**Contents**

[1. INTRODUCTION 1](#_Toc135679461)

[1.1. Background 1](#_Toc135679462)

[1.2. How a School DBMS can be lucrative 1](#_Toc135679463)

[1.3. The Ideal Path in Creating a School Management System Database? 1](#_Toc135679464)

[2. OBJECTIVES OF THE PROJECT 2](#_Toc135679465)

[2.1. General Objective 2](#_Toc135679466)

[2.2. Specific Objectives 2](#_Toc135679467)

[3. PREREQUISITES OR REQUIREMENTS 2](#_Toc135679468)

[3.1. Hardware Requirements 2](#_Toc135679469)

[3.2. Software Requirements 3](#_Toc135679470)

[4. PROJECT EXPLANATION 4](#_Toc135679471)

[4.1. Scope 5](#_Toc135679472)

[4.2. Existing System 5](#_Toc135679473)

[4.3. Proposed System 6](#_Toc135679474)

[5. CONCEPTUAL DESIGN PHASE 7](#_Toc135679475)

[5.1. Determining Entities & Attributes 7](#_Toc135679476)

[5.2. Determining Relationships and Cardinality 10](#_Toc135679477)

[6. LOGICAL DESIGN PHASE 12](#_Toc135679478)

[6.1. Relational Schemas and Mapping 12](#_Toc135679479)

[6.2. Normalization 15](#_Toc135679480)

[7. CONCLUSION 17](#_Toc135679481)

# INTRODUCTION

## Background

DBMS or Database Management System is a software system that helps in the management of electronic data. In the context of a school, DBMS plays a pivotal role in managing student data, administrative tasks, financial records, attendance records, and more. Implementing DBMS in school management improves the efficiency of daily tasks, reduces errors, and streamlines the management process. Hence In order facilitate administrative tasks, manage student and staff information more effectively, and promote greater transparency and communication among all stakeholders in the educational community, we have made an effort to develop a school management database system as part of our project.

## How a School DBMS can be lucrative

One of the significant benefits of DBMS in school management is operational efficiency. DBMS allows schools to manage all data electronically, which is faster and more efficient than manual data entry. For instance, using DBMS for attendance tracking can help automate the process, reducing the time spent on manual data entry, and providing accurate information. Another advantage of DBMS in school management is improved communication. DBMS allows multiple users to access the same database simultaneously, even though millions won’t need to access a school management database system it means that all stakeholders or staff members have real-time access to relevant data. This facilitates communication between teachers, students, and administrators and allows them to work seamlessly together towards the development of the school.

## The Ideal Path in Creating a School Management System Database?

Implementing DBMS in school management requires careful planning and execution. First, the school needs to identify the software that best fits its needs, considering factors such as cost, features, and compatibility. If the school already has a system in place, the DBMS should be integrated with the existing system to ensure data consistency. Choosing a reliable database solution provider is essential to ensure successful implementation. The implementation process should be carefully planned and executed in stages to ensure that all stakeholders are well-informed and trained to use the system. The school management should also establish backup and recovery procedures to ensure data protection in case of system failure.

In a nutshell, DBMS plays a critical role in school management and can improve the efficiency and effectiveness of the school's daily functions. With DBMS in place, schools can streamline their operations, improve communication, and enhance student performance. From here onward we’ll go on to discuss the requirements, elaboration of the contents of the projects as well as common concepts raised when dealing with database system such as Entity relationship diagrams, functional dependencies and normalization.

# OBJECTIVES OF THE PROJECT

## General Objective

The overall purpose of utilizing a tool like DBMS has a myriad of applications among which we strive to achieve include:

* Providing a streamlined methodology of storing and manipulating school data.
* Eliminate risks of unauthorized access and sustain the integrity of data.
* Enable quick access of information by ruling out obsolete systems.
* Ensuring safety of data by providing a way to back-up the database

## Specific Objectives

Within the context of a school management system database there are particular pretensions that are sought in order to achieve previously outlined general objectives. some of the modernized operations that users can perform as part of the school DBMS which ascertain the objectives include:

* Seamless entry of datum concerning students or teachers
* Monitor attendance of both students and teachers with ease
* Allow user to manipulate data simultaneously.

# PREREQUISITES OR REQUIREMENTS

## Hardware Requirements

* For educational institutions to efficiently manage and store information related to students, teachers, staff, and other administrative functions. The hardware requirements of a school management database system depend on the size of the institution and the number of users who will access it. For large institutions like schools with multiple branches a robust server and network infrastructure are critical to ensure high availability, reliability, and security of the system.
* In this article, we will discuss the key hardware components necessary to build a regular scaled school management database system that can effectively handle the influx of data and user traffic of one branch. In our case we will not go as far as an over-complicated and exhaustive format, hence in context to a smaller scale institution like an average School in a region would have hardware requirements as follows:

Minimum Hardware Requirements:

* Processor: Dual-core processor
* RAM: 2GB or higher
* Storage: 100GB or higher
* Network Card: 10/100 Ethernet
* Display: 1024x768 resolution
* Stable internet connection (either WIFI or Ethernet)

Recommended Hardware Requirements:

* Processor: Quad-core processor or higher
* RAM: 8GB or higher
* Storage: 500GB or higher
* Network Card: 1 Gbps Ethernet
* Display: 1920x1080 resolution
* Stable internet connection (either WIFI or Ethernet)

## Software Requirements

* To ensure the smooth functioning of the school management database system, there are certain software requirements that need to be considered. Hence, we will explore the essential software requirements for a school management database system, their benefits, and the different options available.
* A school management database system can use a combination of different software applications depending on the institutes’ needs and preferences. Keep in mind that the software we choose will depend on our specific needs, budget, and technical expertise. More likely software requirements for a school management system database can be:
* Operating System: Windows 7 and later, Mac OS X or Linux distros (e.g., Ubuntu)
* Database Management System: MySQL, Microsoft SQL Server, Oracle or Postgres
* Programming Languages: Java, PHP, Python, C or C++
* As the above is the software required to run a School management system database below are commonly used software for creating an advanced school management database system (note that some of these are optional and not exclusively required):
* **Database Management System (DBMS):** This is the most essential software required to create a database system that can store, process, and manage data accurately. Examples of popular DBMS include Oracle, MySQL, and Microsoft SQL Server (which we are currently using in the Laboratory).
* **Development Tools**: You'll need development tools to create the user interface and programming logic for your database system. Some popular development tools for building school management systems include Java, .NET Framework, and Python.
* **Web Development Tools**: If you plan to build a web-based school management system, you'll need web development tools to create the frontend and backend of your system. Some web development tools that are commonly used include HTML, CSS, JavaScript, and PHP.
* **Integrated Development Environment (IDE)**: An IDE is an essential tool for creating, testing, and debugging software code. IDEs like Eclipse, Visual Studio, and IntelliJ IDEA can help you create complex database systems with ease.
* **Reporting Tools**: Reporting tools help you generate visual reports that enable you to analyze data and make informed decisions. Some popular reporting tools are Crystal Reports, Jasper Reports, and Microsoft SQL Server Reporting Services.
* **Document Management Software**: You may need document management software to manage important school documents such as admission forms, student records, and academic transcripts. Examples of document management software include Microsoft SharePoint, Alfresco, and OpenKM.

# PROJECT EXPLANATION

Prior to developing the aforementioned project i.e., the simple school management system database, it is of utmost importance to examine and figure out the common inquires that will arise when dealing with these subjects. Hence, in this section we will determine and answer all related Inquiries.

* **Which of the Software and Hardware Options Are Suitable for Our Project?**

Among the plethora of database management systems one can find it a hard task to select an apt software that can satisfy their needs. In our case its best to utilize tools we’re most knowledgeable and acquainted to. Furthermore, since we’re creating a basic database system, the software we use won’t require exclusive features that are available to specific software so it will be functional as long as it fulfills the minimum requirement. Ergo, we will be using:

* **DBMS**: SQL Server Management Studio 19
* **Operating System**: Windows 10
* **Interface Programming Language**: Visual C++
* **Integrated Development Environment (IDE)**: Visual Studio 2022
* **Framework**: .NET framework

Consequently, a simple database system like ours’ wouldn’t require high-performance set-ups so most commonly found hardware would be sufficient as long as it’s not below the minimum requirements mentioned above.

* **What form of data will the enterprise handle?**

As one might expect schools mostly deal with numeric and non-numeric data like student names, teacher names, salary (in birr) etc... which can be stored in two dimensional tables. One difference from conventional school data is the way the students are marked. Most schools outside of Ethiopia use grades (A, B or F) as a scale to mark students which are later used to calculate GPA. However, this is not the case in Ethiopia even though grades and GPAs are used in higher institutions like colleges and universities, in most Ethiopian schools, marks in the 100th scale is used for each subject which are later used to calculate the Average as an alternative to the GPA. For such entangled data a relational database where multiple tables that accommodate these records and create links between them is necessary. This will eliminate redundancy as well as improve the efficiency of data handling technique.

* 1. ScopeoftheProject

The development of the school management system database reserves a vast scope of features depending on the demands the client institute. Generally, this project covers the faculty of teachers to manage their students’ assessment by keeping track of and setting exam results, assignments and projects, and Average marks; analyses the registration of new students and integrates new data into the entire database; monitor employment of new teachers as well as manage salary raise or deduction. Additionally, it keeps track of class rooms available and which student belongs to which class room; the attendance status of each student; and the subject each student takes. Finally, it can enable students to follow up their assessment by showing them their exam results, project and assignment results as well as their final average mark. The functionality of such a system can maximize the efficiency of the staff by modernizing obsolete techniques.

* 1. ExistingSystem

A majority schools in Ethiopia handle their data either in the old-fashioned manual approach or the somehow better but still obsolete file-based approach while a miniscule minority utilize RDBMS. Former mentioned methods have numerous issues that emerge along with them. Here we will discuss how most schools in Ethiopia approach data handling along with the side effects that are inevitable when applying these approaches.

* **Manual Approach**
* Records are stored on paper usually in alphabetical order.
* A collection of these paper records labeled under some category (class room or grade level) is collected and stored in hard cover cases.
* The collection is yet again stored on a cabinet shelf under a more general category like school year or batch year.
* Staff members who want access to a specific file must first choose the correct shelf, then the correct case and so on…
* Drawbacks of the Manual Approach
* Tiresome work just to access a certain record.
* Highly susceptible to data loss (e.g., in case of fire all data will be lost forever)
* Staff members can misplace the records of students (e.g., placing the data of a grade 6th student among the records of 12th grade students)
* Very limited storage space
* Can make multiple duplicates of the same file if staff members haven’t communicated.
* Not secure as a door is the only constraint that keeping attackers from accessing the data.
* **File Based Approach**
* Records are stored digitally most likely on excel spreadsheets.
* Each staff members creates their own separate spreadsheet with respect to their application.
* Each application has its own spreadsheet (list of students in Class A and students of batch 2022 will each have their own tables)
* Drawbacks of the File Based Approach
* Restricted sharing of data among staff members (since each handles their own)
* Redundant data
* Distinctive format of different data (e.g., date might have DD/MM/YYYY format in one file and MM/DD/YYYY format in another)
* Deletion error (a deleted data in one file might not be deleted in another)
* Insertion and update errors (new or altered data in one file might not be updated on the other).
* Waste of storage (since there are multiple files for each program and significant redundancy).
  1. ProposedSystem

The system that we offer, in addition to eliminating all the drawback of the previous approaches. It will have modern features such as:

* Can be connected to a centralized online server.
* Multiple users can simultaneously alter the data and will automatically update the central database accordingly.
* Require authentication to provide access and alter the database
* Has a C++ and .NET framework-based windows application graphical user interface (GUI) which allows a user-friendly login and registration authentication.
* A simple C++ application that allows CRUD (create, read, update and delete) operations from the terminal.

1. CONCEPTUALDESIGNPHASE
   1. DeterminingEntitiesandAttributes

Albeit it may vary according to the institutions there are undeniable or most common entities along with their attributes that are always used when creating a school management system database. And they are:

* **School**
* **Teacher**
* **Student**
* **ClassRoom**
* **Subject**
* **Assessment**
* **Attendance**

1. **School**

|  |  |  |  |
| --- | --- | --- | --- |
| Field/Attribute | Data Type | Constraints | Description |
| SchoolID | Int | PRIMARY KEY | Unique ID for the School |
| SchoolName | Varchar (50) | NOT NULL | Name of the School |
| Address | Varchar (100) | NOT NULL | Address of the School |
| Phone | Varchar (20) | NOT NULL, UNQIUE | Phone number of the school |

1. **Teacher**

|  |  |  |  |
| --- | --- | --- | --- |
| Field/Attribute | Data Type | Constraints | Description |
| TeacherID | Int | PRIMARY KEY | Unique ID to identify the teacher |
| FirstName | Varchar (250) | NOT NULL | Teachers’ first name |
| LastName | Varchar (50) | NOT NULL | Teachers’ last name |
| Email | Char | NOT NULL | Multi-valued attribute, teacher’s Email |
| Phone | Varchar (20) | NOT NULL | Multi-valued attribute, phone number of the teacher |
| DOB | Date | NOT NULL | Date of birth of the teacher |
| Gender | Char | NOT NULL | Gender of the teacher |

1. **Student**

|  |  |  |  |
| --- | --- | --- | --- |
| Field/Attribute | Data Type | Constraints | Description |
| StudentID | Int | PRIMARY KEY | Unique ID that identifies the student |
| FirstName | Varchar (50) | NOT NULL | First name of the student |
| LastName | Varchar (50) | NOT NULL | Last name of the student |
| Gender | Char | NOT NULL | Sex of the student |
| Phone | Varchar (20) | NOT NULL | Multi-valued attribute, phone number of the student |
| DOB | Date | NOT NULL | Date of birth of the student |
| Email | Varchar (100) | NOT NULL | Multivalued attribute, student’s Email |

1. **ClassRoom**

|  |  |  |  |
| --- | --- | --- | --- |
| Field/Attributes | Data Type | Constraints | Description |
| ClassRoomID | Int | PRIMARY KEY | Unique ID that represents the class |
| ClassRoomName | Varchar (3) | NOT NULL | Class and Section name (e.g., 12A) |
| ClassYear | Date | NOT NULL | Batch year of the class |
| Capacity | Int | NOT NULL | Number of students the classroom the has |

1. **Subject**

|  |  |  |  |
| --- | --- | --- | --- |
| Field/Attribute | Data Type | constraints | Description |
| SubjectID | Int | PRIMARY KEY | Unique ID that identifies the subject |
| SubjectName | Varchar (50) | NOT NULL | Name of the subject |

1. **Assessment**

|  |  |  |  |
| --- | --- | --- | --- |
| Field/Attribute | Data Type | Constraints | Description |
| AssessmentID | Int | PRIMARY KEY | Unique ID that distinguishes the assessment |
| AssessmentType | Varchar (50) | NOT NULL | Name of the Assessment Type (e.g., Quiz, Project etc.…) |
| AssessmentDate | Date | NOT NULL | Date of the Assessment |
| Score | Float | - | Score of the student for a certain assessment |
| Total | Float | NOT NULL | Derived attribute which is the sum of each assessment types of a subject a student takes |
| Average | Float | NOT NULL | Also, a Derived Attribute 100-point scale Average of all subjects’ Total of a student |

1. **Attendance**

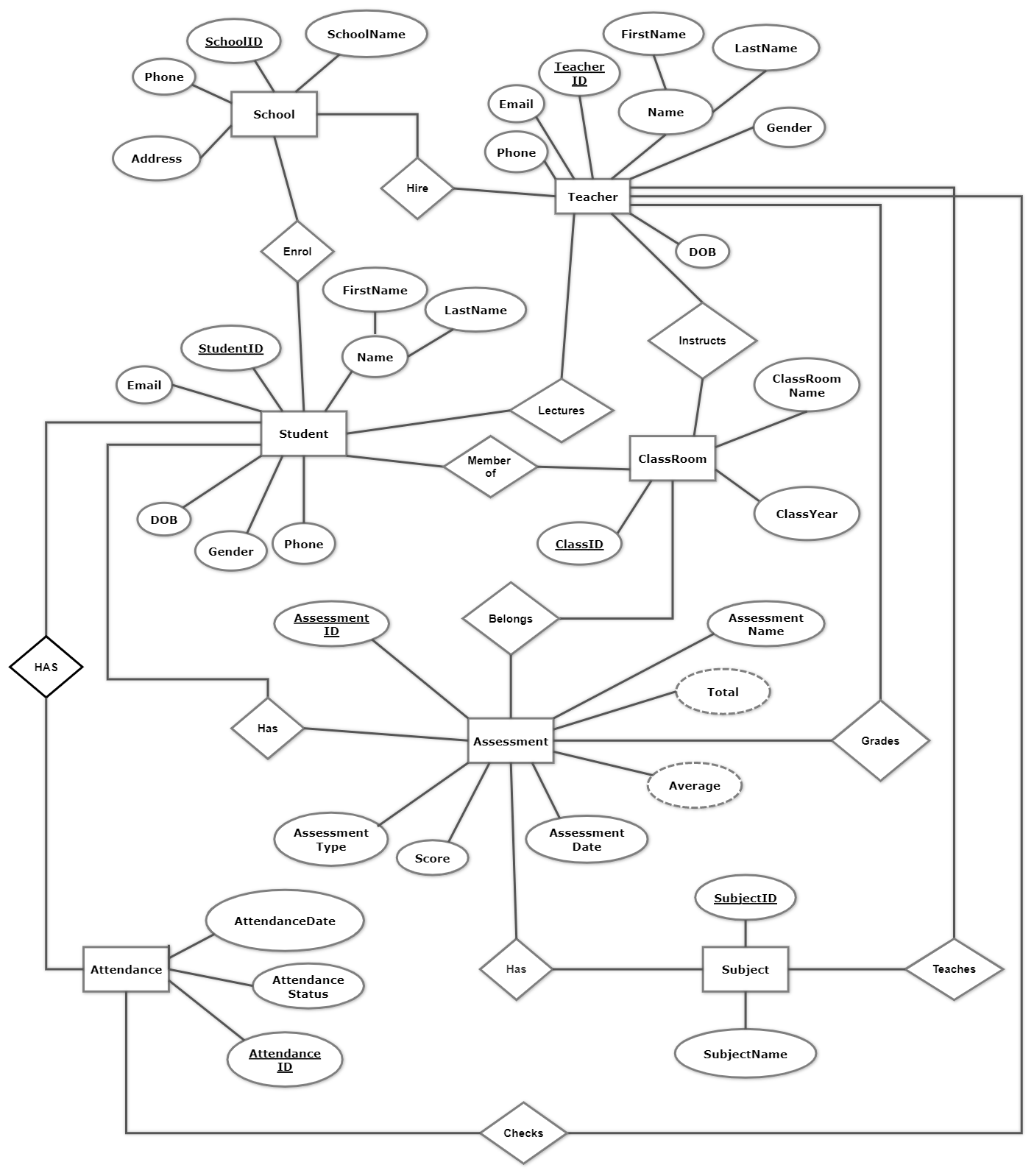
|  |  |  |  |
| --- | --- | --- | --- |
| Field/Attribute | Data Type | Constraints | Description |
| AttendanceID | int | PRIMARY KEY | Unique ID to identify the Attendance |
| AttendanceStatus | Varchar (10) | NOT NULL | Status of Attendance either Present or Absent |
| AttendanceDate | Date | NOT NULL | Date of the Attendance record |

## Determining Relationships and Cardinality

Before moving on to creating the ER diagram for the entities and attributes that we listed above, we must consider what links those entities amongst each other according to our system ‘s requirements. As well as what form of link they have AKA their cardinality i.e., one to one, one to many, many to one or many to many. Hence, below is the list of relationships our entities have with each other.

* A school can hire multiple teachers yet a teacher can work for one school, hence cardinality ratio is 1: N.
* A school can enroll multiple students however a student can be enrolled in only one school. Therefore, the cardinality type is also 1: N.
* A teacher can instruct multiple classes, and a class can be instructed by many teachers so the cardinality is M: N.
* A teacher can lecture many students, and a student can be lectured by many teachers hence the cardinality is M: N as well.
* A teacher can grade many assessments, yet one instance of an assessment can be graded by one teacher, as a result the cardinality is 1: N.
* A teacher can teach only one subject. However, a subject can be taught by many teachers. Therefore, the cardinality is N: 1.
* A Teacher can check multiple attendance records, but a single attendance record can only be checked by one teacher. Hence the cardinality is 1: N.
* A student can be a member of only one class room, yet a classroom can have multiple students. Hence the cardinality is N: 1.
* A student can have multiple assessments but an assessment can be given to one student, so the cardinality is 1: N.
* A student can have multiple attendance records, but a single attendance record can be given to one student. So, the cardinality is 1: N.
* A class room can have multiple assessments and an assessment can belong to multiple classrooms. Therefore, the cardinality is M: N.
* A subject can have multiple assessments but a single assessment can be given to only one subjects. Therefore, the cardinality is 1: N.
* Now that we have identified the entities and attributes, and assigned the relationship among them we can create the visual representation of this design using an ER diagram. After that we can proceed to the logical design phase.

**ER DIAGRAM**



**N**

**1**

**1**

**N**

**N**

**M**

**N**

**M**

**1**

**N**

**M**

**N**

**1**

**N**

**N**

**1**

**N**

**1**

**1**

**N**

**1**

**N**

**N**

**1**

# LOGICAL DESIGN PHASE

## Relational Schemas and Mapping

In this phase we will convert the ER diagram to a Relational Table. And to achieve that there are necessary steps to follow. The first rule is that the name of each table is the name of the entity. Second, we need to map columns to the attributes we listed in the ER diagram. The third and final step is to map the relationships using primary key of one table as the foreign key for another, in this step the cardinality is very relevant as it tells us where the foreign keys will be placed. Thus, we will now move on to creating the relational tables and mappings for each entity.

1. **Step One and Two**

* **Creating relational tables and mapping columns to attributes**

**School**

|  |  |  |  |
| --- | --- | --- | --- |
| SchoolID | SchoolName | Address | Phone |

**Teacher**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TeacherID | FirstName | LastName | Email | Phone | DOB | Gender |

**Student**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| StudentID | FirstName | LastName | Email | Phone | DOB | Gender |

**ClassRoom**

|  |  |  |  |
| --- | --- | --- | --- |
| ClassRoomID | ClassName | ClassYear | Capacity |

**Subject**

|  |  |
| --- | --- |
| SubjectID | SubjectName |

**Assessment**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AssessmentID | AssessmentType | AssessmentDate | Score | Total | Average |

**Attendance**

|  |  |  |
| --- | --- | --- |
| AttendanceID | AttendanceStatus | AttendanceDate |

1. **Step Three**

* **Mapping relationships using foreign keys and junction tables.**



## Normalization

Normalization is a design technique, that is used to organize tables in a way that minimizes redundancy and dependency of data. It splits up larger tables into smaller tables divisions and connect them using relationships. There multiple levels of normalization called normal forms (1NF, 2NF, .... all the way to 5NF) where each level has a requirement to fulfill to be cleared of that level. From here onward we will discuss the first three levels of normalization first through third normal form.

1. **First Normal Form (1NF)**

For a table to a fulfill the first form first, no row can have multivalued attributes that is every attribute must be atomic and second, each record must be unique. To demonstrate this, we will use the Teacher table with some sample data.

**UNNORMALIZED TABLE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TeacherID | FirstName | LastName | Email | Phone | DOB | Gender |
| 100 | Yosef | Alemu | Yosef.alemu@gmail.com | +251901234567, +251912345678 | 7/03/1982 | M |
| 101 | Mekonen | Siraj | Mekonen.siraj@gmail.com, MekonenSJ@yahoo.com | +251923456789 | 21/07/1990 | M |
| 102 | Biruk | Mesfin | Biruk.mesfin@gmail.com | +251934567890 | 17/10/1978 | M |

**1NF NORMALIZED TABLE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TeacherID | FirstName | LastName | Email | Phone | DOB | Gender |
| 100 | Yosef | Alemu | Yosef.alemu@gmail.com | +251901234567 | 7/03/1982 | M |
| 100 | Yosef | Alemu | Yosef.alemu@gmail.com | +251912345678 | 7/03/1982 | M |
| 101 | Mekonen | Siraj | Mekonen.siraj@gmail.com | +251923456789 | 21/07/1990 | M |
| 101 | Mekonen | Siraj | MekonenSJ@yahoo.com | +251923456789 | 21/07/1990 | M |
| 102 | Biruk | Mesfin | Biruk.mesfin@gmail.com | +251934567890 | 17/10/1978 | M |

1. **Second Normal Form (2NF)**

For a table to clear the second normal form requirements it must first be in first normal form and second it every attribute and non-primary key must fulfill the concept of full functional dependency in other terms no partial dependencies are allowed on the primary key. Full functional dependency implies that for a primary key, all non-key attributes must be fully dependent on the primary key. If we added a table called Teacher Assessment to represent assessments given by each teacher, the following problem would arise.

**UNNORMALIZED TABLE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TeacherID | AssessmentID | FirstName | LastName | AssessmentType | AssessmentDate |
| 100 | 01 | Yosef | Alemu | Quiz | 12/04/2014 |
| 102 | 02 | Biruk | Mesfin | Project | 25/03/2014 |

* **TeacherID determines FirstName and LastName but AssessmentType and AssessmentDate are not dependent on TeacherID. Same goes for AssessmentID it cannot determine FirstName and LastName hence it doesn’t satisfy full functional dependency.**

**SECOND FORM NORMALIZED TABLE (2NF)**

**By splitting the table into two we can normalize this table.**

|  |  |  |
| --- | --- | --- |
| TeacherID | FirstName | LastName |
| 100 | Yosef | Alemu |
| 102 | Biruk | Mesfin |

|  |  |  |
| --- | --- | --- |
| AssessmentID | AssessmentType | AssessmentDate |
| 01 | Quiz | 12/04/2014 |
| 02 | Project | 25/03/2014 |

1. **Third Normal Form (3NF)**

And finally, for a table to pass the third normal form we must eliminate any existing transitive functional dependencies. Transitive functional dependency implies if a field A determines B and B determines C then A determines C. In simple terms, every non-key attribute in the table must depend on the key, the whole key and nothing but the key. To demonstrate let’s look at the table below:

# CONCLUSION