**1. Introduction of Database Management System (DBMS)**

In todays competitive environment data and its efficient management is the critical business objective of an organization. It is a difficult task to get the right information at the right time to take right decision. Therefore the success of an organization is now more than ever dependent on its ability to acquire accurate, reliable and timely data about its business or operation for effective decision-making process.

Database system is a tool that simplifies the above tasks of managing the data and extracting useful information in a timely fashion.

In fact database and database management system (DBMS) have some essential for managing our business, governments, banks, electronic commerce, engineering, law, education, medical and every other kind of human endeavour.

**1.1 Data :-** Itmay be defined as raw facts from which required information is produced. For example, names, telephone numbers and addresses of the people.

**1.2 Database**:- It is a collection of data which contains information relevant to an enterprise.

**1.3 Database Management System:-** Itis a collection of interrelated data and a set of programs to access those data The primary goal of a DBMS is to provide away to store and retrieve database information that is both *convenient and efficient.*

The DBMS is hence a general-purpose software system that facilitates the processes of defining, constructing , manipulating and sharing databases among various users and application.

* **Defining :**specifying the data types, structures, and constraints for the data.
* **Constructing :**includes storing the data itself on some storage medium.
* **Manipulating :**includes querying the database to retrieve specific data, updating the data to reflect changes in data, generating reports from data.
* **Sharing :**allows multiple users to access the database concurrently.

**2. Applications of DBMS**

Database are widely used. The some of the representative applications are:

1. **Banking :**for customer information, accounts and loans and banking transactions.
2. **Universities :**for student registrations and grades.
3. **Online shopping :**Everyone wants to shop from home. Everyday new products are added and sold only with the help of DBMS. Purchase information, invoice bills and payment, all of these are done with the help of DBMS.
4. **Airlines :**for reservations and schedule information.
5. **Credit card transactions :**for purchases on credit cards and generation of monthly statements.
6. **Library Management System :**maintain all the information relate to book issue dates, name of the book, author and availability of the book.
7. **Telecommunications :**for keeping records of call made, generating monthly bills, maintaining balances on prepaid calling cards.
8. **Sales :**for customer, product and purchase information.
9. **Finance :**for storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
10. **Manufacturing :**for management of supply chain and for tracking production of items in factories, inventories of items and orders for items.
11. **Human Resource :**for information about employees, salaries, payroll taxes and benefits.

3. **Purpose of Database Systems**

Database systems arose in response to early methods of computerized management of commercial data. As an example of such methods, typical of the 1960s, consider part of a university organization that, among other data, keeps information about all instructors, students, departments, and course offerings. One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the information, the system has a number of application programs that manipulate the files, including programs to:

1. Add new students, instructors, and courses

2. Register students for courses and generate class rosters

3. Assign grades to students, compute grade point averages (GPA), and generate

Transcripts.

System programmers wrote these application programs to meet the needs of the university.

New application programs are added to the system as the need arises. For example, suppose that a university decides to create a new major (say, computer science).As a result, the university creates a new department and creates new permanent files (or adds information to existing files) to record information about all the instructors in the department, students in that major, course offerings, degree requirements, etc. The university may have to write new application programs to deal with rules specific to the new major. New application programs may also have to be written to handle new rules in the university. Thus, as time goes by, the system acquires more files and more application programs.

This typical file-processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. Before database management systems (DBMSs) were introduced, organizations usually stored information in such systems.

Keeping organizational information in a file-processing system has a number of major disadvantages:

• **Data redundancy and inconsistency:** Since different programmers create the files and application programs over a long period, the various files are likely to have different structures and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files). For example, if a student has a double major (say, music and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Music department and in a file that consists of student records of students in the Mathematics department. This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency; that is, the various copies of the same data may no longer agree. For example, a changed student address may be reflected in the Music department records but not elsewhere in the system.

• **Difficulty in accessing data:** Suppose that one of the university clerks needs to find out the names of all students who live within a particular postal-code area. The clerk asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. There is, however, an application program to generate the list of all students. The university clerk has now two choices: either obtain the list of all students and extract the needed information manually or ask a programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same clerk needs to trim that list to include only those students who have taken at least 60 credit hours. As expected, a program to generate such a list does not exist. Again, the clerk has the preceding two options, neither of which is satisfactory. The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficientmanner. More responsive data-retrieval systems are required for general use.

• **Data isolation:** Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

• **Integrity problems:** The data values stored in the database must satisfy certain types of consistency constraints. Suppose the university maintains an account for each department, and records the balance amount in each account. Suppose also that the university requires that the account balance of a department may never fall below zero. Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

• **Atomicity problems:** A computer system, like any other device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure. Consider a program to transfer $500 from the account balance of department A to the account balance of department B. If a system failure occurs during the execution of the program, it is possible that the $500 was removed from the balance of department A but was not credited to the balance of department B, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur. That is, the funds transfer must be atomic—it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.

• **Concurrent-access anomalies:** For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. Indeed, today, the largest Internet retailers may have millions of accesses per day to their data by shoppers. In such an environment, interaction of concurrent updates is possible and may rsult in inconsistent

data. Consider department A, with an account balance of $10,000. If two department clerks debit the account balance (by say $500 and $100, respectively) of department A at almost exactly the same time, the result of the concurrent executions may leave the budget in an incorrect (or inconsistent) state. Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value $10,000, and write back $9500 and $9900, respectively. Depending on which one writes the value last, the account balance of department A may contain either $9500 or $9900, rather than the correct value of $9400. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously. As another example, suppose a registration program maintains a count of students registered for a course, in order to enforce limits on the number of students registered. When a student registers, the program reads the current count for the courses, verifies that the count is not already at the limit, adds one to the count, and stores the count back in the database. Suppose two students register concurrently, with the count at (say) 39. The two program executions may both read the value 39, and both would then write back 40,leading to an incorrect increase of only 1, even though two students successfully registered for the course and the count should be 41. Furthermore, suppose the course registration limit was 40; in the above case both students would be able to register, leading to a violation of the limit of 40 students.

• **Security problems:** Not every user of the database system should be able to access all the data. For example, in a university, payroll personnel need to see only that part of the database that has financial information. They do not need access to information about academic records. But, since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult.

These difficulties, among others, prompted the development of database systems. In what follows, we shall see the concepts and algorithms that enable database systems to solve the problems with file-processing systems.

## 4. Advantages of DBMS

#### 4.1 Controlling of Redundancy :

Data redundancy refers to the **duplication of data (i.e storing same data multiple times).** In a database system, by having a centralized database and centralized control of data by the DBA the unnecessary duplication of data is avoided. It also eliminates the extra time for processing the large volume of data. It results in saving the storage space.

#### 4.2 Improved Data Sharing :

DBMS allows a user to share the data in any number of application programs.

#### 4.3 Data Integrity :

Data integrity is the overall completeness, accuracy and consistency of data. For example a bank maintains separate customer files for each type of account, when a customer moves to a new address, his/her address field must be updated in all customer files containing this customer record.

Integrity of data is necessary to avoid confusion that may result when one file is updated while others are not.

#### 4.4 Security :

Having complete authority over the operational data, enables the DBA in ensuring that the only mean of access to the database is through proper channels. The DBA can define authorization checks to be carried out whenever access to sensitive data is attempted.

#### 4.5 Data Consistency :

By eliminating data redundancy, we greatly reduce the opportunities for inconsistency. For example: is a customer address is stored only once, we cannot have disagreement on the stored values. Also updating data values is greatly simplified when each value is stored in one place only. Finally, we avoid the wasted storage that results from redundant data storage.

#### 4.6 Efficient Data Access :

In a database system, the data is managed by the DBMS and all access to the data is through the DBMS providing a key to effective data processing

#### 4.7 Enforcements of Standards :

With the centralized of data, DBA can establish and enforce the data standards which may include the naming conventions, data quality standards etc.

#### 4.8 Data Independence :

In a database system, the database management system provides the interface between the application programs and the data. When changes are made to the data representation, the meta data obtained by the DBMS is changed but the DBMS is continues to provide the data to application program in the previously used way. The DBMs handles the task of transformation of data wherever necessary.

#### 4.9 Reduced Application Development and Maintenance Time :

DBMS supports many important functions that are common to many applications, accessing data stored in the DBMS, which facilitates the quick development of application.

**5. Disadvantages of DBMS**

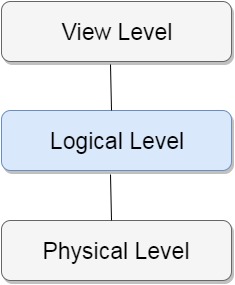
1. Increased Complexity
2. Requirement of New and Specialized Manpowers
3. Large Size of DBMS
4. Increased installation and management cost
5. Additional hardware cost
6. Conversion cost
7. Need for explicit backup and recovery
8. Organizational conflict

**6. Database System versus File System**

|  |  |
| --- | --- |
| **DBMS** | **File Processing System** |
| Minimal data redundancy problem in DBMS | Data Redundancy problem exits |
| Data Inconsistency does not exist | Data Inconsistency exist here |
| Accessing database is easier | Accessing is comparatively difficult |
| The problem of data isolation is not found in database | Data is scattered in various files and files may be of different format, so data isolation problem exists |
| Transactions like insert, delete, view, updating, etc are possible in database | In file system, transactions are not possible |
| Concurrent access and recovery is possible in database | Concurrent access and recovery is not possible |
| Security of data | Security of data is not good |
| A database manager (administrator) stores the relationship in form of structural tables | A file manager is used to store all relationships in directories in file systems. |

## 7. Data Abstraction

For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Developers hide the complexity from users through several levels of abstraction to simplify users interactions with the system.



* **Physical Level :**The lowest level of abstraction describes **how** the data are actually stored.
* **Logical Level :**The next-higher level of abstraction describes **what** data are stored in the database, and what relationships exist among those data.
* **View Level :**The highest level of abstraction describes only part of the entire database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many view for the same database.

**7. Data Independence**

Data Independence can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

Data Independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed.

There are two types of data independence :

1. Logical Data Independence
2. Physical Data Independence

### 7.1 Logical Data Independence

It is the capacity to change the conceptual schema without having to change external schemas or application program.

We may change the conceptual schema to expand the database, to change constraints, or to reduce the database.

### 7.2 Physical Data Independence

It is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schema need not to be changed as well.

Changes to the internal schema may be needed because some physical files had to be reorganized.

**DBMS Database Models**

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the **Relational Model** is the most widely used database model, there are other models too:

* Hierarchical Model
* Network Model
* Entity-relationship Model
* Relational Model

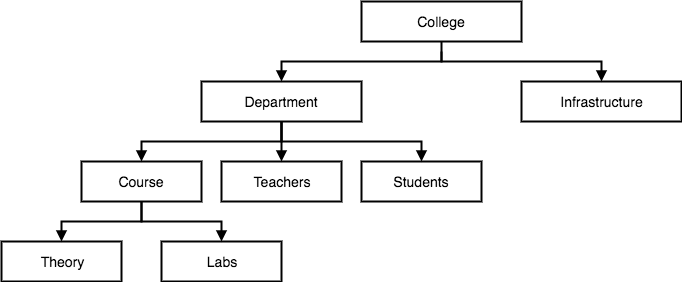
**Hierarchical Model**

This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node.

This model efficiently describes many real-world relationships like index of a book, recipes etc.

In hierarchical model, data is organized into tree-like structure with one **one-to-many relationship** between two different types of data, for example, one department can have many courses, many professors and of-course many students.

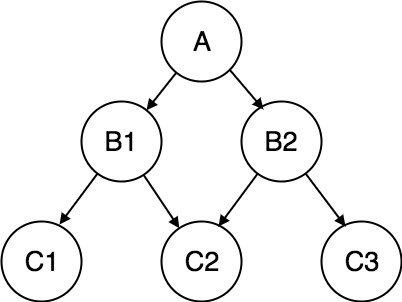


**Network Model**

This is an extension of the Hierarchical model. In this model data is organized more like a graph, and are allowed to have more than one parent node.

In this database model, data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map **many-to-many** data relationships.

This was the most widely used database model, before Relational Model was introduced.



**Entity-relationship Model**

In this database model, relationships are created by dividing object of interest into **entity** and its characteristics into **attributes.**

Different entities are related using relationships.

E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.



E-R model of real world

\_ Entities (objects)

✔ E.g. customers, accounts, bank branch

\_ Relationships between entities

✔ E.g. Account A-101 is held by customer Johnson

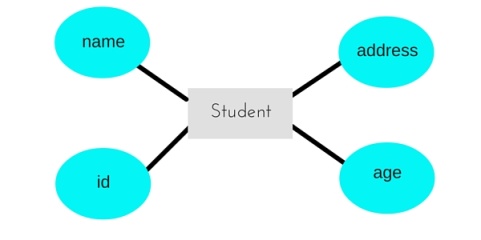
✔ Relationship set *depositor* associates customers with accounts

\_ Widely used for database design

\_ Database design in E-R model usually converted to design in the relational model (coming up next) which is used for storage and processing.

This model is good to design a database, which can then be turned into tables in relational model(explained below).

Let's take an example, If we have to design a School Database, then **Student** will be an **entity** with **attributes** name, age, address etc. As **Address** is generally complex, it can be another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them.

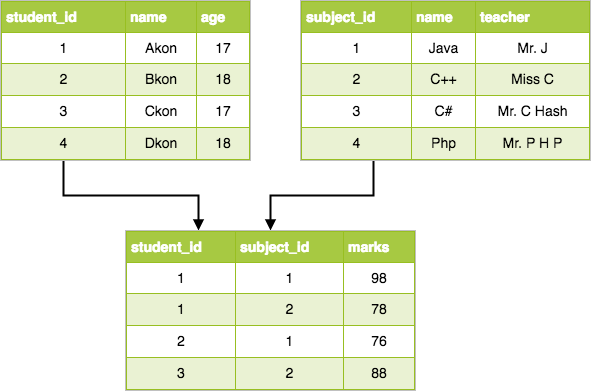


**Relational Model**

In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field. This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, we can say the only database model used around the world.

The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.

Hence, tables are also known as **relations** in relational model.





SQL | DDL, DML, DCL and TCL Commands

Structured Query Language(SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. SQL uses certain commands like Create, Drop, Insert etc. to carry out the required tasks.

These SQL commands are mainly categorized into four categories as discussed below:

1. **DDL(Data Definition Language) :**DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in database.

**Examples of DDL commands:**

* + [**CREATE**](https://www.geeksforgeeks.org/sql-create/) – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
  + [**DROP**](https://www.geeksforgeeks.org/sql-drop-truncate/) – is used to delete objects from the database.
  + [**ALTER**](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/)-is used to alter the structure of the database.
  + [**TRUNCATE**](https://www.geeksforgeeks.org/sql-drop-truncate/)–is used to remove all records from a table, including all spaces allocated for the records are removed.
  + [**COMMENT**](https://www.geeksforgeeks.org/sql-comments/) –is used to add comments to the data dictionary.
  + [**RENAME**](https://www.geeksforgeeks.org/sql-alter-rename/)–is used to rename an object existing in the database.

1. **DML(Data Manipulation Language) :**The SQL commands that deals with the manipulation of data present in database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

**Examples of DML:**

* + **[SELECT](https://www.geeksforgeeks.org/sql-select-clause/)** – is used to retrieve data from the a database.
  + **[INSERT](https://www.geeksforgeeks.org/sql-insert-statement/)** – is used to insert data into a table.
  + **[UPDATE](https://www.geeksforgeeks.org/sql-update-statement/)** – is used to update existing data within a table.
  + **[DELETE](https://www.geeksforgeeks.org/sql-delete-statement/)** – is used to delete records from a database table.

1. **DCL(Data Control Language) :**DCL includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.

**Examples of DCL commands:**

* + **GRANT**-gives user’s access privileges to database.
  + **REVOKE**-withdraw user’s access privileges given by using the GRANT command.

**Data Definition Language (DDL)**

\_ Specification notation for defining the database schema

\_ E.g.

**create table** *account* ( *account-number* **char**(10), *balance* **integer**)

\_ DDL compiler generates a set of tables stored in a *data dictionary*

\_ Data dictionary contains metadata (i.e., data about data)

\_ database schema

\_ Data *storage and definition* language

✔ language in which the storage structure and access methods used by the database system are specified

✔ Usually an extension of the data definition language

**Data Manipulation Language (DML)**

\_ Language for accessing and manipulating the data organized by the appropriate data model

\_ DML also known as query language

\_ Two classes of languages

\_ Procedural – user specifies what data is required and how to get those data

\_ Nonprocedural – user specifies what data is required without specifying how to get those data

\_ SQL is the most widely used query language

**Database Administrator**

\_ Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise’s information resources and needs.

\_ Database administrator's duties include:

\_ Schema definition

\_ Storage structure and access method definition

\_ Schema and physical organization modification

\_ Granting user authority to access the database

\_ Specifying integrity constraints

\_ Acting as liaison with users

\_ Monitoring performance and responding to changes in

requirements

**Database Users**

\_ Users are differentiated by the way they expect to interact with

the system

\_ Application programmers – interact with system through DML calls

\_ Sophisticated users – form requests in a database query language

\_ Specialized users – write specialized database applications that do not fit into the traditional data processing framework

\_ Naïve users – invoke one of the permanent application programs that have been written previously

\_ E.g. people accessing database over the web, bank tellers, clerical staff



# ER Model - Basic Concepts

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

## Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

## Attributes

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

### Types of Attributes

* **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
* **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first\_name and last\_name.
* **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average\_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data\_of\_birth.
* **Single-value attribute** − Single-value attributes contain single value. For example − Social\_Security\_Number.
* **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email\_address, etc.

These attribute types can come together in a way like −

* simple single-valued attributes
* simple multi-valued attributes
* composite single-valued attributes
* composite multi-valued attributes

### Entity-Set and Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, the roll\_number of a student makes him/her identifiable among students.

* **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
* **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.
* **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

## Relationship

The association among entities is called a relationship. For example, an employee **works\_at** a department, a student **enrolls** in a course. Here, Works\_at and Enrolls are called relationships.

### Relationship Set

A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.

### Degree of Relationship

The **number of participating entities** in a relationship defines the degree of the relationship.

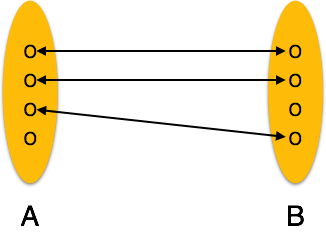
* Binary = degree 2
* Ternary = degree 3
* n-ary = degree

### Mapping Constraints :- An E-R scheme may define certain constraints to which the contents of a database must conform.

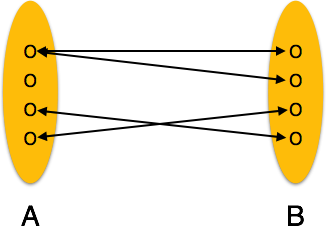
### Mapping Cardinalities

**Cardinality** defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

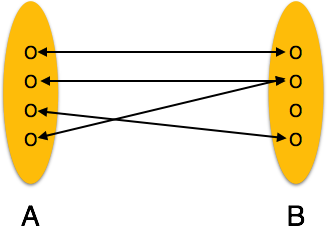
* **One-to-one** − One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



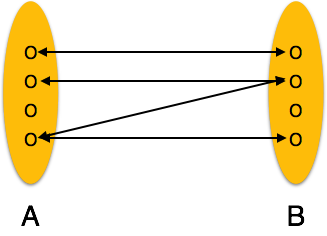
* **One-to-many** − One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



* **Many-to-one** − More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



* **Many-to-many** − One entity from A can be associated with more than one entity from B and vice versa.



**Existence Dependencies:** if the existence of entity X depends on the existence of entity Y, then X is said to be **existence dependent** on Y. (Or we say that Y is the **dominant** entity and X is the **subordinate** entity.)

For example,

* + Consider *account* and *transaction* entity sets, and a relationship *log* between them.
  + This is one-to-many from account to transaction.
  + If an *account* entity is deleted, its associated *transaction* entities must also be deleted.
  + Thus *account* is dominant and *transaction* is subordinate.

# ER Diagram Representation

Let us now learn how the ER Model is represented by means of an ER diagram. Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram.

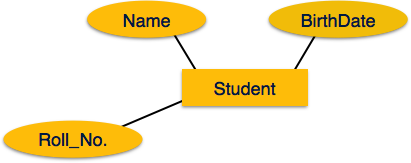
## Entity

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

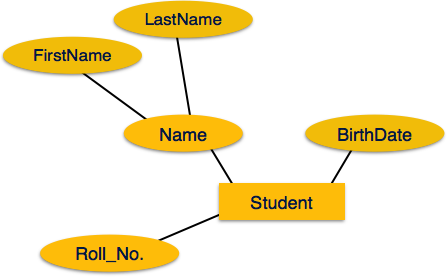
Entities in a school database

## Attributes

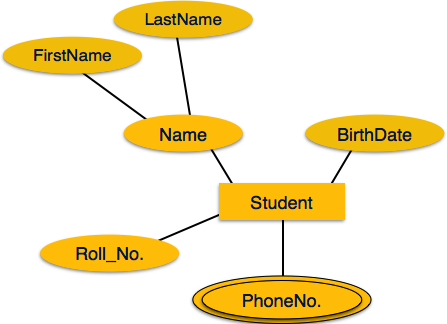
Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



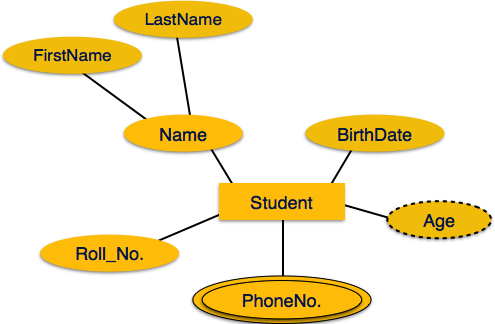
If the attributes are **composite**, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



**Multivalued** attributes are depicted by double ellipse.



**Derived** attributes are depicted by **dashed** ellipse.



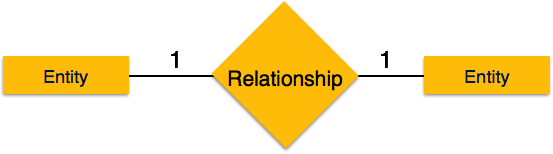
## Relationship

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

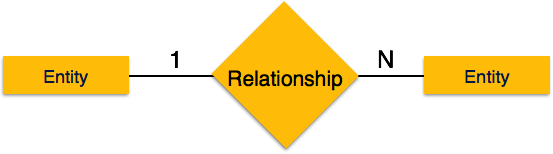
### Binary Relationship and Cardinality

A relationship where two entities are participating is called a **binary relationship**. Cardinality is the number of instance of an entity from a relation that can be associated with the relation.

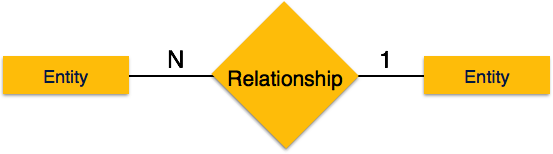
* **One-to-one** − When only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.



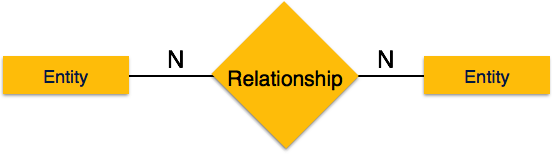
* **One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship.



* **Many-to-one** − When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship.

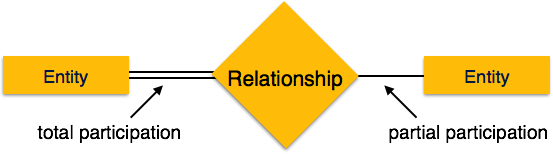


* **Many-to-many** − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.



### Participation Constraints

* **Total Participation** − Each entity is involved in the relationship. Total participation is represented by double lines.
* **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.



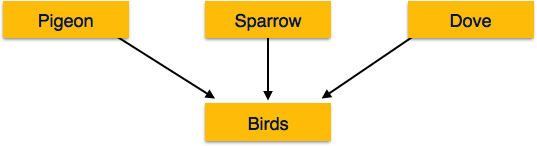
# Generalization Aggregation

The ER Model has the power of expressing database entities in a conceptual hierarchical manner. As the hierarchy goes up, it generalizes the view of entities, and as we go deep in the hierarchy, it gives us the detail of every entity included.

Going up in this structure is called **generalization**, where entities are clubbed together to represent a more generalized view. For example, a particular student named Mira can be generalized along with all the students. The entity shall be a student, and further, the student is a person. The reverse is called **specialization** where a person is a student, and that student is Mira.

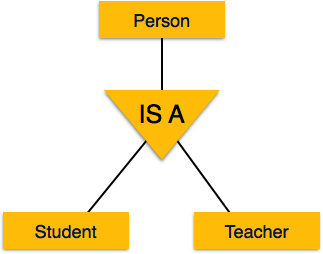
**Generalization**

As mentioned above, the process of generalizing entities, where the generalized entities contain the properties of all the generalized entities, is called generalization. In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.



**Specialization**

Specialization is the opposite of generalization. In specialization, a group of entities is divided into sub-groups based on their characteristics. Take a group ‘Person’ for example. A person has name, date of birth, gender, etc. These properties are common in all persons, human beings. But in a company, persons can be identified as employee, employer, customer, or vendor, based on what role they play in the company.

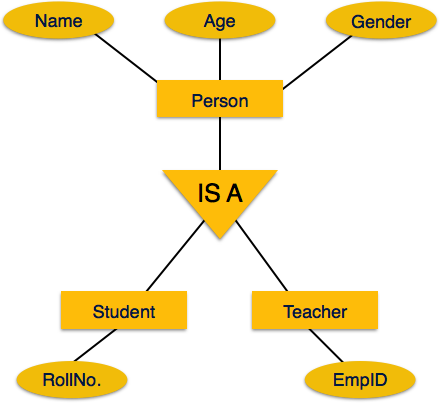


Similarly, in a school database, persons can be specialized as teacher, student, or a staff, based on what role they play in school as entities.

**Inheritance**

We use all the above features of ER-Model in order to create classes of objects in object-oriented programming. The details of entities are generally hidden from the user; this process known as **abstraction**.

Inheritance is an important feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.



For example, the attributes of a Person class such as name, age, and gender can be inherited by lower-level entities such as Student or Teacher.

# ER Model to Relational Model

ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand. ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram. We cannot import all the ER constraints into relational model, but an approximate schema can be generated.

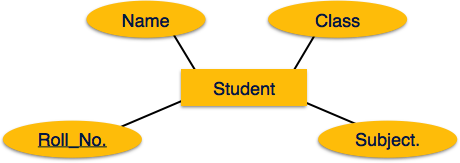
There are several processes and algorithms available to convert ER Diagrams into Relational Schema. Some of them are automated and some of them are manual. We may focus here on the mapping diagram contents to relational basics.

ER diagrams mainly comprise of −

* Entity and its attributes
* Relationship, which is association among entities.

## Mapping Entity

An entity is a real-world object with some attributes.

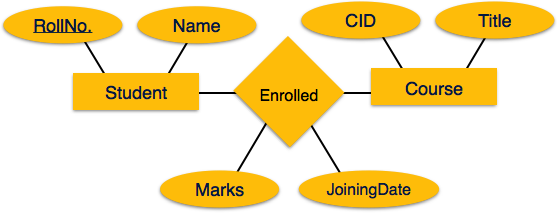


### Mapping Process (Algorithm)

* Create table for each entity.
* Entity's attributes should become fields of tables with their respective data types.
* Declare primary key.

## Mapping Relationship

A relationship is an association among entities.

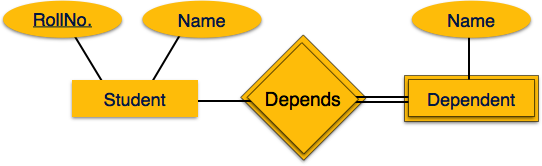


### Mapping Process

* Create table for a relationship.
* Add the primary keys of all participating Entities as fields of table with their respective data types.
* If relationship has any attribute, add each attribute as field of table.
* Declare a primary key composing all the primary keys of participating entities.
* Declare all foreign key constraints.

## Mapping Weak Entity Sets

A weak entity set is one which does not have any primary key associated with it.

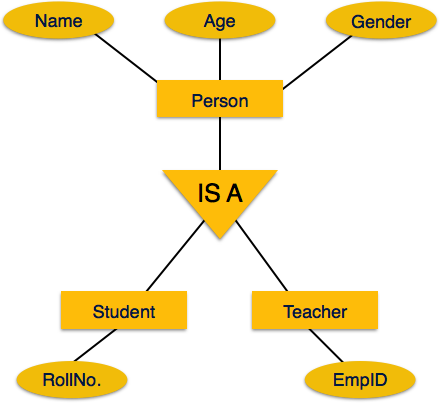


### Mapping Process

* Create table for weak entity set.
* Add all its attributes to table as field.
* Add the primary key of identifying entity set.
* Declare all foreign key constraints.

## Mapping Hierarchical Entities

ER specialization or generalization comes in the form of hierarchical entity sets.



### Mapping Process

* Create tables for all higher-level entities.
* Create tables for lower-level entities.
* Add primary keys of higher-level entities in the table of lower-level entities.
* In lower-level tables, add all other attributes of lower-level entities.
* Declare primary key of higher-level table and the primary key for lower-level table.
* Declare foreign key constraints.