College Database Management System

Problem Statement

A college contains many departments. Each department can offer any number of courses. Many instructors can work in a department, but an instructor can work only in one department. For each department, there is a head, and an instructor can be head of only one department. Each instructor can take any number of courses, and acourse can be taken by only one instructor. A student can enroll for any number of courses and each course can have any number of students.

Group Members (Group IV)

- BT20CSE159 Shubham Hambarde
- BT20CSE160 Prajwal Alaladinni
- BT20CSE161 Anantha Sneha Roopa Sri
- BT20CSE164 Prashant Tiwari
- BT20CSE166 Maitreya Zalte
- BT20CSE167 Khushiram Meena

Requirement Analysis :

- College has several departments.
- College has several instructors.
- Several instructors can work in one department.
- An instructor can work in only one department.
- Every department has a head who is an instructor.
- An instructor can head only one department.
- Each department can offer any number of courses.
- An instructor can only take a course offered by his department.
- Each instructor can take any number of courses.
- A course can be taken by only one instructor.
- College has several students.
- A student can enroll for any number of courses.
- Each course can have any number of students.

Assumptions :

- Every department has a unique *Department Numbe*r and *Department Name*.
- Every course has a globally unique *Course ID* and *Course Name*.
- A person in general (instructor or student) has a *First Name*, *Middle Name*
- (optional), *Last Name* (optional) and *Gender*.
- Every instructor, in addition, has a unique *Instructor ID*.
- Every student, in addition, has a unique *Student ID*.
- Both instructors and students have a saved *Password Hash* required for login.
- Every course has a count of the *ClassesTaken* it is taken by its instructor.
- Every course associated with a student has a record of the *Attendance* and *Marks* obtained by the student in that course.

Entity Sets :

As per the given constraints, the entity sets are as follows –

- Department
- Course
- Student
- Instructor

Head of the Department (HOD) is not an entity set. It is a relationship between the instructor and department entities.

Relationship Between Entity Sets :

- The department offers multiple courses and each course belongs to only one department, hence cardinality between department and course is one to many.

- One course is undertaken by multiple students and one student for multiple courses. Hence, relationships are many to many.

- One department has multiple instructors and one instructor belongs to one and only one department, hence the relationship is one to many.

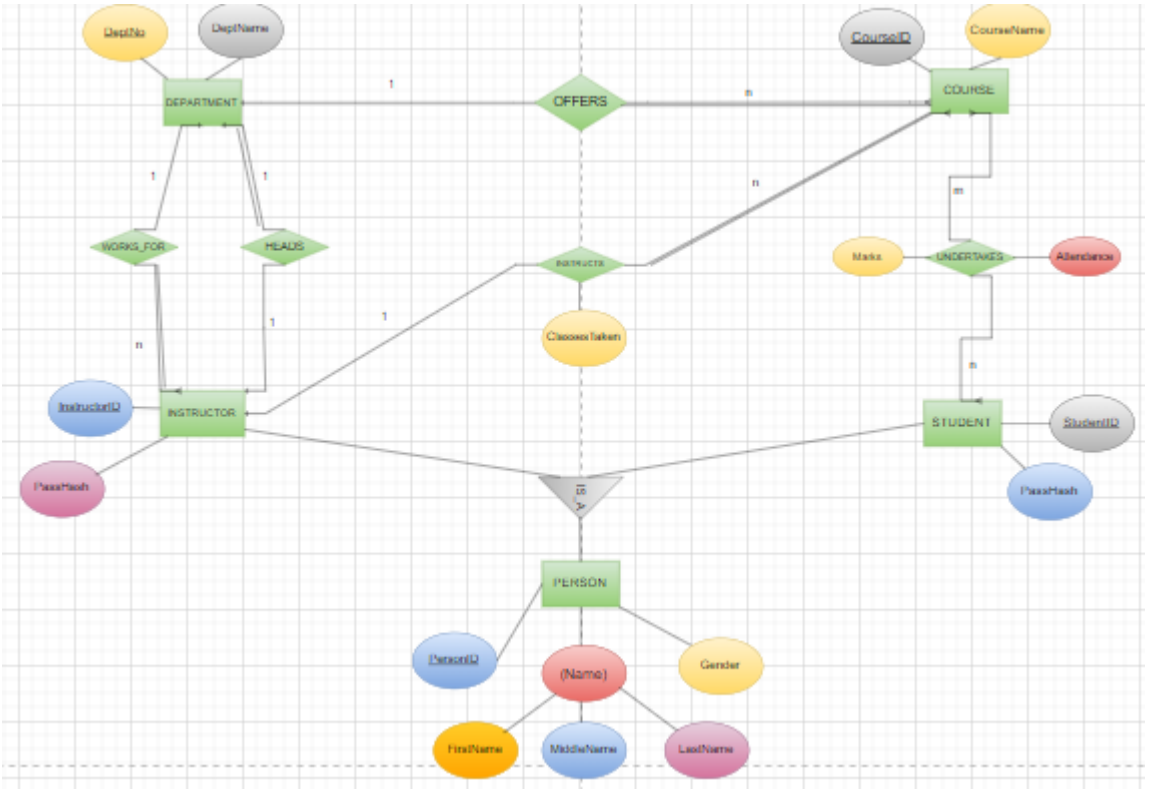
- Each department has one “HOD” and one instructor is “HOD” for only one department, hence the relationship is one to one. Here, HOD refers to the head of the department.

- One course is taught by only one instructor but one instructor teaches many courses hence the relationship between course and instructor is many to one.

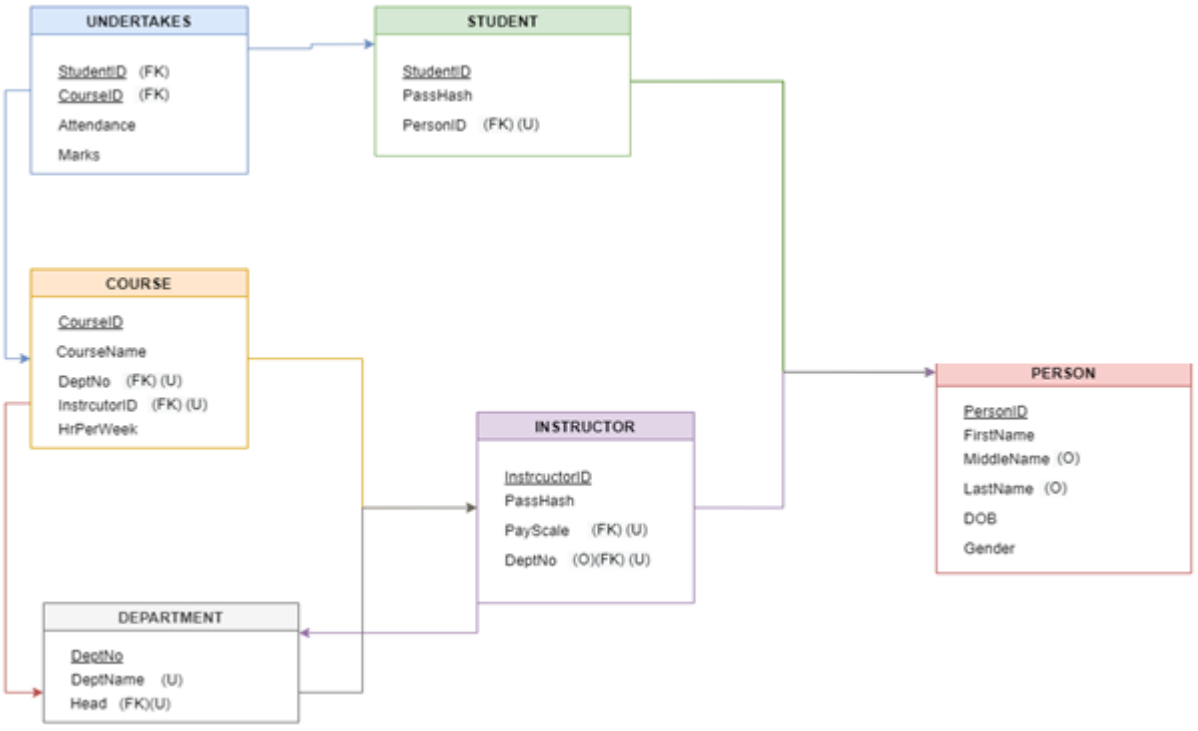
The relationship between instructor and student is not defined because of the following reasons –

- There is no significance in the relationship.
- We can always derive this relationship indirectly through course and instructors, and course and student.

ER Diagram :


DeptNo
DeptName
DEPARTMENT
OFFERS
CourseID
CourseName
COURSE
WORKS_FOR
HEADS
INSTRUCTS
Marks
UNDERTAKES
Attendance
ClassesTaken
InstructorID
INSTRUCTOR
STUDENT
StudentID
PassHash
PassHash
IS
PERSON
PersonID
(Name)
Gender
FirstName
MiddleName
LastName

Relational Schema :



Tables :

DEPARTMENT

DeptNo	DeptName	**Head**

PERSON

PersonID	FirstName	MiddleName	LastName	DOB	Gender

INSTRUCTOR

InstructorID	PassHash	**PersonID**	PayScale	**DeptNo**

COURSE

CourseID	CourseName	**DeptNo**	**InstructorID**	HrPerWeek

STUDENT

StudentID	PassHash	**PersonID**

UNDERTAKES

StudentID	**CourseID**	Attendance	Marks

Normalization :

First Normal Form :

In all tables, every attribute is atomic and takes only one value from its domain. Hence all are in their First Normal Form (1NF).

Second Normal Form :

Super keys of the tables are as:

Table	Super Key
DEPARTMENT	DeptNo
PERSON	PersonID
PHONE	PersonID
INSTRUCTOR	InstructorID
COURSE	CourseID
STUDENT	StudentID
UNDERTAKES	StudentID, CourseID

In all but the last table, there is only one attribute forming the super key. Hence, in all such tables, the super key is the candidate key. Therefore, the relations are in their Second Normal Form (2NF).

In table *UNDERTAKES*, the functional dependencies are:

$(StudentID, CourseID) \longrightarrow Attendance$

$(StudentID, CourseID) \longrightarrow Marks$

Removing either StudentID or CourseID cannot identify the Attendance and Marks uniquely. Hence, the super key is the candidate key. Therefore, the relation is in its Second Normal Form (2NF).

Third Normal Form :

The functional dependencies are:

DEPARTMENT:

$DeptNo \longrightarrow DeptNo$

$DeptNo \longrightarrow DeptName$

$DeptNo \longrightarrow Head$

$DeptName \longrightarrow Head$

We can remove the transitive dependency $DeptNo \longrightarrow DeptName \longrightarrow Head$.Hence, new FDs are:

$DeptNo \longrightarrow DeptNo$

$DeptNo \longrightarrow DeptName$

$DeptNo \longrightarrow Head$

PERSON:

$PersonID \longrightarrow PersonID$

$PersonID \longrightarrow FirstName$

$PersonID \longrightarrow MiddleName$

$PersonID \longrightarrow LastName$

$PersonID \longrightarrow PersonID$

$\longrightarrow Gender$

There are no transitive dependencies; relation is in its Third Normal Form (3NF).

There are no transitive dependencies; relation is in its Third Normal Form (3NF).

INSTRUCTOR:

InstructorID

$\longrightarrow$*InstructorID*

InstructorID $\longrightarrow$

PassHashInstructorID $\longrightarrow$

PersonID InstructorID $\longrightarrow$

PayScale InstructorID $\longrightarrow$
DeptNo

There are no transitive dependencies; relation is in its Third Normal Form (3NF).

COURSE:

CourseID $\longrightarrow$*CourseID*

CourseID $\longrightarrow$*CourseName*

CourseID $\longrightarrow$ *DeptNo*

CourseID $\longrightarrow$

InstructorID CourseID $\longrightarrow$
ClassesTaken

There are no transitive dependencies; relation is in its Third Normal Form (3NF).

STUDENT:

StudentID $\longrightarrow$*StudentID*

StudentID $\longrightarrow$ *PassHash*

StudentID $\longrightarrow$*PersonID*

There are no transitive dependencies; relation is in its Third Normal Form (3NF).

UNDERTAKES:

(*StudentID*, *CourseID*) $\longrightarrow$*StudentID,*

CourseID(*StudentID*, *CourseID*) $\longrightarrow$

Attendance (*StudentID*, *CourseID*) $\longrightarrow$*Marks*

There are no transitive dependencies; relation is in its Third Normal Form (3NF).

Boyce-Codd Normal Form :

In all tables, for all functional dependencies, the L.H.S. is a super key. Hence, all relations are in their Boyce-Codd Normal Form (BCNF).

Normalised Tables :

DEPARTMENT

DeptNo	DeptName

HEAD

DeptNo	Head

PERSON

PersonID	FirstName	MiddleName	LastName	DOB	Gender

INSTRUCTOR

InstructorID	PassHash	PersonID	PayScale	DeptNo

COURSE

CourseID	CourseName	DeptNo	InstructorID	HrPerWeek

STUDENT

StudentID	PassHash	PersonID

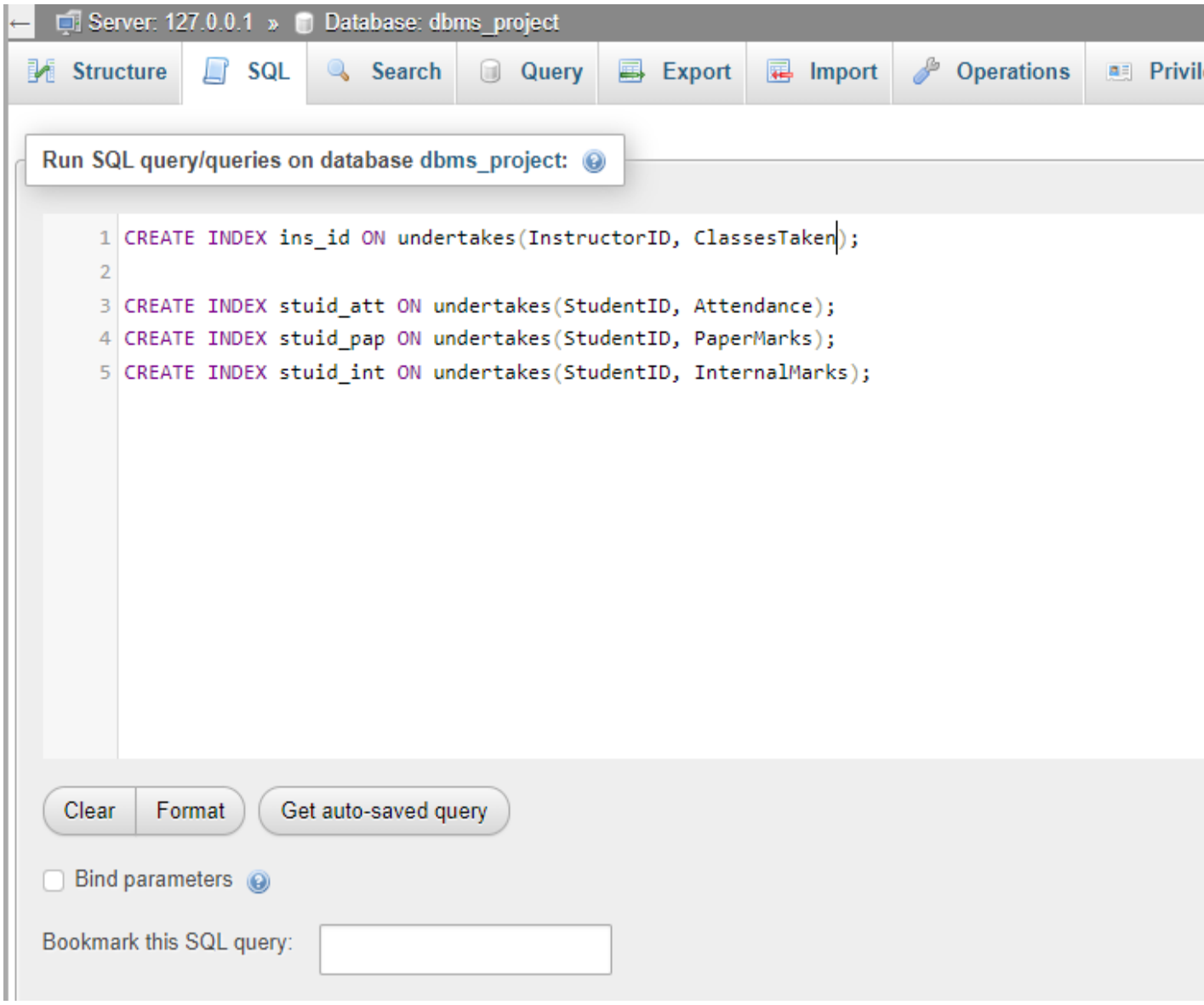
UNDERTAKES

StudentID	CourseID	Attendance	Marks

Functionality :

- Administrator, Instructor and Student have Login privileges.
- Students can **VIEW** their profile(personal details, marks, attendance & grades), **ENROL** in any course and view exam timetable.
- Instructors can view their profiles which include the courses they have taken and **UPDATE** the student info in those courses.
- Department Heads can change the info related to their respective departments.
- Administrator can view all the info **edit** *all information* about instructors and students.
    - Admins can **INSERT, UPDATE & DELETE** Data from Student & Instructor Records, also can **EDIT** Exam Timetable.

Use Of Indexing :


Server: 127.0.0.1 » Database: dbms_project
Structure
SQL
Search
Query
Export
Import
Operations
Run SQL query/queries on database dbms_project:
1 CREATE INDEX ins_id ON undertakes(InstructorID, ClassesTaken);
2
3 CREATE INDEX stuid_att ON undertakes(StudentID, Attendance);
4 CREATE INDEX stuid_pap ON undertakes(StudentID, PaperMarks);
5 CREATE INDEX stuid_int ON undertakes(StudentID, InternalMarks);
Clear
Format
Get auto-saved query
Bind parameters
Bookmark this SQL query:

****