

Fundamentals Of Database

Section 1

Assignment Phase #3

**Department Of Computer Science**

**Micro Link Information Technology College**

**Submitted To: Mr. Fisseha**

**Submission Date: Sep 9, 2023 GC**

**Group StudentsName............................ID No....**

1. Gemechis Chala 16,703/22

2.

3.

4.

5.

6.

7.

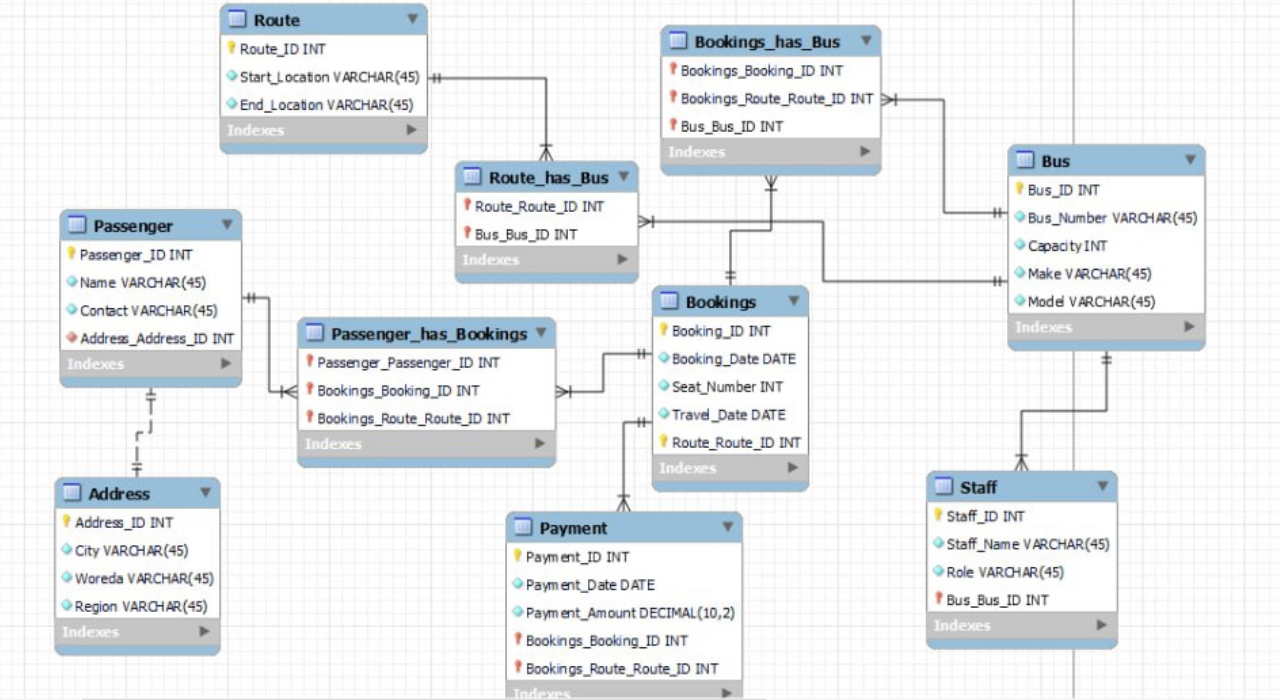
8.

9.

10.

11.

**Bus Booking System**



**Creating Database:**

CREATE DATABASE `BusTicketSystem`;

USE `BusTicketSystem`;

CREATE TABLE `Passenger` (

`Passenger\_ID` INT NOT NULL PRIMARY KEY,

`Name` VARCHAR(50),

`Contact\_Information` VARCHAR(100),

`Address\_ID` INT,

FOREIGN KEY (`Address\_ID`) REFERENCES `Address`(`Address\_ID`)

);

CREATE TABLE `Bus` (

`Bus\_ID` INT NOT NULL PRIMARY KEY,

`Bus\_Number` VARCHAR(20),

`Capacity` INT,

`Make` VARCHAR(50),

`Model` VARCHAR(50)

);

CREATE TABLE `Route` (

`Route\_ID` INT NOT NULL PRIMARY KEY,

`Start\_Location` VARCHAR(100),

`End\_Location` VARCHAR(100)

);

CREATE TABLE `Booking` (

`Booking\_ID` INT NOT NULL PRIMARY KEY,

`Booking\_Date` DATE,

`Seat\_Number` INT,

`Travel\_Date` DATE,

`Passenger\_ID` INT,

`Bus\_ID` INT,

`Route\_ID` INT,

`Payment\_ID` INT,

FOREIGN KEY (`Passenger\_ID`) REFERENCES `Passenger`(`Passenger\_ID`),

FOREIGN KEY (`Bus\_ID`) REFERENCES `Bus`(`Bus\_ID`),

FOREIGN KEY (`Route\_ID`) REFERENCES `Route`(`Route\_ID`),

FOREIGN KEY (`Payment\_ID`) REFERENCES `Payment`(`Payment\_ID`)

);

CREATE TABLE `Payment` (

`Payment\_ID` INT NOT NULL PRIMARY KEY,

`Payment\_Date` DATE,

`Payment\_Amount` DECIMAL(10,2)

);

CREATE TABLE `Staff` (

`Staff\_ID` INT NOT NULL PRIMARY KEY,

`Name` VARCHAR(50),

`Role` VARCHAR(50),

`Bus\_ID` INT,

FOREIGN KEY (`Bus\_ID`) REFERENCES `Bus`(`Bus\_ID`)

);

CREATE TABLE `Address` (

`Address\_ID` INT NOT NULL PRIMARY KEY,

`Street` VARCHAR(100),

`City` VARCHAR(100),

`State` VARCHAR(50),

`Postal\_Code` VARCHAR(10),

`Country` VARCHAR(50)

);

**Insert Sample Data In Each Table:**

-- Insert sample data into the Passenger table

INSERT INTO Passenger (Passenger\_ID, Name, Contact\_Information, Address\_ID)

VALUES

(1, 'Abel Tesfaye', 'abel@example.com', 1),

(2, 'Sara Kebede', 'sara@example.com', 2),

(3, 'Daniel Tadesse', 'daniel@example.com', 3);

-- Insert sample data into the Bus table

INSERT INTO Bus (Bus\_ID, Bus\_Number, Capacity, Make, Model)

VALUES

(101, 'ET-101', 50, 'Volvo', 'S90'),

(102, 'ET-102', 45, 'Mercedes', 'Sprinter');

-- Insert sample data into the Route table

INSERT INTO Route (Route\_ID, Start\_Location, End\_Location)

VALUES

(201, 'Addis Ababa', 'Bahir Dar'),

(202, 'Addis Ababa', 'Gondar');

-- Insert sample data into the Booking table

INSERT INTO Booking (Booking\_ID, Booking\_Date, Seat\_Number, Travel\_Date, Passenger\_ID, Bus\_ID, Route\_ID, Payment\_ID)

VALUES

(1001, '2023-08-20', 10, '2023-08-25', 1, 101, 201, 10001),

(1002, '2023-08-21', 5, '2023-08-26', 2, 101, 201, 10002),

(1003, '2023-08-22', 15, '2023-08-27', 3, 102, 202, 10003);

-- Insert sample data into the Payment table

INSERT INTO Payment (Payment\_ID, Payment\_Date, Payment\_Amount)

VALUES

(10001, '2023-08-22', 75.00),

(10002, '2023-08-23', 60.00),

(10003, '2023-08-24', 90.00);

-- Insert sample data into the Staff table

INSERT INTO Staff (Staff\_ID, Name, Role, Bus\_ID)

VALUES

(1001, 'Mulu Adamu', 'Driver', 101),

(1002, 'Teshome Kassa', 'Conductor', 101),

(1003, 'Selamawit Negash', 'Driver', 102);

-- Insert sample data into the Address table

INSERT INTO Address (Address\_ID, Street, City, State, Postal\_Code, Country)

VALUES

(1, '123 Main St', 'Addis Ababa', 'Addis Ababa', '1000', 'Ethiopia'),

(2, '456 Elm St', 'Bahirdar', 'Amhara', '2000', 'Ethiopia'),

(3, '789 Oak St', 'Gondar', 'Amhara', '3000', 'Ethiopia');

**Frequently Needed Queries:**

1. **List All Bookings for a Specific Passenger:**

SELECT \* FROM Booking WHERE Passenger\_ID = 1;

**2. List Available Buses for a Given Route and Date:**

SELECT Bus.Bus\_Number, Bus.Capacity

FROM Bus

JOIN Route ON Bus.Route\_ID = Route.Route\_ID

WHERE Route.Start\_Location = 'Addis Ababa' AND Route.End\_Location = 'Bahir Dar';

**3. Calculate Total Revenue for a Specific Bus:**

SELECT Bus.Bus\_Number, SUM(Payment.Payment\_Amount) AS Total\_Revenue

FROM Bus

JOIN Booking ON Bus.Bus\_ID = Booking.Bus\_ID

JOIN Payment ON Booking.Payment\_ID = Payment.Payment\_ID

WHERE Bus.Bus\_ID = 101

GROUP BY Bus.Bus\_Number;

**4. List Passengers and Their Contact Information for a Specific Route:**

SELECT Passenger.Name, Passenger.Contact\_Information

FROM Passenger

JOIN Booking ON Passenger.Passenger\_ID = Booking.Passenger\_ID

JOIN Route ON Booking.Route\_ID = Route.Route\_ID

WHERE Route.Start\_Location = 'Addis Ababa' AND Route.End\_Location = 'Bahir Dar';

**5. Find the Bus with the Most Bookings:**

SELECT Bus.Bus\_Number, COUNT(Booking.Booking\_ID) AS Bookings\_Count

FROM Bus

JOIN Booking ON Bus.Bus\_ID = Booking.Bus\_ID

GROUP BY Bus.Bus\_Number

ORDER BY Bookings\_Count DESC

LIMIT 1;

**6. List Staff Members Assigned to a Specific Bus:**

SELECT Staff.Name, Staff.Role

FROM Staff

WHERE Staff.Bus\_ID = 101;

**7. Find Passengers with Bookings on the Same Bus for a Given Date:**

SELECT DISTINCT P1.Name, P2.Name AS Co\_Passenger

FROM Passenger AS P1

JOIN Booking AS B1 ON P1.Passenger\_ID = B1.Passenger\_ID

JOIN Booking AS B2 ON B1.Bus\_ID = B2.Bus\_ID

JOIN Passenger AS P2 ON B2.Passenger\_ID = P2.Passenger\_ID

WHERE B1.Travel\_Date = '2023-08-25' AND B1.Bus\_ID = 101

AND P1.Name < P2.Name;

**8. Calculate the Average Payment Amount per Booking:**

SELECT AVG(Payment\_Amount) AS Avg\_Payment\_Amount

FROM Payment;

**Trigger:**

* This trigger automatically updates the "Capacity" of a bus whenever a new booking is made, this means after each new booking (INSERT operation) is made, it reduces the bus's capacity by 1:

DELIMITER //

CREATE TRIGGER UpdateBusCapacity

AFTER INSERT ON Booking

FOR EACH ROW

BEGIN

UPDATE Bus

SET Capacity = Capacity - 1

WHERE Bus\_ID = NEW.Bus\_ID;

END;

//

DELIMITER ;

**Stored Procedure:**

* This stored procedure retrieves all bookings for a specific passenger, by taking ‘passengerName’ as input:

DELIMITER //

CREATE PROCEDURE GetPassengerBookings(IN passengerName VARCHAR(50))

BEGIN

SELECT Booking.Booking\_ID, Booking.Booking\_Date, Booking.Seat\_Number

FROM Booking

JOIN Passenger ON Booking.Passenger\_ID = Passenger.Passenger\_ID

WHERE Passenger.Name = passengerName;

END;

//

DELIMITER ;

**Backup and Recovery Plan:**

1. **Regular Backups:** Implement automated daily backups of the entire database to ensure data integrity and availability. Store backups in secure off-site locations.
2. **Point-in-Time Recovery:** Enable point-in-time recovery to allow restoration to a specific moment in time, minimizing data loss.
3. **Test Backups:** Regularly test backup restoration procedures to verify their reliability and effectiveness.
4. **Disaster Recovery Plan:** Develop a comprehensive disaster recovery plan that includes steps for data restoration in case of catastrophic events.

**Access Levels and Privileges:**

1. **Administrator (Superuser):** This level has full access to the database, including the ability to create, modify, and delete tables, users, and roles.
2. **Database Developer:** This role can create and modify tables, views, and stored procedures, but does not have administrative privileges.
3. **Data Analyst:** Data analysts can execute SELECT queries, view data, and generate reports but cannot make structural changes to the database.
4. **Customer Support:** This role can access specific customer-related data for support purposes but cannot alter the database schema.

**Security Levels Enforcement:**

1. **Authentication:** Enforce strong password policies and two-factor authentication for user access.
2. **Authorization:** Implement role-based access control (RBAC) to ensure users only have access to data and actions relevant to their roles.
3. **Encryption:** Encrypt data in transit using SSL/TLS and encrypt sensitive data at rest using industry-standard encryption algorithms.
4. **Audit Trails:** Enable database auditing to track user activity and detect suspicious behavior.
5. **Regular Updates:** Keep the database management system and all associated software up to date with security patches and updates.
6. **Data Masking:** Implement data masking to protect sensitive information, ensuring it's only visible to authorized personnel.
7. **Firewall and Intrusion Detection:** Utilize firewalls and intrusion detection systems to monitor and prevent unauthorized access.
8. **Security Training:** Provide security training for all database users to raise awareness and promote best practices.

By implementing these measures, we ensure the confidentiality, integrity, and availability of the database, mitigating security risks and enabling efficient backup and recovery processes. This approach aligns with our commitment to excellence and our vision of uplifting Ethiopia and the African continent through transformative software solutions.