Projeto de Bases de Dados

Parte 3

Grupo 19 – Turno BD225179L09

Prof. Gabriel Pestana

81082 – Nuno Gonçalves (25 horas)

81205 – Alice Dourado (25 horas)

81500 – Rodrigo Rato (25 horas)

1. Queries SQL

SQL – Query #1 (espaços c/postos nunca alugados)

```
SELECT e.morada, e.codigo
FROM espaco e, posto p
WHERE e.morada = p.morada AND
          e.codigo = p.codigo_espaco AND
    (p.morada, p.codigo)  NOT IN (SELECT morada, codigo FROM aluga)
GROUP BY e.morada, e.codigo;
```

SQL – Query #2 (edificios com #reservas > avg(#reservas))

```
SELECT morada
FROM aluga a
GROUP BY morada
HAVING count(*) >= (SELECT avg(r.num_reservas) as avg_num_reservas
                    FROM (SELECT COUNT(*) as num_reservas
                          FROM aluga a
                          GROUP BY morada) as r);
```

SQL – Query #3 (users c/alugáveis fiscalizados por um só fiscal)

```
SELECT DISTINCT u.*
FROM fiscaliza f, arrenda a, user u
WHERE f.codigo = a.codigo
    AND a.morada = f.morada
    AND u.nif = a.nif
GROUP BY u.nif
HAVING COUNT(DISTINCT f.id) = 1;
```

SQL – Query #4 (montante pago p/espaço em 2016 (total ou p/postos))

```
SELECT DISTINCT morada, codigo,
        SUM(DATEDIFF(data_fim, data_inicio)*tarifa) as montante_pago_2016
FROM paga NATURAL JOIN aluga NATURAL JOIN espaco NATURAL JOIN oferta
NATURAL JOIN  estado
WHERE estado = 'Aceite' AND
       YEAR(time_stamp) = 2016
GROUP BY morada, codigo
UNION
SELECT DISTINCT morada, codigo_espaco,
        SUM(DATEDIFF(data_fim, data_inicio)*tarifa) as montante_pago_2016
FROM paga NATURAL JOIN aluga NATURAL JOIN posto NATURAL JOIN oferta NATURAL
JOIN  estado
WHERE estado = 'Aceite' AND
        YEAR(time_stamp) = 2016
GROUP BY morada, codigo_espaco;
```

SQL – Query #5 (espaços com postos todos alugados)

```
SELECT morada, codigo_espaco as codigo
FROM posto p NATURAL JOIN oferta o NATURAL JOIN estado e NATURAL JOIN aluga a
WHERE estado = 'Aceite'
GROUP BY codigo_espaco
HAVING COUNT(codigo) = (SELECT count(p2.codigo)
                        FROM posto p2
                        WHERE p2.codigo_espaco = p.codigo_espaco AND
                        p2.morada = p.morada
                        GROUP BY p2.codigo_espaco);
```

2. Triggers (restrições de integridade)

SQL – Trigger 1 (primeira restrição de integridade)

```
DROP TRIGGER IF EXISTS overlappedDates;
DELIMITER //
CREATE TRIGGER overlappedDates BEFORE INSERT ON oferta
   FOR EACH ROW
   BEGIN
      IF EXISTS(SELECT *
            FROM oferta
            WHERE morada = new.morada AND
                codigo = new.codigo AND
                ((new.data_inicio BETWEEN data_inicio AND data_fim) OR
                 (new.data_fim BETWEEN data_inicio AND data_fim))) THEN

         CALL ERR_OVERLAPPEDDATES_TRIGGER;
      END IF;
   END //
DELIMITER ;
```

SQL – Trigger 2.1 (segunda restrição de integridade)

```
DROP TRIGGER IF EXISTS estadoPaga1;
DELIMITER //
CREATE TRIGGER estadoPaga1 BEFORE INSERT ON paga
   FOR EACH ROW
   BEGIN
     SET @estado_maxtimestamp = (SELECT MAX(time_stamp)
                                 FROM estado
                                 WHERE numero = new.numero);

      IF @estado_maxtimestamp > new.data THEN
         CALL ERR_ESTADOPAGA_TRIGGER;

      END IF;
   END //
DELIMITER ;
```

SQL – Trigger 2.2 (segunda restrição de integridade)

```
DROP TRIGGER IF EXISTS estadoPaga2;
DELIMITER //
CREATE TRIGGER estadoPaga2 BEFORE INSERT ON estado
   FOR EACH ROW
   BEGIN
    SET @paga_data = (SELECT data
                      FROM paga
                      WHERE numero = new.numero);

    IF ((!(@paga_data IS NULL)) AND (@paga_data < new.time_stamp)) THEN
       CALL ERR_ESTADOPAGA_TRIGGER;
    END IF;
   END //
DELIMITER ;
```

3. PHP

Uma versão desta aplicação (com o schema do ficheiro **schema.sql** em anexo e populada com o **populate.sql** em anexo) encontra-se online para efeitos de teste em: http://web.ist.utl.pt/~rodrigorato/bd/bd-instantoffice/p3/web/.

PHP – setup.php: assim obtemos o Data Object (PDO)

```
<?php
  function getPDO(){
      $host = "db.ist.utl.pt";
      $user ="istxxxxxx"; // Username e password omitidos neste relatório
      $password = "xxxxxxx";
      $dbname = $user;

      $db = new PDO("mysql:host=$host;dbname=$dbname", $user, $password);
      $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
       return $db;
  }
?>
```

PHP – edificio/gerir_edificio.php: exemplo de leitura da base de dados

```
<?php
   try{
      include "../setup.php";
      $db = getPDO();

      $sql = "SELECT * FROM edificio;";

      $result = $db->query($sql);

      echo "<div class=\"menu\"> <br>";
      echo("<table border=\"0\" cellspacing=\"5\">\n");
      echo "<th>Morada</th>";
      foreach($result as $row){
         echo("<tr>\n");
         echo("<td>{$row['morada']}</td>\n");
         echo("<td><a href=\"remover_edificio.php?morada=");
         echo("{$row['morada']}\">Remover Edifício</a></td>\n");
         echo("<td><a href=\"verificar_total_realizado_espaco.php?morada=");
         echo("{$row['morada']}\">Verificar Total Realizado</a></td>\n");
         echo("</tr>\n");
      }
      echo("</table>\n");
      echo "</div>";

      $db = null;
   }
   catch (PDOException $e)
   {
      echo("<p>ERROR: {$e->getMessage()}</p>");
   }
?>
```

PHP – edificio/inserir_edificio.php: exemplo de escrita na base de dados

```
<?php
   $morada = $_REQUEST['morada'];
   try
   {
      include "../setup.php";
      $db = getPDO();
      $db->query("start transaction;");
      $sql = "INSERT INTO edificio(`morada`)VALUES(:morada);";
      $stmt = $db->prepare($sql);
      $stmt->bindParam(':morada',$morada,PDO::PARAM_STR);
      $stmt->execute();
      $db->query("commit;");
      echo("<p>Edifício inserido com sucesso.</p>");
      $db = null;
   }
   catch (PDOException $e)
   {
      $db->query("rollback;");
      echo("<p>ERROR: {$e->getMessage()}</p>");
   }
?>
```

PHP – edificio/remover_edificio.php: exemplo de remoção na base de dados

```
<?php
   $morada = $_REQUEST['morada'];
   try
   {
      include "../setup.php";
      $db = getPDO();
      $db->query("start transaction;");
      $sql = "DELETE FROM edificio WHERE edificio.morada = :morada;";
      $stmt = $db->prepare($sql);
      $stmt->bindParam(':morada',$morada,PDO::PARAM_STR);
      $stmt->execute();
      $db->query("commit;");
      echo("<p>Edifício removido com sucesso.</p>");
      $db = null;
   }
   catch (PDOException $e)
   {
      $db->query("rollback;");
      echo("<p>ERROR: {$e->getMessage()}</p>");
   }
?>
```