Shiksha Mandal's

G. S. College of Commerce, Wardha

DEPARTMENT OF B. COM. COMPUTER APPLICATION

BCCA Part-II SEM-III

Database Management System

Unit –II

By

Prof. Harsha N. Gangavane

Dr. Revati Bangre
Co-ordinator

Dr. Anil Ramteke
Principal (Officiating)

Unit II

• Data models

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.

• Classification of Data Model

Data Model may be classified base on the level at which it is used as well as the type of the data model. The classification based on

Level 1: User views or the external views

Level 2: Conceptual Data Model

Level 3: Internal Data Model

Level 4: Physical Data Model

- **Level 1: User views or the external views**

The different subschemas of the data base are called as user views or external views. In this case of a conventional business system. In the relational model, the **external schema** also presents data as a set of relations.

An external schema specifies a **view** of the data in terms of the conceptual level. It is tailored to the needs of a particular category of users. Portions of stored data should not be seen by some users and begins to implement a level of security and simplifies the view for these users

Examples:

- Students should not see faculty salaries.
- Faculty should not see billing or payment data.

Information that can be derived from stored data might be viewed as if it were stored.

- **Level 2: Conceptual Data Model**

It is combined view of all user views. This gives a complete view of the entire database. Also referred to as the Logical level. Hides details of the physical level.

- In the relational model, the conceptual schema presents data as a set of tables.

The DBMS maps data access between the conceptual to physical schemas automatically.

- Physical schema can be changed without changing application:
- DBMS must change mapping from conceptual to physical.
- Referred to as **physical data independence**.

- **Level 3: Internal Data Model**

If the conceptual data model is into any one of the three data models (HDM, NDM, RDM) then it is called as internal data model.

- **Level 4: Physical Data Model**

It is final level of the data model. This is derived from the internal model after integrating access method in it and defining all storage level specifications. The **physical schema** describes details of how data is stored: files, indices, etc. on the random-access disk system. It also typically describes the record layout of files and type of files.

Problem:

- Routines are hardcoded to deal with physical representation.
- Changes to data structures are difficult to make.
- Application code becomes complex since it must deal with details.
- Rapid implementation of new features very difficult.

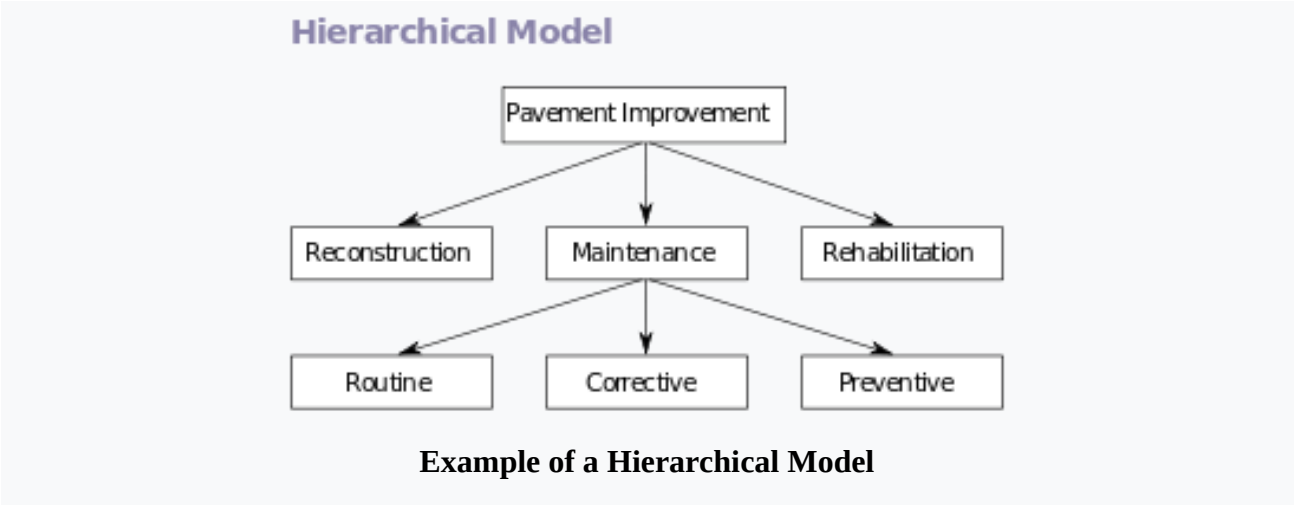
- **Data Models can be classified based on the type of Data. (Any one of these model may be required in real life situation)**

1. Hierarchical Data Model
2. Network Data Model
3. Relational Data Model

- **Hierarchical Data Model**

This model consist of set of nested relationships having one-to-many (1: M) and/ or one-to-one (1: 1) Associations. A **hierarchical database model** is a data model in which the data is organized into a tree-like structure.

The data is stored as **records** which are connected to one another through **links**. A record is a collection of fields, with each field containing only one value. The **entity type** of a record defines which fields the record contains.



Example of a Hierarchical Model

A record in the hierarchical database model corresponds to a row (or tuple) in the relational database model and an entity type corresponds to a table (or relation).The hierarchical database model commands that each child record has only one parent, whereas each parent record can have one or more child records. In order to retrieve data from a hierarchical database the whole tree needs to be traversed starting from the root node.

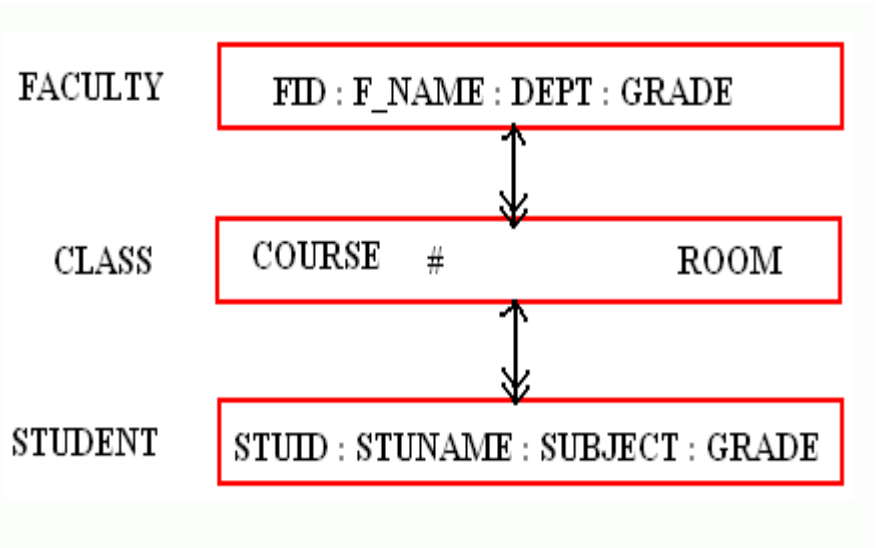


Figure 4

- **Figure 4 shows** a data structure diagram for a tree representing the STUDENT, FACULTY and CLASS.
- The root node chosen is faculty, CLASS as a child of faculty and STUDENT as a child of class.
- The cardinality between CLASS and FACULTY is one to many cardinality as a FACULTY teaches one or more CLASS.
- The cardinality between a CLASS and a STUDENT is also one to many cardinality because a CLASS has many STUDENTS.

Figure 5 shows an **occurrence of the FACULTY-CLASS-STUDENT**.

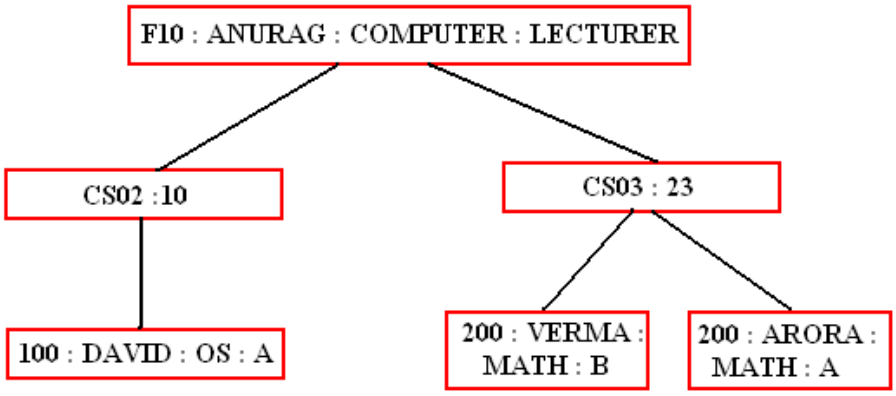


Figure 5

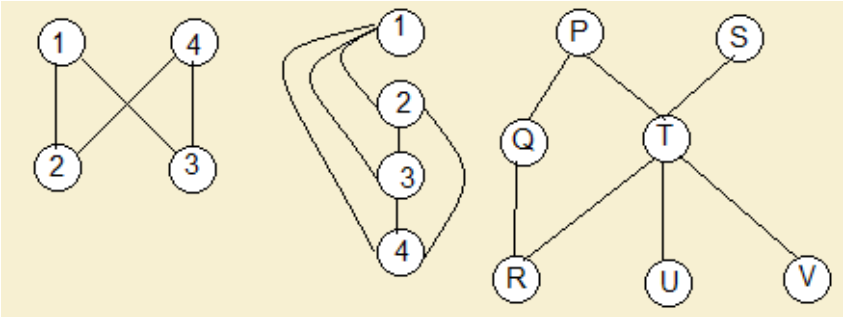
- **Advantages of Hierarchical Model**
  - Easy to understand
  - Performance is better than relational data model
- **Disadvantages of Hierarchical Model**
  - Difficult to access values at lower level
  - This model may not be flexible to accommodate the dynamic needs of an organization
  - Deletion of parent node result in deletion of child node forcefully
  - Extra space is required for the storage of pointers

- **Network Data Model of**

A Network data model consist set of files and pairwise association between the files. Every conceptual data model is almost like a network data model. The Network Data Model can be classified in to three types they are follows

- Simple Network Data Model.
- Complex Network Data Model.
- Limited Network Data Model.

The network data model uses plexus structure as its basic data structure. A network is a directed graph consisting of nodes connected by links or directed arcs. The nodes correspond to record types and the links to pointers or relationships. All the relationship are hardwired or pre-computed and build into structure of database itself because they are very efficient in space utilization and query execution time. The network data structure looks like a tree structure except that a dependent node which is called a child or member, may have more than one parent or owner node.



Advantages of Network Model/ Network Database :

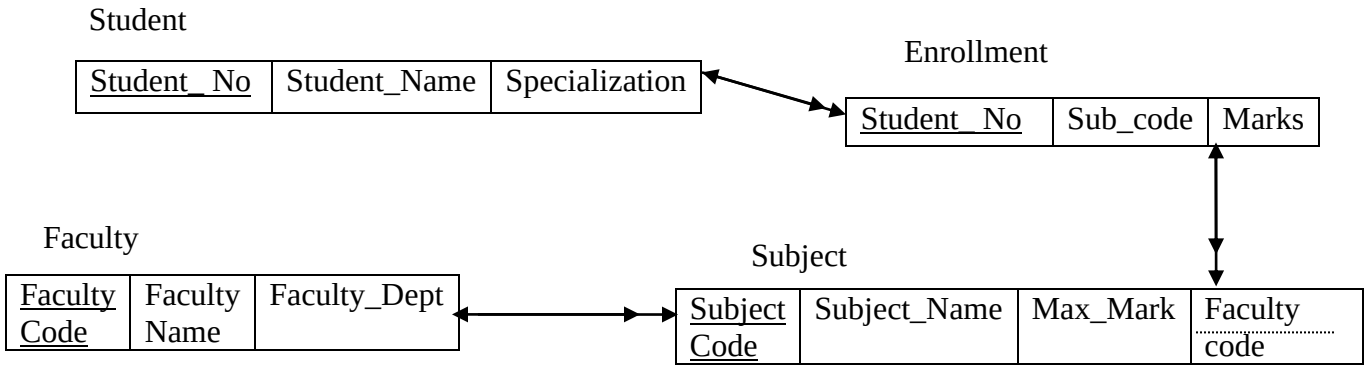
- Easy access to data.
- Flexible
- Efficient
- This model can be applied to real world problems, that require routine transactions.

Disadvantages of Network Model/ Network Database :

- Complex to design and develop.
- Extra memory is required for storage of pointers
- Performance is inflexible and difficult to use.
- Operation and maintenance are time consuming and expensive for large databases.

- **Simple network data model**

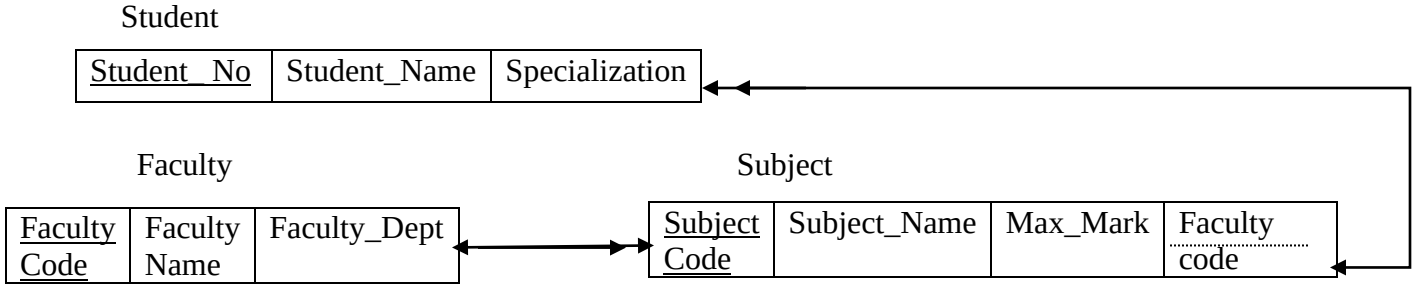
A simple network data model consists of a set of files with one-to- one and/ or one-to-many pair – wise association. Many-to-Many association is not permitted in the simple network data model. If the reality requires a network data model, then one has to map the conceptual data model. If the reality into simple network data model for easy implementation.



An Example of Simple Network Data Model

- **Complex Network Data Model**

If a network data model has at least one Many-to-Many Association, then it is called a Complex Network Data Model. This will make the model very difficult to implement. Hence this type of model exists only in theory. In reality, if there is any such model, it is always the practice of converting each and every Many-to-Many associations with a meaningful intersection file having concatenated key.



An Example of Complex Network Data Model

In this figure the association between the Student file and Subject file is many-to many. Hence this network data model, it will not be possible to represent the actual mark obtained by student in a given subject. This is the difficulty

Relational data model

The relational model is a lower level model. It is based on the concept of a relation, which is physically represented as a **table**. A table is a collection of rows & columns. The relational model uses a collection of tables to represent both data and the relationships among those data. The tables are used to hold information about the objects to be represented in the database. A relation or a table is represented as a two-dimensional form in which the rows of the table corresponds to individual records and the columns corresponds to attributes. Each row is called a **tuple** and each column is called an **attribute**. For example, a student relation is represented by the STUDENT table having columns for attributes S. No, S.NAME and Status and City

S.No	S.Name	Status	City
S1	Smith	40	London
S2	Jhon	20	London
S3	Smith	30	Nizeria

S.No : Key
Number of Records = Cardinality
Number of Fields = Arity
Student (S.No,Name,Branch) = Relational Schema (Table Abstraction)

The S.No here is the primary key as it identifies a student record or tuple uniquely.(A primary key is the key applied on an attribute(S.No) which recognize a tuple. The Cardinality of the Relation or table is defined as the number of records in the STUDENT relation which is 4. The Arity is defined as the number of fields or columns in the relation.

- **Domain of an Attribute -**

Domain of an attribute is the set of allowable values for that attribute. It is a pool of values from which the actual values appearing in a given column are drawn. For example, the values appearing in the S.No column are drawn from the domain of all S.No. Domains may be distinct, or two or more attributes may have same domain.

- **Operations in Relational Model -**

1. Insertion - A new student record can be easily inserted in the table.
2. Deletion - An existing student record or tuple can easily be deleted from the STUDENT relation.

3. Updation - An existing student record can be update easily. For example, if a student S2 changes its BRANCH from CS to IT, then it can easily be changed

➢ **Advantages of Relational Model -**

- Easy to use an understand
- Very flexible.
- Widely used.
- Provides excellent support for adhoc queries.
- Users need not consider issues such as storage structure and access strategy.
- Specify control and authorization can be implemented more easily.
- Data independence is achieved more easily with normalization structure used in a relational database.

- **Disadvantages of Relational Model -**

- For large databases, the performance in responding to queries is definitely degraded.
- The processing requirements need to construct the indexes. So, the index position of the file must be created and maintained along with the file records themselves.
- The file index must be searched sequentially before the actual file records are obtained. These wastes time.

- **Entity Relationship Model**

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.

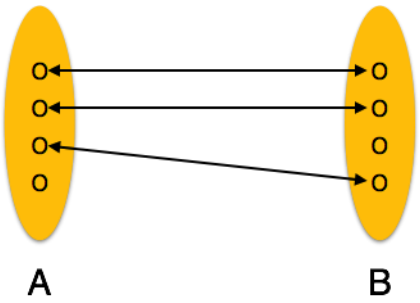
- **Attributes**

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.

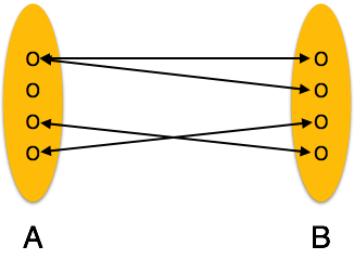
- **Mapping Cardinalities**

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.

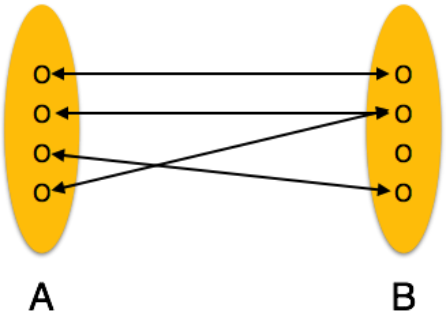
- **One-to-one** − One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



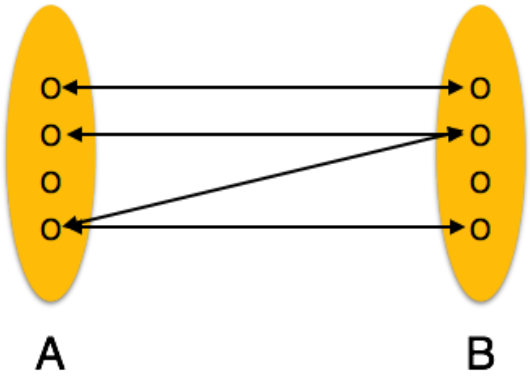
- **One-to-many** − One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



- **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 Model**

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**

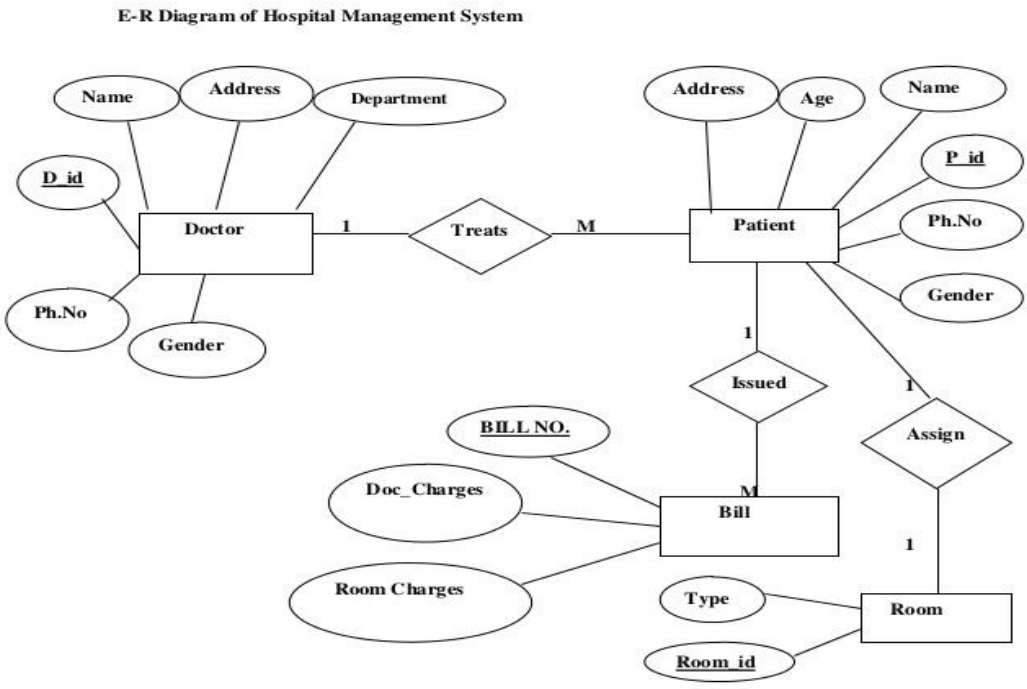
Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

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.

- **ENTITY RELATIONSHIP MODEL**

It is a special form of a network data model augmented with a diamond symbol to indicate the name of the association between the meaningful pairs of relations in the conceptual data model.

The fields of each relation are shown within ellipses which are placed around that relation. The name of each relation will be in rectangle. In short, it is called as E_R diagram of the conceptual data model.



(Fig.), which includes the type of association like1:1 or 1:M,

The one- to- many (1:M) association from the student relation to the Enrollment relation is indicated by writing 1 on the line closer to the student relation and M on the line closer to the Enrollment relation. Similarly, the other associations are indicated in Fig.4.8.

This E-R diagram helps the database designer to have better insight of the model. This will in turn help the database designer to implement the conceptual data model into any of the three data models with accuracy and consistency.

- # Symbols used for Creating ER-Diagram

- **Entities:**

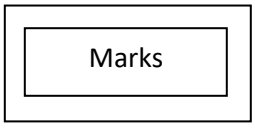
An entity is an object or concept about which user want to store information. An entity is an object that exists and is distinguishable from other objects. For eg. Specific person, company, event, plant, etc. entities are of two types

1) Regular entities or strong entities

A strong entity is an entity whose existence does not depends on existence of another entity. Eg. Student or employee. It is denoted by rectangle

2) Weak entities

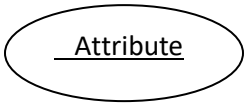
A weak entity is an entity whose existence depends on another entity. It is defined by a foreign key relationship with another entity. It cannot be uniquely identified by its own attributes alone. For example marks entity existence which contains student‟s marks depends on entity student. It is denoted by double outlined rectangle.



- **Types of Attributes:**

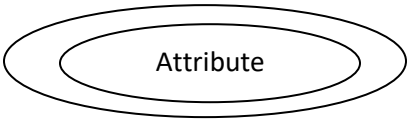
a) Key attributes

A key attribute is the unique, distinguishable characteristic of the entity. For example, an employee’s number might be the employee’s key attribute.



b) Multivalued attribute

A Multivalued attribute can have more than one value. For example, an employee entity can have multiple skill values.

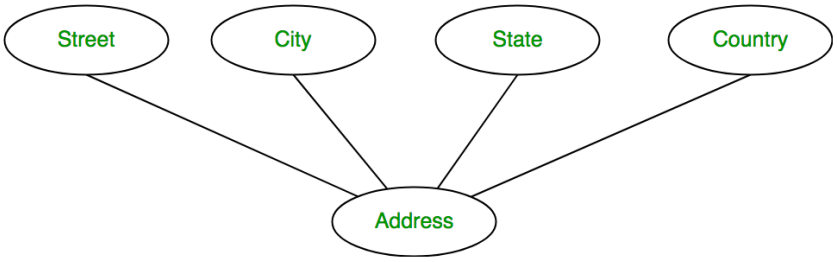


c) Derived attribute:

A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary

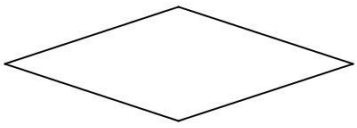
d) Composite Attribute

"Composite attribute is an attribute where the values of that attribute can be further subdivided into meaningful sub-parts." Typical examples for composite attribute are; Name – may be stored as first name, last name, middle initial.



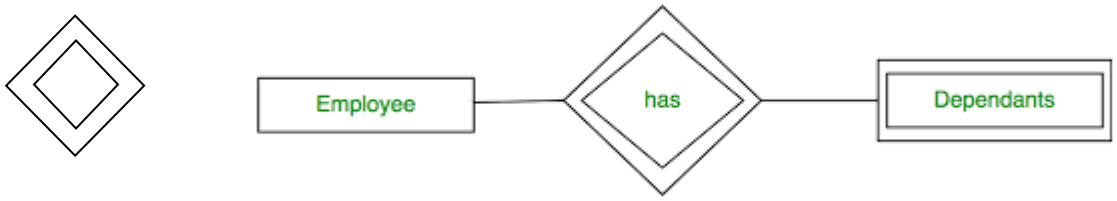
C) Relationship:

- Relationship are in association of entities and E-R relationship can be one to one or one to many or many to many. It is denoted by rectangle

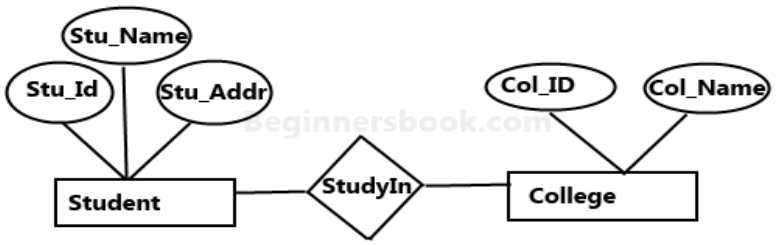


- **Weak Entity Relationship**.

  A weak or non-identifying relationship exists between two entities when the primary key of one of the related entities does not contain a primary key component of the other related entities.



Example of E-R Diagram



Sample E-R Diagram

Database Design

Database design is the process of constructing a stable database structure from user requirement analysis. The database design is considered to be the most important task while following database approach for a reality

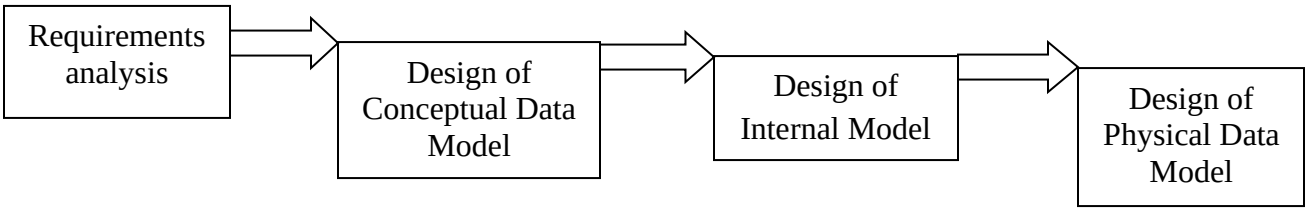
The database designs structure the grouping of fields into different files and then establishes meaningful association between different files in an optional manner which helps to minimize the response time while accessing and manipulating the database during its use. Once's the basic need s are identified the conceptual data model can be created

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

The main objectives of database designing are to produce logical and physical designs models of the proposed database system. The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically. The physical data design model

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

➢ Steps of Database Design



The steps of database design are listed below

- Requirements analysis
- Design of conceptual data model
- Design of internal model This is alternatively known as implementation design
- Design of physical data model

➢ Requirements Analysis

The first step of database design is to perform requirement analysis. The requirements of user of the proposed system are to be identified either through interviews with member concerned or through different report that are presently used by user. This means that the format of each and every report is to be collected along with related processing requirements such as periodically and nature of urgency of every report

Another way to identify the report

- Name of report
- List of data items from the report which is to be stored in the database
- The association between different data items in the report

Then for each data elements in each of the reports the following should also be identified

- Name of the data items
- Type of the data item (Numeric, String, date etc.)
- Width of the data item
- Update authority for data item

- Allowable range if the data item is numeric
- Source of the data item

➢ Design of Conceptual data

In the process of designing a DBMS independent data model which gives a stable comprehensive view of the entire database. It is logical data model the basic input for this step is the set of relations of different reports (user view)

The steps for designing a Conceptual Data Model

- Input the requirement specification in the form of relation
- Apply the normalization on each and every relation up to a specified level of normal form
- Perform view integration
- Construct the Conceptual data model
- Design a LOGICAL ACCESS MAP (LAM) for each user

➢ Design of Internal Model / Implementation Design

The Conceptual Data Model is independent of the data base environment after designing the Conceptual Data Model it should be mapped (implemented) into any one of the following data models based on the reality

- Hierarchical Data Model
- Network Data Model
- Relational Data Model

Selecting the right type of model to suite the reality ia an important task, since it will improve the efficiency of operation of the system

➢ Design of Physical Data Model

After designing the internal data model, the next step is to design a physical data model it is concerned with designing /deciding the following

- Stored record formats
- Selecting access methods
- Database security
- Integrity

- Back Procedure
- Recovery Methods

➢ Normalization and different normal forms:

Normalization is the analysis of functional dependency between attribute /data items of user views. It reduces a complex user view to a set of small and stable subgroups of Fields /Relations This process helps to design LOGICAL DATA MODEL known as Conceptual Data Model. Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

Normalization is a scientific method of breaking down complex table structures into simple table structures by using certain rules. Using this method, you can, reduce redundancy in a table and eliminate the problems of inconsistency and disk space usage. You can also ensure that there is no loss of information.

Normalization has several benefits. It enables faster sorting and index creation, more clustered indexes, few indexes per table. Few nulls and makes the database compact. Normalization helps to simplify the structure of tables. The performance of an application is directly linked to the database design. A poor design hinders the performance of the system. The logical design of the database lays the foundation for an optimal database.

Properties of relational databases

- Values are atomic.
- All of the values in a column have the same data type.
- Each row is unique.
- The sequence of columns is insignificant.
- The sequence of rows is insignificant.
- Each column has a unique name.
- Integrity constraints maintain data consistency across multiple tables.

Some rules that should be followed to achieve a good database design are:

- Each table should have an identifier
- Each table should store data for a single type of entity.
- Columns that accept nulls should be avoided
- The repetition of values or columns should be avoided

Normalization results in the formation of tables that satisfy certain specified rules and represent certain normal forms. The Normal forms are used to ensure that various types of anomalies and inconsistencies are not introduced in the database. A table structure is always in a certain normal form. Several normal forms have been identified. The most important and widely used normal forms are it is also known as levels of Normalization:

- Unnormalized form (UNF)
- First Normal form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal form (4NF)
- Fifth Normal form (5NF)

• Unnormalized form (UNF)

This the relation in which satisfied all the properties of relation. It is a simple **database** data model (organization of data in a **database**) lacking the efficiency of **database** normalization?

• First Normal Form (1NF)

A relation said to be in the first normal form if it is already in unnormalized form and it has no repeating group. The definition of 1 NF says that all the attributes in a relation must have atomic i.e. simple and indivisible values from their respective domains. Multi-valued attributes, composite attributes, nested relations are split such that they can be made atomic. Nested relations are the relations within relations.

A table is said to be in the 1 NF when each cell of the table contains precisely one value. Consider the following table Project.

Ecode	Dept	Depthead	ProjCode	Hours
E101	System	E901	P27 P51 P20	90 101 60
E305	Sales	E906	P27 P22	109 98
E508	Admin	E908	P51 P27	Null 72

The data in the table is not normalized because the cells in projcode and Hours have more than one value. By applying the 1NF definition to the Project table you arrive at the following table.

Ecode	Dept	DeptHead	ProjCode	Hours
E101	System	E901	P27	90
E101	System	E901	P51	101
E101	System	E901	P20	60
E305	Sales	E906	P27	109
E305	Sales	E906	P22	98
E508	Admin	E908	P51	Null

- **Functional Dependency**

The Normalization theory is based on the fundamental notion of functional dependency. First let‟s examines the concept of functional dependency.

Given a relation R, attribute A is functionally dependent on attribute B if each value of A in R is associated with precisely one value of B.

In other words, attribute A is functionally dependent on B if and only if, for each value of B, there is exactly one value of A

Attribute B is called the determinant.

Consider the following table Employee

Code	Name	City
E1	Mac	Delhi
E2	Sandra	CA
E3	Henry	Paris

Given a particular value code, there is precisely one corresponding value for Name. for example, for Code E1 there is exactly one value of Name, Mac. Hence, Name is functionally dependent on Code.

Similarly, there is exactly one value of City for each value of Code hence the attribute City is functionally dependent on the attribute Code. The attribute code is the determinant. you can also say that Code determines City and Name.

- **Second Normal Form (2NF)**

A relation is said to be in the second normal form if it is already in the first normal form and it has no partial dependency.

A relation is said to be in 2NF if it is in 1 NF if it is in 1NF and every non-key attribute of the relation is full, functionally dependent on the key attribute. By fully functional dependency, we mean that if any attribute from the key attributes is removed, the dependency is not preserved. Consider the Project table:

Ecode	ProjCode	Dept	DeptHead	Hours
E101	P27	System	E901	90
E305	P27	Finance	E909	10
E508	P51	Admin	E908	Null
E101	P51	System	E901	101
E101	P20	System	E901	60
E508	P27	Admin	E908	72

- **Insertion:**

  The department of a particular employee cannot be recorded until the employee is assigned a project

- **Updation:**

  For a given employee the employee code, department name, and department head are repeated several times. Hence if an employee is transferred to another department, this change will have to recorded in every row of the employee table pertaining to that employee. Any omission will lead to inconsistencies.

- **Deletion:**

  When an employee completes work on a project, the employee's record is deleted. The information regarding the department to which the employee belongs will also be lost.

➢ **Third Normal Form (3NF):**

Third normal form is applied when the relations are in 2NF and there is any non-key attribute that depends transitively on the primary key. The attributes are said to be transitively dependent it. Consider the table Employee;

Ecode	Dept	DeptHead
E101	System	E901
E305	Finance	E909
E402	Sales	E906
E508	Admin	E908
E607	Finance	E909
E608	Finance	E909

The problems with dependencies of this kind are:

a. Insertion

The department head of a new department that does not have any employees at present cannot be entered in the deptHead column. This is because the primary key is unknown.

b. Updation

For a given department the code for a particular department head is repeated several times hence if a department head moves to another department, the change will have to be made consistently across the table.

c. Delation

If the record of an employee is deleted, the information regarding the head of the department will also be deleted hence, there will be a loss of information.

- **Boyce-codd Normal Form**

The original definition of 3NF was inadequate in some situations. It was not satisfactory for the tables

- That had multiple candidate keys
- Where the multiple candidate keys were composite
- Where the multiple candidate keys overlapped.

Therefore, a new normal form, the Boyce-Code normal form was introduced. You must understand that in tables where the above three condition do not apply you can stop at the third normal form. In such cases the third NF is the same as the Boyce-Codd normal form.

- **Fourth and Fifth Normal Form (4NF and 5NF)**

The relations can be further decomposed to 4NF and 5NF if still there exist multi-valued dependencies. However, this decomposition may lead to increased overhead on the system like maintaining the complex relationship between the tables, indexing mechanism, etc.

- **To Understand the Dependencies**

1. **Partial Dependency: -** In relation having more than one key field, a subset of non-Key fields may depend on all key fields but another subset/ a particular non key field(s) may depend on only one of the key fields (i.e., may not depend on all the key fields). Such dependency is called partial dependency
2. **Transitive Dependency:** - In a relation, there may be dependency among non-key fields. Such as dependency is called as transitive dependency.
3. **Determinant:** - A determinant is any field (Simple field or composite field) on which some other field is fully functional dependent.

4. **Multivalued Dependency:** - Consider three fields X.Y and Z in a relation. If for each value of X, there is a well-defined set of values of Y and a well- defined set of values of Z and the set of values of Y is independent of the set of values of Z, then multivalued dependency exists.
5. **Join Dependency:** - A relation which has a join dependency cannot be decomposed by projection into other relation without any difficulty and undesirable results. The occurrence of this type of dependency is very rare.

➢ **Case Study Problem: - Integrated Case Study – Database Design for Academic Institution (Please refer this case study from your text book of Database Management System and solve the following)**

Case Study Problem: -

1. **Integrated Case Study – Database Design for Pharmaceutical Shop (Medical shop)**
2. **Integrated Case Study – Database Design for Sper Market**
3. **Integrated Case Study – Database Design for Academic Institution**

➢ Data Volume and Usage Analysis

- **Data Volume Analysis: -** Data Volume Analysis estimates the number of records in each of the files in the conceptual data model
- **Data Usage Analysis:** - After the data volume analysis, the usages pattern of the database is to be analyzed. This involves the following step
  - Defining the database transactions
  - Constructing Logical Access Map (LAM) for each data base transaction

- **Database Transaction: -** To print each and every report for which the database is designed, now the required logical steps must be stated. This sequence steps are called as database transaction

- **Logical Access Map: -** This gives a kind of flowchart showing the access path on the conceptual data model as per database transaction. This can be classified into
  1. Preliminary Map
  2. Final Map

Preliminary Map: - This Map shows the sequence of access path on conceptual data model without altering the conceptual data model.

Final Map: - The Final Map is again a sequence of access path of conceptual data model with modification in the conceptual data model structure itself by refining various associations on it.

Implementation Design

- **Introduction**

The conceptual data model which has been presented in the earlier. The implementation phase is where you install the DBMS on the required hardware, optimize the database to run best on that hardware and software platform, and create the database and load the data. Tune the setup variables according to the hardware, software and usage conditions. Create the database and tables.

The following are steps in the implementation.

1. Install the DBMS.
2. Tune the setup variables according to the hardware, software and usage conditions.
3. Create the database and tables.
4. Load the data.
5. Set up the users and security.
6. Implement the backup regime.

Implementation Design

Implementation design is an important step of the database design which decides the selection of the best fitting data model and then maps the conceptual data model into the selected data model.

Step 1: Input the following:

1. DBMS independent schema/Conceptual Data Model
2. Data Volume and usage details.
3. Characteristics of selected DBMS data model.

Step 2: Map Conceptual data model into the desired logical data model based on the guidelines of each type of the data models.

Step 3: Design subschemas of different user of the system

Step 4: Prepare program design guideline for each subschema

Step 5: Perform review of design and carry out corrections if necessary.