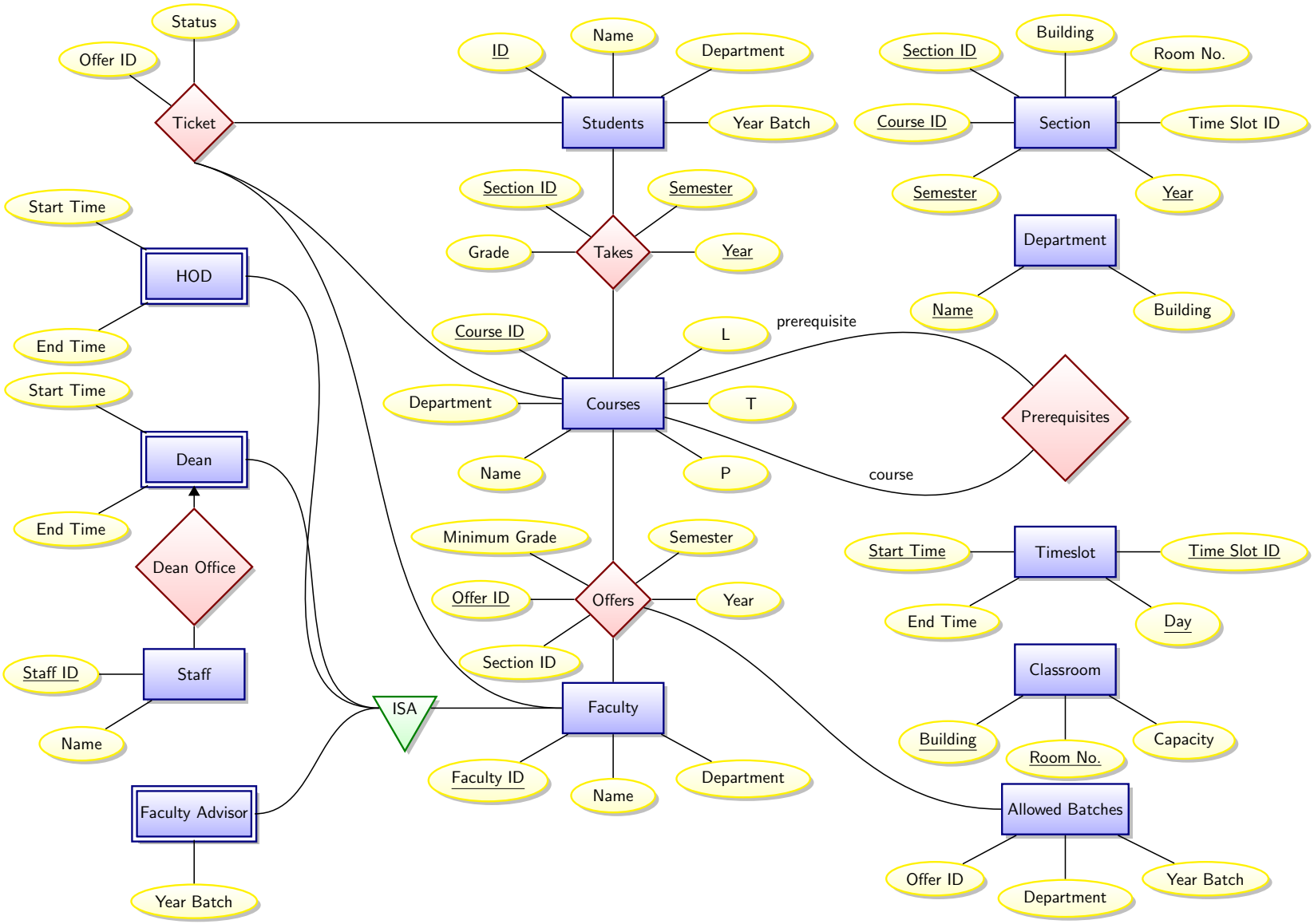
CSL301 Project
E/R Diagram and implementation details

Aditya Gupta(2015CSB1003)
Vinit Kothawade(2015CSB1039)
Shreya Dubey(2015CSB1074)

E/R Diagram


Status
Offer ID
Ticket
ID
Name
Department
Students
Year Batch
Section ID
Building
Room No.
Course ID
Section
Time Slot ID
Semester
Year
Start Time
HOD
End Time
Section ID
Semester
Grade
Takes
Year
Department
Name
Building
Course ID
L
prerequisite
Department
Courses
T
Prerequisites
Name
P
course
Start Time
Dean
End Time
Dean Office
Minimum Grade
Semester
Offer ID
Offers
Year
Section ID
Start Time
Timeslot
Time Slot ID
End Time
Day
Staff ID
Staff
Name
ISA
Faculty
Classroom
Building
Room No.
Capacity
Faculty ID
Name
Department
Faculty Advisor
Allowed Batches
Year Batch
Offer ID
Department
Year Batch

Foreign Keys & Primary Keys I

1. The underlined attributes of the entities form the primary key of that entity.
2. The underlined attributes along with the primary keys of corresponding relations of a relationship forms the primary key of the relationship.
3. (Section ID,Course ID,Semester,Year) in Takes refers to (Section ID,Course ID,Semester,Year) in Section
4. (Section ID,Course ID,Semester,Year) in Offers refers to (Section ID,Course ID,Semester,Year) in Section
5. Course ID in Section refers to Course ID in courses
6. Time slot ID in section refers to Time Slot ID in Timeslot relation
7. (Building,Room No.) in Section refers to (Building,Room No.) in Classroom relation

Foreign Keys & Primary Keys II

8. Offer ID in Allowed Batches refers to Offer ID in Offers relationship
9. Department in Allowed Batches refers to Name in Department
10. Department in Students refers to Name in Department
11. Department in Faculty refers to Name in Department
12. Department in Courses refers to Name in Department
13. Offer ID in Ticket relationship refers to Offer ID in Offers relationship

Portals

We will have 5 different portals, one for

- Students
- Faculty
- HoD
- Dean Academics
- Staff Dean's Office

Each of them will have a separate login ID and respective portal will open. Each portal will have different functionalities.

Functionalities of Student Portal

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

Functionalities of Faculty Portal

- Offer a course
- View grades of all the students
- Update grades of students of his course
- Accept/Reject/Forward a ticket

Functionalities of HoD Portal

- View grades of all the students
- Accept/Reject/Forward a ticket

Functionalities of Dean Academics Portal

- Add or Delete courses from course catalog
- View grades of all the students
- Accept/Reject a ticket

Functionalities of Staff Dean's Office Portal

- View grades of all the students
- Report generation (like transcripts of students. list of probation students)

Relations

In our database we have Relations:

- Students
- Faculty (isa members):
  - Dean
  - HoD
  - Faculty Advisor
- Courses
- Department
- Section
- Timeslot
- Classroom
- Allowed Batches
- Staff

Relationships

We have following relationships :

- many to many relationship “Offers” between “Faculty” and “Courses”
- many to many relationship “Takes” between “Students” and “Courses”
- many to many relationship “Prerequisites” between “Courses” and “Courses”
- many to many multi-way relationship “Ticket” between “Student”, “Courses” and “Faculty”.
- many to one relationship “Dean Office” from “Staff” to “Dean”

Implementation Details I

1. **Course Catalog**: We have created a separate table for courses which consists of L,T,P as attributes and each course is uniquely identified by its primary key "Course ID". We have defined a relationship Prerequisites which is a many to many relationship from courses to courses.
2. **Course Offerings**: We have a many to many relationship "Offers" from faculty to course that is each faculty can offer multiple course and each course can be offered by multiple faculty members. The "Course ID" here refers to the "Course ID" attribute in "Courses" table. We have a table for "Allowed Batches" which has "Offer ID" which refers to "Offers" table, "Batch" and "Department" as attributes. The "Offers" relationship has an attribute "Minimum Grade". So the student can register for a course only if his/her Grade is greater that this.

Implementation Details II

3. **Student Registration**: We have a many to many relationship from "Students" to "Courses" thus a student can take many courses. The "1.25 rule" can be implemented in the following way:

   We will group rows by the given "Student ID" and last semester and find sum of the credits of all those courses by referring to the "Course" relation and we will do the same for second last semester and calculate the average of both the sums and multiply it with 1.25. This will be the maximum number of credits a student can register in current semester. We will be using the check constraints to check if the total credits registered by a student is less than this limit.

Implementation Details III

4. **Ticket Generation**: We have a relationship for tickets between "Students", "Faculty" and "Courses" which has "Offer ID" and status. Status is an integer which can have following values:

Status ID	Description
0	Ticket just generated by student
1	Accepted and closed by Faculty
2	Rejected by Faculty
3	Accepted and Forwarded by Faculty
4	Accepted and Closed by Faculty Advisor
5	Rejected by Faculty Advisor
6	Accepted and Forwarded by Faculty Advisor
7	Accepted and Closed by HoD
8	Rejected by HoD
9	Accepted and Forwarded by HoD
10	Accepted and Closed by Dean
11	Rejected by Dean

Implementation Details IV

If a student wants to register for a course for which he is not eligible he can generate a ticket corresponding to which and entry will be added to the Ticket table with status 0 and the status will be updated corresponding to Approval/Rejection by respective authorities. Finally if status is 1/4/7/10 then we will insert the entry of this course into "Takes" relationship and delete it from "Ticket" relationship.

5. **Report Generation**: Staff of Dean's Office will be able to access the grades of all the students so they can view the Grades of all the students and hence create various kinds of reports like student transcripts, list of probation students, Aggregate views of grades.
6. **Grade Entry by Course Instructors**: The Faculty will have the sole access to update the grade attribute of the "Takes" relationship.

Various Checks, Stored Procedures, Constraints and Privileges to be implemented I

1. **To ensure that a student is not allowed to register for courses which are scheduled in the same Time Slot**: When a student wants to register for a course we access the "Takes" relation and select the rows having same "Student ID" and then using the "Section ID" we access the section table and check whether the Time Slot of current course conflicts with any of the Time Slot IDs in the selected rows. And we add this as a Check Constraint while inserting the entry into "Takes" table.

Various Checks, Stored Procedures, Constraints and Privileges to be implemented II

2. **Checking Prerequisites**: If a student wants to register for a particular course, the prerequisites are checked. Using the "Course ID" we select a list (say List-I) of "Course ID"s of all the prerequisites of that course from the Prerequisites relationship and in the "Takes" table we select the "Course ID"s from rows having same "Student ID" where Grade greater than 2 (for passing a course grade should be greater than 2, i.e. E) (say List-II) and we check if all the entries in List-I are present in List-II by using ALL in SQL queries. We add this as a check constraint while inserting the entry into "Takes" table.
3. We grant permission to update the "Courses" relation only to the Staff Dean's Office and Dean Academics using the GRANT command in SQL.

Various Checks, Stored Procedures, Constraints and Privileges to be implemented III

4. To ensure that a student can see only his/her grades and a Faculty, HoD, Dean Academics and Staff can see all the grades, we will grant access to SELECT only those rows from "Takes" table WHERE Takes.Student_ID = Student.ID. And permissions to SELECT and UPDATE all the entries will be granted to Dean Academics and Staff Dean's Office using the GRANT command of SQL.
5. It is a tedious procedure to calculate a Grades of a student from the database and it contains many SQL commands as well as arithmetic operations. As this procedure is used multiple times, we will create a stored procedure for calculating the Grade of a student.

Various Checks, Stored Procedures, Constraints and Privileges to be implemented IV

```
1  CREATE PROCEDURE calculate_CGPA
2  @Student_ID Integer
3  AS
4  SELECT S / Credits
5  FROM
6          (SELECT SUM(Grade * (Courses.L +
              Courses.T + Courses.P/2)) FROM
               Takes, Courses) AS S,
7          (SELECT SUM(Courses.L + Courses.T
               + Courses.P / 2) FROM Takes,
              Courses) AS Credits
8  WHERE
9          Courses.ID=Takes.Course_ID AND
10         Takes.Student_ID = @Student_ID
11 GO
```

Various Checks, Stored Procedures, Constraints and Privileges to be implemented V

6. If a student wants to register for a course we calculate his grade using the stored procedure and then compare it with the "Minimum Grade" in the "Offers" relationship with corresponding "Course ID", "Semester", "Faculty ID" and "Year".
7. We can also have a stored procedure for calculating the total credits of a student by adding all the credits in the courses that he passed in the "Takes" relationship.

Use of Triggers

- In case any faculty leaves, the corresponding rows are deleted in the "Faculty" relation and all the rows in the "Offers" relationship with the given Faculty ID are automatically deleted by the use of trigger.
- Using the BEFORE option with the Trigger, the code within the trigger will be executed before the INSERT into the table occurs. This will be used to verify the input values of the INSERT. For example, if the last name of the student is more than 10 letters, before insertion or updation, we take the substring and change the name value to this.

Assertions

- Check each department has only one HoD.
- Two Dean's durations must not collide.