

**Projeto de Bases de Dados**

Parte 3

**Grupo 38**

4ª Feira - 11:00H

Docente André Pereira

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

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;

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;

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;

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/>