UNIVERSITETI® METROPOLITAN TIRANA

HOSPITAL PARKING MANAGEMENT SYSTEM

Keli Mucaj

Introduction

Hospital parking is a vital resource that is used by hospital staff and patients on a daily basis, but managing it can be a challenging task. One of the common problems faced in such sites is severe underutilisation of certain car parks while overcrowding in other areas. This issue arises due to the varying schedules and shifts of doctors, nurses, surgeons, and other hospital staff. Additionally, managing visitor parking can also be a challenging task as it comes with its own unique set of requirements. To achieve efficient parking management, it is essential to control and regulate the parking lots, differentiating between visitors and people who may abuse hospital parking spaces. There is a need for a system that can cater to the diverse needs of staff and visitors.

The Hospital Parking Management System (HPMS) is a solution designed for patients and visitors who face parking space shortages due to mismanagement of parking spaces by not only other patients, but also staff at hospitals in Albania. The system enables users to reserve parking spots in advance and check the parking status at the hospital. The aim of the project was to solve the problem of parking space shortages by developing a web-based system that allows users to reserve available parking spaces in advance.

HPMS has two modules i.e. admin and regular.

Business Description

The hospital parking system is a database management system designed to manage parking spaces and reservations for a hospital's parking lot. It is designed to ensure that patients, hospital staff, and visitors have access to available parking spaces as per their requirements.

The organization responsible for this project is the hospital's facilities management department, which oversees the operation and maintenance of the hospital's physical infrastructure. The project will involve collaboration with IT staff and possibly third-party vendors for implementation and deployment.

The project is expected to take approximately 6 months, including design, development, testing, and deployment. During this time, the team will work closely with stakeholders to ensure the system meets their requirements and provides a user-friendly experience.

The project's scope is the entire hospital parking lot and all users who park their vehicles there. The system will provide services for visitors, staff, and other authorized users who need to park their vehicles on hospital premises. The system will also include features for administrators to manage the parking lot, such as monitoring usage and generating reports.

The hospital parking system has ample parking spots for different types of vehicles, equipped with advanced sensors that detect vehicle presence and update availability in real-time. Users can reserve parking spots in advance through the online portal or mobile app, and the system supports different user roles such as admin, staff, and patients/visitors. Parking tickets are generated for users who do not make reservations, and the system supports various payment methods, including credit/debit cards, mobile payments, and cash, with an electronic receipt generated for each payment. The system also allows the hospital administration to monitor and manage the parking lot's capacity, status of parking spaces, and generate reports on parking revenue.

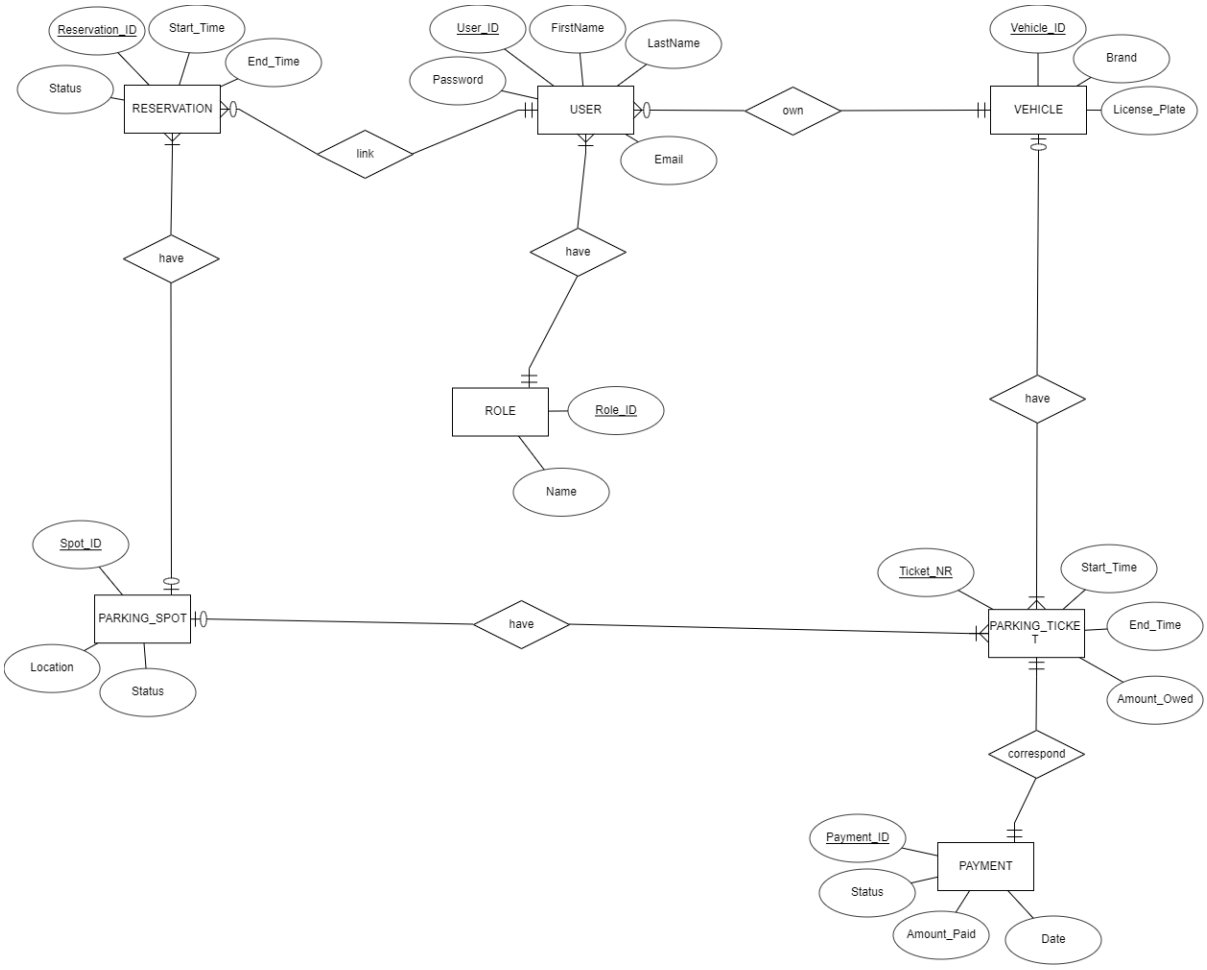
List of information requirements:

1. What is the occupancy rate of the parking lot for a specific day?
2. Which parking spots are most frequently used and which are least used?
3. How many reservations were made for a specific day?
4. Which vehicles have the highest number of parking tickets?
5. What is the revenue generated from the parking system for a specific period?
6. Which users have the highest number of reservations?
7. How many parking spots are currently available and what is their location?
8. How many vehicles have parked in the parking lot for a specific day?
9. What is the role of each user?
10. What is the duration of the average parking session?

Processes/Operations Supported by the Proposed Database:

- Creating new user accounts
- Assigning roles to users
- Assigning vehicles to users
- Reserving parking spots for specific users and vehicles
- Starting and ending parking sessions for vehicles
- Generating parking tickets for each parking session
- Recording payments for parking sessions
- Tracking the availability status of each parking spot

ERD (Entity Relationship Diagram)



Entities and Attributes:

An entity must be **relevant** and it must be **stored.**

An attribute is depiction of a characteristic of an entity.

ParkingSpot: represents the parking spot in the hospital parking lot.

Attributes:

spot_id: unique identifier of the parking spot.

location: the physical location of the parking spot in the parking lot.

status: the current status of the parking spot (e.g. vacant, occupied, reserved).

User: represents a driver who uses the parking lot and the DBMS.

Attributes:

user_id: unique identifier of the user.

name: name of the user.

email: email address of the user.

phone: phone number of the user.

Vehicle: represents a vehicle that is parked in the hospital parking lot.

Attributes:

vehicle_id: unique identifier of the vehicle.

plate: the license plate number of the vehicle.

make: the make of the vehicle.

model: the model of the vehicle.

Reservation: represents a reservation made by a user for a parking spot through the web.

Attributes:

resv_id: unique identifier of the reservation.

user_id: foreign key referencing User entity.

vehicle_id: foreign key referencing Vehicle entity.

spot_id: foreign key referencing ParkingSpot entity.

time_start: the start time of the reservation.

time_end: the end time of the reservation.

ParkingTicket: represents a parking ticket issued to a user for a parking spot that doesn't use the reservation system and/or the DBMS website.

Attributes:

ticket_nr: unique identifier of the parking ticket.

time_start: the start time of the parking duration.

time_end: the end time of the parking duration.

Role: represents the different levels of access and permissions that the users can have in the parking system, such as administrator or regular user.

Attributes:

role_id: unique identifier,

name: role name, i.e.: admin, regular.

Payment: represents the payment information for the parking reservation made by the user.

Attributes:

payment_id: unique identifier

amount paid: amount of money paid

date: date of payment

Relationships:

1. Reservation-User Relationship:

Cardinality: One-to-Many (1:N)

Optionality: Mandatory on the Reservation side, optional on the User side

Explanation: Each reservation must be associated with a user, but a user may have zero or more reservations.

2. User-Role Relationship:

Cardinality: One-to-Many (1:N)

Optionality: Mandatory on both sides

Explanation: Each user must have a role, and each role can be assigned to multiple users.

3. User-Vehicle Relationship:

Cardinality: One-to-Many (1:N)

Optionality: Optional on the Vehicle side, mandatory on the User side

Explanation: A user can have zero or more vehicles, but each vehicle must belong to a user.

4. ParkingSpot-Reservation Relationship:

Cardinality: One-to-Many (1:N)

Optionality: Optional on the Reservation side, mandatory on the ParkingSpot side

Explanation: A parking spot can have zero or more reservations, but each reservation must be associated with a parking spot. **one parking spot can have many reservations (one-to-many)**

5. Reservation-User Relationship:

Cardinality: One-to-Many (1:N)

Optionality: Mandatory on the Reservation side, optional on the User side

Explanation: Each reservation must be associated with a user, but a user may have zero or more reservations.

6. ParkingSpot-ParkingTicket Relationship:

Cardinality: One-to-Many (1:N)

Optionality: Optional on the ParkingTicket side, mandatory on the ParkingSpot side

Explanation: A parking spot can have zero or more parking tickets associated with it, but each parking ticket must be associated with a parking spot.

7. Vehicle-ParkingTicket Relationship:

Cardinality: One-to-Many (1:N)

Optionality: Optional on the ParkingTicket side, mandatory on the Vehicle side

Explanation: A vehicle can have zero or more parking tickets associated with it, but each parking ticket must be associated with a vehicle.

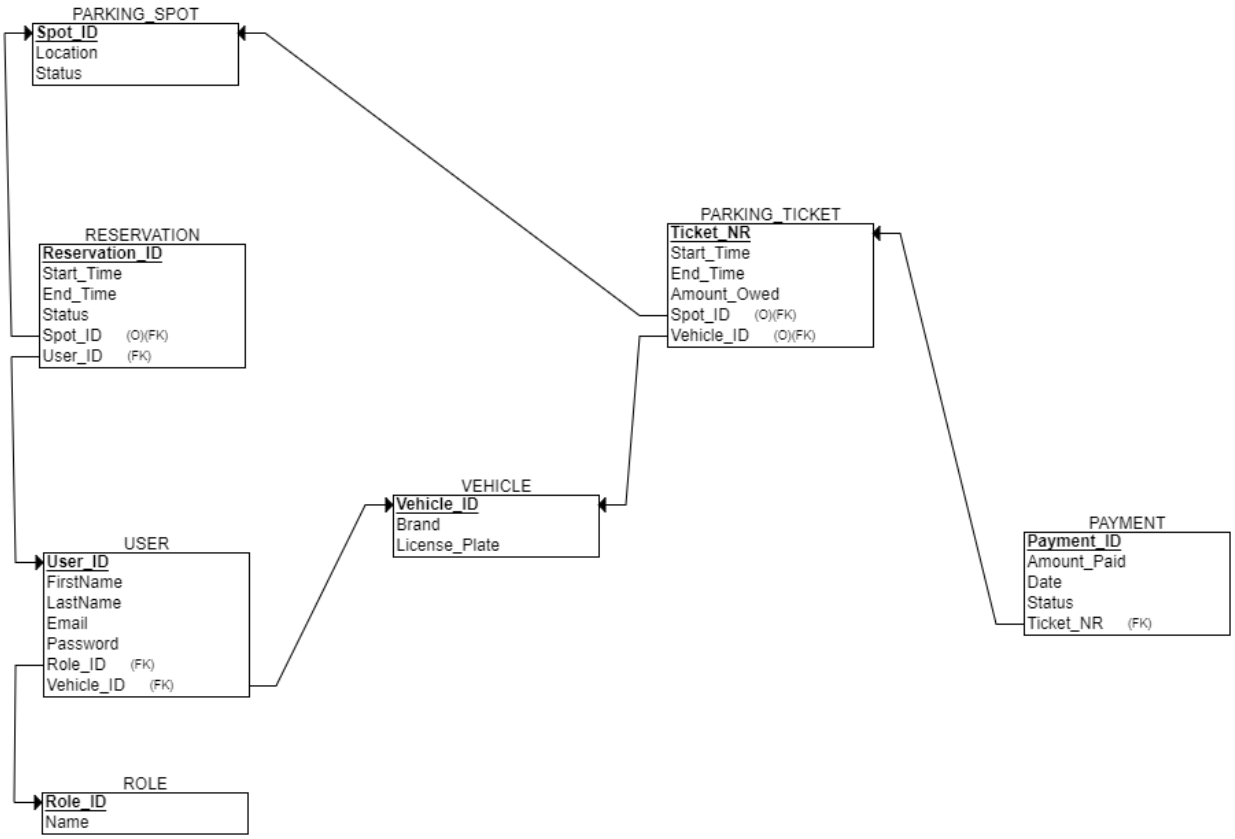
8. ParkingTicket-Payment Relationship:

Cardinality: One-to-One (1:1)

Optionality: Mandatory on both sides

Explanation: Each parking ticket must have exactly one payment associated with it, and each payment must be associated with exactly one parking ticket.

Relational Schema



Database implementation

- **Create database and tables:**

```
CREATE database hospitalparking;
USE hospitalparking;

SET FOREIGN_KEY_CHECKS=0;

CREATE TABLE parking_spot (
  id INT PRIMARY KEY,
  location VARCHAR(255) NOT NULL,
  status VARCHAR(255) NOT NULL,
  vehicle_id INT,
  FOREIGN KEY (vehicle_id) REFERENCES vehicle(id)
);

CREATE TABLE user (
  id INT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL,
  password VARCHAR(255) NOT NULL,
  role_id INT,
  FOREIGN KEY (role_id) REFERENCES role(id)
);

CREATE TABLE vehicle (
  id INT PRIMARY KEY,
  make VARCHAR(255) NOT NULL,
  model VARCHAR(255) NOT NULL,
  license_plate VARCHAR(255) NOT NULL,
  user_id INT,
  FOREIGN KEY (user_id) REFERENCES user(id)
);

CREATE TABLE reservation (
  id INT PRIMARY KEY,
  user_id INT,
  parking_spot_id INT,
  vehicle_id INT,
  start_time DATETIME NOT NULL,
  end_time DATETIME NOT NULL,
  FOREIGN KEY (user_id) REFERENCES user(id),
  FOREIGN KEY (parking_spot_id) REFERENCES parking_spot(id),
  FOREIGN KEY (vehicle_id) REFERENCES vehicle(id)
);

CREATE TABLE role (
  id INT PRIMARY KEY,
  name VARCHAR(255) NOT NULL
);

CREATE TABLE parking_ticket (
  id INT PRIMARY KEY,
  parking_spot_id INT,
```

```
  vehicle_id INT,
  start_time DATETIME NOT NULL,
  end_time DATETIME NOT NULL,
  payment_id INT,
  FOREIGN KEY (parking_spot_id) REFERENCES parking_spot(id),
  FOREIGN KEY (vehicle_id) REFERENCES vehicle(id),
  FOREIGN KEY (payment_id) REFERENCES payment(id)
);

CREATE TABLE payment (
  id INT PRIMARY KEY,
  user_id INT,
  amount DECIMAL(10,2) NOT NULL,
  payment_time DATETIME NOT NULL,
  FOREIGN KEY (user_id) REFERENCES user(id)
);
```

- **INSERT INTO values**

```
-- Populating the Database
-- Data for parking_spot table
INSERT INTO parking_spot (id, location, status, vehicle_id) VALUES
(1, 'Street A, Block 1, Lot 5', 'available', NULL),
(2, 'Street B, Block 2, Lot 10', 'occupied', 1),
(3, 'Street C, Block 3, Lot 15', 'available', NULL),
(4, 'Street D, Block 4, Lot 20', 'available', NULL),
(5, 'Street E, Block 5, Lot 25', 'occupied', 2);

-- Data for user table
INSERT INTO user (id, name, email, password, role_id) VALUES
(1, 'John Smith', 'john.smith@example.com', 'password123', 2),
(2, 'Jane Doe', 'jane.doe@example.com', 'password456', 1),
(3, 'Bob Johnson', 'bob.johnson@example.com', 'password789', 2);

-- Data for vehicle table
INSERT INTO vehicle (id, make, model, license_plate, user_id) VALUES
(1, 'Honda', 'Accord', 'ABC123', 2),
(2, 'Toyota', 'Camry', 'XYZ789', 1),
(3, 'Ford', 'Mustang', 'DEF456', 3);

-- Data for reservation table
INSERT INTO reservation (id, user_id, parking_spot_id, vehicle_id,
start_time, end_time) VALUES
(1, 1, 1, NULL, '2022-05-19 10:00:00', '2022-05-01 11:00:00'),
(2, 2, 3, 2, '2022-05-20 13:00:00', '2022-05-02 15:00:00'),
(3, 3, 4, 3, '2022-05-21 09:00:00', '2022-05-03 12:00:00'),
(4, 3, 2, 5, '2023-02-23 10:00:00', '2023-02-23 12:00:00'),
(5, 5, 4, 7, '2023-02-24 09:00:00', '2023-02-24 11:00:00'),
(6, 2, 3, 8, '2023-02-25 08:00:00', '2023-02-25 10:00:00'),
(7, 1, 1, 6, '2023-02-25 12:00:00', '2023-02-25 14:00:00'),
(8, 4, 6, 9, '2023-02-26 13:00:00', '2023-02-26 15:00:00'),
(9, 1, 9, 10, '2023-02-27 10:00:00', '2023-02-27 12:00:00'),
(10, 2, 5, 11, '2023-02-27 14:00:00', '2023-02-27 16:00:00'),
(11, 3, 8, 12, '2023-02-28 11:00:00', '2023-02-28 13:00:00'),
```

```
(12, 4, 7, 13, '2023-02-28 14:00:00', '2023-02-28 16:00:00'),
(13, 5, 2, 14, '2023-03-01 09:00:00', '2023-03-01 11:00:00');

-- Data for role table
INSERT INTO role (id, name) VALUES
(1, 'Regular'),
(2, 'Admin');

-- Data for parking_ticket table
INSERT INTO parking_ticket (id, parking_spot_id, vehicle_id, start_time,
end_time, payment_id) VALUES
(1, 2, 1, '2022-05-04 08:00:00', '2022-05-04 10:00:00', 1),
(2, 5, 2, '2022-05-05 12:00:00', '2022-05-05 15:00:00', 2),
(3, 1, 3, '2022-05-06 09:00:00', '2022-05-06 11:00:00', 3);

-- Data for payment table
INSERT INTO payment (id, user_id, amount, payment_time) VALUES
(1, 1, 5.00, '2022-05-04 10:00:00'),
(2, 2, 7.50, '2022-05-05 15:00:00'),
(3, 3, 10.00, '2022-05-06 11:00:00');
```

- **SELECT SQL statements that respond to managerial queries**

1. What is the occupancy rate of the parking lot for a specific day?

```
-- the occupancy rate of the parking lot for a specific day
SELECT COUNT(DISTINCT parking_spot_id) / COUNT(*) AS occupancy_rate
FROM reservation
WHERE start_time >= '2023-02-28 00:00:00' AND end_time <= '2023-02-28
23:59:00';
```

2. Which parking spots are most frequently used and which are least used?

```
-- count of each parking spot based on the number of reservations made
SELECT parking_spot_id, COUNT(*) AS usage_count
FROM reservation
GROUP BY parking_spot_id
ORDER BY usage_count DESC;
```

3. How many reservations were made for a specific day?

```
-- counts the number of reservations for a specific time range
SELECT COUNT(*) AS reservation_count
FROM reservation
WHERE start_time >= '2023-02-01 00:00:00' AND end_time <= '2023-05-01
23:59:59';
```

4. Which vehicles have the highest number of parking tickets?

```
-- ticket count for each vehicle that has parked in the parking lot
SELECT vehicle_id, COUNT(*) AS ticket_count
FROM parking_ticket
GROUP BY vehicle_id
ORDER BY ticket_count DESC;
```

5. What is the revenue generated from the parking system for a specific period?

```
-- total revenue generated by the parking lot for a specific time range
SELECT SUM(amount) AS revenue
FROM payment
WHERE payment_time >= '2022-02-01 00:00:0' AND payment_time <= '2023-05-01
00:00:00';
```

6. Which users have the highest number of reservations?

```
-- reservation count for each user who has made a reservation
SELECT user_id, COUNT(*) AS reservation_count
FROM reservation
GROUP BY user_id
ORDER BY reservation_count DESC;
```

7. How many parking spots are currently available and what is their location?

```
-- retrieves the parking_spot_id and location for all available parking spots
SELECT id, location
FROM parking_spot
WHERE status = 'available';
```

8. How many vehicles have parked in the parking lot for a specific day?

```
-- number of vehicles parked in the parking lot on a specific day
SELECT COUNT(DISTINCT vehicle_id) AS parked_vehicle_count
FROM reservation
WHERE start_time >= '2023-02-28 00:00:00' AND end_time <= '2023-02-28
23:59:59';
```

9. What is the role of each user?

```
-- role of each user
SELECT user.name, role.name AS role_name
FROM user
JOIN role ON user.role_id = role.id;
```

10. What is the duration of the average parking session?

```
--  average duration of a parking reservation
SELECT AVG(TIMESTAMPDIFF(MINUTE, start_time, end_time)) as avg_duration
FROM reservation;
```

Data Warehouse Implementation

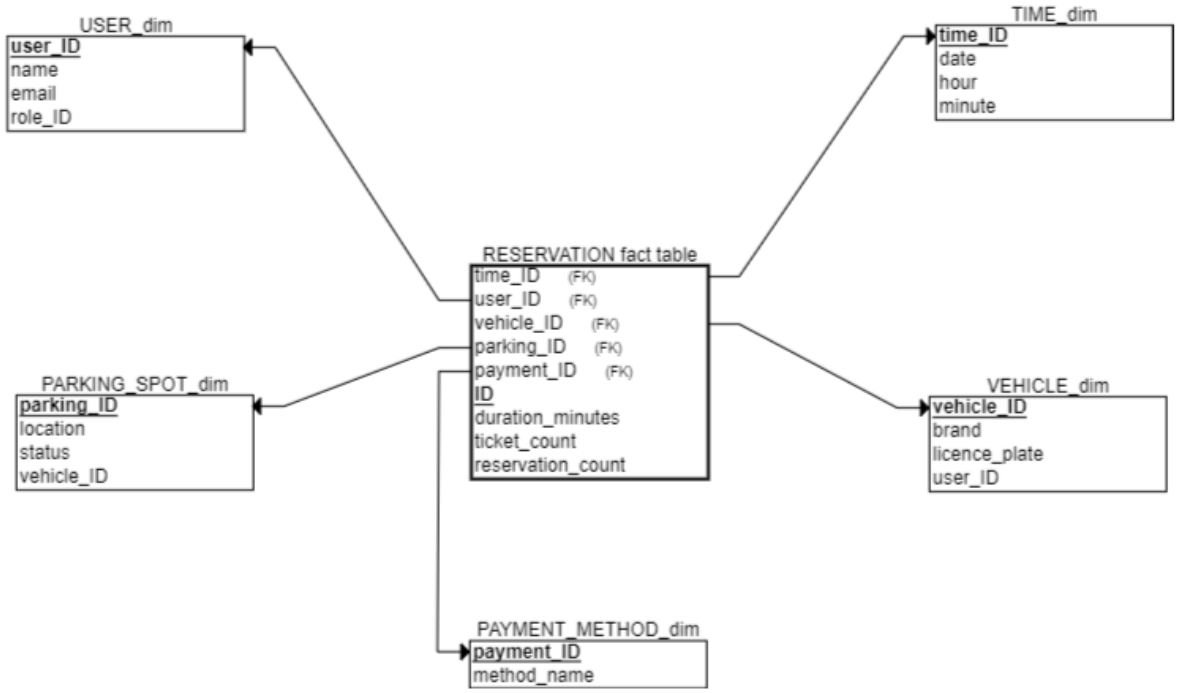
Measures/metrics of interest for a Hospital Parking Management System might include:

- Parking spot occupancy rate: the percentage of available parking spots that are occupied at any given time
- Average parking spot usage duration: the average amount of time a vehicle occupies a parking spot
- Parking revenue: the total amount of revenue generated by parking fees over a given period
- Parking ticket count: the number of parking tickets issued over a given period
- Reservation count: the number of parking reservations made over a given period

Example queries could include:

- What was the average parking spot usage duration for the month of January?
- How many parking tickets were issued for each day in the month of February?
- What was the occupancy rate of the parking garage during the month of March?
- What was the total parking revenue for the year 2022?
- How many parking reservations were made by each user role (e.g. employee, visitor, patient) in the month of May?

- **STAR schema**



Fact tables are the central tables in the data warehouse that store the measures or metrics of interest. In this case, the "parking_fact" table is the fact table that contains information about the parking activities, such as the duration of parking, the number of parking tickets issued, and the number of parking reservations made. The fact table has foreign key references to the dimension tables.

Dimension tables contain the attributes used to slice and dice the data. In this case, there are four dimension tables: "time_dim", "user_dim", "vehicle_dim", "parking_spot_dim", and "payment_method_dim". The dimension tables provide descriptive information about the entities involved in the parking activities, such as the time of the activity, the user who made the reservation or payment, the vehicle involved, the location and status of the parking spot, and the payment method used.

Each dimension table has a primary key and additional columns to store descriptive attributes. The dimension tables are related to the fact table through foreign key references, which allow for joining the fact table with the dimension tables to analyze the data from different perspectives.

Create data warehouse

- **CREATE SQL statements**

```
CREATE DATABASE hospital_parking_dw;
USE hospital_parking_dw;

SET FOREIGN_KEY_CHECKS=0;


CREATE TABLE parking_fact (
  id INT PRIMARY KEY,
  time_id INT NOT NULL,
  user_id INT NOT NULL,
  vehicle_id INT NOT NULL,
  parking_spot_id INT NOT NULL,
  payment_method_id INT NOT NULL,
  duration_minutes INT NOT NULL,
  ticket_count INT NOT NULL,
  reservation_count INT NOT NULL,
  FOREIGN KEY (time_id) REFERENCES time_dim(id),
  FOREIGN KEY (user_id) REFERENCES user_dim(id),
  FOREIGN KEY (vehicle_id) REFERENCES vehicle_dim(id),
  FOREIGN KEY (parking_spot_id) REFERENCES parking_spot_dim(id),
  FOREIGN KEY (payment_method_id) REFERENCES payment_method_dim(id)
);

CREATE TABLE time_dim (
  id INT PRIMARY KEY,
  date DATE NOT NULL,
  hour INT NOT NULL,
  minute INT NOT NULL
);

CREATE TABLE user_dim (
  id INT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL,
  role_id INT NOT NULL,
  FOREIGN KEY (role_id) REFERENCES role(id)
);

CREATE TABLE vehicle_dim (
  id INT PRIMARY KEY,
  brand VARCHAR(255) NOT NULL,
  license_plate VARCHAR(255) NOT NULL,
  user_id INT NOT NULL,
  FOREIGN KEY (user_id) REFERENCES user_dim(id)
);

CREATE TABLE parking_spot_dim (
  id INT PRIMARY KEY,
```

```
  location VARCHAR(255) NOT NULL,
  status VARCHAR(255) NOT NULL,
  vehicle_id INT,
  FOREIGN KEY (vehicle_id) REFERENCES vehicle_dim(id)
);

CREATE TABLE payment_method_dim (
  id INT PRIMARY KEY,
  method_name VARCHAR(255) NOT NULL
);
```

- ## ETL queries

```
-- ETL for loading data

INSERT INTO user_dim (id, name, email, role_id)
SELECT id, name, email, role_id FROM user;


INSERT INTO parking_spot_dim (id, location, status, vehicle_id)
SELECT id, location, status, vehicle_id FROM parking_spot;

INSERT INTO payment_method_dim (id, method_name)
SELECT id, method_name FROM payment_method;
```

- ### SELECT statements for data analysis

Count the number of parking reservations made by user role:

```
-- number of parking reservations made by user role
SELECT user_dim.role_id, COUNT(parking_fact.reservation_count) as
reservation_count
FROM parking_fact
JOIN user_dim ON parking_fact.user_id = user_dim.id
GROUP BY user_dim.role_id;
```

Calculate the average duration of parking spot usage by location and status:

```
-- average duration of parking spot usage by location and status
SELECT parking_spot_dim.location, parking_spot_dim.status,
AVG(parking_fact.duration_minutes) as avg_usage_duration
FROM parking_fact
JOIN parking_spot_dim ON parking_fact.parking_spot_id = parking_spot_dim.id
GROUP BY parking_spot_dim.location, parking_spot_dim.status;
```

Summary and Conclusions:

The Hospital Parking System is a computerized system designed to address the problems and challenges faced by hospitals and medical centers in managing parking spaces for their staff, patients, and visitors. With the increasing number of vehicles on the road, finding a parking space was a significant issue in urban areas, especially around hospitals and medical centers.

Traditionally, managing parking spaces was a manual process that required a lot of labor and resources. However, with the introduction of the Hospital Parking System, hospitals and medical centers can now manage their parking spaces more efficiently and with fewer resources.

One of the main benefits of the Hospital Parking System is that it allows clients and drivers to book parking spaces from home. This means that they don't have to waste time and effort finding a parking space when they arrive at the hospital or medical center. Instead, they can book a parking space in advance and have the peace of mind of knowing that they have a guaranteed space waiting for them when they arrive.

In conclusion, the Hospital Parking System developed using SQL is a highly beneficial tool for both clients and administrators. By allowing users to reserve parking spaces remotely and providing an efficient means of managing the parking process, this system is set to revolutionize the way parking is managed in busy urban areas. With its user-friendly interface and adaptability for future enhancements, this database management system addresses the drawbacks of the existing manual system while providing a more effective, streamlined solution. Ultimately, the Hospital Parking System offers an innovative solution to a long-standing problem and promises to make parking easier and more efficient for everyone involved.