Projeto de Bases de Dados

Parte 3

Grupo 38

4ª Feira - 11:00H

Docente André Pereira

Henrique Dias - 189455 : 33.3(3)%
Isabel Soares - 189466 : 33.3(3)%
Rodrigo Sousa - 189535 : 33.3(3)%

Comandos de criação da base de dados

```
DROP TABLE IF EXISTS correcao CASCADE;
DROP TABLE IF EXISTS proposta_de_correcao CASCADE;
DROP TABLE IF EXISTS incidencia CASCADE;
DROP TABLE IF EXISTS duplicado CASCADE;
DROP TABLE IF EXISTS anomalia_traducao CASCADE;
DROP TABLE IF EXISTS anomalia CASCADE;
DROP TABLE IF EXISTS item CASCADE;
DROP TABLE IF EXISTS local_publico CASCADE;
DROP TABLE IF EXISTS utilizador_qualificado CASCADE;
DROP TABLE IF EXISTS utilizador_regular CASCADE;
DROP TABLE IF EXISTS utilizador CASCADE;

CREATE OR REPLACE FUNCTION
utilizador_nao_qualificado (email_utilizador VARCHAR)
RETURNS BOOLEAN
AS $$
BEGIN
      RETURN NOT EXISTS (
        SELECT email
        FROM utilizador_qualificado
        WHERE email = email_utilizador
    );
END;
$$ LANGUAGE PLPGSQL;

CREATE OR REPLACE FUNCTION
utilizador_nao_regular (email_utilizador VARCHAR)
RETURNS BOOLEAN
AS $$
BEGIN
      RETURN NOT EXISTS (
        SELECT email
        FROM utilizador_regular
        WHERE email = email_utilizador
    );
END;
$$ LANGUAGE PLPGSQL;

CREATE OR REPLACE FUNCTION
lingua_nao_repetida (idAnomalia INT, lingua2 VARCHAR)
RETURNS BOOLEAN
AS $$
BEGIN
      RETURN NOT EXISTS (
        SELECT *
        FROM anomalia
        WHERE id = idAnomalia AND lingua = lingua2
    );
END;
$$ LANGUAGE PLPGSQL;

CREATE OR REPLACE FUNCTION
caixa_nao_sobrepoe (idAnomalia INT, zona2 BOX)
RETURNS BOOLEAN
AS $$
BEGIN
      RETURN NOT EXISTS (
        SELECT *
        FROM anomalia
        WHERE id = idAnomalia AND zona && zona2
    );
END;
$$ LANGUAGE PLPGSQL;

CREATE TABLE utilizador
(
    email VARCHAR NOT NULL PRIMARY KEY,
    "password" VARCHAR NOT NULL
);

CREATE TABLE utilizador_regular (
    email VARCHAR NOT NULL PRIMARY KEY CHECK
(utilizador_nao_qualificado(email)),
    FOREIGN KEY (email) REFERENCES utilizador(email) ON DELETE
CASCADE
);
```

```
CREATE TABLE utilizador_qualificado (
    email VARCHAR NOT NULL PRIMARY KEY,
    FOREIGN KEY (email) REFERENCES utilizador(email) ON DELETE
CASCADE
);

CREATE TABLE local_publico (
    latitude FLOAT NOT NULL CHECK (-90 <= latitude AND
latitude <= 90),
    longitude FLOAT NOT NULL CHECK (-180 <= longitude AND
longitude <= 180),
    nome VARCHAR NOT NULL,
    PRIMARY KEY(latitude, longitude)
);

CREATE TABLE item (
    id SERIAL PRIMARY KEY,
    descricao VARCHAR NOT NULL,
    localizacao VARCHAR NOT NULL,
    latitude FLOAT NOT NULL,
    longitude FLOAT NOT NULL,
    FOREIGN KEY (latitude, longitude) REFERENCES
local_publico(latitude, longitude) ON DELETE CASCADE
);

CREATE TABLE anomalia (
    id SERIAL PRIMARY KEY,
    zona BOX NOT NULL,
    imagem BYTEA NOT NULL,
    lingua VARCHAR NOT NULL,
    ts TIMESTAMP NOT NULL,
    descricao VARCHAR NOT NULL,
    tem_anomalia_redacao BOOLEAN NOT NULL
);

CREATE TABLE anomalia_traducao (
    id INT PRIMARY KEY,
    zona2 BOX NOT NULL CHECK(caixa_nao_sobrepoe(id, zona2)),
    lingua2 VARCHAR NOT NULL CHECK(lingua_nao_repetida(id,
lingua2)),
    FOREIGN KEY (id) REFERENCES anomalia(id) ON DELETE CASCADE
);

CREATE TABLE duplicado (
    item1 INT NOT NULL,
    item2 INT NOT NULL CHECK (item1 < item2),
    PRIMARY KEY(item1, item2),
    FOREIGN KEY (item1) REFERENCES item(id) ON DELETE CASCADE,
    FOREIGN KEY (item2) REFERENCES item(id) ON DELETE CASCADE
);

CREATE TABLE incidencia (
    anomalia_id INT NOT NULL PRIMARY KEY,
    item_id INT NOT NULL,
    email VARCHAR NOT NULL,
    FOREIGN KEY (anomalia_id) REFERENCES anomalia(id) ON
DELETE CASCADE,
    FOREIGN KEY (item_id) REFERENCES item(id) ON DELETE
CASCADE,
    FOREIGN KEY (email) REFERENCES utilizador(email) ON DELETE
CASCADE
);

CREATE TABLE proposta_de_correcao (
    email VARCHAR NOT NULL,
    nro INT NOT NULL,
    data_hora TIMESTAMP NOT NULL,
    texto VARCHAR NOT NULL,
    PRIMARY KEY (email, nro),
    FOREIGN KEY (email) REFERENCES
utilizador_qualificado(email) ON DELETE CASCADE
);
```

```
CREATE TABLE correcao (
    email VARCHAR NOT NULL,
    nro INT NOT NULL,
    anomalia_id INT NOT NULL,
    PRIMARY KEY (email, nro, anomalia_id),
    FOREIGN KEY (email, nro) REFERENCES
proposta_de_correcao(email, nro) ON DELETE CASCADE,
    FOREIGN KEY (anomalia_id) REFERENCES
incidencia(anomalia_id) ON DELETE CASCADE
);
```

Consultas em SQL

1.

```
SELECT nome
FROM (SELECT id AS item_id, latitude, longitude
      FROM item) AS i NATURAL JOIN
      (SELECT id AS anomalia_id
      FROM anomalia) AS a NATURAL JOIN
      (SELECT item_id, anomalia_id
      FROM incidencia) AS inc NATURAL JOIN
      (SELECT latitude, longitude, nome
      FROM local_publico) AS lp
GROUP BY nome
HAVING COUNT (anomalia_id) >= ALL (
      SELECT COUNT(anomalia_id)
      FROM (SELECT id AS item_id, latitude, longitude
            FROM item) AS i NATURAL JOIN
            (SELECT id AS anomalia_id
            FROM anomalia) AS a NATURAL JOIN
            (SELECT item_id, anomalia_id
            FROM incidencia) AS inc
      GROUP BY latitude, longitude
)
ORDER BY nome;
```

2.

```
SELECT email
FROM (SELECT email
      FROM utilizador_regular) AS u NATURAL JOIN
      (SELECT anomalia_id
      FROM incidencia) AS inc NATURAL JOIN
      (SELECT id AS anomalia_id, ts
      FROM anomalia) AS a NATURAL JOIN
      (SELECT id AS anomalia_id
      FROM anomalia_traducao) AS at
WHERE ts BETWEEN '2019-01-01 00:00:00' AND '2019-06-30
23:59:59'
GROUP BY email
```

```
HAVING COUNT (anomalia_id) >= ALL (
      SELECT COUNT(anomalia_id)
      FROM (SELECT email
            FROM utilizador_regular) AS u NATURAL JOIN
            (SELECT anomalia_id
            FROM incidencia) AS inc NATURAL JOIN
            (SELECT id AS anomalia_id, ts
            FROM anomalia) AS a NATURAL JOIN
            (SELECT id AS anomalia_id
            FROM anomalia_traducao) AS at
      WHERE ts BETWEEN '2019-01-01 00:00:00' AND '2019-06-30
23:59:59'
      GROUP BY email
)
ORDER BY email;
```

3.

```
SELECT email
FROM (SELECT email, COUNT(nome) AS nlocais
      FROM (SELECT anomalia_id, item_id, email
            FROM incidencia) AS inc NATURAL JOIN
            (SELECT id AS anomalia_id
            FROM anomalia
            WHERE ts BETWEEN '2019-01-01 00:00:00' AND '2019-
12-31 23:59:59') AS a NATURAL JOIN
            (SELECT id AS item_id, latitude, longitude
            FROM item
            WHERE latitude > 39.336775) AS i NATURAL JOIN
            (SELECT latitude, longitude, nome
            FROM local_publico) AS lp
      GROUP BY email) AS c
WHERE nlocais = (SELECT COUNT(*)
      FROM local_publico
      WHERE latitude > 39.336775)
ORDER BY email;
```

4.

```
SELECT email
FROM (SELECT anomalia_id, item_id, email
      FROM incidencia) AS inc NATURAL JOIN
      (SELECT id AS anomalia_id
      FROM anomalia
      WHERE extract(year from ts) = extract(year from NOW()))
AS a NATURAL JOIN
      (SELECT id AS item_id, latitude, longitude
      FROM item
      WHERE latitude < 39.336775) AS i
WHERE (email, anomalia_id) NOT IN (
      SELECT email, anomalia_id
      FROM correcao
      );
```

Arquitetura da aplicação PHP

A aplicação PHP é composta inicialmente por uma página onde é possível selecionar a estrutura de dados a alterar (Local Público, Item, Duplicados, Anomalia, etc.) e só depois na página correspondente estas estruturas são listadas, removidas, alteradas e/ou filtradas de acordo com o pedido no enunciado sobre cada uma delas. Todas as sub-páginas permitem ao utilizador voltar ao menu inicial através de um link no topo da página.

Tipicamente neste segundo nível (após o menu inicial) as estruturas são listadas e possivelmente existem *forms* que permitem ao utilizador filtrar e / ou adicionar mais uma instância dessa estrutura de dados, no caso de adição de uma nova instância o utilizador é redirecionado para uma página onde esta é efetivamente realizada e caso exista algum erro, o utilizador é informado deste. O utilizador será automaticamente redirecionado ao fim de 5 segundos desta página para a da respetiva estrutura.

Se nos tiver sido exigido a possibilidade de remoção e / ou edição de instâncias desta estrutura, é dado ao utilizador a possibilidade através de um / dois link's em frente de cada instância.

No caso de remoção remetem para uma página onde a respetiva instância é efetivamente removida, similar à da inserção, sendo só aqui que esta é efetivamente realizada ou transmitido ao utilizador os erros ocorridos. No caso de edição, o utilizador é remetido primeiro para uma página onde lhe será dada a oportunidade de edição dos campos possíveis e só depois para uma página onde é efetivamente editada como as mencionada acima (inserção e remoção).

Em todas as ações realizadas pelo o utilizador que efetivamente alteram a Base de Dados é garantido que não existe a possibilidade de SQL Injection através dos métodos *'prepare'* e *'execute'*. É também garantida a sua atomicidade através de Transações.

Existe um ficheiro para cada uma das páginas existentes, no entanto todas as funções de Query encontram-se no ficheiro *'/lib/lib.sql'*, desta forma as páginas só lidam com o que é retornado destas Queries.

A aplicação pode ser testada no link: https://web.tecnico.ulisboa.pt/ist189535/bd/BD-1920/Parte3/web/