PROJETO DE BASES DE DADOS

Parte 3

Grupo 40 – Turno BD225179L03

Rodrigo Lousada 81115 – 12 horas

Carlos Antunes 81525 – 12 horas

Nelson Duarte 67045 – 12 horas

- Base de Dados

Na primeira fase de desenvolvimento do projeto o grupo começou por trabalhar na criação das tabelas da base de dados usando como referência o esquema relacional apresentado no enunciado do projeto. Mais tarde o corpo docente publicou na página da cadeira dois scripts SQL para criação das tabelas (schema.sql) e população da base de dados (populate.sql), razão pela qual o grupo passou a utilizar os scripts publicados em favor dos própios tendo só adicionado ao schema.sql as chaves SQL "ON UPDATE CASCADE" a todas as foreign keys de cada tabela. Embora inicialmente o grupo tenha achado que seria indicada a utilizada a chave "ON DELETE CASCADE" após uma breve conversa com o professor entendemos a razão para este não ser aplicado no projeto.

A base de dados esta incluida no ficheiro zip com o nome schema.sql.

- Queries SQL

a) Quais os espaços com postos que nunca foram alugados?
Nesta alínea o grupo optou por excluir todos os espaços que não possuem postos do output.

```
SELECT DISTINCT morada, codigo_espaco AS codigo
FROM posto
WHERE   (morada, codigo) NOT IN
        (SELECT morada, codigo
        FROM aluga NATURAL JOIN posto);
```

b) Quais edifícios com um número de reservas superior à média?

```
SELECT morada, S.num_reservas
FROM
        (SELECT edificio.morada, count(aluga.morada) AS num_reservas
        FROM edificio LEFT OUTER JOIN aluga ON edificio.morada =
        aluga.morada
        GROUP BY edificio.morada) AS S
HAVING S.num_reservas > (SELECT AVG(num_reservas) FROM
        (SELECT edificio.morada, count(aluga.morada) AS num_reservas
        FROM edificio LEFT OUTER JOIN aluga ON edificio.morada =
        aluga.morada GROUP BY edificio.morada) AS A);
```

c) Quais utilizadores cujos alugáveis foram fiscalizados sempre pelo mesmo fiscal?

```
SELECT nif
FROM
        (SELECT DISTINCT id, nif
        FROM fiscaliza NATURAL JOIN arrenda
        GROUP BY id, nif) AS A
GROUP BY nif
HAVING count (*) = 1;
```

d) Qual o montante total realizado (pago) por cada espaço durante o ano de 2016? Assuma que a tarifa indicada na oferta é diária. Deve considerar os casos em que o espaço foi alugado totalmente ou por postos.

```
SELECT morada, codigo, SUM(montante) AS montante_total
FROM
        (SELECT morada, codigo, 0 AS montante
        FROM espaco
        UNION
        SELECT morada, codigo_espaco AS codigo, datediff(data_fim,
        data_inicio) * tarifa AS montante
        FROM posto NATURAL JOIN oferta NATURAL JOIN paga
        NATURAL JOIN aluga WHERE year(data) = 2016
        UNION
        SELECT morada,codigo,datediff(data_fim, data_inicio) * tarifa AS
        montante
        FROM espaco NATURAL JOIN oferta NATURAL JOIN paga NATURAL JOIN
        aluga
        WHERE year(data) = 2016) AS A
GROUP BY morada, codigo;
```

e) Quais os espaços de trabalho cujos postos nele contidos foram todos alugados? (Por alugado entende-se um posto de trabalho que tenha pelo menos uma oferta aceite, independentemente das suas datas.)
Nesta alínea o grupo baseou-se na Álgebra Relacional da entrega passada e acabou por decidir, tal como na alínea a), excluir todos os espaços que não tenham postos.

```
SELECT DISTINCT morada, codigo_espaco AS codigo
FROM posto
WHERE   (morada, codigo_espaco) NOT IN
        (SELECT morada, codigo_espaco
        FROM posto
        WHERE (morada, codigo) NOT IN
                (SELECT morada, codigo
                FROM estado NATURAL JOIN aluga NATURAL JOIN posto
                WHERE estado = 'Aceite'));
```

- Restrições de Integridade

As Restrições de Integridade foram representadas como triggers no ficherio storedprocstriggers.sql
a) RI-1: "Não podem existir ofertas com datas sobrepostas"

```
CREATE TRIGGER ofertas_sobrepostas
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 ERROR_OFERTA;
    END IF;
```

b) RI-2: "A data de pagamento de uma reserva paga tem de ser superior ao timestamp do último estado dessa reserva"
Neste caso era escusado irmos buscar o máximo timestamp pois se data fosse menor que qualquer um dos timestamps então seria obrigatoriamente menor que o maior timestamp, no entanto por uma razão de lógica e compreensão achámos por bem manter.

```
CREATE TRIGGER pagamento_timestamp
BEFORE INSERT ON paga
FOR EACH ROW
BEGIN
        IF EXISTS       (SELECT * FROM estado
                        WHERE new.numero = numero AND new.data <= (SELECT
                        MAX (time_stamp) FROM estado WHERE new.numero =
                        numero))
                THEN CALL ERROR_PAYMENT;
        END IF;
```

- Desenvolvimento da Aplicação

Todas as instâncias da nossa aplicação são acedidas através da página inicial index.html (localizada na pasta web) todos os restantes ficheiros php são usados de forma a complementar as funcionalidades do mesmo. O grupo acabou por separar em muito ficheiros as diferentes funcionalidades deforma a que não só conseguissem trabalhar os 3 em simultâneo em ambos, como a obter uma maior abstração. A grande maioria é decomposta nos ficheiros cria, insere, remove.

a) Inserir e Remover Edifícios, Espaços e Postos de trabalho.

- Para inserir edificios, espaços e postos de trabalho na base de dados são usadas as seguintes queries nas quais o utilizador deve inserir para cada novo elemento as suas correspondentes chaves. Nas inserções de novos espaços e postos tambem é criado um novo elemento alugavel associado ao espaço/posto correspondente.

```
$sql =      "insert into edificio values('$morada');";

$sql =      "insert into alugavel values('$morada', '$codigo', '$foto');

            insert into espaco values('$morada', '$codigo'); ";

$sql =      "insert into alugavel values('$morada', '$codigo', '$foto');

            insert into posto values('$morada', '$codigo', '$codigo_espaco'); ";
```

- Para remover edificios, espaços e postos de trabalho são usadas as seguintes queries nas quais establecemos qual é o elemento a remover em função das chaves primarias de cada elemento. Nas remoções de espaços e postos tambem é removido o elemento alugavel a associado ao espaço/posto correspondente.

```
$sql =    "delete from edificio WHERE morada = '$morada';";

$sql =    "delete from espaco WHERE morada = '$morada' AND codigo = '$codigo';

          delete from alugavel WHERE morada = '$morada' AND codigo = '$codigo';";

$sql =    "delete from posto WHERE morada = '$morada' AND codigo = '$codigo';

          delete from alugavel WHERE morada = '$morada' AND codigo = '$codigo';";
```

b) Criar e Remover Ofertas

- Para criar ofertas usamos a seguinte query na qual o utilizador deve inserir as instâncias para a nova oferta inserida na base de dados.

```
$sql =        "insert into oferta values('$morada', '$codigo', '$data_inicio', '$data_fim', $tarifa);";
```

- Para remover ofertas usamos a seguinte query na qual apagamos da base de dados a oferta que o utilizador selecionou em função das chaves primarias do elemento.

```
$sql =    "delete from oferta WHERE codigo = '$codigo' AND morada = '$morada' AND data_inicio =
          '$data_inicio';";
```

c) Criar reservas sobre Ofertas.

- Para a criação de reservas são usadas as query seguintes queries localizadas em ficheiros php diferentes.

A primeira query apenas insere uma nova reserva na base de dados com um numero introduzido pelo utilizador não estando esta associada a nenhuma oferta. Não sendo de facto necessário para a aplicação.

```
$sql =        "insert into reserva values('$numero');";
```

A segunda query insere uma nova reserva que é associada a uma oferta através da criação de um novo elemento aluga e um novo elemento estado com as chaves introduzidas pelo utilizador. O terceiro campo do estado é preenchida com o 'Pendente'.

```
$sql =        "insert into reserva values('$numero');
              insert into aluga values('$morada', '$codigo', '$data_inicio', '$nif', '$numero');
              insert into estado values('$numero', '$time_stamp', 'Pendente');";
```

d) Pagar reservas.

- Para o pagamento de reservas é inserido na base de dados um novo elemento paga e um novo elemento estado a ele associado com as chaves introduzidas pelo utilizador. O terceiro campo do estado é preenchido com 'Pago'.

```
$sql =        "insert into paga values('$numero', '$data', '$metodo');
              insert into estado values('$numero', '$data', 'Pago');";
```

e) Para um dado edifício mostrar o total realizado para cada espaço.
- Para mostrar o montante total de cada espaço foi usada a seguinte query na qual visitamos cada espaço e cada posto do edificio e sumamos os seus correspondentes montantes. Esta query sofreu poucas alterações como retirar as restrições do ano ser 2016 e restringir apenas à morada a que nos estávamos a referir.

```
$sql =      "SELECT codigo, SUM(montante) AS montante_total
            FROM    (SELECT morada, codigo, 0 AS montante
                     FROM espaco
                     UNION
                     SELECT morada, codigo_espaco AS codigo, datediff(data_fim, data_inicio) * tarifa AS
                     montante
                     FROM posto NATURAL JOIN oferta NATURAL JOIN paga NATURAL JOIN aluga
                     UNION
                     SELECT morada,codigo,datediff(data_fim, data_inicio) * tarifa AS montante
                     FROM espaco NATURAL JOIN oferta NATURAL JOIN paga NATURAL JOIN aluga) AS A
            WHERE morada = '$morada'
            GROUP BY morada, codigo;";
```