Projeto de Bases de Dados

Parte 3

Grupo 16

Turno BD225179577L05 – 2ªf 14h-15h30

Prof. Duarte Alexandre Galvão

Número	Nome	Esforço (horas)	Esforço (%)
89476	João Fonseca	15	33
89544	Tiago Pires	15	33
89552	Tomás Lopes	15	33

Comandos de criação da BD (schema.sql)

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

CREATE TABLE local_publico
(
    latitude NUMERIC(8,6) NOT NULL,
    longitude NUMERIC(9,6) NOT NULL,
    nome VARCHAR(100) NOT NULL,
    CONSTRAINT pk_local_publico PRIMARY KEY(latitude, longitude),
    CONSTRAINT check_latitude CHECK (latitude>=-90 AND latitude<=90),
    CONSTRAINT check_longitude CHECK (longitude>=-180 AND longitude<=180)
);

CREATE TABLE item
(
    id SERIAL NOT NULL UNIQUE,
    descricao TEXT NOT NULL,
    localizacao VARCHAR(100) NOT NULL,
    latitude NUMERIC(8,6) NOT NULL,
    longitude NUMERIC(9,6) NOT NULL,
    CONSTRAINT pk_item PRIMARY KEY(id),
    CONSTRAINT fk_item_local_publico FOREIGN KEY(latitude, longitude)
        REFERENCES local_publico(latitude, longitude) ON DELETE CASCADE
);

CREATE TABLE anomalia
(
    id SERIAL NOT NULL UNIQUE,
    zona BOX NOT NULL,
    imagem VARCHAR(1000) NOT NULL,
    lingua VARCHAR(100) NOT NULL,
    ts TIMESTAMP NOT NULL,
    descricao TEXT NOT NULL,
    tem_anomalia_redacao BOOLEAN NOT NULL,
    CONSTRAINT pk_anomalia PRIMARY KEY(id)
);

CREATE TABLE anomalia_traducao
(
    id INTEGER NOT NULL UNIQUE,
    zona2 BOX NOT NULL,
    lingua2 VARCHAR(100) NOT NULL,
    CONSTRAINT pk_anomalia_traducao PRIMARY KEY(id),
    CONSTRAINT fk_anomalia_traducao_anomalia FOREIGN KEY(id) REFERENCES anomalia(id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE duplicado
(
    item1 INTEGER NOT NULL,
    item2 INTEGER NOT NULL,
    CONSTRAINT pk_duplicado PRIMARY KEY(item1, item2),
    CONSTRAINT fk_duplicado_item1 FOREIGN KEY(item1)
        REFERENCES item(id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_duplicado_item2 FOREIGN KEY(item2)
        REFERENCES item(id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT check_items CHECK (item1<item2)
);

CREATE TABLE utilizador
(
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(100) NOT NULL,
    CONSTRAINT pk_utilizador PRIMARY KEY(email),
    CONSTRAINT check_email CHECK (email LIKE '%@%.%')
);

CREATE TABLE utilizador_qualificado
(
    email VARCHAR(100) NOT NULL UNIQUE,
    CONSTRAINT pk_utilizador_qualificado PRIMARY KEY(email),
    CONSTRAINT fk_utilizador_qualificado_utilizador FOREIGN KEY(email)
        REFERENCES utilizador(email) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE utilizador_regular
(
    email VARCHAR(100) NOT NULL UNIQUE,
    CONSTRAINT pk_utilizador_regular PRIMARY KEY(email),
    CONSTRAINT fk_utilizador_regular_utilizador FOREIGN KEY(email)
        REFERENCES utilizador(email) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE incidencia
(
    anomalia_id INTEGER NOT NULL UNIQUE,
    item_id INTEGER NOT NULL,
    email VARCHAR(100) NOT NULL,
    CONSTRAINT pk_incidencia PRIMARY KEY(anomalia_id),
    CONSTRAINT fk_incidencia_anomalia FOREIGN KEY(anomalia_id)
        REFERENCES anomalia(id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_incidencia_item FOREIGN KEY(item_id)
        REFERENCES item(id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_incidencia_utilizador FOREIGN KEY(email)
        REFERENCES utilizador(email) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE proposta_de_correcao
(
    email VARCHAR(100) NOT NULL,
    nro INTEGER NOT NULL,
    data_hora TIMESTAMP NOT NULL,
    texto TEXT NOT NULL,
    CONSTRAINT pk_proposta_de_correcao PRIMARY KEY(email, nro),
    CONSTRAINT fk_proposta_de_correcao_utilizador_qualificado FOREIGN KEY(email)
        REFERENCES utilizador_qualificado(email) ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE correcao
(
    email VARCHAR(100) NOT NULL,
    nro INTEGER NOT NULL,
    anomalia_id INTEGER NOT NULL,
    CONSTRAINT pk_correcao PRIMARY KEY(email, nro, anomalia_id),
    CONSTRAINT fk_correcao_proposta_de_correcao FOREIGN KEY(email, nro)
        REFERENCES proposta_de_correcao(email,nro) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_correcao_incidencia FOREIGN KEY(anomalia_id)
        REFERENCES incidencia(anomalia_id) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Consultas em SQL (queries.sql)

```
/*Query #1*/

SELECT nome
FROM item NATURAL JOIN local_publico NATURAL JOIN incidencia
WHERE id=item_id
GROUP BY nome, latitude, longitude
HAVING COUNT(*)>=ALL (
    SELECT COUNT(*)
    FROM item NATURAL JOIN incidencia
    WHERE id=item_id
    GROUP BY latitude, longitude
);


/*Query #2*/

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

```
/*Query #3*/

SELECT email
FROM utilizador
WHERE email NOT IN (
    SELECT DISTINCT email
    FROM (
        (SELECT email,latitude,longitude
        FROM utilizador,local_publico
        WHERE latitude>39.336775)
        EXCEPT
        (SELECT DISTINCT email,latitude,longitude
        FROM anomalia,incidencia,item NATURAL JOIN local_publico
        WHERE item_id=item.id AND
              anomalia_id=anomalia.id AND
              EXTRACT(year FROM ts)=2019 AND
              latitude>39.336775)
    ) AS R
);


/*Query #4*/

SELECT DISTINCT email
FROM (
    (SELECT email,anomalia_id
    FROM anomalia,incidencia,item NATURAL JOIN local_publico
    WHERE item.id=item_id AND
          anomalia.id=anomalia_id AND
          latitude<39.336775 AND
          EXTRACT(year FROM ts)=EXTRACT(year FROM CURRENT_DATE) AND
          email IN (SELECT email FROM utilizador_qualificado))
    EXCEPT
    (SELECT email,anomalia_id FROM correcao)
) AS R;
```

Explicação da arquitetura da aplicação PHP

A aplicação PHP pode ser iniciada abrindo a página inicial (`index.php`), que contém *links* para cada uma das seis funcionalidades pedidas em diferentes páginas - por exemplo, alínea a) na página a.php, alínea b) na página b.php, etc.

A primeira página (`a.php`) contém quatro tabelas, correspondentes aos **Locais Públicos**, aos **Itens**, às **Anomalias** e às **Anomalias de Tradução**. É possível apagar uma entrada de qualquer uma das tabelas clicando no botão X da respetiva entrada a apagar (ou, no caso das Anomalias de Tradução, apagando a entrada correspondente na tabela de Anomalias), e é possível adicionar uma entrada a cada tabela clicando no botão + debaixo da respetiva tabela e preenchendo o formulário que se segue a esse clique (ou, no caso das Anomalias de Tradução, criando uma nova Anomalia com a *checkbox* "Anomalia de Redação" não selecionada). A página é atualizada depois de cada adição/remoção de alguma entrada. Optou-se por guardar as imagens como forma de referência (em string), sendo que a imagem de exemplo para todas as anomalias está guardada localmente (`image.jpeg`), funcionando também com links para imagens online, estando apresentada efetivamente como imagem na aplicação e não como string. Importante notar que no formulário de inserção de uma nova anomalia, o campo "Zona" deve ter o formato "(x1,y1),(x2,y2)" (sem as aspas) e o campo "Time stamp" deve ter o formato "AAAA-MM-DD HH:MinMin:SS" (sem as aspas).

A segunda página (`b.php`) contém duas tabelas, correspondentes às **Correções** e às **Propostas de Correção**. É possível apagar entradas de cada tabela de forma análoga à primeira página. Quanto à inserção, esta também é feita de forma equivalente, embora com algumas nuances. Nomeadamente, quando se pretende inserir uma Proposta de Correção, o formulário contém um campo "Id da anomalia da correção correspondente" – quando se insere uma proposta, é também inserida uma correção com anomalia_id igual ao id inserido, visto que cada proposta de correção tem de ter pelo menos uma correção associada. A inserção de correções serve para associar mais correções a uma proposta já existente. Quando se apagam todas as correções associadas a uma determinada proposta, esta última é também removida. Edições de entradas destas tabelas estão limitadas à alteração do e-mail e do texto associado a uma proposta de correção, sendo o parâmetro "nro" atualizado automaticamente (não se permite alterar a data pois considera-se que esta corresponde à data de criação).

Na terceira página (`c.php`), são listados os *e-mails* de todos os **Utilizadores** presentes na base de dados, por ordem alfabética, bem como duas tabelas adicionais que listam os e-mails dos utilizadores separados nas duas categorias: **Utilizadores Regulares** e **Utilizadores Qualificados**.

Na quarta página (`d.php`), são apresentadas as tabelas referentes às **Incidências** e aos **Duplicados**. É possível adicionar novas entradas a cada uma das tabelas de forma análoga ao que foi explicado para a primeira página.

A quinta página (`e.php`) redireciona o utilizador para um formulário, que depois de preenchido apresenta os resultados pretendidos, ou seja, lista todas as **Anomalias** de **Incidências** registadas na área compreendida entre os dois **Locais Públicos** introduzidos.

Finalmente, a última página (`f.php`) funciona de forma semelhante à página `e.php`, listando as **Anomalias** registadas nos últimos três meses a mais ou menos **(dX,dY)** graus de **(lat,long)**, sendo dX, dY, lat e long valores introduzidos pelo utilizador.

Em relação a detalhes de implementação, optou-se for usar modificações aos *links* de cada página, em conjunto com formulários, como forma de obter a ação pretendida pelo utilizador (adicionar, remover, editar…) e os dados que se pretendem inserir, para que estas possam ser acessadas como valores do array `$_GET`. Garante-se a atomicidade de operações complexas que envolvam mais do que uma *query* usando *transactions*. Além disso, a base de dados está prevenida contra ataques *SQL INJECTION* devido ao uso de *prepared statements* para todas as *queries* executadas.

As páginas podem ser testadas usando um servidor local PHP (`php -S localhost:8000`).