

School Management System

Table of Contents:

1. Introduction
2. Conceptual Design
3. Description of Entities
4. Logical Design
5. Physical Design
6. CRUD Operations
7. Scope of the Project
8. Problems Faced
9. Lessons Learned
10. Future Scope
11. Requirement Table

Introduction:

The domain of our project is the School Management System. This is a platform created to facilitate an organization's efficient operation by digital and automating numerous academic and administrative processes. This will make it possible to manage procedures like admission, online payment, submission, attendance, online class, and online examinations without using paper. It can lessen staff workload, enable cost reduction, improve data security, and help both students and teachers save time. All of these will eventually raise the productivity and cost-effectiveness of an institution.

Conceptual Design:

This database design process identifies the entities, properties, and relationships between the entities.

This Database Design consists of 8 entities. They are -

Entity	Attributes
Courses	CourseID, CourseName
Enrollment	EnrollmentID, StudentID, LectureID, CourseID, Term

Exams	ExamID, Date, CourseID
Grades	Grade, MinMarks, MaxMarks
Lecturers	LectureID, LectureName, Email, PhoneNumber
Marks	ExamID, EnrollmentID, Marks
Parent	ParentID, StudentID, ParentName, Email, PhoneNumber, Address
Student	StudentID, FirstName, LastName, Email, DOB, Gender, Major

Description of Entities:

1. Courses: This entity contains details of the course such as CourseID and CourseName
2. Enrollment: This entity contains information like EnrollmentID, StudentID, LectureID, CourseID and Term. Every student's chosen course must be set by the school's administration, who must then designate suitable instructors for the topics covered by each course.
3. Exams: This entity contains information about exams like ID, Date of the exam, and CourseID.
4. Grades: This entity contains information about grades, the grade of a student is decided based on the range of Max and Min marks obtained.
5. Lecturers: This entity contains information about Lecturers such as ID, Name, Email, and PhoneNumber.
6. Marks: This entity contains Student marks. Using Exam and EnrollmentID a student can access his/her marks.
7. Parent: This entity contains Parent details such as Parent Name, email, Phone number, and Address.
8. Student: This entity plays a significant role because it compiles vital data about the students. It contains full details of the student such as StudentID, FirstName, LastName, Email, DOB, Gender, and Major. Each student is uniquely identified by StudentID.

Logical Design :

The database design process is now in its second stage. It is a method of converting conceptual schema's entities, attributes, and relationships into a data model assisting DBMS, such as the relational or object-oriented data model. It also includes creating data types and defining primary/foreign key links between the tables. The problem/s with querying the logical schema's cost and convenience of use, as well as the expenses of constraint maintenance and storage, are

all addressed by this. Entity-Relationship diagrams and relational databases are primarily the subjects of this entry.

Two steps are-

1. Normalisation:

Normalization avoids data duplication and gets rid of undesired traits. The rules of normalization break huge tables into smaller tables and link them using relationships. SQL normalization serves the dual purpose of removing unnecessary data and ensuring logical data storage.

2. Entity Relationship Diagram:

A graphical representation that shows relationships between individuals, things, locations, concepts, or events within an information technology (IT) system is called an entity relationship diagram (ERD), also known as an entity-relationship model. Data modeling methods are used in an ERD to help describe business processes and provide the framework for a relational database.

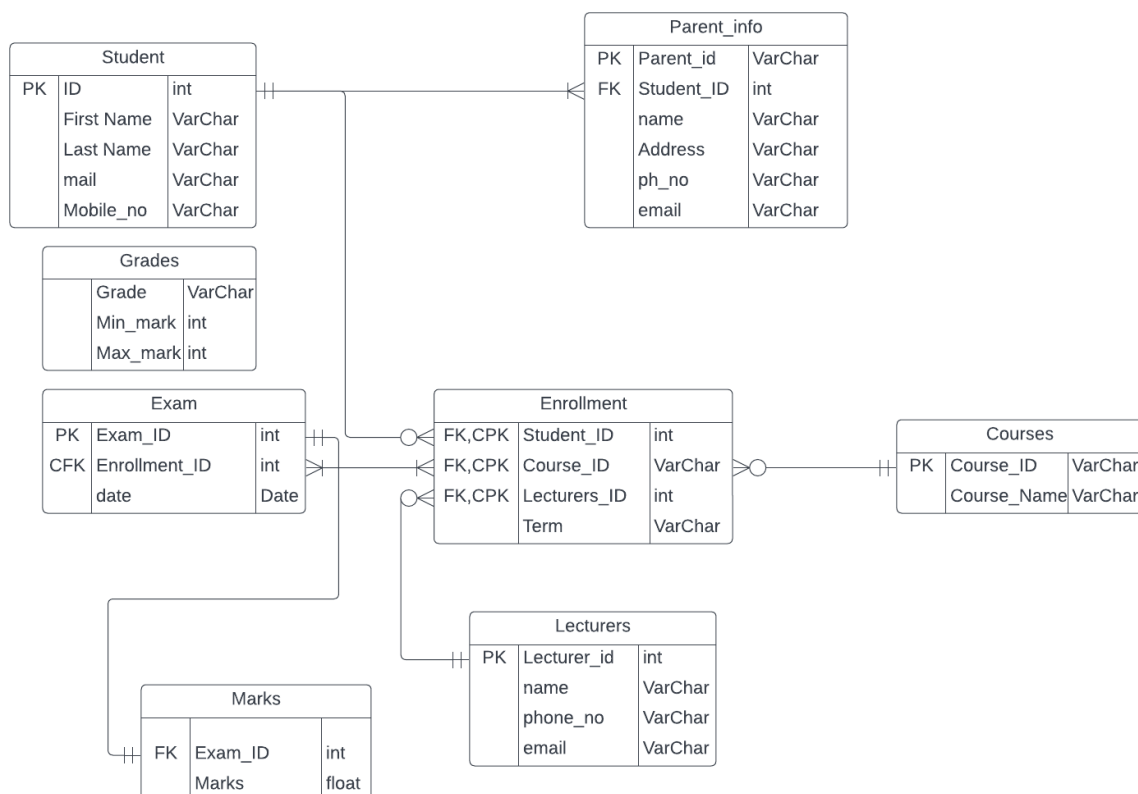
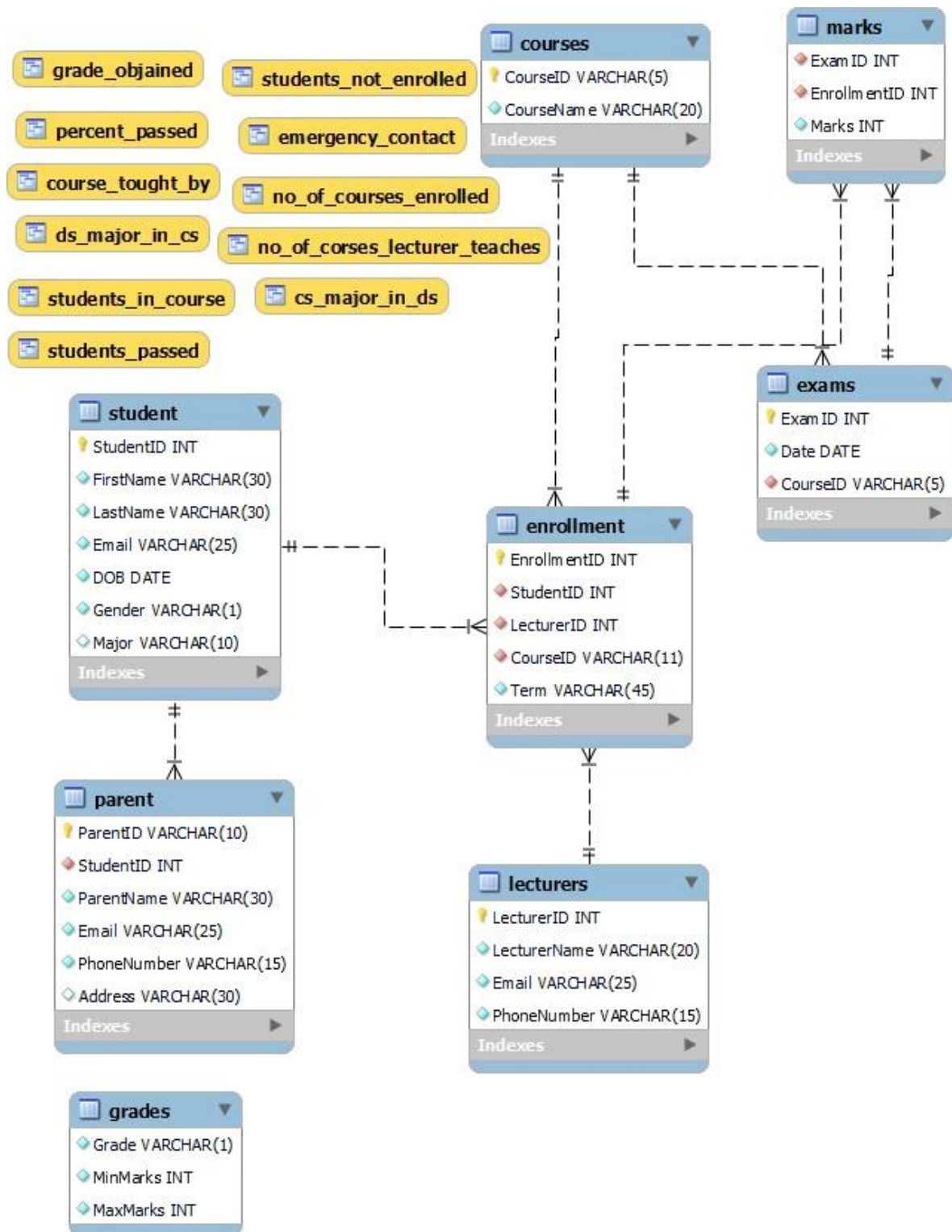


Figure 1: ER Diagram

Physical Design:

This is the third stage of the database design. The physical design is where the translation of schemas into actual database structures takes place. After that, you convert the entities into tables, instances into rows, and attributes into columns. This requires setting up the database on the MySQL server. Since we normalise the relations before implementing, there were no changes made to the ERD. In order to perform the CRUD actions, we first insert the sample data into the tables.

The forward ERD enrollment table was expected to have a composite primary key of student id, course id, and lecturer id but it had no physical significance so to keep it simple made serial no as a primary key. Missed a few entities like date of birth, phone no, and address in a few tables but added them during physical design.



CRUD Operations :

Query 1 : To Retrieve grade and marks obtained by a student in each course.

```
Select StudentID, CourseName, Marks, Grade from marks  
left join grades on marks.Marks >= grades.MinMarks and marks.Marks <= MaxMarks  
left join enrollment using(EnrollmentID)  
left join student using(StudentID)  
left join courses using(CourseID);
```

StudentID	CourseName	Marks	Grade
100	OS	65	D
101	OS	86	B
102	Intro to DS	86	B
104	OS	29	F
105	Intro to DS	53	F
106	Data Management	27	F
107	Intro to DS	24	F
108	Networks	57	E
109	OS	62	E
110	OS	47	F
100	Intro to DS	90	B
101	DAA	29	F
102	Networks	43	F
104	CS ethics	44	F
105	Capstone in DS	94	A
106	DAA	57	E
107	CS ethics	73	C
108	DAA	85	B
109	Intro to DS	28	F
110	Data Management	82	B
104	Capstone in DS	93	A
105	Machine Learning	27	F
106	Big Data Processing	21	F
107	Capstone in DS	98	A
108	Big Data Processing	49	F

Query 2 : Total number of students for each course who gave exam and results were graded

```
select CourseName , count(Grade)
from (select StudentID,CourseName, Marks, Grade from marks
left join grades on marks.Marks>= grades.MinMarks and marks.Marks<=MaxMarks
left join enrollment using(EnrollmentID)
left join student using(StudentID)
left join courses using(CourseID)) as graded
group by CourseName;
```

	CourseName	count(Grade)
►	OS	5
	Intro to DS	5
	Data Management	2
	Networks	2
	DAA	3
	CS ethics	2
	Capstone in DS	3
	Machine Learning	1
	Big Data Processing	2

Query 3 : Number of Students who passed the exams for each course

```
select CourseName , count(Grade)
from (select StudentID,CourseName, Marks, Grade from marks
left join grades on marks.Marks>= grades.MinMarks and marks.Marks<=MaxMarks
left join enrollment using(EnrollmentID)
left join student using(StudentID)
left join courses using(CourseID)) as graded
where Grade != "F"
group by CourseName;
```

	CourseName	count(Grade)
▶	OS	3
	Intro to DS	2
	Networks	1
	Capstone in DS	3
	DAA	2
	CS ethics	1
	Data Management	1

Query 4 :To view lecturer for each course

```
select distinct( CourseName), LecturerName
from courses
join enrollment using (CourseID)
join lecturers using (LecturerID);
```

	CourseName	LecturerName
▶	OS	Tima Kshetry
	Big Data Processing	Tima Kshetry
	Data Management	Sara Swaden
	DAA	Sara Swaden
	Networks	John Wan
	CS ethics	Ergun Simsik
	Capstone in DS	Ergun Simsik
	Intro to DS	Masood Souresh
	Machine Learning	Mehmet Sarica

Query 5 : How many subjects a professor is teaching

```
select LecturerName, count(CourseName) No_Of_Courses_Tought
from
(select distinct( CourseName), LecturerName
from courses
join enrollment using (CourseID)
join lecturers using (LecturerID)) as CoursesTought
group by LecturerName
order by No_Of_Courses_Tought desc;
```


	LecturerName	No_Of_Courses_Tought
▶	Tima Kshetry	2
	Sara Swaden	2
	Ergun Simsik	2
	John Wan	1
	Masood Souresh	1
	Mehmet Sarica	1

Query 6 : Parent contact details for each student

```
select StudentID , ParentName, PhoneNumber
from parent
join student using (studentID)
Order By StudentID;
```

	StudentID	ParentName	PhoneNumber
▶	100	Alen Wallace	1234567890
	101	Stone Levinson	1234567891
	102	Ben Scott	1234567892
	103	Max Martin	1234567893
	104	Elon Kapoor	1234567894
	105	Gim Hudson	1234567895
	106	Pia Porter	1234567896
	106	Amy Porter	1234567896
	107	Samantha Bernard	1234567897
	108	Louis Halpert	1234567898
	109	Murali Gonuguntla	1234567899
	109	Leelavathi Gonug...	1234567899


Query 7 : Number of courses each student is enrolled in

```
select student.StudentID, ifnull(No_Of_Courses,0) No_Of_Courses
from student
left join
(select student.StudentID, concat(LastName," ",FirstName) as Name, count(EnrollmentID)
No_Of_Courses
from student
join enrollment using(StudentID)
group by student.StudentID) as No_Of using(StudentID)
where No_Of_Courses >0
order by No_Of_Courses;
```

	StudentID	No_OF_Courses
▶	100	2
	101	2
	102	2
	109	2
	110	2
	104	3
	105	3
	106	3
	107	3
	108	3

Query 8 : Number of DS major students enrolled in CS courses

```
select student.StudentID,FirstName,LastName,CourseName from enrollment
join student on student.StudentID = enrollment.StudentID
join courses using(CourseID)
where enrollment.courseid like 'CS%' and student.major like 'DS%';
```



Result Grid				
		Filter Rows:		Export:  Wrap Cell Cor
	StudentID	FirstName	LastName	COurseName
▶	102	Michael	Scott	Networks
	106	Josh	Porter	DAA
	107	Andy	Bernard	CS ethics
	109	Praveen	Gonuguntla	OS
	110	Thulasi	Gabbita	OS

Query 9 : Number of CS major students enrolled in DS courses

```

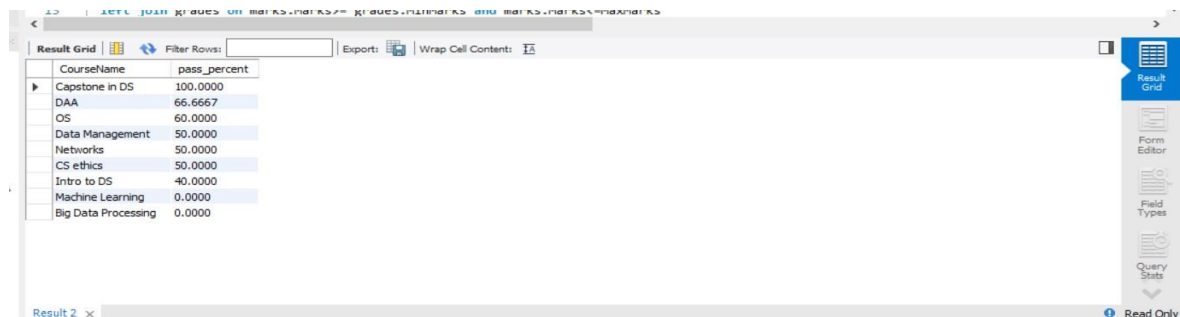
select student.StudentID,FirstName,LastName,CourseName from enrollment
join student on student.StudentID = enrollment.StudentID
join courses using(CourseID)
where enrollment.courseid like 'DA%' and student.major like 'CS%';

```

Result Grid				
		Filter Rows:		Export:  Wrap Cell Content: 
	StudentID	FirstName	LastName	COurseName
▶	100	David	Wallace	Intro to DS
	104	Kelly	Kapoor	Capstone in DS
	108	Jim	Halpert	Big Data Processing

Query 10 : Percentage of graded students who were passed for each course

```
select CourseName ,ifnull((passed/students)*100,0) as pass_percent from
(select CourseName , count(Grade) as students from
(select StudentID,CourseName, Marks, Grade from marks
left join grades on marks.Marks>= grades.MinMarks and marks.Marks<=MaxMarks
left join enrollment using(EnrollmentID)
left join student using(StudentID)
left join courses using(CourseID)) as graded
group by CourseName) as Total_Students
left join(
select CourseName , count(Grade) as passed
from (select StudentID,CourseName, Marks, Grade from marks
left join grades on marks.Marks>= grades.MinMarks and marks.Marks<=MaxMarks
left join enrollment using(EnrollmentID)
left join student using(StudentID)
left join courses using(CourseID)) as graded
where Grade != "F"
group by CourseName) as passed using(CourseName)
order by pass_percent desc;
```

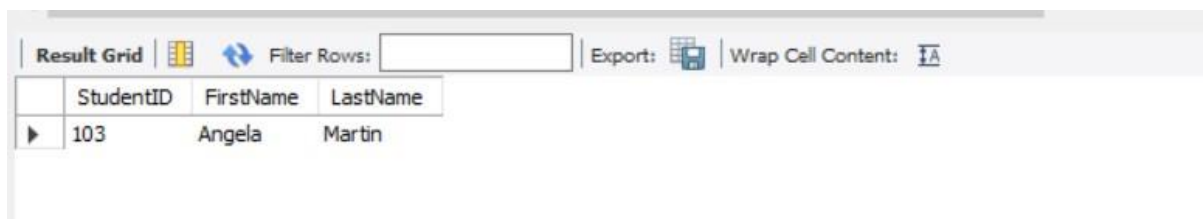


The screenshot shows a database query result grid with two columns: CourseName and pass_percent. The data is sorted in descending order of pass_percent. The courses and their corresponding pass percentages are as follows:

CourseName	pass_percent
Capstone in DS	100.0000
DAA	66.6667
OS	60.0000
Data Management	50.0000
Networks	50.0000
CS ethics	50.0000
Intro to DS	40.0000
Machine Learning	0.0000
Big Data Processing	0.0000

Query 11 : Students who are not registered for any course

```
select StudentID, FirstName, LastName from(  
select * from  
student  
left join enrollment using(StudentID)) as registered  
where EnrollmentID is null;
```



The screenshot shows a database query result grid. The grid has three columns: StudentID, FirstName, and LastName. There is one row of data with StudentID 103, FirstName Angela, and LastName Martin. The grid is part of a software interface with a toolbar at the top containing options like 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'.

StudentID	FirstName	LastName
103	Angela	Martin

Scope of the Project:

The school's daily operations can be managed using the above School Management System. The project is currently dealing with storing the students, parents and lecturers' data, total courses offered in school, courses offered by each lecturer, details of students enrollment to classes, exams and marks for each exam etc. The information can easily be shared within the school management, and required data can be accessed effortlessly. It enables users to store all the school information electronically. With the School Management System, schools will no longer have to deal with heaps of paper records and finding student information can now take a few seconds.

Problems Faced:

1. Understanding the domain
2. Adapting to challenges, modifications and new features from initial analysis to final stage of db schema
3. Searching for mock data
4. Finding right data with foreign key values and other specific requirements
5. Coming up with valid analysis points to write queries and store them as views
6. Working with low amount of data while ensuring wide range of features
7. Unable to work simultaneously
8. Technical limitations due to different operating systems and different version of MySQL workbench
9. Limited time

Lessons Learned:

1. There could be minor deviations or challenges while creating a database. So, proper time and efforts should be estimated keeping them in mind
2. We understood the school system and how it works
3. Tools used should be installed after team deciding on the version

Future Scope:

Below features can be added to our database for further implementation and upgradation.

1. Fee structure and details for each grade/course.
2. Storing the school buildings or classrooms information.
3. Storing the students and teachers' identity pictures for future security or emergency purposes.
4. Recording fee payment details for each student.
5. Students' attendance details.
6. Salary structure of lecturers.

Requirement Table:

Query Number	Req. A	Req. B	Req. C	Req. D	Req. E
Query 1	✓	✓		✓	
Query 2	✓	✓	✓	✓	✓
Query 3	✓	✓	✓	✓	✓
Query 4	✓			✓	
Query 5	✓	✓	✓	✓	✓
Query 6	✓	✓		✓	
Query 7	✓	✓	✓	✓	✓
Query 8	✓	✓		✓	
Query 9	✓	✓		✓	
Query 10	✓	✓	✓	✓	✓
Query 11	✓	✓		✓	✓