A6: Indexes, triggers, user functions and population

1. Database workload

A study of the predicted system load (database load), organized in subsections.

1.1 Tuple Estimation

Relation Reference	Relation Name	Order of magnitude	Estimated growth
R01	user	hundreds	units per day
R02	event	hundreds	units per day
R03	localization	hundreds	units per day
R04	image	hundreds	units per day
R05	city	tens	units per week
R06	country	tens	units per month
R07	post	hundreds	dozens per day
R08	poll	tens	units per day
R09	option	tens	units per day
R10	friendRequest	hundreds	dozens per day
R11	friendActivity	hundreds	dozens per day
R12	eventInvite	hundreds	dozens per day
R13	owner	tens	units per day
R14	participant	hundreds	units per day
R15	done	hundreds	units per day
R16	notDone	hundreds	units per day
R17	admin	units	no growth
R18	friendship	tens	units per day
R19	eventDeleteWarning	tens	units per month
R20	eventUpdateWarning	hundreds	units per day
R21	currentDate	units	hundreds per day
R22	rating	tens	units per day

1.2 Most frequent queries

Querie reference	SELECT01
Querie description	User's information
Querie frequency	Hundreds per day

```
SELECT username, last_name, first_name, email, image_path, city_id
  FROM users
  WHERE users.id = $user_id;
```

Querie reference	SELECT02
Querie description	Event's information
Querie frequency	Hundreds per day

```
SELECT events.id, events.name, events.category,
events.description, events."date", users.username
  FROM events, users
  WHERE events.owner_id = users.id AND events.id = $event_id;

SELECT posts.description, posts.id, posts.image_path, posts.user_id
  FROM posts,events
  WHERE posts.event_id = $event_id;

SELECT users.username, users.image_path
  FROM participants
  WHERE users.id = participants.user_id AND
participants.event_id=$event_id;
```

Querie reference	SELECT03
Querie description	Search event
Querie frequency	Hundreds per day

```
SELECT id, "name", "date", localization, category
FROM events
  WHERE "name" LIKE %$search% OR localization LIKE %$search% AND
event_type = 'public'
  ORDER BY "name";
```

Querie reference	SELECT04
Querie description	Search user
Querie frequency	Hundreds per day

```
SELECT id, username, image_path
FROM users
  WHERE username LIKE %$search%
  ORDER BY username;
```

Querie reference	SELECT05
Querie description	Search by categories
Querie frequency	Hundreds per day

```
SELECT id, "name", "date", localization, category
FROM events
  WHERE "name" LIKE %$search% OR localization LIKE %$search% AND
event_type = 'public' AND events.category LIKE %$categories%
  ORDER BY "name";
```

Querie reference	SELECT06
Querie description	User's notifications
Querie frequency	Hundreds per day

```
SELECT sender_id
  FROM friend_requests
  WHERE receiver_id = $user_id;

SELECT sender_id, event_id
  FROM friend_activities
  WHERE receiver_id = $user_id;

SELECT owner_id, event_id
  FROM event_invites
  WHERE receiver_id = $user_id;

SELECT event_name
FROM event_delete_warnings
WHERE receiver_id = $user_id;

SELECT event_id
FROM event_update_warnings
WHERE receiver_id = $user_id;
```

Querie reference	SELECT07
Querie description	Owner of a specific event
Querie frequency	Hundreds per day

```
SELECT event_id
  FROM owners
  WHERE user_id = $user_id;
```

Querie reference	SELECT08
Querie description	User's events
Querie frequency	Hundreds per day

```
SELECT event_id
  FROM participants
  WHERE user_id = $user_id;
```

Querie reference	SELECT09
Querie description	User's friends
Querie frequency	Hundreds per day

```
SELECT user_id_1, user_id_2
  FROM friendships
  WHERE user_id_1 = $user_id OR user_id_2 = $user_id;
```

Querie reference	SELECT010
Querie description	Who can be invited to an event
Querie frequency	Hundreds per day

```
SELECT users.username
  FROM users, events
  WHERE users.id!= event.owner_id AND users.id  NOT IN (
    SELECT user_id
    FROM participants
    WHERE user_id IS NOT NULL AND participants.event_id=$event_id) ;
```

Querie reference	SELECT11
Querie description	Rating of a past event
Querie frequency	Hundreds per day

```
SELECT rating
  FROM dones
  WHERE dones.event_id= $event_id;
```

1.3 Most frequent modifications

Querie reference	UPDATE01
Querie description	Update users information
Querie frequency	Hundreds per month

```
UPDATE "users"
SET password = $password,
    email = $email,
    first_name = $first_name,
    last_name = $last_name,
    image_path = $image_path,
    city_id = $city_id
WHERE id = $id;
```

Querie reference	UPDATE02
Querie description	Update events information
Querie frequency	Hundreds per month

```
UPDATE events
SET name = $name,
    date = $date,
    description = $description,
    localization_id = $localization_id,
    event_type = $event_type,
    category = $category
WHERE id = $id;
```

Querie reference	UPDATE03
Querie description	Update options description
Querie frequency	Tens per month

```
UPDATE options
SET description = $description
WHERE id = $id;
```

Querie reference	UPDATE04
Querie description	Update posts information
Querie frequency	Hundreds per month

```
UPDATE posts
SET description = $description,
    date = $date,
    image_path = $image_path
WHERE id = $id;
```

Querie reference	UPDATE05
Querie description	Update current_date
Querie frequency	Thousands per month

```
UPDATE "current_date"
SET date = $date
WHERE id = $id;
```

Querie reference	INSERT01
Querie description	Regist new user
Querie frequency	Tens per month

```
INSERT INTO users
 (username,password,email,regist_date,first_name,last_name,
image_path,city_id)
           VALUES ($username, $password, $email, $regist_date,
$first_name, $last_name, $image_path, $city_id);
```

Querie reference	INSERT02
Querie description	Create new event
Querie frequency	Tens per month

```
INSERT INTO events
(name,date,description,owner_id,localization_id,type,category)
  VALUES ($name,$date,$description,$owner_id,$localization_id,
$type,$category);
```

Querie reference	INSERT03
Querie description	Create new post
Querie frequency	Tens per month

```
INSERT INTO posts (description,date,event_id, user_id, image_path)
 VALUES ($description,$date,$event_id, $user_id, $image_path);
```

Querie reference	INSERT04
Querie description	Create new event done
Querie frequency	Tens per month

```
INSERT INTO dones (event_id)
            VALUES ($event_id);
```

Querie reference	INSERT05
Querie description	Create new event not_done
Querie frequency	Tens per month

```
INSERT INTO dones (event_id)
            VALUES ($event);
```

Querie reference	INSERT06
Querie description	Create new participant
Querie frequency	Tens per month

```
INSERT INTO participants (user_id,event_id)
            VALUES ($user_id,$event_id);
```

Querie reference	INSERT07
Querie description	Create new owner
Querie frequency	Units per month

```
INSERT INTO owners (user_id,event_id)
            VALUES ($user_id,$event_id);
```

Querie reference	INSERT08
Querie description	Create new image
Querie frequency	Tens per month

```
INSERT INTO images (event_id, path) VALUES ($event_id, $path);
```

Querie reference	INSERT09
Querie description	Create new localization
Querie frequency	Units per month

```
 INSERT INTO localizations (name,address,latitude,longitude,city_id)
 VALUES ($name,$address,$latitude,$longitude,$city_id);
```

Querie reference	INSERT10
Querie description	Create new city
Querie frequency	Units per month

```
INSERT INTO cities (name,country_id) VALUES ($name,$country_id);
```

Querie reference	INSERT11
Querie description	Create new country
Querie frequency	Units per month

```
  INSERT INTO countries (name) VALUES ($name);
```

Querie reference	INSERT12
Querie description	Create new poll
Querie frequency	Units per month

```
INSERT INTO polls (post_id) VALUES ($posts_id);
```

Querie reference	INSERT13
Querie description	Create new option
Querie frequency	Tens per month

```
INSERT INTO options (description,poll_id) VALUES
($description,$poll_id);
```

Querie reference	INSERT14
Querie description	Create new friend request
Querie frequency	Tens per month

```
INSERT INTO friend_requests (sender_id,receiver_id)
 VALUES ($sender_id,$receiver_id);
```

Querie reference	INSERT15
Querie description	Create new friend activity
Querie frequency	Tens per month

```
INSERT INTO friend_activities (sender_id,receiver_id,event_id)
VALUES ($sender_id,$receiver_id,$event_id);
```

Querie reference	INSERT16
Querie description	Create new event invite
Querie frequency	Tens per month

```
INSERT INTO event_invites (event_id,owner_id,receiver_id)
 VALUES ($event_id,$owner_id,$receiver_id);
```

Querie reference	INSERT17
Querie description	Create new friendship
Querie frequency	Tens per month

```
INSERT INTO friendships (user_id_1, user_id_2)
VALUES ($user_id_1, $user_id_2);
```

Querie reference	DELETE01
Querie description	Delete an user
Querie frequency	Units per month

```
DELETE FROM "users"
WHERE id = $id;
```

Querie reference	DELETE02
Querie description	Delete an event
Querie frequency	Units per month

```
DELETE FROM events
WHERE id = $id;
```

Querie reference	DELETE03
Querie description	Delete a post
Querie frequency	Units per month

```
DELETE FROM posts
WHERE id = $id;
```

Querie reference	DELETE04
Querie description	Delete an option
Querie frequency	Tens per month

```
DELETE FROM options
WHERE id = $id;
```

Querie reference	DELETE05
Querie description	Delete a poll
Querie frequency	Units per month

```
DELETE FROM polls
WHERE id = $id;
```

Querie reference	DELETE06
Querie description	Delete a friendship
Querie frequency	Units per month

```
DELETE FROM friendships
WHERE id = $id;
```

Querie reference	DELETE07
Querie description	Delete a friend_request
Querie frequency	Tens per month

```
DELETE FROM friend_requests
WHERE id = $id;
```

Querie reference	DELETE08
Querie description	Delete an event_invite
Querie frequency	Tens per month

```
DELETE FROM event_invites
WHERE id = $id;
```

2. Proposed indexes

2.1 Performance indexes

Index reference	IDX01
Related queries	SELECT01
Index relation	user
Index atribute	username
Index type	Hash
Cardinality	High
Clustering	No
Justification	Query SELECT01 has to be fast as it is executed many times; doesn't need range query support; cardinality is high because email is an unique key; it's not a good candidate for clustering.

```
CREATE INDEX user_username ON users USING hash (username);
```

Index reference	IDX02
Related queries	SELECT02
Index relation	events
Index atribute	owner_id
Index type	Hash
Cardinality	high
Clustering	No
Justification	Query has to be fast as it is executed many times; doesn't need range query support; cardinality is high because owner_id is an unique key; it's not a good candidate for clustering.

```
CREATE INDEX owner_events ON events USING hash(owner_id);
```

2.2 Full-text search indexes

Index reference	IDX03
Related queries	SELECT03
Index relation	event
Index atribute	name
Index type	GIST
Clustering	No
Justification	To improve the performance of full text searches while searching for events and their names; GiST because it's better for dynamic data.

```
CREATE INDEX search_events ON events USING GIST (to_tsvector('english', name));
```

3. Triggers

Trigger reference	TRIGGER01
Trigger description	When the current_date is updated, it is verified if it is greater than the event_date. In that case, the event is added to the dones table and removed from the not_dones.

```
CREATE OR REPLACE FUNCTION set_event_as_done() RETURNS TRIGGER AS
$BODY$
BEGIN
  IF EXISTS (SELECT event_id FROM not_done WHERE NEW.event_id = id)
  THEN
    INSERT INTO dones VALUES (NEW.event_id, NULL);
    DELETE FROM not_dones WHERE id = NEW.event_id;
  END IF;
  RETURN NEW;
END
$BODY$
LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS set_event_as_done ON "current_date";
CREATE TRIGGER set_event_as_done
  BEFORE UPDATE OF date ON "current_date"
  FOR EACH ROW
  WHEN NEW.date = GETDATE()
    EXECUTE PROCEDURE set_event_as_done();
```

Trigger reference	TRIGGER02
Trigger description	When an event is deleted a notification is sent to the participants of that event.

```
CREATE OR REPLACE FUNCTION notificate_event_delete() RETURNS TRIGGER AS
$BODY$
DECLARE
    idx int;
BEGIN
    FOR idx IN SELECT id FROM participants WHERE participants.event_id
= OLD.id
    LOOP
        INSERT INTO event_delete_warnings (event_name, receiver_id)
VALUES (OLD.name, idx);
    END LOOP;
    RETURN OLD;
END;
$BODY$
LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS notificate_event_delete ON "events";
CREATE TRIGGER notificate_event_delete
  BEFORE DELETE ON events
  FOR EACH ROW
    EXECUTE PROCEDURE notificate_event_delete();
```

Trigger reference	TRIGGER03
Trigger description	When an event is updated a notification is sent to the participants of that event.

```
CREATE OR REPLACE FUNCTION notificate_event_update() RETURNS TRIGGER AS
$BODY$
DECLARE
    idx int;
BEGIN
    FOR idx IN SELECT id FROM participants WHERE participants.event_id
= OLD.id
    LOOP
        INSERT INTO event_update_warnings (event_id, receiver_id)
VALUES (OLD.id, idx);
    END LOOP;
    RETURN OLD;
END;
$BODY$
LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS notificate_event_update ON "events";
CREATE TRIGGER notificate_event_update
  BEFORE DELETE ON events
  FOR EACH ROW
    EXECUTE PROCEDURE notificate_event_update();
```

Trigger reference	TRIGGER04
Trigger description	When a rating is inserted, the dones table is updated with the new rating average for that event.

```
CREATE OR REPLACE FUNCTION rating_update() RETURNS TRIGGER AS
$BODY$
BEGIN
  UPDATE dones SET rating = (SELECT AVG("value") FROM ratings WHERE
New.event_id = event_id) WHERE event_id = New.event_id;
  RETURN NULL;
END
$BODY$

LANGUAGE plpgsql;

CREATE TRIGGER rating_update
  AFTER INSERT ON ratings
  FOR EACH ROW
    EXECUTE PROCEDURE rating_update();
```

Trigger reference	TRIGGER05
Trigger description	When a friend request is accepted, a new friendship is added.

```
CREATE OR REPLACE FUNCTION accept_friend_request() RETURNS TRIGGER AS
$BODY$
BEGIN
  IF New.answer = 'yes'
  THEN
    INSERT INTO friendships (user_id_1, user_id_2) VALUES
(New.sender_id, New.receiver_id);
  END IF;
  RETURN NULL;
END
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER accept_friend_request
  AFTER UPDATE ON friend_requests
  FOR EACH ROW
    EXECUTE PROCEDURE accept_friend_request();
```

Trigger reference	TRIGGER06
Trigger description	When an event invite is accepted, the user that had been invited is added to the participants of that event.

```
CREATE OR REPLACE FUNCTION accept_event_invite() RETURNS TRIGGER AS
$BODY$
BEGIN
  IF New.answer = 'yes'
  THEN
    INSERT INTO participants (user_id, event_id) VALUES
(New.receiver_id, New.event_id);
  END IF;
  RETURN NULL;
END
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER accept_event_invite
  AFTER UPDATE ON event_invites
  FOR EACH ROW
    EXECUTE PROCEDURE accept_event_invite();
```

4. SQL Code

Lbaw1765db.sql

Inserts.sql

```sql
--Types Enums

DROP TYPE IF EXISTS categories CASCADE;
CREATE TYPE categories AS ENUM(
    'Music',
    'Sports',
    'Entertainment',
    'Educational',
    'Business',
    'Other'
);
DROP TYPE IF EXISTS types_of_event CASCADE;
CREATE TYPE types_of_event AS ENUM(
    'Public',
    'Private'
);

--Tables
DROP TABLE IF EXISTS admins CASCADE;
CREATE TABLE admins (
    id SERIAL NOT NULL,
    username text NOT NULL,
    password text NOT NULL,
    email text NOT NULL,
    CONSTRAINT admins_pk PRIMARY KEY (id),
    CONSTRAINT admins_email_uk UNIQUE (email)
);

DROP TABLE IF EXISTS cities CASCADE;
CREATE TABLE cities (
    id SERIAL NOT NULL,
    name text NOT NULL,
    country_id INTEGER NOT NULL,
    CONSTRAINT cities_pk PRIMARY KEY (id),
    CONSTRAINT cities_name_uk UNIQUE (name)

);
```

```sql
DROP TABLE IF EXISTS "current_date" CASCADE;
CREATE TABLE "current_date" (
    id SERIAL NOT NULL,
    date TIMESTAMP WITH TIME zone NOT NULL,
    CONSTRAINT current_date_pk PRIMARY KEY (id)
);

DROP TABLE IF EXISTS countries CASCADE;
CREATE TABLE countries (
    id SERIAL NOT NULL,
    name text NOT NULL,
    CONSTRAINT countries_pk PRIMARY KEY (id),
    CONSTRAINT countries_name_uk UNIQUE (name)
);

DROP TABLE IF EXISTS dones CASCADE;
CREATE TABLE dones (
    event_id INTEGER NOT NULL,
    rating FLOAT,
    CONSTRAINT dones_pk PRIMARY KEY (event_id),
    CONSTRAINT rating_ck CHECK (((rating >= 1) AND (rating <= 5)))
);

DROP TABLE IF EXISTS events CASCADE;
CREATE TABLE events (
    id SERIAL NOT NULL,
    name text NOT NULL,
    date TIMESTAMP WITH TIME zone NOT NULL,
    description text NOT NULL,
    owner_id INTEGER NOT NULL,
    localization_id INTEGER,
    type types_of_event NOT NULL,
    category categories NOT NULL,
    CONSTRAINT events_pk PRIMARY KEY (id),
    CONSTRAINT date_ck CHECK ((date > now()))
);

DROP TABLE IF EXISTS event_invites CASCADE;
CREATE TABLE event_invites (
    id SERIAL NOT NULL,
    answer text,
    event_id INTEGER NOT NULL,
    owner_id INTEGER,
    receiver_id INTEGER NOT NULL,
    CONSTRAINT event_invites_pk PRIMARY KEY (id)
);

DROP TABLE IF EXISTS event_delete_warnings CASCADE;
CREATE TABLE event_delete_warnings (
```

```
    id SERIAL NOT NULL,
    event_name text NOT NULL,
    receiver_id INTEGER NOT NULL,
    CONSTRAINT event_delete_warnings_pk PRIMARY KEY (id)
);

DROP TABLE IF EXISTS event_update_warnings CASCADE;
CREATE TABLE event_update_warnings (
    id SERIAL NOT NULL,
    event_id INTEGER NOT NULL,
    receiver_id INTEGER NOT NULL,
    CONSTRAINT event_update_warnings_pk PRIMARY KEY (id),
    CONSTRAINT event_update_warnings_event_id_fk FOREIGN KEY (event_id)
REFERENCES
    events(id) ON DELETE CASCADE
);

DROP TABLE IF EXISTS friend_activities CASCADE;
CREATE TABLE friend_activities (
    id SERIAL NOT NULL,
    sender_id INTEGER NOT NULL,
    receiver_id INTEGER NOT NULL,
    event_id INTEGER NOT NULL,
    CONSTRAINT friend_activities_pk PRIMARY KEY (id),
    CONSTRAINT friend_activities_event_id_fk FOREIGN KEY (event_id)
REFERENCES
    events(id) ON DELETE CASCADE
);

DROP TABLE IF EXISTS friend_requests CASCADE;
CREATE TABLE friend_requests (
    id SERIAL NOT NULL,
    answer text,
    sender_id INTEGER NOT NULL,
    receiver_id INTEGER NOT NULL,
    CONSTRAINT friend_requests_pk PRIMARY KEY (id)
);
DROP TABLE IF EXISTS friendships CASCADE;
CREATE TABLE friendships (
    id SERIAL NOT NULL,
    user_id_1 INTEGER NOT NULL,
    user_id_2 INTEGER NOT NULL,
    CONSTRAINT friendships_users_ids_uk UNIQUE (user_id_1, user_id_2)
);

DROP TABLE IF EXISTS images CASCADE;
CREATE TABLE images (
    id SERIAL NOT NULL,
    event_id INTEGER NOT NULL,
```

```
    path text NOT NULL,
    CONSTRAINT images_pk PRIMARY KEY (id),
    CONSTRAINT images_path_uk UNIQUE (path)
);

DROP TABLE IF EXISTS localizations CASCADE;
CREATE TABLE localizations (
    id SERIAL NOT NULL,
    name text NOT NULL,
    address text NOT NULL,
    latitude FLOAT,
    longitude FLOAT,
    city_id INTEGER NOT NULL,
    CONSTRAINT localizations_pk PRIMARY KEY (id),
    CONSTRAINT localizations_city_id_fk FOREIGN KEY (city_id) REFERENCES
    cities(id) ON DELETE SET NULL
);

DROP TABLE IF EXISTS not_dones CASCADE;
CREATE TABLE not_dones (
    event_id INTEGER NOT NULL,
    CONSTRAINT not_dones_pk PRIMARY KEY (event_id),
    CONSTRAINT not_dones_event_id_fk FOREIGN KEY (event_id) REFERENCES
    events(id) ON DELETE CASCADE
);

DROP TABLE IF EXISTS options CASCADE;
CREATE TABLE options (
    id SERIAL NOT NULL,
    description text NOT NULL,
    poll_id INTEGER NOT NULL,
    CONSTRAINT options_pk PRIMARY KEY (id)
);

DROP TABLE IF EXISTS owners CASCADE;
CREATE TABLE owners (
    id SERIAL NOT NULL,
    user_id INTEGER NOT NULL,
    event_id INTEGER NOT NULL,
    CONSTRAINT owners_pk PRIMARY KEY (id),
    CONSTRAINT owners_user_id_event_id_uk UNIQUE (user_id, event_id),
    CONSTRAINT owners_event_id_fk FOREIGN KEY (event_id) REFERENCES
    events(id) ON DELETE CASCADE
);

DROP TABLE IF EXISTS participants CASCADE;
CREATE TABLE participants (
    id SERIAL NOT NULL,
    user_id INTEGER NOT NULL,
```

```
    event_id INTEGER NOT NULL,
    CONSTRAINT participants_pk PRIMARY KEY (id),
    CONSTRAINT participants_user_id_event_id_uk UNIQUE (user_id,
event_id),
    CONSTRAINT participants_event_id_fk FOREIGN KEY (event_id) REFERENCES
    events(id) ON DELETE CASCADE
);

DROP TABLE IF EXISTS polls CASCADE;
CREATE TABLE polls (
    id SERIAL NOT NULL,
    post_id INTEGER NOT NULL,
    CONSTRAINT polls_pk PRIMARY KEY (id)
);

DROP TABLE IF EXISTS posts CASCADE;
CREATE TABLE posts (
    id SERIAL NOT NULL,
    description text NOT NULL,
    date TIMESTAMP WITH TIME zone NOT NULL,
    event_id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    image_path text NOT NULL,
    CONSTRAINT posts_pk PRIMARY KEY (id),
    CONSTRAINT posts_event_id_fk FOREIGN KEY (event_id) REFERENCES
    events(id) ON DELETE CASCADE
);

DROP TABLE IF EXISTS ratings CASCADE;
CREATE TABLE ratings (
    id SERIAL NOT NULL,
    "value" INTEGER NOT NULL,
    event_id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    CONSTRAINT ratings_pk PRIMARY KEY (id),
    CONSTRAINT ratings_user_id_event_id_uk UNIQUE (user_id, event_id),
    CONSTRAINT ratings_event_id_fk FOREIGN KEY (event_id) REFERENCES
    events(id) ON DELETE CASCADE
);

DROP TABLE IF EXISTS users CASCADE;
CREATE TABLE users (
    id SERIAL NOT NULL,
    username text NOT NULL,
    password text NOT NULL,
    email text NOT NULL,
    regist_date TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
    first_name text NOT NULL,
    last_name text NOT NULL,
```

```
    image_path text NOT NULL,
    city_id INTEGER,
    CONSTRAINT users_pk PRIMARY KEY (id),
    CONSTRAINT users_name_uk UNIQUE (username),
    CONSTRAINT users_email_uk UNIQUE (email),
    CONSTRAINT users_city_id_fk FOREIGN KEY (city_id) REFERENCES
    cities(id) ON DELETE SET NULL
);


ALTER TABLE ONLY cities
    ADD CONSTRAINT cities_country_id_fk FOREIGN KEY (country_id)
REFERENCES
    countries(id) ON DELETE SET NULL;

ALTER TABLE ONLY dones
    ADD CONSTRAINT dones_event_id_fk FOREIGN KEY (event_id) REFERENCES
     events(id) ON UPDATE CASCADE;

ALTER TABLE ONLY events
    ADD CONSTRAINT events_owner_id_fk FOREIGN KEY (owner_id) REFERENCES
    users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY events
    ADD CONSTRAINT events_localization_id_fk FOREIGN KEY
(localization_id) REFERENCES
    localizations(id) ON DELETE SET NULL;

ALTER TABLE ONLY event_invites
    ADD CONSTRAINT event_invites_event_id_fk FOREIGN KEY (event_id)
REFERENCES
    not_dones(event_id) ON DELETE CASCADE;

ALTER TABLE ONLY event_invites
    ADD CONSTRAINT event_invites_owner_id_fk FOREIGN KEY (owner_id)
REFERENCES
    owners(id) ON DELETE SET NULL;

ALTER TABLE ONLY event_invites
    ADD CONSTRAINT event_invites_receiver_id_fk FOREIGN KEY (receiver_id)
REFERENCES
    users(id) ON DELETE CASCADE;

ALTER TABLE ONLY event_delete_warnings
    ADD CONSTRAINT event_delete_warnings_receiver_id_fk FOREIGN KEY
(receiver_id) REFERENCES
    users(id) ON DELETE CASCADE;

ALTER TABLE ONLY event_update_warnings
```

```
    ADD CONSTRAINT event_update_warnings_receiver_id_fk FOREIGN KEY
(receiver_id) REFERENCES
    users(id) ON DELETE CASCADE;

ALTER TABLE ONLY friend_activities
    ADD CONSTRAINT friend_activities_sender_id_fk FOREIGN KEY (sender_id)
REFERENCES
    users(id) ON DELETE CASCADE;

ALTER TABLE ONLY friend_activities
    ADD CONSTRAINT friend_activities_receiver_id_fk FOREIGN KEY
(receiver_id) REFERENCES
    users(id) ON DELETE CASCADE;

ALTER TABLE ONLY friend_requests
    ADD CONSTRAINT friend_requests_sender_id_fk FOREIGN KEY (sender_id)
REFERENCES
    participants(id) ON DELETE CASCADE;

ALTER TABLE ONLY friend_requests
    ADD CONSTRAINT friend_requests_receiver_id_fk FOREIGN KEY
(receiver_id) REFERENCES
    users(id) ON DELETE CASCADE;

ALTER TABLE ONLY friendships
    ADD CONSTRAINT friendships_user_id_1 FOREIGN KEY (user_id_1)
REFERENCES
    users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY friendships
    ADD CONSTRAINT friendships_user_id_2 FOREIGN KEY (user_id_2)
REFERENCES
    users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY options
    ADD CONSTRAINT options_poll_id_fk FOREIGN KEY (poll_id) REFERENCES
    polls(id) ON DELETE CASCADE;

ALTER TABLE ONLY owners
    ADD CONSTRAINT owners_user_id_fk FOREIGN KEY (user_id) REFERENCES
    users(id) ON UPDATE CASCADE;


ALTER TABLE ONLY participants
    ADD CONSTRAINT participants_user_id_fk FOREIGN KEY (user_id)
REFERENCES
    users(id) ON UPDATE CASCADE;

ALTER TABLE ONLY polls
```

```
    ADD CONSTRAINT polls_post_id_fk FOREIGN KEY (post_id) REFERENCES
    posts(id) ON DELETE CASCADE;

ALTER TABLE ONLY posts
    ADD CONSTRAINT posts_user_id_fk FOREIGN KEY (user_id) REFERENCES
    users(id) ON DELETE CASCADE;

ALTER TABLE ONLY ratings
    ADD CONSTRAINT ratings_user_id_fk FOREIGN KEY (user_id) REFERENCES
    users(id) ON DELETE SET NULL;


CREATE OR REPLACE FUNCTION set_event_as_done() RETURNS TRIGGER AS
$BODY$
BEGIN
  IF EXISTS (SELECT event_id FROM not_done WHERE NEW.event_id = id)
  THEN
    INSERT INTO dones VALUES (NEW.event_id, NULL);
    DELETE FROM not_dones WHERE id = NEW.event_id;
  END IF;
  RETURN NEW;
END
$BODY$
LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS set_event_as_done ON "current_date";
CREATE TRIGGER set_event_as_done
  BEFORE UPDATE OF date ON "current_date"
  FOR EACH ROW
  WHEN NEW.date = GETDATE()
    EXECUTE PROCEDURE set_event_as_done();

CREATE OR REPLACE FUNCTION notificate_event_delete() RETURNS TRIGGER AS
$BODY$
DECLARE
    idx int;
BEGIN
    FOR idx IN SELECT id FROM participants WHERE participants.event_id =
OLD.id
    LOOP
        INSERT INTO event_delete_warnings (event_name, receiver_id)
VALUES (OLD.name, idx);
    END LOOP;
    RETURN OLD;
END;
$BODY$
LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS notificate_event_delete ON "events";
```

```
CREATE TRIGGER notificate_event_delete
  BEFORE DELETE ON events
  FOR EACH ROW
    EXECUTE PROCEDURE notificate_event_delete();

CREATE OR REPLACE FUNCTION notificate_event_update() RETURNS TRIGGER AS
$BODY$
DECLARE
    idx int;
BEGIN
    FOR idx IN SELECT id FROM participants WHERE participants.event_id =
OLD.id
    LOOP
        INSERT INTO event_update_warnings (event_id, receiver_id) VALUES
(OLD.id, idx);
    END LOOP;
    RETURN OLD;
END;
$BODY$
LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS notificate_event_update ON "events";
CREATE TRIGGER notificate_event_update
  BEFORE DELETE ON events
  FOR EACH ROW
    EXECUTE PROCEDURE notificate_event_update();

CREATE OR REPLACE FUNCTION rating_update() RETURNS TRIGGER AS
$BODY$
BEGIN
  UPDATE dones SET rating = (SELECT AVG("value") FROM ratings WHERE
New.event_id = event_id) WHERE event_id = New.event_id;
  RETURN NULL;
END
$BODY$

LANGUAGE plpgsql;

CREATE TRIGGER rating_update
  AFTER INSERT ON ratings
  FOR EACH ROW
    EXECUTE PROCEDURE rating_update();


CREATE OR REPLACE FUNCTION accept_friend_request() RETURNS TRIGGER AS
$BODY$
BEGIN
  IF New.answer = 'yes'
  THEN
```

```
    INSERT INTO friendships (user_id_1, user_id_2) VALUES (New.sender_id,
New.receiver_id);
  END IF;
  RETURN NULL;
END
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER accept_friend_request
  AFTER UPDATE ON friend_requests
  FOR EACH ROW
    EXECUTE PROCEDURE accept_friend_request();


CREATE OR REPLACE FUNCTION accept_event_invite() RETURNS TRIGGER AS
$BODY$
BEGIN
  IF New.answer = 'yes'
  THEN
    INSERT INTO participants (user_id, event_id) VALUES (New.receiver_id,
New.event_id);
  END IF;
  RETURN NULL;
END
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER accept_event_invite
  AFTER UPDATE ON event_invites
  FOR EACH ROW
    EXECUTE PROCEDURE accept_event_invite();



--> INDEXES

 CREATE INDEX user_username ON users USING hash (username);

 CREATE INDEX owner_events ON events USING hash(owner_id);

 CREATE INDEX search_events ON events USING GIST (to_tsvector('english',
name));
```

INSERTS to populate the database

```
--> INSERTS
```

```sql
-- Here goes the SQL code - INSERTS

INSERT INTO countries (name) VALUES ('Portugal');
INSERT INTO countries (name) VALUES ('Espanha');
INSERT INTO countries (name) VALUES ('USA');


INSERT INTO cities (name,country_id) VALUES ('Braga',1);
INSERT INTO cities (name,country_id) VALUES ('Porto',1);
INSERT INTO cities (name,country_id) VALUES ('Lisboa',1);

INSERT INTO localizations (name,address,latitude,longitude,city_id)
VALUES ('Restaurante O Pirata','Rua da Isabelinha',41.452993,-
8.5775364,1);
INSERT INTO localizations (name,address,latitude,longitude,city_id)
VALUES ('FEUP','Rua Roberto Frias',41.1779401,-8.5998763,2);
INSERT INTO localizations (name,address,latitude,longitude,city_id)
VALUES ('Parque da BelaVista','Av. Arlindo Vicente',38.7507558,-
9.1265431,3);
INSERT INTO localizations (name,address,latitude,longitude,city_id)
VALUES ('Passeio Maritimo de Alges','Alges',38.697318,-9.2375993,3);
INSERT INTO localizations (name,address,latitude,longitude,city_id)
VALUES ('Hotel Douro','Rua de Agramonte',41.1564707,-8.6288115,2);
INSERT INTO localizations (name,address,latitude,longitude,city_id)
VALUES ('Norte shopping','Matosinhos',41.1825143,-8.6803795,2);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('sodales','GUL95ZXR9EX','sodales.at@curae.co.uk',NOW(),'Zeph','Griffin',
'/imgs/natu.jpg',2);


INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('uso1','12345','aliquam.iaculis.lacus@amet.co.uk',NOW(),'Ben','Warren','
/imgs/natur.jpg',1);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('robin','pass123','amet.ante@faucibusleo.net',NOW(),'Robin','Wright','/i
mgs/natur.jpg',2);
```

```
INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('bar123','semper123','ut.dolor@gmail.com',NOW(),'Barry','Allen','/imgs/n
atur.jpg',3);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('reddevil','LBAW','Nulla@et.net',NOW(),'Andrew','Irons','/imgs/november.
jpg',2);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('rpedro10','lbaw1765','rpedro10@iol.pt',NOW(),'Rui','Araujo','/imgs/fer.
jpg',3);


INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('joss123','CKB15AAW5MM','ante@fleo.com',NOW(),'Joss','Stone','/imgs/natu
r.jpg',1);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('top123','SXT16TTW3MH','cursus.et@orciUt.co.uk',NOW(),'Chris','Harris','
/imgs/natur.jpg',1);


INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('roland1','ZYS24FHN5GR','eget.dictum@orciDonec.edu',NOW(),'Roland','Schi
tt','/imgs/natur.jpg',2);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
```

```
            VALUES
('david123','ZUS29FRTGVJ','amet@faucibusleo.net',NOW(),'David','Rose','/i
mgs/natur.jpg',2);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES ('catones',
'QQQ8EFHNGNR','amec.donec@faucibusleo.net',NOW(),'Carlos','Antonio','/img
s/november.jpg',2);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('emanem','ASDFGHJKL','donex@sapo.net',NOW(),'Raheem','Sterling','/imgs/n
atur.jpg',3);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('ufoExtis','ZXCVBNM','amet@iol.net',NOW(),'Delle','Alli','/imgs/pyr.jpg'
,2);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('ragnar','QAZWSXEDC','risus.In.mi@egestas.com',NOW(),'Thor','Ragnarok','
/imgs/fer.jpg',1);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('seth','QWERTYUIOP','aliquet.diam.Sed@tinciduntnibh.co.uk',NOW(),'Seth',
'Byers','/imgs/natu.jpg',2);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('edNorton','TYT71DOD7YN','scelerisque.scelerisque.dui@arcuiaculisenim.ca
',NOW(),'Ed','Norton','/imgs/natur.jpg',1);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
```

```
            VALUES
('Pacquiao','SXT16TTW3MH','magnis@cursuset.edu',NOW(),'Paky','Barret','/i
mgs/natur.jpg',3);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('steven','MPS10QPK6UE','arcu.Vestibulum@amet.org',NOW(),'Donovan','Steve
nson','/imgs/pyr.jpg',1);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('porter123','QIG24ZOK3EM','dui.nec@ultriciesadipiscing.co.uk',NOW(),'Por
ter','Osborn','/imgs/natu.jpg',2);

INSERT INTO users
(username,password,email,regist_date,first_name,last_name,
image_path,city_id)
            VALUES
('human','ZYG87WQA6FX','facilisis.magna.tellus@sociis.net',NOW(),'Hu','Ra
ndolphe','/imgs/pyr.jpg',2);


INSERT INTO events
(name,date,description,owner_id,localization_id,type,category)
            VALUES ('Antonys Birthday Party', '2018-12-04
12:30:19.000000', 'nunc ac mattis ornare, lectus',1,1,'Public','Sports');

INSERT INTO events
(name,date,description,owner_id,localization_id,type,category)
            VALUES ('ENEI','2018-04-24 12:30:19.000000','Nunc quis arcu
vel quam',2,2,'Public','Sports');

INSERT INTO events
(name,date,description,owner_id,localization_id,type,category)
            VALUES ('RockInRio','2018-06-04 12:30:19.000000','tempus eu,
ligula. Aenean euismod',3,3,'Public','Sports');

INSERT INTO events
(name,date,description,owner_id,localization_id,type,category)
            VALUES ('Nos Alive','2018-08-04 12:30:19.000000','dignissim
pharetra. Nam ac nulla.',3,4,'Public','Sports');

INSERT INTO events
(name,date,description,owner_id,localization_id,type,category)
```

```
            VALUES ('Christmas Dinner','2018-10-04
12:30:19.000000','dignissim pharetra. Nam ac
nulla.',4,5,'Public','Sports');

INSERT INTO events
(name,date,description,owner_id,localization_id,type,category)
            VALUES ('Mark Birthday Party','2018-04-13
12:30:19.000000','dignissim pharetra. Nam ac
nulla.',1,1,'Public','Sports');

INSERT INTO events
(name,date,description,owner_id,localization_id,type,category)
            VALUES ('WebSummit','2018-04-12 12:30:19.000000','lorem
lorem, luctus ut, pellentesque',6,6,'Public','Sports');

INSERT INTO events
(name,date,description,owner_id,localization_id,type,category)
            VALUES ('Ted Talk','2018-04-24 12:30:19.000000','sollicitudin
orci sem eget massa.',12,2,'Public','Sports');

INSERT INTO events
(name,date,description,owner_id,localization_id,type,category)
            VALUES ('Teaches Conference','2018-04-13
12:30:19.000000','dignissim pharetra. Nam ac
nulla.',1,1,'Private','Business');



INSERT INTO admins (username,password,email) VALUES
('admin1','password','sapo@iol.pt');


INSERT INTO images (event_id,path) VALUES
(1,'/imgs/natur.jpg');
INSERT INTO images (event_id,path) VALUES (2,'/imgs/natu.jpg');
INSERT INTO images (event_id,path) VALUES (3,'/imgs/pyr.jpg');
INSERT INTO images (event_id,path) VALUES
(4,'/imgs/november.jpg');
INSERT INTO images (event_id,path) VALUES (5,'/imgs/taj.jpg');
INSERT INTO images (event_id,path) VALUES (6,'/imgs/fer.jpg');
INSERT INTO images (event_id,path) VALUES (7,'/imgs/fa1.jpg');
INSERT INTO images (event_id,path) VALUES (8,'/imgs/fa2.jpg');




INSERT INTO dones (event_id)
            VALUES (2);
INSERT INTO dones (event_id)
```

```
            VALUES (7);

INSERT INTO not_dones (event_id)
            VALUES (1);
INSERT INTO not_dones (event_id)
            VALUES (3);
INSERT INTO not_dones (event_id)
            VALUES (4);
INSERT INTO not_dones (event_id)
            VALUES (5);
INSERT INTO not_dones (event_id)
            VALUES (6);
INSERT INTO not_dones (event_id)
            VALUES (8);
INSERT INTO not_dones (event_id)
            VALUES (9);


INSERT INTO ratings ( "value", event_id, user_id)
            VALUES (4, 2, 2);


INSERT INTO participants (user_id,event_id)
            VALUES (1,1);
INSERT INTO participants (user_id,event_id)
            VALUES (2,1);
INSERT INTO participants (user_id,event_id)
            VALUES (2,2);
INSERT INTO participants (user_id,event_id)
            VALUES (3,1);
INSERT INTO participants (user_id,event_id)
            VALUES (4,1);
INSERT INTO participants (user_id,event_id)
            VALUES (5,8);
INSERT INTO participants (user_id,event_id)
            VALUES (6,8);
INSERT INTO participants (user_id,event_id)
            VALUES (7,3);
INSERT INTO participants (user_id,event_id)
            VALUES (12,3);
INSERT INTO participants (user_id,event_id)
            VALUES (13,3);
INSERT INTO participants (user_id,event_id)
            VALUES (14,3);
INSERT INTO participants (user_id,event_id)
            VALUES (15,1);
INSERT INTO participants (user_id,event_id)
            VALUES (16,2);
INSERT INTO participants (user_id,event_id)
```

```
            VALUES (17,3);
INSERT INTO participants (user_id,event_id)
            VALUES (18,2);
INSERT INTO participants (user_id,event_id)
            VALUES (19,1);
INSERT INTO participants (user_id,event_id)
            VALUES (20,3);
INSERT INTO participants (user_id,event_id)
            VALUES (12,6);
INSERT INTO participants (user_id,event_id)
            VALUES (12,5);
INSERT INTO participants (user_id,event_id)
            VALUES (10,4);


INSERT INTO owners (user_id,event_id)
            VALUES (1,1);
INSERT INTO owners (user_id,event_id)
            VALUES (2,2);
INSERT INTO owners (user_id,event_id)
            VALUES (3,3);
INSERT INTO owners (user_id,event_id)
            VALUES (3,4);
INSERT INTO owners (user_id,event_id)
            VALUES (5,5);
INSERT INTO owners (user_id,event_id)
            VALUES (1,6);
INSERT INTO owners (user_id,event_id)
            VALUES (6,7);
INSERT INTO owners (user_id,event_id)
            VALUES (12,8);
INSERT INTO owners (user_id,event_id)
            VALUES (20,9);

INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-02-12
15:55:12',1,1,'/img/new.jpg');
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-02-12
15:55:12',1,3,'/img/new.jpg');
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-02-12
15:55:12',1,4,'/img/new.jpg');
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ',NOW(),1,16,'/img/new.jpg');
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-02-12
15:55:12',2,18,'/img/new.jpg');
```

```
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-02-12
15:55:12',4,10,'/img/panda.jpg');
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-02-12
15:55:12',3,17,'/img/panda.jpg');
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-02-12
15:55:12',3,14,'/img/panda.jpg');
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-02-12
15:55:12',4,5,'/img/panda.jpg');
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-01-12
15:55:12',2,16,'/img/panda.jpg');
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-02-12
15:55:12',3,12,'/img/sports.jpg');
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-02-12
15:55:12',1,15,'/img/sports.jpg');
INSERT INTO posts (description,date,event_id, user_id, image_path) VALUES
('Lorem ipsum dolor sit amet. ','2018-02-12
15:55:12',5,12,'/img/sports.jpg');

INSERT INTO polls (post_id) VALUES (1);
INSERT INTO polls (post_id) VALUES (2);
INSERT INTO polls (post_id) VALUES (3);
INSERT INTO polls (post_id) VALUES (4);

INSERT INTO options (description,poll_id) VALUES ('Bar',1);
INSERT INTO options (description,poll_id) VALUES ('Cafe',1);
INSERT INTO options (description,poll_id) VALUES ('Club',1);
INSERT INTO options (description,poll_id) VALUES ('Home',1);
INSERT INTO options (description,poll_id) VALUES ('12/05/2018',2);
INSERT INTO options (description,poll_id) VALUES ('13/05/2018',2);
INSERT INTO options (description,poll_id) VALUES ('Great',3);
INSERT INTO options (description,poll_id) VALUES ('Good',3);
INSERT INTO options (description,poll_id) VALUES ('Available',4);
INSERT INTO options (description,poll_id) VALUES ('Not Available',4);

INSERT INTO friend_requests (sender_id, receiver_id) VALUES (1, 2);
INSERT INTO friend_requests (sender_id, receiver_id) VALUES (3, 4);

INSERT INTO event_invites (event_id, owner_id, receiver_id) VALUES (1, 1,
10);
INSERT INTO event_invites (event_id, owner_id, receiver_id) VALUES (1, 1,
12);
```

Revision History

Changes made to the first submission:

1. Tuple estimation correction;
2. Added and corrected triggers;
3. Corrected tables size and position in the PDF;
4. Added and corrected some queries;

GRUPO1765, 11/04/2018

- Mariana Duarte Guimarães, up201307777@fe.up.pt
- Rui Emanuel Cabral de Almeida Quaresma, up201503005@fe.up.pt
- Rui Pedro Machado Araújo, up201403263@fe.up.pt
- Tiago Duarte Carvalho, up201504461@fe.up.pt