MUDIT GUPTA : 2020315
SIYA GARG : 2020577
SRIJAN ARORA : 2020342
SRISHTI JAIN : 2020543

DBMS MID- EVALUATION
Indian Railway System (like IRCTC)

- **Scope of project** :

  - ❖ We have tried to implement the Railway Reservation System like IRCTC. We have included a train, bus, and flight reservation and booking system along with a payment portal.
  - ❖ Trains will also serve meals and their types available so that the user can choose from them.
  - ❖ For ticket pricing for all the transportations, we have added price per kilometer for each transport and we also have a price multiplier different for each class type. So if a user travels from any XYZ class they will pay *price_per_km x price_multiplier x distance* for the ticket
  - ❖ We have assumed that all the transports run every day and reach the destination within 24 hours for the simplicity of our database.
  - ❖ Our reservation system for all transports works on the following basis: Each vehicle has multiple stations associated with it where it stops, and travelers can board and unboard it there. The user will input the source and destination stations and then choose the train from the displayed options to cater to this.

- **Stakeholders** :

  a. **User/Traveler** - Users can log in to the website using their user id and password and book tickets for buses, flights, and trains.

  b. **Agent/Agencies** - Users can hire an agent or contact an agency and ask them to book the tickets for themselves. These agents are the travel agencies present in the locality. They will charge a nominal hiring cost to do the ticket booking task for users.

  c. **Indian Railway Management** - They will have access as an admin to view the number of seats booked, seats vacant, user information, etc. Simply put, they can access all the data present in the database.

- **Entities defined, with an underlined primary key, Relationship established**

  present in the attached file of ER Diagram.

- **Identification of Weak Entity and Why?**

Train classes and Airport Classes are weak entities because their existence is dependent on train and flight entities, respectively. Train and flight classes are represented by their names which are standard across all trains but are not independent of the train/flight they belong to.

- **Entity participation type** :

  a. We have **total participation with our weak classes,** i.e., Train classes and Airport classes, as it is a justifiable rule that all weak entities have to have an identifying entity, a tuple of a strong class.
  b. All **payments** have total participation with the booking (bus, train, flight) relation as every payment has to be linked with a booked ticket.
  c. All the other classes have partial participation with each other.

- **Relationship roles and Constraints:**

  mentioned in the attached file of ER Diagram properly

- **Identification of ternary relationship, if any, and why?**

  There are 3 ternary (in fact, quaternary) relations in our project
  1. Flight_booking between the user, flight, airport, and payment
  2. Train_booking between the user, train, station, and payment
  3. Bus_booking between the user, bus, bus_stop, and payment

  All three follow the same logic:
  Booking a ticket includes the user, the train/bus/flight that the user wishes to book, the source and destination stations/bus stops/airports to which the user wishes to go, and the payment for the ticket.

- **Relational Schema**

  ❖ **Entities:**

  User(<u>user_ID</u>, user_password, contact_number, DOB, gender, first_name, last_name, house_number, locality, city, state, pincode)

  Agent(<u>agent_ID</u>, agent_password)

  Train(<u>train_ID</u>, train_type, price_per_km)

  Station(<u>station_id</u>, station_name)

  Meal(<u>type_of_meal</u>, veg_menu, non_veg_menu)

  Train_classes(<u>train_id</u>, <u>class_type</u>, count_of_seats, price_multiplier)

  Bus_stop(<u>bus_stop_id</u>, bus_stop_name)

  Bus(<u>bus_ID</u>, bus_name, count_of_seats, bus_type, price_per_km)

  Airport(<u>airport_ID</u>, airport_name)

  Flight(<u>flight_ID</u>, flight_name, base price)

  flight_classes(<u>flight_id</u>, <u>class_type</u>, count_of_seats, price_multiplier)

Payment(payment_id, amount, payment_type)

- **Relationships with attributes:**

hires(user_id, agent_id, cost_of_hiring)

Train_stops(train_ID, station_ID, arrival_hour, arrival_minute, departure_hour, departure_minute)

Train_distance_from(source_id, destination_id, distance)

Train_booking(train_id, source_id, destination_id, user_id, ticket_id, payment_id)

Serves(train_id, type_of_meal)

Bus_distance_from(source_id, destination_id, distance)

Bus_booking(bus_id, source_id, destination_id, ticket_id, user_id, payment_id)

Bus_stops_at(bus_stop_id, bus_id, arrival_hour, arrival_minute, departure_hour, departure_minute)

Lands_at(flight_id, airport_id, arrival_hour, arrival_minute, departure_hour, departure_minute)

Flight_booking(flight_id, source_id, destination_id, user_id, ticket_id, payment_id)

- **Sufficient and valid constraints in DDL (of tables)**

```
create database test_project;
show databases;
use test_project;

create table User(user_ID int NOT NULL primary key, user_password
varchar(50), contact_number int8, DOB date, gender char(10), first_name
char(50), last_name char(50), house_number varchar(15), locality varchar(50),
city varchar(50), state varchar(50), pincode varchar(10));
create table Agent(agent_ID int NOT NULL primary key, agent_password
varchar(50));
create table hires
(user_ID int NOT NULL,
 agent_ID int,
 cost_of_hiring int,
 primary key(user_ID, agent_ID),
 foreign key(user_id) references User(user_id),
 foreign key(agent_ID) references Agent(agent_ID));
show tables;
create table Payment
(payment_id int NOT NULL primary key,
 amount int,
 payment_type varchar(20));
```

```
create table Train
(train_ID int NOT NULL,
train_type varchar(50),
price_per_km float,
primary key(train_ID));

create table Station(station_id int NOT NULL primary key, station_name char(50));
select * from agent;

create table Train_distance_from(source_id int, destination_id int, distance int,
primary key(source_id, destination_id),
foreign key(source_id) references Station(station_id),
foreign key(destination_id) references Station(station_id));

create table Train_stops(train_ID int, station_ID int, arrival_hour int, arrival_minute
int, departure_hour int, departure_minute int, primary key(train_ID, station_ID),
foreign key(train_ID) references Train(train_ID),
foreign key(station_id) references Station(station_id));

create table Train_booking(train_id int , source_id int, destination_id int, user_id
int, ticket_id int, payment_id int,
primary key(train_id, source_id, destination_id, ticket_id, user_id),
foreign key(train_id) references Train(train_ID),
foreign key(source_id) references Station(station_ID),
foreign key(destination_ID) references Station(station_ID),
foreign key(user_ID) references User(user_ID),
foreign key(payment_id) references Payment(payment_id));

create table train_classes(train_id int, class_type varchar(50), count_of_seats int,
price_multiplier int,
primary key(train_id, class_type),
foreign key(train_id) references train(train_id));

create table Bus_stop(bus_stop_id int NOT NULL primary key, bus_stop_name
char(50));

create table Bus_distance_from(source_id int, destination_id int, distance int,
primary key(source_id, destination_id),
foreign key(source_id) references Bus_stop(bus_stop_id),
foreign key(destination_id) references Bus_stop(bus_stop_id));

create table Bus(bus_ID int NOT NULL primary key, bus_name varchar(50),
count_of_seats int, bus_type varchar(50), price_per_km int);
```

```
create table Bus_stops_at(bus_stop_id int , bus_ID int, arrival_hour int,
arrival_minute int, departure_hour int, departure_minute int,
primary key(bus_stop_id, bus_ID),
foreign key(bus_stop_id) references Bus_stop(bus_stop_id),
foreign key(bus_id) references Bus(bus_id));

create table Bus_booking(bus_id int, source_id int, destination_id int, ticket_id int,
user_id int, payment_id int,
primary key(bus_id, source_id, destination_id, ticket_id),
foreign key(bus_id) references Bus(bus_id),
foreign key(source_id) references Bus_stop(bus_stop_id),
foreign key(destination_id) references Bus_stop(bus_stop_id),
foreign key(payment_id) references Payment(payment_id));

create table Airport(airport_ID int NOT NULL primary key, airport_name char(50));

create table Flight(flight_ID int NOT NULL primary key, flight_name char(50),
base_price int);

create table Lands_at(flight_id int, airport_id int, arrival_hour int, arrival_minute
int, departure_hour int, departure_minute int,
primary key(flight_id, airport_id),
foreign key(flight_id) references Flight(flight_id),
foreign key(airport_id) references Airport(airport_id));

create table flight_classes(flight_id int, class_type char(50), count_of_seats int,
price_multiplier int,
primary key(flight_id, class_type),
foreign key(flight_id) references Flight(flight_id));

create table Flight_booking(flight_id int , source_id int, destination_id int, user_id
int, ticket_id int, payment_id int,
primary key(flight_id, source_id, destination_id, user_id, ticket_id),
foreign key(flight_id) references Flight(flight_id),
foreign key(source_id) references Airport(airport_id),
foreign key(destination_id) references Airport(airport_id),
foreign key(user_id) references User(user_id),
foreign key(payment_id) references Payment(payment_id));

create table Meal
(type_of_meal varchar(50) primary key,
 veg varchar(200),
 non_veg varchar(200));
```

```
create table Serves
(type_of_meal varchar(200),
 train_id int,
 primary key(train_id, type_of_meal),
 foreign key(type_of_meal) references Meal(type_of_meal),
 foreign key(train_id) references Train(train_id));

 show tables;
```

- **Valid data (How to populate data, insertion of data, cardinality)**
    - We have populated our tables using the using the online platform mackaroo (https://www.mockaroo.com/)
    - We also specified a few constraints to make sure the random data that has been filled makes sense and matches our pre-requisites.
    - We arbitrarily filled the data in our table. There may be some unrealistic data in our tables, but we generated them randomly, applying constraints and formulas to our best abilities.
    - Some data has been filled by hand in order to ensure that our data makes sense and our attached queries return some values.
    - We then implemented the scripts that were generated into our MySQL program.
    - We had hand-written the code for table creation and took care of the cardinality during the data population.

- **SQL Queries**

1. Create view for agent to show class-wise price list for a particular flight from given source to given destination

```
create view flightPrice_agent as
select Flight.flight_id, Flight.base_price, flight_classes.class_type,
flight_classes.price_multiplier, Flight.base_price*flight_classes.price_multiplier*0.9
as true_price
from lands_at l1, lands_at l2, Flight join flight_classes on flight.flight_id =
flight_classes.flight_id
where l1.airport_id<l2.airport_id
and l1.flight_id=l2.flight_id
and l1.flight_id = flight.flight_ID
and l1.airport_id = 4414
and l2.airport_id = 8221
and Flight.flight_id=7925;
```

2. List all destinations possible from a source train station.

```
select distinct l2.station_id
```

```
from train_stops l1, train_stops l2
where l1.station_id=1721
and l1.train_id=l2.train_id
and l1.station_id!=l2.station_id;
```

3. Given/selected a station, list all trains that arrive at the selected station.

```
select train.train_id, train_type, train_stops.station_id
from train join train_stops join station
on train.train_id=train_stops.train_id
and station.station_id=train_stops.station_id
and train_stops.station_id=3303;
```

4. Given the source and destination, list all the trains traveling from source to the destination.

```
select train.train_id, train.train_type
from train, train_stops t1, train_stops t2
where t1.train_id=train.train_id
and t1.train_id=t2.train_id and t1.station_id=1721 and t2.station_id=2365 and
t1.arrival_hour<t2.arrival_hour;
```

5. List all the users traveling in a selected train.

```
select user.user_id, user.first_name, user.last_name
from user join train_booking on user.user_id=train_booking.user_id
where train_booking.train_id=70980;
```

6. List all the trains serving the given/selected type of meal.

```
select train.train_id, train.train_type
from train join serves on train.train_id=serves.train_id
where serves.type_of_meal="north west";
```

7. List all the planes' departure time and waiting time for a given/selected airport_id

```
select flight_id, departure_hour,departure_minute, departure_hour-arrival_hour as
waiting_hour,  departure_minute-arrival_minute as waiting_minute
 from lands_at
 where airport_id=8667;
```

8. List all agents who have been hired by more than one user.

```
select h1.agent_id, count(*) as hired
from hires h1, hires h2
where h1.user_ID != h2.user_ID and h1.agent_ID = h2.agent_ID
group by h1.agent_id;
```

9. Given the source and destination airports and flight class for an airplane, list the prices of all airplanes satisfying the above conditions.

```
select Flight.flight_id, Flight.base_price, flight_classes.class_type,
flight_classes.price_multiplier, Flight.base_price*flight_classes.price_multiplier as
true_price
from lands_at l1, lands_at l2, Flight join flight_classes on flight.flight_id =
flight_classes.flight_id
where l1.airport_id<l2.airport_id
and l1.flight_id=l2.flight_id
and l1.flight_id = flight.flight_ID
and l1.airport_id = 4414
and l2.airport_id = 8221 ;
```

10. Given source, destination and bus type for a bus list all bus names satisfying the above conditions

```
select bus_name
from bus_stops_at b1, bus_stops_at b2, Bus
where b1.bus_id=b2.bus_id
and b1.bus_stop_id=129
and b2.bus_stop_id=202
and Bus.bus_type="non-AC";
```