LOW LEVEL DESIGN(LLD)

TABELS WITH ATTRIBUTES & KEYS

STUDENT Table:

Attributes

- Std_id
- Phn_no
- Email
- Address
- DOB
- Admission_date
- Dept_id

Keys

- Primary Key: Std_id
- Foreign Key: Dept_id
- Unique Key: Phn_no,Email

DEPARTMENT Table:

Attributes

- Dept_id
- Dept_name
- Total_faculty
- Prof_id

Keys

- Primary Key: Dept_id
- Foreign Key: Prof_id
- Unique Key: Dept_name

PROFESSOR Table:

Attributes

- Prof_id
- Prof_name
- Designation
- Dept_id
- Phn_no
- Email
- DOB
- Salaray
- Join_date

Keys

- Primary Key: Prof_id
- Foreign Key: Dept_id
- Unique Key: Phn_no,Email

COURSE Table:

Attributes

- Course_id
- Course_name
- Dept_id
- Credits
- Std_Capacity

Keys

- Primary Key: Course_id

- Foreign Key: Dept_id
- Unique Key: Course_name

ENROLLMENT Table:

Attributes

- Enroll_id
- Std_id
- Course_id
- Sem_id
- Enroll_date

Keys

- Primary Key: Enroll_id
- Foreign Key: Std_id,Course_id,Sem_id

SEMSTER Table:

Attributes

- Sem_id
- Start_date
- End_date
- Year

Keys

- Primary Key: Sem_id

TEACHES Table:

Attributes

- Teach_id
- Prof_id
- Course_id

Keys

- Primary_Key: Teach_id
- Foreign_Key: Prof_id,Course_id

PREREQUISTIES Table:

Attributes

- Pre_id
- Pre_course_id
- Course_id

Keys

- Primary Key: Pre_id
- Foreign Key: Course_id

GRADE Table:

Attributes

- Grade_id
- Grade_val
- Grade_point
- Enroll_id

Keys

- Primary Key: Grade_id
- Foreign Key: Enroll_id

RELATIONSHIP MAPPINGS:

1.One to Many Relationship

Department->Professor

- One department has many professors

Department->Student

- One department has many students

Department->Courses

- One department has many courses
  Courses->Enrollment
- Courses can have many enrollments
  Student->Enrollment
- Student can enroll in many courses
  Semester->Enrollment
- Semester includes many enrollments
  Enrollment ->Grade
- Each enrolment has one grade

2.Many to Many Relationships

Professor<->Courses

- Professor teaches many courses
- Course is taught by many professors
- Teaches table is link table
  Courses<->Prerequisites
- Course may have multiple prerequisites

3.One to One Relationships

Department->Professor(HoD)

- Department.HoD refers exactly one Professor

NORMALIZATION EXPLANATION:

First Normal form(1NF):

A relation is in 1NF when all its attributes hold indivisible values and there are no repeating fields.
In our database design every column stores single value. For example, each Student has exactly one phone number and one email, each student belongs to only one department and no tables contains repeating values.
Therefore, all tables satisfy 1NF rules.

Second Normal Form(2NF):

A relation is in 2NF if it satisfies the conditions of 1NF and additionally. No partial dependency exists, meaning every non-key attribute must depend on the entire primary key, not just a part of it.
In our database design ,every table uses a single-column primary key such as prof_id,dept_id,std_id,course_id ,etc since no tables uses composite key ,partial dependencies cannot occur.
Thus, all tables in the database are already in 2NF.

Third Normal Form(3NF):

A relation is in 3NF if it satisfies 2NF and additionally, there are no transitive dependencies. In simpler terms, non-prime attributes should not depend on other non-prime attributes.
In our database design , In the Professor table, attributes like name, designation, phone, salary, and join date all depend only on prof_id. Department details such as dept_name are stored separately in the Department table, so no transitive dependency exists.
In the Student table, attributes like name, phone number, email, and address depend directly on std_id. The student's department information is linked via dept_id, not stored within Student.
In the Courses table, course information (name, credits, capacity) depends only on course_id. Department-related attributes remain in the Department table.
Relationship tables such as enrollment, Grade, Teaches, and Prerequisites contain only their primary key and foreign keys, along with attributes that depend solely on their primary key.
Since each non key attribute depends directly on the primary key and not on another non key attribute, all relations satisfy 3NF conditions.

INDEXING STRATEGY:

Purpose of Indexing

Indexing in DBMS is used to speed up data retrieval by minimizing disk scans. Instead of searching through all rows, the DBMS uses index structures to quickly locate data using key values.
When an index is created, it stores sorted key values and pointers to actual data rows. This reduces the number of disk accesses, improving performance especially on large datasets.

Indexing Decisions in this Database

Primary Keys Indexes:
Primary keys generate indexes automatically ,depending on this database all primary keys-std_id,prof_id,course_id,enroll_id,grade_id,dept_id,sem_id all are automatically create clustered or non-clustered Indexes.

Foreign Key Indexes:
To enhance join performance ,creates index for foreign keys – Student(dept_id),Professor(dept_id),Courses(dept_id),Enrollment(std_id),Enrollment(course_id),Enrollment(sem_id),Grade(enroll_id),Teahces(prof_id),Teaches(course_id) .These indexes optimizes queries involving subqueries,joins.

TRANSACTION PSEUDO CODE:

BEGIN TRANSACTION;
Step 1: Check if course is seats available
IF available_seats > 0 THEN
Step 2: Check if the student has completed prerequisites
IF prerequisites_completed = TRUE THEN
Step 3: Insert enrollment record
INSERT INTO Enrollment;
Step 4: Reduce course seat count
UPDATE Course SET seats = seats - 1;
COMMIT;
PRINT "Registration Successful";
ELSE
ROLLBACK; PRINT "Cannot Register: Prerequisite Not Completed";
ELSE
ROLLBACK; PRINT "Cannot Register: Course is Full";

SECURITY ROLE DESIGN(RBAC):

It ensures that each user only accesses data required for their responsibilities
Role 1: Admin
Privileges:
Provides full rights on all table CREATE,ALTER,DROP,SELECT,INSERT,UPDATE,DELETE
Can handle user and role management
Query:
CREATE ROLE admin_role;
GRANT ALL PRIVILEGES ON DATABASE CollegeDB TO admin_role;
Role 2: Professor
Privileges:
Can only view student records ,but upgrade grades
Cannot modify student or course data
Query:
CREATE ROLE professor_role;
GRANT SELECT ON Enrollment TO professor_role;
GRANT SELECT ON Student TO professor_role;
GRANT SELECT ON Courses TO professor_role;

GRANT UPDATE (Grade_val, Grade_point) ON Grade TO professor_role;

Role 3: Student

Privileges:

Cannot modify the academic records ,read only access

Query:

```
CREATE ROLE student_role;
GRANT SELECT ON Grade TO student_role;
GRANT SELECT ON Enrollment TO student_role;
```