eHotels Project

CSI2132 - Databases I

Fall 2018

School of Electrical Engineering and Computer Science
University of Ottawa

Professor: Dr. Verena Kantere

Students:
Nicolas Paré 7084451
Jonathan Popowick-Bastien 8432984

Submission Date: April 7th, 2019

DBMS and Programming Languages

Our website runs on a Postgres database as was used in the Labs— the only exception is that we ran it exclusively on our local servers to avoid the various issues we experienced early on with university of Ottawa's server stability issues.

PHP was used for the server-side programming of the website to connect it to our Postgres database and our front-end. The front-end was built with a combination of HTML, CSS, Bootstrap and some Javascript for certain user experience features.

Steps to install

1. Install Postgres
2. Install PHP (this was tested on v7.0. Here's a link to help for install php here).
3. Install pgAdmin4
    a. Create a new database
        i. Name the database "csi2132_project"
    b. Create a new postgres user named "web" with password "webapp" in pgAdmin4.
    c. Go to the project submission package and find "allSQL_code.sql"
        i. Copy all contents inside "allSQL_code.sql" to clipboard.
        ii. Paste all contents of "allSQL_code.sql" to pgAdmin4 query tool
        iii. WARNING: The next step involves running a query which includes a "DROP Schema public".
        iv. Run the query to drop all existing tables in the public schema and create the database, which includes triggers, views and mock data.
4. Open terminal on your operating system

a. Once there, you need to change directory in to the location of the zipped file. (Refer to this guide to resolve issues on changing directory link).
b. Then, change directory one more time into the “CSI2132Project” folder.
c. Run the following command to initialize your php server: “php -s localhost:8080”.
d. Do not close this terminal for the duration of testing or it will not work.

5. Open a web browser
    a. Use “localhost:8080” in the URL bar to navigate to your php server.
    b. This will be redirected to the index.php file and can begin using our application.
6. Close web browser and terminal when finished.

Triggers and Queries:

Note: Below you will see Query 3 and 4 include both a Dynamic PHP query and a SQL example. After speaking with professor Kantere, we discussed that most of our queries were used in conjunction with PHP-- she requested us to post both the real query (dynamic php query), and an example in SQL.

1. SELECT *
   FROM hotel_room_capacity;

2. SELECT *
   FROM rooms_by_area;

3. **Dynamic PHP Query:**

```
SELECT *
FROM hotel_chain
WHERE hotel_chain_id = '{$_GET["line"]["hotel_chain_id"]}'";
```

Example in SQL:

```
SELECT *
FROM hotel_chain
WHERE hotel_chain_id = 1;
```

4. **Dynamic PHP Query:**

```
SELECT amenity
FROM room_amenities
WHERE room_number = {$_GET["room_number"]};
```

Example in SQL:

```
SELECT amenity
FROM hotel_chain
WHERE hotel_chain_id = 2;
```

Trigger 1:

```
CREATE TRIGGER add_hotel AFTER INSERT ON hotel
        FOR EACH ROW EXECUTE PROCEDURE increment_hotel_count();

CREATE OR REPLACE FUNCTION increment_hotel_count() RETURNS trigger AS $body$
BEGIN
   UPDATE hotel_chain SET number_of_hotels = number_of_hotels + 1
         WHERE hotel_chain_id = NEW.hotel_chain_id;
   RETURN NEW;
END;
$body$
language plpgsql;
```

Trigger 2:

```
CREATE TRIGGER add_archive AFTER INSERT ON booking
        FOR EACH ROW EXECUTE PROCEDURE insert_archive();

CREATE OR REPLACE FUNCTION insert_archive() RETURNS trigger AS
$body$
BEGIN
   INSERT INTO archive (Booking_ID, Time_Created, check_in_date, check_out_date,
Is_Renting, username, Is_Paid, Room_Number, Hotel_ID) VALUES (NEW.Booking_ID,
NEW.Time_Created, NEW.check_in_date, NEW.check_out_date, NEW.Is_Renting,
NEW.username, NEW.Is_Paid, NEW.Room_Number, NEW.Hotel_ID);
   RETURN NEW;
END;
$body$
language plpgsql;
```

Note for SQL Code Images Below:

*(the mock data is not included given it is 1700 lines long which would be excessive for a text document, but please refer to the allSQL_code.sql file for mock data)

```sql
DROP SCHEMA public CASCADE;
CREATE SCHEMA public;

CREATE TABLE Hotel_Chain (
    Hotel_Chain_ID SERIAL,
    hotel_chain_name VARCHAR(20) NOT NULL,
    Central_Office VARCHAR(50) NOT NULL,
    Number_Of_Hotels INTEGER DEFAULT 0,
    Contact_Email VARCHAR(50) NOT NULL,

    PRIMARY KEY (Hotel_Chain_ID)
);

CREATE TABLE HotelChain_PhoneNumbers (
    Hotel_Chain_ID INTEGER,
    Phone_Number NUMERIC(10),

    PRIMARY KEY (Hotel_Chain_ID, Phone_Number),
    FOREIGN KEY (Hotel_Chain_ID) REFERENCES hotel_chain (Hotel_Chain_ID) ON DELETE CASCADE
);

CREATE TABLE Hotel (
    Hotel_Chain_ID INTEGER NOT NULL,
    Hotel_ID SERIAL,
    Hotel_name VARCHAR(50) NOT NULL,
    Hotel_City VARCHAR(50) NOT NULL,
    Hotel_Contact_Email VARCHAR(50) NOT NULL,
    Number_Of_Rooms INTEGER DEFAULT 0 NOT NULL,
    Rating INTEGER CHECK (Rating BETWEEN 1 AND 5),
    manager_ssn NUMERIC(9),

    PRIMARY KEY (Hotel_ID),
    FOREIGN KEY (Hotel_Chain_ID) REFERENCES Hotel_Chain(Hotel_Chain_ID) ON DELETE CASCADE
);

CREATE TABLE Hotel_PhoneNumbers (
    Hotel_ID INTEGER,
    Phone_Number NUMERIC(10),

    PRIMARY KEY (Hotel_ID, Phone_Number),
    FOREIGN KEY (Hotel_ID) REFERENCES hotel(Hotel_ID) ON DELETE CASCADE
);
```

```sql

CREATE TABLE Room (
    Hotel_ID INTEGER,
    Room_Number INTEGER,
    Can_Be_Extended BOOLEAN NOT NULL,
    Has_Sea_View BOOLEAN NOT NULL,
    Has_Mountain_View BOOLEAN NOT NULL,
    Room_Capacity INTEGER NOT NULL,
    Price INTEGER NOT NULL,

    PRIMARY KEY (Hotel_ID, Room_Number),
    FOREIGN KEY (Hotel_ID) REFERENCES Hotel(Hotel_ID) ON DELETE CASCADE
);

CREATE TABLE Room_Amenities (
    hotel_id INTEGER,
    Room_Number INTEGER,
    Amenity VARCHAR(100),

    PRIMARY KEY (Hotel_ID, Room_Number, Amenity),
    FOREIGN KEY (hotel_id) REFERENCES Hotel(Hotel_ID) ON DELETE CASCADE,
    FOREIGN KEY (hotel_id, Room_Number) REFERENCES Room(hotel_id, Room_Number) ON DELETE CASCADE
);

CREATE TABLE Room_List_of_Problems (
    hotel_id INTEGER,
    Room_Number INTEGER,
    Problem VARCHAR(100),

    PRIMARY KEY (Hotel_ID, Room_Number, Problem),
    FOREIGN KEY (Hotel_ID) REFERENCES Hotel(Hotel_ID) ON DELETE CASCADE,
    FOREIGN KEY (hotel_id, Room_Number) REFERENCES Room(hotel_id, Room_Number) ON DELETE CASCADE
);

CREATE TABLE users (
    username VARCHAR(20),
    password VARCHAR(20) NOT NULL,
    type VARCHAR(10) DEFAULT 'user' CHECK (type IN ('user', 'employee', 'admin')),
    PRIMARY KEY (username)
);

```

```sql

CREATE TABLE Booking (
    Booking_ID SERIAL,
    Time_Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    check_in_date DATE NOT NULL,
    check_out_date DATE NOT NULL,
    Is_Renting BOOLEAN DEFAULT false,
    username VARCHAR(20),
    Is_Paid BOOLEAN DEFAULT false,
    Hotel_ID INTEGER NOT NULL,
    Room_Number INTEGER NOT NULL,

    FOREIGN KEY (username) REFERENCES users(username) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (Hotel_ID) REFERENCES Hotel(Hotel_ID) ON DELETE CASCADE,
    FOREIGN KEY (hotel_id, Room_Number) REFERENCES Room(hotel_id, Room_Number) ON DELETE CASCADE,
    PRIMARY KEY (Booking_ID)
);
CREATE TABLE Employee (
    SSN NUMERIC(9) NOT NULL,
    Name VARCHAR(20) NOT NULL,
    username VARCHAR(20),
    Hotel_ID INTEGER NOT NULL,

    FOREIGN KEY (Hotel_ID) REFERENCES Hotel(Hotel_ID) ON DELETE SET NULL, --supposing we don't fire them if the hotel closes
    FOREIGN KEY (username) REFERENCES users(username) ON UPDATE CASCADE ON DELETE CASCADE,
    PRIMARY KEY (SSN)
);
ALTER TABLE hotel ADD CONSTRAINT manages_a FOREIGN KEY (manager_ssn) REFERENCES employee(ssn) ON DELETE SET NULL;


CREATE TABLE Archive (
    Booking_ID INTEGER,
    Time_Created TIME,
    check_in_date DATE,
    check_out_date DATE,
    Is_Renting BOOLEAN,
    username VARCHAR(20),
    Is_Paid BOOLEAN,
    Room_Number INTEGER,
    Hotel_ID INTEGER,

    PRIMARY KEY (Booking_ID)
);
```

```
127
128  CREATE VIEW rooms_by_area AS SELECT hotel_city, count(*) FROM hotel JOIN room ON hotel.hotel_id = room.hotel_id GROUP BY hotel_city;
129  CREATE VIEW hotel_room_capacity AS SELECT room_number, room_capacity FROM room WHERE hotel_id = 40;
130
131  GRANT ALL ON SCHEMA public TO postgres;
132  GRANT ALL ON SCHEMA public TO web;
133  GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO postgres;
134  GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO web;
135  GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO postgres;
136  GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO web;
137
```

```
146  CREATE OR REPLACE FUNCTION increment_hotel_count() RETURNS trigger AS
147  $body$
148  BEGIN
149      UPDATE hotel_chain SET number_of_hotels = number_of_hotels + 1
150          WHERE hotel_chain_id = NEW.hotel_chain_id;
151      RETURN NEW;
152  END;
153  $body$
154  language plpgsql;
155
156  CREATE OR REPLACE FUNCTION decrement_hotel_count() RETURNS trigger AS
157  $body$
158  BEGIN
159      UPDATE hotel_chain SET number_of_hotels = number_of_hotels - 1
160          WHERE hotel_chain_id = OLD.hotel_chain_id;
161      RETURN NEW;
162  END;
163  $body$
164  language plpgsql;
165
166  CREATE OR REPLACE FUNCTION increment_room_count() RETURNS trigger AS
167  $body$
168  BEGIN
169      UPDATE hotel SET number_of_rooms = number_of_rooms + 1
170          WHERE hotel_id = NEW.hotel_id;
171      RETURN NEW;
172  END;
173  $body$
174  language plpgsql;
175
176  CREATE OR REPLACE FUNCTION decrement_room_count() RETURNS trigger AS
177  $body$
178  BEGIN
179      UPDATE hotel SET number_of_rooms = number_of_rooms - 1
180          WHERE hotel_id = OLD.hotel_id;
181      RETURN NEW;
182  END;
183  $body$
184  language plpgsql;
```

```
CREATE TRIGGER add_hotel AFTER INSERT ON hotel
    FOR EACH ROW EXECUTE PROCEDURE increment_hotel_count();

CREATE TRIGGER del_hotel AFTER DELETE ON hotel
    FOR EACH ROW EXECUTE PROCEDURE decrement_hotel_count();

CREATE TRIGGER add_room AFTER INSERT ON room
    FOR EACH ROW EXECUTE PROCEDURE increment_room_count();

CREATE TRIGGER del_room AFTER DELETE ON room
    FOR EACH ROW EXECUTE PROCEDURE decrement_room_count();


CREATE OR REPLACE FUNCTION insert_archive() RETURNS trigger AS
$body$
BEGIN
    INSERT INTO archive
        (Booking_ID, Time_Created, check_in_date, check_out_date, Is_Renting, username, Is_Paid, Room_Number, Hotel_ID)
    VALUES
        (NEW.Booking_ID, NEW.Time_Created, NEW.check_in_date, NEW.check_out_date, NEW.Is_Renting, NEW.username,
            NEW.Is_Paid, NEW.Room_Number, NEW.Hotel_ID);
    RETURN NEW;
```

ystem Settings

```
language plpgsql;

CREATE OR REPLACE FUNCTION update_archive() RETURNS trigger AS
$body$
BEGIN
    UPDATE archive SET
        Time_Created = NEW.Time_Created, check_in_date = NEW.check_in_date, check_out_date = NEW.check_out_date,
            Is_Renting = NEW.Is_Renting, username = NEW.username, Is_Paid = NEW.Is_Paid, Room_Number = NEW.Room_Number,
            Hotel_ID = NEW.Hotel_ID WHERE Booking_ID = NEW.Booking_ID;
    RETURN NEW;
END;
$body$
language plpgsql;


CREATE TRIGGER add_archive AFTER INSERT ON booking
    FOR EACH ROW EXECUTE PROCEDURE insert_archive();

CREATE TRIGGER update_archive AFTER UPDATE ON booking
    FOR EACH ROW EXECUTE PROCEDURE update_archive();
```