DBMS

* **Data**
  + Data is raw fact or figures or entity, from which required information is produced.
  + When activities in the organization takes place, the effect of these activities need to be recorded which is known as Data.
  + For example, names, telephone numbers and addresses of the people
  + known facts that can be recorded and that have implicit meaning
* **Information**
  + Processed data is called information
  + The purpose of data processing is to generate the information required for carrying out the business activities.

1. **Tasks in data management**

• **Data capture:** Which is the task associated with gathering the data as and when they originate.

**• Data classification**: Captured data has to be classified based on the nature and intended usage.

• **Data storage:** The segregated data has to be stored properly.

• **Data arranging**: It is very important to arrange the data properly

• **Data retrieval:** Data will be required frequently for further processing. Hence it is very important to create some indexes so that data can be retrieved easily

• **Data maintenance:** Maintenance is the task concerned with keeping the data upto-date.

**• Data Verification:** Before storing the data it must be verified for any error.

**• Data Coding**: Data will be coded for easy reference.

**• Data Editing**: Editing means re-arranging the data or modifying the data for presentation.

**• Data transcription**: This is the activity where the data is converted from one form into another.

**• Data transmission:** This is a function where data is forwarded to the place where it would be used further.

1. **Metadata**

* **Metadata** (meta data, or sometimes meta information) is "data about data", of any sort in any media.
* An item of metadata may describe a collection of data including multiple content items and hierarchical levels, for example a database schema.
* Eg: A text document's metadata may contain information about how long the document is, who the author is, when the document was written, and a short summary of the document.
* Metadata within web pages can also contain descriptions of page content, as well as key words linked to the content.[[14]](https://en.wikipedia.org/wiki/Metadata) These links are often called "Metatags",

1. **Applications of DBMS**

The primary goal of a DBMS is to provide an environment that is both convenient and efficient for people to use in retrieving and storing information.

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.
12. **Database**

• Database may be defined in simple terms as a collection of data

• A database is a collection of related data.

• The database can be of any size and of varying complexity.

• A database may be generated and maintained manually or it may be computerized.

**Database Management System** - A Database Management System (DBMS) is a collection of program that enables user to create and maintain a database.

Def : general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.

**Defining** a database involves specifying the data types, structures, and constraints for the data to be stored in the database.

**Constructing** the database is the process of storing the data itself on some storage medium that is controlled by the DBMS.

**Manipulating** a database includes such functions as querying the database to retrieve specific data, updating the database as required, and generating reports from the data.

**Sharing** a database allows multiple users and programs to access the database concurrently.

**Maintaining**A typical large database may have a life cycle of many years, so the DBMS must be able to maintain the database system by allowing the system evolve as requirements change over time.

**Protection** includes both *system protection* against hardware or software malfunction (or crashes), and *security protection* against unauthorized or malicious access.

1. **Functions of DBMS**

**• Data Definition**: The DBMS provides functions to define the structure of the data in the application. These include defining and modifying the record structure, the type and size of fields and the various constraints to be satisfied by the data in each field.

**• Data Manipulation:** Once the data structure is defined, data needs to be inserted, modified or deleted. These functions which perform these operations are part of DBMS.

**• Data Security & Integrity**: The DBMS contains modules which handle the security and integrity of data in the application.

**• Data Recovery and Concurrency**: Recovery of the data after system failure and concurrent access of records by multiple users is also handled by DBMS.

**• Data Dictionary Maintenance:** Maintaining the data dictionary which contains the data definition of the application is also one of the functions of DBMS.

**• Performance:** Optimizing the performance of the queries is one of the important functions of DBMS.

1. **Advantages of DBMS**

**1. Data independency:**

Application program should not be exposed to details of data representation and storage DBMS provides the abstract view that hides these details.

#### 2. 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.

#### 3. Improved Data Sharing:

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

#### 4. 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.

Data is accessed through DBMS, it can enforce integrity constraints.

E.g.: Inserting salary information for an employee.

#### 5. 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.

#### 6. 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.

#### Efficient Data Access:

DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently.

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

#### 8. 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.

#### 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.

**10. Data Administration:**

When users share data, centralizing the data is an important task, Experience professionals can minimize data redundancy and perform fine tuning which reduces retrieval time.

**11. Concurrent access and Crash recovery:**

DBMS schedules concurrent access to the data. DBMS protects user from the effects of system failure.

1. **Disadvantages of DBMS**
2. Increased Complexity
3. Requirement of New and Specialized Manpower
4. Large Size of DBMS
5. Increased installation and management cost
6. Additional hardware cost
7. Conversion cost
8. Need for explicit backup and recovery
9. Organizational conflict
10. **Database System versus File System**

|  |  |
| --- | --- |
| File System  1. File system is a collection of data. Any management with the file system, user has to write the procedures  2. File system gives the details of the data representation and Storage of data.  3. In File system storing and retrieving of data cannot be done efficiently.  4. Concurrent access to the data in the file system has many problems like  a. Reading the file while other deleting some information, updating some information  5. File system doesn’t provide crash recovery mechanism.  6. Protecting a file under file system is very difficult. | DBMS  1. DBMS is a collection of data and user is not required to write the procedures for managing the database.  2. DBMS provides an abstract view of data that hides the details.  3. DBMS is efficient to use since there are wide varieties of sophisticated techniques to store and retrieve the data.  4. DBMS takes care of Concurrent access using some form of locking.  5. DBMS has crash recovery mechanism, DBMS protects user from the effects of system failures.  6. DBMS has a good protection mechanism. |

1. **Database Users**

There are four different types of database-system users, differentiated by the way theyexpect to interact with the system. Different types of user interfaces have been designedfor the different types of users.

* 1. **Na¨ıve users** are unsophisticated users who interact with the system by using predefined user interfaces, such as web ormobile applications. The typical user interface for na¨ıve users is a forms interface, where the user can fill in appropriate fields of the form. Na¨ıve users may also view read *reports* generated from the database.

As an example, consider a student, who during class registration period, wishes

to register for a class by using a web interface. Such a user connects to a web

application program that runs at a web server. The application first verifies the

identity of the user and then allows her to access a form where she enters the

desired information. The form information is sent back to the web application

at the server, which then determines if there is room in the class (by retrieving

information from the database) and if so adds the student information to the class

roster in the database.

* 1. **Application programmers** are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.
  2. **Sophisticated users** interact with the system without writing programs. Instead,they form their requests either using a database query language or by using tools such as data analysis software. Analysts who submit queries to explore data in the database fall in this category.
  3. **The Data base Administrator (DBA)** who is like the super-user of the system.

1. **Roles of DBA**

* **Schema definition:** The DBA creates the original database schema by executing aset of data definition statements in the DDL.
* **Liaising with users**: The DBA needs to interact continuously with the users to understand the data in the system and its use.
* **Defining Security & Integrity checks:** The DBA finds about the access restrictions to be defined and defines security checks accordingly. Data Integrity checks are defined by the DBA.
* **Defining Backup/Recovery Procedures**: The DBA also defines procedures for backup and recovery. Defining backup procedure includes specifying what data is to be backed up, the periodicity of taking backups and also the medium and storage place to backup data.
* **Monitoring performance:** The DBA has to continuously monitor the performance of the queries and take the measures to optimize all the queries in the application.
* **Storage structure and access-method definition**. The DBA may specify some parameters pertaining to the physical organization of the data and the indices to be created.
* **Schema and physical-organization modification.** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
* **Granting of authorization for data access.** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever a user tries to access the data in the system.
* **Routine maintenance**. Examples of the database administrator’s routine maintenance activities are:

° Periodically backing up the database onto remote servers, to prevent loss of

data in case of disasters such as flooding.

° Ensuring that enough free disk space is available for normal operations, and

upgrading disk space as required.

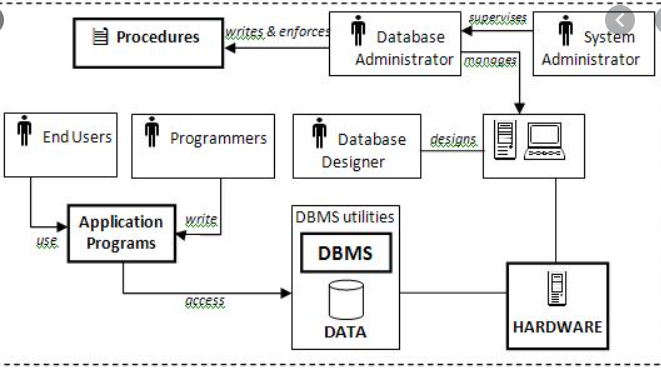
° Monitoring jobs running on the database and ensuring that performance is not

degraded by very expensive tasks submitted by some users.

1. **Database System Environment**

A database environment is a collective system of components that comprise and regulates the group of data, management, and use of data

Consists of: software, hardware, people, techniques of handling database (procedures), and the data



* + 1. **Hardware:**Hardware refers to all of the system's physical devices; for example, computers storage devices, printers, network devices and etc
    2. **Software:** To make the database system work properly, three types of software are needed: operating system, DBMS software, and application programs.

a)       Operating system: It manages all hardware components and allows other software to run on the computers. Examples of operating system software include Windows, Linux and etc.

b)       DBMS software: It manages the database within the database system. Some examples of DBMS software include Oracle, Access, MySql and etc.

c)       Application programs: These are used to access and manipulate data in the DBMS and to manage the computer environment in which data access and manipulation take place. Application programs are most commonly used to access data to generate reports.  Most of the application programs provide GUI.

3. **People:** This component includes all users of the database system. According to the job nature, five types of users can be identified: systems administrators, database administrators, database designers, systems analysts and programmers, and end users.

a)       System administrators: They supervise the database system's general operations.

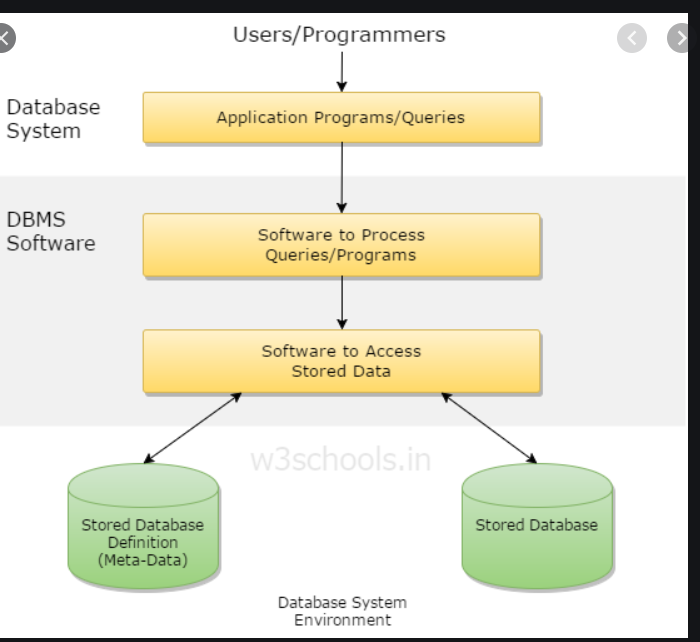
b)       Database administrators: They are also known as DBAs. They manage the DBMS and ensure that the data­base is functioning properly.

c)       Database designers: They design the database structure. They are the database architects. As this is very critical, the designer's job responsibilities are increased.

d)       Systems analysts and programmers: They design and implement the application programs. They design and create the data entry screens, reports, and procedures through which end users can access and manipulate the data.

e)       End users: They are the people who use the application programs to run the organization's daily operations. For example, sales-clerks, supervisors, managers are classified as end users.

1. **Procedures:** Procedures are the instructions and rules that supervise the design and use of the data­base system. Procedures are a critical component of the system. Procedures play an important role in a company because they enforce the standards by which business is conducted in an organization
2. **Data:** Data refers the collection of facts stored in the database. Because data are the raw material from which information is generated, no database can exist without database.



A database management system (DBMS) is a collection of programs that enables users to create and maintain database.

The DBMS is a general purpose software system that facilitates the process of defining, constructing, manipulating and sharing databases among various users and applications.

Defining a database specifying the database involves specifying the data types, constraints and structures of the data to be stored in the database.

The descriptive information is also stored in the database in the form database catalog or dictionary; it is called meta-data.

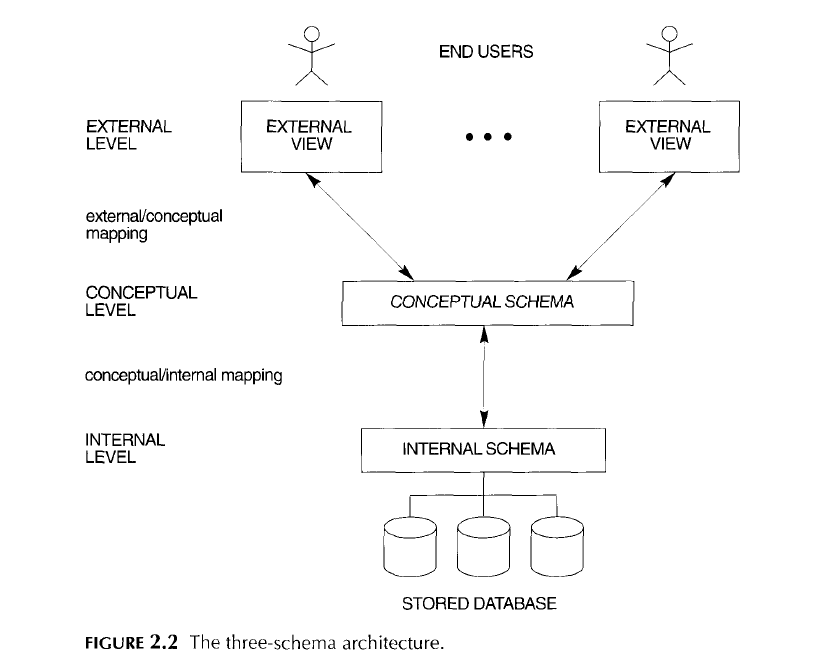
Manipulating the data includes the querrying the database to retrieve the specific data. An application program accesses the database by sending the qurries or requests for data to DBMS.

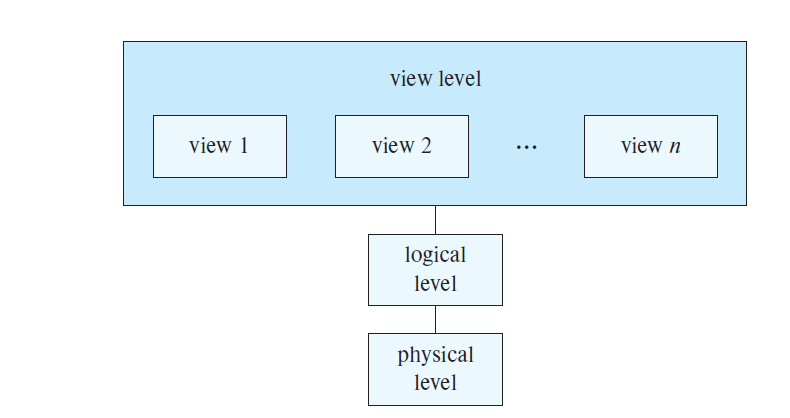
The important function provided by the DBMS includes protecting the database and maintain the database.

1. **Why a Spreadsheet Is Not a Database**

While a spreadsheet allows for the creation of multiple tables, it does not support even the most basic database functionality such as support for self-documentation through metadata, enforcement of data types or domains to ensure consistency of data within a column, defined relationships among tables, or constraints to ensure consistency of data across related tables. Most users lack the necessary training to recognize the limitations of spreadsheets for these types of tasks

1. **3-Tier Architecture of DBMS**





A commonly used views of data approach is the three-level architecture suggested by ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee).

The three levels of the architecture are three different views of the data:

**External - individual user view:**

* It represents the user’s view of the database.
* This view is often a restricted view of the database and the same database may provide a number of different views for different classes of users.
* It is closely related to the real world as perceived by each user.
* It is quite abstract in nature and closer to the way a programming language would model a record
* The view level exists to simplify the interaction of users with the system.
* It allows independent customized user views.
* It hides the physical storage details from users
* The external or view level includes a number of external schemas or user views. 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
* s/w and h/w independent
* Software independence means that the model does not depend on the DBMS software used to implement the model, i.e, the view of data for a given user will be same, regardless of the software he uses.
* Hardware independence means that the model does not depend on the hardware used in the implementation of the model, that is it is unaffected by the choice of the computer on which the software is installed. Therefore, a change in storage devices or even a change in operating systems will not affect the internal model

**Conceptual - community user view – logical level**

* Present a “community view”: the logical structure of the entire database
* The conceptual level is a way of describing what data is stored within the whole database and how the data is inter-related. It does not specify how the data is physically stored.
* This view is normally more stable than the other two views. [Change in physical level does not affect conceptual level]
* The conceptual view is defined by the conceptual schema which includes definitions of each of the various types of data.
* The conceptual model represents a global view of the entire database as viewed by the entire organization. That is, the conceptual model integrates all external views (entities, relationships, constraints, and processes) into a single global view of the data in the enterprise. Also known as a conceptual schema,

The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints

* Show relationships among data including:

Constraints

Semantic information (e.g., business rules)

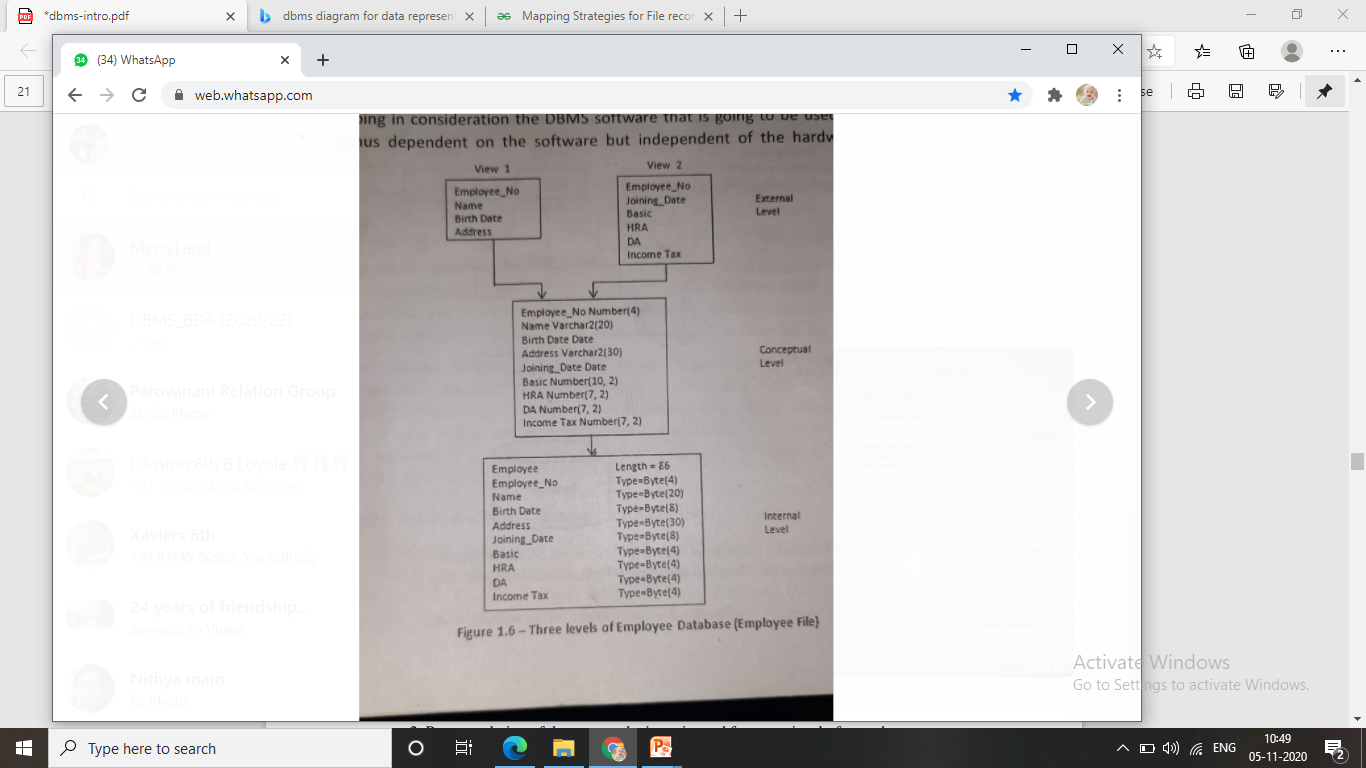
Security and integrity information

* Some facts: DBA works at this level. Describes the structure of all users. Only DBA can define this level. Global view of database.
* Because the internal model depends on specific database software, it is said to be software-dependent. Therefore, a change in the DBMS software requires that the internal model be changed to fit the characteristics and requirements of the implementation database model
* Hardware independence means that the model does not depend on the hardware used in the implementation of the model, that is it is unaffected by the choice of the computer on which the software is installed. Therefore, a change in storage devices or even a change in operating systems will not affect the internal model

**Internal - physical or storage view**

* The internal view is the view about the actual physical storage of data. It tells us what data is stored in the database and how.
* At least the following aspects are considered at this level:
  + - 1. Storage allocation e.g. B-trees, hashing etc.
      2. Access paths e.g. specification of primary and secondary keys, indexes and pointers and sequencing.
      3. Miscellaneous e.g. data compression and encryption techniques, optimization of the internal structures.
* provides information regarding the physical organisation of the data in the database
* The internal model is the representation of the database as “seen” by the database
* Provide concepts that describe the details of how data are stored in the computer’s memory
* deals with
  + Run-time performance
  + Storage utilization and compression
  + File organization and access methods
  + Data encryption
* The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
* the physical model is dependent on the DBMS, methods of accessing files, and types of hardware storage devices supported by the operating system. Therefore h/w s/w dependent.
* managed by the operating system (OS)

Example:



The three-schema architecture is a convenient tool with which the user can visualize the schema levels in a database system. Most DBMSs do not separate the three levels completely, but support the three-schema architecture to *some* extent. Some DBMSs may include physical-level details in the conceptual schema.

**Mappings** : the three schernas are only *descriptions* of data; the only data that *actually*

exists is at the physical level. In a DBMS based on the three-schema architecture, each

user group refers only to its own external schema. Hence, 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 stored database. If the

request is a database retrieval, the data extracted from the stored database must be

reformatted to match the user's external view. The processes of transforming requests and

results between levels are called **mappings**

1. **Data Independence**

Data independence can be defined as the capacity to change the schema at one level without changing the schema at 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.

They are of two types:

* + - * 1. **Logical data independence:**
* is the capacity to change the conceptual schema without having to change the external schema, In other words, changes to the logical schema (e.g., alterations to the structure of the database like adding a column or other tables) should not affect the function of the application (external views).
* Changes in entities, attributes, or relationships should not affect views not dealing directly with them
* We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item). In the last case, external schemas that refer only to the remaining data should not be affected.
* Only the view definition and the mappings need be changed in a DBMS that supports logical data independence. After the conceptual schema undergoes a logical reorganization, application programs that reference the external schema constructs must work as before.
  + - * 1. **Physical data independence:**
* is the capacity to change the internal schema without changing the conceptual schema, Hence, the external schemas need not be changed as well.
* Changes to the internal schema may be needed because some physical files had to be reorganized-for example, by creating additional access structures-to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema.
* In other words, it is Immunity of the conceptual schema to changes in the internal schema.
* Changes in storage structures, hardware, indexes, hashing, defragmentation, etc. - Only effect might be a change in performance.

Ques : Is database architecture same as levels of abstraction?

Q: why is conceptual level most stable? More stable than physical level is understood, but why wrt to user level?

DBMS Database Models

A data model is a relatively simple representation, usually graphical, of more complex real-world data structures.

Within the database environment, a data model represents data structures and their characteristics,

relations, constraints, transformations, and other constructs with the purpose of supporting a specific problem domain.

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.

Basics

* The basic building blocks of all data models are entities, attributes, relationships, and constraints.
* An entity is anything (a person, a place, a thing, or an event) about which data are to be collected and stored. An entity represents a particular type of object in the real world. Because an entity represents a particular type of object, entities are “distinguishable”—that is, each entity occurrence is unique and distinct
* An attribute is a characteristic of an entity.
* A relationship describes an association among entities.
* One-to-many (1:M or 1..\*) relationship
* Many-to-many (M:N or \*..\*) relationship
* One-to-one (1:1 or 1..1) relationship.
* A constraint is a restriction placed on the data. Constraints are important because they help to ensure data integrity. Constraints are normally expressed in the form of rules

While the Relational Model is the most widely used database model, there are other models too:

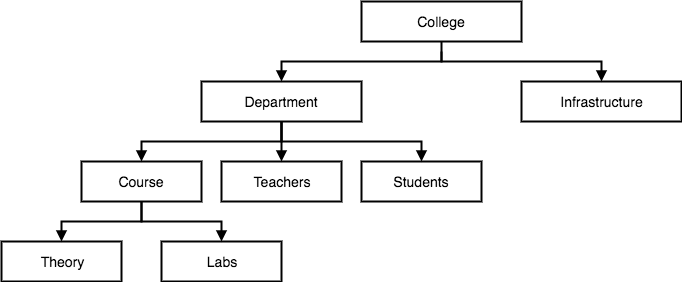
* Hierarchical Model
* Network Model
* Entity-relationship Model
* Relational Model

Hierarchical Model

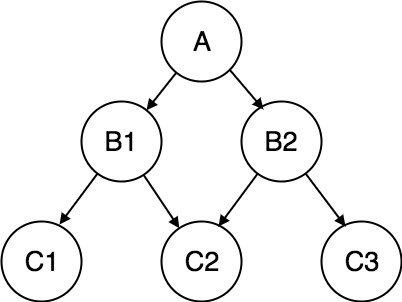
* The hierarchical model was developed in the 1960s to manage large amount of data for complex manufacturing project.
* This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the Root data, and expands like a tree, adding child nodes to the parent nodes.
* This model is used to depict one-to-many(1:M) relationship between Parent and Child.

o Each Parent may have child

o Each child has only one Parent.

* This model efficiently describes many real-world relationships like index of a book, recipes etc.
* All attributes of a specific record are listed under an entity type.
* 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.
* 
* Advantages
* Easy to design
* Easy to share
* Helps in data integrity
* Disadvantages:
* Complex to implement
* A clear picture of relations and attributes should be well planned beforehand.
* Difficult to manage : Difficult to modify the structure as and when needed.
* Lack of standards
* Implementation limitation
* Can be used in only 1-many relationships.

Network Model

* The network model was created to represent complex data relationship more effectively than hierarchical model, to improve database performance, and to impose a database standard.
* 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.
* 

While the network database model is generally not used today, the definitions of standard database *concepts* that emerged with the network model are still used by

modern data models. Some important concepts that were defined at this time are:

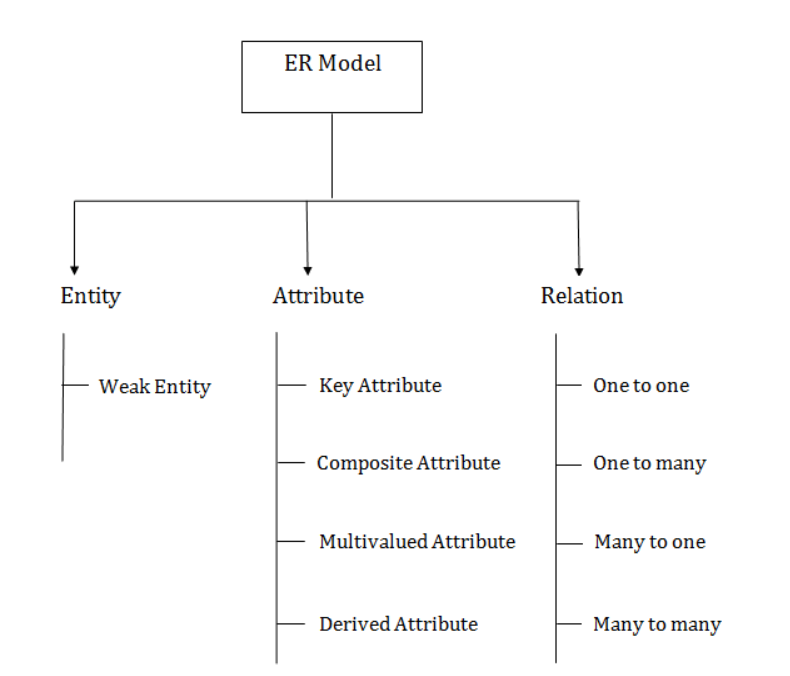
* The schema, which is the conceptual organization of the entire database as viewed by the database administrator.
* The subschema, which defines the portion of the database “seen” by the application programs that actually produce the desired information from the data contained within the database.
* A data management language (DML), which defines the environment in which data can be managed and to work with the data in the database.
* A schema data definition language (DDL), which enables the database administrator to define the schema components.

Advantages

* Easy to design
* Many to many relations
* Superior data access than hierarchial
* Has some standards.
* Integrity..

Disadvantages:

* Complex system due to too many connections.
* Too cumbersome
* Lack of ad-hoc query capability
* Any structural change need to change in whole application – no structural independence



The Entity-Relationship Model:-

 The entity-relationship (E-R) data model was developed to facilitate database design that represents the overall logical structure of a database.

 The E-R model is one of several semantic data model.

 The E-R model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.

* + 1. 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 an enterprise is an entity. An entity has a set of properties, and the values for some set of properties may uniquely identify an entity.

 For instance, a person may have a person\_id property whose value uniquely identifies that person.

 An entity might be

* An object with physical existence (e.g., a lecturer, a student, a car)
* An object with conceptual existence (e.g., a course, a job, a positio

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

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

 Each entity has a value for each of its attributes.

 Weak Entity: An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle. For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.

* + 1. Attributes:-

 For each attribute, there is a set of permitted values, called the domain, or value set, of that attribute.

1. Key attribute:

* A key attribute can uniquely identify an entity from an entity set.
* Cannot have null or duplicate values.

 Simple and composite attributes:-

 The attributes have been simple; that is they have not been divided into subparts.

 Composite attributes on the other hand, can be divided into subparts.

 For example, an attribute name could be structured as a composite attribute consisting of first\_name, middle\_initial, and last\_name.

2. Composite attribute:

An attribute that is a combination of other attributes is known as composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country

 required and optional attributes:-

 A required attribute is an attribute that must have a value; in other words, it cannot be left empty\

 An optional attribute is an attribute that does not require a value; therefore, it can be left empty

 Single-valued and mutlivalued attributes:-

 There may be instance where an attributes has a sert of values for specific entity.

 Consider an employee entity set with the attribute phone\_number.

 An employee may have zero, one, or several phone numbers.

 This type of attribute is said to be multivalued.

3. Multivalued attribute:

* An attribute that can hold multiple values is known as multivalued attribute. It is represented with double ovals in an ER Diagram. For example – A person can have more than one phone numbers so the phone number attribute is multivalued.

 Derived Attributes:-

 The value for this type of attribute can be derived from the values of other related attributes or entities.

 Customer entity set has an attribute age that indicates the customer’s age.

 If the customer entity set also has an attribute date\_of\_birth, we can calculate age from date\_of\_birth and current date. Thus, age is derived attribute.

 An attribute takes a null value when an entity does not have a value for it.

 The null value may indicate “not-applicable”- that is, that the value does not exist for the entity.

4. Derived attribute:

* A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by dashed oval in an ER Diagram. For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).

Read RDBMS TextBook for this.

* + 1. Relationship Sets:-

 A relationship is an association among several entities.

 A relationship set is a set of relationships of the same type.

 Consider the two entity sets customer and loan. We define the relationship set borrower to denote the association between customers and the bank loans that the customers have.

 The association between entity sets is referred to as participation; that is, the entity sets E1, E2…, En Participate in relationship set R.

 The function that an entity plays in a relationship is called that entity’s role.

 Since entity set participating in a relationship set are generally distinct, roles are implicit and are not usually specified.

 However, they are useful when the meaning a recursive relationship set.

 A relationship may also have attributes called descriptive attributes.

Mapping Cardinalities:-

 Mapping cardinalities, or cardinality rations, express the number of entities to which another entity can be associated via a relationship set.

 For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:

 Oneto-one. An entity is A is associated with at most one entity in B, an entity in B is associated with at most one entity in A.

 One-to-many. An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A.

 Many-to-one. An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number of entities in A.

 Many-to-many. An entity in A is associated with any number of entities in B, and an entity in B, is associated with any number of entities in A.

Keys:-

 A key allows us to identify a set of attributes that suffice to distinguish entities from each other.

 Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

 A superkey is a set of one or more attributes that, taken collectively, allow us to identify uniquely an entity in the entity set.

 For example, the customer\_id attribute of the entity set customer is sufficient to distinguish one customer entity from another.

 Thus, customer\_is is a superkey.

 Similarly, the combination of customer\_name and customer\_id is a superkey for the entity set customer.

 The customer\_name attribute of customer is not a superkey, because several people might have the same name.

 Such minimal superkeys are called candidate keys.

 Suppose that combination of customer\_name and customer\_street is sufficient to distinguish among members of the customer entity sets.

 Then, both {customer\_id} and {cus tomer\_name, cutomer\_street} are candidate keys.

 Primary key to donate a candidate key that is chosen by the database designer as the principal means of identifying entities within an entity set.

Participation Constraints:-

 The participation of an entity set E in a relationship set R is said to be a total if every entity in E participates in at least one relationship in R.

 If only some entities in E participate in relationship in R, the participation of entity set E in relationship R is said to be partial.

 For example, we expect every loan entity to be related to at least one customer through the borrower relationship.

 Therefore the participation of loan in the relationship set borrower is total.

 An individual can be a bank customer whether or not she has a loan with the bank.

 Participation off customer in the borrower relationship set is therefore partial.

Entity-Relationship Diagrams:-

 Rectangles, which represent entity set

 Ellipses, which represent attribute

 Diamonds, which represent relationship sets

 Lines, which link attributes to entity sets and entity sets to relationship sets

 Double ellipses, which represent multivalued attributes

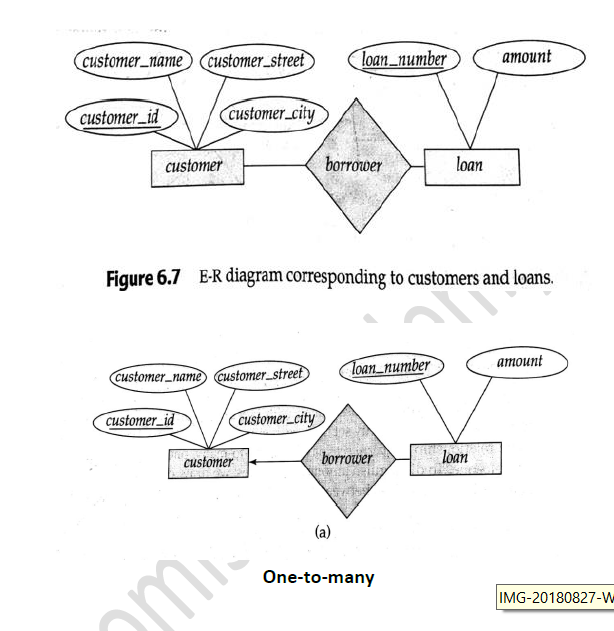
 Dashed ellipses, which denote derived attributes

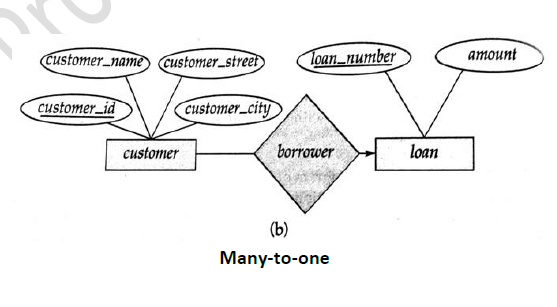
 Double lines, which indicate total participation of an entity in a relationship set

 Double rectangles, which represent weak entity set

 Underline represents primary key

See mam notes for how to draw.





Weak and Strong Entity Sets:-

 An entity set may not have sufficient attributes to form a primary key.

 Such an entity set is termed a weak entity.

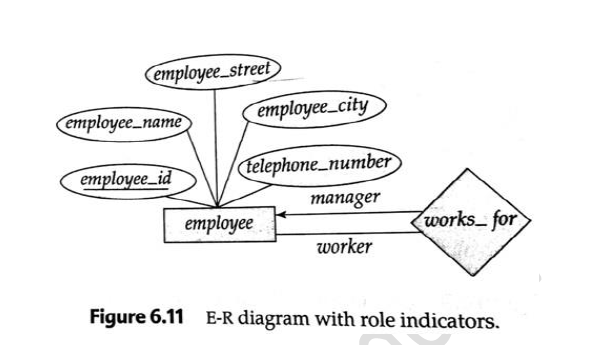
 An entity set that has a primary key is termed a strong entity set.

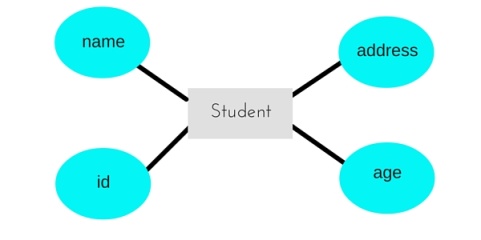
 For a weak entity set to be meaningful it must be associated with another entity set, called the identifying or owner entity set.

 Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be existence dependent on the identifying entity set.

 The identifying entity set is said to own the weak entity set that it identifies.

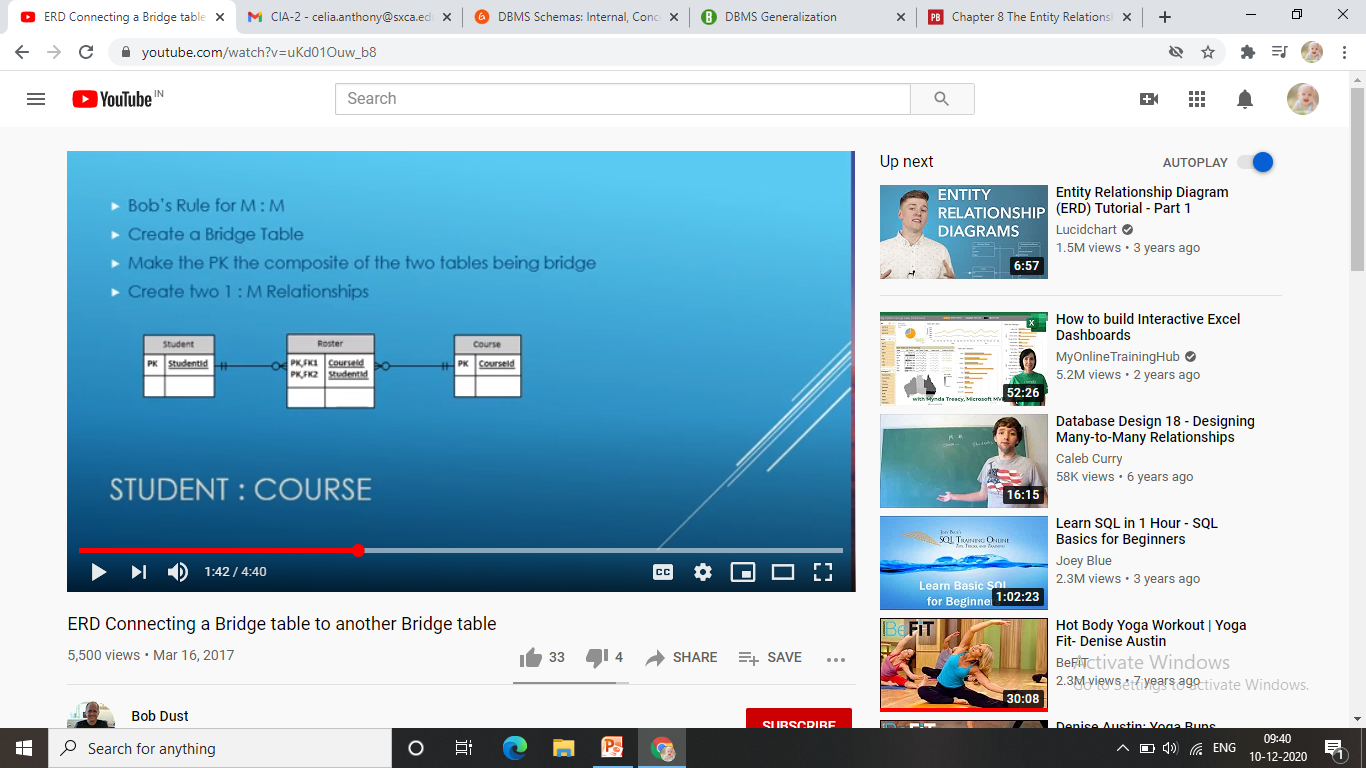
 The relationship associating the weak entity set with the identifying entity set is called the identifying relationship.



* 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.
* 

Composite Entity:

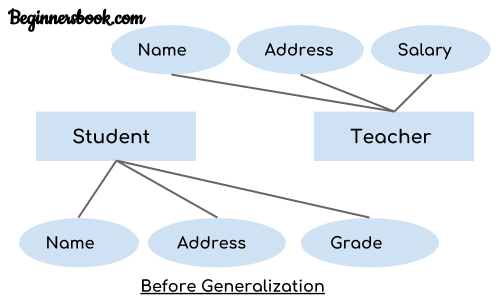
A composite entity is also known as a “bridge” entity. This “bridge” is used to handle the many-to-many relationships that the traditional entity could not handle. This entity lies between the two entities that are of interest and this composite entity shares the primary keys from both the connecting tables. This composite entity is also known as a “gerund” because it has the characteristics of an entity and a relationship



* Generalization is a process in which the common attributes of more than one entities form a new entity. This newly formed entity is called generalized entity.
* 1. Generalization uses bottom-up approach where two or more lower level entities combine together to form a higher level new entity.  
  2. The new generalized entity can further combine together with lower level entity to create a further higher level generalized entity.

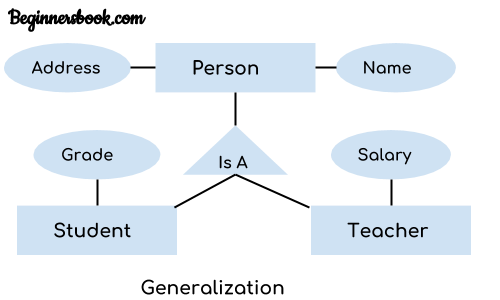
## Generalization Example

Lets say we have two entities Student and Teacher.  
Attributes of Entity Student are: Name, Address & Grade  
Attributes of Entity Teacher are: Name, Address & Salary

**The ER diagram before generalization looks like this:**  


These two entities have two common attributes: Name and Address, we can make a generalized entity with these common attributes. Lets have a look at the ER model after generalization.

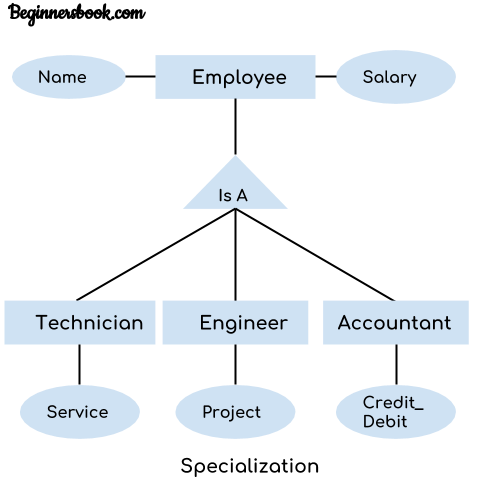
**The ER diagram after generalization:**  
We have created a new generalized entity Person and this entity has the common attributes of both the entities. As you can see in the following [ER diagram](https://beginnersbook.com/2015/04/e-r-model-in-dbms/) that after the generalization process the entities Student and Teacher only has the specialized attributes Grade and Salary respectively and their common attributes (Name & Address) are now associated with a new entity Person which is in the relationship with both the entities (Student & Teacher).



**Specialization** is a process in which an entity is divided into sub-entities. You can think of it as a reverse process of [generalization](https://beginnersbook.com/2018/11/dbms-generalization/), in generalization two entities combine together to form a new higher level entity. Specialization is a top-down process.

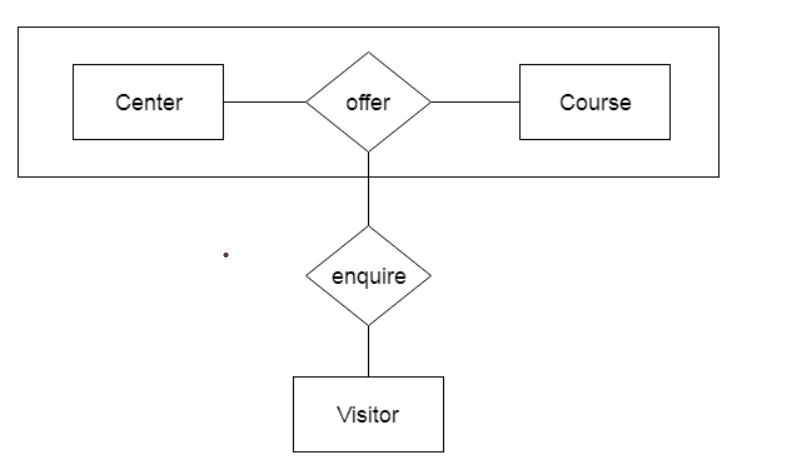
The idea behind Specialization is to find the subsets of entities that have few distinguish attributes. For example – Consider an entity employee which can be further classified as sub-entities Technician, Engineer & Accountant because these sub entities have some distinguish attributes.

## Specialization Example



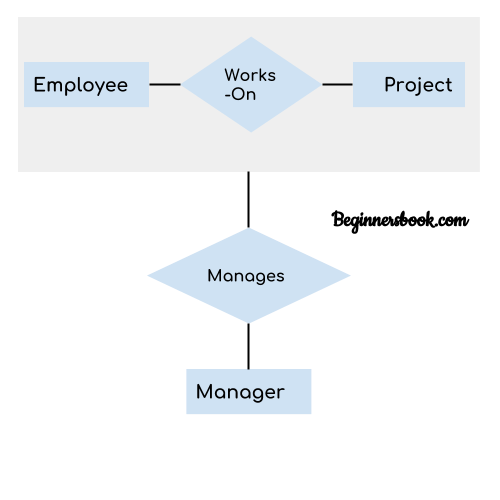
In the above diagram, we can see that we have a higher level entity “Employee” which we have divided in sub entities “Technician”, “Engineer” & “Accountant”. All of these are just an employee of a company, however their role is completely different and they have few different attributes. Just for the example, I have shown that Technician handles service requests, Engineer works on a project and Accountant handles the credit & debit details. All of these three employee types have few attributes common such as name & salary which we had left associated with the parent entity “Employee” as shown in the above diagram.

For example: Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.

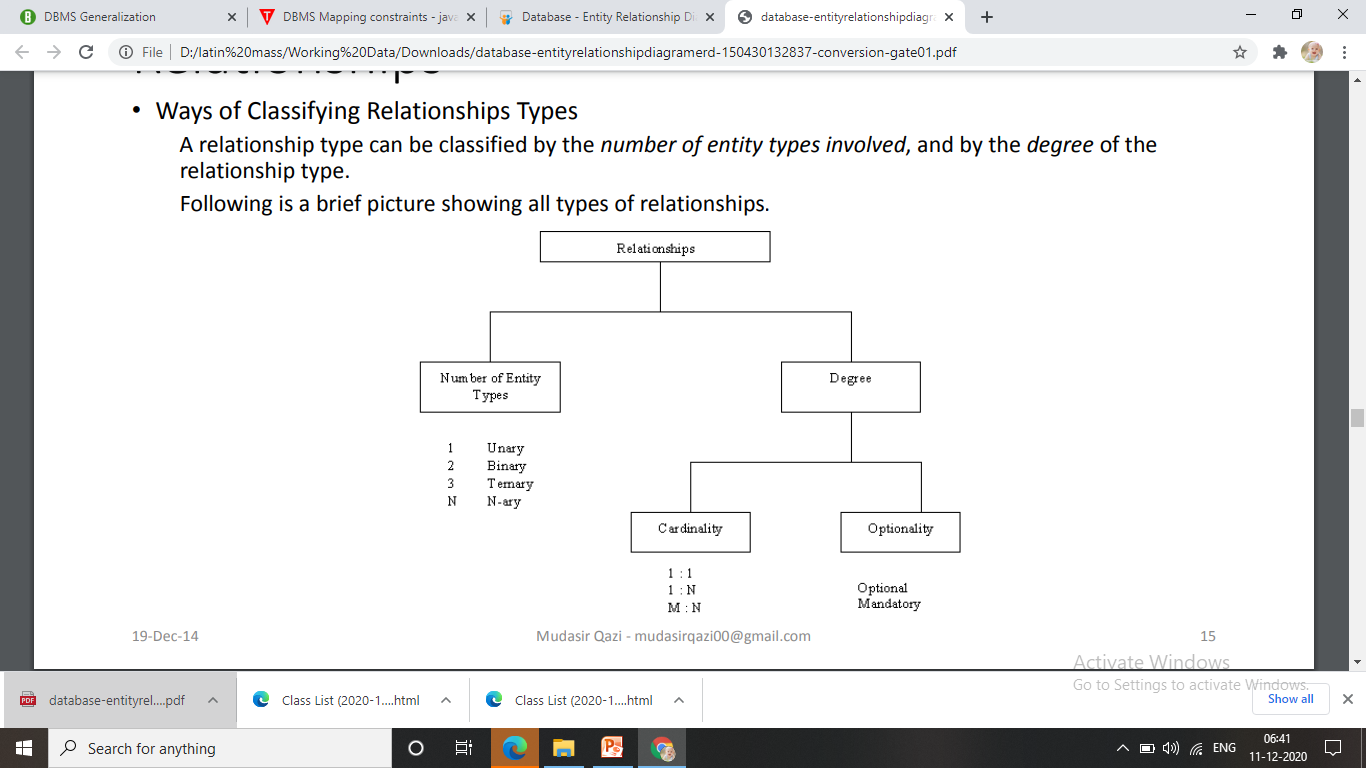


**Aggregation** is a process in which a single entity alone is not able to make sense in a relationship so the relationship of two entities acts as one entity. I know it sounds confusing but don’t worry the example we will take, will clear all the doubts.

## Aggregration Example

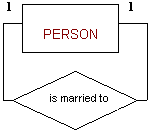


In real world, we know that a manager not only manages the employee working under them but he has to manage the project as well. In such scenario if entity “Manager” makes a “manages” relationship with either “Employee” or “Project” entity alone then it will not make any sense because he has to manage both. In these cases the relationship of two entities acts as one entity. In our example, the relationship “Works-On” between “Employee” & “Project” acts as one entity that has a relationship “Manages” with the entity “Manager”

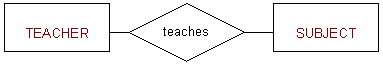


The *degree* of a relationship is the number of entity types that participate in the relationship. The three most common relationships in ER models are *Binary, Unary*and*Ternary*

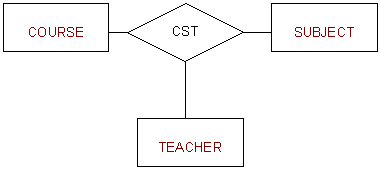
A unary relationship is when both participants in the relationship are the same entity.



A binary relationship is when two entities participate and is the most common relationship degree



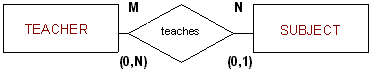
A ternary relationship is when three entities participate in the relationship



Cardinality expresses the minimum and maximum number of entity occurrences associated with one occurrence of the related entity. In the ERD, cardinality is indicated by placing the appropriate numbers beside the entities, using the format (x,y). The first value represents the minimum number of associated entities, while the second value represents the maximum number of associated entities

However, keep in mind that the DBMS cannot handle the implementation of the cardinalities at the table level—that capability is provided by the application software or by triggers

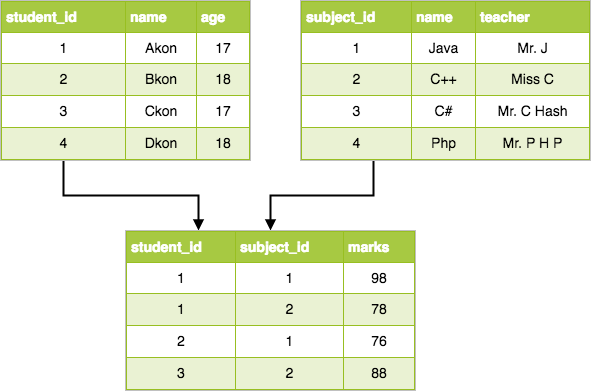
connectivity is used to describe the relationship classification. Just to show the relation existence and type of relation between entities.



Mandatory/Optional Relationships:

Participation by an entity in a relationship may be optional or mandatory.

**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.
* 
* Conceptually simple
* In relational model data is store in the **relation** called **table**(combination of rows and columns).
* Each row in a table is called a **tupple**. Each column in a table is called a attribute.
* **Relational diagram** is used to represent relational database’s entities, attributes and relationship between entities.
* To access the data from relational database **SQL(structure query language)** is used. SQL allow user to specify what must be done without specifying how it must be done.