IN 2400 - Database Management Systems
Assignment 02: Group Assignment

Group 14

Team Data Pirates

Faculty of Information Technology

University of Moratuwa

2021

IN 2400 - Database Management Systems
Assignment 02: Group Assignment

Group 14

Group Members

Index No	Name
194002A	Abeysekara S.D. P
194158F	Silva S.A.Y. S
194035C	Dimal H.P. I
194137P	Ruwanpathirana Y. S
194172R	Tilanka K.P. P

Faculty of Information Technology

University of Moratuwa

2021

Table of Contents

1 Short description of the project scenario

This scenario is based on a Tourist guide web of Sri Lanka. Basically, this web application is designed to guide tourists who visit Sri Lanka. From using this website, tourist can make a trip with a book hotel.

There are two types of **users** they are Operator and Tourist. Tourist has attributes like Tourist_ID, Phone_Number, Email, Name, DOB, Country, Password, Passport No. Tourist_ID is unique to each other. Also, tourist can add more than one Phone Number.

When a new user is in the process of registration, and after tourist submit the registration form. After the registration is successful tourist can log in to the web. All tourists can view this website, but only registered tourists can make a trip using this website.

A tourist can **select** a location where he/she likes to visit. But that location should be in the location list. The **Location** is consistent with Location_Name, District, Town, Description, Photo, Category, latitude, longitude. Location_Name, District and Town are unique to each location. Also, Operator can upload many photos according to the relevant location.

After a tourist selects the location, he/she can use the Other **Transport** option in check transportation to reach the selected location. There are different kinds of transportation like Taxis, Three-wheeler and Motorcycle Under the Other Transport. It has common attributes like Vehicle_No, Price_for_first_Km, Price_for_other_Km, Contact_No, Capacity, Condition, Photo and Type. Vehicle_no is unique to each vehicle. Operator can add Many photos (about three photos) for relevant vehicles. Here we divide prices like Price_for_first_Km and Price_for_other_Km. In that case we mean, Price_for_first_Km includes the cost of the first kilometer and Price_for_other_Km includes the cost of the other kilometers except the first kilometer.

But the Bus category is not relevant to Other Transport. Tourist do not want details about Buses like other transportation details. Under the Bus category, tourists only want to know Bus route. Destination and Route_No are attributes of Bus entity. Route_No is the Primary key.

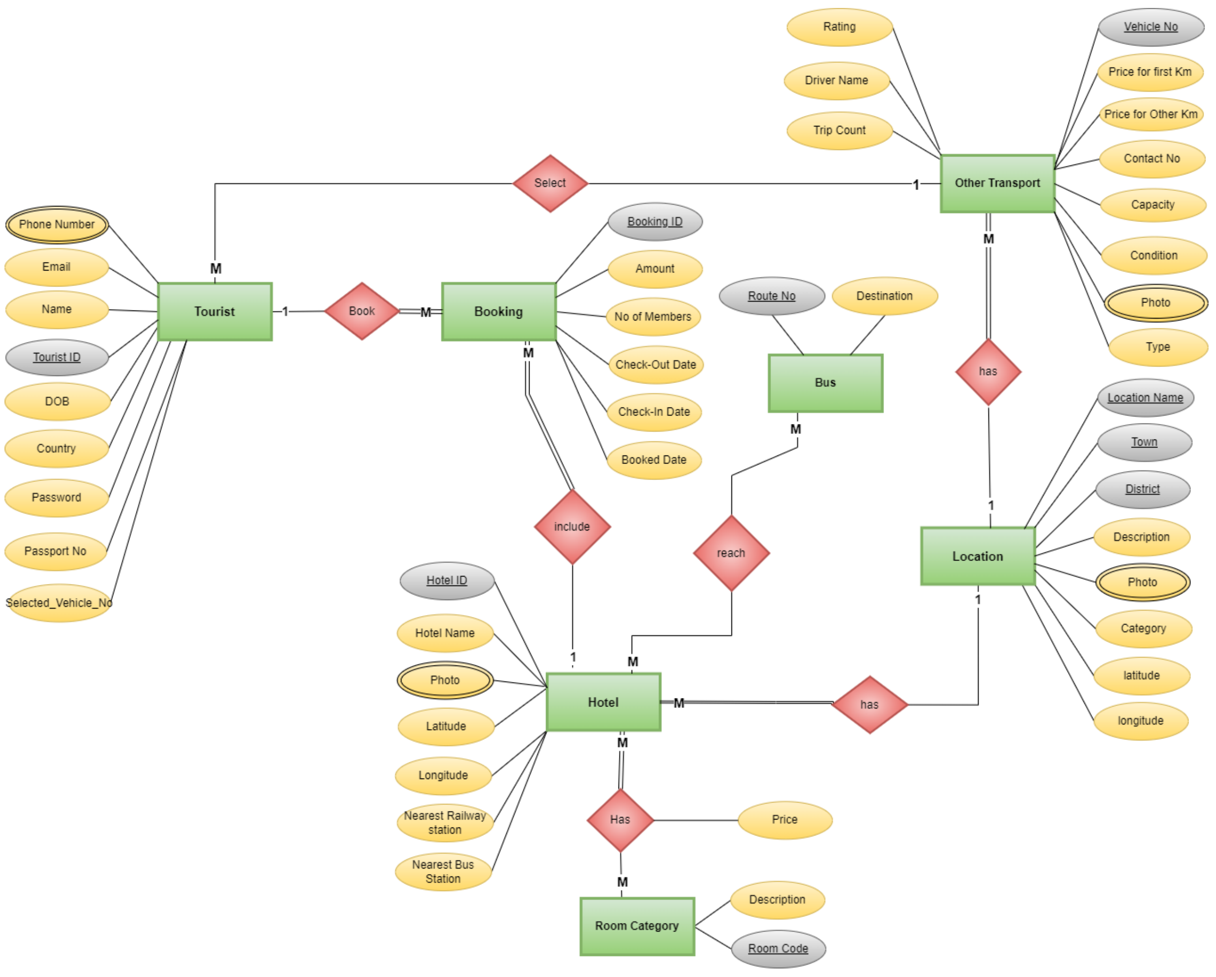
A tourist can **Book** a **Hotel** Around from the selected location. There is Hotel_ID, Hotel_Name, Photo, Latitude, Longitude, Nearest_Railway_Station, Nearest_Bus_Station. Operator can add many photos relevant to the hotel. Hotel_ID is the Primary Key.

Hotel has various kinds of **Room categories.** As an example, A/C room with bathtub or Non-A/C room with bathtub etc. There is Room_Code and Description. Room_Code is unique to each room category.

2 List of query requirements

- Create All tables in the database.
- Get Location details
- Add location
- Get Location, By name
- Get Booking details
- Get Register details of user
- Get hotel details
- Add hotels
- Delete hotels
- Update rating by vehicle No
- Get transportation details

3 ER/EER diagram



Assumptions

- We assumed one booking has one hotel.
- We assumed photo attribute implies path of the photo.
- We do not take selected_vehicle_no from the tourist. When tourist selects the transport method selected_vehicle_no is automatically updated.

4 Use of integrity constraints

Constraints are the rules that ensures the consistency, Accuracy and reliability of the Database. There rules are enforced on the columns of a table. Constraints could be either column level or a table level. Following are some rules we used when creating database.

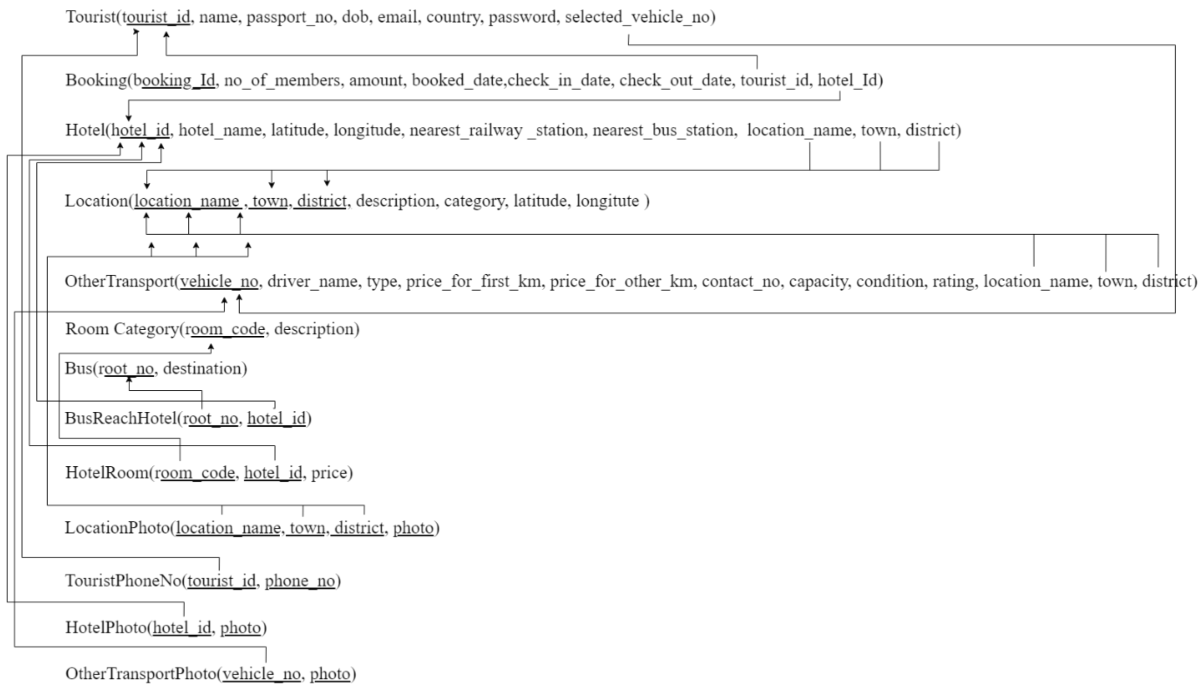
- **Not Null Constraint –** In tourist guide Application, most of the times we use not null constraint. It ensures that column cannot have null value.

- **Primary Key Constraint –** Primary key constraints are used to uniquely identify each record in a table. When uniquely identify each row, we use primary key constraints. As example Location table each row can uniquely identify using district, town, and location name. therefore, these three columns are used as primary key.

- **Foreign Key Constraints –** Foreign key constraints are used to ensure set of fields in one relation that is used to refer to a tuple in another relation. When creating foreign key columns in the tables most of the times, **CASCADE** should be used as the foreign key integrity constraint when **UPDATE** and **DELETE** data in our Database. As example we do not want hotel without the location. When we delete the location, relevant hotel lists around the location should also be deleted automatically.

- **Domain Constraints –** Domain constraints are used to ensure value of each attribute must be an atomic value from the domain. We use relevant data types to ensure Domain Constraint.

5 Normalizing the database to ensure efficient functionality



Tourist(tourist_id, name, passport_no, dob, email, country, password, selected_vehicle_no)

Booking(booking_Id, no_of_members, amount, booked_date,check_in_date, check_out_date, tourist_id, hotel_Id)

Hotel(hotel_id, hotel_name, latitude, longitude, nearest_railway _station, nearest_bus_station, location_name, town, district)

Location(location_name , town, district, description, category, latitude, longitute)

OtherTransport(vehicle_no, driver_name, type, price_for_first_km, price_for_other_km, contact_no, capacity, condition, rating, location_name, town, district)

Room Category(room_code, description)

Bus(root_no, destination)

BusReachHotel(root_no, hotel_id)

HotelRoom(room_code, hotel_id, price)

LocationPhoto(location_name, town, district, photo)

TouristPhoneNo(tourist_id, phone_no)

HotelPhoto(hotel_id, photo)

OtherTransportPhoto(vehicle_no, photo)

There are not any multivalued, composite, or nested relations. Because of that all the values should be atomic. Because of that above relation is in 1st normal form, because no values are given.

There are not any non-prime attributes partially dependent on primary. All non-prime attributes are now fully functional depending on the primary key. Because of that, the above relation is now in 2nd normal form.

There are not any non-prime attributes that transitively depend on the primary key. Because of that, the relation is now in 3rd normal form.

6 SQL statements to fulfill query requirements

```
-- Create Table Location
create table location(
   district varchar(50) NOT NULL,
   town varchar(50) NOT NULL,
   location_name varchar(50) NOT NULL,
   category varchar(50),
   description longtext,
   latitude varchar(100),
   longitude varchar(100),
   constraint location_pk primary key(district, town, location_name)
);

-- Create Table otherTransport
create table otherTransport(
   vehicle_no varchar(20),
   driver_name varchar(50),
   type varchar(50),
   price_for_first_km real,
   price_for_other_kms real,
   contact_no varchar(20),
   capacity int,
   v_condition varchar(100),
   rating int,
   district varchar(50),
   town varchar(50),
   location_name varchar(50),
   constraint otherTransport_pk primary key(vehicle_no),
   constraint other_transport_fk foreign key(district, town, location_name) references
location(district, town, location_name) on update cascade on delete cascade);

-- Create Table Tourist
create table tourist(
   tourist_id int NOT NULL AUTO_INCREMENT,
   tourist_name varchar(50),
   passport_no varchar(50),
   dob date,
   email varchar(50),
   country varchar(50),
   t_password varchar(100),
   selected_vehicle_no varchar(20),
   constraint tourist_pk primary key(tourist_id),
   constraint tourist_fk foreign key(vehicle_no) references otherTransport(vehicle_no)
on update cascade on delete no action);
```

```
-- Create Table Hotel
create table hotel(
   hotel_id int NOT NULL AUTO_INCREMENT,
   hotel_name varchar(100),
   latitude varchar(100),
   longitude varchar(100),
   nearest_railway_stat varchar(100),
   nearest_bus_stat varchar(100),
   district varchar(50),
   town varchar(50),
   location_name varchar(50),
   constraint hotel_pk primary key(hotel_id),
   constraint hotel_fk foreign key(district, town, location_name) references
location(district, town, location_name) on update cascade on delete cascade
);


-- Create Table Booking
create table booking(
   booking_id int NOT NULL AUTO_INCREMENT,
   no_of_members int,
   amount real,
   check_in_date date,
   check_out_date date,
   booked_date datetime,
   tourist_id int,
   hotel_id int,
   constraint booking_pk primary key(booking_id),
   constraint booking_fk1 foreign key(tourist_id) references tourist(tourist_id) on
update cascade on delete cascade,
   constraint booking_fk2 foreign key(hotel_id) references hotel(hotel_id) on update
cascade on delete no action
);

-- Create Table roomCategory
create table roomCategory(
   room_code varchar(10),
   r_description longtext,
   constraint roomCategory_pk primary key(room_code)
);
```

```
-- Create Table Bus
create table bus(
   root_no varchar(20),
   r_description mediumtext,
   constraint bus_pk primary key(root_no)
);

-- Create Table busReachHotel
create table busReachHotel(
   root_no varchar(20),
   hotel_id int,
   constraint busReachHotel_pk primary key(root_no, hotel_id),
   constraint busReachHotel_fk1 foreign key(root_no) references bus(root_no) on
update cascade on delete no action,
   constraint busReachHotel_fk2 foreign key(hotel_id) references hotel(hotel_id) on
update cascade on delete cascade
);

-- Create Table hotelRoom
create table hotelRoom(
   room_code varchar(10),
   hotel_id int,
   price real,
   constraint hotelRoom_pk primary key(room_code, hotel_id),
   constraint hotelRoom_fk2 foreign key(room_code) references
roomCategory(room_code) on update cascade on delete no action,
   constraint hotelRoom_fk1 foreign key(hotel_id) references hotel(hotel_id) on
update cascade on delete cascade
);

-- Create Table locationPhoto
create table locationPhoto(
   district varchar(50),
   town varchar(50),
   location_name varchar(50),
   photo varchar(100),
   constraint locationPhoto_pk primary key(district, town, location_name, photo),
   constraint locationPhoto_fk foreign key(district, town, location_name) references
location(district, town, location_name) on update cascade on delete cascade
);
```

```
-- Create Table touristPhoto
create table touristPhone(
    tourist_id int,
    phone_no varchar(20),
    constraint touristPhone_pk primary key(tourist_id, phone_no),
    constraint touristPhone_fk foreign key(tourist_id) references tourist(tourist_id) on
update cascade on delete cascade
);

-- Create Table hotelPhoto
create table hotelPhoto(
    hotel_id int,
    photo varchar(100),
    constraint hotelPhoto_pk primary key(hotel_id, photo),
    constraint hotelPhoto_fk foreign key(hotel_id) references hotel(hotel_id) on update
cascade on delete cascade
);

-- Create Table otherTransportPhoto
create table otherTransportPhoto(
    vehicle_no varchar(20),
    photo varchar(100),
    constraint otherTransportPhoto_pk primary key(vehicle_no, photo),
    constraint otherTransportPhoto_fk foreign key(vehicle_no) references
otherTransport(vehicle_no) on update cascade on delete cascade
);
```

7 Use of Views, Stored Procedures, Functions and Triggers

- **Views**

– 01. show hotel details to user

```
CREATE VIEW hotelDetailsView
AS
SELECT h.hotel_id, h.hotel_name, hr.price, rc.r_description, h.district, h.town,
h.location_name
FROM hotelRoom hr, roomCategory rc, hotel h
WHERE hr.room_code = rc.room_code and h.hotel_id = hr.hotel_id;
```

– 02. Show location details to user

```
CREATE VIEW locationDetailsView
AS
SELECT district, town, location_name, description
FROM location;
```

--03. Show Booking Details to user

```
CREATE VIEW bookingDetailsView
AS
SELECT t.name, b.check_in, b.check_out, b.amount
FROM booking b, tourist t
WHERE b.tourist_id=t.tourist_id;
```

--04. Show all registration Details to user

```
CREATE VIEW viewTouristDetails
AS
SELECT  Name, Email, Password, Phone Number, Country, Date of Birth,
Passport_No
FROM Tourist;
```

--05. Show Bus Details

```
CREATE VIEW TransportBusView
AS
SELECT b.root_no, b.r_description, h.nearest_bus_stat
FROM bus b, hotel h, busreachhotel br
WHERE br.hotel_id = h.hotel_id AND b.root_no = br.root_no;
```

--06. Show Other Transport Details

```
CREATE VIEW OtherTransportView
AS
SELECT o.*
FROM othertransport o, hotel h
```

```
        WHERE h.location_name = o.location_name AND h.district = o.district AND
h.town = o.town
```

- **Stored Procedures**

- 01. Add Locations to database Stored procedure

```
CREATE PROCEDURE addTourist(
@tourist_name varchar(50),
@ passport_no varchar(50),
@dob date,
@email varchar(50),
@country varchar(50),
@t_password varchar(100),
)
AS
BEGIN
  insert into tourist values(null, @tourist_name, @passport_no, @dob, @email,
@country, @t_password, null)
END
```

- 02. Add Locations to database Stored procedure

```
CREATE PROCEDURE addLocation(
@district varchar(50),
@town varchar(50),
@location_name varchar(50),
@category varchar(50),
@description longtext,
@latitude varchar(100),
@longitude varchar(100)
)
AS
BEGIN
  insert into location values(@district, @town, @location_name, @category,
@description, @latitude, @town, @longitude)
END
```

- 03. Add Hotels Stored procedure

```
CREATE PROCEDURE addhotels
@hname varchar(100),
@latitude varchar(100),
@longtude varchar(100),
@n_railway_stat varchar(100),
@n_bus_stat varchar(100),
@district varchar(50),
@town varchar(50),
@loc_name varchar(50),
AS
BEGIN
  insert into hotel
values(@hname,@latitude,@longtude,@n_railway_stat,@n_bus_sta,@district,@tow
n,@loc_name)
END
```

- 04. Delete Hotels Stored Procedure

```
CREATE PROCEDURE deletehotels
@hname
AS
BEGIN
 DELETE FROM hotel
 WHERE hotel_name = '@hname';
END
```

- 05. Delete Locations Stored Procedure

```
CREATE PROCEDURE deleteLocations
@locationName
AS
BEGIN
 DELETE FROM location
 WHERE location_name = '@locationName';
END
```

- 06 Update rating by vehicle No

```
CREATE PROCEDURE RatingOtherTransport
@rating int
@vehicle_no varchar(20)
AS
BEGIN
  update othertransport set rating = @rating where vehicle_no = @vehicle_no
END
```

8 The applicability of concurrent transaction management and locking

In single user systems data can be manipulated without risk. But when it comes to multi-user systems more than one user has the ability to access data, therefore concurrency control is essential.

Tourist guide Application is a multi-user system. Many tourists should have to access the same location at the same time, Multiple users should have access to hotels at the same time and also multiple users should have to book a hotel at the same time.

But all the users read only locations and hotels. When it comes to the booking there can suffer concurrency issues such as Lost Update problem, Dirty Read Problem and Incorrect Summary Issue. Therefore, we need concurrency control mechanisms to maintain integrity.

When it comes to booking hotels, multiple users can book the same hotel at the same time. Sometimes only one room is available in the hotel. Without concurrency controlling in Database Management System, it's possible to book that hotel room in the same hotel. But it is not possible. Because only one room is available in the hotel. However, the concurrency control method does not allow this to happen. Both tourists can still access information, but concurrency control can only provide a booking tourist who completed the transaction process first.

Therefore, we need a concurrency control protocol to handle the above problems. Different concurrency control protocols offer different benefits. But Tourist Guide Application We use Lock Based Protocols to ensure Concurrency and Serializability.

Lock-based protocols in DBMS transactions cannot read or write data until it acquires lock. It allows a transaction to access a data item only if that transaction holds a lock on that item.

In the tourist guide app, the tourist who completed the booking process first, can only read or write to data. In lock-based protocols there are two Locking Modes. There are Shared Locks (Read Lock) and Exclusive Locks (Write Lock). In Shared Locks Data items can only be read. In Exclusive locks data items can be both read and write.

Lock Based protocols also have problems. There are Deadlocks and Starvation. In deadlocks what happens is to create a waiting loop between transactions. In starvation what happens is One transaction waiting a very long period of time while other transactions release locks.

9 Appropriate backup and recovery policy

Backup is an additional copy of data that can be used for restoring and recovery purposes. In here the Back-up and recovery policy is used to safeguard the tourist, hotel, and location information and to prevent the loss of data**.** There are many ways that data can be loss, such as Security Breaches, Hardware Failures, Application Failures, Human Factors and Disasters. This backup and recovery policy contains guidelines to be followed in such a scenario of data loss.

There are three types of back-ups

1. **Full Back-up**

Full back-up means copy all the data in the system every time its back-up. This require lot of space but total data recovery is fast. In our tourist guide system, there are lot of tourist, hotels and locations that do daily activities. Therefore, using this type will cost lot of space. Because of that this method of data back-up is not suitable for our system.

2. **Incremental Back-up**

Incremental Back-up means copy all the data that has been changed since last partial back-up. This method takes less space and less time to back-up but recovering data take long time because all files are not in same place.

3. **Differential Back-up**

Differential Back-up means copy all the data that have changed since last full back-up. This includes any data that has been created, updated, or altered in any way and does not copy all of the data every time. Therefore, this method takes less space than full back-up but slower than incremental back-up.

Data belonging to Tourist, Hotel and Location can be entered frequently to the database and once they are entered those are mostly static data. Those should back up weekly basis. Since the most data in this database is static and we need to get the maximum out of the resources "Incremental backup" must be used in our system.

10 Use of index files to enhance performance

Indexes are very important for improving the performance and efficiency of the database. Indexes are very important for searching large tables effectively. Since our project Is Tourist guide app it has many locations, Hotels, transports. These details will increase day by day. So, getting this data very efficiently for users Indexes has a very important role in our Project. Some Indexes use for our project

```
CREATE CLUSTERED INDEX  loca_name ON location(location_name);
```

This Index is used to search Location faster by user. Anyone who searches locations by name can get that location faster.

```
CREATE NONCLUSTERED INDEX h_name ON hotel (hotel_name);
```

Using This Index is able to search Hotels faster by name. In the hotel table there may be several insertions available, so we use a non-clustered index to this one.

```
CREATE NONCLUSTERED INDEX b_ate ON booking(booked_date desc);
```

Here we used a no clustered index to show the latest booking details of the app. There we use descending order so we can find the latest booking at the top.

11 Access control

Access control is a method of guaranteeing that users are who they say they are and that they have the appropriate access to the website. Authorization and Authentication are the main parts in the user login process. There are two users in the tourist guide web application.

When an operator or tourist log in to the website, he/she should enter username and password. After verifying username and password tourist or operator can enter into the site. User Password is saved by encoding it in the Database. Because passwords are sensitive data.

After finishing the authentication part, the system should do the authorization part. From using the user password, the system can realize the user is a tourist or operator. According to our web site Operator has access to enter the dashboard. But tourists cannot view the dashboard. The Tourist can view only the home page. login page, select location page, hotels page and transportation page. Other than that, the operator can access the dashboard. But the operator does not even show the tourist sensitive data