

Department Of Artificial Intelligence and Data-science

Course Code	22PC0AD11			
Course Title	Database Management Systems Lab			
Branch	B. Tech. Artificial Intelligence and Data Science			
Year and Semester	II Year II Semester			
Category	Professional Subjects – Core			
Scheme and Credits	L	T	P	Credits
	3	0	0	3
Prerequisites	22ES0AD01 – Data Structures			

Course Objectives:

1. To learn the ER data model, database design and normalization
2. To Learn SQL basics for data definition and data manipulation

Course Outcomes: After the completion of this course, the students would be able to

1. Demonstrate ER Modeling concepts to design the Database
2. Apply integrity constraints on a database
3. Make use of DDL, DML, DCL, TCL commands in creation and manipulation of Database
4. Implementation of database queries using PL/SQL
5. Experiment with triggers to maintain the referential integrity of data

LIST OF EXPERIMENTS:

1. Design ER Model for a given application & Convert ER model to Relational Model.
2. Creating users - roles and Granting privileges.
3. Creating and altering tables for various relations in SQL using Integrity Constraints.
4. Implementing queries in SQL using
 - 4.1 Insertion
 - 4.2 Retrieval (operations like union - intersect - minus - in - exists - group by and having)
 - 4.3 Updation
 - 4.4 Deletion
5. Implementing the concepts of Rollback – commit, checkpoints and Views

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE LAB MANUAL (AIDS-II-II)

6. Implementing joins - sub queries - nested and co related nested queries.
7. Experiment with built in functions in oracle (Numeric, String, Date, Aggregate function set.)
8. Implementing operations on relations using PL/SQL.
9. Implementing functions, stored procedures using PL/SQL
10. Implementing cursors using PL/SQL
11. Implement Exception Handling using PL/SQL
12. Creating triggers using PL/SQ

Case Study – Give options to the students to do case study on their own / faculty may give list of case study in beginning of the semester

1. Inventory control management System
2. College Management System
3. Hospital management System
4. Library management System
5. Payroll management System
6. Health care organization Management System
7. Restaurant Management System
8. Blood Donation Management System
9. Art Gallery Management System
10. Hotel Management System
11. School Management System
12. Salary Management System
13. Wholesale Management System
14. Timetable Management System
15. Website Management

Experiment1: Designing ER Model and Conversion to Relational Model for Library Management System

Tools Used

ER diagramming tool (such as Lucid chart, Draw.io, etc.)

Database management system (e.g., MySQL) for implementing the relational model.

Objective:

To design an Entity-Relationship (ER) model and convert it into a Relational Model for a Bus Management System.

Requirements:

Understanding of Entity-Relationship (ER) modeling concepts.

Knowledge of relational database design principles.

Familiarity with notation standards for ER diagrams.

Experiment:

1. Entity-Relationship (ER) Model:

Entities:

Bus: Represents a bus in the system.

Attributes: BusID (Primary Key), Route, Capacity, Type.

Driver: Represents a driver associated with a bus.

Attributes: DriverID (Primary Key), Name, LicenseNumber.

Passenger: Represents a passenger using the bus service.

Attributes: PassengerID (Primary Key), Name, Age, ContactInfo.

Route: Represents a route followed by buses.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE LAB MANUAL (AIDS-II-II)

Attributes: RouteID (Primary Key), Source, Destination, Distance.

Ticket: Represents a ticket issued to a passenger for a bus journey.

Attributes: TicketID (Primary Key), PassengerID (Foreign Key), BusID (Foreign Key), RouteID (Foreign Key), Date, Fare.

Relationships:

Bus-Driver: One-to-One relationship between Bus and Driver entities.

Bus-Passenger: One-to-Many relationship between Bus and Passenger entities.

Route-Bus: Many-to-Many relationship between Route and Bus entities.

Ticket-Route: Many-to-One relationship between Ticket and Route entities.

2. Relational Model:

Tables:

Bus table:

Columns: BusID (Primary Key), Route, Capacity, Type, DriverID (Foreign Key).

Driver table:

Columns: DriverID (Primary Key), Name, LicenseNumber.

Passenger table:

Columns: PassengerID (Primary Key), Name, Age, ContactInfo.

Route table:

Columns: RouteID (Primary Key), Source, Destination, Distance.

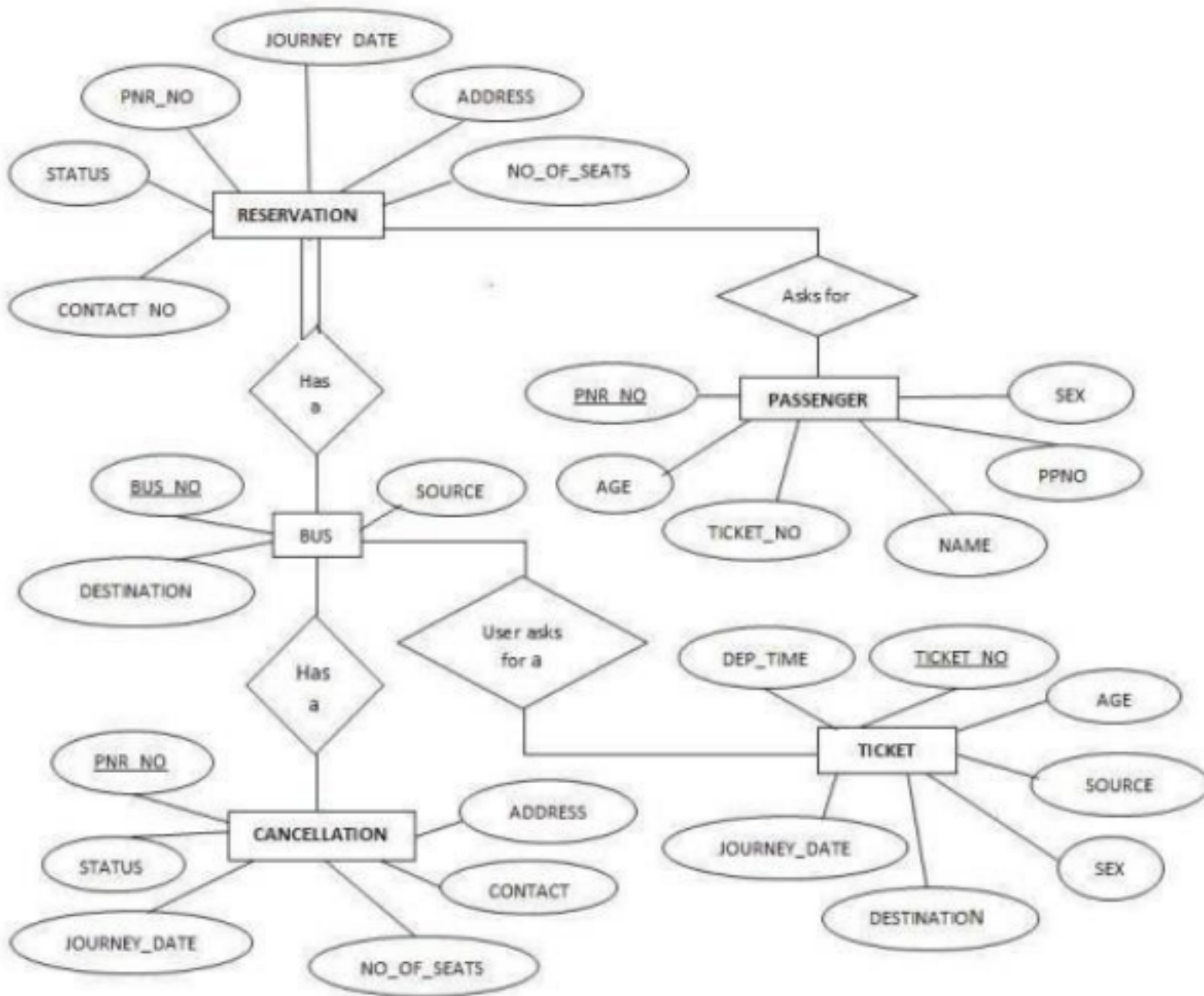
Ticket table:

Columns: TicketID (Primary Key), PassengerID (Foreign Key), BusID (Foreign Key), RouteID (Foreign Key), Date, Fare.

Conclusion:

The Entity-Relationship (ER) model provides a conceptual representation of the Bus Management System, illustrating entities, their attributes, and relationships. The Relational Model converts this conceptual design into a set of normalized tables, facilitating efficient data storage and management within a relational database system.

ER-Diagram: Bus-Management Systems



Experiment2. Creating users - roles and Granting privileges

Objective:

To create users, define roles, and grant privileges in a database management system.

Requirements:

Access to a database management system (e.g., MySQL).

Basic understanding of SQL (Structured Query Language).

Familiarity with user management and privilege granting in a database environment.

Experiment:

1. Creating Users:

Open your preferred database management tool (e.g., MySQL Workbench, pgAdmin).

Connect to the database server using appropriate credentials.

Execute the following SQL command to create a new user:

```
CREATE USER 'username'@'hostname' IDENTIFIED BY 'password';
```

Replace 'username' with the desired username, 'hostname' with the hostname or IP address from which the user will connect (e.g., 'localhost' for local connections), and 'password' with the desired password.

2. Defining Roles:

Decide on the roles needed in your database environment (e.g., admin, regular user).

Execute the following SQL command to create a new role:

```
CREATE ROLE rolename;
```

Replace 'rolename' with the name of the role you want to create.

3. Granting Privileges:

Determine which privileges each role should have (e.g., SELECT, INSERT, UPDATE, DELETE).

Execute the following SQL command to grant privileges to a role or user:

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE LAB MANUAL (AIDS-II)

GRANT privilege1, privilege2 ON database.object TO user_or_role;

Replace 'privilege1', 'privilege2' with the specific privileges (e.g., SELECT, INSERT) you want to grant, 'database.object' with the database object (e.g., table, view), and 'user_or_role' with the username or role to which you want to grant privileges.

– Create UserA and UserB

CREATE USER UserA IDENTIFIED BY password1;

CREATE USER UserB IDENTIFIED BY password2;

– Grant CREATE SESSION to both users to allow them to log in

GRANT CREATE SESSION TO UserA;

GRANT CREATE SESSION TO UserB;

– Create a role named Manager

CREATE ROLE Manager;

– Assuming you have a table in a schema that you wish to grant access to, replace 'your_schema.your_table' with the actual schema and table name

– Grant SELECT, INSERT, and UPDATE privileges on a specific table to the Manager role

GRANT SELECT, INSERT, UPDATE ON your_schema.your_table TO Manager;

– Assign the Manager role to UserA

GRANT Manager TO UserA;

– (Optional) Check the privileges of UserA – this step requires querying the database's system views or being logged in as UserA

Example 1:

Suppose we want to create a role called 'staff' with SELECT privilege on a table called 'books' and grant this role to a user named 'john'. We can execute the following SQL commands:

CREATE ROLE staff;

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE LAB MANUAL (AIDS-II-II)

GRANT SELECT ON library.books TO staff;

GRANT staff TO 'john'@'localhost';

Conclusion:

Creating users, defining roles, and granting privileges are essential tasks in database management. By following these steps, database administrators can control access to database objects and ensure data security and integrity.

EXAMPLE 2:

– Create UserA and UserB

CREATE USER UserA IDENTIFIED BY password1;

CREATE USER UserB IDENTIFIED BY password2;

– Grant CREATE SESSION privilege to both users

GRANT CREATE SESSION TO UserA;

GRANT CREATE SESSION TO UserB;

– Create a role named Manager

CREATE ROLE Manager;

– Grant SELECT, INSERT, UPDATE, and DELETE privileges on the 'employees' table to the Manager role

GRANT SELECT, INSERT, UPDATE, DELETE ON hr.employees TO Manager;

– Assign the Manager role to UserA

GRANT Manager TO UserA;

– Create the 'employees' table in the 'hr' schema

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE LAB MANUAL (AIDS-II-II)

```
CREATE TABLE hr.employees (  
    employee_id NUMBER PRIMARY KEY,  
    first_name VARCHAR2(50),  
    last_name VARCHAR2(50),  
    email VARCHAR2(100),  
    hire_date DATE,  
    job_id VARCHAR2(50),  
    salary NUMBER  
);
```

– Insert sample data into the 'employees' table

```
INSERT INTO hr.employees (employee_id, first_name, last_name, email, hire_date, job_id,  
salary)
```

```
VALUES
```

```
(1001, 'John', 'Doe', 'john.doe@example.com', TO_DATE('2022-01-15', 'YYYY-MM-DD'),  
'Manager', 50000),
```

```
(1002, 'Jane', 'Smith', 'jane.smith@example.com', TO_DATE('2022-02-20', 'YYYY-MM-  
DD'), 'Engineer', 40000),
```

```
(1003, 'Alice', 'Johnson', 'alice.johnson@example.com', TO_DATE('2022-03-25', 'YYYY-  
MM-DD'), 'Analyst', 35000);
```

– Grant SELECT, INSERT, UPDATE, DELETE privileges on the 'employees' table to the 'Manager' role

```
GRANT SELECT, INSERT, UPDATE, DELETE ON hr.employees TO Manager;
```

Experiment3. Creating and altering tables for various relations in SQL using Integrity Constraints

Objective: Learn how to create and alter tables in SQL with integrity constraints to maintain data consistency.

Tasks:

Create a Table for Employees:

Create a table named employees with the following columns:

employee_id (Primary Key)

first_name

last_name

email

hire_date

job_id

salary

Alter Table Structure:

Add a new column named department_id to the employees table to store department information.

Define a foreign key constraint on the department_id column referencing a departments table (if available) or a predefined list of department IDs.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE LAB MANUAL (AIDS-II-II)

Create a Table for Employees:

```
CREATE TABLE employees (  
    employee_id INT PRIMARY KEY,  
    first_name VARCHAR(50),  
    last_name VARCHAR(50),  
    email VARCHAR(100) UNIQUE,  
    hire_date DATE,  
    job_id VARCHAR(50),  
    salary DECIMAL(10, 2) CHECK (salary > 0)  
);
```

Add Integrity Constraints:

- Adding a foreign key constraint on job_id
- Replace 'jobs' with the actual table name or list of job codes if available

```
ALTER TABLE employees
```

```
ADD CONSTRAINT fk_job_id FOREIGN KEY (job_id) REFERENCES jobs(job_id);
```

- Adding a unique constraint on email

```
ALTER TABLE employees
```

```
ADD CONSTRAINT unique_email UNIQUE (email);
```

Alter Table Structure:

- Adding a new column department_id

```
ALTER TABLE employees
```

```
ADD department_id INT;
```

- Adding a foreign key constraint on department_id
- Replace 'departments' with the actual table name or list of department IDs if available

```
ALTER TABLE employees
```

```
ADD CONSTRAINT fk_department_id FOREIGN KEY (department_id) REFERENCES  
departments(department_id);
```

Experiment 4. Implementing queries in SQL using

4.1 Insertion

4.2 Retrieval (operations like union - intersect - minus - in - exists - group by and having)

4.3 Updation

4.4 Deletion

DML Commands:

1. SELECT – retrieve data from the database
2. INSERT - insert data into a table
3. UPDATE - updates existing data within a table
4. DELETE – deletes all records from a table, the space for the records remain

Commands:

MySQL>Insert into Bus values('AP123', 'Hyderabad', 'Bangalore', 'Volvo');

MySQL>Insert into Bus values('AP234', 'Mumbai', 'Hyderabad', 'Semi-sleeper');

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE LAB MANUAL (AIDS-II-II)

Mysql> Select * from Bus;

```

C:\MySQL\bin\mysql.exe
mysql>
mysql> insert into bus6 values('AP123','HYDERABAD','BANGALORE','UOLUO');
Query OK, 1 row affected (0.03 sec)

mysql> insert into bus6 values('AP126','UIZAG','BANGALORE','SEMISLEEPER');
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM BUS6;
+-----+-----+-----+-----+
| busno | source | destination | coachtype |
+-----+-----+-----+-----+
| AP123 | HYDERABAD | BANGALORE | UOLUO |
| AP126 | UIZAG | BANGALORE | SEMISLEEPER |
+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> _

```

sql>Insert into Passenger values(82302, 'Smith',23, 'M', 'Hyderabad', '9982682829');

Mysql> Select * from Passenger;

PassportID	Name	Age	Sex	Address	Contact No
8939034	Smith	23	M	Hyderabad	983893023
9820023	John	24	M	Mumbai	983893093
8738939	Kavitha	22	F	Hyderabad	998383673

Mysql> Insert into Passenger_Ticket values('AP123',82302);

//Insert 5 or more records like-wise//

Mysql> Select * from Passenger_Ticket;

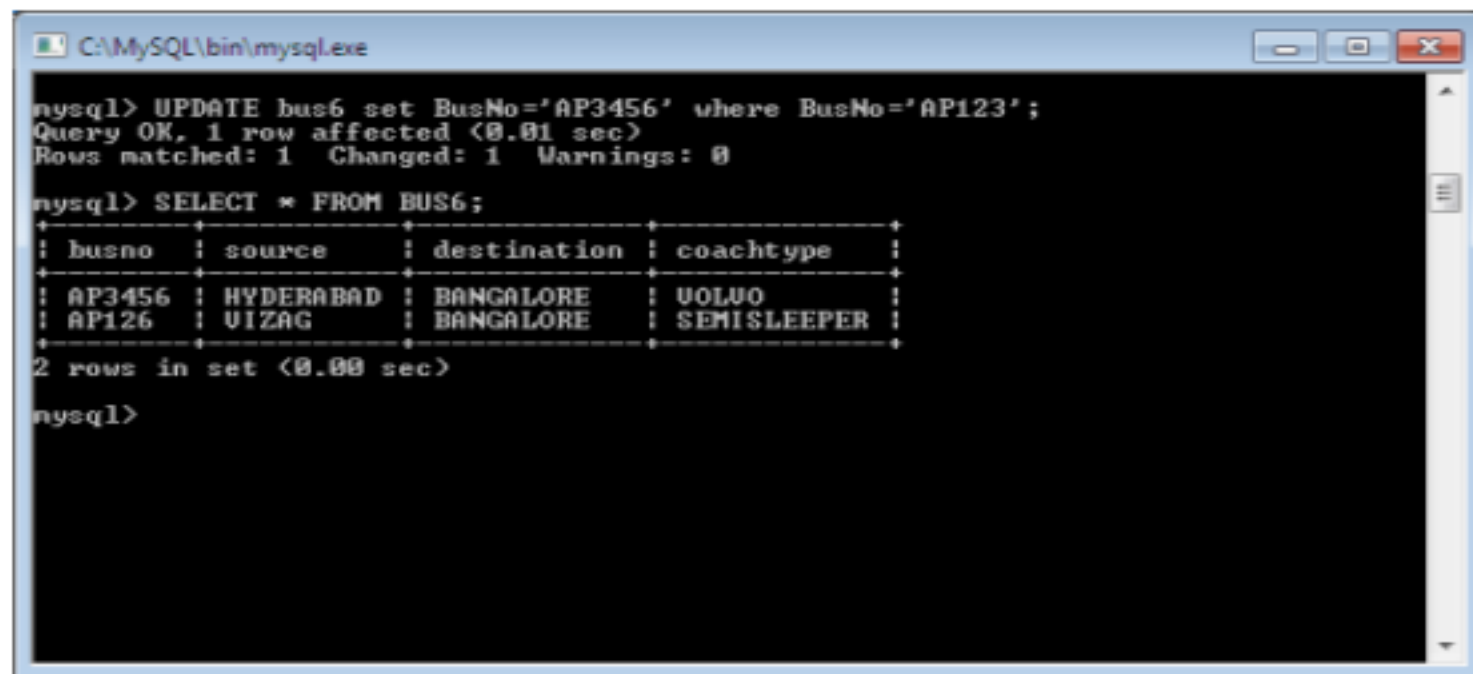
PassportID	TicketNo
8738939	453
5443243	332

Mysql> Insert into Ticket values(29823, 'AP123',82302, '21-03-2014', '4:00PM', '98202030334');

//Insert 5 or more records like-wise//

TicketNo	BusNo	PassportID	DOJ	Dept_time	Contact NO
29823	AP123	82302	21-03-2014	4 pm	9832434354
34353	AP234	32243	12-04-2014	5pm	9855645433

Mysql>UPDATE BUS set BusNo='AP3456' where BusNo='AP123';



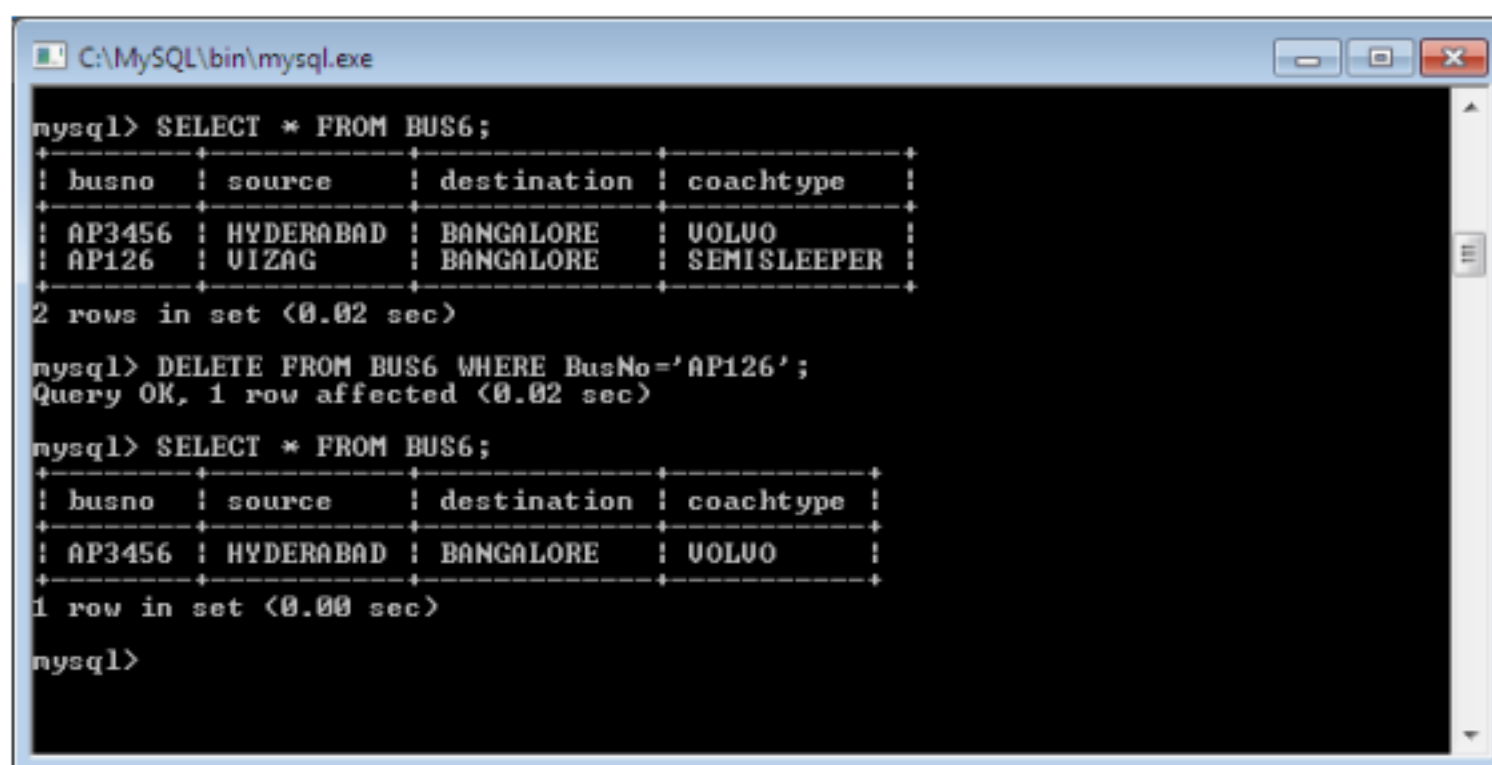
```
C:\MySQL\bin\mysql.exe

mysql> UPDATE bus6 set BusNo='AP3456' where BusNo='AP123';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM BUS6;
+-----+-----+-----+-----+
| busno | source | destination | coachtype |
+-----+-----+-----+-----+
| AP3456 | HYDERABAD | BANGALORE | VOLVO |
| AP126  | VIZAG    | BANGALORE  | SEMISLEEPER |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Mysql> DELETE FROM BUS WHERE BusNo='AP126';



```
C:\MySQL\bin\mysql.exe

mysql> SELECT * FROM BUS6;
+-----+-----+-----+-----+
| busno | source | destination | coachtype |
+-----+-----+-----+-----+
| AP3456 | HYDERABAD | BANGALORE | VOLVO |
| AP126  | VIZAG    | BANGALORE  | SEMISLEEPER |
+-----+-----+-----+-----+
2 rows in set (0.02 sec)

mysql> DELETE FROM BUS6 WHERE BusNo='AP126';
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM BUS6;
+-----+-----+-----+-----+
| busno | source | destination | coachtype |
+-----+-----+-----+-----+
| AP3456 | HYDERABAD | BANGALORE | VOLVO |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

5. Implementing the concepts of Rollback – commit, checkpoints and Views