Projecto de Bases de Dados, Parte 3

Professor Gabriel Pestana

Turno: Quinta-feira, 11:00-12:30

Grupo 45

Inês Sequeira, nº 81719

Pedro Gomes, nº 81534

Rafael Belchior, nº 80970

Esforço (em horas)

81719 ------- 12 horas

81534 ------- 12 horas

80970 ------- 12 horas

Não alterámos o ficheiro schema.sql disponibilizado pelos professores.
Trocámos a ordem da inserção nas tabelas paga e estado, pois se criarmos o esquema da base de dados e acrescentarmos os triggers antes de usar o populate.sql disponibilizado, não vai ser possível inserir registos na tabela paga, pois esta necessita que exista um registo na tabela estado para a reserva em causa.

Queries:

```
-- query a)

SELECT DISTINCT esp.codigo, esp.morada

FROM espaco esp JOIN posto p1 ON esp.codigo = p1.codigo_espaco AND esp.morada = p1.morada

WHERE NOT EXISTS(

        SELECT DISTINCT p2.codigo, p2.morada

        FROM posto p2 NATURAL JOIN aluga a

                NATURAL JOIN estado e

        WHERE p2.codigo = p1.codigo

                AND e.estado = 'Aceite');

-- query b)
SELECT morada
FROM aluga a1
GROUP BY  a1.morada
HAVING COUNT(*) > (
        SELECT AVG(ac.reservas)
        FROM (SELECT a2.morada, COUNT(*) AS reservas
                FROM aluga a2
                GROUP BY  a2.morada) ac);
-- query c)
SELECT DISTINCT a.nif
FROM arrenda a NATURAL JOIN fiscaliza f
GROUP BY  a.nif
HAVING COUNT(DISTINCT f.id) <= 1;
-- query d)
-- juntamos as 3 tabelas descritas nos comentarios e somamos todos os valores realizados por espaco, para ter o valor realizado total por espaco
SELECT res.morada, res.codigo, SUM(res.TotalEspaco) AS Total
FROM -- vemos os espacos com reserva paga em 2016 e calculamos o valor realizado para esse espaco
        (SELECT e.morada, e.codigo, SUM(oe.tarifa * datediff(oe.data_fim, oe.data_inicio)) AS TotalEspaco
        FROM espaco e NATURAL JOIN oferta oe
                NATURAL JOIN aluga ae
                NATURAL JOIN paga pge
        WHERE pge.data BETWEEN '2016-01-01' AND '2016-12-31'
        GROUP BY  e.morada, e.codigo
        UNION
        -- vemos os postos por espaco com reserva paga em 2016 e calculamos o valor realizado para esses postos agrupado por espaco
        SELECT p.morada, p.codigo_espaco, SUM(o.tarifa * datediff(o.data_fim, o.data_inicio)) AS TotalPosto
        FROM posto p NATURAL JOIN oferta o
                NATURAL JOIN aluga a
                NATURAL JOIN paga pg
        WHERE pg.data BETWEEN '2016-01-01' AND '2016-12-31'
        GROUP BY  p.morada, p.codigo_espaco
        UNION
```

```
	-- vemos os espacos que nao tem reserva paga em 2016 e pomos o valor realizado para esse espaco a
zero
	SELECT ez.morada, ez.codigo AS codigo_espaco, 0
	FROM espaco ez
	WHERE (ez.morada, ez.codigo) NOT IN(
		SELECT az.morada, az.codigo
		FROM aluga az NATURAL JOIN paga pgz
		WHERE pgz.data BETWEEN '2016-01-01' AND '2016-12-31')
	) res
GROUP BY  res.morada, res.codigo;
-- query e)
SELECT DISTINCT esp.codigo, esp.morada
FROM espaco esp JOIN posto p1 ON esp.codigo = p1.codigo_espaco AND esp.morada = p1.morada
WHERE NOT EXISTS(
	SELECT p.codigo, p.morada
	FROM posto p
	WHERE p.codigo_espaco = esp.codigo
		AND NOT EXISTS(
			SELECT a.codigo, a.morada
			FROM aluga a NATURAL JOIN estado e
			WHERE p.codigo = a.codigo
				AND e.estado = 'Aceite'));
```

Restrições de Intergridade:

```
DROP TRIGGER IF EXISTS data_check;
DROP TRIGGER IF EXISTS pagamento_check;
-- -- -- -- -- -- -- -- -- -- --  Restricao de Integridade 1: -- -- -- -- -- -- -- -- --
-- verificamos se nao ha sobreposicao de datas de ofertas entre o alugavel e ele proprio
-- e entre ele e os seus postos, se for um espaco, e entre ele e o espaco onde esta contido, se for um posto
DELIMITER	//
CREATE TRIGGER data_check
BEFORE INSERT ON oferta
FOR EACH ROW
BEGIN
	IF EXISTS (SELECT ofertaPostoEspaco.morada, ofertaPostoEspaco.codigo
		FROM (SELECT op.morada, op.codigo, pp.codigo_espaco, op.data_inicio, op.data_fim
		FROM oferta op NATURAL JOIN posto pp
		UNION
		SELECT oe.morada, pe.codigo, oe.codigo, oe.data_inicio, oe.data_fim
		FROM oferta oe LEFT JOIN posto pe ON oe.codigo = pe.codigo_espaco AND oe.morada =
pe.morada) ofertaPostoEspaco
		WHERE NEW.morada = ofertaPostoEspaco.morada
			AND (NEW.codigo = ofertaPostoEspaco.codigo
				OR NEW.codigo = ofertaPostoEspaco.codigo_espaco)
			AND ((NEW.data_inicio BETWEEN ofertaPostoEspaco.data_inicio AND
ofertaPostoEspaco.data_fim)
				OR (NEW.data_fim BETWEEN ofertaPostoEspaco.data_inicio AND
ofertaPostoEspaco.data_fim)
				OR (ofertaPostoEspaco.data_inicio BETWEEN NEW.data_inicio AND
NEW.data_fim)
				OR (ofertaPostoEspaco.data_fim BETWEEN NEW.data_inicio AND
NEW.data_fim))
				) THEN
		CALL oferta_datas_sobrepostas;
	END IF;
END //
DELIMITER ;
-- -- -- -- -- -- -- -- -- -- --  Restricao de Integridade 2: -- -- -- -- -- -- -- -- --
```

```
DELIMITER          //
CREATE TRIGGER pagamento_check
BEFORE INSERT ON paga
FOR EACH ROW
BEGIN
        SET @estado_mais_recente = (SELECT MAX(e1.time_stamp)
                                    FROM estado e1
                                    WHERE e1.numero = NEW.numero);
        SET @nova_data_pag = NEW.data;
        IF (@nova_data_pag < @estado_mais_recente) THEN
                CALL paga_data_maior_timestamp_estado;
        END IF;
END //
DELIMITER ;
```

PHP:

Criámos uma sessão com o nif do utilizador. O nif é usado na inserção de um espaço/posto na tabela arrenda (insereEdfEspPosto.php) e na inserção de uma reserva na tabela aluga (inserirReserva.php).
Para fazer as queries à base de dados usámos prepared statements quando há input do utilizador, de modo a evitar que haja sql injection.
Quando precisamos de remover registos em mais do que uma tabela e é necessário que se remova em todas as tabelas, de forma a que os dados estejam consistentes, usamos uma transação para que se execute de forma atómica. Ou seja, remove os registos de todas tabelas, apenas após o commit, ou não remove nenhum, se houver erro é lançada uma excepção é feito rollback. (O mesmo se aplica a inserir.)
Quando se tenta remover um registo de uma tabela em que esse registo é referenciado por um conjunto de atributos de outra tabela, nós decidimos não apagar esse registo. Se apagassemos o registo em causa perderiamos informação definitivamente. Por exemplo, se um utilizador tenta remover um edificio, e se nós tivessemos decidido elimina-lo, teriamos de remover todos os seus dependentes e perderiamos toda a informação sobre os espaços e postos nele contido e, consequentemente perderiamos as ofertas e reservas. Assim, para além de perder o histórico, perderiamos também informação actual.

```
//Connectar com a base de dados:
$host = "db.ist.utl.pt";
$user ="ist******";
$password = "*****";
$dbname = $user;
$db = new PDO("mysql:host=$host;dbname=$dbname", $user, $password);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
//codigo
$db = null;

function inserePosto($db,$morada,$codigo,$codigoEsp,$nif)  {
   try{
     $db->beginTransaction();
     $values = array($morada,$codigo, 'http://lorempixel.com/400/200/');
     $sql = "INSERT INTO alugavel(morada, codigo, foto) VALUES(?, ?, ?)";
     $stmt = $db->prepare($sql);
     $affected_rows  = $stmt->execute($values);

     $values = array($morada,$codigo,$codigoEsp);
     $sql = "INSERT INTO posto(morada, codigo, codigo_espaco) VALUES(?, ?, ?)";
     $stmt = $db->prepare($sql);
     $affected_rows  = $stmt->execute($values);

     $values = array($morada,$codigo,$nif);
     $sql = "INSERT INTO arrenda(morada, codigo,nif) VALUES (?, ?, ?)";
     $stmt = $db->prepare($sql);
```

```
    $affected_rows  = $stmt->execute($values);
    $db->commit();
   }
   catch (PDOException $e)  {
    $db->rollBack();
    echo("<p>Erro a inserir posto: {$e->getMessage()}</p>");
   }
 }
function removePosto($db,$morada,$codigo,$codigoEsp)  {
   try{
      $db->beginTransaction();
      $values = array($morada,$codigo);
      $sql = "DELETE FROM arrenda WHERE morada = ? AND codigo = ?";
      $stmt = $db->prepare($sql);
      $affected_rows  = $stmt->execute($values);

      $valuesPosto = array($morada,$codigo,$codigoEsp);
      $sql = "DELETE FROM posto WHERE morada = ? AND codigo = ? AND codigo_espaco = ?";
      $stmt = $db->prepare($sql);
      $affected_rows  = $stmt->execute($valuesPosto);

      $valuesAlugavel = array($morada,$codigo);
      $sql = "DELETE FROM alugavel WHERE morada = ? AND codigo = ?";
      $stmt = $db->prepare($sql);
      $affected_rows  = $stmt->execute($valuesAlugavel);
      $db->commit();
   }
   catch (PDOException $e)  {
    $db->rollBack();
    echo("<p>Erro a remover posto: {$e->getMessage()}</p>");    }
 }

function insertOferta ($db,$morada,$codigo,$dataInicio,$dataFim,$tarifa)  {
   try{
      $valores= array($morada,$codigo,$dataInicio,$dataFim,$tarifa);
      $sql = "INSERT INTO oferta(morada, codigo, data_inicio, data_fim, tarifa) VALUES(?,?,?,?,?)" ;
      $stmt = $db->prepare($sql);
      $affected_rows  = $stmt->execute($valores);
   }
   catch (Exception $e)  {
    echo("<p>Nao e possivel inserir o Oferta.</p>");
   }
}
function removeOferta ($db,$morada,$codigo,$dataInicio,$dataFim,$tarifa)  {
   try{
    $valores= array($morada,$codigo,$dataInicio,$dataFim,$tarifa);
    $sql = "DELETE FROM  oferta WHERE morada = ? AND codigo = ? AND data_inicio = ? AND data_fim = ? AND
tarifa = ?";
     $stmt = $db->prepare($sql);
     $affected_rows  = $stmt->execute($valores);
   }
   catch (Exception $e)  {
    echo("<p>Nao e possivel remover o Oferta.</p>");
   }
}

function insertReserva ($db,$morada,$codigo,$dataInicio,$nif)  {
```

```
    try{
        $currentYear = date(Y);
        $numFormat = $currentYear."-%";
        $sql = "SELECT r.numero
                FROM reserva r
                WHERE r.numero LIKE '".$numFormat."'";
        $rows  = $db->query($sql);
        $maximo = 0;
        foreach ($rows as $valor) {
            $aux = str_split($valor['numero'],5);
            $aux2 = $aux[1];
            if ($aux2 >= $maximo) {
                $maximo = $aux2;
            }
        }
        $maximo = $maximo+1;
        $numero = $currentYear."-".$maximo;
        $valoresReserva= array($numero);

        $db->beginTransaction();
        $sql = "INSERT INTO reserva(numero) values(?)";
        $stmt = $db->prepare($sql);
        $affected_rows  = $stmt->execute($valoresReserva);

        $valoresAluga= array($morada,$codigo,$dataInicio,$nif, $numero);
        $sql = "INSERT INTO aluga(morada, codigo, data_inicio, nif, numero) values(?,?,?,?,?)";
        $stmt = $db->prepare($sql);
        $affected_rows  = $stmt->execute($valoresAluga);

        $time_stamp = date("Y-m-d H:i:s");
        $valores = array($numero,$time_stamp,'Pendente');
        $sql = "INSERT INTO estado(numero, time_stamp, estado) values(?,?,?)";
        $stmt = $db->prepare($sql);
        $affected_rows  = $stmt->execute($valores);
        $db->commit();
    }
    catch (PDOException $e)  {
     $db->rollBack();
     echo("<p>Nao e possivel criar reserva: {$e->getMessage()}</p>");
    }
}

function Pagar ($db,$metodo, $data, $numero)  {
    try{
        $db->beginTransaction();
        $valores= array($numero,$data,$metodo);
        $sql = "INSERT INTO paga(numero, data, metodo) values(?,?,?)";
        $stmt = $db->prepare($sql);
        $affected_rows  = $stmt->execute($valores);

        $valores= array($numero,$data,"Paga");
        $sql = "INSERT INTO estado(numero, time_stamp, estado) values(?,?,?)";
        $stmt = $db->prepare($sql);
        $affected_rows  = $stmt->execute($valores);
        $db->commit();
    }
    catch (PDOException $e)  {
```

```
      $db->rollBack();
      echo("<p>Nao e possivel inserir o pagamento: {$e->getMessage()}</p>");
   }
}
function TotalRealizado ($db,$morada)  {
   try{
      $valores=array($morada);
      $sql = "SELECT res.morada, res.codigo, SUM(res.TotalEspaco) AS Total
            FROM
               (SELECT e.morada, e.codigo, SUM(oe.tarifa * datediff(oe.data_fim, oe.data_inicio)) AS TotalEspaco
               FROM espaco e NATURAL JOIN oferta oe
                  NATURAL JOIN aluga ae
                  NATURAL JOIN paga pge
               GROUP BY  e.morada, e.codigo
               UNION
               SELECT p.morada, p.codigo_espaco, SUM(o.tarifa * datediff(o.data_fim, o.data_inicio)) AS
TotalPosto
               FROM posto p NATURAL JOIN oferta o
                  NATURAL JOIN aluga a
                  NATURAL JOIN paga pg
               GROUP BY  p.morada, p.codigo_espaco
               UNION
               SELECT ez.morada, ez.codigo AS codigo_espaco, 0
               FROM espaco ez
               WHERE (ez.morada, ez.codigo) NOT IN(
                  SELECT az.morada, az.codigo
                  FROM aluga az NATURAL JOIN paga pgz)
               ) res
            WHERE res.morada =?
            GROUP BY  res.morada, res.codigo";
      $stmt = $db->prepare($sql);
      $affected_rows  = $stmt->execute($valores);
      $result = $stmt -> fetchAll(PDO::FETCH_ASSOC);
      echo("<table border=\"1\">\n");
      echo("<tr><td>Lista de Total Realizado por Espaço</td></tr>\n");
      echo("<tr><td>Morada</td><td>Codigo do Espaco</td><td>Total Realizado</td></tr>\n");
      foreach($result as $row) {
         echo("<tr><td>");
         echo($row['morada']);
         echo("</td><td>");
         echo($row['codigo']);
         echo("</td><td>");
         echo($row['Total']);
         echo("</td></tr>\n");
      }
      echo("</table>\n"); }
   catch (PDOException $e)  {
     echo("<p>Nao e possivel mostrar total realizado: {$e->getMessage()}</p>");
   }
```

}

No ficheiro mostraReservaOferta.php, para mostrar as ofertas que podem ter reservas (reservas sem estado aceite) fazemos a seguinte query:

```
$sql = "SELECT o.morada, o.codigo, o.data_inicio, o.data_fim, o.tarifa
        FROM oferta o
        WHERE NOT EXISTS(
          SELECT o2.morada, o2.codigo, o2.data_inicio, o2.data_fim, o2.tarifa
          FROM oferta o2 NATURAL JOIN aluga a
            NATURAL JOIN estado e
          WHERE o2.morada = o.morada
            AND o2.codigo = o.codigo
            AND o2.data_inicio = o.data_inicio
            AND e.estado = 'Aceite');";
$result = $db->query($sql);
```

No ficheiro mostraReservaPagar.php, para mostrar as reservas que podem ser pagas(reservas com estado aceite e sem estado paga) fazemos a seguinte query:

```
$sql = "SELECT r1.numero
        FROM reserva r1 NATURAL JOIN estado e1
        WHERE e1.estado = 'Aceite'
          AND NOT EXISTS(
            SELECT *
            FROM reserva r2 NATURAL JOIN estado e2
            WHERE r2.numero = r1.numero
              AND e2.estado = 'Paga');";

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

Links para o site:
http://web.tecnico.ulisboa.pt/~ist180970/bd/
ou
http://web.tecnico.ulisboa.pt/~ist181719/bd/