MODULE I: Introduction:

What is Database Management System

What is a Database?

A database is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populated with data for a certain task.

What is DBMS?

Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.
DBMS allows users to create their own databases as per their requirement. The term "DBMS" includes the user of the database and other application programs. It provides an interface between the data and the software application.

Advantage of DBMS over File Processing System

File System: A File Management system is a DBMS that allows access to single files or tables at a time. In a File System, data is directly stored in set of files. It contains flat files that have no relation to other files (when only one table is stored in single file, then this file is known as flat file).
DBMS: A Database Management System (DBMS) is a application software that allows users to efficiently define, create, maintain and share databases. Defining a database involves specifying the data types, structures and constraints of the data to be stored in the database. Creating a database involves storing the data on some storage medium that is controlled by DBMS. Maintaining a database involves updating the database whenever required to evolve and reflect changes in the miniworld and also generating reports for each change. Sharing a database involves allowing multiple users to access the database. DBMS also serves as an interface between the database and end users or application programs. It provides control access to the data and ensures that data is consistent and correct by defining rules on them.
An application program accesses the database by sending queries or requests for data to the DBMS. A query causes some data to be retrieved from database.

Advantages of DBMS over File system:

- **Data redundancy and inconsistency:** Redundancy is the concept of repetition of data i.e. each data may have more than a single copy. The file system cannot control the redundancy of data as each user defines and maintains the needed files for a specific application to run. There may be a possibility that two users are maintaining the data of the same file for different applications. Hence changes made by one user do not reflect in files used by second users, which leads to inconsistency of data. Whereas DBMS controls redundancy by maintaining a single repository of data that is defined once and is accessed by many users. As there is no or less redundancy, data remains consistent.
- **Data sharing:** The file system does not allow sharing of data or sharing is too complex. Whereas in DBMS, data can be shared easily due to a centralized system.
- **Data concurrency:** Concurrent access to data means more than one user is accessing the same data at the same time. Anomalies occur when changes made by one user get lost because of changes made by another user. The file system does not provide any

procedure to stop anomalies. Whereas DBMS provides a locking system to stop anomalies to occur.

- **Data searching:** For every search operation performed on the file system, a different application program has to be written. While DBMS provides inbuilt searching operations. The user only has to write a small query to retrieve data from the database.
- **Data integrity:** There may be cases when some constraints need to be applied to the data before inserting it into the database. The file system does not provide any procedure to check these constraints automatically. Whereas DBMS maintains data integrity by enforcing user-defined constraints on data by itself.
- **System crashing:** In some cases, systems might have crashed due to various reasons. It is a bane in the case of file systems because once the system crashes, there will be no recovery of the data that's been lost. A DBMS will have the recovery manager which retrieves the data making it another advantage over file systems.
- **Data security:** A file system provides a password mechanism to protect the database but how long can the password be protected? No one can guarantee that. This doesn't happen in the case of DBMS. DBMS has specialized features that help provide shielding to its data.
- **Backup:** It creates a backup subsystem to restore the data if required.
- **Interfaces**: It provides different multiple user interfaces like graphical user interface and application program interface.
- **Easy Maintenance**: It is easily maintainable due to its centralized nature.

DBMS is continuously evolving from time to time. It is a power tool for data storage and protection. In the coming years, we will get to witness an AI-based DBMS to retrieve databases of ancient eras.

Introduction and applications of DBMS

Introduction of DBMS

Important Terminology

Database: Database is a collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc. For Example, university database organizes the data about students, faculty, and admin staff etc. which helps in efficient retrieval, insertion and deletion of data from it.

DDL is short name of Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

- CREATE: to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- ALTER: alters the structure of the existing database
- DROP: delete objects from the database
- TRUNCATE: remove all records from a table, including all spaces allocated for the records are removed
- COMMENT: add comments to the data dictionary
- RENAME: rename an object

DML is short name of Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

- SELECT: retrieve data from a database
- INSERT: insert data into a table
- UPDATE: updates existing data within a table
- DELETE: Delete all records from a database table
- MERGE: UPSERT operation (insert or update)
- CALL: call a PL/SQL or Java subprogram
- EXPLAIN PLAN: interpretation of the data access path
- LOCK TABLE: concurrency Control

Database Management System: The software which is used to manage database is called Database Management System (DBMS). For Example, MySQL, Oracle etc. are popular commercial DBMS used in different applications. DBMS allows users the following tasks:
Data Definition: It helps in creation, modification and removal of definitions that define the organization of data in database.
Data Updation: It helps in insertion, modification and deletion of the actual data in the database.
Data Retrieval: It helps in retrieval of data from the database which can be used by applications for various purposes.
User Administration: It helps in registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control and recovering information corrupted by unexpected failure.

Paradigm Shift from File System to DBMS

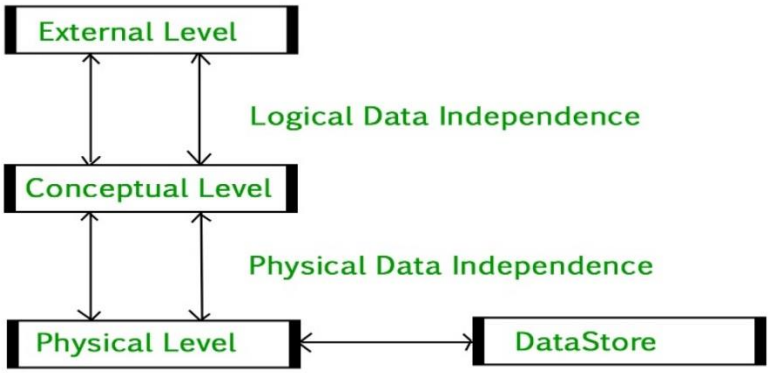
File System manages data using files in hard disk. Users are allowed to create, delete, and update the files according to their requirement. Let us consider the example of file based University Management System. Data of students is available to their respective Departments, Academics Section, Result Section, Accounts Section, Hostel Office etc. Some of the data is common for all sections like Roll No, Name, Father Name, Address and Phone number of students but some data is available to a particular section only like Hostel allotment number which is a part of hostel office. Let us discuss the issues with this system:

- **Redundancy of data:** Data is said to be redundant if same data is copied at many places. If a student wants to change Phone number, he has to get it updated at various sections. Similarly, old records must be deleted from all sections representing that student.
- **Inconsistency of Data:** Data is said to be inconsistent if multiple copies of same data does not match with each other. If Phone number is different in Accounts Section and Academics Section, it will be inconsistent. Inconsistency may be because of typing errors or not updating all copies of same data.
- **Difficult Data Access:** A user should know the exact location of file to access data, so the process is very cumbersome and tedious. If user wants to search student hostel allotment number of a student from 10000 unsorted students' records, how difficult it can be.
- **Unauthorized Access:** File System may lead to unauthorized access to data. If a student gets access to file having his marks, he can change it in unauthorized way.
- **No Concurrent Access:** The access of same data by multiple users at same time is known as concurrency. File system does not allow concurrency as data can be accessed by only one user at a time.
- **No Backup and Recovery:** File system does not incorporate any backup and recovery of data if a file is lost or corrupted.

These are the main reasons which made a shift from file system to DBMS.

DBMS 3-tier Architecture

DBMS 3-tier architecture divides the complete system into three inter-related but independent modules as shown below:



1. **Physical Level:** At the physical level, the information about the location of database objects in the data store is kept. Various users of DBMS are unaware of the locations of these objects.In simple terms,physical level of a database describes how the data is being stored in secondary storage devices like disks and tapes and also gives insights on additional storage details.
2. **Conceptual Level:** At conceptual level, data is represented in the form of various database tables. For Example, STUDENT database may contain STUDENT and COURSE tables which will be visible to users but users are unaware of their storage.Also referred as logical schema,it describes what kind of data is to be stored in the database.
3. **External Level:** An external level specifies a view of the data in terms of conceptual level tables. Each external level view is used to cater to the needs of a particular category of users. For Example, FACULTY of a university is interested in looking course details of students, STUDENTS are interested in looking at all details related to academics, accounts, courses and hostel details as well. So, different views can be generated for different users. The main focus of external level is data abstraction.

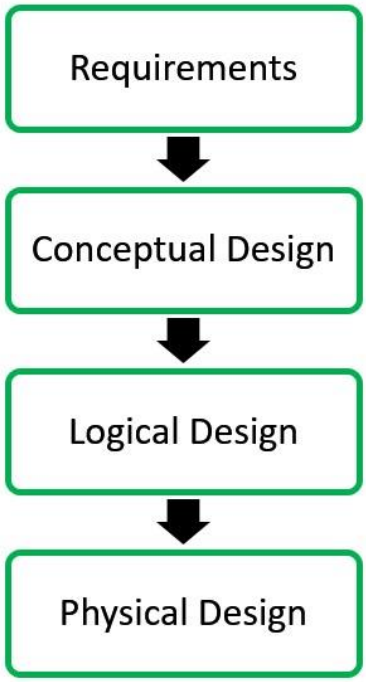
Data Independence

Data independence means a change of data at one level should not affect another level. Two types of data independence are present in this architecture:

1. **Physical Data Independence:** Any change in the physical location of tables and indexes should not affect the conceptual level or external view of data. This data independence is easy to achieve and implemented by most of the DBMS.
2. **Conceptual Data Independence:** The data at conceptual level schema and external level schema must be independent. This means a change in conceptual schema should not affect external schema. e.g.; Adding or deleting attributes of a table should not affect the user's view of the table. But this type of independence is difficult to achieve as compared to physical data independence because the changes in conceptual schema are reflected in the user's view.

Phases of database design

Database designing for a real-world application starts from capturing the requirements to physical implementation using DBMS software which consists of following steps shown below:



Conceptual Design: The requirements of database are captured using high level conceptual data model. For Example, the ER model is used for the conceptual design of the database.

Logical Design: Logical Design represents data in the form of relational model. ER diagram produced in the conceptual design phase is used to convert the data into the Relational Model.
Physical Design: In physical design, data in relational model is implemented using commercial DBMS like Oracle, DB2.

Advantages of DBMS

DBMS helps in efficient organization of data in database which has following advantages over typical file system:

- **Minimized redundancy and data inconsistency:** Data is normalized in DBMS to minimize the redundancy which helps in keeping data consistent. For Example, student information can be kept at one place in DBMS and accessed by different users.This minimized redundancy is due to primary key and foreign keys
- **Simplified Data Access:** A user need only name of the relation not exact location to access data, so the process is very simple.
- **Multiple data views:** Different views of same data can be created to cater the needs of different users. For Example, faculty salary information can be hidden from student view of data but shown in admin view.
- **Data Security:** Only authorized users are allowed to access the data in DBMS. Also, data can be encrypted by DBMS which makes it secure.
- **Concurrent access to data:** Data can be accessed concurrently by different users at same time in DBMS.
- **Backup and Recovery mechanism:** DBMS backup and recovery mechanism helps to avoid data loss and data inconsistency in case of catastrophic failures.

Application of DBMS

There are different fields where a database management system is utilized. Following are a few applications which utilize the information base administration framework –

1. **Railway Reservation System –**
   In the rail route reservation framework, the information base is needed to store the record or information of ticket appointments, status about train's appearance, and flight. Additionally, if trains get late, individuals become acquainted with it through the information base update.

2. **Library Management System –**
   There are lots of books in the library so; it is difficult to store the record of the relative multitude of books in a register or duplicate. Along these lines, the data set administration framework (DBMS) is utilized to keep up all the data identified with the name of the book, issue date, accessibility of the book, and its writer.

3. **Banking –**
   Database the executive's framework is utilized to store the exchange data of the client in the information base.

4. **Education Sector –**
   Presently, assessments are led online by numerous schools and colleges. They deal with all assessment information through the data set administration framework (DBMS). In spite of that understudy's enlistments subtleties, grades, courses, expense, participation, results, and so forth all the data is put away in the information base.

5. **Credit card exchanges –**
   The database Management framework is utilized for buying on charge cards and age of month to month proclamations.

6. **Social Media Sites –**
   We all utilization of online media sites to associate with companions and to impart our perspectives to the world. Every day, many people group pursue these online media

accounts like Pinterest, Facebook, Twitter, and Google in addition to. By the utilization of the data set administration framework, all the data of clients are put away in the information base and, we become ready to interface with others.

7. **Broadcast communications –**
   Without DBMS any media transmission organization can't think. The Database the executive's framework is fundamental for these organizations to store the call subtleties and month to month postpaid bills in the information base.

8. **Account –**
   The information base administration framework is utilized for putting away data about deals, holding and acquisition of monetary instruments, for example, stocks and bonds in a data set.

9. **Online Shopping –**
   These days, web-based shopping has become a major pattern. Nobody needs to visit the shop and burn through their time. Everybody needs to shop through web based shopping sites, (for example, Amazon, Flipkart, Snapdeal) from home. So all the items are sold and added uniquely with the assistance of the information base administration framework (DBMS). Receipt charges, installments, buy data these are finished with the assistance of DBMS.

10. **Human Resource Management –**
    Big firms or organizations have numerous specialists or representatives working under them. They store data about worker's compensation, assessment, and work with the assistance of an information base administration framework (DBMS).

11. **Manufacturing –**
    Manufacturing organizations make various kinds of items and deal them consistently. To keep the data about their items like bills, acquisition of the item, amount, inventory network the executives, information base administration framework (DBMS) is utilized.

12. **Airline Reservation System –**
    This framework is equivalent to the railroad reservation framework. This framework additionally utilizes an information base administration framework to store the records of flight takeoff, appearance, and defer status.

Purpose of database system

The purpose of database systems is to make the database user-friendly and do easy operations. Users can easily insert, update, and delete. Actually, the main purpose is to have more control of the data.

The purpose of database systems is to manage the following insecurities:

- data redundancy and inconsistency,
- difficulty in accessing data,
- data isolation,
- atomicity of updates,
- concurrent access,
- security problems, and
- supports multiple views of data.

Avoid data redundancy and inconsistency:

If there are multiple copies of the same data, it just avoids it. It just maintains data in a single repository. Also, the purpose of database systems is to make the database consistent.

Difficulty in accessing data:

A database system can easily manage to access data. Through different queries, it can access data from the database.

Data isolation:

Data are isolated in several fields in the same database.

Atomicity of updates:

In case of power failure, the database might lose data. So, this feature will automatically prevent data loss.

Concurrent access:

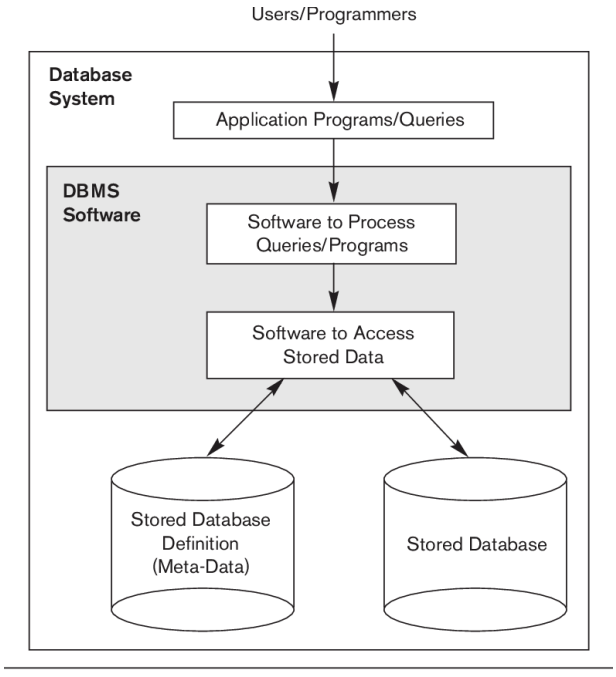
Users can have multiple access to the database at the same time.

Security problems:

Database systems will make the restricted access. So, the data will not be vulnerable.

Supports multiple views of data:

It can support multiple views of data to give the required view as their needs. Only database admins can have a complete view of the database. We cannot allow the end-users to have a view of developers.



Views of data

View of data in DBMS narrate how the data is visualized at each level of data abstraction? **Data abstraction** allow developers to keep complex data structures away from the users. The developers achieve this by hiding the complex data structures through **levels of abstraction**.

View of Data in DBMS

Data Abstraction

Data abstraction is **hiding the complex data structure** in order to **simplify the user's interface** of the system. It is done because many of the users interacting with the database system are not that much computer trained to understand the complex data structures of the database system.

To achieve data abstraction, we will discuss a **Three-Schema architecture** which abstracts the database at three levels discussed below:

Three-Schema Architecture:

The main objective of this architecture is to have an effective separation between the **user interface** and the **physical database**. So, the user never has to be concerned regarding the internal storage of the database and it has a simplified interaction with the database system.

The three-schema architecture defines the view of data at three levels:

1. Physical level (internal level)
2. Logical level (conceptual level)
3. View level (external level)

1. Physical Level/ Internal Level

The physical or the internal level schema describes **how the data is stored in the hardware**. It also describes how the data can be accessed. The physical level shows the data abstraction at the lowest level and it has **complex data structures**. Only the database administrator operates at this level.

2. Logical Level/ Conceptual Level

It is a level above the physical level. Here, the data is stored in the form of the **entity set**, **entities**, their **data types**, the **relationship** among the entity sets, **user operations** performed to retrieve or modify the data and certain **constraints on the data**. Well adding constraints to the view of data adds the security. As users are restricted to access some particular parts of the database.

It is the developer and database administrator who operates at the logical or the conceptual level.

> **View of Data in DBMS**

View of data in DBMS narrate how the data is visualized at each level of data abstraction? **Data abstraction** allow developers to keep complex data structures away from the users. The developers achieve this by hiding the complex data structures through **levels of abstraction**.

There is one more feature that should be kept in mind i.e. the **data independence**. While changing the data schema at one level of the database must not modify the data schema at the next level. In this section, we will discuss the view of data in DBMS with data abstraction, data independence, data schema in detail.

Content: View of Data in DBMS

Data Abstraction

Data abstraction is **hiding the complex data structure** in order to **simplify the user's interface** of the system. It is done because many of the users interacting with the database system are not that much computer trained to understand the complex data structures of the database system.

To achieve data abstraction, we will discuss a **Three-Schema architecture** which abstracts the database at three levels discussed below:

Three-Schema Architecture:

The main objective of this architecture is to have an effective separation between the **user interface** and the **physical database**. So, the user never has to be concerned regarding the internal storage of the database and it has a simplified interaction with the database system.

The three-schema architecture defines the view of data at three levels:

1. Physical level (internal level)
2. Logical level (conceptual level)
3. View level (external level)

1. Physical Level/ Internal Level

The physical or the internal level schema describes **how the data is stored in the hardware**. It also describes how the data can be accessed. The physical level shows the data abstraction at the lowest level and it has **complex data structures**. Only the database administrator operates at this level.

2. Logical Level/ Conceptual Level

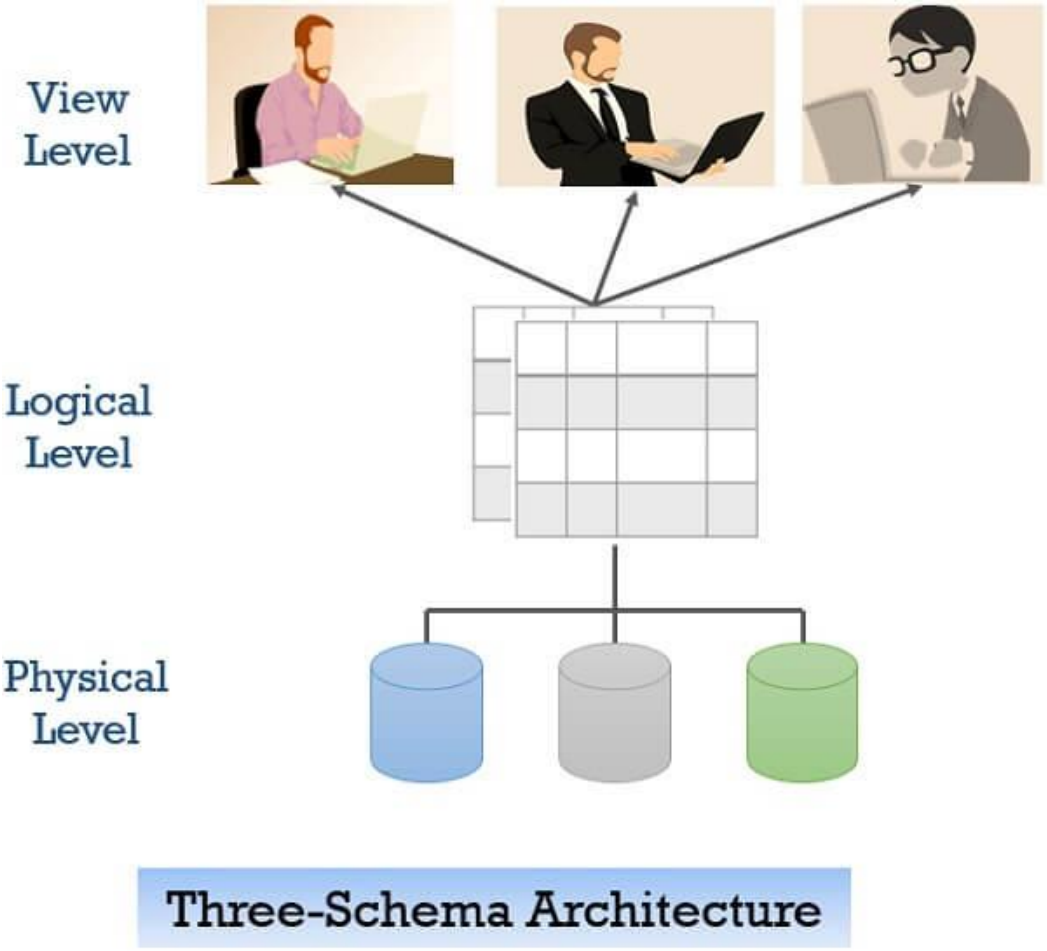
It is a level above the physical level. Here, the data is stored in the form of the **entity set**, **entities**, their **data types**, the **relationship** among the entity sets, **user operations** performed to retrieve or modify the data and certain **constraints on the data**. Well adding constraints to the view of data adds the security. As users are restricted to access some particular parts of the database.

It is the developer and database administrator who operates at the logical or the conceptual level.

3. View Level/ User level/ External level

It is the highest level of data abstraction and exhibits only a part of the whole database. It exhibits the data in which the user is interested. The view level can describe many views of the same data. Here, the user retrieves the information using different application from the database.

The figure below describes the three-schema architecture of the database:



Three-Schema Architecture

In the figure above you can clearly distinguish between the three levels of abstraction. To understand it more clearly let us take an example:

We have to create a database of a **college**. Now, what entity sets would be involved? **Student, Lecturer, Department, Course** and so on…

Now, the entity sets Student, Lecturer, Department, Course will be stored in the storage as the **consecutive blocks of the memory location**. This is the **physical or internal level** and is hidden from the programmers but the database administrator is it aware of it.

At the **logical level,** the programmers define the entity sets and relationship among these entity sets using a programming language like SQL. So, the programmers work at the logical level and even the database administrator also operates at this level.

At the **view level,** the users have the **set of applications** which they use to retrieve the data they are interested in.

Data Independence

Data independence defines the extent to which the data schema can be changed at one level without modifying the data schema at the next level. Data independence can be classified as shown below:

Logical Data Independence:

Logical data independence describes the degree up to which the logical or conceptual schema can be changed without modifying the external schema. Now, a question arises what is the need to change the data schema at a logical or conceptual level?

Well, the changes to data schema at the logical level are made either to **enlarge** or **reduce** the database by adding or deleting more entities, entity sets, or changing the constraints on data.

Physical Data Independence:

Physical data independence defines the extent up to which the data schema can be changed at the physical or internal level without modifying the data schema at logical and view level.

Well, the physical schema is changed if we add additional storage to the system or we reorganize some files to enhance the retrieval speed of the records.

Instances and Schemas

What is an instance?

We can define an instance as the information stored in the database at a particular point of time. Let us discuss it with the help of an example.

As we discussed above the database comprises of several entity sets and the relationship between them. Now, the data in the database keeps on changing with time. As we keep inserting or deleting the data to and from the database.

What is schema?

Whenever we talk about the database the developers have to deal with the definition of database and the data in the database.

The definition of a database comprises of the description of what data it would contain what would be the relationship between the data. This definition is the database schema.

Database Users and DBA

Database users are categorized based up on their interaction with the data base.

These are seven types of data base users in DBMS.

1. **Database Administrator (DBA) :**
   Database Administrator (DBA) is a person/team who defines the schema and also

controls the 3 levels of database.
The DBA will then create a new account id and password for the user if he/she need to access the data base.
DBA is also responsible for providing security to the data base and he allows only the authorized users to access/modify the data base.

- DBA also monitors the recovery and back up and provide technical support.
- The DBA has a DBA account in the DBMS which called a system or superuser account.
- DBA repairs damage caused due to hardware and/or software failures.

2. **Naive / Parametric End Users :**
   Parametric End Users are the unsophisticated who don't have any DBMS knowledge but they frequently use the data base applications in their daily life to get the desired results.
   For examples, Railway's ticket booking users are naive users. Clerks in any bank is a naive user because they don't have any DBMS knowledge but they still use the database and perform their given task.

3. **System Analyst :**
   System Analyst is a user who analyzes the requirements of parametric end users. They check whether all the requirements of end users are satisfied.

4. **Sophisticated Users :**
   Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database. They can develop their own data base applications according to their requirement. They don't write the program code but they interact the data base by writing SQL queries directly through the query processor.

5. **Data Base Designers :**
   Data Base Designers are the users who design the structure of data base which includes tables, indexes, views, constraints, triggers, stored procedures. He/she controls what data must be stored and how the data items to be related.

6. **Application Program :**
   Application Program are the back end programmers who writes the code for the application programs.They are the computer professionals. These programs could be written in Programming languages such as Visual Basic, Developer, C, FORTRAN, COBOL etc.

7. **Casual Users / Temporary Users :**
   Casual Users are the users who occasionally use/access the data base but each time when they access the data base they require the new information, for example, Middle or higher level manager.

Database Languages

Read, Update, Manipulate, and Store data in a database using Database Languages. The following are the database languages −

- Data Definition Language
- Data Manipulation Language
- Data Control Language
- Transaction Control Language

Let us begin with Data Definition Language:

Data Definition Language

The language is used to create database, tables, alter them, etc. With this, you can also rename the database, or drop them. It specifies the database schema.

The DDL statements include −

- **CREATE**: Create new database, table, etc.
- **ALTER**: Alter existing database, table, etc.
- **DROP**: Drop the database
- **RENAME**: Set a new name for the table.

Data Manipulation Language

The language used to manipulate the database like inserting data, updating table, retrieving record from a table, etc. is known as Data Manipulation Language −

- **SELECT**: Retrieve data from the database
- **INSERT**: Insert data
- **UPDATE**: Update data
- **DELETE**: Delete all records

Data Control Language

Grant privilege to a user using the GRANT statement. In the same way, revoke the privilege using the REVOKE statement. Both of these statements come under the Data Control Language (DCL). −

- **GRANT**: Give privilege to access the database.
- **REVOKE:** Take back the privilege to access the database.

Transaction Control Language

Manage transactions in the Database using the Transaction Control Language −

- **COMMIT**: Save the work.
- **SAVEPOINT**: Set a point in transaction to rollback later
- **ROLLBACK**: Restores since last commit

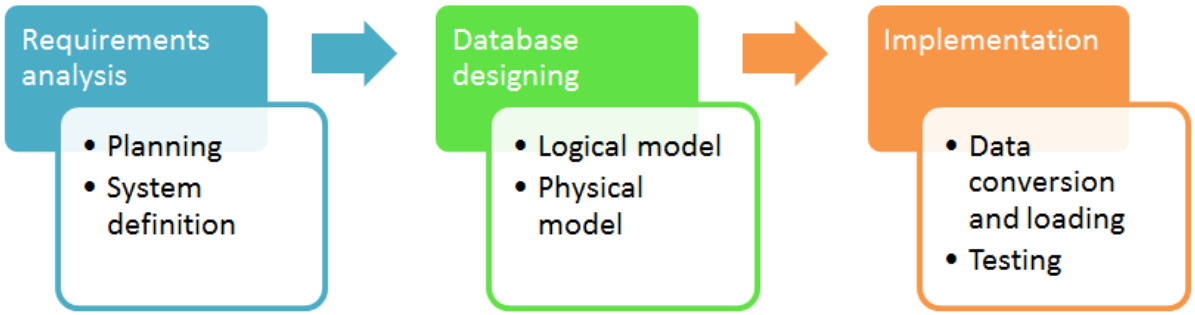
Introduction to Database design

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored.
The main objectives of database design in DBMS are to produce logical and physical designs models of the proposed database system.

The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.

The physical data design model involves translating the logical DB design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

Database development life cycle



The database development life cycle has a number of stages that are followed when developing database systems.

The steps in the development life cycle do not necessarily have to be followed religiously in a sequential manner.

On small database systems, the process of database design is usually very simple and does not involve a lot of steps.

In order to fully appreciate the above diagram, let's look at the individual components listed in each step for overview of design process in DBMS.

Requirements analysis

- **Planning** – This stages of database design concepts are concerned with planning of entire Database Development Life Cycle. It takes into consideration the Information Systems strategy of the organization.
- **System definition** – This stage defines the scope and boundaries of the proposed database system.

Database designing

- **Logical model** – This stage is concerned with developing a database model based on requirements. The entire design is on paper without any physical implementations or specific DBMS considerations.
- **Physical model** – This stage implements the logical model of the database taking into account the DBMS and physical implementation factors.

Implementation

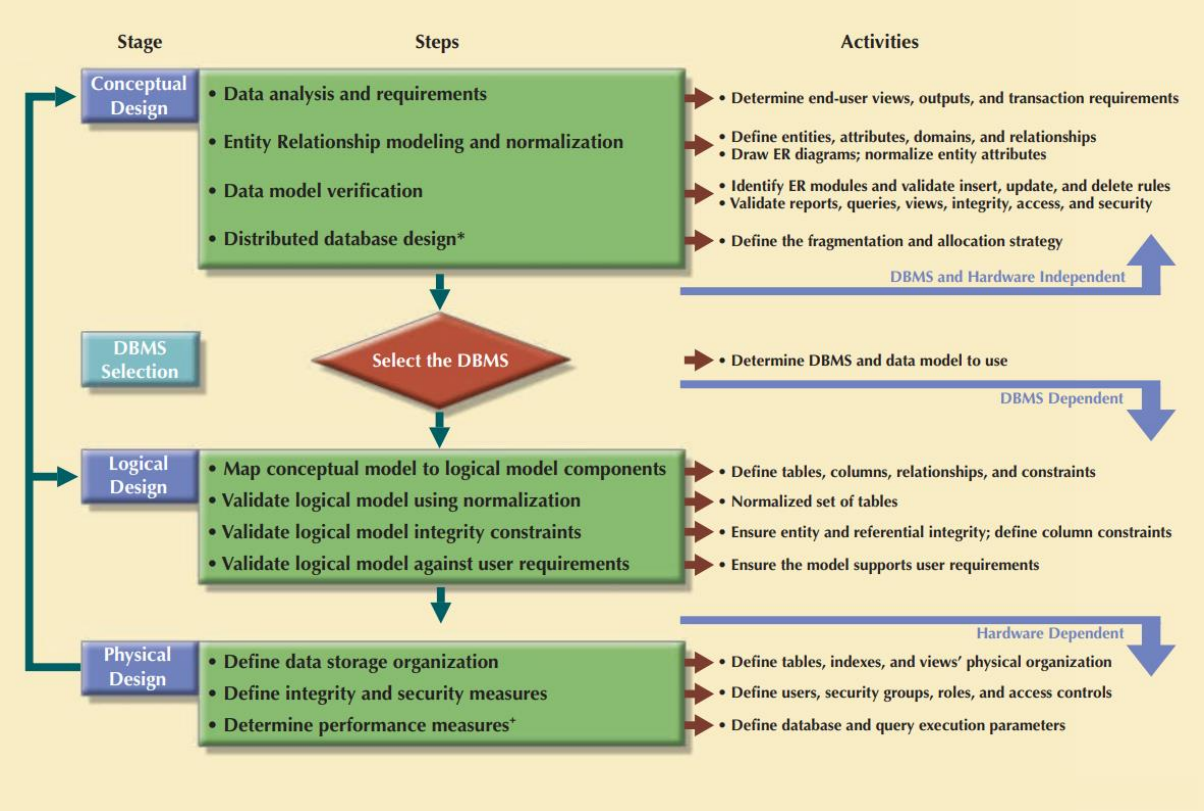
- **Data conversion and loading** – this stage of relational databases design is concerned with importing and converting data from the old system into the new database.
- **Testing** – this stage is concerned with the identification of errors in the newly implemented system. It checks the database against requirement specifications.

Two Types of Database Techniques

1. **Normalization**
2. **ER Modeling**

Design process

Database design process is a series of instructions detailing the creation of tables, attributes, domains, views, indexes, security constraints, and storage and performance guidelines. In this process, you implement all these design specifications.



The process starts with conceptual design and moves to the logical and physical design stages. At each stage, more details about the data model design are determined and documented. You could think of the conceptual design as the overall data as seen by the end-user, the logical design as the data as seen by the DBMS, and the physical design as the data as seen by the operating system's storage management devices.

1. Conceptual design

Conceptual design is the process that designs a conceptual data model that describes the main data entities, attributes, relationships, and constraints of a given problem domain represents real-world objects in the most realistic way possible. It must embody a clear understanding of the business and its functional areas. The design must be software- and hardware-independent.

The process should follow a minimal data rule: All that needed is there, and all that is there is needed. However, as you apply the minimal data rule, avoid excessive short-term bias. Focus not only on the immediate data needs of the business but on future data needs. Thus, the database design must leave room for future modifications and additions, ensuring that the business's investment in information resources will endure.

The conceptual design has four steps:

1.1 Data analysis and requirements

In order to design appropriate data element characteristics that can be transformed into appropriate information, the efforts should focus on

- Information needs. What kind of information is needed? That is, what output (reports and queries) must be generated by the system, what information does the current system generate, and to what extent is that information adequate?
- Information users. Who will use the information? How is the information to be used? What are the various end-user data views?
- Information sources. Where is the information to be found? How is the information to be extracted once it is found?
- Information constitution. What data elements are needed to produce the information? What are the data attributes? What relationships exist among the data? What is the data volume? How frequently are the data used? What data transformations will be used to generate the required information?

The answers to those questions are from a variety of sources to compile the necessary information:

- Developing and gathering end-user data views.
- Directly observing the current system.
- Interfacing with the systems design group.
- Understanding of the total business.

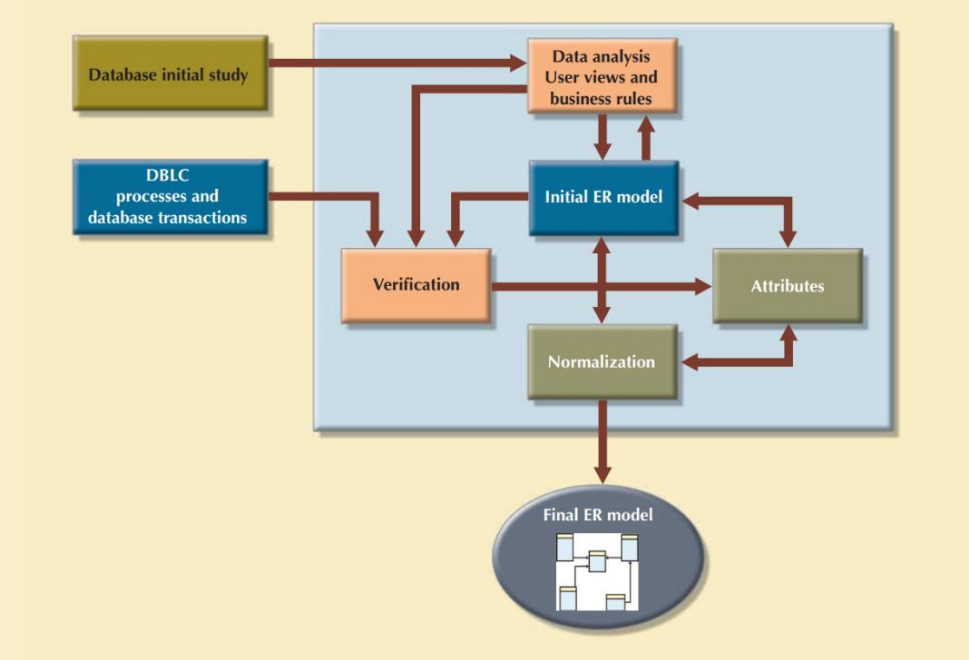
1.2 Entity relationship modeling and normalization

In this step, we could communicate and enforce appropriate standards to be used in the documentation of the design(diagrams and symbols, documentation writing style, layout, and any other conventions). Designers often overlook this very important requirement, especially when they are working as members of a design team.

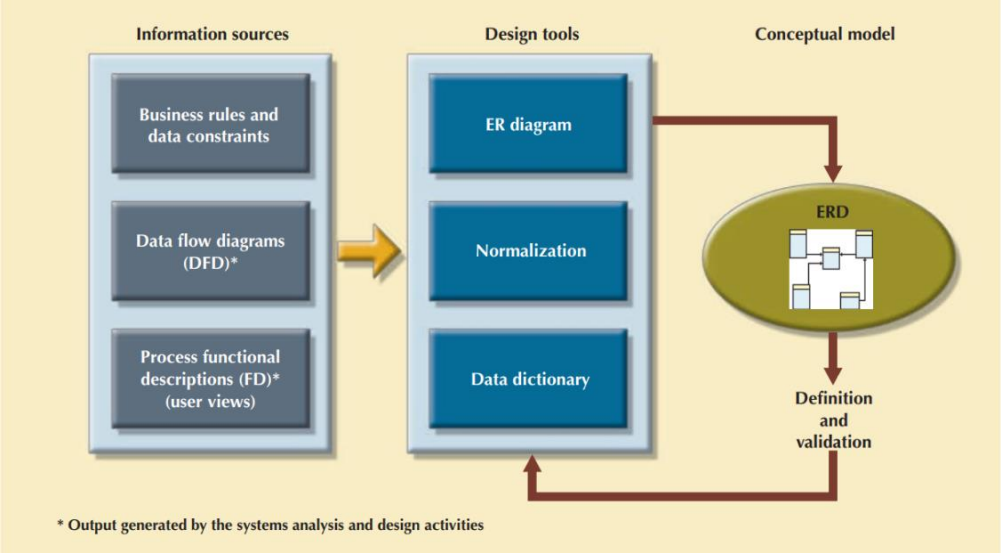
After that, the designer could incorporate business rules into the conceptual model using ER diagrams by following the steps:

1. Identify, analyze, and refine the business rules.
2. Identify the main entities, using the results of Step 1.
3. Define the relations among the entities, using the result of Steps 1 and 2.
4. Define the attributes, primary keys, and foreign keys for each of the entities.
5. Normalize the entities.
6. Complete the initial ER digaram.
7. Validate the ER model against the end users' information and processing requirements.
8. Modify the ER model, using the result of Step 7.

The activities often take place in parallel, and the process is iterative until you are satisfied that the ER model accurately represents a database design that can meet the required system demands



Below is the array of design tools and information sources that the designer can use to produce the conceptual model



All objects (entities, attributes, relations, views, and so on) are defined in a data dictionary, which is used in tandem with the normalization process to help eliminate data anomalies and redundancy problems. During this ER modeling process, the designer must:

- Define entities, attributes, primary keys, and foreign keys. (The foreign keys serve as the basis for the relationships among the entities.)
- Make decisions about adding new primary key attributes to satisfy end-user and processing requirements.

- Make decisions about the treatment of composite and multivalued attributes.
- Make decisions about adding derived attributes to satisfy processing requirements.
- Make decisions about the placement of foreign keys in 1:1 relationships.
- Avoid unnecessary ternary relationships.
- Draw the corresponding ER diagram.
- Normalize the entities.
- Include all data element definitions in the data dictionary.
- Make decisions about standard naming conventions.

The naming conventions requirement is important, yet it is frequently ignored at the designer's risk. Real database design is generally accomplished by teams. Therefore, it is important to ensure that team members work in an environment in which naming standards are defined and enforced. Proper documentation is crucial to the successful completion of the design, and adherence to the naming conventions serves database designers well. In fact, a common refrain from users seems to be: "I didn't know why you made such a fuss over naming conventions, but now that I'm doing this stuff for real, I've become a true believer."

1.3 Data model verification

It is one of the last steps in the conceptual design stage, and it is one of the most critical. In this step, the ER model must be verified against the proposed system processes to corroborate that they can be supported by the database model. Verification requires that the model be run through a series of tests against:

- End-user data views.
- All required transactions: SELECT, INSERT, UPDATE, and DELETE operations.
- Access rights and security.
- Business-imposed data requirements and constraints.

Because real-world database design is generally done by teams, the database design is probably divided into major components known as modules. A module is an information system component that handles a specific business function, such as inventory, orders, or payroll. Under these conditions, each module is supported by an ER segment that is a subset or fragment of an enterprise ER model. Working with modules accomplishes several important ends:

- The modules (and even the segments within them) can be delegated to design groups within teams, greatly speeding up the development work.

- The modules simplify the design work.
- The modules can be prototyped quickly.
- Even if the entire system cannot be brought online quickly, the implementation of one or more modules will demonstrate that progress is being made and that at least part of the system is ready to begin serving the end users.

As useful as modules are, they represent a loose collection of ER model fragments that could wreak havoc in the database if left unchecked. For example, the ER model fragments:

- Might present overlapping, duplicated, or conflicting views of the same data.
- Might not be able to support all processes in the system's modules.

To avoid these problems, it is better if the modules' ER fragments are merged into a single enterprise ER model. This process starts by selecting a central ER model segment and iteratively adding more ER model segments one at a time. At each stage, for each new entity added to the model, you need to validate that the new entity does not overlap or conflict with a previously identified entity in the enterprise ER model.

Merging the ER model segments into an enterprise ER model triggers a careful reevaluation of the entities, followed by a detailed examination of the attributes that describe those entities. This process serves several important purposes:

- The emergence of the attribute details might lead to a revision of the entities themselves.
- The focus on attribute details can provide clues about the nature of relationships as they are defined by the primary and foreign keys.
- To satisfy processing and end-user requirements, it might be useful to create a new primary key to replace an existing primary key.
- Unless the entity details (the attributes and their characteristics) are precisely defined, it is difficult to evaluate the extent of the design's normalization.
- A careful review of the rough database design blueprint is likely to lead to revisions.

After finishing the merging process, the resulting enterprise ER model is verified against each of the module's processes

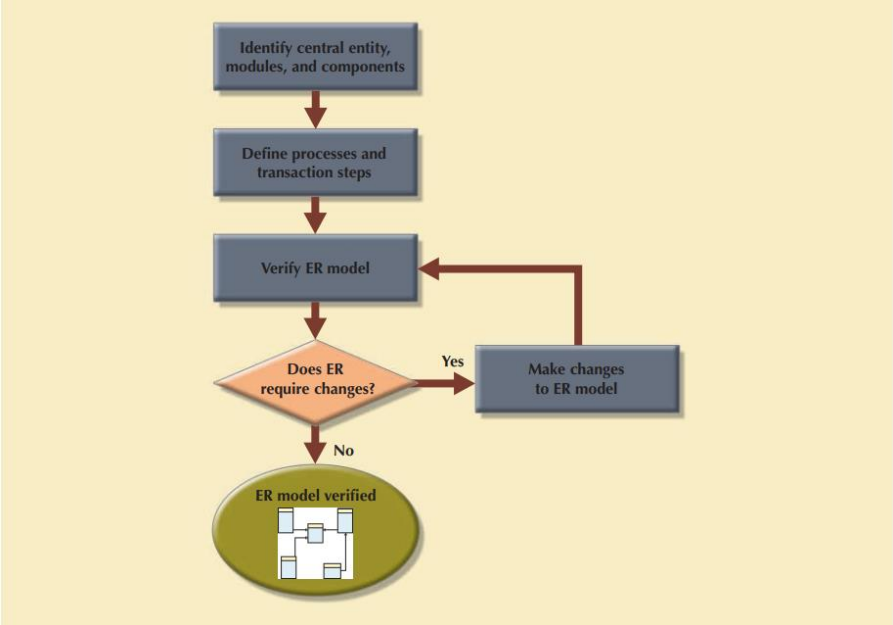
1. Identify the ER model's center entity.
2. Identify each model and its components.

3. Identify each module’s transaction requirements

   Internal: updates/inserts/delets/queries/reports

   External: modeul interfaces
4. Verify all process against system requirments.
5. Make all necessary changes suggested in Step 4.
6. Repeat Steps 2–5 for all modules.

Keep in mind that this process requires the continuous verification of business transactions as well as system and user requirements. The verification sequence must be repeated for each of the system’s modules.



The verification process starts with selecting the central (most important) entity, which is the focus for most of the system’s operations.

To identify the central entity, the designer selects the entity involved in the greatest number of the model’s relationships. In the ER diagram, it is the entity with more lines connected to it than any other.

The next step is to identify the module or subsystem to which the central entity belongs and to define that module’s boundaries and scope. The entity belongs to the module that uses it most frequently.

Once each module is identified, the central entity is placed within the module’s framework to let you focus on the module’s details.

Within the central entity/module framework, you must

- Ensure the module’s cohesivity.

- Analyze each module's relationships with other modules to address module coupling.

Processes may be classified according to their

- Frequency (daily, weekly, monthly, yearly, or exceptions).
- Operational type (INSERT or ADD, UPDATE or CHANGE, DELETE, queries and reports, batches, maintenance, and backups).

All identified processes must be verified against the ER model. If necessary, appropriate changes are implemented. The process verification is repeated for all of the model's modules. You can expect that additional entities and attributes will be incorporated into the conceptual model during its validation. At this point, a conceptual model has been defined as hardware- and software-independent. Such independence ensures the system's portability across platforms. Portability can extend the database's life by making it possible to migrate to another DBMS and hardware platform.

1.4 Distributed database design

Although not a requirement for most databases, some may need to be distributed among multiple geographical locations. Processes that access the database may also vary from one location to another. For example, a retail process and a warehouse storage process are likely to be found in different physical locations. If the database data and processes will be distributed across the system, portions of a database, known as database fragments, may reside in several physical locations. A database fragment is a subset of a database that is stored at a given location. The database fragment may be a subset of rows or columns from one or multiple tables.

Distributed database design defines the optimum allocation strategy for database fragments to ensure database integrity, security, and performance. The allocation strategy determines how to partition the database and where to store each fragment.

2. DMBS software selection

DBMS software selection is critical, we should carefully study the advantages and disadvantages. The common factors that affect the selection are:

- Cost.
- DBMS features and tools.
- Underlying model.
- Portability.
- DBMS hardware requirements.

3. Logical design

Logical design goal is to design an enterprise-wide database that is based on a specific data model but independent of physical-level details. It requires that all objects in the conceptual model be mapped to the specific constructs used by the selected database model.

The logical design generally performed in four steps:

- Map the conceptual model to logical model components.
- Validate the logical model using normalization.
- Validate the logical model integrity constraints.
- Validate the logical model against user requirements.

4. Physical design

Physical design is the process of determining the data storage organization and data access characteristics of the database to ensure its integrity, security, and performance. The storage characteristics are a function of the types of devices supported by the hardware, the type of data access methods supported by the system, and the DMBS.

The physical design consists of 3 steps:

- Define data storage organization.
- Define integrity and security measures.
- Determine performance measurements.

Entity Relation Model

Entity relationship (ER) models are based on the real-world entities and their relationships. It is easy for the developers to understand the system by simply looking at the ER diagram. ER models are normally represented by ER-diagrams.

Components

ER diagram basically having three components:

- **Entities** − It is a real-world thing which can be a person, place, or even a concept. For Example: Department, Admin, Courses, Teachers, Students, Building, etc are some of the entities of a School Management System.
- **Attributes** − An entity which contains a real-world property called an attribute. For Example: The entity employee has the property like employee id, salary, age, etc.
- **Relationship** − Relationship tells how two attributes are related. For Example: Employee works for a department.

An entity has a real-world property called attribute and these attributes are defined by a set of values called domain.

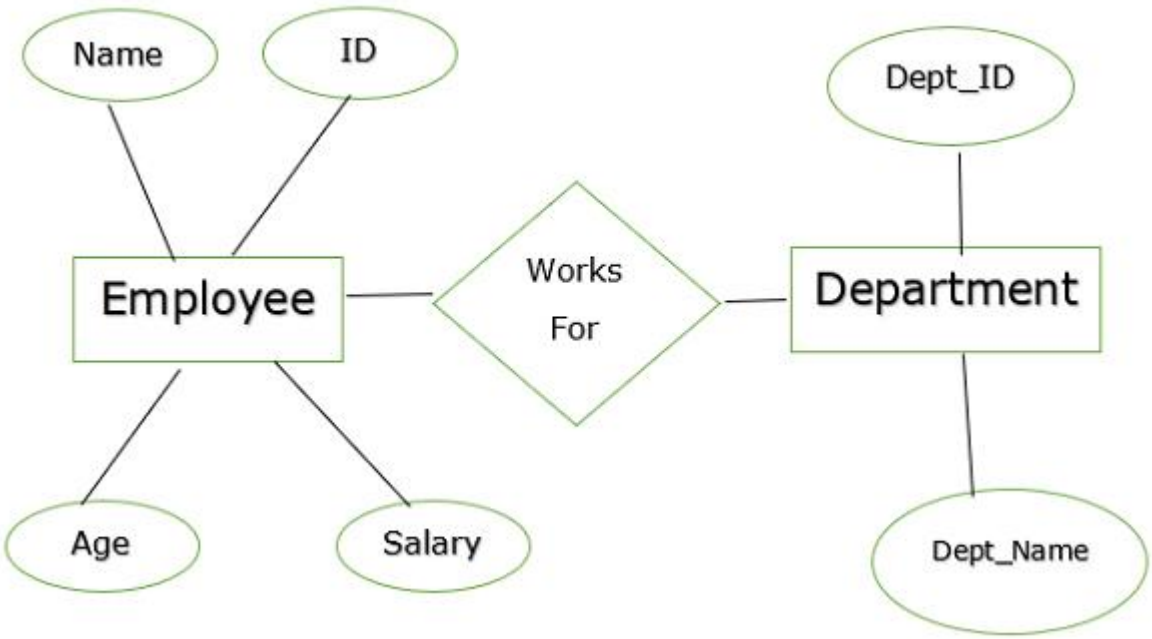
Example 1

In a university,

- A student is an entity,
- University is the database,
- Name and age and sex are the attributes.
- The relationships among entities define the logical association between entities.

Example 2

Given below is another example of ER:



In the above example,

Entities − Employee and Department.

Attributes −

- Employee − Name, id, Age, Salary
- Department − Dept_id, Dept_name

The two entities are connected using the relationship. Here, each employee works for a department.

Features of ER

The features of ER Model are as follows −

- **Graphical Representation is Better Understanding** − It is easy and simple to understand so it can be used by the developers to communicate with the stakeholders.

- **ER Diagram** − ER diagrams are used as a visual tool for representing the model.
- **Database Design** − This model helps the database designers to build the database.

Advantages

The advantages of ER are as follows −

- The ER model is easy to build.
- This model is widely used by database designers for communicating their ideas.
- This model can easily convert to any other model like network model, hierarchical model etc.
- It is integrated with the dominant relational model.

Disadvantages

The disadvantages of ER are as follows −

- There is no industry standard for developing an ER model.
- Information might be lost or hidden in the ER model.
- There is no Data Manipulation Language (DML).
- There is limited relationship representation.

ER diagram

Why use ER Diagrams?

Here, are prime reasons for using the ER Diagram

- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ERD Diagram allows you to communicate with the logical structure of the database to users

Facts about ER Diagram Model

Now in this ERD Diagram Tutorial, let's check out some interesting facts about ER Diagram Model:

- ER model allows you to draw Database Design
- It is an easy to use graphical tool for modeling data

- Widely used in Database Design
- It is a GUI representation of the logical structure of a Database
- It helps you to identifies the entities which exist in a system and the relationships between those entities

ER Diagrams Symbols & Notations

Entity Relationship Diagram Symbols & Notations mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes. There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

Following are the main components and its symbols in ER Diagrams:

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes

ER Diagram Symbols

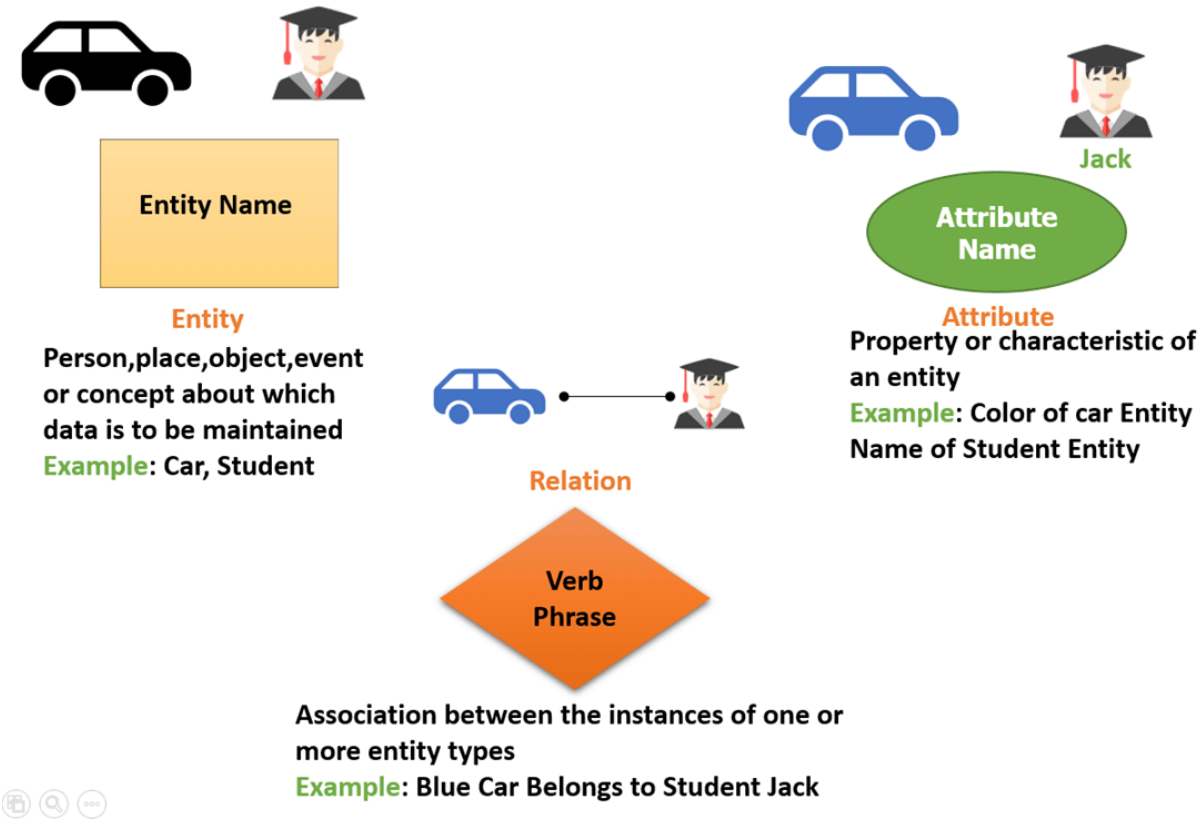
Components of the ER Diagram

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships

ER Diagram Examples

For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.



Components of the ER Diagram

WHAT IS ENTITY?

A real-world thing either living or non-living that is easily recognizable and nonrecognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.

An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some 'attributes' which represent that entity.

Examples of entities:

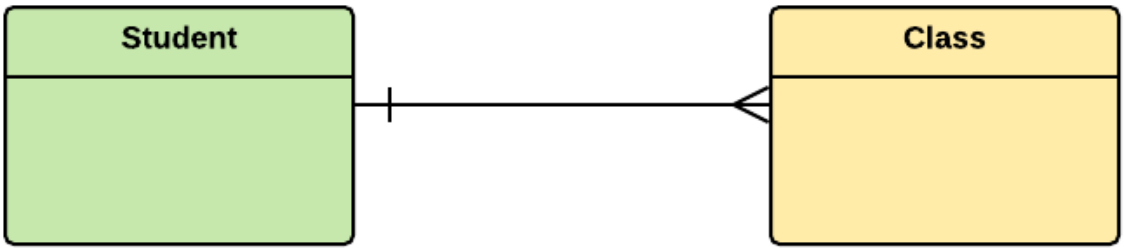
- **Person:** Employee, Student, Patient
- **Place:** Store, Building
- **Object:** Machine, product, and Car
- **Event:** Sale, Registration, Renewal
- **Concept:** Account, Course

Notation of an Entity

Entity set:

Student

An entity set is a group of similar kind of entities. It may contain entities with attribute sharing similar values. Entities are represented by their properties, which also called attributes. All attributes have their separate values. For example, a student entity may have a name, age, class, as attributes.



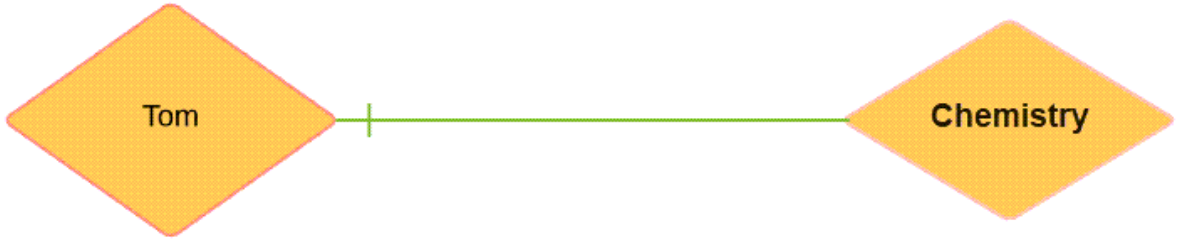
Example of Entities:

A university may have some departments. All these departments employ various lecturers and offer several programs.

Some courses make up each program. Students register in a particular program and enroll in various courses. A lecturer from the specific department takes each course, and each lecturer teaches a various group of students.

Relationship

Relationship is nothing but an association among two or more entities. E.g., Tom works in the Chemistry department.



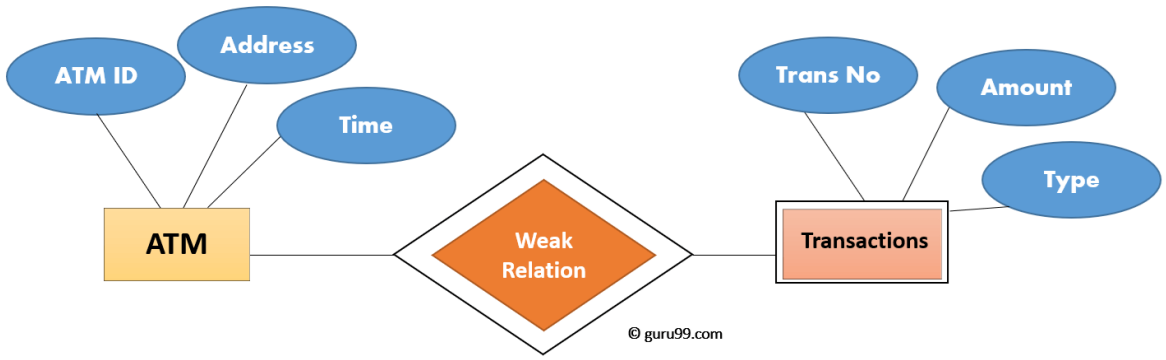
Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

For example:

- You are attending this lecture
- I am giving the lecture
- Just loke entities, we can classify relationships according to relationship-types:
- A student attends a lecture
- A lecturer is giving a lecture.

Weak Entities

A weak entity is a type of entity which doesn't have its key attribute. It can be identified uniquely by considering the primary key of another entity. For that, weak entity sets need to have participation.



In above ER Diagram examples, “Trans No” is a discriminator within a group of transactions in an ATM.

Let’s learn more about a weak entity by comparing it with a Strong Entity

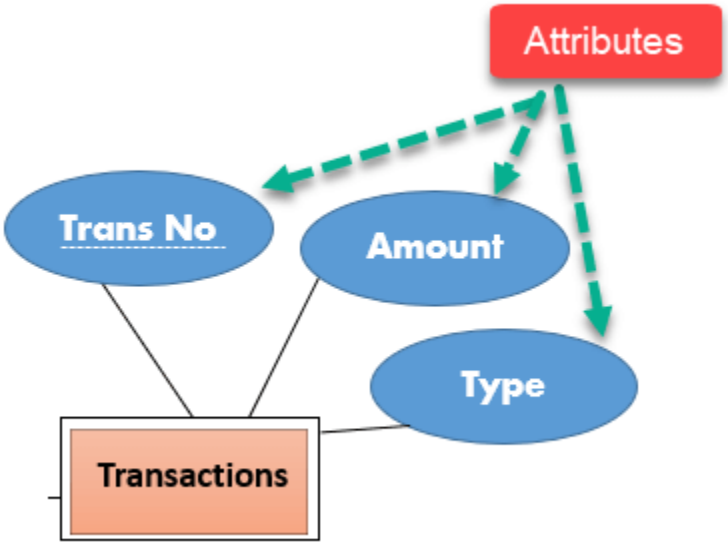
Strong Entity Set	Weak Entity Set
Strong entity set always has a primary key.	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.
Primary Key is one of its attributes which helps to identify its member.	In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

Attributes

It is a single-valued property of either an entity-type or a relationship-type.

For example, a lecture might have attributes: time, date, duration, place, etc.

An attribute in ER Diagram examples, is represented by an Ellipse



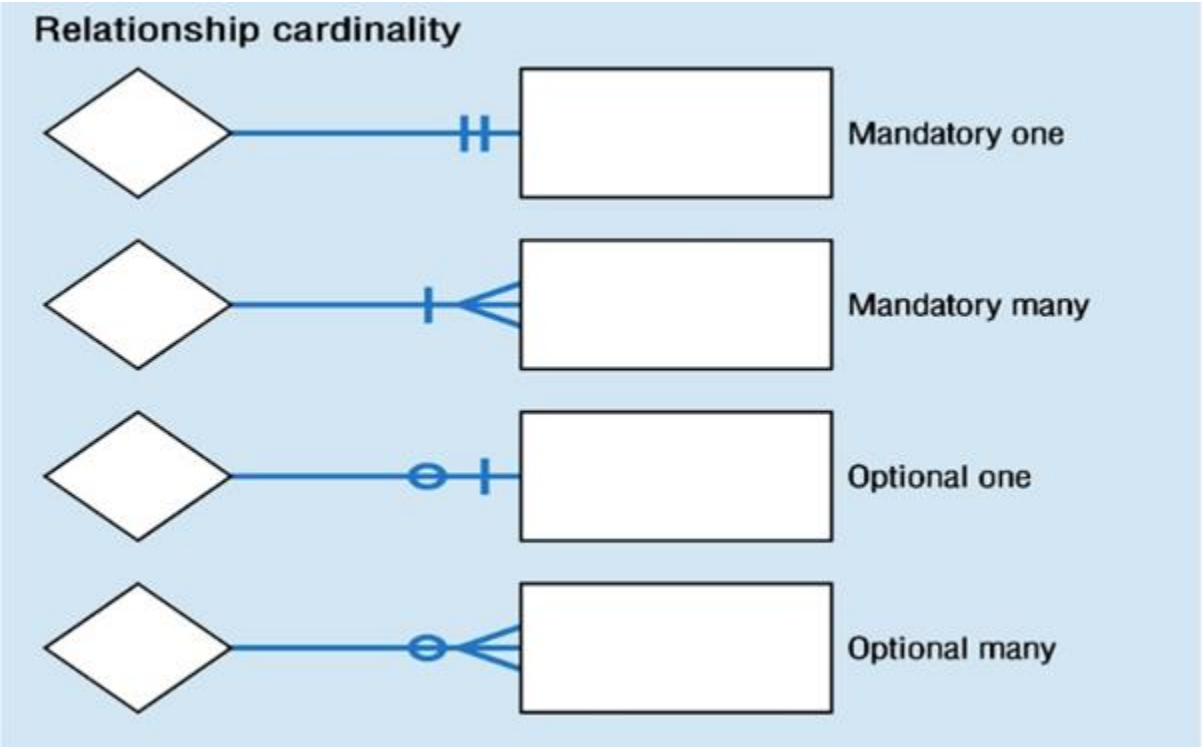
Types of Attributes	Description
Simple attribute	Simple attributes can't be divided any further. For example, a student's contact number. It is also called an atomic value.
Composite attribute	It is possible to break down composite attribute. For example, a student's full name may be further divided into first name, second name, and last name.
Derived attribute	This type of attribute does not include in the physical database. However, their values are derived from other attributes present in the database. For example, age should not be stored directly. Instead, it should be derived from the DOB of that employee.
Multivalued attribute	Multivalued attributes can have more than one values. For example, a student can have more than one mobile number, email address, etc.

Cardinality

Defines the numerical attributes of the relationship between two entities or entity sets.

Different types of cardinal relationships are:

- One-to-One Relationships
- One-to-Many Relationships
- May to One Relationships
- Many-to-Many Relationships



1.One-to-one:

One entity from entity set X can be associated with at most one entity of entity set Y and vice versa.

Example: One student can register for numerous courses. However, all those courses have a single line back to that one student.

2.One-to-many:

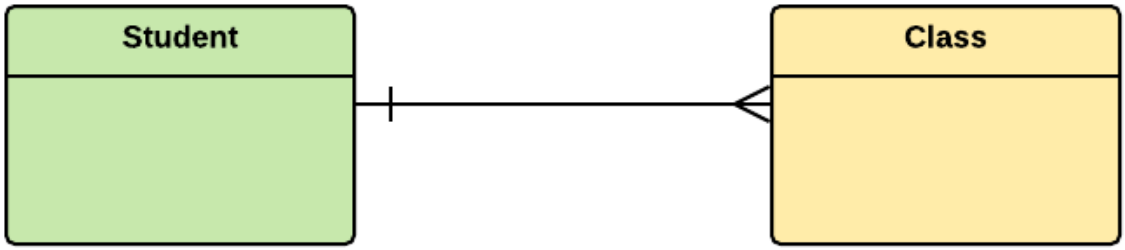
One entity from entity set X can be associated with multiple entities of entity set Y, but an entity from entity set Y can be associated with at least one entity.

For example, one class is consisting of multiple students.

3. Many to One

More than one entity from entity set X can be associated with at most one entity of entity set Y. However, an entity from entity set Y may or may not be associated with more than one entity from entity set X.

For example, many students belong to the same class.



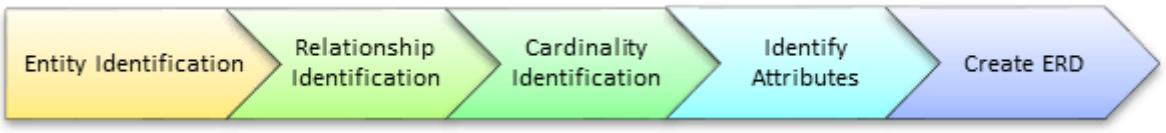
4. Many to Many:

One entity from X can be associated with more than one entity from Y and vice versa.

For example, Students as a group are associated with multiple faculty members, and faculty members can be associated with multiple students.

How to Create an Entity Relationship Diagram (ERD)

Now in this ERD Diagram Tutorial, we will learn how to create an ER Diagram. Following are the steps to create an ER Diagram:



Steps to Create an ER Diagram

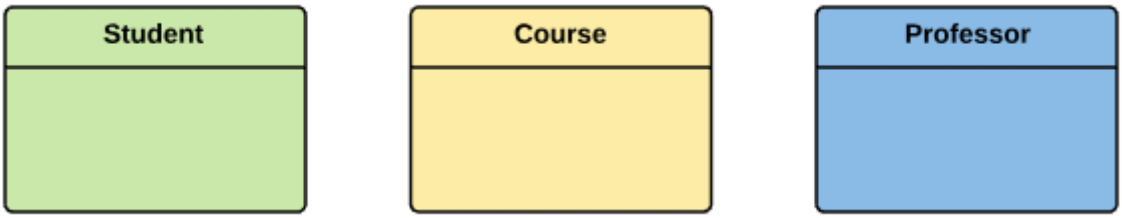
Let's study them with an Entity Relationship Diagram Example:

In a university, a Student enrolls in Courses. A student must be assigned to at least one or more Courses. Each course is taught by a single Professor. To maintain instruction quality, a Professor can deliver only one course

Step 1) Entity Identification

We have three entities

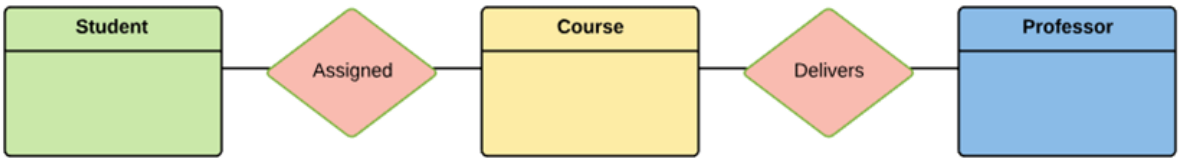
- Student
- Course
- Professor



Step 2) Relationship Identification

We have the following two relationships

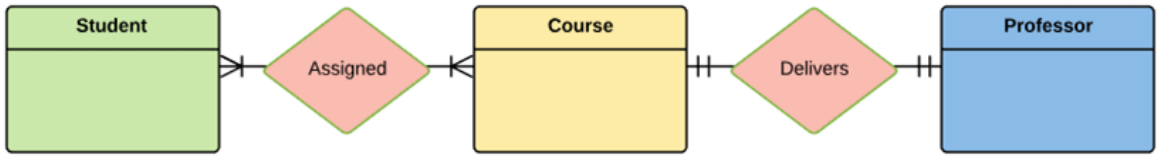
- The student is **assigned** a course
- Professor **delivers** a course



Step 3) Cardinality Identification

For them problem statement we know that,

- A student can be assigned **multiple** courses
- A Professor can deliver only **one** course



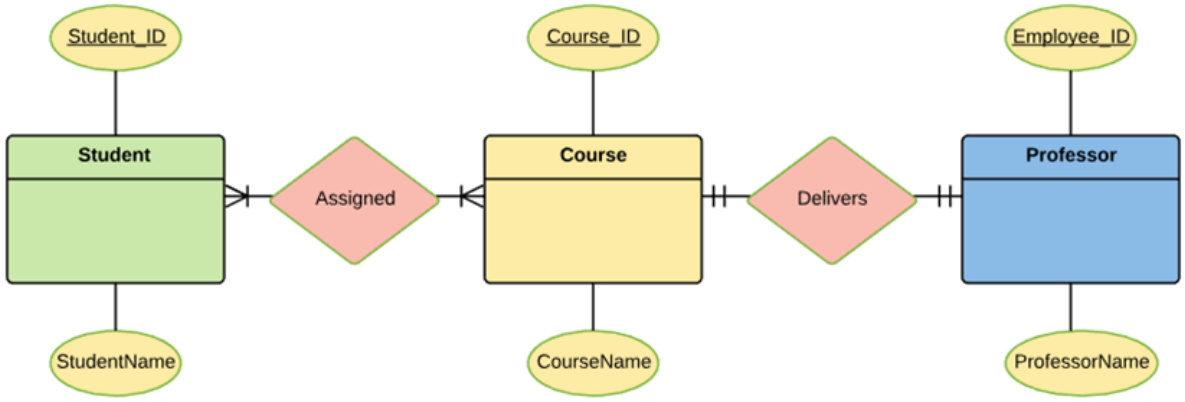
Step 4) Identify Attributes

You need to study the files, forms, reports, data currently maintained by the organization to identify attributes. You can also conduct interviews with various stakeholders to identify entities. Initially, it's important to identify the attributes without mapping them to a particular entity.

Once, you have a list of Attributes, you need to map them to the identified entities. Ensure an attribute is to be paired with exactly one entity. If you think an attribute should belong to more than one entity, use a modifier to make it unique.

Once the mapping is done, identify the primary Keys. If a unique key is not readily available, create one.

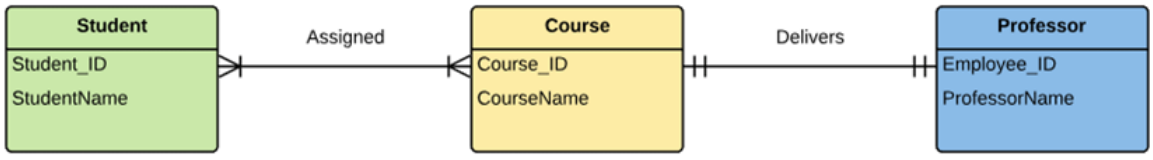
Entity	Primary Key	Attribute
Student	Student_ID	StudentName
Professor	Employee_ID	ProfessorName
Course	Course_ID	CourseName



For Course Entity, attributes could be Duration, Credits, Assignments, etc. For the sake of ease we have considered just one attribute.

Step 5) Create the ERD Diagram

A more modern representation of Entity Relationship Diagram Example



Best Practices for Developing Effective ER Diagrams

Here are some best practice or example for Developing Effective ER Diagrams.

- Eliminate any redundant entities or relationships
- You need to make sure that all your entities and relationships are properly labeled
- There may be various valid approaches to an ER diagram. You need to make sure that the ER diagram supports all the data you need to store
- You should assure that each entity only appears a single time in the ER diagram
- Name every relationship, entity, and attribute are represented on your diagram
- Never connect relationships to each other
- You should use colors to highlight important portions of the ER diagram