Instituto Superior Técnico

Projeto de Bases de Dados

Parte 3

Eduardo Janicas: 78974 (21H)

Diana Antunes: 82448 (30H)

Nuno Fernandes: 80774 (21H)

Número de grupo: 44

Turno: BD225179L06

Criação da Base de Dados

Para a criação da Base de Dados usámos dois ficheiros fornecidos pelos professores, o schema.sql e o populate.sql que foi utilizado para popular a base de dados.

SQL

a) SELECT DISTINCT morada, codigo_espaco

```
FROM posto p
WHERE NOT EXISTS (
        SELECT morada
        FROM aluga a
        WHERE p.codigo = a.codigo
);
```

Selecionámos a morada e o código de espaço de todos os postos exceto dos que já foram ou estão alugados, ou seja, na tabela aluga. Para tal usámos uma sub-query que seleciona a morada dos postos alugados. Ficando apenas com os espaços que têm postos que nunca foram alugados.

b) SELECT morada

```
FROM aluga NATURAL JOIN reserva
GROUP BY morada
HAVING COUNT(morada) >= (
        SELECT AVG(temp.count)
        FROM (
                SELECT COUNT(morada) as count
                FROM aluga NATURAL JOIN reserva
                GROUP BY morada) as temp
);
```

Selecionámos a morada dos edifícios onde, por morada, verificámos que o número de reservas é superior à média de número de reservas de todos os edifícios. Para tal usámos uma sub-query que seleciona o número de vezes que a morada se encontra reservada e comparamos esse número com a média de modo a ficar com os edifícios com maior número de reservas que a média.

c) SELECT nif, nome

```
FROM user
NATURAL JOIN(
        SELECT nif
        FROM aluga NATURAL JOIN fiscaliza
        GROUP BY nif
        HAVING COUNT(DISTINCT id) = 1
) AS temp;
```

Selecionámos o nif e o nome do utilizador cujos alugáveis foram fiscalizados sempre pelo mesmo fiscal. Para tal usámos uma sub-query que para cada nif de utilizador seleciona-o se tiver apenas um id de fiscal associado.

d) SELECT morada, codigo_espaco, pago

```
FROM
(SELECT morada, codigo_espaco, sum((datediff(data_fim, data_inicio))*tarifa)
as pago
FROM oferta NATURAL JOIN aluga NATURAL JOIN paga NATURAL JOIN posto
WHERE (data between '2016-01-01 00:00:01' and '2016-12-31 23:59:59')
GROUP BY codigo_espaco, morada) as temp
UNION
(SELECT morada, codigo as codigo_espaco, sum((datediff(data_fim, data_inicio))*tarifa) as
pago
FROM oferta NATURAL JOIN aluga NATURAL JOIN paga NATURAL JOIN espaco
WHERE (data between '2016-01-01 00:00:01' and '2016-12-31 23:59:59')
GROUP BY codigo_espaco, morada);
```

Selecionámos em ambas as sub-queries a morada, código de espaço e soma da tarifa total realizada por cada um dos espaços em 2016. A primeira sub-query funciona para espaços e a segunda para postos. Fizemos um UNION entre ambas, o que permitiu juntar todos os valores numa só tabela o que permite com que possamos selecionar a morada, código de espaço e montante total realizado durante o ano de 2016 pelo espaço.

e) SELECT morada, codigo_espaco

```
FROM (SELECT morada, codigo_espaco, COUNT(codigo) as count
        FROM posto
```

```
        GROUP BY morada, codigo_espaco) as temp
        NATURAL JOIN
        (SELECT aceite.morada as morada, aceite.codigo_espaco as codigo_espaco,
        count(aceite.codigo) as count
        FROM (SELECT DISTINCT morada, codigo_espaco, codigo
                FROM posto NATURAL JOIN aluga NATURAL JOIN estado
                WHERE estado = 'aceite') AS aceite
GROUP BY morada, codigo_espaco) as total_aceite;
```

Selecionámos a morada e código_espaco do NATURAL JOIN de duas sub-queries de forma a obter os espaços de trabalho cujos postos nele contidos foram todos alugados. Começámos por criar uma sub-query que resulta numa tabela com as colunas morada, código de espaço e número de postos que o espaço contém. De seguida criámos outra tabela usando uma sub-query com as colunas morada, código de espaço e número de postos que o espaço contém que já tiveram um estado aceite (já foram alugados). Como ambas as tabelas têm os mesmos nomes das colunas fazemos NATURAL JOIN entre as duas de forma a obter as linhas em que o número de postos é igual ao número de postos que já tiveram postos aceites.

Restrições de Integridade

a) DELIMITER //

```
DROP TRIGGER IF exists overlapping_offers;
CREATE TRIGGER overlapping_offers
BEFORE INSERT ON oferta
FOR EACH ROW
BEGIN
        IF exists(SELECT * FROM oferta WHERE codigo = NEW.codigo AND morada =
        NEW.morada
                AND ((new.data_inicio >= data_inicio AND new.data_inicio <= data_fim)
                        OR (new.data_fim >= data_inicio AND new.data_fim <= data_fim)
                        OR (new.data_inicio <= data_inicio AND new.data_fim >= data_fim)))
                THEN call erro_overlapping_offers();
        END IF;
END //
DELIMITER ;
```

Com esta restrição fazemos com que não seja possível adicionar ofertas para o mesmo alugável com datas sobrepostas. Para definir esta restrição utilizámos um TRIGGER.

b) DELIMITER //

```
DROP TRIGGER IF exists pay_date_check;
CREATE TRIGGER pay_date_check
BEFORE INSERT ON paga
FOR EACH ROW
BEGIN
        IF exists(SELECT * FROM estado WHERE numero = NEW.numero AND NEW.data <=
        time_stamp)
                THEN call erro_pay_date_check();
        END IF;
END //
DELIMITER ;
```

Com esta restrição fazemos com que a data de pagamento de uma reserva paga não seja superior ao timestamp do último estado dessa reserva. Para defenir esta restrição utilizámos um TRIGGER.

Aplicação

Separámos a aplicação em duas partes:
- **Edifícios**, onde se pode inserir e remover Edifícios, Espaços e Postos de trabalho; e também se pode listar o total realizado por Espaço de trabalho.
- **Ofertas**, onde se pode criar e remover Ofertas, criar Reservas sobre Ofertas e pagar Reservas.

Utilizamos prepared statements na maior parte das queries do código php, com bindParam para evitar SQL Injections e tornar a aplicação mais eficiente.
Utilizamos os métodos Post, Get e a superglobal _SESSION para armazenar e passar variáveis entre ficheiros.
Sempre que actualizamos a base de dados, tanto no ficheiro insert.php como no remove.php, utilizamos transacções, que executam rollback caso a actualização não seja bem sucedida.

- Para inserir um Edifício, executamos a seguinte função:

```
function insereEdificio($morada) {...
$sql = "INSERT INTO edificio (morada) VALUES(:morada)"; ... }
```

- Para inserir um Espaço de Trabalho num dado Edifício, a morada é passada em variável e o utilizador insere o código e a foto do novo Espaço. É chamada a função insereEspaco, que chama a insereAlugavel e só depois adiciona o Espaço.

```
function insereEspaco($morada, $codigo, $foto) {...
        insereAlugavel($morada, $codigo, $foto);
        $sql = "INSERT INTO espaco (morada, codigo) VALUES(:morada, :codigo)";... }

function insereAlugavel($morada, $codigo, $foto) {...
        $sql = "INSERT INTO alugavel (morada, codigo, foto) VALUES(:morada,
:codigo, :foto)";...}
```

- Para inserir um Posto de Trabalho num dado Espaço de um dado Edifício, a morada e o código do Espaço são passados em variáveis e o utilizador insere o código e a foto do novo Posto. É chamada a função inserePosto, que chama a insereAlugavel e só depois adiciona o Posto.

```
function inserePosto($morada, $codigo, $codigo_espaco, $foto) {...
        insereAlugavel($morada, $codigo, $foto);
        $sql = "INSERT INTO posto (morada, codigo, codigo_espaco)
        VALUES(:morada, :codigo, :codigo_espaco)"; ...}
```

- Para criar uma Oferta ,o utilizador insere a morada, código, data_inicio, data_fim e tarifa da nova Oferta. Apenas pode ser inserida uma oferta num Edificio e Alugavel existentes, com morada e codigo existentes.

```
function insereOferta($morada, $codigo, $data_inicio, $data_fim, $tarifa) {...
        $sql = "INSERT INTO oferta (morada, codigo, data_inicio, data_fim, tarifa)
        VALUES(:morada, :codigo, :data_inicio, :data_fim, :tarifa)"; ...}
```

- Para criar uma Reserva sobre uma Oferta, são passados em variáveis os valores de morada, codigo e data_inicio da respectiva Oferta, e o utilizador insere o seu NIF e o numero da Reserva. O NIF terá de pertencer à base de dados (tabela user). Além de se inserir a nova entrada na tabela reserva, adiciona-se também na tabela aluga e

estado (com o respectivo timestamp gerado na execução da função, e com o parâmetro ‘Aceite’ na coluna estado).

```
function insereReserva($morada, $codigo, $data_inicio, $nif, $numero) {...
        $sql = "INSERT INTO reserva (numero) VALUES(:numero)";
        ...
        $sql2 = "INSERT INTO aluga (morada, codigo, data_inicio, nif, numero)
        VALUES(:morada, :codigo, :data_inicio, :nif, :numero)";
        ...
        $sql3 = "INSERT INTO estado (numero, time_stamp, estado)
        VALUES(:numero, :time_stamp, 'Aceite')";...}
```

- Para pagar uma Reserva, o numero da Reserva é passado em variável e o utilizador insere o metodo. Inserimos a nova entrada na tabela paga, e adiciona-se também na tabela estado (com o respectivo timestamp gerado na execução da função, e com o parâmetro ‘Paga’ na coluna estado).

```
function inserePaga($numero, $metodo) {...
        $sql = "INSERT INTO paga (numero, data, metodo)
        VALUES(:numero, :time_stamp, :metodo)";
        ...
        $sql2 = "INSERT INTO estado (numero, time_stamp, estado)
        VALUES(:numero, :time_stamp, 'Paga')"; ...}
```

- Para remover um Edifício, temos de remover todos os Alugáveis nele contidos e só depois removemos o Edifício.

Para remover os Alugáveis contidos no Edifício, removemos todos os Postos, Espaços, Arrendas e Ofertas com a morada do Edifício a remover, e só depois removemos os Alugáveis. Para remover todas as entradas de Arrenda relacionadas com os Alugáveis a remover, temos antes de remover todas as entradas da relação Fiscaliza com a morada correspondente. Similarmente, para remover as Ofertas relacionadas com o Alugável a remover, removemos antes todas as entradas na tabela Aluga com a morada correspondente.

- Para remover um Espaco ou um Posto de Trabalho de um dado Edifício, chamamos a função removeAlugável que, como anteriormente, remove todas as associações com as respectivas moradas e códigos.

- Para remover uma Oferta, removemos todas as entradas na tabela Aluga com morada, codigo e data_inicio correspondentes.

```
function removeEdificio($morada) {...
```

```
        removeAlugavel($morada, NULL);
        $sql = "DELETE FROM edificio WHERE morada = :morada;"; ...}

function removeAlugavel($morada, $codigo) { ...
        removePosto($morada, $codigo, NULL);
        removeEspaco($morada, $codigo);
        removeArrenda($morada, $codigo);
        removeOferta($morada, $codigo, NULL);
        $sql = "DELETE FROM alugavel WHERE morada = :morada AND (:codigo IS
NULL OR codigo = :codigo);"; ...}

function removeArrenda($morada, $codigo) {...
        removeFiscaliza($morada, $codigo);
        $sql = "DELETE FROM arrenda WHERE morada = :morada AND (:codigo IS
NULL OR codigo = :codigo);"; ...}

function removeFiscaliza($morada, $codigo) {...
        $sql = "DELETE FROM fiscaliza WHERE morada = :morada AND (:codigo IS
NULL OR codigo = :codigo);"; ... }

function removeEspaco($morada, $codigo) {...
        removePosto($morada, NULL, $codigo); //remove todos os postos nele
contidos
        $sql = "DELETE FROM espaco WHERE morada = :morada AND (:codigo IS
NULL OR codigo = :codigo);"; ...}

function removePosto($morada, $codigo, $codigo_espaco) {...
        $sql = "DELETE FROM posto WHERE morada = :morada
        AND (:codigo IS NULL OR codigo = :codigo)
        AND (:codigo_espaco IS NULL OR codigo_espaco = :codigo_espaco);"; ...}

function removeOferta($morada, $codigo, $data_inicio) {...
        removeAluga($morada, $codigo, $data_inicio);
        $sql = "DELETE FROM oferta WHERE (morada = :morada
        AND (:codigo IS NULL OR codigo = :codigo)
        AND (:data_inicio IS NULL OR data_inicio = :data_inicio));"; ...}

function removeAluga($morada, $codigo, $data_inicio) {...
        $sql = "DELETE FROM aluga WHERE (morada = :morada
        AND (:codigo IS NULL OR codigo = :codigo)
        AND (:data_inicio IS NULL OR data_inicio = :data_inicio));"; ...}
```

- Finalmente, para mostrar o total realizado para cada espaço para um dado Edifício, passamos a morada como variável no espaco.php e executamos a seguinte query no ficheiro total.php:

```
SELECT morada, codigo_espaco, pago FROM
(SELECT morada, codigo_espaco, sum((datediff(data_fim, data_inicio))*tarifa) as pago
FROM oferta NATURAL JOIN aluga NATURAL JOIN paga NATURAL JOIN posto
WHERE (morada = '$morada')
GROUP BY codigo_espaco, morada) as temp
UNION
(SELECT morada, codigo as codigo_espaco, sum((datediff(data_fim, data_inicio))*tarifa) as
pago
FROM oferta NATURAL JOIN aluga NATURAL JOIN paga NATURAL JOIN espaco
WHERE (morada = '$morada')
GROUP BY codigo_espaco, morada);
```