Projeto de Base de Dados, Parte 3

Grupo 14

Turno de Sexta Feira, 8h30

Docente: Paulo Carreira

Gonçalo Velhinho	90718	21 horas	39%
Stefano Gonçalves	87706	16 horas	30%
Diogo Dias	90792	17 horas	31%

Criação da Base de Dados

Para criar a base de dados utilizar no psql o comando “\i schema.sql”. O ficheiro query.sql contém as queries especificadas abaixo. O ficheiro populate.sql possui dados de teste para correr as queries.

Os comandos para a criação da Base de Dados são os seguintes:

```
CREATE TABLE public_location (
    latitude numeric(9, 6),
    longitude numeric(8, 6),
    location_name varchar(255) NOT NULL,

    PRIMARY KEY (latitude, longitude)
);

CREATE TABLE item (
    id SERIAL,
    description_text text NOT NULL,
    location_name varchar(255) NOT NULL,
    latitude numeric(9, 6),
    longitude numeric (8, 6),

    PRIMARY KEY (id),
    FOREIGN KEY (latitude, longitude)
        REFERENCES public_location(latitude, longitude)
);

CREATE TABLE anomaly (
    id SERIAL,
    area varchar(45) NOT NULL,
    image_path varchar(253) NOT NULL,
    lang char(3) NOT NULL,
    tmstmp timestamp NOT NULL,
    description_text text NOT NULL,
    has_wording_anomaly boolean NOT NULL,

    PRIMARY KEY (id)
);

CREATE TABLE translation_anomaly (
    id SERIAL,
    area2 varchar(45) NOT NULL,
    lang2 char(3) NOT NULL,

    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES anomaly(id)
);

CREATE TABLE duplicate (
    item1 SERIAL,
    item2 SERIAL,

    PRIMARY KEY (item1, item2),
    FOREIGN KEY (item1) REFERENCES item(id),
    FOREIGN KEY (item2) REFERENCES item(id),
    CONSTRAINT self_duplicate CHECK(item1 < item2)
);

CREATE TABLE user_table (
    user_email varchar(254),
    user_password varchar(254) NOT NULL,

    PRIMARY KEY (user_email)
);

CREATE TABLE qualified_user (
    user_email varchar(254),

    PRIMARY KEY (user_email),
    FOREIGN KEY (user_email) REFERENCES user_table(user_email)
);

CREATE TABLE regular_user (
    user_email varchar(254),

    PRIMARY KEY (user_email),
    FOREIGN KEY (user_email) REFERENCES user_table(user_email)
);

CREATE TABLE incident (
    anomaly_id SERIAL,
    item_id SERIAL,
    user_email varchar(254),

    PRIMARY KEY (anomaly_id),
    FOREIGN KEY (anomaly_id) REFERENCES anomaly(id),
    FOREIGN KEY (item_id) REFERENCES item(id),
    FOREIGN KEY (user_email) REFERENCES user_table(user_email)
);

CREATE TABLE correction_proposal (
    user_email varchar(254),
    tmstmp timestamp NOT NULL,
    correction_text text NOT NULL,
    nro SERIAL,

    UNIQUE(nro),
    PRIMARY KEY (user_email, nro),
    FOREIGN KEY (user_email) REFERENCES qualified_user(user_email)
);

CREATE TABLE correction (
    user_email varchar(254),
    anomaly_id serial,
    nro SERIAL,

    PRIMARY KEY (user_email, nro, anomaly_id),
    FOREIGN KEY (user_email, nro) REFERENCES correction_proposal(user_email, nro),
    FOREIGN KEY (anomaly_id) REFERENCES incident(anomaly_id)
);
```

SQL

1.

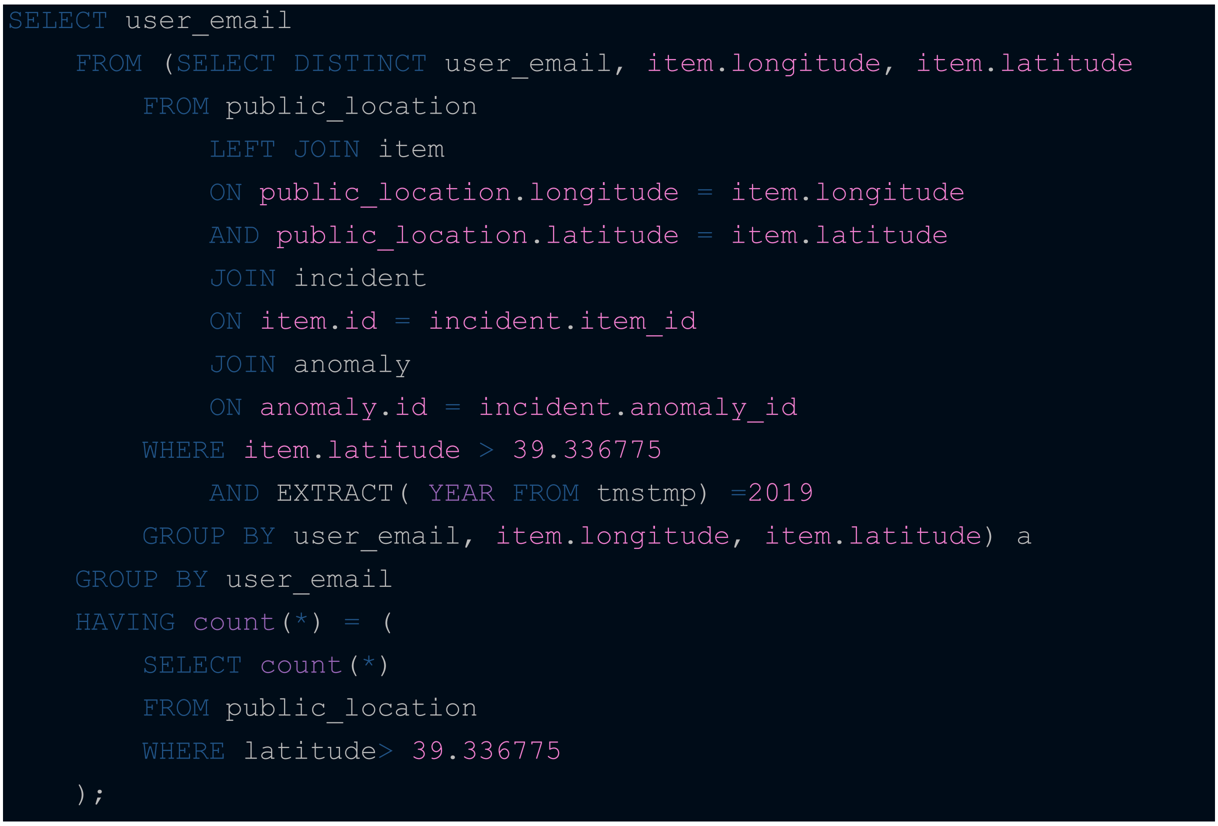
```
SELECT public_location.location_name, count(incident.anomaly_id) AS anomaly_count
    FROM public_location
    INNER JOIN item
        ON item.latitude = public_location.latitude
        AND item.longitude = public_location.longitude
    INNER JOIN incident
        ON incident.item_id = item.id
    GROUP BY public_location.location_name
    HAVING count(incident.anomaly_id) >= all (
        SELECT count(incident.anomaly_id) AS anomaly_count
        FROM public_location
        INNER JOIN item
            ON item.latitude = public_location.latitude
            AND item.longitude = public_location.longitude
        INNER JOIN incident
            ON incident.item_id = item.id
        GROUP BY public_location.location_name
    );
```

2.

```
SELECT regular_user.user_email, count(anomaly.id) AS anomaly_count
            FROM anomaly
            INNER JOIN translation_anomaly
                        ON anomaly.id = translation_anomaly.id
            INNER JOIN incident
                        ON incident.anomaly_id = anomaly.id
```

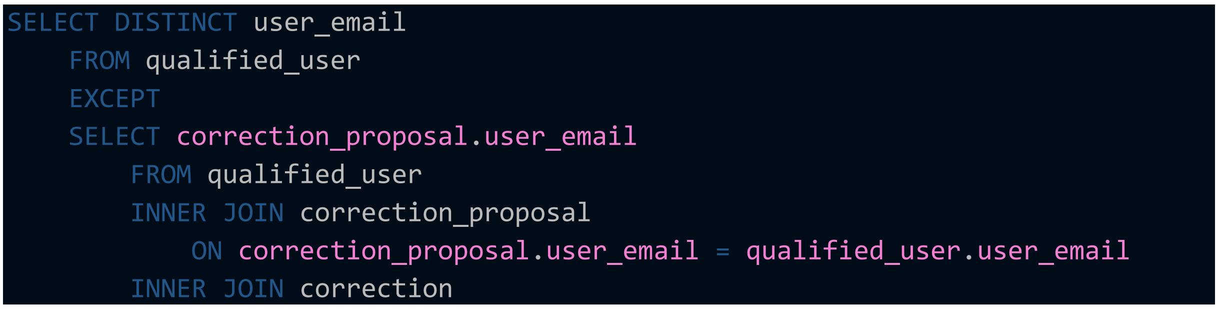
```
INNER JOIN regular_user
        ON incident.user_email = regular_user.user_email
WHERE tmstmp BETWEEN '2019-01-01' AND '2019-06-01'
GROUP BY regular_user.user_email
HAVING count(anomaly.id) >= all (
SELECT count(anomaly.id) AS anomaly_count
        FROM anomaly
        INNER JOIN translation_anomaly
                ON anomaly.id = translation_anomaly.id
        INNER JOIN incident
                ON incident.anomaly_id = anomaly.id
        INNER JOIN regular_user
                ON incident.user_email = regular_user.user_email
        WHERE tmstmp BETWEEN '2019-01-01 0:00:00' AND '2019-06-01 0:00:00'
        GROUP BY regular_user.user_email);
```

3.

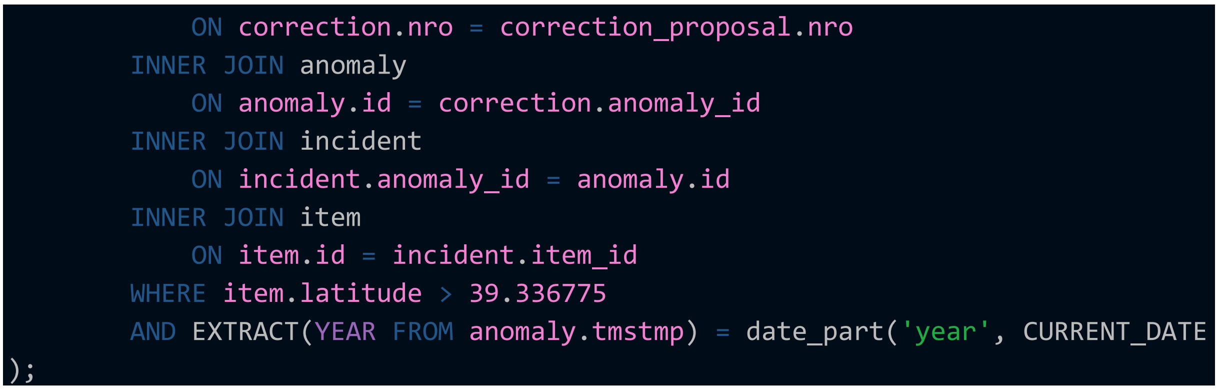


```
SELECT user_email
    FROM (SELECT DISTINCT user_email, item.longitude, item.latitude
        FROM public_location
            LEFT JOIN item
            ON public_location.longitude = item.longitude
            AND public_location.latitude = item.latitude
            JOIN incident
            ON item.id = incident.item_id
            JOIN anomaly
            ON anomaly.id = incident.anomaly_id
        WHERE item.latitude > 39.336775
            AND EXTRACT( YEAR FROM tmstmp) =2019
        GROUP BY user_email, item.longitude, item.latitude) a
    GROUP BY user_email
    HAVING count(*) = (
        SELECT count(*)
        FROM public_location
        WHERE latitude> 39.336775
    );
```

4)



```
SELECT DISTINCT user_email
    FROM qualified_user
    EXCEPT
    SELECT correction_proposal.user_email
        FROM qualified_user
        INNER JOIN correction_proposal
            ON correction_proposal.user_email = qualified_user.user_email
        INNER JOIN correction
```



```
            ON correction.nro = correction_proposal.nro
        INNER JOIN anomaly
            ON anomaly.id = correction.anomaly_id
        INNER JOIN incident
            ON incident.anomaly_id = anomaly.id
        INNER JOIN item
            ON item.id = incident.item_id
        WHERE item.latitude > 39.336775
        AND EXTRACT(YEAR FROM anomaly.tmstmp) = date_part('year', CURRENT_DATE
);
```

Desenvolvimento da Aplicação

O ficheiro index.html contém botões em que cada botão corresponde a uma alínea da secção Desenvolvimento da Aplicação, e.g. “first exercise” corresponde a alínea a).

O primeiro exercício contém três instruções diferentes. Por isso contém uma página principal para inserir os valores, mas depois cada instrução de “insert” tem uma página de PHP respetiva.