19CS402	DATABASE MANAGEMENT SYSTEMS

Unit -I Notes

UNIT-I	INTRODUCTION	9
Purpose of Database System -– View of data – Data models, Database Management system - Three-schema architecture of DBMS, Components of DBMS. Entity –Relationship Model - Conceptual data modeling - motivation, entities, entity types, attributes, relationships, relationship types, E/R diagram notations, Examples		

DBMS:

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

Purpose of Database System:

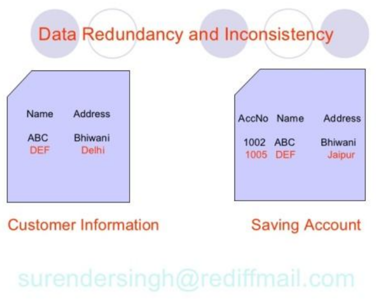
The typical file processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files.Keeping organizational information in a file-processing system has a number of major disadvantages:

Data redundancy and inconsistency:

In file processing, every user group maintains its own files for handling its data processing applications.

Example:

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:

File processing environments do not allow needed data to be retrieved in a convenient and efficient manner.

Example:

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 now has 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

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.

Example:

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 theconsistent state that existed prior to the failure. Consider a banking system with a program to transfer $500 from account A to account B. If a system failure occurs during the execution of the program, it is possible that the $500 was removed from the balance of account A but was not credited to the balance of account 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.

In such an environment, interaction of concurrent updates is possible and may result in inconsistent data.

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.

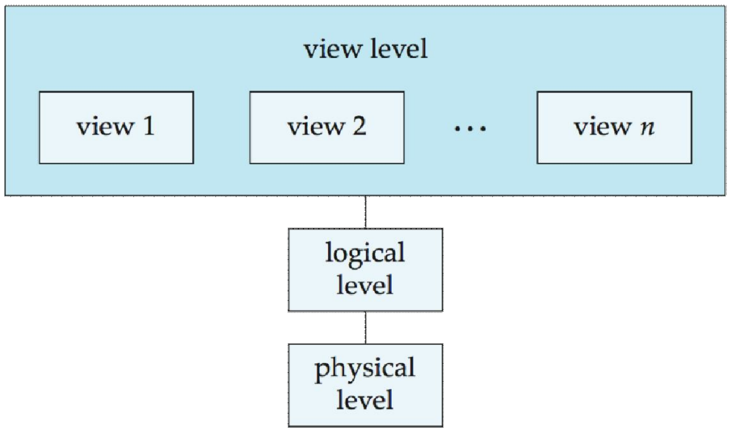
Security problems:

Enforcing security constraints to the file processing system is difficult

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.

View of data:

A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data. A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained.



Physical level: The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.

Logical level: describes data stored in database, and the relationships among the data.. The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. This is referred to as physical data independence. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction

View level: The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database

Data Models:

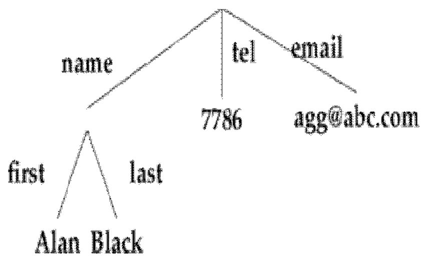
Underlying the structure of a database is the data model: a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. There are a number of different data models that we shall cover in the text. The data models can be classified into four different categories:

• **Relational Model**. The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations. The relational model is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.

- Advantage :
  - Structure independence
  - Conceptual (Theoretical) Simplicity
  - A powerful database management system
- Disadvantage
  - Transaction Process is not efficient
  - Processing time is low

• **Entity-Relationship Model.** The entity-relationship (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects. An entity is a “thing” or “object” in the real world that is distinguishable from other objects. The entity-relationship model is widely used in database design.

• **Semi-structured Data Model**. The semi-structured data model permits the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. JSON and Extensible Markup Language (XML) are widely used semi-structured data representations.



Object-Based Data Model.: Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology. This led initially to the development of a distinct object-oriented data model, but today the concept of objects is well integrated into relational databases. Standards exist to store objects in relational tables.

Database systems allow procedures to be stored in the database system and executed by the database system. This can be seen as extending the relational model with notions of encapsulation, methods, and object identity.

- Advantage :
  - Exceptional conceptual simplicity
  - Visual representation
  - Effective communication tool
  - Integrated with the relational database model
- Disadvantage
  - Limited constraint representation
  - Limited relationship representation
  - No data manipulation language
  - Loss of information content

Database Management system:

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases can be very large.
- Databases touch all aspects of our lives

Three-schema architecture of DBMS:

The three-schema architecture divides the database into three-level used to create a separation between the physical database and the user application. In simple terms, this architecture hides the details of physical storage from the user.

The database administrator (DBA) responsible is to change the structure of database storage without affecting the user's view. It deals with the data, the relationship between them and the different access methods implemented on the database. The logical design of database is called a schema

This architecture contains three layers of database management system, which are as follows −

- External level
- Conceptual level

- Internal level

External/ View level

This is the highest level of database abstraction. It includes a number of external schemas or user views. This level provides different views of the same database for a specific user or a group of users. An external view provides a powerful and flexible security mechanism by hiding the parts of the database from a particular user.

Conceptual or Logical level

This level describes the structure of the whole database. It acts as a middle layer between the physical storage and user view. It explains what data to be stored in the database, what the data types are, and what relationship exists among those data. There is only one conceptual schema per database.

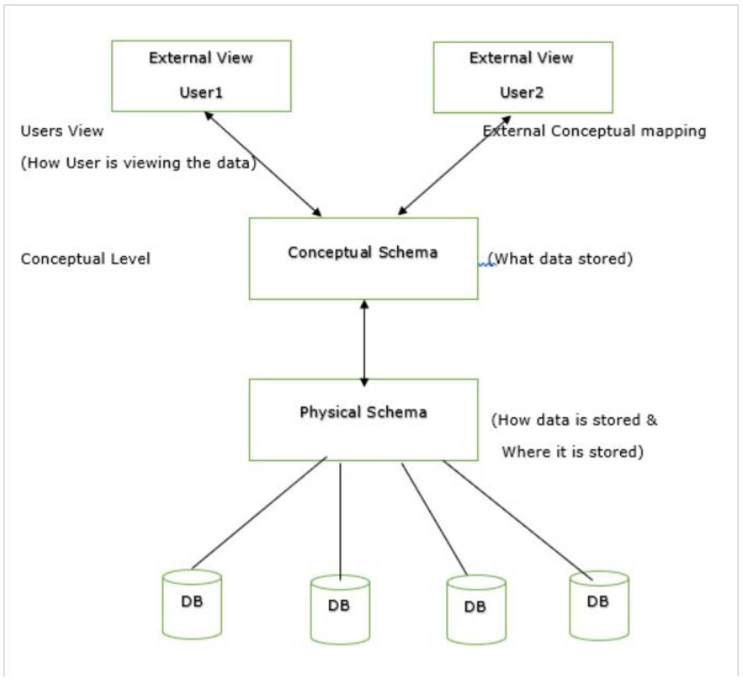
This level describes the structure of the whole database. It acts as a middle layer between the physical storage and user view. It explains what data to be stored in the database, what the data types are, and what relationship exists among those data. There is only one conceptual schema per database.

Internal or Physical level

This is the lowest level of database abstraction. It describes how the data is stored in the database and provides the methods to access data from the database. It allows viewing the physicalrepresentation of the database on the computer system.

The interface between the conceptual and internal schema identifies how an element in the conceptual schema is stored and how it may be accessed. It is one which is closest to physical storage. The internal schema not only defines different stored record types, but also specifies what indices exist, how stored fields are represented.

The three level schema architecture in DBMS is given below –



Components of DBMS:

- User Interface
- Data Manager
- File Manager
- Disk Manager
- Physical Database

User Interface:

- The user interface is the aggregate of means by which the people –the user interacts with the system a particular machine, device, computer program or other complex tools.
- The user interface provides the means of:

  -Input, allowing the users to manipulate the system.

  -Output, allowing the system to produce the effects of the users manipulation.

- It refers to the graphical, textual and auditory information the programme presents to the user and the control sequences the user employs to the program.

Data Manager:

- It is a program which allows you to process and manipulate your data in a easy and logical manner using a graphical interface.
- Data Manager reads and writes delimited files such as comma separated files (CSV) and also can read data from ODBC Data Sources.
- It allows you to construct a conceptual design on how you are going to process your data and transform it into another form.
- You form your design by adding functional nodes and linking them such that the links form the data flow through nodes on a graphical work area.
- You form your design by adding functional nodes and linking them such that the links form the data flow through nodes on a graphical work area.
- Each node performs a single function on your data, once it completes it passes your data to the node it is linked to and the process continues until the data encounters a output node.
- You can form a simple design or a complicated design with hundreds of nodes and multiple input and output nodes.

File Manager:

- A file manager or file browser is a computer program that provides a user interface to work with file systems.
- They are very useful for speeding up interaction with files
- The most common operations on files are create, open, edit, view, print, play, rename, move, copy, delete, attributes, properties, search/find, and permissions.

- File managers may contain features inspired by web browsers, including forward and back navigational buttons.
- File managers also provide the ability to extend operations using user written scripts.

It passes request to disk manager.

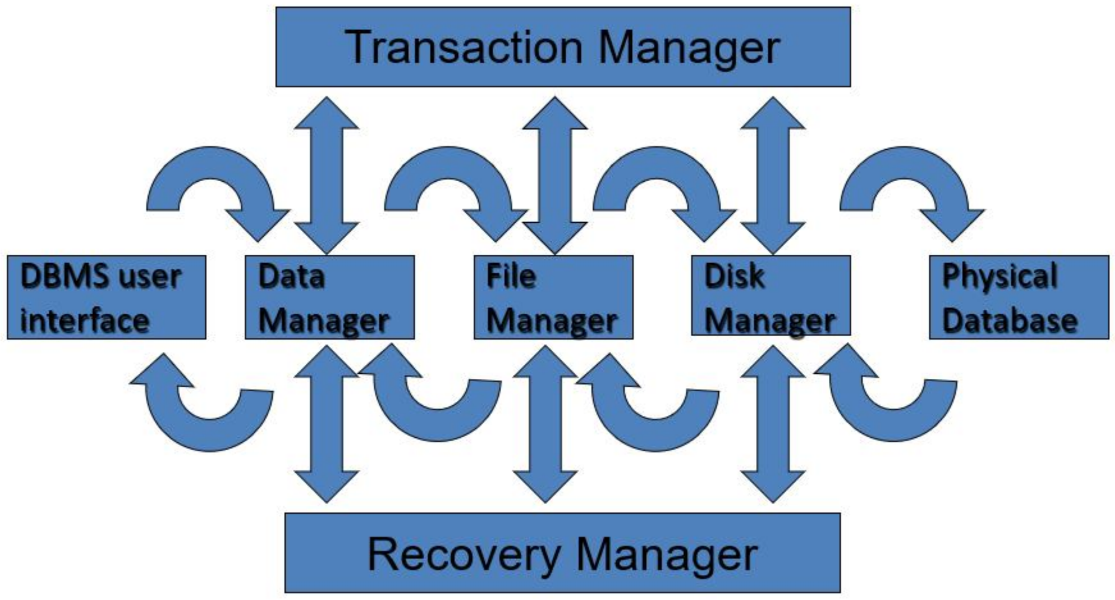
Disk Manager:

- Disk manager is a simple filesystem configurator that allows you to:

  -Automatically detect new partitions at startup.

  -Fully manage configuration of filesystem.
- Disk Manager logs every change you make to the filesystem configuration
- explaining hardware concepts
- documenting switches of many of the existing disks
- putting into place custom software drivers, notably those related to maximum disk or partition size providing testing and informational utilities

Interaction of DBMS components:



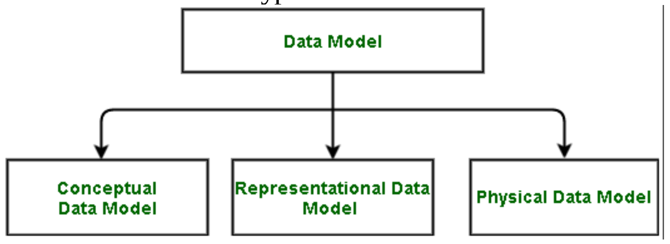
- The user requests for specific information with the help of user interface.
- This request is processed by data manager and after processing ,data manager request for specific records to the file manager.
- The file manager then request for the specific block to the disk manager.
- The disk manager then then retrives the block and sends it to file manager,which sends the required record to data manager.

- The transaction manager supervises the data transactions that is carried out between the data manager, file manager, and the disk manager.
- The recovery manager keeps a check on the transacted data so that in case of system failure, the data can be protected.

The Entity-Relationship Model:

Conceptual data modelling:

A **Data Model** in Database Management System (DBMS), is the concept of tools that are developed to summarize the description of the database.
It is classified into 3 types:



Conceptual data model:

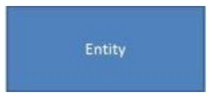
Conceptual data model describes the database at a very high level and is useful to understand the needs or requirements of the database. It is this model, that is used in the requirement gathering process i.e., before the Database Designers start making a particular database. One such popular model is the entity/relationship model (ER model). The E/R model specializes in entities, relationships and even attributes which are used by the database designers. In terms of this concept, a discussion can be made even with non-computer science (non-technical) users and stakeholders, and their requirements can be understood.

The Entity-Relationship Model:

The entity-relationship (E-R) data model was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database.The E-R model is very useful in mapping the meanings and interactions of realworld enterprises onto a conceptual schema.

The E-R data model employs three basic concepts: **entity sets, relationship sets, and attributes**

A graphical technique for understanding and organizing the data independent of the actual database implementation. We need to be familiar with the following terms to go further.
Entity
Any thing that has an independent existence and about which we collect data. It is also known as entity type. In ER modelling, notation for entity is given below



Entity instance:

Entity instance is a particular member of the entity type.

Example for entity instance : A particular employee

Regular Entity:

An entity which has its own key attribute is a regular entity.

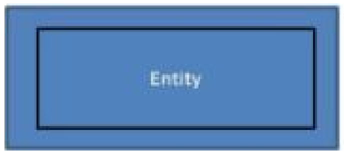
Example for regular entity : Employee

Weak entity:

An entity which depends on other entity for its existence and doesn't have any key attribute of its own is a weak entity.

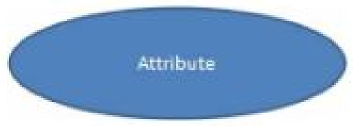
Example for a weak entity : In a parent/child relationship, a parent is considered as a strong entity and the child is a weak entity.

In ER modeling, notation for weak entity is given below



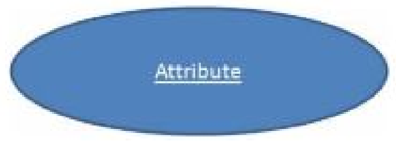
Attributes:

Properties/characteristics which describe entities are called attributes.
In ER modeling, notation for attribute is given below



Domain of Attributes
The set of possible values that an attribute can take is called the domain of the attribute. For example, the attribute day may take any value from the set {Monday, Tuesday ... Friday}. Hence this set can be termed as the domain of the attribute day.

Key attribute
The attribute (or combination of attributes) which is unique for every entity instance is called key attribute.E.g the employee_id of an employee, pan_card_number of a person etc.If the key attribute consists of two or more attributes in combination, it is called a composite key.
In ER modeling, notation for key attribute is given below



Simple attribute
If an attribute cannot be divided into simpler components, it is a simple attribute.
Example for simple attribute : employee_id of an employee

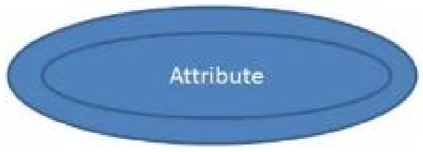
Composite attribute
If an attribute can be split into components, it is called a composite attribute.
Example for composite attribute : Name of the employee which can be split into First_name, Middle_name, and Last_name.

Single valued Attributes

If an attribute can take only a single value for each entity instance, it is a single valued attribute.
example for single valued attribute : age of a student. It can take only one value for a particular student

Multi-valued Attributes

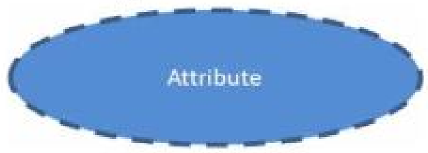
If an attribute can take more than one value for each entity instance, it is a multi-valued attribute.
Multi-valued
example for multi valued attribute : telephone number of an employee, a particular employee may have multiple telephone numbers.
In ER modeling, notation for multi-valued attribute is given below



Stored Attribute
An attribute which need to be stored permanently is a stored attribute
Example for stored attribute : name of a student.

Derived Attribute

An attribute which can be calculated or derived based on other attributes is a derived attribute.
Example for derived attribute : age of employee which can be calculated from date of birth and current date.
In ER modeling, notation for derived attribute is given below.



Entity Sets:

An entity is a “thing” or “object” in the real world that is distinguishable from all other objects.

For example, each person in a university is an entity. An entity has a set of properties, and the values for some set of properties must uniquely identify an entity. For instance, a person may have a person id property whose value uniquely identifies that person.

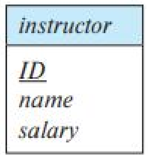
An entity set is a set of entities of the same type that share the same properties, or attributes.

The set of all people who are instructors at a given university, for example, can be defined as the entity set instructor. Similarly, the entity set student might represent the set of all students in the university.

An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set

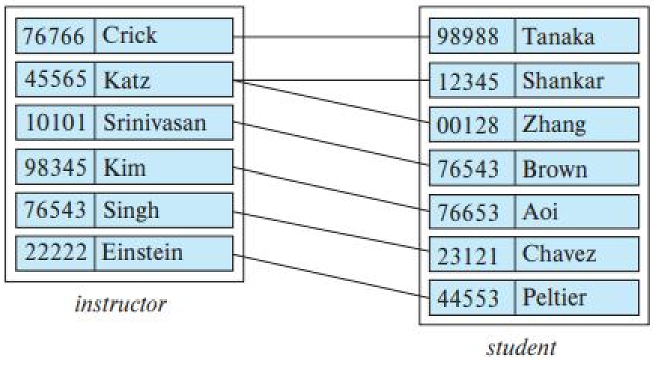
Possible attributes of the instructor entity set are ID, name, dept name, and salary

An entity set is represented in an E-R diagram by a rectangle, which is divided into two parts. The first part, contains the name of the entity set. The second part contains the names of all the attributes of the entity set.

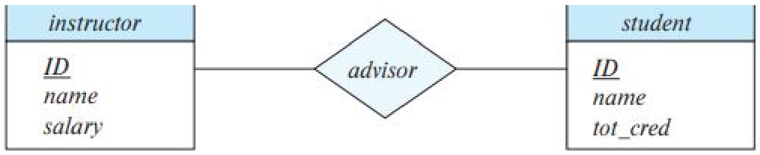


Relationship Sets:

A relationship is an association among several entities. For example, we can define a relationship advisor that associates instructor Katz with student Shankar. This relationship specifies that Katz is an advisor to student Shankar. A relationship set is a set of relationships of the same type.Consider two entity sets instructor and student. We define the relationship set advisor to denote the associations between students and the instructors who act as their advisors.



A relationship instance in an E-R schema represents an association between the named entities in the real-world enterprise that is being modeled. As an illustration, the individual instructor entity Katz, who has instructor ID 45565, and the student entity Shankar, who has student ID 12345, participate in a relationship instance of advisor.This relationship instance represents that in the university, the instructor Katz is advising student Shankar.A relationship set is represented in an E-R diagram by a diamond, which is linked via lines to a number of different entity sets (rectangles).



It Shows two entity sets instructor and student, related through a **binary relationship set** advisor.

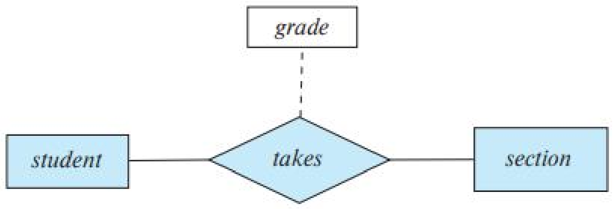
Formally, a relationship set is a mathematical relation on n ≥ 2 (possibly nondistinct) entity sets. If E1, E2, …, En are entity sets, then a relationship set R is a subset of

$$\{(e1, e2, \ldots, en) \mid e1 \in E1, e2 \in E2, \ldots, en \in En\}$$

where (e1, e2, …, en) is a relationship instance.

The association between entity sets is referred to as participation; i.e., the entity sets E1, E2, …, En participate in relationship set R.

A relationship may also have attributes called descriptive attributes. As an example of descriptive attributes for relationships, consider the relationship set takes which relates entity sets student and section. We may wish to store a descriptive attribute grade with the relationship to record the grade that a student received in a course offering.



It is possible to have more than one relationship set involving the same entity sets.

For example, suppose that students may be teaching assistants for a course. Then, the entity sets section and student may participate in a relationship set teaching assistant, in addition to participating in the takes relationship set

It is possible to have more than one relationship set involving the same entity sets

The relationship sets advisor and takes provide examples of a binary relationship set—that is, one that involves two entity sets. Most of the relationship sets in a database system are binary. Occasionally, however, relationship sets involve more than two entity sets. The number of entity sets that participate in a relationship set is the degree of the relationship set. A binary relationship set is of degree 2; a ternary relationship set is of degree 3

Cardinality of a Relationship:

Relationship cardinalities specify how many of each entity type is allowed. Relationships can have four possible connectivities as given below

1.One to one (1:1) relationship

2.One to Many (1:N) relationship

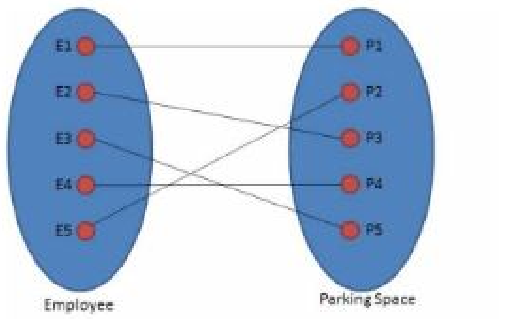
3. Many to one (M:1) relationship

4. Many to many (M:N) relationship

The minimum and maximum values of this connectivity is called the cardinality of the relationship

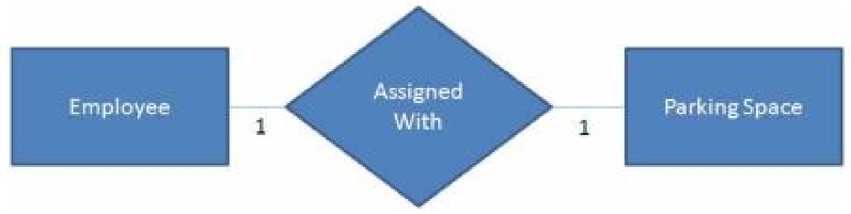
Example for Cardinality – One-to-One (1:1)

Employee is assigned with a parking space.



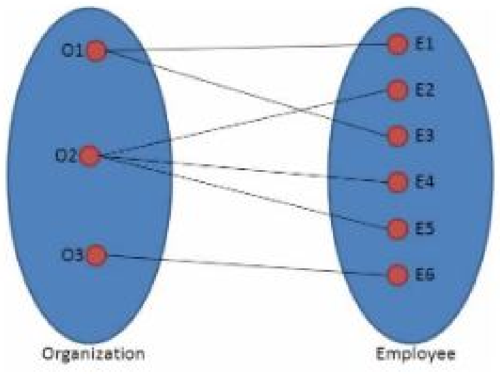
One employee is assigned with only one parking space and one parking space is assigned to only one employee. Hence it is a 1:1 relationship and cardinality is One-To-One (1:1)

In ER modeling, this can be mentioned using notations as given below



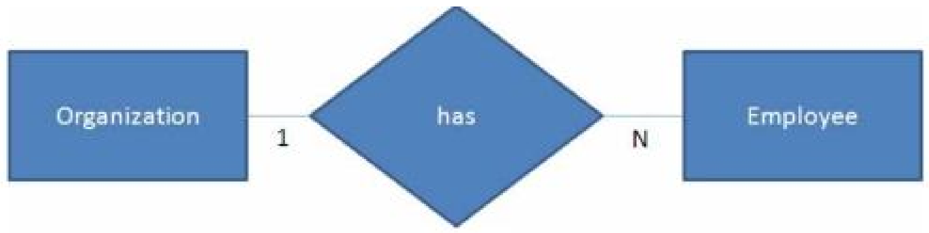
Example for Cardinality – One-to-Many (1:N)

Organization has employees



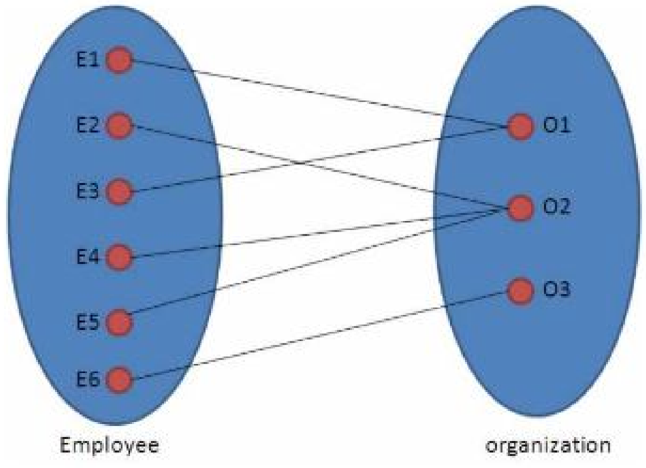
One organization can have many employees , but one employee works in only one organization. Hence it is a 1:N relationship and cardinality is One-To-Many (1:N)

In ER modeling, this can be mentioned using notations as given below



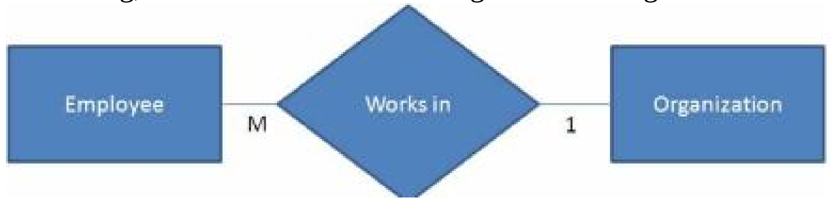
Example for Cardinality – Many-to-One (M :1)

It is the reverse of the One to Many relationship. employee works in organization



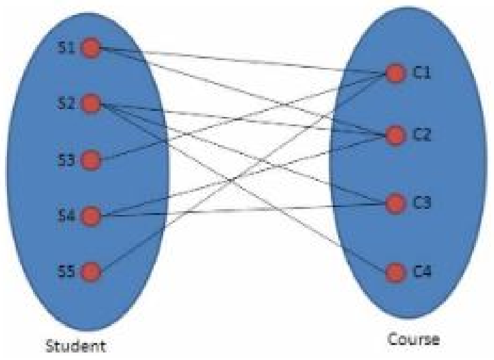
One employee works in only one organization But one organization can have many employees. Hence it is a M:1 relationship and cardinality is Many-to-One (M :1)

In ER modeling, this can be mentioned using notations as given below:

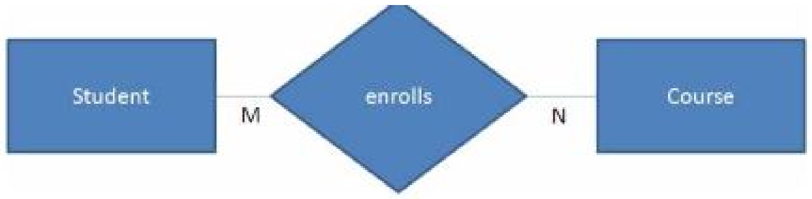


Cardinality – Many-to-Many (M:N)

Students enrolls for courses



One student can enroll for many courses and one course can be enrolled by many students. Hence it is a M:N relationship and cardinality is Many-to-Many (M:N)
In ER modeling, this can be mentioned using notations as given below



Relationship Participation:
1. Total
In total participation, every entity instance will be connected through the relationship to another instance of the other participating entity types
2. Partial
Example for relationship participation
Consider the relationship - Employee is head of the department.
Here all employees will not be the head of the department. Only one employee will be the head of the department. In other words, only few instances of employee entity participate in the above relationship. So employee entity's participation is partial in the said relationship.
However each department will be headed by some employee. So department entity's participation is total in the said relationship.

Advantages and Disadvantages of ER Modeling (Merits and Demerits of ER Modeling)

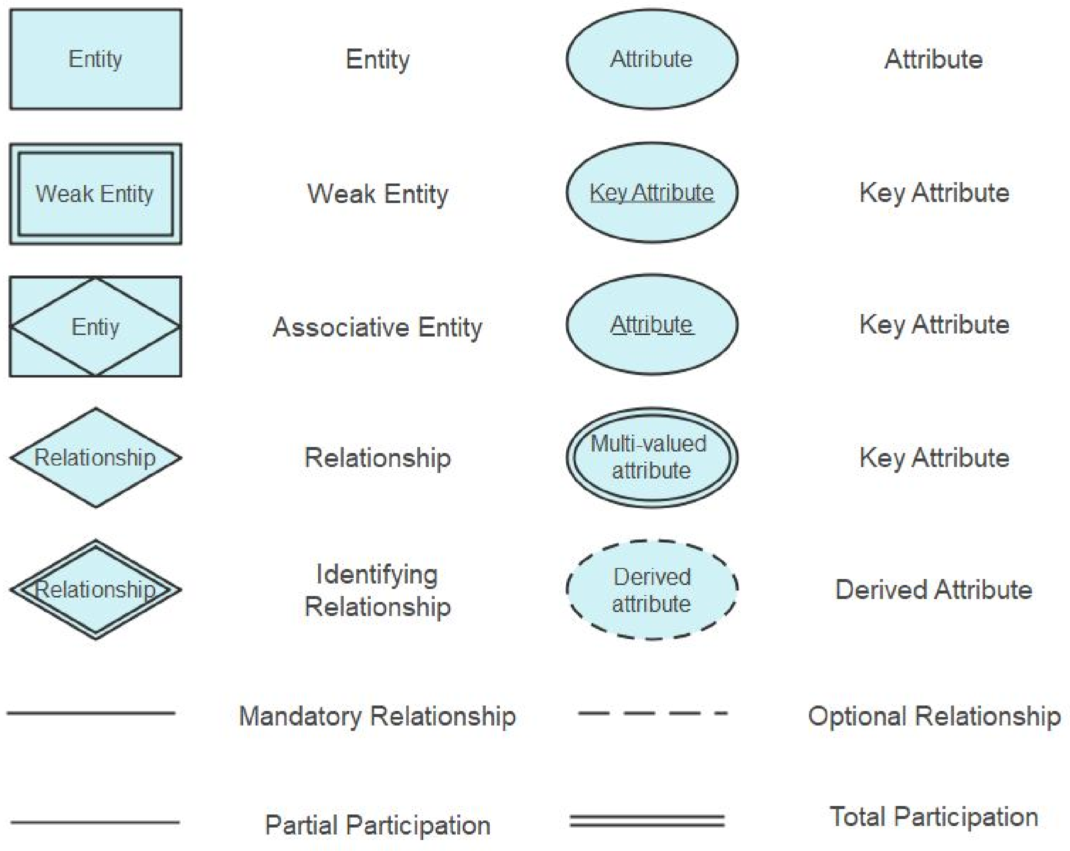
Advantages

1. ER Modelling is simple and easily understandable. It is represented in business users language and it can be understood by non-technical specialist.
2. Intuitive and helps in Physical Database creation.
3. Can be generalized and specialized based on needs.
4. Can help in database design.
5. Gives a higher level description of the system.

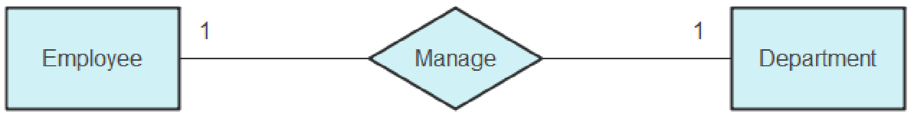
Disadvantages

1. Physical design derived from E-R Model may have some amount of ambiguities or inconsistency.
2. Sometime diagrams may lead to misinterpretations.

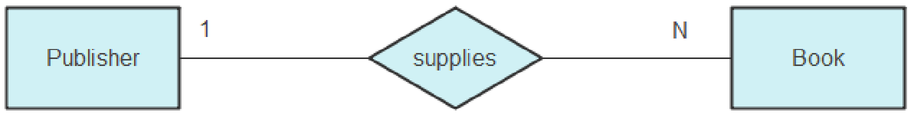
E R Diagram Notation:



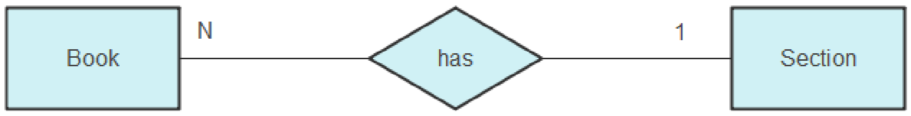
one-to-one (1:1)



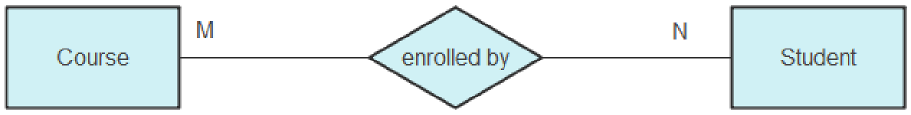
one-to-many (1:N)



many-to-one (N:1)



many-to-many (M:N)



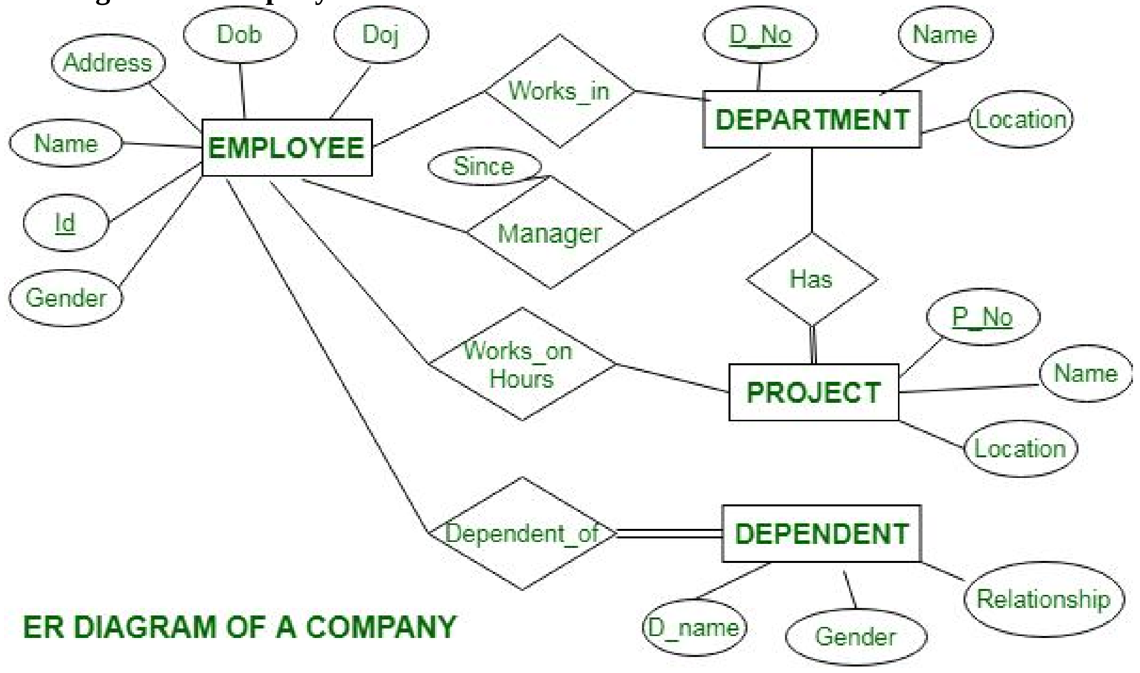
Example of ER Diagram:

ER Diagram is known as Entity-Relationship Diagram, it is used to analyze to structure of the Database. It shows relationships between entities and their attributes. An ER Model provides a means of communication.

ER diagram of Company has the following description :

- Company has several departments.
- Each department may have several Location.
- Departments are identified by a name, D_no, Location.
- A Manager control a particular department.
- Each department is associated with number of projects.
- Employees are identified by name, id, address, dob, dat e_of_joining.
- An employee works in only one department but can work on several project.
- We also keep track of number of hours worked by an employee on a single project.
- Each employee has dependent
- Dependent has D_name, Gender and relationship.

ER Diagram of Company :


Dob
Doj
Address
Name
EMPLOYEE
Id
Gender
Works_in
D_No
Name
DEPARTMENT
Location
Since
Manager
Has
P_No
Works_on
Hours
Name
PROJECT
Location
Dependent_of
DEPENDENT
Relationship
ER DIAGRAM OF A COMPANY
D_name
Gender