Introduction

For the implementation, we are going to use Eclipse as the designate IDE and SQLite as the database to develop SCRS.

Eclipse

First of all, you need to download the latest java IDE Eclipse from:
https://eclipse.org/downloads/

SQLite Database Installation

Before we start using SQLite in our Java programs, we need to make sure that we have SQLite JDBC Driver and Java set up on the machine. You can check Java tutorial for Java installation on your machine. Now, let us check how to setup SQLite JDBC driver.

- Download latest version of sqlite-jdbc-(VERSION).jar from sqlite-jdbc repository.
- Add downloaded jar file sqlite-jdbc-(VERSION).jar in your class path, or you can use it along with -classpath option as explained below in examples.

Following section assumes you have little knowledge about Java JDBC concepts. If you don't, then it is suggested to spent half an hour with JDBC Tutorial to become comfortable with concepts explained below.

SQL Tutorial

If you are new to sql, here is an excellent SQL tutorial from SQLite website that can help you pick up basic sql queries in a short time.

Select Tutorial:
http://www.tutorialspoint.com/sqlite/sqlite_select_query.htm
Update Tutorial:
http://www.tutorialspoint.com/sqlite/sqlite_update_query.htm
Delete Tutorial:
http://www.tutorialspoint.com/sqlite/sqlite_delete_query.htm

This website also provides more sql tutorials if you are interested in them.

Database Operations

The database interfaces we provided to you includes “Query Interface”, “Update Interface”, “Insert Interface”, and “Delete Interface”. We designed the Update, Insert, and Delete interface in a way that can prevent malicious SQL injection. These three interfaces take three parameters (sql command, arraylist of values, arraylist of data type).

The following shows you how to prepare data for using these interfaces:

```
// Sql Command, actual value are replaced by question marks
String sqlCmd = "INSERT INTO STUDENTANDCOURSE (COURSEID,
GRADINGTYPE, COURSETERM, STUDENTID) VALUES (?, ?, ?, ?);";

// Declarations of these two ArrayList
ArrayList<String> coursePropertyValue = new ArrayList<String>();
ArrayList<Constants.PrimitiveDataType> coursePropertyType = new
ArrayList<Constants.PrimitiveDataType>();

// For each question mark in the sqlCmd, we store the the actual value in the form,
// and its original data type respectively
coursePropertyValue.add(Integer.toString(courseID));
coursePropertyType.add(Constants.PrimitiveDataType.INT);
```

Database Table Abstractions

Student Table:

```
CREATE TABLE STUDENT
(ID INT PRIMARY KEY     NOT NULL,                // Student ID
FIRSTNAME      TEXT    NOT NULL,                 // Firstname
LASTNAME       TEXT    NOT NULL,                 // Lastname
DATEOFBIRTH    DATE    NOT NULL,                 // Date of birth, “sqldate” type
X500ACCOUNT    CHAR(50) NOT NULL,
X500PASSWORD   CHAR(20) NOT NULL,
TYPE           CHAR(10) NOT NULL,
GENDER         CHAR(10),
ADVISOR        CHAR(20),
PLAN           CHAR(30) NOT NULL,                // Undergraduate, Master, or PHD
CREDITS        INT      NOT NULL,
DEPARTMENT     CHAR(50) NOT NULL);
```

Administrator Table:

```
CREATE TABLE ADMINISTRATOR
(ID INT PRIMARY KEY     NOT NULL,                // Admin ID
FIRSTNAME      TEXT    NOT NULL,
LASTNAME       TEXT    NOT NULL,
DATEOFBIRTH    DATE    NOT NULL,
X500ACCOUNT    CHAR(50) NOT NULL,
X500PASSWORD   CHAR(20) NOT NULL,
GENDER         CHAR(10),
DEPARTMENT     CHAR(50) NOT NULL);
```

Instructor Table:

```
CREATE TABLE INSTRUCTOR
ID INT PRIMARY KEY     NOT NULL,
FIRSTNAME      TEXT    NOT NULL,
LASTNAME       TEXT    NOT NULL,
DATEOFBIRTH    DATE    NOT NULL,
X500ACCOUNT    CHAR(50) NOT NULL,
X500PASSWORD   CHAR(20) NOT NULL,
GENDER         CHAR(10),
TITLE          CHAR(20),                          // Associate Professor, Professor
SALARY         INT(1),
DEPARTMENT     CHAR(50) NOT NULL);
```

Course Table:

```
CREATE TABLE COURSE
(ID INT PRIMARY KEY     NOT NULL,
NAME           CHAR(50) NOT NULL,
CREDITS        INT NOT NULL,
INSTRUCTOR     CHAR(100) NOT NULL,                // Assume each course has only
                                                  // one instructor
FIRSTTIME      DATE NOT NULL,                     // The date of the first class
SECONDTIME DATE NOT NULL,                         // The date of the last class
STARTTIME      DATE NOT NULL,                     //  E.g. 9:00am
ENDTIME        DATE NOT NULL,                     //  E.g. 11:00am
CLASSDAYS      CHAR(50) NOT NULL,                 //  E.g. Tu, Th
LOCATION       CHAR(100) NOT NULL,
TYPE           CHAR(20) NOT NULL,                 //   on Campus or Unite
PREREQUISITE   TEXT,
DESCRIPTION    TEXT NOT NULL,
DEPARTMENT     CHAR(50) NOT NULL);
```

StudentAndCourse Table:

```
CREATE TABLE STUDENTANDCOURSE
(ID INTEGER PRIMARY KEY AUTOINCREMENT    NOT NULL,
COURSEID  INT REFERENCES COURSE(ID) ON UPDATE CASCADE,
GRADINGTYPE CHAR(10) NOT NULL,                        //  A/F, S/N, AUD
COURSETERM CHAR(20) NOT NULL,
STUDENTID INT REFERENCES STUDENT(ID) ON UPDATE CASCADE);
```

InstructorAndCourse Table:

```
CREATE TABLE INSTRUCTORANDCOURSE
(ID INTEGER PRIMARY KEY AUTOINCREMENT    NOT NULL,
COURSEID  INT REFERENCES COURSE(ID) ON UPDATE CASCADE,
INSTRUCTORID INT REFERENCES STUDENT(ID) ON UPDATE CASCADE);
```