UNIT – I INTRODUCTION

1.1 File Processing System

A file processing system is a collection of files and programs that access/modify these files. Typically, new files and programs are added over time (by different programmers) as new information needs to be stored and new ways to access information are needed.

Problems with file processing systems:

• data redundancy and inconsistency

• difficulty of accessing data

• problems with concurrent access

Disadvantages of File Processing System

Duplicate Data

Data is stored more than once in different files, that means duplicate data may occur in all these files. Since all the files are independent on each other so it is very difficult to overcome this error and if anyone finds this error then it will take time and effort to solve this issue.

For Example: A student is having record in college library and in Examination department. Then his name, roll number, fathers name and class will be same in both the departments. Also these departments are not dependent on each other. So it create lots of duplicates value about that student and when he needs any change for his name or class then he has to go to both the departments to make these changes happen otherwise it will create problem for him.

Inconsistency

In file processing system, various copies of same data may contain different values. Data is not consistent in this system, it means if a data item needs to be changed then all the files containing that data need to be modified. It may create a risk of out dated values of data.

For Example: If you change student name in library then his name should be changed in all the departments related to the student.

Accessing Anomalies

Accessing anomalies means that it is not easy to access data in a desired or efficient way. It makes supervision of department very difficult. If a user wants information in a specific manner then he requires creating a program for it.

For Example: Let’s say , if admin of the college wants any student information like his name, fathers name, roll number, marks and class then program for it is written but if he wants records of whose students whose numbers are more than 80 percent then he require to create a different program for it.

Poor Data Integrity

A collection of data is integrated if it meets certain consistency constraints. A programmer always puts these constraints in the programs by adding some codes. In File Processing System, poor data integrity often arises and it becomes very difficult to add new constraints at that time.

For Example: The maximum marks of the student can never be more than 100.

Poor Data Security

Poor data security is the most threatening problem in File Processing System. There is very less security in File Processing System as anyone can easily modify and change the data stored in the files. All the users must have some restriction of accessing data up to a level.

For Example: If a student can access his data in the college library then he can easily change books issued date. Also he can change his fine detains to zero.

Atomicity Problem

Atomicity is required to save the data values, it means that information is completely entered or canceled at all. Any system may fail at any time and at that time it is desired that data should be in a consistent state.

For Example: If you are buying a ticket from railway and you are in the process of money transaction. Suddenly, your internet got disconnected then you may or may not have paid for the ticket. If you have paid then your ticket will be booked and if not then you will not be charged anything. That is called consistent state, means you have paid or not.

Same atomicity is not present in File Processing System.

Wastage of Labor and Space

Labor is very costly in this era and no organization can afford wastage of their precious labor. File Processing System needs lots of copied data in different files that cause wastage of labor. Also maintaining same data again and again leads to wastage of space too.

For Example: Maintaining student's record in many departments that are not dependent on each other cause wastage of labor and space.

Data Isolation

Data is isolated in File Processing System and data is stored in different files. These files can be in different formats. If you want to extract data from two file then you are required to which part of the file is needed and how they are related to each other.

But still in spite of so many disadvantages, File Processing System is still good for small organizations because it does not require costly softwares and programmers to handle it.

1.2 Difference Between File Processing System and DBMS

File Processing System	Database Management System
On the other hand, a file system is a more unstructured data store for storing arbitrary, probably unrelated data.	A database is generally used for storing related, structured data, with well defined data formats, in an efficient manner for insert, update and/or retrieval.
A file system provides much looser guarantees about consistency, isolation and durability.	While databases are consistent at any instant in time,provide,isolated transactions and durable writes,
A file-processing system coordinates only the physical access to the data.	A database management system coordinates both the physical and the logical access to the data
A file-processing system is designed to allow predetermined access to data (i.e. compiled programs)	A database management system is designed to allow flexible access to data (i.e. queries)
Unauthorized access cannot be restricted in DBMS	Unauthorized access is restricted in DBMS
Redundancy cannot be controlled in File processing system.	Redundancy can be controlled in File processing system.

1.3 Advantages of Database Management System:

The DBMS has a number of advantages as compared to traditional computer file processing approach. The DBA must keep in mind these benefits or capabilities during designing databases, coordinating and monitoring the DBMS.

The major advantages of DBMS are described below.

1. Controlling Data Redundancy:

In non-database systems (traditional computer file processing), each application program has its own files. In this case, the duplicated copies of the same data are created at many places. In DBMS, all the data of an organization is integrated into a single database. The data is recorded at only one place in the database and it is not duplicated. For example, the dean's faculty file and the faculty payroll file contain several items that are identical. When they are converted into database, the data is integrated into a single database so that multiple copies of the same data are reduced to-single copy.

2. Data Consistency:

By controlling the data redundancy, the data consistency is obtained. If a data item appears only once, any update to its value has to be performed only once and the updated value (new value of item) is immediately available to all users.

If the DBMS has reduced redundancy to a minimum level, the database system enforces consistency. It means that when a data item appears more than once in the database and is updated, the DBMS automatically updates each occurrence of a data item in the database.

3. Data Sharing:

In DBMS, data can be shared by authorized users of the organization. The DBA manages the data and gives rights to users to access the data. Many users can be authorized to access the same set of information simultaneously. The remote users can also share same data. Similarly, the data of same database can be shared between different application programs.

4. Data Integration:

In DBMS, data in database is stored in tables. A single database contains multiple tables and relationships can be created between tables (or associated data entities). This makes easy to retrieve and update data.

5. Integrity Constraints:

Integrity constraints or consistency rules can be applied to database so that the correct data can be entered into database. The constraints may be applied to data item within a single record or they may be applied to relationships between records.

Examples:

The examples of integrity constraints are:

(i) 'Issue Date' in a library system cannot be later than the corresponding 'Return Date' of a book.

(ii) Maximum obtained marks in a subject cannot exceed 100.

For example, when you draw amount from the bank through ATM card, then your account balance is compared with the amount you are drawing. If the amount in your account balance is less than the amount you want to draw, then a message is displayed on the screen to inform you about your account balance.

6. Data Security:

Data security is the protection of the database from unauthorized users. Only the authorized persons are allowed to access the database. Some of the users may be allowed to access only a part of database i.e., the data that is related to them or related to their department. Mostly, the DBA or head of a department can access all the data in the database. Some users may be permitted only to retrieve data, whereas others are allowed to retrieve as well as to update data.

The database access is controlled by the DBA. He creates the accounts of users and gives rights to access the database. Typically, users or group of users are given usernames protected by passwords.

7. Data Atomicity:

A transaction in commercial databases is referred to as atomic unit of work. For example, when you purchase something from a point of sale (POS) terminal, a number of tasks are performed such as;

Company stock is updated.

Amount is added in company's account.

Sales person's commission increases etc.

All these tasks collectively are called an atomic unit of work or transaction. These tasks must be completed in all; otherwise partially completed tasks are rolled back. Thus through DBMS, it is ensured that only consistent data exists within the database.

8. Database Access Language:

Most of the DBMSs provide SQL as standard database access language. It is used to access data from multiple tables of a database.

9. Development of Application:

The cost and time for developing new applications is also reduced. The DBMS provides tools that can be used to develop application programs. For example, some wizards are available to generate Forms and Reports. Stored procedures (stored on server side) also reduce the size of application programs.

10. Creating Forms:

Form is very important object of DBMS. You can create Forms very easily and quickly in DBMS, Once a Form is created, it can be used many times and it can be modified very easily. The created Forms are also saved along with database and behave like a software component. A Form provides very easy way (user-friendly interface) to enter data into database, edit data, and display data from database. The non-technical users can also perform various operations on databases through Forms without going into the technical details of a database.

11. Report Writers:

Most of the DBMSs provide the report writer tools used to create reports. The users can create reports very easily and quickly. Once a report is created, it can be used many times and it can be modified very easily. The created reports are also saved along with database and behave like a software component.

12. Control over Concurrency:

In a computer file-based system, if two users are allowed to access data simultaneously, it is possible that they will interfere with each other. For example, if both users attempt to perform update operation on the same record, then one may overwrite the values recorded by the other.

13. Backup and Recovery Procedures:

In a computer file-based system, the user creates the backup of data regularly to protect the valuable data from damaging due to failures to the computer system or application program. It is a time consuming method, if volume of data is large. Most of the DBMSs provide the 'backup and recovery' sub-systems that automatically create the backup of data and restore data if required. For example, if the computer system fails in the middle (or end) of an update operation of the program, the recovery sub-system is responsible for making sure that the database is restored to the state it was in before the program started executing.

14. Data Independence:

The separation of data structure of database from the application program that is used to access data from database is called data independence. In DBMS, database and application programs are separated from each other. The DBMS sits in between them. You can easily change the structure of database without modifying the application program. For example you can modify the size or data type of a data items (fields of a database table).

15. Advanced Capabilities:

DBMS also provides advance capabilities for online access and reporting of data through Internet. Today, most of the database systems are online. The database technology is used in conjunction with Internet technology to access data on the web servers.

1.4 Disadvantages of Database Management System (DBMS):

Although there are many advantages but the DBMS may also have some minor **disadvantages**. These are:

1. Cost of Hardware & Software:

A processor with high speed of data processing and memory of large size is required to run the DBMS software. It means that you have to upgrade the hardware used for file-based system. Similarly, DBMS software is also Very costly.

2. Cost of Data Conversion:

When a computer file-based system is replaced with a database system, the data stored into data file must be converted to database files. It is difficult and time consuming method to convert data of data files into database. You have to hire DBA (or database designer) and system designer along with application programmers; alternatively, you have to take the services of some software houses. So a lot of money has to be paid for developing database and related software.

3. Cost of Staff Training:

Most DBMSs are often complex systems so the training for users to use the DBMS is required. Training is required at all levels, including programming, application development, and database administration. The organization has to pay a lot of amount on the training of staff to run the DBMS.

4. Appointing Technical Staff:

The trained technical persons such as database administrator and application programmers etc are required to handle the DBMS. You have to pay handsome salaries to these persons. Therefore, the system cost increases.

5. Database Failures:

In most of the organizations, all data is integrated into a single database. If database is corrupted due to power failure or it is corrupted on the storage media, then our valuable data may be lost or whole system stops.

What is Data?

Data is any set of characters that has been gathered and translated for some purpose, usually analysis. It can be any character, including text and numbers, pictures, sound, or video. If data is not put into context, it doesn't do anything to a human or computer.

What is Database?

A database is a collection of information that is organized so that it can be easily accessed, managed and updated.

Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases process workloads to create and update themselves, querying the data they contain and running applications against it.

What is DBMS?

Defn 1: Database Management System (DBMS) is a collection of programs which enables its users to access database, manipulate data, reporting / representation of data.

It also helps to control access to the database.

Defn 2: A database management system (DBMS) is a software package designed to define, manipulate, retrieve and manage data in a database. A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure. It also defines rules to validate and manipulate this data.

1.5 DBMS - Data Model:

Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

The very first data model could be flat data-models, where all the data used are to be kept in the same plane. Earlier data models were not so scientific, hence they were prone to introduce lots of duplication and update anomalies.

Entity-Relationship Model

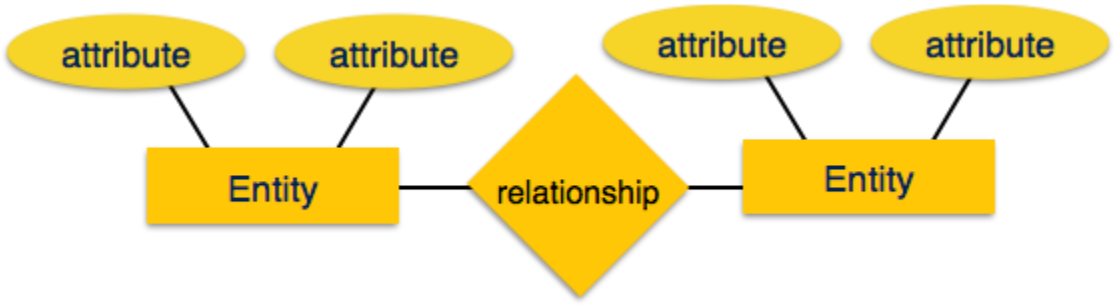
Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of a database.

ER Model is based on −

- **Entities** and their *attributes.*
- **Relationships** among entities.

These concepts are explained below.



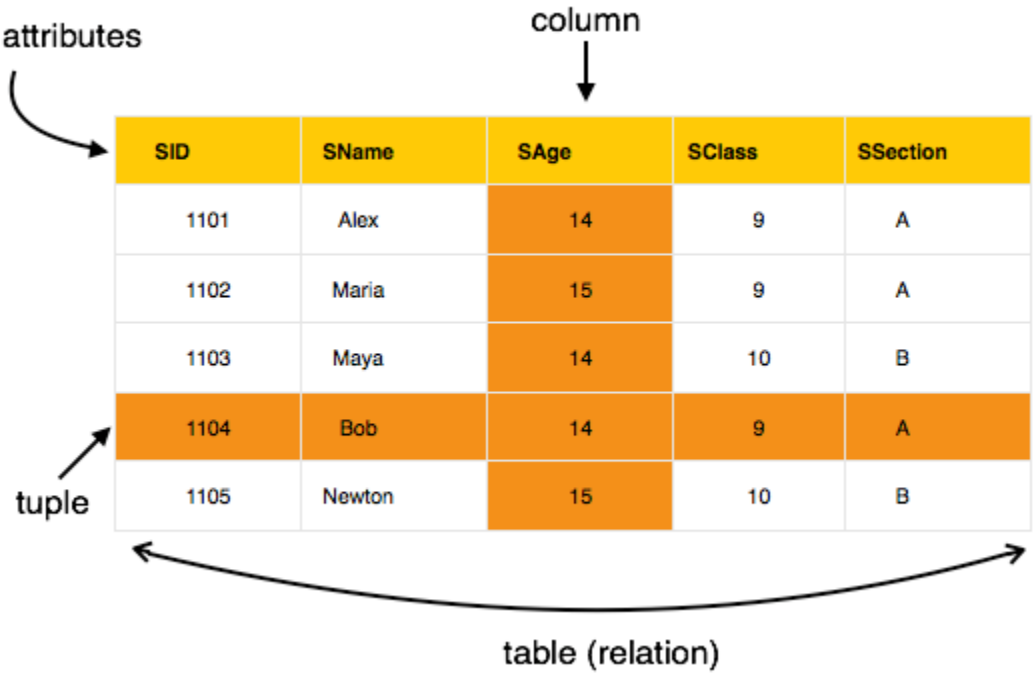
- **Entity** − An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.
- **Relationship** − The logical association among entities is called ***relationship***. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.

Mapping cardinalities −

- one to one
- one to many
- many to one
- many to many

Relational Model

The most popular data model in DBMS is the Relational Model. It is more scientific a model than others. This model is based on first-order predicate logic and defines a table as an **n-ary relation**.



The main highlights of this model are −

- Data is stored in tables called **relations**.
- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from a same domain.

Other Data Models

The purpose of a data model is to represent data and to make the data understandable. There have been many data models proposed in the literature. They fall into three broad categories:

- Object Based Data Models
- Physical Data Models
- Record Based Data Models

The object based and record based data models are used to describe data at the conceptual and external levels, the physical data model is used to· describe data at the internal level.

Object based data models use concepts such as entities, attributes, and relationships.

The object oriented data model extends the definition of an entity to include, not only the attributes that describe the state of the object but also the actions that are associated with the object, that is, its behavior.

Physical data models describe how data is stored in the computer, representing information such as record structures, record ordering, and access paths.

Record based logical models are used in describing data at the logical and view levels.

The three most widely accepted record based data models are:

- Hierarchical Model
- Network Model
- Relational Model

1.6 DBMS – Data Independence

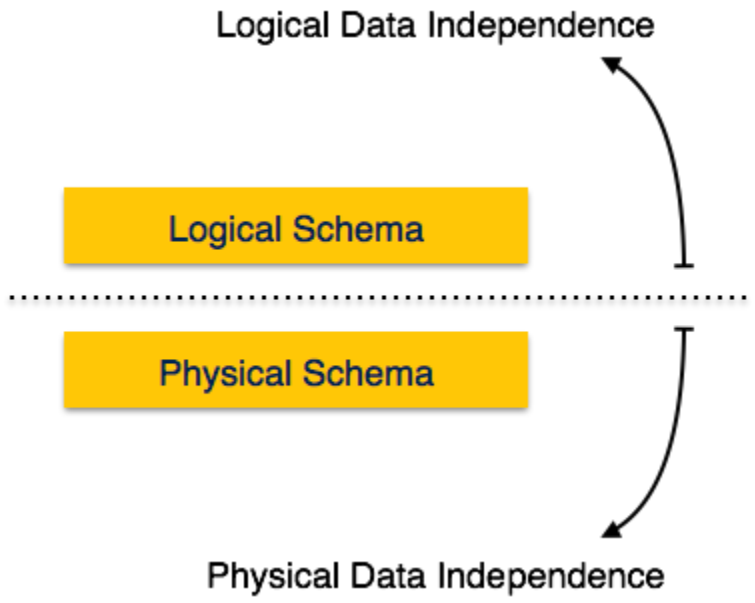
A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.

Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.



Physical Data Independence

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

1.7 Data Catalog

SELF-DESCRIBING NATURE OF A DATABASE SYSTEM:

- ✓ A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
- ✓ The description is called **meta-data**.
- ✓ Used by the DBMS software and also by database users who need information about the database structure.
- ✓ This allows the DBMS software to work with any number of database applications – for example a banking database or a company database.
- ✓ These definitions are specified by the database designer prior to creating the actual database and are stored in the catalog.
- ✓ In traditional file processing, data definition is typically part of the application programs themselves. Hence, these programs are constrained to work with only one specific database.

- ✓ **For Example**, the database catalog of fig 1.4 is represented in fig 1.5.

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Fig. 1.4 A database that stores student and course information.

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerated type with all known majors.
XXXXNNNN is used to define a type with four alpha characters followed by four digits.

Fig 1.5. Database catalog for the database in Fig 1.4.

1.8 DBMS System Structure Architecture

There are about 4 different types of database users differentiated by their way of interaction with the database system. Brief description has been given below.

1. **Naive Users:**

This type of users generally interacts with the system through previously created programs. Such as, a user may use the system to transfer some balance from one account to another. This will be done a program, like 'Transfer'. The user will only fill some required fields of the program like the balance, the accounts etc. The typical interfaces for these users are forms including some required fields to be filled.

2. **Application Programmers:** Application programmers are computer professionals. They develop user interfaces. Rapid Application Development (RAD) is a tool that enables them to construct forms and reports with minimal programming effort.
3. **Sophisticated Users:** Sophisticated users form their requests in a database query language and submit them to a query processor to make the storage manager understand the requests. Analysts who submit queries to explore data in the database fall in this category.

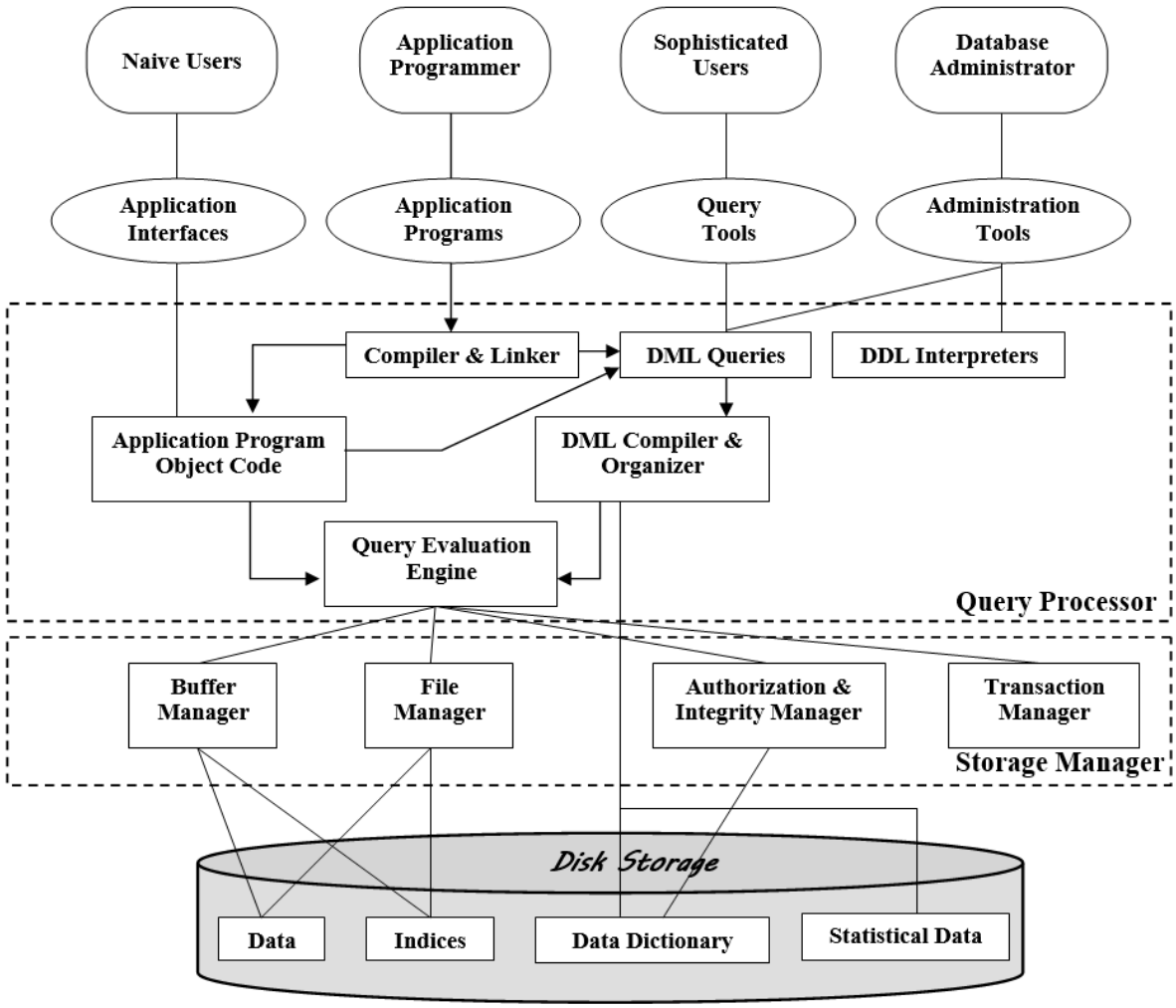


Figure: System Architecture

4. **Specialized Users:**

Specialized users are sophisticated users who write specialized database application that does not fit into the traditional data-processing framework. Among these applications are computer aided-design systems, knowledge-base and expert systems etc.

Query Processor:

As query is very much necessary to find out only the data user need from tons of data of the database, query processor is very important to process these query requests. Query processors come with the following components,

1. **DDL Interpreter:**
   It interprets the DDL statements and records the definitions in data dictionary.
2. **DML Compiler:** It translates the DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation understands. It also performs query optimization which actually picks up the lowest cost evaluation plan from various alternatives.
3. **Query Evaluation Engine:** It executes the low level instruction compiled by the DML compiler.

Storage Manager Component –

A Storage Manager is a component or program module that provides the interface between the low-level data stored in the database and the application programs/queries submitted to the system. The Storage Manager Components include –

1. **File Manager-** File manager manages the file space and it takes care of the structure of the file. It manages the allocation space on disk storage and the data structures used to represent info stored on other media.
2. **Buffer Manager –** It transfers blocks between disk (or other devices) and Main Memory. A DMA (Direct Memory Access) is a form of Input/Output that controls the exchange of blocks process. When a processor receives a request for a transfer of a block, it sends it to the DMA Controller which transfers the block uninterrupted.
3. **Authorization and Integrity Manager –** This Component of storage manager checks for the authority of the users to access and modify information, as well as integrity constraints (keys, etc).
4. **Disk Manager-** The block requested by the file manager is transferred by the Disk Manager.

The Structures maintained by Storage manager are-

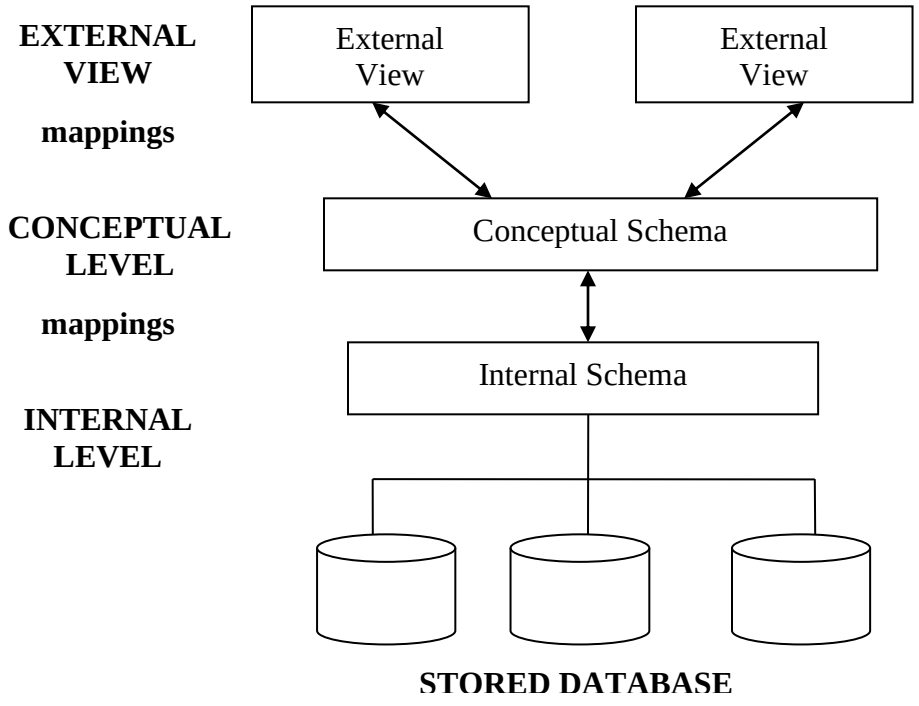
1. **Data Files-** Data files contains the data portion of the data base.
2. **Data Dictionary-** DBMS must a data dictionary function. The dictionary contains the data about the data. Rather than just raw data. The information about attributes, entity, mapping & cross reference information is contained in the data dictionary.
3. **Indices or Indexing and Access Aids –** An index is a small table having two columns in which the first column contains a copy of the primary or candidate key of a table and the second column contains a set of pointers holding the address of the disk block where that particular key value can be found. The advantage of using indices is that index makes search operation perform very fast.
   In a data base system, a set of access aids in the form of indexes are usually provided to improve the performance of a database system.

Three Schema Architecture

Remember from the previous chapters, three of the main characteristics of database systems, these are:

1. Insulation of programs and data
2. Support of multiple views
3. Use of a catalogue to store the database description (schema)

The three schema architecture helps to achieve these characteristics.



Three Schema Architecture

The goal of the three schema architecture is to separate the user applications and the physical database. The schemas can be defined at the following levels:

1. The internal level – has an internal schema which describes the physical storage structure of the database. Uses a physical data model and describes the complete details of data storage and access paths for the database.

2. The conceptual level – has a conceptual schema which describes the structure of the database for users. It hides the details of the physical storage structures, and concentrates on describing entities, data types, relationships, user operations and constraints. Usually a representational data model is used to describe the conceptual schema.

3. The External or View level – includes external schemas or user vies. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. Represented using the representational data model.

The three schema architecture is used to visualize the schema levels in a database. The three schemas are only descriptions of data, the data only actually exists is at the physical level.

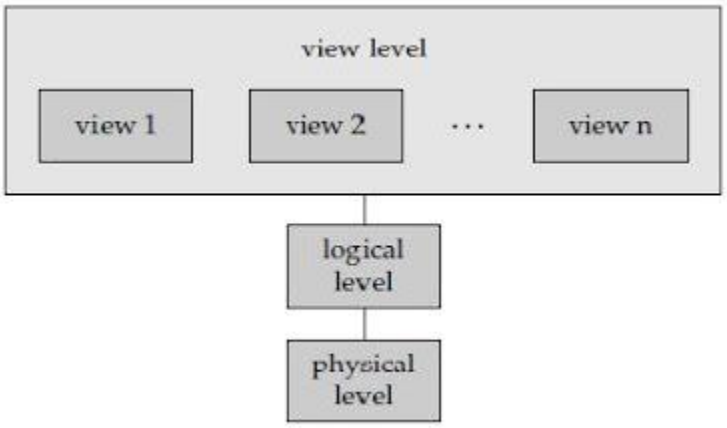
Each user group refers only to its own external schema. The DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the database. The process of transforming requests and results between levels is called mapping.

1.9 DBMS - 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. Since many database-systems users are not computer trained, 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. The physical level describes complex low-level data structures in detail.
- **Logical Level:** 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. 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.

1.10 DBMS Languages

A DBMS must provide appropriate languages and interfaces for each category of users to express database queries and updates. Database Languages are used to create and maintain database on computer. There are large numbers of database languages like Oracle, MySQL, MS Access, dBase, FoxPro etc. SQL statements commonly used in Oracle and MS Access can be categorized as data definition language (DDL), data control language (DCL) and data manipulation language (DML).

Data Definition Language (DDL)

It is a language that allows the users to define data and their relationship to other types of data. It is mainly used to create files, databases, data dictionary and tables within databases.

It is also used to specify the structure of each table, set of associated values with each attribute, integrity constraints, security and authorization information for each table and physical storage structure of each table on disk.

Data Manipulation Language (DML)

A Data Manipulation Language is a database language which enables users to access and modify stored data in a database. The types of access are as follows,

Retrieval of information stored in the database.

Insertion of new information into the database.

Deletion of information stored in the database.

Modification of information stored in the database.

There are basically two types,

Procedural DMLs: Procedural DMLs requires a user to specify what data are needed and how to get those data.

Declarative DMLs: Declarative DML's requires a user only to specify what data are needed.

In the case of retrieval of information some statements are used which are called queries. The portion of DML that contains information retrieval statement is called query language.

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

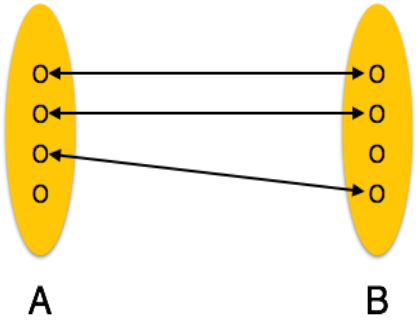
The number of participating entities in a relationship defines the degree of the relationship.

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

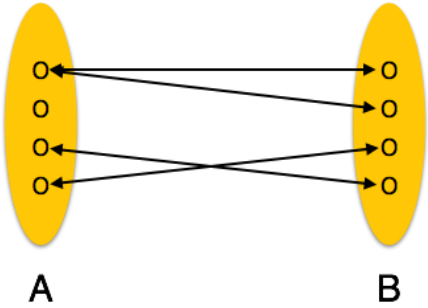
Mapping Cardinalities

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.

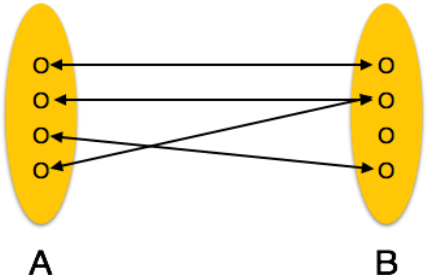
- **One-to-one** − One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



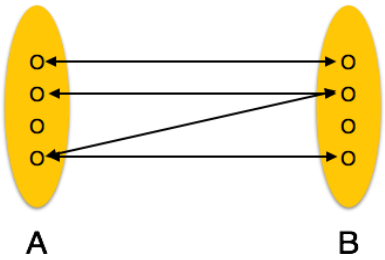
- **One-to-many** − One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



- **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.



ER Diagram Representation

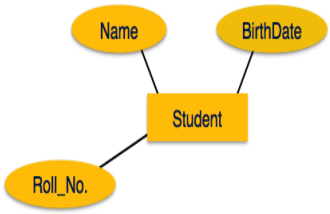
Let us now learn how the ER Model is represented by means of an ER diagram. Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram.

Entity

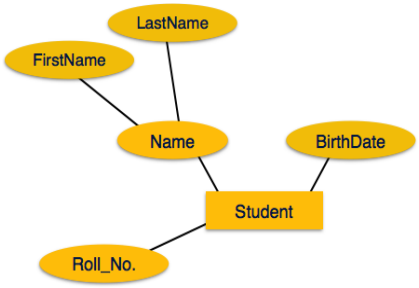
Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

Attributes

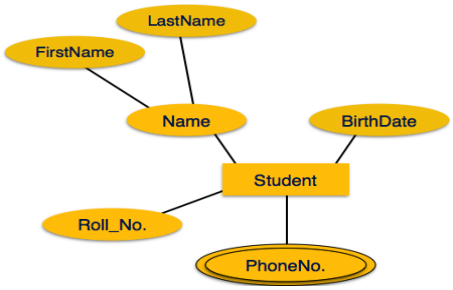
Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



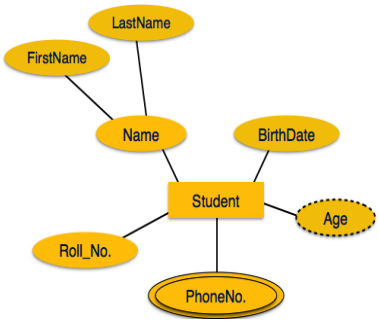
If the attributes are **composite**, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



Multivalued attributes are depicted by double ellipse.



Derived attributes are depicted by dashed ellipse.



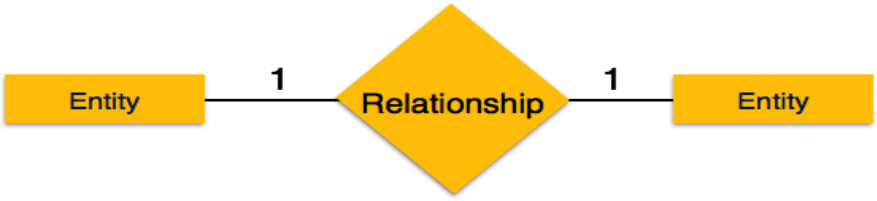
Relationship

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.

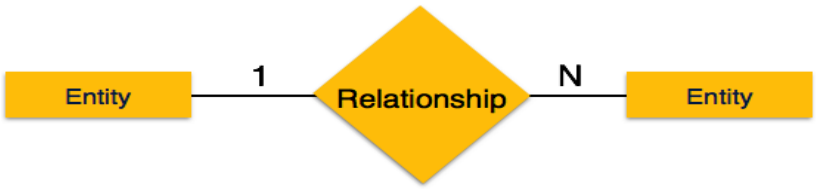
Binary Relationship and Cardinality

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.

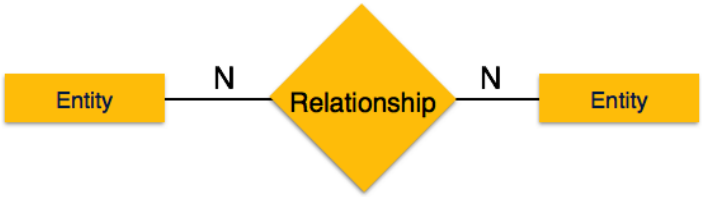


- **One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many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



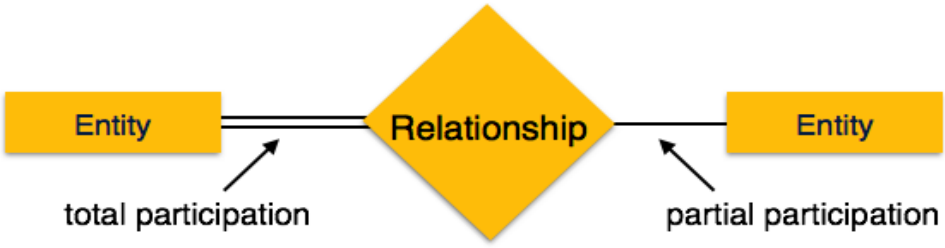
- **Many-to-one** − When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one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

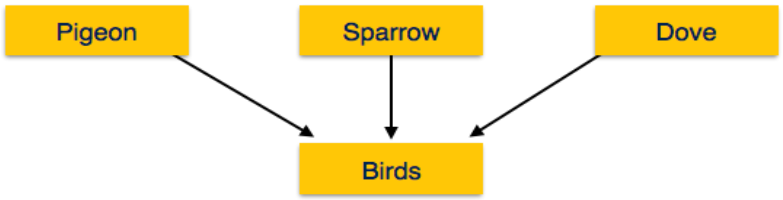


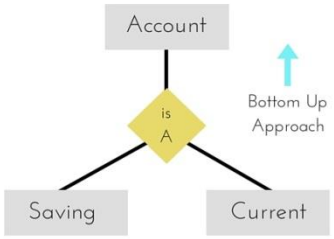
GENERALIZATION

Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower

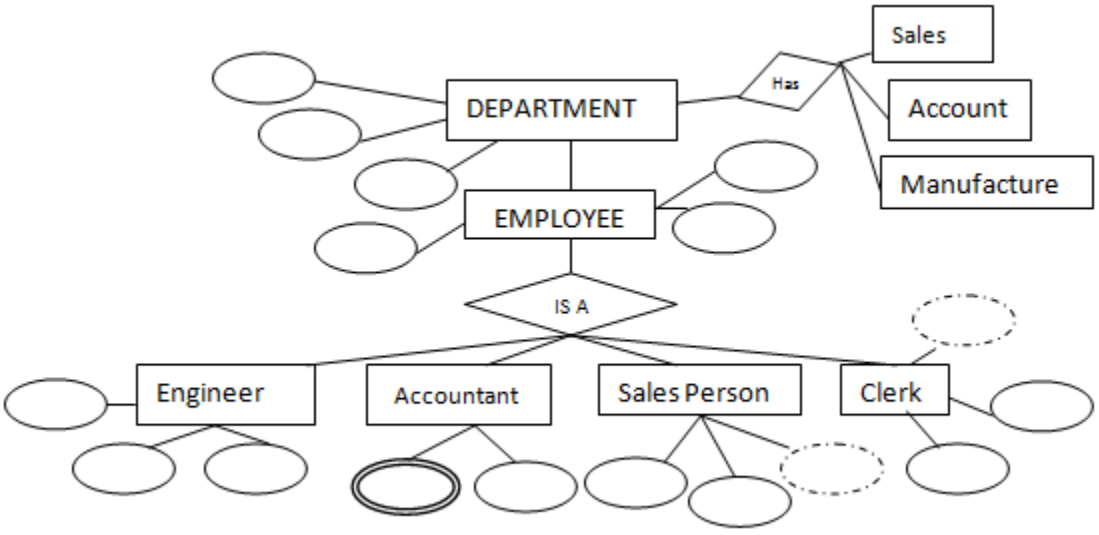
level entity to make further higher level entity. In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics.

For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.



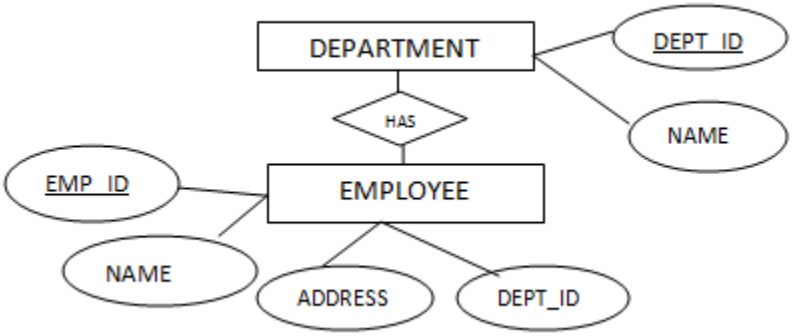


In our Employee example, we have seen different types of employees like Engineer, Accountant, Salesperson, Clerk etc. Similarly each employee belongs to different departments. We can represent it in an ER diagram as below. When you see this diagram for the first time, you will not understand it quickly. One will take time to understand it or he might misunderstand some requirement.



What if we group all the sub departments into one department and different employees into one employee? However sub departments and different employee types have same features in their own domain. So if we merge the child entities into their parent, it makes the diagram simpler, hence easy to understand. This method of merging the branches into one is called generalization.

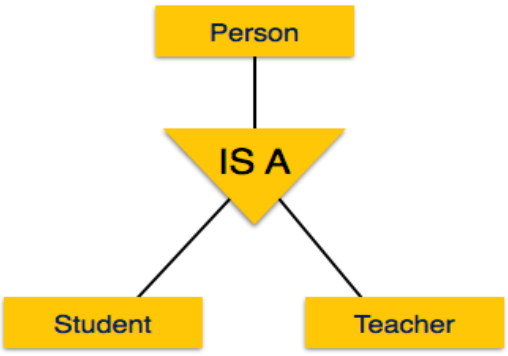
We can see the generalized structure of requirement to understand it quickly. So above ER diagram will be changed to as below:

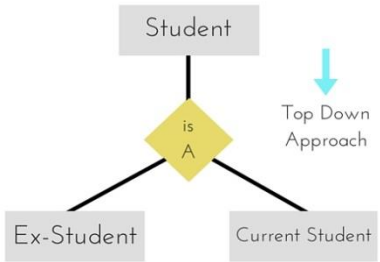


Isn't it simpler? Generalization is the bottom up approach which helps to design the requirement at high level. Thus making one to understand quickly.

SPECIALIZATION

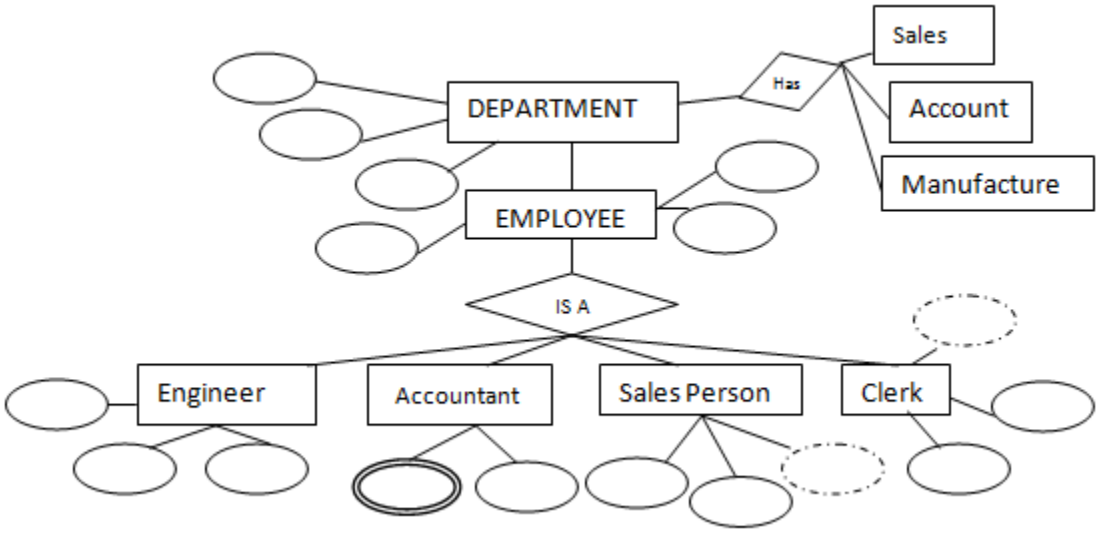
Specialization is the opposite of generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entities. In specialization, some higher level entities may not have lower-level entity sets at all. Take a group 'Person' for example. A person has name, 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.

It is opposite approach of generalization. Here, each entity is further divided into sub levels to understand it deeper. In the above example, Department entity is further divided into sub departments to understand how they are scattered. This method of representation helps the developer to code correctly and quickly. It is a top down approach of breaking higher level entity to low level entity. Once the entities are understood at higher level, it makes easy to understand the requirement at low level.

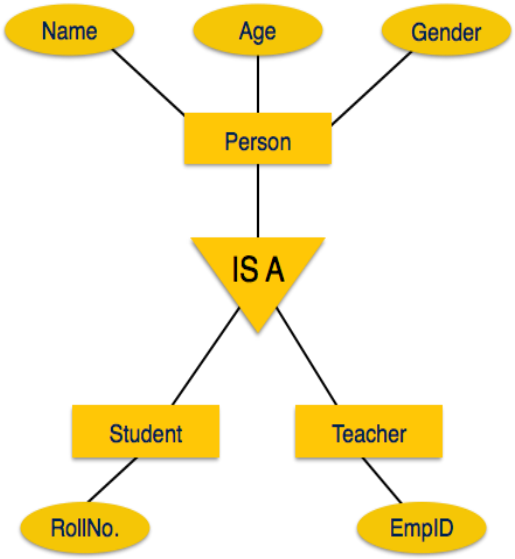


One more example of specialization would be Person. We can further divide person as STUDENT, TEACHER, ENGINEER, SOLDIER etc. (Merging STUDENT, TEACHER, ENGINEER etc into PERSON is an example of generalization).

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**.

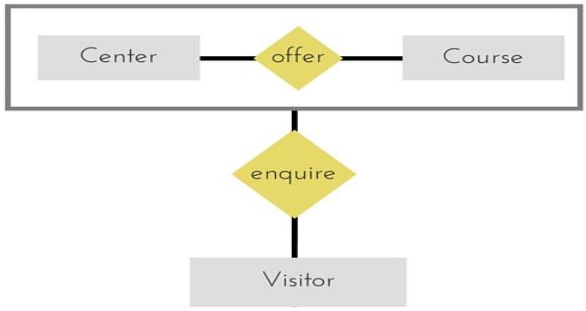
Inheritance is an important feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.



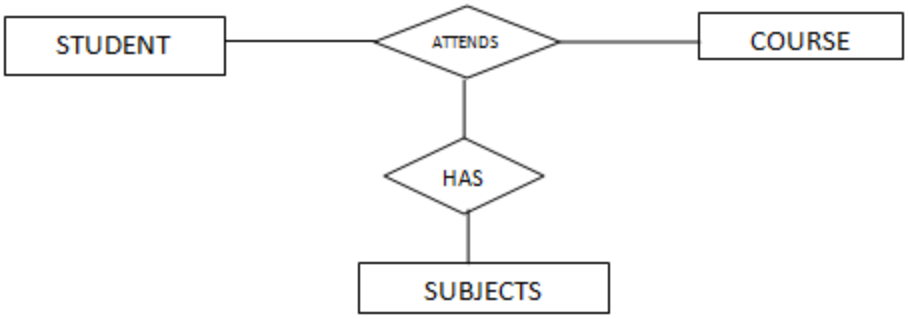
For example, the attributes of a Person class such as name, age, and gender can be inherited by lower-level entities such as Student or Teacher.

Aggregation

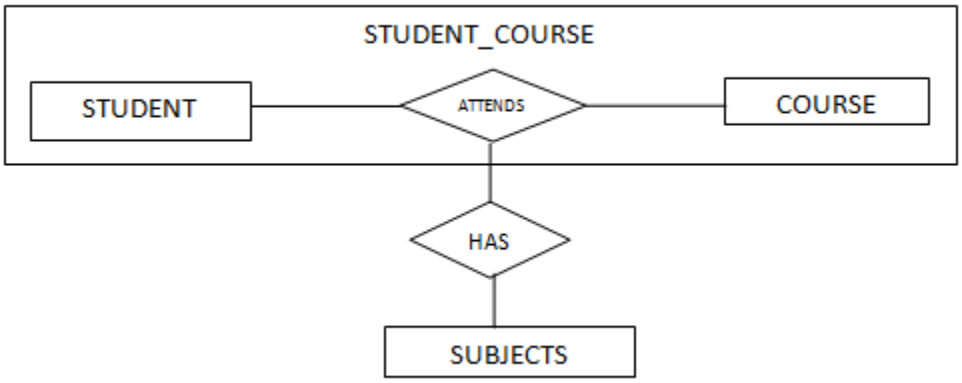
Aggregation is a process when relation between two entities is treated as a single entity. Here the relation between Center and Course, is acting as an Entity in relation with Visitor.



Look at below ER diagram of STUDENT, COURSE and SUBJECTS. What does it infer? Student attends the Course, and he has some subjects to study. At the same time, Course offers some subjects. Here a relation is defined on a relation. But ER diagram does not entertain such a relation. It supports mapping between entities, not between relations. So what can we do in this case?



If we look at STUDENT and COURSE from SUBJECT's point of view, it does not differentiate both of them. It offers it's subject to both of them. So what can we do here is, merge STUDENT and COURSE as one entity. This process of merging is called aggregation. It is completely different from generalization. In generalization, we **merge entities of same domain** into one entity. In this case we **merge related entities** into one entity.



Here we have merged STUDENT and COURSE into one entity STUDENT_COURSE. This new entity forms the mapping with SUBJECTS. The new entity STUDENT_COURSE, in turn has two entities STUDENT and COURSE with 'Attends' relationship.

Attributes vs. Entity-Sets

• When to represent a value as an attribute?

• When to represent a value as a separate entity-set?

• Representing as a separate entity-set allows details to be added later

• Example: – Phone number is a good candidate for representing as a separate entity-set – Employee name definitely should stay an attribute