Indian Institute of Technology Ropar CS301 – Databases

Group members
1. Nema Ram Meghwal (2019csb1101)
2. Bablu (2019csb1079)

# Academic Portal

**28th October 2021**

## OVERVIEW

We have implemented a database system for a university academic portal. Following are the functionalities implemented by us:

1. **Students**:
   - Can credit, audit, drop or withdraw in a course if he/she satisfies the course requirements (prerequisites, cgpa constraint, slot constraint, batch constraints [for example 2019 cse batch and 2018 ee batch may only be allowed to take a particular course] etc.)
   - Otherwise, a student can generate a ticket for special permission from dean academics to enroll in that course.
2. **Instructors**:
   - Can offer a course in the current semester (and specify eligibility for enrolling in the course, the eligibility criteria can be updated while the add\drop window is open)
   - Upload grades of the students who credited/audited that course
   - Should give his/her feedback on the ticket generated by a student for special permission to enroll in a course he/she is ineligible for otherwise.
3. **Batch Advisors**:
   - Should provide feedback on the ticket generated by a student for special permission to enroll in a course he/she is ineligible for otherwise.
4. **Dean Academics**:
   - Can add a new course to the course catalog (which can then be offered by an instructor)
   - Add new students to the database
   - Add new instructors to the database
   - Make an instructor a batch advisor
   - Generate transcripts of students
   - Open and close add/drop window

- Open and close withdrawal window

Each student and instructor has a login. A single academic dean user is present. Batch advisors are essentially instructors. Privileges given to each of the above roles are discussed in detail later.

## TABLES

1. **department** (id, name)
2. **batch** (id, year, dept_id)
   - foreign key (dept_id) references department(id)
3. **course** (id, name, l, t, p, s, c, dept_id)
   - foreign key (dept_id) references department(id)
4. **student** (entry_number, name, email, batch_id)
   - foreign key (batch_id) references batch(id)
5. **instructor** (id, name, email, dept_id)
   - foreign key (dept_id) references department(id)
6. **advisor** (inst_id, batch_id)
   - foreign key (inst_id) references instructor(id)
   - foreign key (batch_id) references batch(id)
7. **slot** (id, duration)
8. **offering** (id, course_id, inst_id, sem_offered, year_offered, slot_id))
9. **registration_status** (id, add_open, withdraw_open, current_sem, current_year)
   - Description: This is a table constrained to contain a single row which contains current semester, current year and whether the add/drop or withdrawal window is open. The id's data type is boolean and is constrained to be true, effectively limiting the size of this table to 1 row.
10. **prereq** (course_id, prereq_id)
    - foreign key (course_id) references course(id),
    - foreign key (prereq_id) references course(id)
    - Description: This is a self referential one to many relation from course table to course table. So for tuples { ('CS301', 'CS201'), ('CS301', 'CS204') }, the interpretation is that CS201 and CS204 are prerequisites for the course 'CS301'
11. **credit_<course_id>** (entry_number, grade)
    - foreign key (entry_number) references student(entry_number)

- Description: Stores the grades for students who credited the course <course_id>

12. **audit_<course_id>** (entry_number, grade)
    - foreign key (entry_number) references student(entry_number)
    - Description: Stores the grades for students who audited the course <course_id>
13. **drop_<course_id>** (entry_number)
    - foreign key (entry_number) references student(entry_number)
    - Description: Stores the entry number of students who dropped from the course <course_id>
14. **withdraw_<course_id>** (entry_number)
    - foreign key (entry_number) references student(entry_number)
    - Description: Stores the entry number of students who withdrew from the course <course_id>
15. **constr_<course_id>** (batch_id, min_gpa)
    - foreign key (batch_id) references batch(id)
    - Description: Stores the constraints for enrolling in the course. For example, if only the CS'19 (batch_id = 2, say) students with CGPA greater than 8 and EE'18 (batch_id = 11, say) students with CGPA greater than 8.5 are allowed to enroll in the course CS301, then the table constr_CS301 will have 2 tuples { (2, 8), (11, 8.5) }
16. **credit_<entry_number>** (offering_id, grade)
    - foreign key (offering_id) references offering(id)
    - Description: Stores the grades of the student with entry number <entry_number> in all the courses credited by him/her.
17. **audit_<entry_number>** (offering_id, grade)
    - foreign key (offering_id) references offering(id)
    - Description: Stores the grades of the student with entry number <entry_number> in all the courses audited by him/her.
18. **drop_<entry_number>** (offering_id)
    - foreign key (offering_id) references offering(id)
    - Description: Stores the offering id of courses from which the student with entry number <entry_number> dropped.
19. **withdraw_<entry_number>** (offering_id)
    - foreign key (offering_id) references offering(id)
    - Description: Stores the offering id of courses from which the student with entry number <entry_number> withdrew.

20. **s_ticket_<entry_number>** (id, offering_id, i_verdict, b_verditct, d_verdict)
    - foreign key (offering_id) references offering(id)
    - Description: The student can insert into this table to generate a ticket with all verdict values set to null (else EXCEPTION will be raised), procedure for this is explained in further section procedures.
21. **i_ticket_<inst_id>** (id, entry_number, verdict)
    - foreign key (entry_number) references student(entry_number)
    - Description: Instructor can give his feedback by updating the verdict column.
22. **b_ticket_<inst_id>** (id, entry_number, verdict)
    - foreign key (entry_number) references student(entry_number)
    - Description: Branch advisor can give his feedback by updating the verdict column.

## TRIGGERS

1. **constr_update_<offering_id>**
    - before update on constr_<course_id>
    - check made to ensure that GPA constraint isn't tightened and a previously eligible-marked batch is not marked ineligible now
2. **add_offering**
    - after insert on offering
    - creates dynamic tables to store record of students who credit, audit, drop from or withdraw from this course
    - additionally, a table that has gpa and batch constraints is also generated dynamically
    - for constraint table, the constr_update_<offering_id> trigger and corresponding trigger function is also generated dynamically
    - for all the procedures and tables generated dynamically, the appropriate privileges to appropriate roles (as mentioned later) are granted.
3. **add_offering_security_check**
    - before insert on offering
    - to ensure that some other instructor doesn't insert to the offering table on behalf of an instructor

4. **add_s_ticket_<entry_number>**
    - after insert on s_ticket_<entry_number>
    - check that ticket raised by student is valid (doesn't directly insert the verdict into the table, it should be null initially and if insertion is successful, the ticket is propagated to corresponding instructor for feedback)
5. **enroll_credit_<entry_number>**
    - before insert on credit_<entry_number>
    - check if the student is eligible for enrolling in that offering i.e passes batch constraint, gpa constraint, the offering being enrolled into is active this semester, 1.25 rule is being followed, add/drop window is open. Even if some constraint is not passed but dean has approved the ticket raised by the student for this offering, he can enroll (add/drop must be open however)
    - if this course is currently audited by the student, it is removed from audit_<entry_number> table and added to credit_<entry_number> table (equivalent to updating audit to credit for this course)
6. **enroll_audit_<entry_number>**
    - before insert on audit_<entry_number>
    - check if the student is eligible for enrolling in that offering i.e passes batch constraint, gpa constraint, the offering being enrolled into is active this semester, add/drop window is open. Even if some constraint is not passed but dean has approved the ticket raised by the student for this offering, he can enroll (add/drop must be open however)
    - if this course is currently credited by the student, it is removed from credit_<entry_number> table and added to audit_<entry_number> table (equivalent to updating credit to audit for this course)
7. **enroll_drop_<entry_number>**
    - before insert on drop_<entry_number>
    - if this course is neither in credit_<entry_number> nor in audit_<entry_number>, then exception is raised
    - otherwise, corresponding tuple in one of those tables is removed and added to drop_<entry_number>
8. **enroll_withdraw_<entry_number>**

- before insert on withdraw_<entry_number>
- if this course is in either audit_<entry_number> or credit_<entry_number>, then it is removed from there and added to enroll_withdraw_<entry_number>
- ofcourse, this is only allowed if withdraw window is open, else exception is raised

9. **add_student**
   - before insert on student
   - role for the student is created (for him/her to login)
   - generates dynamic tables [audit|credit|drop|withdraw|s_ticket]_<entry_number> and gives privileges to different roles on these tables appropriately
   - also dynamically generates triggers mentioned in (4th to 8th) points above
10. **i_ticket_verdict_<inst_id>**
    - before update on i_ticket_<inst_id>
    - allows verdict to be given only once, i.e update is allowed only once. (it won't make sense if instructor could change his verdict after the verdict by advisor or dean is given)
    - ticket is propagated to advisor for feedback (irrespective of instructor's feedback)
    - update is made to s_ticket_<entry_number> accordingly for the attribute i_verdict
11. **add_instructor**
    - after insert on instructor
    - dynamically generates table i_ticket_<inst_id> and grants appropriate access to users on this table
    - trigger #10 is dynamically generated
12. **b_ticket_verdict_<inst_id>**
    - before update on b_ticket_<inst_id>
    - allows verdict to be given only once, i.e update is allowed only once. (it won't make sense if advisor could change his verdict after the verdict by dean is given)
    - ticket is propagated to dean for final decision (irrespective of advisor's feedback)
    - update is made to s_ticket_<entry_number> accordingly for the attribute b_verdict
13. **add_advisor**

- after insert on advisor
- dynamically generates table b_ticket_<inst_id> and grants appropriate access to users on this table
- trigger 12 is dynamically generated

14. **d_ticket_verdict_<inst_id>**
    - Before update on d_ticket
    - allows verdict to be given only once, i.e update is allowed only once. (it won't make sense if student enrolled in a course and then dean changes the verdict)
    - update is made to s_ticket_<entry_number> accordingly for the attribute d_verdict and this is the final verdict for the ticket

## PROCEDURES & FUNCTIONS (excluding the trigger functions)

1. **enroll_credit(**offering_id**)**
    - Called by <student_id>
2. **enroll_audit(**offering_id**)**
    - Called by <student_id>
3. **drop_offering(**offering_id**)**
    - Called by <student_id>
4. **withdraw_offering(**offering_id**)**
    - Called by <student_id>
5. **generate_ticket(**offering_id**)**
    - Called by <student_id>
6. **ticket_verdict_i(**ticket_id, entry_number, verdict**)**
    - Called by <instructor_id>
    - To give his/her feedback about the ticket. The feedback is updated in the table i_ticket_<inst_id> and propagated to column i_verdict of table s_ticket_<entry_number>
7. **ticket_verdict_b(**ticket_id, entry_number, verdict**)**
    - Called by <instructor_id>, who is also an advisor
    - To give his/her feedback about the ticket. The feedback is updated in the table b_ticket_<inst_id> and propagated to column b_verdict of table s_ticket_<entry_number>
8. **ticket_verdict_d(**ticket_id, entry_number, verdict**)**
    - Called by dean_acad

    - To give the verdict about the ticket. The feedback is updated in the table d_ticket_<inst_id> and propagated to column d_verdict of table s_ticket_<entry_number>
9. **add_offering(**course_id, slot_id, constraints**)**
    - Called by <instructor_id>
10. **add_constraints(**offering_id, batch_id, min_gpa**)**
    - Called by <instructor_id>
11. **start_add(**current_year, current_sem**)**
    - Called by dean_acad
    - To open the add/drop window
12. **stop_add()**
    - Called by dean_acad
    - To close the add/drop window
13. **start_withdraw()**
    - Called by dean_acad
    - To open the withdraw window
14. **stop_withdraw()**
    - Called by dean_acad
    - To stop the withdraw window
15. **update_credit_grades(**filepath, offering_id**)**
    - Called by <instructor_id>
    - To import grades of students who credited the course from a csv located at filepath
16. **update_audit_grades(**filepath, offering_id**)**
    - Called by <instructor_id>
    - To import grades of students who audited the course from a csv located at filepath
17. **generate_transcript(**entry_number, sem, year**)**
    - Called by dean_acad
    - To print the detailed grades, gpa (through raise INFO)

## HELPER FUNCTIONS

1. **get_sgpa (**entry_number, sem, year**)**
2. **get_cgpa** (entry_number)
3. **get_id** ()
4. **get_current_year** ()
5. **get_current_sem** ()
6. **is_add_open** ()

7. **is_withdraw_open** ()
8. **grade_to_number** (credit_grade)
9. **get_gpa** (entry_number)
10. **is_offering_offered_in_current_sem_and_year** (offering_id)
11. **is_slot_conflicting_for_instructor** (inst_id, slot_id)
12. **is_slot_conflicting_for_student** (entry_number, offering_id)
13. **does_student_satisfy_prereq** (entry_number, offering_id)
14. **is_student_eligible_for_credit** (entry_number, offering_id)
15. **is_student_eligible_for_audit** (entry_number, offering_id)

## CUSTOM DATA TYPES

- **credit_grade:** enum ('F', 'E', 'D-', 'D', 'C-', 'C', 'B-', 'B', 'A-', 'A')
- **audit_grade:** enum ('NP', 'NF')

## ROLES

*(Whenever not mentioned, the permissions are inherited, otherwise there is no access.)*

### Groups (roles without login)

1. **student**

| Table | Permissions |
|---|---|
| department | select |
| slot | select |
| batch | select |
| course | select |
| student | select |
| instructor | select |
| advisor | select |
| offering | select |
| constr_<offering_id> | select |
| prereq | select |
| registration_status | select |

2. **instructor**

| Table | Permissions |
|---|---|
| department | select |
| slot | select |
| batch | select |
| course | select |
| student | select |
| instructor | select |
| advisor | select |
| offering | select, insert |
| constr_<offering_id> | select |
| prereq | select |
| registration_status | select |

3. **advisor**

| Table | Permissions |
|---|---|
| department | select |
| slot | select |
| batch | select |
| course | select |
| student | select |
| instructor | select |
| advisor | select |
| offering | select |
| constr_<offering_id> | select |
| prereq | select |
| registration_status | select |

## Users (roles with login)

1. **<student_id>**
    a. Inherits permissions from `student` group
    b. For their own tables (suffixed by `<student_id>`)

| Table | Permissions |
|---|---|
| s_ticket_<student_id> | select, insert |
| credit_<student_id> | select, insert |
| audit_<student_id> | select, insert |
| drop_<student_id> | select |
| withdraw_<student_id> | select |

**2. <instrutor_id>**

a. Inherits permission from `instructor` group

b. When `<offering_id>` is offered by `<instructor_id>`

| Table | Permissions |
|---|---|
| i_ticket_<instructor_id> | select, update |
| credit_<offering_id> | select |
| audit_<offering_id> | select |
| drop_<offering_id> | select |
| withdraw_<offering_id> | select |
| constr_<offering_id> | select, insert, update |

c. When instructor is also an advisor

i. Inherit from `advisor` group

| Table | Permissions |
|---|---|
| b_ticket_<instructor_id> | select, update |

**3. dean_acad**

a. Can SELECT all tables

| Table | Permissions |
|---|---|
| department | all |
| slot | all |
| batch | all |
| course | all |
| student | all |
| instructor | all |
| advisor | all |

| | |
|---|---|
| offering | select |
| constr_<offering_id> | select |
| prereq | all |
| registration_status | select, update |
| d_ticket | select, update |