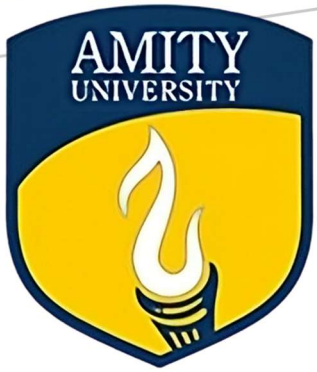


AMITY UNIVERSITY
PATNA

LAB 1

Aryan Baranwal

A45304821014

BCA VI 'A'

Submitted To: Dr. Naveen Kumar Singh

CRUD OPERATIONS

Problem Description

The goal of this project is to implement a basic console-based application that performs Create, Read, Update, and Delete (CRUD) operations on a database, specifically focusing on managing employee records. The application will provide an interactive interface for users to manipulate the employee data stored in a relational database management system (RDBMS) such as MySQL, PostgreSQL, or SQLite.

Functional Requirements:

1. **Create (Insertion):**
    - The application must allow users to add new employee records to the database. Each employee record consists of the following fields: ID, First Name, Last Name, Age, and Date of Birth (DOB). The application should prompt the user to enter these details.
2. **Read (Selection):**
    - The application should be capable of displaying employee records. It must support two types of read operations:
        - **View All:** Display all employee records stored in the database.
        - **View Specific:** Prompt the user to enter an employee ID and display the corresponding employee's details if found.
3. **Update:**
    - The application must enable users to update existing employee records. Users should be able to specify an employee ID and then choose which details to update (e.g., name, age, and/or DOB). The application should then prompt the user for the new values.
4. **Delete:**
    - Users should be able to delete an employee record from the database by specifying the employee's ID. The application should confirm the deletion.
5. **Exit:**
    - The application should offer an option to exit the program gracefully.

Design Description

The design of the CRUD operations application for managing Employee records encompasses several critical components, including system architecture, database design, class design, user interaction flow, and technical considerations. This structured approach ensures a well-organized and maintainable application.

System Architecture

The application is organized into two main packages to separate concerns and enhance maintainability:

- **bca.model:** Contains the business logic for performing CRUD operations directly with the database.
- **bca.drive:** Hosts the application's main method and handles user interactions through a console-based menu system.

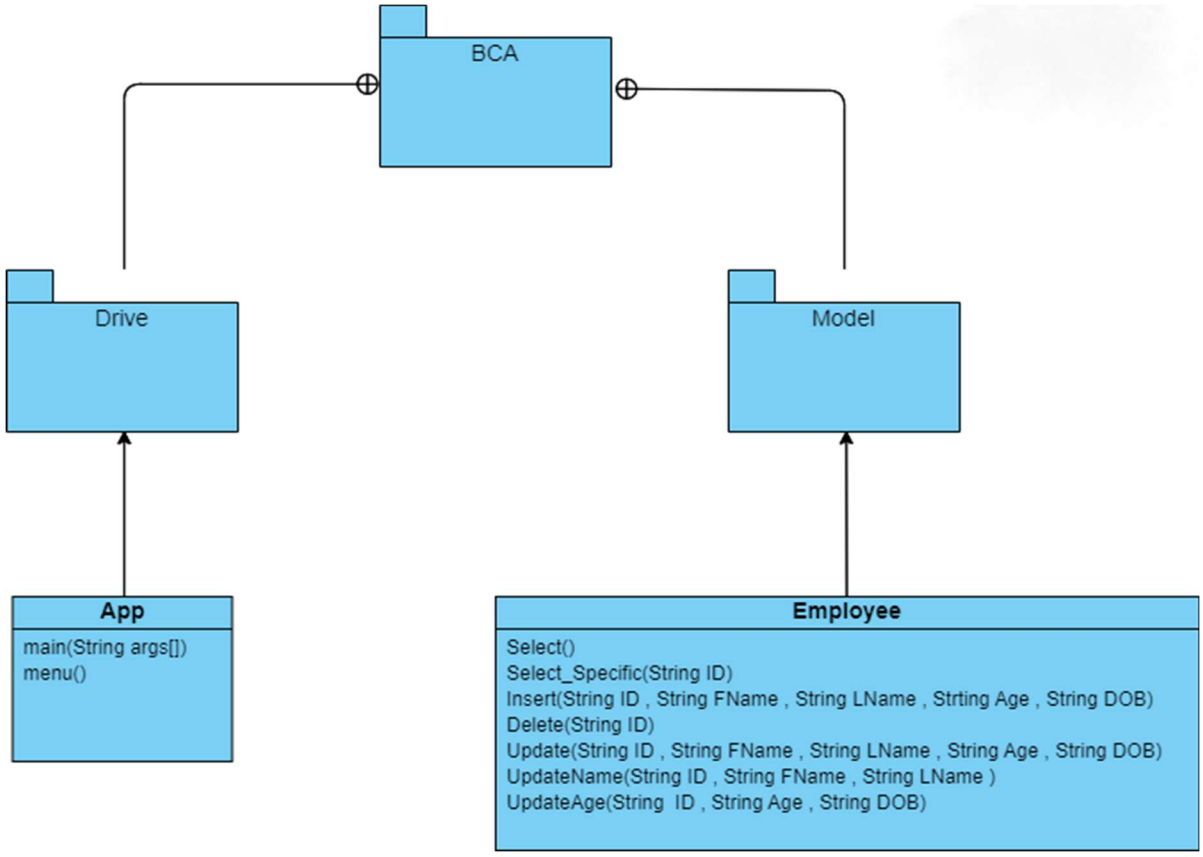
User Interaction Flow

The application follows a straightforward interaction flow:

1. **Start**: The application displays a menu with options for each CRUD operation and exit.
2. **User Input**: The user selects an option by entering the corresponding number.
3. **Process Request**: Depending on the user's choice, the application either performs a database operation (via the **Employee** class) or exits.
4. **Feedback**: The application provides feedback based on the action's outcome (e.g., success message, error message) and then returns to the main menu unless the user chooses to exit.
5. **Exit**: The application terminates when the user selects the exit option.

Class Design

The application's functionality is encapsulated within two main classes:



- **Employee (in bca.model)**:
    - **Purpose:** Manages database operations related to employee records, providing a layer of abstraction over direct database access for CRUD (Create, Read, Update, Delete) operations on employee data.
    - **Methods:**
        - **Employee():** Constructor method that initializes the database connection and ensures the Employee table exists by executing a table creation SQL statement if it doesn't already exist.
        - **Select():** Retrieves and displays all employee records from the database, printing the results to the console.
        - **Insert(String id, String fname, String lname, String age, String date):** Inserts a new employee record into the database using the provided parameters for the employee's ID, first name, last name, age, and date of birth.

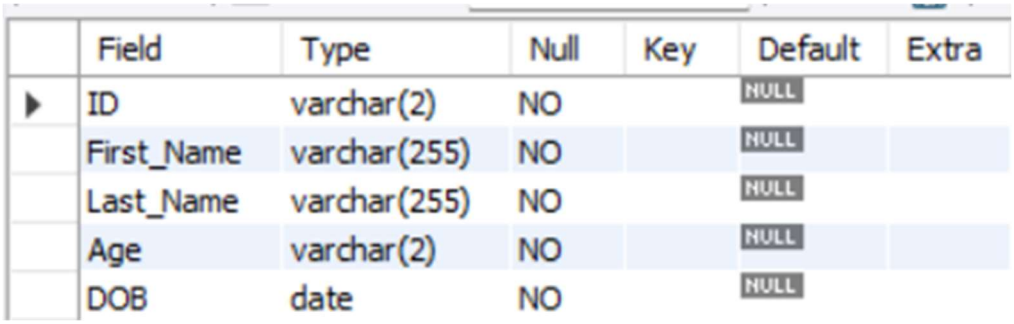
- **Update(String id, String fname, String lname, String age, String Date):** Updates an existing employee record identified by id with new values for first name, last name, age, and date of birth.
- **Delete(String id):** Deletes the employee record corresponding to the specified id from the database.
- **Select_Specific(String id):** Retrieves and displays a specific employee record identified by id, printing the result to the console.
- **UpdateName(String id, String fname, String lname):** Specifically updates only the first name and last name of an employee record identified by id.
- **UpdateAge(String id, String age, String date):** Specifically updates only the age and date of birth of an employee record identified by id.

- **App (in bca.drive)**:
  - **Purpose:** Acts as the user interface for the application, offering a command-line menu that allows users to interact with the employee database through various operations.
  - **Methods:**
    - **main(String[] args):** The entry point of the application. It initializes a **Scanner** object for reading user input and enters an infinite loop to display the menu and process user selections.
    - **menu(Scanner sc):** Displays a menu of options for the user to choose from, including viewing all employees, viewing a specific employee, inserting a new employee, updating an existing employee, deleting an employee, and exiting the application. It reads the user's choice and calls the corresponding method from the **Employee** class based on the selection.

Database Design

A crucial part of the application is its interaction with a database to store, retrieve, update, and delete student records. The database schema is designed as follows:

Table Schema: Student



Field	Type	Null	Key	Default	Extra
ID	varchar(2)	NO		NULL	
First_Name	varchar(255)	NO		NULL	
Last_Name	varchar(255)	NO		NULL	
Age	varchar(2)	NO		NULL	
DOB	date	NO		NULL	

1. **ID**
    - **Type:** VARCHAR(2)
    - **Constraints:** Primary Key
    - **Description:** Acts as the unique identifier for each employee. The choice of VARCHAR(2) suggests an intention to use alphanumeric IDs, though the length of 2 characters may be limiting for larger organizations. It's essential for uniquely identifying records in the Employee table.

2. **First_Name**
    - **Type:** VARCHAR(255)
    - **Constraints:** Not Null
    - **Description:** Represents the first name of the employee. The VARCHAR type with a length of 255 characters is used to accommodate first names of varying lengths and complexities, catering to a wide range of names.

3. **Last_Name**
    - **Type:** VARCHAR(255)
    - **Constraints:** Not Null
    - **Description:** Stores the last name or family name of the employee. Similar to **First_Name**, using VARCHAR(255) allows for a broad spectrum of last names to be stored, including those with hyphens, spaces, or apostrophes.

4. **Age**
    - **Type:** VARCHAR(2)
    - **Constraints:** Not Null
    - **Description:** Contains the age of the employee.

5. **DOB (Date of Birth)**
    - **Type:** DATE
    - **Constraints:** Not Null
    - **Description:** Stores the student's date of birth. The DATE type ensures that the data is stored in a standardized format (YYYY-MM-DD), facilitaϴng easy sorϴng, querying, and age calculaϴon based on this field.

Code

Employee.java

```
package bca.model;

import java.sql.*;

public class Employee
{
	public Employee() throws ClassNotFoundException, SQLException
	{
		String query = "Create Table if not exists Employee ("+
				"ID varchar(2) Primary key,"+
				"First_Name varchar(255) ,"+
				"Last_Name varchar(255)  ," +
				"Age varchar(2) ,"+
				"DOB date)";
		Class.forName("com.mysql.cj.jdbc.Driver");

		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/BCA","root","Aryan@04");

		Statement st = con.createStatement();

		st.executeUpdate(query);
	}


	public void Select() throws ClassNotFoundException, SQLException
	{
		Class.forName("com.mysql.cj.jdbc.Driver");

		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/BCA","root","Aryan@04");

		Statement st = con.createStatement();

		ResultSet rs = st.executeQuery("select * from Employee");
		System.out.println("Roll\tFirst Name\tLast Name\tAge\t\tDOB");
		while (rs.next())
		{

	System.out.println(rs.getString(1)+"\t"+rs.getString(2)+"\t\t"+rs.getString(3)+"\
t"+rs.getString(4)+"\t\t"+rs.getDate(5));
		}

		con.close();
	}
	public void Delete(String id) throws ClassNotFoundException, SQLException
	{
		Class.forName("com.mysql.cj.jdbc.Driver");

		Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/BCA","root","Aryan@04");


		String query = "Delete from Employee where id = ?";
		PreparedStatement pst =  con.prepareStatement(query);
		pst.setString(1, id);
		int affected_rows = pst.executeUpdate();

		if (affected_rows > 0)
		{
```

```
                System.out.println("Data Deleted");
            }

            con.close();
      }
      public void Insert(String id , String fname , String lname , String age , String
date) throws ClassNotFoundException, SQLException
      {
            Class.forName("com.mysql.cj.jdbc.Driver");

            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/BCA","root","Aryan@04");

            String query = "Insert into Employee values (? , ? , ? , ? ,?)";
            PreparedStatement pst = con.prepareStatement(query);
            pst.setString(1, id);
            pst.setString(2, fname);
            pst.setString(3, lname);
            pst.setString(4, age);
            pst.setString(5, date);

            int affected_rows = pst.executeUpdate();

            if (affected_rows > 0)
            {
                  System.out.println("Data Inserted");
            }
            con.close();
      }
      public void Update(String id , String fname , String lname , String age , String
Date) throws ClassNotFoundException, SQLException
      {
            Class.forName("com.mysql.cj.jdbc.Driver");

            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/BCA","root","Aryan@04");

            String query = "Update Employee set First_Name = ? , Last_Name = ? , Age =
? , Dob = ? where ID = ?";
            PreparedStatement pst = con.prepareStatement(query);
            pst.setString(1, fname);
            pst.setString(2, lname);
            pst.setString(3, age);
            pst.setString(4, Date);
            pst.setString(5, id);

            int affected_rows = pst.executeUpdate();

            if (affected_rows > 0)
            {
                  System.out.println("Data Updated");
            }
            con.close();
      }
      public void UpdateName(String id , String fname , String lname ) throws
ClassNotFoundException, SQLException
      {
            Class.forName("com.mysql.cj.jdbc.Driver");

            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/BCA","root","Aryan@04");

            String query = "Update Employee set First_Name = ? , Last_Name = ? where ID
= ?";
            PreparedStatement pst = con.prepareStatement(query);
```

```
            pst.setString(1, fname);
            pst.setString(2, lname);
            pst.setString(3, id);

            int affected_rows = pst.executeUpdate();
            if (affected_rows > 0)
            {
                  System.out.println("Data Updated");
            }
            con.close();
      }
      public void UpdateAge(String  id , String age , String date) throws SQLException,
ClassNotFoundException
      {
            Class.forName("com.mysql.cj.jdbc.Driver");

            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/BCA","root","Aryan@04");

            String query = "Update Employee set Age = ? , DOB = ? where ID = ?";
            PreparedStatement pst = con.prepareStatement(query);
            pst.setString(1, age);
            pst.setString(2, date);
            pst.setString(3,id);

            int affected_rows = pst.executeUpdate();
            if (affected_rows > 0)
            {
                  System.out.println("Data Updated");
            }
            con.close();
      }
      public void Select_Specific(String roll) throws ClassNotFoundException,
SQLException
      {
            Class.forName("com.mysql.cj.jdbc.Driver");

            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/BCA","root","Aryan@04");
            String query = "select * from Employee where id = ?";
            PreparedStatement pst = con.prepareStatement(query);
            pst.setString(1, roll);

            ResultSet rs = pst.executeQuery();
            System.out.println("Roll\tFirst Name\tLast Name\tAge\t\tDOB");
            while (rs.next())
            {

      System.out.println(rs.getString(1)+"\t"+rs.getString(2)+"\t\t"+rs.getString(3)+"\
t"+rs.getString(4)+"\t\t"+rs.getDate(5));
            }

            con.close();
      }
}
```

App.java

```
package bca.drive;

import java.sql.*;
import java.util.Scanner;

import bca.model.Employee;

public class App
{
    public static void main(String[] args) throws ClassNotFoundException, SQLException
    {
        Scanner sc = new Scanner(System.in);
        while (true)
        {
            menu(sc);
        }

    }
    public static void menu(Scanner sc) throws ClassNotFoundException, SQLException
    {

        Employee s = new Employee();
        System.out.println("Enter Your Choice :\n1 -> View All\n2 -> View Specific\n3 -
> Insert\n4 -> Delete\n5 -> Update\n6 -> Exit");
        int ch = sc.nextInt();
        switch (ch)
        {
            case 1:
            {
                s.Select();
                break;
            }
            case 2:
            {
                System.out.println("Enter The ID whose Details to be seen");
                String id = sc.next();
                s.Select_Specific(id);
                break;
            }
            case 3:
            {
                System.out.print("Enter The Details of the Employee\nID = ");
                String roll = sc.next();
                System.out.print("First Name = ");
                String fname = sc.next();
                System.out.print("Last Name = ");
                String lname = sc.next();
                System.out.print("Age = ");
                String age = sc.next();
                System.out.print("DOB (YYYY-MM-DD) = ");
```

```
                String date = sc.next();
                s.Insert(roll, fname, lname , age, date);
                break;
            }
            case 4:
            {
                System.out.println("Enter The Id whose Details needs to be Deleted");
                String id = sc.next();
                s.Delete(id);
                break;
            }
            case 5:
            {
                System.out.println("Enter The ID of the Employee Whose Data needs to be
Updated");
                String id = sc.next();
                System.out.println("Present Details");
                s.Select_Specific(id);
                System.out.println("Select What Data you need to change\n1 -> Name\n2 -
> Age(Date As well)\n3 -> Multiple Data");
                int c = sc.nextInt();
                if(c==1)
                {
                    System.out.print("Enter First Name : ");
                    String fname = sc.next();
                    System.out.print("Enter Last Name : ");
                    String lname = sc.next();
                    s.UpdateName(id, fname, lname);
                }
                else if(c==2)
                {
                    System.out.print("Enter Age : ");
                    String ag = sc.next();
                    System.out.print("Date Of Birth (YYYY-MM-DD) : ");

                    String dOB = sc.next();
                    s.UpdateAge(id,ag,dOB);
                }
                else if(c==3)
                {
                    System.out.print("Enter First Name : ");
                    String fname = sc.next();
                    System.out.print("Enter Last Name : ");
                    String lname = sc.next();
                    System.out.print("Enter Age : ");
                    String ag = sc.next();
                    System.out.print("Date Of Birth (YYYY-MM-DD) : ");
                    String dOB = sc.next();
                    s.Update(id, fname, lname, ag, dOB);
                }
                break;
            }
```

```
            case 6:
            {
                System.out.println("System Exited");
                System.exit(0);
            }
            default:
                System.out.println("Please Enter  a Valid Option");
        }
    }
}
```

Input/Output

Main

```
Enter Your Choice :
1 -> View All
2 -> View Specific
3 -> Insert
4 -> Delete
5 -> Update
6 -> Exit
```

View Table

```
1
Roll    First Name      Last Name       Age             DOB
1       Aryan           Baranwal        20              2003-08-04
2       Harshit         Kumar   20              2003-03-03
3       Keshav          Kant_Mishra     20              2003-12-12
```

View Specific Data

```
2
Enter The ID whose Details to be seen
1
Roll    First Name      Last Name       Age             DOB
1       Aryan           Baranwal        20              2003-08-04
```

Insert Data

```
3
Enter The Details of the Employee
ID = 1
First Name = Aryan
Last Name = Baranwal
Age = 20
DOB (YYYY-MM-DD) = 2003-08-04
Data Inserted
```

Update Complete Data

```
5
Enter The ID of the Employee Whose Data needs to be Updated
2
Present Details
Roll    First Name      Last Name       Age             DOB
2       Harshit         Kumar   20              2003-03-03
Select What Data you need to change
1 -> Name
2 -> Age(Date As well)
3 -> Multiple Data
3
Enter First Name : Harshit
Enter Last Name : Agarwal
Enter Age : 23
Date Of Birth (YYYY-MM-DD) : 2004-06-02
Data Updated
```

Update Specific Data

- ### Update Name

```
5
Enter The ID of the Employee Whose Data needs to be Updated
2
Present Details
Roll    First Name      Last Name       Age             DOB
2       Harshit         Agarwal 23              2004-06-02
Select What Data you need to change
1 -> Name
2 -> Age(Date As well)
3 -> Multiple Data
1
Enter First Name : Harshit
Enter Last Name : Kumar
Data Updated
```

- ### Update Age

```
5
Enter The ID of the Employee Whose Data needs to be Updated
2
Present Details
Roll    First Name      Last Name       Age             DOB
2       Harshit         Kumar   23              2004-06-02
Select What Data you need to change
1 -> Name
2 -> Age(Date As well)
3 -> Multiple Data
2
Enter Age : 20
Date Of Birth (YYYY-MM-DD) : 2003-03-03
Data Updated
```

Delete Data

```
4
Enter The Id whose Details needs to be Deleted
2
Data Deleted
```

Exit

```
6
System Exited
```