# Design document

1. Procedures and functions
    a) Show tables:
        i. In order to print the content of each table, we can use cursor to iterate each table and use dbms_output.put_line to combine the columns of each tuple together.
        ii. But, as we need to redirect the sql output to java output, we need to either use function or procedure with OUT properties to transmit the data from sql package to java. In our code, we use procedure with OUT parameters.
        iii. In the help of type “SYS_REFCURSOR”, which we can register in the java, and retrieve it after execution of SQL procedure, we successfully print out all content in each table in certain format.
    b) Add student
        i. By easily transfer each parameter of new tuple into SQL procedure; we can insert a new student into students table.
    c) Show all classes of a certain student
        i. This one combine a) and b), which need an IN parameter and an OUT parameter to communication between SQL and java. Notice that there are two “SYS_REFCURSOR”, which one of them is for checking student validation while the other is for classes taken by this student.
    d) Show all students taking a certain class
        i. Same as c), only some differences in the content.
    e) Delete a student
        i. Identical with b) except that this would use a trigger to delete all enrollment relations of the deleted student.
    f) Prerequisite
        i. To accomplish this function, we need to recursively call a procedure or a java functions so that, we can collect all prerequisite classes in multiple levels.
        ii. But in order to use the tuples’ contents in java, we choose to create a helper function which would recursively call itself.
        iii. Also, we arrange the places for creation and close of “OracleResultSet” and “CallableStatement” in the main function, so that there is only one open and close for each SQL object.
    g) Enroll a new class to a student
        i. To enroll a class to a student, we first test the validation of classid, sid and make sure that the enrollment relation has not been there yet.
        ii. The size of the enroll class need to add one, which is done in a trigger. We also make sure that enrolling this student would not make the size of class exceed the limit.
        iii. To make sure that a student can only take no more than 4 classes in the same year and same semester, we iterate the classes table which have the same year and

semester with the new class and count the number. If the count number is smaller than 4, we can enroll it, otherwise, we would reject it.

iv. To make sure a student can only take a course after he/she finished all the prerequisite courses. We create a new helper function to determine this aspect, which is, if has student has not take or finish (which can check if the lgrade is null) any of the prerequisite class.

v. In the beginning, we use exception to accomplish all logics. But in order to be called by jdbc, we change the procedure by providing multiple options, which can be set in the SQL procedure, and pass to java to trigger appropriate sentences to be printed.

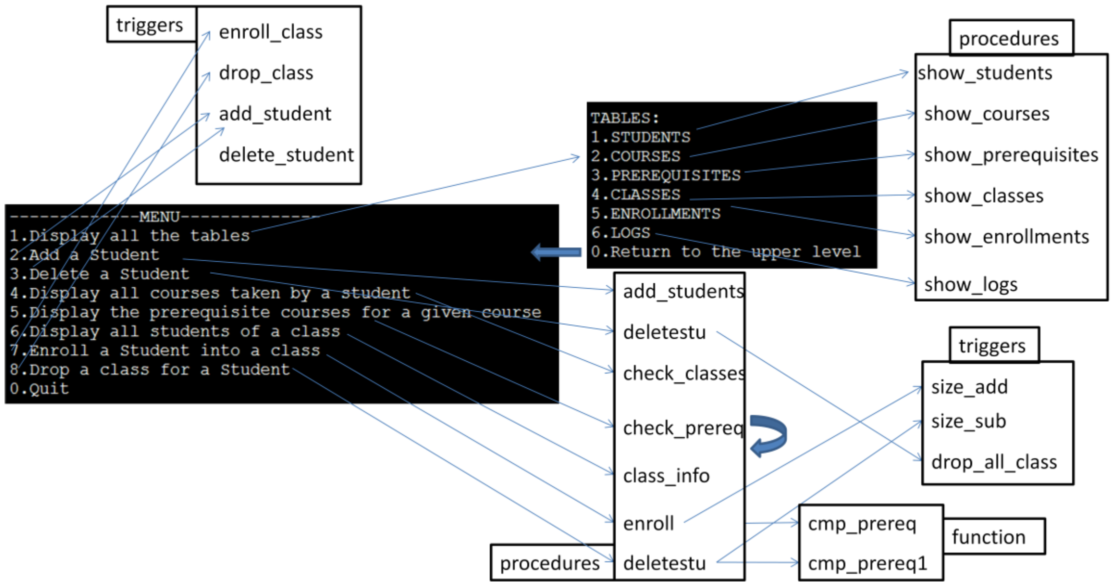
h) Drop a class for certain student

i. Identical with g)

i) Log table triggers

i. When doing above actions in the java option, we use a trigger to add tuples about the detail of the action.

2.



```
--------------MENU--------------
1.Display all the tables
2.Add a Student
3.Delete a Student
4.Display all courses taken by a student
5.Display the prerequisite courses for a given course
6.Display all students of a class
7.Enroll a Student into a class
8.Drop a class for a Student
0.Quit
```

```
TABLES:
1.STUDENTS
2.COURSES
3.PREREQUISITES
4.CLASSES
5.ENROLLMENTS
6.LOGS
0.Return to the upper level
```