Projeto de Bases de Dados, Parte 3

Grupo 1, BD225179L08
Daniel Correia, 80967
Carolina Inês Xavier, 81172
Inês Leite, 81328
Esforço: 24 horas

Criação da Base de Dados (schema.sql e populate.sql fornecidos pelos professores)

```
drop table if exists estado;
drop table if exists paga;
drop table if exists aluga;
drop table if exists reserva;
drop table if exists oferta;
drop table if exists posto;
drop table if exists espaco;
drop table if exists fiscaliza;
drop table if exists arrenda;
drop table if exists alugavel;
drop table if exists edificio;
drop table if exists fiscal;
drop table if exists user;

create table user (
  nif varchar(9) not null unique,
  nome varchar(80) not null,
  telefone varchar(26) not null,
  primary key(nif));

create table fiscal (
  id int not null unique,
  empresa varchar(255) not null,
  primary key(id));

create table edificio (
  morada varchar(255) not null unique,
  primary key(morada));

create table alugavel (
  morada varchar(255) not null,
  codigo varchar(255) not null,
  foto varchar(255) not null,
  primary key(morada, codigo),
  foreign key(morada) references edificio(morada));

create table arrenda (
  morada varchar(255) not null,
  codigo varchar(255) not null,
  nif varchar(9) not null,
  primary key(morada, codigo),
  foreign key(morada, codigo) references
alugavel(morada, codigo),
  foreign key(nif) references user(nif));

create table fiscaliza (
  id int not null,
  morada varchar(255) not null ,
  codigo varchar(255) not null ,
  primary key(id, morada, codigo),
  foreign key(morada, codigo) references
arrenda(morada, codigo),
  foreign key(id) references fiscal(id));

create table reserva (
  numero varchar(255) not null unique,
  primary key(numero));


create table espaco (
  morada varchar(255) not null,
  codigo varchar(255) not null,
  primary key(morada, codigo),
  foreign key(morada, codigo) references
alugavel(morada, codigo));

create table posto (
  morada varchar(255) not null,
  codigo varchar(255) not null,
  codigo_espaco varchar(255) not null,
  primary key(morada, codigo),
  foreign key(morada, codigo) references
alugavel(morada, codigo),
  foreign key(morada, codigo_espaco) references
espaco(morada, codigo));

create table oferta (
  morada varchar(255) not null,
  codigo varchar(255) not null,
  data_inicio date not null,
  data_fim date not null,
  tarifa numeric(19,4) not null,
  primary key(morada, codigo, data_inicio),
  foreign key(morada, codigo) references
alugavel(morada, codigo));

create table aluga (
  morada varchar(255) not null,
  codigo varchar(255) not null,
  data_inicio date not null,
  nif varchar(9) not null,
  numero varchar(255) not null,
  primary key(morada, codigo, data_inicio, nif,
numero),
  foreign key(morada, codigo, data_inicio) references
oferta(morada, codigo, data_inicio),
  foreign key(nif) references user(nif),
  foreign key(numero) references reserva(numero));

create table paga (
  numero varchar(255) not null unique,
  data timestamp not null,
  metodo varchar(255) not null,
  primary key(numero),
  foreign key(numero) references reserva(numero));

create table estado (
  numero varchar(255) not null,
  time_stamp timestamp not null,
  estado varchar(255) not null,
  primary key(numero, time_stamp),
  foreign key(numero) references reserva(numero))
```

SQL (queries.sql)

a) Espaços com postos que nunca foram alugados

```
SELECT DISTINCT morada, codigo_espaco
FROM posto
WHERE (morada, codigo) NOT IN(
    SELECT DISTINCT A.morada, A.codigo FROM aluga A, estado B
    WHERE A.numero=B.numero
    AND B.estado="Aceite");
```

- Primeiro selecionamos todos os postos ou espaços que foram alugados e depois escolhemos os postos que não existem neste primeiro conjunto. Estes são os postos que nunca foram alugados, donde conseguimos obter a morada e código do espaço a que pertencem.

b) Edifícios com um número de reservas superior à média

```
SELECT DISTINCT morada
FROM aluga
GROUP BY morada
HAVING count(numero) > (SELECT AVG(valor)
                        FROM(
                          SELECT count(numero) AS valor
                          FROM aluga
                          GROUP BY morada
                          ) AS S);
```

- Primeiro contamos quantas reservas é que estão associadas a cada edificio. Depois calculamos a média de reservas a partir destas contagens. Finalmente, selecionamos os edificios com o número de reservas superior a esta média calculada.

c) Utilizadores cujos alugáveis foram fiscalizados sempre pelo mesmo fiscal

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

- Selecionamos um conjunto com informações dos alugáveis arrendados por cada utilizador e do fiscal que lhe está associado.
- Para cada utilizador, contamos o número de fiscais diferentes: se o total de fiscais for 1, então os alugáveis desse utilizador foram fiscalizados sempre pelo mesmo fiscal.

d) Total realizado (pago) por cada espaço durante o ano de 2016

```
SELECT Res.morada, Res.codigo, SUM(Res.total) AS totalRealizado
FROM( ( SELECT S.morada, S.codigo, SUM(S.total) AS total
        FROM(( SELECT morada, codigo, SUM(tarifa * (data_fim - data_inicio)) AS total
               FROM espaco NATURAL JOIN aluga NATURAL JOIN oferta NATURAL JOIN paga
               WHERE year(paga.data) = '2016'
               GROUP BY morada, codigo)
             UNION
             ( SELECT morada, codigo_espaco, SUM(tarifa * (data_fim - data_inicio)) AS total
               FROM posto NATURAL JOIN aluga NATURAL JOIN oferta NATURAL JOIN paga
               WHERE year(paga.data) = '2016'
               GROUP BY morada, codigo)
        ) AS S
        GROUP BY morada, codigo)
        UNION
      ( SELECT morada, codigo, 0 AS total FROM espaco )
) AS Res
GROUP BY morada, codigo;
```

- Como podem existir espaços com total realizado nulo (ou seja, cuja reserva nunca foi paga), consideramos que o total realizado para qualquer espaço começa com valor zero.
- Calculamos o total realizado por cada espaço ou posto durante o ano de "2016". Depois, com base nestes valores, para cada espaço, determinamos o seu total (total dos seus postos + total do espaço).
- O resultado final consiste em somar, para cada espaço, os totais calculados aos valores iniciais (zero). Os espaços com total zero no resultado final são espaços que não foram pagos em 2016.

e) Espaços de trabalho cujos postos nele contidos foram todos alugados

```
SELECT DISTINCT morada, codigo_espaco
    FROM posto
    WHERE(morada, codigo_espaco) NOT IN (
        SELECT DISTINCT morada, codigo_espaco
            FROM posto
            WHERE (morada, codigo) NOT IN(
                SELECT DISTINCT A.morada, A.codigo
                FROM aluga A, estado B
                WHERE A.numero=B.numero AND B.estado= 'Aceite' ));
```

- Primeiro selecionamos todos os postos ou espaços que foram alugados, escolhemos os postos que não existem neste primeiro conjunto e devolvemos a informação dos seus espaços associados, ou seja, ficamos com os espaços com postos que nunca foram alugados.
- Finalmente, selecionamos os espaços que têm postos e retiramos aqueles que pertencem ao conjunto anterior, ou seja, sobram apenas os espaços cujos postos foram todos alugados.

Restrições de Integridade (triggers.sql)

RI-1: Não podem existir ofertas com datas sobrepostas

```
DROP TRIGGER IF EXISTS ofertasDatasSobrepostas;
DELIMITER //
CREATE TRIGGER ofertasDatasSobrepostas BEFORE INSERT ON oferta
FOR EACH ROW
BEGIN
IF EXISTS(
    SELECT morada, codigo
    FROM (
        SELECT morada, codigo, data_inicio, data_fim
        FROM oferta
        WHERE morada = new.morada AND codigo = new.codigo
    ) AS O
    WHERE new.data_inicio <= O.data_fim AND new.data_fim >= O.data_inicio
    ) THEN
CALL ERROR;
END IF;
END//
DELIMITER ;
```

- Criamos um trigger do tipo "BEFORE INSERT" sobre a tabela oferta para verificar se a nova oferta a ser inserida está de acordo com a RI-1.
- Considerámos que esta restrição só se aplica a ofertas para o mesmo alugável.
- O trigger verifica se existe algum registo na tabela oferta, em relação ao mesmo alugável da nova oferta, cuja data de fim é maior ou igual à data de inicio da nova oferta e cuja data de inicio é menor ou igual à data de fim da nova oferta. Esta condição garante que existe sobreposição dos intervalos das datas das ofertas.

RI-2: A data de pagamento de uma reserva paga tem de ser superior ao timestamp do último estado dessa reserva

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

- Criamos um trigger do tipo "BEFORE INSERT" sobre a tabela paga para verificar se o pagamento da reserva a ser inserido está de acordo com a RI-2.
- O trigger verifica se existe algum registo na tabela estado com número de reserva igual ao número de reserva do pagamento a ser inserido, cujo time_stamp mais recente (ou seja, MAX(time_stamp)) é superior à data do pagamento. Nesse caso, o insert não é válido de acordo com a RI-2.

Desenvolvimento da Aplicação (php)

Utilização de Prepared Statements

```
function prepareAndExecuteStatement($connection, $statement, $arguments){
    $stmt = $connection->prepare($statement);
    $stmt->execute($arguments);
}
```

- No desenvolvimento da aplicação, utilizámos prepared statements nas operações sobre a base de dados que utilizem input do utilizador. Desta maneira conseguimos prevenir situações de SQL injection.
- A função descrita acima corresponde a uma função que criámos para abstrair o processo de utilização dos prepared statements (prepare + execute). Esta função foi utilizada pelas funcionalidades correspondentes às alíneas a), b), c) e d).

a) Inserir/Remover Edificio, Espaço ou Posto (feature 1)

```
$sql = "INSERT INTO alugavel VALUES (:moradaPosto, :codigoPosto, :fotografia);";
$data = array(':moradaPosto' => $moradaPosto, ':codigoPosto' => $codigoPosto, ':fotografia' => $fotografia);
prepareAndExecuteStatement($db, $sql, $data);
$sql = "INSERT INTO posto VALUES (:moradaPosto, :codigoPosto, :codigoEspaco);";
$data = array(':moradaPosto' => $moradaPosto, ':codigoPosto' => $codigoPosto, ':codigoEspaco' => $codigoEspaco);
prepareAndExecuteStatement($db, $sql, $data);
echo("<p>Executing PDO prepared statement for insert posto...</p>");
```

- Este exemplo corresponde a inserir um posto. O utilizador insere os dados relativos à morada, código do posto, código do espaço e fotografia.
- Ao nível da base de dados, inserir um posto consiste em fazer um insert na tabela alugavel com os dados enviados pelo utilizador e de seguida, fazer um insert na tabela posto.

```
$sql = "DELETE FROM posto WHERE morada = :moradaPosto AND codigo = :codigoPosto AND codigo_espaco = :codigoEspaco;";
$data = array(':moradaPosto' => $moradaPosto, ':codigoPosto' => $codigoPosto, ':codigoEspaco' => $codigoEspaco);
prepareAndExecuteStatement($db, $sql, $data);
$sql = "DELETE FROM alugavel WHERE morada = :moradaPosto AND codigo = :codigoPosto";
$data = array(':moradaPosto' => $moradaPosto, ':codigoPosto' => $codigoPosto);
prepareAndExecuteStatement($db, $sql, $data);
echo("<p>Executing PDO prepared statement for delete posto...</p>");
```

- Este exemplo corresponde a remover um posto. Na aplicação é apresentada uma lista dos postos existentes no sistema e o utilizador escolhe um desses postos para ser removido.
- Ao nível da base de dados, remover um posto corresponde a executar um delete primeiro na tabela posto e depois na tabela alugável, de acordo com os dados do posto escolhido. Esta ordem é necessária porque a tabela posto contêm uma foreign key derivada da tabela alugável.

b) Criar e remover ofertas (feature 2)

```
$sql = "INSERT INTO oferta VALUES (:morada, :codigo, :data_inicio, :data_fim, :tarifa);";
$data = array(':morada' => $moradaOferta, ':codigo' => $codigoOferta,':data_inicio' => $data_inicio,
              ':data_fim' => $data_fim, ':tarifa' => $tarifa,);
prepareAndExecuteStatement($db, $sql, $data);
echo("<p>Executing PDO prepared statement for insert oferta...</p>");
```

- Este exemplo corresponde a criar uma oferta. O utilizador insere os dados relativos à morada e código do alugável, data de inicio, data de fim e tarifa.
- Ao nível da base de dados, inserir uma oferta consiste em fazer um insert na tabela oferta com os dados enviados pelo.

c) Criar reservas sobre ofertas (feature 3)

```
$db->beginTransaction();
$sql = "INSERT INTO reserva VALUES (:numero);";
$data = array(':numero' => $numeroReserva,);
prepareAndExecuteStatement($db, $sql, $data);
echo("<p>Executing PDO prepared statement for insert new reserva...</p>");

$sql = "INSERT INTO estado VALUES (:numero, :time_stamp, :estado);";
$data = array(':numero' => $numeroReserva, ':time_stamp' => $data_inicio, ':estado' => 'Aceite',);
prepareAndExecuteStatement($db, $sql, $data);
echo("<p>Executing PDO prepared statement for insert new estado...</p>");

$sql = "INSERT INTO aluga VALUES (:morada, :codigo, :data_inicio, :nif, :numero);";
$data = array(':morada' => $moradaOferta,':codigo' => $codigoOferta, ':data_inicio' => $data_inicio,
              ':nif' => $nifUser,'numero' => $numeroReserva,);
prepareAndExecuteStatement($db, $sql, $data);
echo("<p>Executing PDO prepared statement for insert new aluga...</p>");
$db->commit();
```

- Para criar uma reserva sobre uma oferta, mostramos a lista de ofertas disponíveis ao utilizador e após a escolha da oferta, o utilizador envia input do número da reserva e do NIF.
- Ao nível da base de dados, criar a reserva consiste em fazer três inserts: um na tabela reserva para registar o novo número, outro na tabela estado para guardar o estado da nova reserva e o último na tabela aluga para registar a ligação entre utilizador, oferta e reserva criada.

d) Pagar reservas (feature 4)

```
$db->beginTransaction();
$sql = "INSERT INTO paga VALUES (:numero, :data, :metodo);";
$data = array(':numero' => $numeroReserva, ':data' => $dataPagamento,':metodo' => $metodoPagamento
prepareAndExecuteStatement($db, $sql, $data);
echo("<p>Executing PDO prepared statement for insert paga...</p>");
$sql = "INSERT INTO estado VALUES (:numero, :time_stamp, :estado);";
$data = array( ':numero' => $numeroReserva,':time_stamp' => $dataPagamento,':estado' => 'Paga',);
prepareAndExecuteStatement($db, $sql, $data);
echo("<p>Executing PDO prepared statement for insert estado...</p>");
$db->commit();
```

- Para pagar uma reserva, mostramos a lista de reserva que ainda não foram pagas ao utilizador e após a escolha da reserva a pagar, o utilizador insere a data do pagamento e o método.
- Ao nível da base de dados, pagar uma reserva consiste em fazer dois inserts: um na tabela paga para registar o método e data do pagamento e outro na tabela estado para atualizar o estado mais recente da recente para "Paga".

e) Listar total realizado por cada espaço de um edifício (feature 5)

```
$sql = "SELECT Res.morada, Res.codigo, SUM(Res.total) as totalRealizado
        FROM( ( SELECT S.morada, S.codigo, SUM(S.total) AS total
                FROM(( SELECT morada, codigo, SUM(tarifa * (data_fim - data_inicio)) AS total
                       FROM espaco NATURAL JOIN aluga NATURAL JOIN oferta NATURAL JOIN paga
                       GROUP BY morada, codigo
                       HAVING morada = :morada)
                     UNION
                     ( SELECT morada, codigo_espaco, SUM(tarifa * (data_fim - data_inicio)) AS total
                       FROM posto NATURAL JOIN aluga NATURAL JOIN oferta NATURAL JOIN paga
                       GROUP BY morada, codigo
                       HAVING morada = :morada)
                ) AS S
                GROUP BY morada, codigo)
                UNION
              ( SELECT morada, codigo, 0 AS total FROM espaco WHERE morada = :morada)
        ) AS Res
        GROUP BY morada, codigo;";
$moradaEdificio = $_REQUEST['morada'];
$data = array(':morada' => $moradaEdificio);
$stmt = $db->prepare($sql);
$stmt->execute($data);
$result = $stmt->fetchAll();
```

- Para listar o total realizado por cada espaço de um edifício, o utilizador insere a morada do edifício sobre o qual pretende listar esta informação.
- Ao nível da base de dados, listar estes dados corresponde a executar a query d) em que em vez de restringirmos aos totais realizados no ano "2016", selecionamos apenas os totais realizados por espaços com morada igual à morada inserida pelo utilizador.