CS301: DATABASES

COURSE PROJECT PHASE - A

ACADEMIC PORTAL

TEAM MEMBERS:

P RAJASEKHAR 2019CSB1105

DHIRESH KUMAR SINGH 2019CSB1255

TABLES CREATED

1. course_catalogue(
   course_id ,
   course_name,
   dept_name,
   ltpsc,
   credits,
   timetable_slot
   );

2. prerequisites(
   course_id ,
   prereq_course_id ,
   );

-- both course_id and prerequisite_course_id are foreign keys.

3. student(
   student_id ,
   name ,
   dept_name ,
   curr_credits ,
   sgpa ,
   cgpa
   );

4. student_registration(
    student_id,
    course_id,
    section_id,
    semester,
    year,
    grade ,
    grade_num,
    status
);

This table is to store the information of students and the courses in which they are registered in the current semester. When the student calls the stored procedure register_in_course(course_id), then a tuple will be added with corresponding information in this table. Initially grade, grade_num will be NULL. status will be pending initially. It will get updated after his ticket is accepted or rejected accordingly.

5. student_completed_courses_so_far(
    student_id,
    course_id ,
    status,
    semester ,
    year,
    grade ,
    grade_num,
);

This table contains information of students and what are the courses he registered in the previous semesters. Like if the student is in 3rd semester now, this table contains his 1st and 2nd semester's information. Basically if a student completed a course and obtained a pass grade in it, it will get added in this table.

6. instructor(

Ins_id,

dept_name,

name

);

7. course_offering(

ins_id ,

course_id,

section_id,

semester,

year ,

cgpa

);

8. Above table contains a list of all courses offered by instructors and we also created a table for each course offering in the above table (as suggested by sir.)

For example:

cs204(

student_id,

grade,

grade_num

);

Grade column contains letters like A, B- etc. grade_num contains corresponding points in that course like 9,8 etc. initially grade and grade_num are null in this table and the instructor will upload grades from the csv file into this table.

9. We created ticket_table for each branch. For example: cs_ticket_table, ee_ticket_table etc for all the branches (in the ug handbook). Because each branch has their own batch_advisor. And it is efficient this way.

Eg:

```
cs_ticket_table:(
                    ticket_id,
                    student_id,
                    course_id,
                    credits,
                    credits_he_enrolled_current_sem,
                    whether_he_will_exceed_24,
                    Instructor_decision,
                    Batch_advisor_decision,
                    Dean_decision
                );
```

In the above table: credits represents credits of the course that student wants to register in (let say this value =x). credits_he_enrolled_current_sem represents the credits the student enrolled in the current_sem (let say this value =y). Whether_he_will_exceed_24 column contains 'yes' if (x+y > 24). 'No' otherwise. **We are including all this information, so that when processing the tickets, the batch advisor, dean and instructor will make their decision after fully knowing how many extra credits a student is going to register for and whether he is exceeding 24 credits or not.**

10. Now we created three other table:

a.Instructor_tickets

b.cs_batch_advisor_tickets (similarly for other branches)

c.dean_tickets

As suggested by sir, when the stored procedure get_tickets_instructor is called by the instructor, it will fetch tickets from cs_ticket_table and add them in instructor_tickets . so

that instructor put his decision in this table after processing the ticket. Similarly to the other two tables.

[the ticket propagation logic is explained in more detail in the stored_procedures section.]

STORED PROCEDURES IMPLEMENTED

1. upload_timetable()
    a. Arguments: csv file path.
    b. Called by the dean academics office. To upload time table for the semester
    c. This function will update the timetable_slot of each course in the course_catalogue.

2. calculate_cgpa()
    a. Arguments: student_id
    b. Function to calculate current cgpa of the student. As discussed with sir, This procedure would take the student_id as an input. Given the student_id , we considered all the course registrations/grades of that particular student from the table student_completed_courses_so_far. [Note: this student_completed_courses_so_far table contains the courses of that student after he obtained a grade in it. If his grade in any course is still not updated by the instructor that course will not be in this table . So we are assuming that this function will be called only after all instructors uploaded their grades.].
    c. Dean and instructor can call this function and see all the student's cgpa (as mentioned in the project specification).

3. report_generation()
    a. This function is to generate a grade card of the student. (as discussed with sir we are assuming that grade card is nothing but displaying that student's grades on screen)
    b. Arguments: two cursors and student_id. Cursors to store the output.
    c. It will be called by dean-academics and it will simply print that student's details and his grades in the courses in which he enrolled currently. And his cgpa and sgpa.

4. grade_entry ()
    a. Arguments: course_id, csv file path.
    b. It will be called by the instructor only, to upload all student grades in his course.
    c. It will update the grade, grade_num column in table mentioned in point 8 (in the tables section.)
    d. After the insert operation in these tables, a trigger will be invoked [explained in triggers section]

5. offer_course()
    a. Arguments:ins_id,course_id,section_id,semester,year,cgpa(this can be null)
    b. If the instructor specified any cgpa cutoff, we used the same cgpa cutoff for all the branches (discussed with sir)
    c. It will be called by the instructor to offer a course
    d. This will add tuples to the course_offering table. There are some triggers that will invoke after insert operation in this course_offering table.[that is explained in the triggers section]

6. register_in_course()
    a. Arguments: s_id,course_id
    b. This function is called by the student , to enroll in a course.
    c. Whenever a student calls this function, it will add tuples in the student_registration table.
    d. Before executing this function body, a trigger will invoke [explained in triggers section].
    e. After the trigger's condition is satisfied, this function will update s_id,course_id ,semester , year in the student_registration table, grade is NULL initially. Status is pending initially.
    f. We are generating a ticket every time a student wants to register in a course.**Because there may be cases like let say an EE student wants to register in CS201 course, in this case he needs permission from cs201 instructor. If we allow students who are below a certain credit limit to register in a course without ticket_generation, then the above case may fail. So we are generating tickets for every course registration.**

7. get_tickets_instructor()
    a. arguments will be c_id (course_id only). and the instructor is the one who is calling.
    b. it should fetch tuples from all branch ticket_tables where course_id=c_id and add them in the instructor_tickets table.
    c. and also in this function only, we execute this query: select * from instructor_tickets where course_id=c_id . so that it will print those tickets on the screen.
    d. After processing the tickets, instructor will put their decision in the instructor_tickets table. [after the decision is put, a trigger will update decision column in cs_ticket_table]

8. get_tickets _batchadvisor

    a. Arguments will be branch names like CS, EE etc.
    b. it should fetch tuples from cs_ticket_table if the argument branch_name =CS and add them in the cs_batch_advisor_tickets table.
    c. After the tickets are added, this function will print them on the screen.
    d. After processing the tickets, the batch advisor will put his decision in the cs_batch_advisor_tickets table. [after the decision is made, a trigger will update the decision column in cs_ticket_table].
    e. Similarly , we did for other branches as well.

9. get_dean_tickets

    a. No arguments. This function will fetch all the tickets in the all branches, where dean has not made his decision yet. And put them in the dean_tickets table.
    b. After processing the tickets, the dean will put his decision in the dean_tickets table. [After the decision is made, a trigger will update the decision column in the respective branch ticket table.].

10. convey_decision_instructor

    a. This stored procedure is called by the instructor to put his decision in the instructor_tickets table.

11. convey_decision_batchadvisor

    a. This stored procedure is called by the batch advisor to put his decision in their batch_advisor_tickets table.

12. convey_final_decision

    a. This stored procedure is called by the dean to put his decision in their dean_tickets table.
    b. This is the final decision.
    c. If the dean approves the tickets irrespective of instructor and batch_advisor decision, that will update status in student_registration table as 'ENROLLED'. We do this step by using triggers.

TRIGGERS IMPLEMENTED

1. set_grade()

    a. Whenever an instructor uploads his course grades using csv file in tables created for each course_offering, this trigger will be invoked after that insertion at row level.
    b. And this trigger will update the grade and grade_num column in student_registration.
    c. Before giving a grade card to the student , dean-academics would like to check these grades. So for that we have grade_copy() stored procedure.

2. create_course_offering_table()

    a. After each insert operation in this course_offering table. This trigger will invoke and it will create a table for that course.
    For example:
    cs204(
    Student_id,
    Grade,
    Grade_num

b. And this cs204 is modified by the instructor when the grade_entry() function is called.

3. check_slot_prerequisite()

a. In the stored procedure register_in_course(), before insert operation this trigger will invoke and it check for two conditions:
    i. Whether the student has already registered in a course with the same slot or not.
    ii. Whether the student has completed prerequisites for this course(course he wants to credit.) or not.
b. If any of the above conditions are not satisfied, the function body(register_in_course) won't execute. And it will not generate any tickets. Because these conditions are necessary for students to register in any course.

4. add_ticket()

a. This trigger will be invoked after insert operation on student_registration table at row level.
b. This trigger will add the ticket information in cs_ticket_table if the student is from cs. Similarly for other branches.

5. ins_decision()

a. This will be invoked when the instructor puts his decision on the instructor_tickets table and it will update the instructor decision in cs_ticket_table, if the instructor is from cs.

6. advisor_decision()
    a. This will be invoked when the batch advisor puts his decision on batch_advisor_ticket table and it will update batch_advisor_decision in cs_ticket_table, if the batch advisor is from cs.

7. final_decision()
    a. When the dean puts his decision in the dean_tickets table, this trigger will be invoked and it will update dean_decision in the respective branch ticket table and the status column in student_registration table.

PERMISSIONS AND ROLES

1. Database name is aims.
2. There is a superuser: dean. He is also the owner of database aims. He has read and write permissions to all tables. The words dean and dean-academics represent the same user here.
3. Roles created:
    a. student, instructor,
    b. cs_batch_advisor, ee_batch_advisor.. Similarly for other branches
4. Each of the above roles have connection privileges to the database aims.
5. Student have read permissions for course_catalogue and course_offering, prerequisites.
6. Only the dean has write permissions to the course_catalogue table.
7. Faculty and dean can see all student's grades and cgpa.
8. We gave only necessary permissions for students. And we implemented stored procedures in such a way that '**no student can see other student's grades'**

9. Instructor have read permission to course_catalogue, prerequisite, course_offering table.
10. Instructor and batch_advisor can write into ticket_table.
11. Instructor have write permission to the course_offering table and their own course_tables where they upload grades from csv file.
12. Instructor also has read permission to the student and student_registration table. Because faculty is allowed to see all students' grades (mentioned in project specification.)
13. Cs_batch_advisor have read and write permission only to cs_ticket_table. Similarly for other branches.