University Of Ottawa

Deliverable #2 - Course Project

CSI 2132 [A]

Edward He - 300176553
Kevin Yu - 300230560

1. The DBMS, programming languages, and development applications used in our implementation of the 2123 course project include; PostgreSQL, Java, HTML, Apache Tomcat, IntelliJ, and Java Script.

2. Included below in the table are the instructions to install each application used in the 2132 course project.

Apache Tomcat
1. Visit the official website of Apache Tomcat 2. Download and unzip Apache Tomcat 3. Place the unzipped folder into the desired location# 4. Open terminal 5. Navigate to the bin directory for the extracted version of Apache Tomcat using the command: cd/path/to/apache-tomcat/bin and replace the path with the respective location mentioned above
PostgreSQL
1. Visit the official website of PostgreSQL 2. Select and download the desired version for your needs 3. Open the download and launch the installer 4. Follow the prompts provided by the Postgre installer to complete the installation 5. Launch the PostgreSQL application 6. Initialize a password for Postgre
IntelliJ
1. Visit the official website of IntelliJ 2. Select and download the desired version for your needs 3. Open the downloaded file and drag the IntelliJ icon into the applications folder 4. Launch IntelliJ and instantiate the settings based on your preference 5. Set a theme based on preference 6. If you require and plugins got to the “Preferences” section in the IntelliJ menu and download any respective plugins

The steps included in the table above are for mac and so users on windows may require a different set of instructions.

3. List of DDL that create your database

Hotel_Chain:

```
CREATE TABLE IF NOT EXISTS public.Hotel_Chain (
   chain_id VARCHAR PRIMARY KEY,
   name VARCHAR(50),
   street_number INTEGER CHECK (street_number >= 0 AND street_number < 1000),
   street_name VARCHAR(100),
   unit_number INTEGER CHECK (unit_number >= 0),
   city VARCHAR(100),
   state VARCHAR(2),
   country VARCHAR(50),
   postal_code VARCHAR(10)
);
```

Hotel:

```
CREATE TABLE IF NOT EXISTS public.Hotel (
   hotel_id VARCHAR(5) PRIMARY KEY,
   chain_id VARCHAR,
   street_number INTEGER,
   street_name VARCHAR(100),
   unit_number INTEGER,
   city VARCHAR(100),
   state VARCHAR(2),
   country VARCHAR(20),
   postal_code VARCHAR(10),
   rating INTEGER CHECK (rating >= 1 AND rating <= 5),
   email VARCHAR,
   -- Add attribute category
   category VARCHAR check (category = 'Luxury Hotel' OR category = 'Casino
Hotel' OR category = 'Business Hotel')
   CONSTRAINT fk_chain_id FOREIGN KEY (chain_id) REFERENCES
Hotel_Chain(chain_id)
);
```

Room:

```
CREATE TABLE IF NOT EXISTS public.Room (
   room_number INTEGER PRIMARY KEY CHECK (room_number >= 0),
   hotel_id VARCHAR,
   price FLOAT(2),
   capacity INTEGER,
   max_capacity INTEGER CHECK (max_capacity > capacity),
   CONSTRAINT fk_hotel_id FOREIGN KEY (hotel_id) REFERENCES Hotel(hotel_id)
);
```

Person:

```
CREATE TABLE IF NOT EXISTS public.Person (
   SIN VARCHAR PRIMARY KEY,
   first_name VARCHAR,
   last_name VARCHAR,
   middle_name VARCHAR,
   street_number INTEGER CHECK (street_number > 0),
   street_name VARCHAR,
   unit_number INTEGER CHECK (unit_number > 0),
   city VARCHAR,
   state VARCHAR(2),
   country VARCHAR,
   postal_code VARCHAR
);
```

Employee:

```
CREATE TABLE IF NOT EXISTS public.Employee (
   employee_id VARCHAR PRIMARY KEY,
   supervisor_id VARCHAR,
   SIN VARCHAR,
   hotel_id VARCHAR(5),
   salary FLOAT CHECK (salary >= 0),
   CONSTRAINT fk_SIN FOREIGN KEY (SIN) REFERENCES Person(SIN),
   CONSTRAINT fk_hotel_id FOREIGN KEY (hotel_id) REFERENCES Hotel(hotel_id)
);
```

Customer:

```
CREATE TABLE IF NOT EXISTS public.Customer (
   SIN VARCHAR PRIMARY KEY,
   registration_date DATE,
   CONSTRAINT fk_SIN FOREIGN KEY (SIN) REFERENCES Person(SIN)
);
```

Booking:

```
CREATE TABLE IF NOT EXISTS public.Booking (
   booking_id VARCHAR PRIMARY KEY,
   check_in_date DATE,
   check_out_date DATE
);
```

Renting:

```
CREATE TABLE IF NOT EXISTS public.Renting (
   booking_id VARCHAR PRIMARY KEY,
   cc_number INTEGER,
   expiry_date DATE,
   CONSTRAINT fk_booking_id FOREIGN KEY (booking_id) REFERENCES
Booking(booking_id)
);
```

Merged table for Hotels and Rooms:

```
CREATE TABLE IF NOT EXISTS public.Hotel_Room_Join (
    hotel_id VARCHAR(5),
    chain_id VARCHAR,
    street_number INTEGER,
    street_name VARCHAR(100),
    unit_number INTEGER,
    city VARCHAR(100),
    state VARCHAR(2),
    country VARCHAR(20),
    postal_code VARCHAR(10),
    rating INTEGER,
    email VARCHAR,
    category VARCHAR,
    room_number INTEGER,
    price FLOAT(2),
    capacity INTEGER,
    max_capacity INTEGER
);
```

6. Give the SQL code for at least 4 queries and 2 triggers of your choice

Queries

```
SELECT * FROM public.Hotel WHERE chain_id = 'C005';
```

```
SELECT * FROM RoomsByCity
```

```
SELECT bookings.booking_id, hotels.hotel_name, rooms.room_number,
bookings.check_in_date, bookings.check_out_date
FROM bookings
JOIN customers ON customers.customer_id = bookings.customer_id
JOIN rooms ON rooms.room_id = bookings.room_id
JOIN hotels ON hotels.hotel_id = rooms.hotel_id
WHERE customers.customer_id = [customer_id];
```

```
SELECT * room_id, room_number, capacity FROM rooms
WHERE hotel_id = [hotel_id];
```

Triggers

```
--Trigger to enforce the maximum capacity constraint
CREATE OR REPLACE FUNCTION check_capacity() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.capacity > NEW.max_capacity THEN
        RAISE EXCEPTION 'The capacity cannot exceed the maximum
capacity';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER capacity_trigger BEFORE INSERT OR UPDATE ON Room
FOR EACH ROW EXECUTE FUNCTION check_capacity();
```

```
--Trigger to calculate the total price of a booking
CREATE OR REPLACE FUNCTION update_booking_total_price() RETURNS TRIGGER
AS $$
BEGIN
  NEW.total_price = (NEW.check_out_date - NEW.check_in_date) * (SELECT
price FROM Room WHERE room_number = NEW.room_number);
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER calculate_booking_total_price
BEFORE INSERT OR UPDATE ON Booking
FOR EACH ROW
EXECUTE FUNCTION update_booking_total_price();
```

7. Implement at least three indexes on the relations of your database and justify why you have chosen these indexes: explain what type of queries and data updates you are expecting on your database and how these indexes are useful to accelerate querying the database.

CREATE INDEX idex_hotelchains_name ON HotelChains(name);
Index for hotel chains by name
CREATE INDEX idex_hotels_hotelchainid ON Hotels(hotel_chain_id);
Index for hotels associated by ID
CREATE INDEX idex_rooms_price ON Rooms(price);
Index for price of rooms

Each of the three indexes implemented above were chosen as these are useful in aiding common queries.

- The index for hotel chains helps accelerate any query that belongs to a specific hotel chain
- The index for hotel chain by associated foreign key helps accelerate queries that get bookings and rentals
- The index for price of rooms helps accelerate any query regarding updating or searching for specific price ranges