Portfolio 2

Daniel Vestmar Norén: 65246

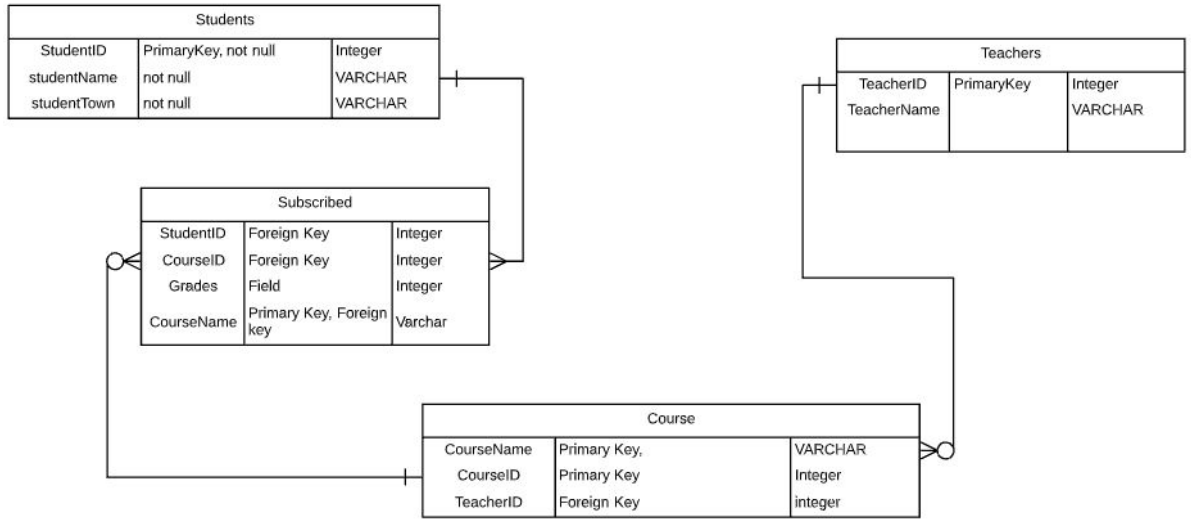
Frederik Røikjær Jensen: 66454

Kathrine Kauffeld Andreasen: 65204

Mikkel Kjær: 66370

Tobias Mikkel Schleiss-Andreassen: 66715

Porteføljen omhandler databasedesign; hvordan man opretter, håndterer og interagerer i den. Vi har fået udleveret et datasæt, hvori der indgår 10 studerendes navne, bynavn, kursus og karakterer. Der indgår også to undervisere i datasættet med tilhørende kursus, hvor de er ansvarlige. Vi har designet en database for de studerendes kursusregistrering og karakterer med videre. I vores entity relation (ER) diagram kan man se sammenhængene.



Figur 1: *ER diagram*

I figur 1 ses gruppen ER diagrammet, som viser hvorledes vores databases forskellige entities eller tables, relaterer til hinanden. Diagrammet viser bl.a. også at vi har 4 entities eller tables kaldet; Students, Subscribed, Courses og Teachers. Disse entities har forskellige attributter, der gør dem unikke, hvorimod nogle af disse tables også deler attributter. Subscribed deler f.eks. StudentID og CourseID med de to andre tables kaldet: Students og Course, hvilket gør

det muligt at samle de forskellige Values, som Integer, VARCHAR osv, i den samme entity eller table.

Ud fra denne model har vi forsøgt at opstille en database, som kan findes i mappen, hvori dette dokument ligger, som hedder: "Portifolio2.db".

Gennem Java biblioteket: JDBC tilslutter vi Java-programmet til vores SQLite database kaldet: "Portifolio2".

```
String url = "jdbc:sqlite:C:\\Users\\Frede\\Documents\\SoftwareDev\\Portifolio2.db";
conn = getConnection(url);
```

Dette gøres gennem et Try and Catch statement, som gør det muligt for os at opsamle nogle data hvis noget ved forbindelsen skulle gå galt.

```
catch (SQLException e) {
      e.printStackTrace();
  } finally {
      if (conn != null) {
          try {
              conn.close();
          } catch (SQLException e) {
              e.printStackTrace();
          }
  }
```

Vi har implementeret et user input gennem 2 if statements, hvoraf brugeren kan indtaste et StudentID for en vilkårlig studerende og dermed få outputtet for gennemsnittet af den studerendes karakterer. Derudover kan brugeren også vælge at indtaste et vilkårligt CourseID, hvoraf programmet kalder kursets karaktergennemsnit. Dette sker gennem en SQL ordre, som argument, der opsamler de ønskede data fra databasen.

```
if (ChooserScanner == 1) {
    System.out.println("Which Student ID do you want the average grade for?");
    scanner = new Scanner(System.in);
    int StudentScanner = scanner.nextInt();
ql string with the read station
    sql = "SELECT avg(Grades) FROM Subscribed WHERE StudentID ='" + StudentScanner + "'";
    ResultSet as = stmt.executeQuery(sql);
    AverageStudentGrade(as);
}
```

```
static public void AverageStudentGrade(ResultSet res)
        throws SQLException {
    if (res == null)
        System.out.println("No Data Found");
    while (res != null & res.next()) {
        // String Course = res.getString("CourseName");
        float Grades = res.getFloat( columnLabel: "avg(Grades)");

                //if you want more attributes you can uncomment these methods.
        // int StudentID = res.getInt("StudentID");
        // int CourseID = res.getInt("CourseID");
        // System.out.println(StudentID + " " + Grades + " " + Course + " " + CourseID);
        System.out.println(Grades);
    }
```

Til slut printer vi også vores Joined table af Student og Subscribed gennem INNER JOIN funktionen i SQL, og et preparedstatement, så vi ikke behøver at printe begge tables to gange og så vores database er mere sikker i forhold til "sql injection attacks".

```
String ptripsql= "SELECT D1.StudentID,D1.StudentName, " +
        "D2.CourseName, D2.Grades" +
        "  FROM Student as D1 " +
        "INNER join Subscribed as D2 on D1.StudentID =D2.StudentID ";
PreparedStatement pTripStmt = conn.prepareStatement(ptripsql);
rs = pTripStmt.executeQuery();
StudentSUBjoin(rs);
```

```
static public void StudentSUBjoin(ResultSet res)
        throws SQLException {
    if (res == null)
        System.out.println("No records found");
    while (res != null & res.next()) {
        String StudentName = res.getString( columnLabel: "Studentname");
        int Grades = res.getInt( columnLabel: "Grades");
        String CourseName = res.getString( columnLabel: "Coursename");
        int StudentID = res.getInt( columnLabel: "StudentID");
        System.out.println("Student name: "
                + StudentName +
                ", Grade: "
                +Grades+
                ", Course: "
                + CourseName +", StudentID: " + StudentID);
    }
}
```