IS 620 Advanced Database Projects
Fall 2019
Group Project
Course Scheduling System

Overview

You will form groups of two to five people in this project. It is recommended that each group consists of at least three people given the amount of work. Please read the whole document carefully before starting your project.

Your assignment is to design a course scheduling system. You will design the database, insert some sample data, and implement a set of required features. Each feature will be implemented as one or more Oracle PL/SQL procedures/functions. You do **NOT** need to write a graphic user interface.

Assumptions:

You can make the following assumptions in this project.

1. The system needs to store information about academic departments. Each department has an ID and a name.
2. The department has a few programs. E.g., the IS department as IS BS program, IS BTA program, IS MS program, HCC MS program, etc. Each program has a program ID, program name, and program type (1 for undergraduate and 2 for master's, 3 for PhD).
3. Each program has a number of courses. Each course has course id, course name, number of credits, grading format (letter, pass), required (1 is required for program, 0 is elective), room type (whether it needs computer lab or regular room), number of sections (number of sections needed for a year and semester), section size (number of students per section, status (1 open, 0 close). If a course is scheduled for a given year and semester, it can have 1 or more sections. E.g., IS 620 has one section in Fall 2019 but IS 633 has two sections in Fall 2019.
4. Each course may have one or more courses as prerequisites. A student needs to complete all prerequisites before taking that course.
5. The system stores information about buildings. Each building has a building ID and a name.
6. Each building has a number of classrooms. Each classroom has room id, room name (e.g., ITE 468), number of seats, and room type (1 as computer lab, 0 as regular room).
7. Each department has a number of instructors. Each instructor has an ID, name, and is either part-time or full-time.
8. Each instructor needs to teach a certain number of courses in a given year and semester. This is called course load. For example, Dr. Chen needs to teach two courses in Fall 2019. Note that multiple sections of the same course will be counted as one course. E.g., if Dr. Chen teaches two sections of IS 633 it is still counted as one course (he will get a TA but you don't need to consider that). So he needs to teach another course to fulfill his load. The system also needs to store the number of courses actually assigned to that instructor.

9. Each instructor can specify a list of courses that he or she can teach as well as the number of sections for each course for a given year and semester. E.g., Dr. Chen may specify he can teach one section of IS 620 and two sections of IS 633 for fall 2019.
10. Each instructor can specify up to two day's of a week that he or she cannot teach. E.g., Dr. Chen can specify he cannot teach on Thursday, and Dr. Karabatis may specify he cannot teach on Thursday or Friday.
11. Each course section is scheduled at a time block. These time blocks are not overlapping. Each time block has an ID, start time, length (2.5 hour or 1 hour 15 minutes), day1, day2 (day 1 and day 2 are the days of week the class will be offered). 1 means Monday, 2 means Tuesday, and so on. If the class is 2.5 hour long, it will be scheduled on one day (so day2 is null). If the class is 1 hour 15 minutes, it will be scheduled on two days. E.g., one time block is 10-11:15 am Monday and Wednesday. Another time block is 4:30 pm to 7 pm Tuesday.
12. One or more sections of a course can be scheduled at a given year and semester. The schedule includes schedule id, course ID, section id, instructor id, year, semester, capacity (#of students allowed in the section), time block id, room id, waiting list capacity, and status (open = 1, 0 = full).
13. Each student has a student ID, name, and enrolled in a program.
14. Each scheduled section can have a waiting list. The list keeps IDs of any student on the list, the schedule ID, and position of the student on the waiting list.
15. A student can register for a scheduled course section. There can be three different registration status: 1 means enrolled, 2 means dropped, and 0 means wait listed. A student will have a grade for that course section: (4: A, 3: B, 2: C, 1: D or pass, 0: F).
16. Special permission can be given to students to allow them to enroll in a section that is either closed or the student has not taken prerequisites. The system stores ID of student, ID of the scheduled section, and the type of special permission (1: enroll in closed class, 2: enroll without prerequisite).

Features (those with * are more difficult and can be counted as 2-3 features when assigning features among group members)**

Course scheduling features

1. Add a new course. Input includes course name, program id, number of credits, grading format, whether the course is required, room type, number of sections. The feature checks if a course with the same name exists in the program. If so, update the existing course with the values of the input parameters. Otherwise, insert a new course and generate a new course id. Please print out the new course id.

2. Add a new instructor. Input includes instructor name, department id, and instructor type (full time or part time). Please check if an instructor with the same name exists at the same department with same instructor type. If so print an error message. Otherwise insert the instructor and print out the generated instructor id.

3. Allow an instructor to enter teaching preferences for a given year and semester. The input includes instructor id, year, semester, course load (the number of courses the instructor needs to teach and multiple sections of the same course count as one course), a list of courses the instructor is willing to teach (this list should be greater or equal than the number of courses need to teach), for each such course, the number of sections

the instructor is willing to teach, and up to two days of week that the instructor cannot teach.

Please first check special cases including 1) the input instructor id is valid. 2) the list of courses is smaller than the course load, 3) the number of blackout days is over two. Print an error message if one of these cases occurs. Otherwise insert these input to appropriate tables.

Hint: you will need varray data type to represent a list of items.

4. Allow a student to search for all courses offered in a given year, semester, and program. The input includes year, semester, program id.

   The procedure prints out for each scheduled course section the name of the course, number of credits, grading format, schedule id, section id, name of instructor, name of classroom, days of class, start and end time of each class, and whether the class is open or full. Please order the results by course id and section number. Please also check whether input program id is valid.

5. Compute the number of assigned courses for all instructors given a year and semester. This feature checks the number of scheduled courses taught by each instructor for that year and semester and store this number in the course_load table. Note that multiple sections in the same course is counted as one course. This feature will be called by other features.

6. *** Given the ID of a course, year, and semester, assign the course to instructors who are willing to teach that course. Please first check whether the course needs more sections (compare number of sections needed for the course from the course table to the number of sections already scheduled). If so assign new sections to instructor who is willing to teach that course and create schedule for this course (you can leave classroom unassigned). Othewise print a message there are enough sections.

   You need to make sure that
   1) enough course sections are assigned (the number of sections is in course table). Print a message if this is not possible.
   2) only instructors who are willing to teach that course will teach the course.
   3) The instructor's course load is not exceeded.
   4) the new schedule's capacity equals section size of the course and waiting list size is 10.

   You can use the following greedy method to assign the courses.
   You can also use a temporary table to store intermediate results (e.g., the set of instructors, the number of sections they are willing to teach, the course load, and their weight as described in step 4)

   Psuedo code of the method to assign course c.
   1) Find a set I of instructors who are willing to teach course c and have not reached their course load (the number of courses they suppose to teach) and have not scheduled to teach that course.

2) If I is empty print an error message “not enough sections are assigned due to lack of faculty”. Stop the procedure.
3) Sort instructors in I in descending order of a weight. The weight = min(#of sections of c an instructor is willing to teach, #of unassigned sections of c) multiply (the instructor's course load - #of assigned courses to that instructor). E.g., suppose both instructor A and B need to teach two courses. A is assigned 1 course, and B is assigned nothing so far. Both A and B are willing to teach 2 sections of c. c has 3 unassigned sections. Now A's weight = min(2,3)*(2-1)=2, B's weight = min(2,3)*2=4. So the sort order is B, A.
4) for each instructor i in the sorted I
5) Assign c to i. The number of sections is the minimal of unassigned sections of c and number of sections i is willing to teach. Create a course schedule for this assignment (you don't need to assign classroom or time at this moment)
6) exit when there is no more section to assign.
7) end for;
8) If there is still sections to assign print there is not enough instructors.

E.g., suppose course c needs 3 sections, instructor A and B both are willing to teach 2 sections of c. Both A and B have a load of two courses and A has been assigned one course so far and B has been assigned 0.
The method will sort A, B by their weight. A's weight = 2, B's weight = 4. So the first round B is assigned to teach 2 sections of c. In the next round only A remains so A will be assigned to teach the remaining section of c.

7. Assign courses in a department for a given year and semester. This feature can call feature 6 to assign a course to instructors. This feature will go through every program in the department and tries to assign required courses first. If there are still instructors who have not reached their course load, this feature will assign the electives in these programs. Finally it will check whether all instructor has been assigned enough courses and print out the names of instructors who have not.

8. *** Assign room and time to a scheduled section. Input includes a schedule id. First check whether the schedule id is valid. If not print an error message. Next check whether the scheduled section already has a room and time block. If so print an error message saying that the course is already assigned. Otherwise find a room and a time block pair that satisfies the following conditions:
   1) The day of week is not one of the blackout days of the instructor.
   2) The instructor is not teaching at that time block.
   3) The room has no class scheduled at that time block.
   4) If the course has multiple sections, no other section is at the same time block.
   5) If the course belongs to a graduate program, choose a time block that is after 4:00 pm.
   6) The number of seats in the room is greater than or equal to class capacity
   7) If the course requires in a computer lab, the room must be of computer lab type.

   If there are multiple (room, time block) pairs satisfying the condition, then you can compute a score for each pair and assign the section to the pair with the lowest score. If there are no pair satisfying the above conditions, print out cannot find such pair.

The score is computed as following. First find out the total number of sections already scheduled in that room at the same year/semester.
Next find out the total number of sections scheduled at the same time block at the same year/semester.

The score = product of these two.

E.g., if a room A has 3 sections scheduled and a room B has 4 sections scheduled, and a time block T1 has 2 sections scheduled and a time block T2 has 3 sections scheduled. Suppose two pairs (room A, T1) and (room B, T2) satisfy the above conditions (1 to 7).
Score of (room A, T1) = 3 * 2 = 6
Score of (room B, T2) = 4 * 2 = 8
The class should be assigned to room A at T1.

Again, you can create a temporary table to store the room, time block pair and score.

9. Assign rooms and time blocks for courses in a department given a year and semester. Input includes department id, year, and semester. This feature can call feature 8 to assign a section to a room and time block. This feature will go through every scheduled section in that department, year, and semester and assign for each section a room and time block. Please first schedule graduate courses and then undergraduate ones. For courses are all graduate or undergraduate, schedule required courses first.

Registration features

10. Enter special permission for a student and a scheduled section. The input includes student id and schedule id. First check whether both are valid. If not print a message. If both are valid enter special permission to special permission table. The special permission includes 1) enroll in a closed class. In this case the student can enroll even if the class is full; 2) enroll without prerequisite. In this case the student can enroll even if the student has not taken prerequisite.

11. Check prerequisite of a class for a given student. Input includes a schedule ID and a student ID. Returns one if the student has taken prerequisites of the scheduled course, 0 otherwise. Please check if the schedule ID is valid first.

12. *** Allow a student to register a course given a schedule id.
    a. First check whether the schedule id and student id are valid. Print an error message if it is invalid.
    b. Next checks whether the student has taken the same course before. Retaking a course is not allowed unless the student got a D before. Print an error message if this happens.
    c. Check whether the student has taken prerequisites of this class. If not and the student does not have special permission to enroll without prerequisite, print an error message.
    d. Finally check whether the class is still open. If so or the student has special permission to enroll in a closed class, add the student to registration table with enrolled status. Please check whether the class reaches capacity now. If so update the status of schedule. If the class is full and there is still room on waiting

list put the student on the next position of the waiting list. If the waiting list is full as well, print an error message.

13. *** Allow a student to drop a course. Input includes student id and schedule id. First check whether the student has registered for that scheduled section (the status could be enrolled or wait listed). If not, print a message the student is not registered with that course.
    Otherwise if the student is on wait list, remove the student, change registration status to dropped, and move up anyone after the student on wait list.

    If the student is enrolled, automatically enroll the first student on waiting list and update the waiting list (that person needs to be dropped from waiting list and all others move up one position).

    If there is no student on waiting list, set the course to be open if the course capacity is not reached.

    E.g., suppose student A drops class 2. Suppose there are two students B and C on waiting list with position 1 and 2. Student B will be enrolled automatically and student C's position will become 1.

14. Allow a student to print a course schedule for a given year and semester. Input includes student id, year, and semester. Print course id, course name, section id, and status of all courses the student has registered in that year and semester. For any course the student is on the waiting list, also print out the waiting list position.

Statistics Features

15. Print enrollment statistics for a department given department id, year, and semester.
    a. Please print out total number of students enrolled for at least one course (do not count dropped or wait listed cases) in any courses in that year and semester in programs in the department.
    b. Please also print out total number of courses in that department, year and semester, total number of course sections, #of students enrolled and wait listed in each course section along with course id, course name, section id.

16. *** Identify for a given year and semester, the top k courses with the longest waiting list, the top k rooms with the fewest scheduled class sections, and the top k time blocks that have the fewest scheduled class sections. Input includes year, semester, and k. Please print out the id and name of classes, their waiting list length. For rooms, please print out room id, room name, and #of scheduled class sections. For time blocks print out time block id, days of the week, and start time.

    Please pay special attention to the case when a time block has no class scheduled or a room has no class scheduled (so they should be in the result).

Deliverables:

There will be 4 deliverables. D1 and D3 will be due before class starts on the due date. D2 and D4 are due midnight of the due date. Delayed submission will result in a penalty of 30% of your score (e.g., if your score for part 2 is 20 but you are late, your score will be 14). The final presentation (D3) is due at class time and no delay is allowed.

1. 10%. Due 9/12. Project Management Schedule.
    a. Include team members and a timeline showing each phase of your project with its tasks and time duration, for the entire effort.
    b. It is expected that every member should participate in all phases of the project. For example, every member should be involved in writing the code.
    c. Each task in the same phase may be assigned to different members. E.g., you can specify that features 1-5 are assigned to member X. Pay attention to difficulty of each feature as well.
    d. Tasks should include system design, populating tables, writing code, testing code, running example queries, writing documents, preparing for presentation, etc. Smaller milestones shall be set for deliverable 3 and 4.
    e. This deliverable will be graded based on whether items a) to d) are included and whether the schedule is reasonable (e.g., enough time is left for testing and integration).

2. 25%. Due 10/10. Design Document which includes the following:
    a. ER diagram of the database. You don't have to follow exact notations of ER diagram, but need to show tables, columns, primary keys, and foreign key links.
    b. SQL statements to create database tables and to insert some sample data (at least 5 rows per table).
    c. Specification for each required feature. The specification should include a description of input parameters and output (usually screen outputs), and an example of how a user can use this feature (e.g., exec XXX(…) where XXX is the procedure name). You don't need to implement any of these features at this point.

3. 30%. Due 12/12. Demonstration. Your work will be demonstrated to the class in real time, where you will present the design of your system and you will run a demo. You don't need to submit anything.

4. 35% Due 12/12. The code should include:
    a. Drop table and drop sequence statements to drop tables if they exist (remember the order of drop should be inverse of create).
    b. Create table and create sequence statements
    c. Insert statements

d. Create procedure statements (with code for the procedures). Each feature can be implemented as one or multiple PL/SQL procedure (in the procedure you may call other procedures or functions). Please include some comments in your code explaining the major steps. You should use create or replace to avoid procedure name conflict.
e. Demo script to show that all your features work correctly. The script shall include some examples to test different cases. E.g., for feature 1, one example for new course (course is not in database) and one example for existing course (just update the existing course). Please include:
    i. PL/SQL script to call the appropriate PL/SQL procedure for this feature. E.g., exec procedure-name(parameter values)
    ii. Explanation of what should be the correct output. The output could be updated tables (you can have some select statement to show the updated tables), some print out, etc.
    iii. Make sure you have tested your examples from beginning to end. Remember that database tables may have been changed in the process. So you may need to start with a clean database (i.e., right after you execute all the drop table, create table, and insert statements).

Grading Guidelines

What I look for while grading software code (deliverable 4):

1. Existence of code and whether all code can be compiled without any error.
2. Comments: Both descriptive and inline for every procedure/function
3. Software quality
    a. Whether it is correct (giving correct results).
    b. Whether it is complete and clear.
    c. Efficiency of code. You shall not use too many SQL statements, and you shall put as much work as possible in SQL. For example, if you can do a join, do not use two select statements and then do a join in your program.
    d. Whether it has considered all special cases.

Regarding the presentation of your project: Each student must participate in the project demonstration by presenting to the entire class some slides. You will be graded on:

1. Timeliness of presentation
2. Presentation Style
3. Demo (running the code)

For the demo, you will be graded on the following items:

1. Existence of tables and data. You need to have at least 5 rows in each table.
2. The correctness of features. This can be shown by checking whether the screen output is correct and the database has been updated correctly.

Each member of the team shall contribute more or less equally. It is unfair for a few members to do most of the work while others do less. You will be asked to evaluate your

teammate's effort at the end of the project. The instructor will adjust the grade based on the evaluation. Normally if most of your teammates agree that you do not contribute at all or contribute too little (e.g., your group has 4 members and you contribute only 5%), you may lose up to 80% of your project grade. If your teammates agree that you contribute much more than anyone else (e.g., your group has 4 members and you contribute 40%), you may gain up to 20% of your project grade (but not exceeding 100% of project grade). A peer evaluation will be conducted at the end of the semester to determine the contribution of each team member.

Tips:

1. Be aware of you and your team members' strengths and limitations. Some of you may have very little programming experiences, and some of you do. So each team shall have at least 2 people who are good at programming. Each team can discuss how to assign the workload fairly and at the same time matching the strengths of team members. Usually easier features can be assigned to members without much programming experience (they should take more responsibility on non programming part to ensure equal contribution). However, every member should write some code (3 features at minimal).
2. Start early. Do not wait until last month to start coding. Do not wait until one week before the demo to start putting things together. Past experiences show that more than 50% of time shall be devoted to testing and putting things together.
3. Learn how to debug SQL and PL/SQL code. Most of time the error is from the SQL part of your code. So you can test SQL part separately (e.g., by copy & paste the SQL statement in a cursor and replace PL/SQL variables/parameters with values). You can insert screen output statements to check intermediate results. Oracle also returns error messages and error code. You can google the error messages and error code to find possible causes. You may also use Oracle SQL Developer which allows you to insert break points during debugging.
4. It is highly recommended to use SQL Developer rather than the web interface for the project.
5. Use homework, in class exercises, and programs in slides as templates of your PL/SQL program. For example, if you need to write a cursor, find a cursor example and use it as a starting point.
6. Make sure special cases are handled.
7. At demo time, different data in the database may lead to different results. So usually you will start with a standard database (with a fixed set of tables and rows), and keep track of the sequence of the demo (e.g., a course can only be scheduled if it has been added first).