
    SELECT forn_primario AS nif
    FROM produto AS prod
    WHERE NOT EXISTS (
        SELECT nome AS categoria FROM categoria_simples
        EXCEPT
        SELECT categoria FROM (
            produto JOIN categoria_simples
            ON produto.categoria = categoria_simples.nome
        ) AS p
        WHERE forn_primario = prod.forn_primario
    )
) AS f USING (nif);
```

c) Quais os produtos (ean) que nunca foram repostos?

```
SELECT ean FROM (
    SELECT ean FROM produto
    EXCEPT
    SELECT DISTINCT ean FROM reposicao
) AS eans;
```

d) Quais os produtos (ean) com um número de fornecedores secundários superior a 10?

```
SELECT ean FROM (
    SELECT COUNT(ean), ean FROM fornece_sec
    GROUP BY ean
) AS counts
WHERE count > 10;
```

e) Quais os produtos (ean) que foram repostos sempre pelo mesmo operador?

```
SELECT ean FROM (
    SELECT COUNT(ean), ean FROM (
        SELECT DISTINCT ean, operador FROM reposicao
    ) AS reposicoes GROUP BY ean
) AS counts
WHERE count = 1;
```

3 Desenvolvimento da Aplicação

Para a interação com o utilizador foram criadas várias páginas php com diversos formulários. Todas as operações diretamente relacionadas com base de dados foram encapsuladas numa classe **SupermarketService.class.php**. Utilizaram-se prepared statements em todas as queries que possuem input do utilizador e foi ainda criada uma classe **SupermarketException.class.php** responsável por tratar eventuais exceções da base de dados e apresentar mensagens informativas ao utilizador, abstraindo-o de detalhes da arquitetura da aplicação.

Exemplo de interação entre página php e classe SupermarketService:

```
<?php
try {
    $db = new SupermarketService();
    $products = $db->getProducts();
    // Close connection
    $db->close();
    $db = null;

    // Display result in a table
?>
<table class="table table-bordered">
    <thead>
        ...
    </thead>
    <tbody>
        <?php foreach($products as $product) { ?>
        <tr>
            <td><?php echo($product['ean']) ?></td>
            <td><?php echo($product['categoria']) ?></td>
            <td><?php echo($product['design']) ?></td>
            <td><?php echo($product['forn_primario']) ?></td>
            <td><?php echo(date("d-m-Y", strtotime($product['data']))) ?></td>
        </tr>
        <?php } ?>
    </tbody>
</table>

<?php
}
catch (PDOException $e) {
    error(SupermarketException::digest($e));
} ?>
```

Classe SupermarketService:

```
class SupermarketService {

        private $db;

        // Constructor.
        public function __construct() {

            $this->db = new PDO("pgsql:host=$dbhost;dbname=$dbname;", $dbuser, $dbpass);
            $this->db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        }

        // Closes PDO connection. Assigns PDO instance pointer a null reference.
        public function close() {
            $this->db = null;
        }

        // Prepares and executes queries.
        private function prepareAndExecuteQuery($query, $params, $mode = false) {
           $params = $this->escapeHTML($params);
            $statement = $this->db->prepare($query);
            $statement->execute($params);
            if ($mode) {
                return $statement->fetchAll(\PDO::FETCH_ASSOC);
            }
        }

        // Executes transaction on given queries.
        private function makeTransaction($queries) {

            try {
                $this->db->beginTransaction();

                foreach($queries as $query => $params) {
                    if (is_array($params[0])) {
                        foreach($params as $param) {
                            $this->prepareAndExecuteQuery($query, $param);
                        }
                    }
                    else {
                        $this->prepareAndExecuteQuery($query, $params);
                    }
                }
                $this->db->commit();

            } catch (PDOException $e) {
                $this->db->rollBack();
                throw $e;
            }
        }
```

a) Inserir e remover categorias e sub-categorias:

```
        public function insertCategory($category) {

            $queries = Array(
                    "INSERT INTO categoria (nome) VALUES (:categoria)"
                    =>
                    Array(":categoria" => $category),
                    "INSERT INTO categoria_simples (nome) VALUES (:categoria)"
                    =>
                    Array(":categoria" => $category)
                );

            $this->makeTransaction($queries);
        }
```

b) Inserir e remover um novo produto:

```
public function insertProduct($ean, $category, $design, $prim_provider, $sec_providers,
    $date) {
    $queries = Array(

                "INSERT INTO produto (ean, categoria, design, forn_primario, data) VALUES
                    (:ean, :categoria, :design, :forn_primario, :data)"
                =>
                Array(":ean" => $ean, ":categoria" => $category, ":design" => $design,
                    ":forn_primario" => $prim_provider, ":data" => $date)

        );

    $sec_providers_params = Array();
    foreach ($sec_providers as $sec_provider) {
        $sec_providers_params[] = Array(":nif" => $sec_provider, ":ean" => $ean);
    }

    $queries["INSERT INTO fornece_sec (nif, ean) VALUES (:nif, :ean)"] =
        $sec_providers_params;

    $this->makeTransaction($queries);
}
```

e) Listar todas as sub-categorias de uma super-categoria, a todos os níveis de profundidade:

```
public function listSubCategories($category) {
    $query = ' WITH RECURSIVE cte (super_categoria, categoria) AS
                (
                    SELECT    super_categoria, categoria
                    FROM      constituida
                    WHERE     super_categoria = :category
                    UNION ALL
                    SELECT    c.super_categoria, c.categoria
                    FROM      constituida AS c
                    INNER JOIN cte
                    ON c.super_categoria = cte.categoria
                )
                SELECT categoria FROM cte ORDER BY categoria;';
    return $this->prepareAndExecuteQuery($query, Array(":category" => $category), true);
}
```

Arquitetura da aplicação e interação com o utilizador:

A nossa aplicação possui um menu principal (ficheiro **index.php**) onde é possível selecionar a ação a realizar. Depois de escolhida a ação, o utilizador é redirecionado para outra página php que contém os formulários a serem preenchidos pelo utilizador. É possível navegar entre as páginas php com a ajuda de separadores. Nos formulários foram usados campos de preenchimento textual, menus dropdown e botões de modo a facilitar ao máximo a tarefa do utilizador. Em cada página são apresentadas informações da base de dados relativas à ação.

Figure 1: Menu principal.

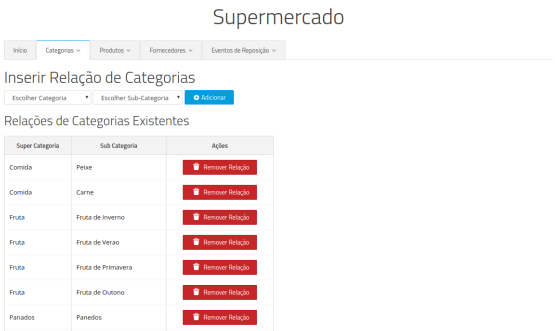

Figure 2: Formulário.