Academic Portal - DBMS

Ritik Garg(**ritikgargg**)
Tarun Singla (**tarun-singla**)

Schema Design - Tables I

Following are the tables present in the database. The primary keys in the corresponding tables are specified in bold font.

1. **terms** (**term_id**, semester, year) - This table contains the details about the academic terms (semester, year).
2. **batches** (**batch_id**, department, degree, join_year, graduate_year) - This tables contains information about the student batches, specified by the department, the degree, the year of joining and the year of graduation.
3. **courses** (**course_id**, title, department, lectures, tutorials, practical, self_study, credits) - The details about the available courses including the course identifiers, their titles, their departments and their LTPSC requirements are specified through this table.

Schema Design - Tables II

4. **prerequisites** (**course_id**, **prerequisite_id**) - This table contains information about the pre-requisites of courses, required to be verified before registering a student into an offered course. The course_id and prerequisite_id are also foreign keys, referencing the course_id field in courses table.
5. **time_slots** (**term_id, slot_id, course_id**) - This table contains details about the courses in particular slots. The term_id and the course_id are foreign keys, referencing term_id in terms table and course_id in courses table, respectively.
6. **course_offerings** (**course_offering_id**, course_id, term_id, section_id, slot_id, batch_ids[], cgpa_cutoff) - This table contains details about the courses offered by a faculty, including details like the time-table slot, the list of batches allowed to register for the course and the CGPA cutoff (if any). term_id, slot_id and course_id are foreign keys referencing term_id, slot_id and course_id respectively from the time_slots table.

Schema Design - Tables III

7. **students** (**student_id**, name, batch_id) - This tables includes the details about the students (student identifier, name and batch identifier). batch_id is a foreign key referencing batch_id from the batches table.
8. **faculty** (**faculty_id**, name, department) - This table contains the details about the faculties, including the faculty identifier, the name and the department of the faculty.
9. **faculty_advisor** (**faculty_id, batch_id**) - This table contains information about the faculties appointed as faculty advisors of corresponding batches. The faculty_id and the batch_id are foreign keys, referencing faculty_id in faculty table and batch_id in batches table, respectively.

Schema Design - Tables IV

10. **dao_tickets** (**course_offering_id**, **faculty_id**, **student_id**, faculty_status, status) - This table contains the tickets propagated to Dean Academic Office. course_offering_id, faculty_id and student_id are foreign keys referencing respectively course_offering_id from the course_offerings table, faculty_id from the faculty table and student_id from the students table.
11. **teaches** (**faculty_id, course_offering_id**) - This table specifies information about the faculty teaching a particular course offering. The faculty_id and the course_offering_id are foreign keys, referencing faculty_id in faculty table and course_offering_id in course_offerings table, respectively.

Schema Design - Individual Tables I

1. Tickets table for each faculty advisor
   tickets_<faculty_id> (**course_offering_id, student_id**, status). The course_offering_id and student_id are foreign keys, referencing course_offering_id in course_offerings table and student_id in students table, respectively.
2. Tickets table for each student, **without** the final status of the tickets. A student has WRITE permission on his/her tickets table, to add new ticket requests.
   tickets_<student_id> (**course_offering_id**) - course_offering_id is also a foreign key, referencing course_offering_id from course_offerings table.

Schema Design - Individual Tables II

3. View tickets table for each student(with only **READ** access), containing the **final status** of the ticket after the decision of Dean Academic Office. A separate table is maintained for the purpose of viewing tickets, because a student is granted WRITE permission on the original tickets table, thereby exposing the system to security threats had the status been included in the same table.
   viewtickets_<student_id> (**course_offering_id**, status) - course_offering_id is also a foreign key, referencing course_offering_id from course_offerings table.
4. Registrations table for each course offering, without the final grades of the registered students.
   registrations_<course_offering_id>(**student_id**) - student_id is also a foreign key, referencing student_id from students table.

Schema Design - Individual Tables III

5. Results table for each course offering, containing the final grades of the registered students. A separate table is maintained for this purpose, due to the security threats associated with having grades in the original registrations table, which has WRITE permissions allotted to students. **results_<course_offering_id>(student_id**, grade) - student_id is also a foreign key, referencing student_id from students table.
6. Registrations table for each student **registrations_<student_id>(course_offering_id)** - course_offering_id is also a foreign key, referencing course_offering_id from course_offerings table.

Schema Design - Individual Tables IV

7. Transcript table for each student(with only **READ** access). A separate table with grades is maintained due to similar motivations as described in earlier cases.
   transcript_<student_id>(**course_offering_id**, grade) - course_offering_id is also a foreign key, referencing course_offering_id from course_offerings table.

Use of Triggers I

- Whenever a new student is inserted in the student table, his individual tables like tables related to tickets and registrations are created via an after trigger. Student roles are also created and granted various permissions. Batch advisors are also given permissions to these tables here.
- For each insertion in course offerings table, an after trigger creates its corresponding registrations and results table, giving their access to the faculty offering the course. A corresponding entry is also inserted into the teaches table.
- On each student registration table, a before trigger is created which checks various constraints like CGPA, batch, etc. for a course offering.
- Similarly, on each student registration table, an after trigger is associated which inserts the corresponding entry in the course registration table for that particular course offering.

Use of Triggers II

- Whenever a new faculty is inserted in the faculty table, his role is created via a after trigger and is granted the required permissions.
- Whenever a new faculty advisor is inserted in the faculty advisor table, his corresponding tickets table is created via an after trigger and given permissions to this table. The corresponding faculty role is also extended various permissions.
- Whenever the DAO updates the status of the students tickets, the corresponding ticket with result is inserted into the corresponding student's viewtickets table via an after trigger.
- Whenever a new entry is inserted into the batches table, a corresponding faculty advisor role is created via an after trigger. This role is later assigned to the faculty appointed as the faculty advisor of the batch.

Roles

We have four different role types, one for each type of stakeholder:

- Students
- Faculty
- Faculty Advisor
- Dean Academics Office

Each of them has a separate login ID.
The login ID for students is the student identifier.
The login ID for faculties is the faculty identifier.
Batch Advisor is also a faculty, with additional privileges.

Functionalities of Student Role

- Register for a course
- View his academic performance
- Generate and view tickets for registering for courses which he is not eligible

Functionalities of Faculty Role

- Offer a course
- View grades of all the students
- Upload grades of the students of his/her course

Functionalities of Faculty Advisor Role

- Privileges of a faculty.
- Additional privileges include accepting/rejecting tickets

Functionalities of Dean Academics Office Role

- Dean Academics Office is a superuser
- It has all the types of accesses on all tables.

Role Privileges Summary I

Tables	**DAO**	**Faculty Advisor**	**Faculty**	**Student**
terms	All permissions	Read	Read	Read
batches	All permissions	Read	Read	Read
courses	All permissions	Read	Read	Read
students	All permissions	Read	Read	Read
faculty	All permissions	Read	Read	Read
faculty_advisor	All permissions	Read	Read	Read
time_slots	All permissions	Read	Read	Read
prerequisites	All permissions	Read	Read	Read
dao_tickets	All permissions	No permission	No permission	No permission
course_offerings	All permissions	Read, Insert	Read, Insert	Read

Role Privileges Summary II

Tables	**DAO**	**Faculty Advisor**	**Faculty**	**Student**
teaches	All permissions	Read, Insert	Read, Insert	Read
tickets_ ||*faculty_id*	All permissions	Read, Insert	No permission	No permission
registrations_ ||*course_offering_id*	All permissions	No extra permission	Read, Insert	Read, Insert
registrations_ ||*student_id*	All permissions	Read	No permission	Read, Insert
results_ ||*course_offering_id*	All permissions	No extra permission	Read, Insert	No permission
transcript_ ||*student_id*	All permissions	No extra permission	Read	Read
tickets_ ||*student_id*	All permissions	Read	No permission	Read, Insert
viewtickets_ ||*student_id*	All permissions	Read	No permission	Read