COMPS203F

Topic 06: Java and Database

Kelvin Lee

PostgreSQL

- PostgreSQL: a free and advanced database software which can be downloaded from
  `http://www.postgresql.org/`
- After downloading, please install it
- You will be asked the password of "postgres", use "myPass19" to avoid forgetting (you can use any password).
- As part of the installation, pgAdmin 4 can be used as the GUI for the database. We can write and run our SQL statements (explained later) using its editor.

SQL: create

- SQL a relational database language that is easy to use
- Relational databases stores data in two-dimensional tables with rows and columns.
- For example,

Student ID	Name	Weight
11223344	Ada	45
11223355	Ben	66.5

- To create a table like this, we can use the SQL statement

```
create table student (
  studentID char(8) primary key,
  name varchar(50),
  weight numeric(4,1)
)
```

- A **primary key** uniquely identifies a row of data
- numeric(4,1) can store 999.9, where 4 is the no. of digits and 1 is the no. of decimal places. To store integers, "int" can be used.

SQL: insert

- After creation, the table is empty. To insert data into it, you can use the statements:

```
insert into student values ('11223344', 'Ada', 45);
insert into student values ('11223355', 'Ben', 66.5);
```

- Strings are enclosed in single quotes while no quotes are needed for numbers
- SQL is **case insensitive** (except inside strings)
- SQL statements can be split into different lines

SQL: select

- To display all the data, we can use:

```
select * from student;
```

Student ID	Name	Weight
11223344	Ada	45
11223355	Ben	66.5

- To display some columns (e.g., the student ID and the name):

```
select studentID, name from student;
```

Student ID	Name
11223344	Ada
11223355	Ben

SQL: select

- To display some rows (e.g., just for ID 11223344):

```
select *
  from student
  where studentID='11223344';
```

Student ID	Name	Weight
11223344	Ada	45

- Use "and", "or", "not" for compound conditions:

```
select *
   from student
   where name='Ben' and weight > 50;
```

Student ID	Name	Weight
11223355	Ben	66.5

SQL: select

- To display all the data in descending order of weight (without "desc", data is displayed in ascending order):

```
select *
  from student
  order by weight desc;
```

Student ID	Name	Weight
11223355	Ben	66.5
11223344	Ada	45

- To find the average weight of students:

```
select avg(weight) as meanWeight
  from student;
```

meanWeight
45

- Other functions available: count, sum, max and min

SQL: select

- Searching for students with name containing "e":

```
select *
  from student
  where name like '%e%';
```

Student ID	Name	Weight
11223355	Ben	66.5

where "%" matches any number of characters (including 0) and "_" matches exactly one character.

SQL: update

- To update the weight of Ben to 68, we can use:

```
update student set weight=68
  where studentID='11223355';
```

- To update the weight of Ben to 68 and his name to Benson, we can use:

```
update student set weight=68, name='Benson'
  where studentID='11223355';
```


SQL: delete

- To delete the row of data of Ben, we use:

```
delete from student
  where studentID='11223355';
```

- To delete **ALL** the rows of the table `student`:

```
delete from student;
```


SQL: References

- To learn more, see:
  - https://www.postgresql.org/docs/current/sql.html
  - https://en.wikipedia.org/wiki/SQL

Exercise

- Using SQL, create a table "item" containing supermarket items with ID, name and price
- insert the data "0011" as the ID, "Orange" as the name and 6.9 as the price.
- Also insert the data "1234", "Apple", 5.5
- list the items with price > 6
- find the items with ID starting with "1"
- change the price of Apple to 5.9
- delete the item with ID "0011"

JDBC

- To use PostgreSQL with Java, we need a JDBC (Java DataBase Connectivity) driver.
- It can be downloaded separately
  https://jdbc.postgresql.org/
- It is a jar file and needs to be included in our CLASSPATH so that Java can find it.

Import and driver

- To use Java to connect to database, the following import statement is needed

```
import java.sql.*;
```

- To load the JDBC driver of PostgreSQL, call loadDriver()

```
public class TestDB {
  public void loadDriver() {
    try {
      Class.forName("org.postgresql.Driver");
    } catch (ClassNotFoundException e) {
      System.out.println("Problem: "+e.getMessage());
    }
  }
  // put other methods here
}
```


Connect to database

- To connect to the database `postgres` with user ID `postgres` and password `myPass19`:

```
private Connection conn = null; // attribute
public void connectDB() {
  try {
    conn = DriverManager.getConnection(
             "jdbc:postgresql://localhost:5433/postgres",
             "postgres", "myPass19");
  } catch (SQLException e) {
    System.out.println("Connect Problem: "+e.getMessage());
  }
}
```


Create Table

```
private PreparedStatement pStatement = null; // attribute
public void createTable() {
  try {
    String sql = "create table student ( "
               + "studentID char(8) primary key, "
               + "name varchar(50), "
               + "weight numeric(4,1) );";
    pStatement = conn.prepareStatement(sql);
    pStatement.executeUpdate();
  } catch (SQLException e) {
    System.out.println("Create Problem: "+e.getMessage());
  }
}
```


Insert Data: part 1

```
public void insertData() {
  try {
    String sql = "insert into student values (?,?,?);";
    pStatement = conn.prepareStatement(sql);
    setData("11223344", "Ada", 45);
    pStatement.executeUpdate();
    setData("11223355", "Ben", 66.5);
    pStatement.executeUpdate();
  } catch (SQLException e) {
    System.out.println("Insert Problem: "+e.getMessage());
  }
}
```


Insert Data: part 2

```
public void setData(String studentID,
                    String name, double weight) {
  try {
    pStatement.setString(1, studentID);
    pStatement.setString(2, name);
    pStatement.setDouble(3, weight);
  } catch (SQLException e) {
    System.out.println("Set Data Problem: "+e.getMessage());
  }
}
```

Retrieve Data: by ID

```
private ResultSet resultSet = null; // attribute
public void selectByID(String studentID) {
  try {
    String sql = "select * from student where studentID=?;";
    pStatement = conn.prepareStatement(sql);
    pStatement.setString(1, studentID);
    resultSet = pStatement.executeQuery();
    while (resultSet.next()) {
      System.out.println(resultSet.getString("studentID") + ", "
                         + resultSet.getString("name") + ", "
                         + resultSet.getDouble("weight"));
    }
  } catch (SQLException e) {
    System.out.println("Select Problem: "+e.getMessage());
  }
}
```


Retrieve Data: all

```
public void selectAllData() {
  try {
    String sql = "select * from student;";
    pStatement = conn.prepareStatement(sql);
    resultSet = pStatement.executeQuery();
    while (resultSet.next()) {
      System.out.println(resultSet.getString("studentID") + ", "
                         + resultSet.getString("name") + ", "
                         + resultSet.getDouble("weight"));
    }
  } catch (SQLException e) {
    System.out.println("Select All Problem: "+e.getMessage());
  }
}
```


Delete Data: by name

```
public void deleteByName(String name) {
  try {
    String sql = "delete from student where name=?;";
    pStatement = conn.prepareStatement(sql);
    pStatement.setString(1, name);
    pStatement.executeUpdate();
  } catch (SQLException e) {
    System.out.println("Delete Problem: "+e.getMessage());
  }
}
```


Delete Data: all

```
public void deleteAllData() {
  try {
    String sql = "delete from student;";
    pStatement = conn.prepareStatement(sql);
    pStatement.executeUpdate();
  } catch (SQLException e) {
    System.out.println("Delete by ID Problem: "+e.getMessage());
  }
}
```


Update Data

```
public void updateName(String studentID, String newName) {
  try {
    String sql = "update student set name=? "
                 +"where studentID=?;";
    pStatement = conn.prepareStatement(sql);
    pStatement.setString(1, newName);
    pStatement.setString(2, studentID);
    pStatement.executeUpdate();
  } catch (SQLException e) {
    System.out.println("Update Problem: "+e.getMessage());
  }
}
```


Close Connection

```
public void closeDB() {
  try {
    conn.close();
  } catch (SQLException e) {
    System.out.println("Close Problem: "+e.getMessage());
  }
}
```

Testing code: page 1

```
public static void main(String[] args) {
  TestDB myDB = new TestDB();
  System.out.println("load driver and connect...");
  myDB.loadDriver();
  myDB.connectDB();
  System.out.println("create table...");
  myDB.createTable();
  //System.out.println("delete all data...");
  //myDB.deleteAllData();
  System.out.println("insert all data...");
  myDB.insertData();
  myDB.selectAllData();
  System.out.println("select by ID...");
  myDB.selectByID("11223344");
```


Testing code: page 2

```
  System.out.println("delete Ada...");
  myDB.deleteByName("Ada");
  myDB.selectAllData();
  System.out.println("delete Ben...");
  myDB.deleteByName("Ben");
  myDB.selectAllData();
  System.out.println("insert all data...");
  myDB.insertData();
  myDB.selectAllData();
  System.out.println("update name...");
  myDB.updateName("11223344", "Alice");
  myDB.selectAllData();
}
```


Output: page 1

```
load driver and connect...
create table...
Create Problem: ERROR: relation "student" already exists
delete all data...
insert all data...
11223344, Ada, 45.0
11223355, Ben, 66.5
select by ID...
11223344, Ada, 45.0
```


Output: page 2

```
delete Ada...
11223355, Ben, 66.5
delete Ben...
insert all data...
11223344, Ada, 45.0
11223355, Ben, 66.5
update name...
11223355, Ben, 66.5
11223344, Alice, 45.0
```


Exercise

Q1: Write a method `storeStudent(Student aStudent)` to store the data of `aStudent` into table `student` of the database.

```
public class Student {
  private String studentID;
  private String name;
  private double weight;

  public void setStudentID(String id) { studentID = id; }
  public String getStudentID() { return studentID; }
  public void setName(String aName) { name = aName; }
  public String getName() { return name; }
  public void setWeight(double aWeight) { weight = aWeight; }
  public double getWeight() { return weight; }
}
```


Exercise (cont'd)

Q2: Write a method `readStudent(Student id)` to return a `Student` object with matching ID in database. Return null if there is no match.

Q3: Write a method `readStudents(String str)` to return a list of `Student` objects with part of the name matching `str` in database. Return an empty list if there is no match.