**IT-301**

**DBMS Mini Project**

**Course Registration System**

Submitted by:

Raghav 13IT129

Avinash Giri 12IT20

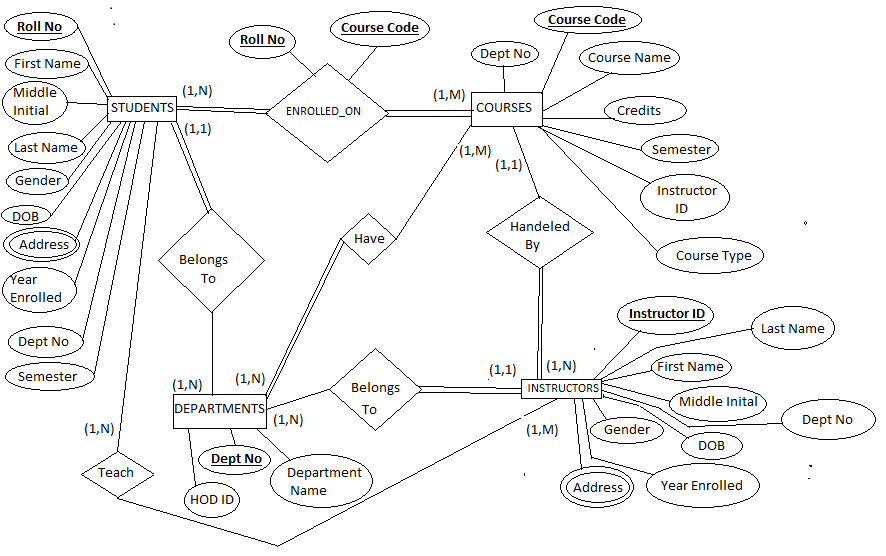
Mishal Kumar 13IT120

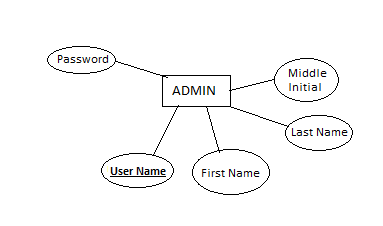
Kishan S 13IT216

**PROBLEM STATEMENT:**

Create a database management system for an INSTITUTE to carry out its course registration system based on the certain requirements of it’s STUDENTS, DEPARTMENTS, COURSES, INSTRUCTORS and ADMIN.

**ER DIAGRAM:**





**Design Schema:**

**INSTRUCTORS**

instructorId

firstName

midInit

lastName

gender

dob

address

yearEnrolled

deptNo

**ENROLLED\_ON**

rollNo int

courseCode

**STUDENTS**

rollNo

firstName

midInit

lastName

gender

dob

address

yearEnrolled

deptNo

semester

**ADMIN**

username

firstName

midInit

lastName

password

**DEPARTMENTS**

deptNo

deptName

hodId

**COURSES**

courseCode

courseName

credits

semester

deptNo

instructorID

**Functional Dependencies:**

1. In table **STUDENTS** :

RollNo -> { FirstName, MidInit, LastName, Gender,Dob, Address, YearEnrolled, DeptNo, Semester }

{ RollNo, YearEnrolled }-> Semester

YearEnrolled -> Semester

1. In table **COURSES:**

CourseCode -> { CourseName, Credits, Semester, DeptNo, InstructorID}

1. In table **DEPARTMENTS** :

DeptNo -> {deptname, HODId}

1. In table **Instructors**:

InstructorId -> { FirstName, MidInit, LastName, Gender,Dob, Address, YearEnrolled, DeptNo }

1. In table **ADMIN**:

UserName -> { FirstName, MidInit, LastName, Password }

**Normalization:**

1. The course a student has enrolled on is show is a separate table ENROLLED\_ON. We do not show this details along with the STUDENTS table to prevent redundancy of values as the same student can enroll in a number of courses.
2. We have a separate table for instructors called INSTRUCTORS. We do not display the instructor information in the DEPARTMENTS table to prevent redundancy as a department can have many instructors.

**TRIGGERS:**

1. Trigger to insert into ADMIN table after insertion of new student into STUDENTS table.

DELIMITER //

CREATE TRIGGER usernm\_insert

AFTER INSERT ON STUDENTS

FOR EACH ROW BEGIN

INSERT INTO ADMIN (UserName,FirstName,MidInit,LastName,Password)

values(NEW.RollNo,NEW.FirstName,NEW.MidInit,NEW.LastName,NEW.RollNo);

END;//

DELIMITER ;

1. Delete from STUDENTS table where a student entry is removed from ADMIN table

DELIMITER //

CREATE TRIGGER usernm\_delete

AFTER DELETE ON ADMIN

FOR EACH ROW BEGIN

DELETE from STUDENTS where RollNo = OLD.UserName;

END;//

DELIMITER ;

**PHYSICAL DESIGN:**

**STUDENTS:**

RollNo (4 bytes)

FirstName (25 bytes)

MidInit (1 byte)

LastName (25 bytes)

Gender (1 byte)

Dob (3 bytes)

Address (150 bytes)

YearEnrolled (4 bytes)

DeptNo (5 bytes)

Semester (2 bytes)

Total bytes: 4+25+1+25+3+150+4+5+2 = 219 bytes

Blocking factor = [512/219] = 2

Assume r=5000 (Number of records).

No. of file blocks required=[5000/2]=2500 blocks.

Storage Requirements:

* By choosing “unspanned record blocking”, the no. of bytes wasted/block =

512-219\*2= 74 Bytes.

* This makes each record effectively use 512/2 = 256 bytes of storage. So the overhead is 37 bytes/record.
* Spanned recod blocking:

=(5000\*219)/512 = 2139 records

Hence here spanned record blocking is better:

In addition IBG(Inter Block Gap) uses upto 106 bytes/block

* Total storage required for STUDENTS File on disk =

(Block Size + Inter-Block-size) \* 2139 = ( (512+106) \* 2139) = **1.26 MB**

**QUERY COST CALCULATIONS:**

1. SELECT \* FROM STUDENTS S,ADMIN A WHERE S.RollNo =A.userName

Nested Loop JOIN:

For each tuple s in S do

for each tuple a in A do

if s.rollno == a.username then

add to result

end if

end loop

End loop

Worst Case: When 1 buffer each allocated for each of the tables (Only 2 buffers for two tables)

Cost (Worst Case ): A + (A \* bfr) \* S = 5000 + (5000 \* 10 \* 5000) = 250005000

Worst Case I/Os (With 2 buffers): 250005000

bfr = Records per Block (Blocking Factor) = 10

Best Case: When both the tables completely fit in memory

Cost (Best Case ): A + S = 5000 + 5000

Best Case I/Os : 10000

**Block Nested Loop JOIN:**

for each block S of s do begin

for each block A of a do begin

for each tuple a in A do begin

for each tuple s in S do begin

Check if (a,s) satisfy the join condition

if they do, add a • s to the result.

end

end

end

end

• Block Nested:

Cost (Worst Case): A + A \* S = 5000 + 5000\*5000

I/Os: 25005000

Cost (Best Case): A+ S = 5000 + 5000

– I/Os: 10000

1. Select \* from COURSES C , STUDENTS S where C.deptNO = S.deptNO and C.semester= S.semester;

Nested Loop JOIN:

Worst Case: When 1 buffer each allocated for each of the tables (Only 2 buffers for two tables)

Cost (Worst Case ): C + (C \* bfr) \* S = 9 + (9 \* 10 \* 5000) = 450009

Worst Case I/Os (With 2 buffers): 450009

bfr = Records per Block (Blocking Factor) = 10

Best Case: When both the tables completely fit in memory

Cost (Best Case ): C + S = 9 + 5000

Best Case I/Os : 5009

**Block Nested Loop JOIN:**

• Block Nested:

Cost (Worst Case): C + C \* S = 9 + 9\*5000

I/Os: 45009

Cost (Best Case): C+ S = 9 + 5000

– I/Os: 5009

1. Insert into ENROLLED\_ON (RollNo,CourseCode)

VALUES (“RollNo”,”CourseCode”);

Best Case: When the table is empty.

I/Os: 1

Worst Case : When the table is filled with n-1 entries.

I/Os: ( No of Courses \* No of Students)