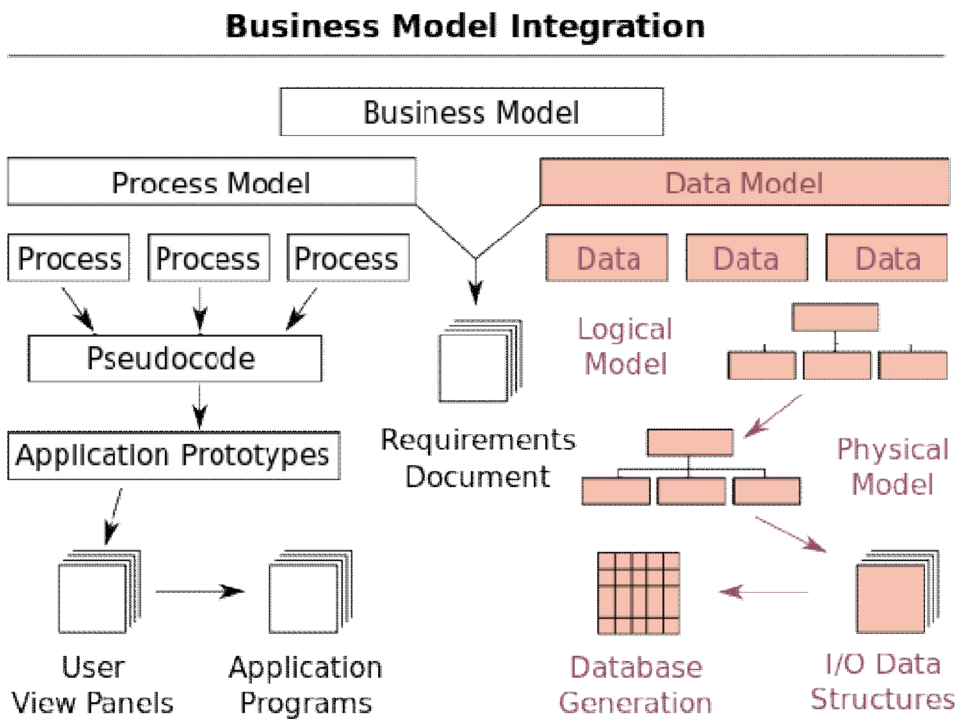
Data Model in DBMS

A model is an abstraction process that represent essential features without including the background details or explanations. It hides superfluous details while highlighting details pertinent to the application at hand. A data model is a mechanism that provides this abstraction for database applications. Data modelling is used for representing entities of interest and their relationships in the data base. A data model defines the logical structure of a data base means that how data is connected to each other and how they are processed and stored inside a system.



A number of models for representing data have been developed. As with programming languages, there is no best choice for all applications but the models maintains the integrity of the by enforcing a set of constraints. Data models differ in their method of representing the associations amongst entities and attributes. The main models or approach are:

The Hierarchical Model - Tree Structure
The Network Model - Plex Structure
The Relational Model - Normalised Structure
The ER Model -Conceptual Model

Data Model Structure and Constraints -

To define the database structure, Constructs are used
Constructs typically include elements (and their data types) as well as group of elements (Example- Entity, Record, Table), and relationships among such groups.
Constraints specify some restriction on valid data; These constraints must be enforced at all times.

Data Model Operations -

These operations are used for specifying database retrievals and updates by referring to constructs of the data model. The Operations may include basic model operations as well as user defined operations.

Basic Model Operations :

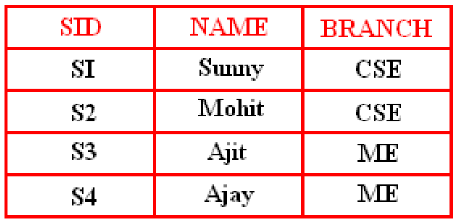
Insert
Delete
Update

User Defined Operations Example :

compute_student_CGPA
compute_student_marks
compute_student_SGPA
update_inventory etc.
The hierarchical model is the oldest DBMS model and the object oriented DBMS being the newest data model.

Relational Model or Relational Database - (Part 1)

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 SID, NAME and BRANCH.



SID	NAME	BRANCH
SI	Sunny	CSE
S2	Mohit	CSE
S3	Ajit	ME
S4	Ajay	ME

SID : Key
Number of Records = Cardinality
Number of Fields = Arity
Student (SID,Name,Branch) = Relational Schema (Table Abstraction)
The SID here is the primary key as it identifies a student record or tuple uniquely.(A primary key is the key applied on an attribute(SID) 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 SID column are drawn from the domain of all SID. 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 normalisation 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. This wastes time.

Database Schema and Database Instance - Relational Model (Part2)

Database Schema -

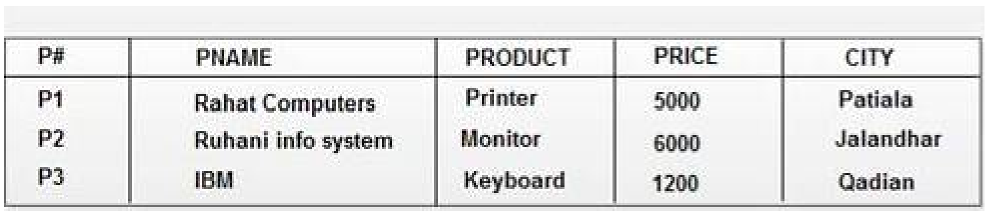
Database Schema is the **overall Design of the Database**. It is the **skeleton structure** that **represents** the **logical view** of the entire database. It **tells** how the data is organized and how the relations among them are associated.
It is sometimes also referred to as an overall model of the data, **a conceptual model** or a **conceptual schema.** These terms mean essentially the same thing. The Database Schema also formulates all the constraints that are to be applied on the data.
A schema is a chart of the types of data that are used. It **gives the names of the entities and attributes**, and **specifies the relation** between them. It is just like a **framework** into which the values of data items can be fitted/stored. All the constraints that are to be applied on the data is also formulated on the schema.

The Database Schema will look like this :

Before describing the Instance of Schema, let us define some terms -
Database State: Refers to the content of a database at a moment in time.
Initial Database State: Refers to the database when it is loaded
Valid State: A state that satisfies the structure and constraints of the database.
Distinction
The **database schema** changes very infrequently. The **database state** changes every time the database is updated.
Schema is also called **intension**, whereas **state/instance of schema** is called **extension**.

Database Instance of Schema -

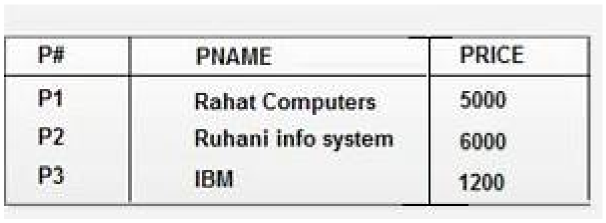
The actual content of the database or say the data at a particular instant is called the **Instance of Schema.**
Database instances tend to change with time.
The Database Management software will ensure that the data or instance filled into the database scheme is in a valid state, by checking all the constraints, validations and conditions that the database designers have imposed.
The instance of above Schema will look like this :



P#	PNAME	PRODUCT	PRICE	CITY
P1	Rahat Computers	Printer	5000	Patiala
P2	Ruhani info system	Monitor	6000	Jalandhar
P3	IBM	Keyboard	1200	Qadian

A database Schema can be divided into two categories -

Physical Database Schema : The Schema which is related to actual storage of data or the Schema that describes database design at physical level is called Physical Database Schema. It defines how the data will be stored in the secondary storage.
Logical Database Schema : The schema that defines all the logical constraints that need to be applied on the data stored or the schema that describes the database design at the logical level is called Logical Database Schema. It defines tables, views, and integrity constraints.
A database may also have several schemas at the view level called subschemas, that describe different views of the database.
SubSchemas -
A subschema inherits the same property that a schema has and is referred to as a subset of the schema.
It is the application programmer's (user's) view of the data which he or she uses and gives the users a window through which he or she can view only that part of the database which is of interest to him. Therefore, different application programs can have different view of data.



P#	PNAME	PRICE
P1	Rahat Computers	5000
P2	Ruhani info system	6000
P3	IBM	1200

Schema Construct -
A component of the schema or an object within the schema, e.g. PNAME, PRODUCT, PRICE etc.

Relational Schema	**Relational Instance**
Table Abstraction	Set of tuples
Schema is the overall Design of the Database.	The actual content of the database or say the data at a particular instant is called the Instance of Schema.
Database Schemas will remain the same.	Database instances tend to change with time.
Relational Schema is just like a framework.	When the schema framework is filled in with data item values, it is referred as an instance of the schema.

Relational Database OR Relational Model-(Part-3)

Relations -

Let A_1, A_2,....,A_n be the attributes with domains D_1, D_2, ..., D_n. A relational schema R consists of set of attributes with their corresponding domains. A relation r is a set of n tuples(A_1:d_1, A_2:d_2, ..., A_n:d_n) such that $d_1 \varepsilon D_1$, $d_2 \varepsilon D_2$,..., $d_n \varepsilon D_n$. We write the relation as a table in which the column headings correspond to attributes and then write the values chosen from the appropriate domains into the rows or tuples. So, we think of the n-tuples as having the form(d_1, d_2, ..., d_n). Thus we conclude that a relation in the relational model a relation in the relational model can be viewed as any subset of the cartesion product of the domains of the attributes.

Properties of a relation -

- In a Table, the order of rows and Columns is immaterial.
- There will be no duplicate rows in a table.
- Each row of the table contains only one value.
- Value in each column in the table must come from the same attribute domain.

Degree and Cardinality :

Degree of a relation -

Degree of the relation is the number of columns in the table.For example, the degree of the employee relation is 5. A relation with one column is called **unary relation.** A relation with two columns is **binary relation** and with three columns is **ternary relation** and so on. Similarly, if a Table or a relation have n columns, them the relation will be called as **n-ary relation**.

Cardinality of a relation -

Cardinality is the number of rows in the table. For example, the cardinality of the employee relation is 4.

Representation of Relational Database or Model Structures -

A database consists of any number of relations. For example, the Relation will be like STUDENT, CLASS, FACULTY, ENROLLMENT etc present in a database. The relation schemes is represented by giving the name of the relation followed by its attribute names in parenthesis. For example,
1. STUDENT(sid, sname, subject, marks)
2. CLASS(coursesID, facultyID, courses, room)
3. FACULTY(facultyID, fname, dept)
4. ENROLLMENT(courseID, sid, grade)

Extensions and Intensions -

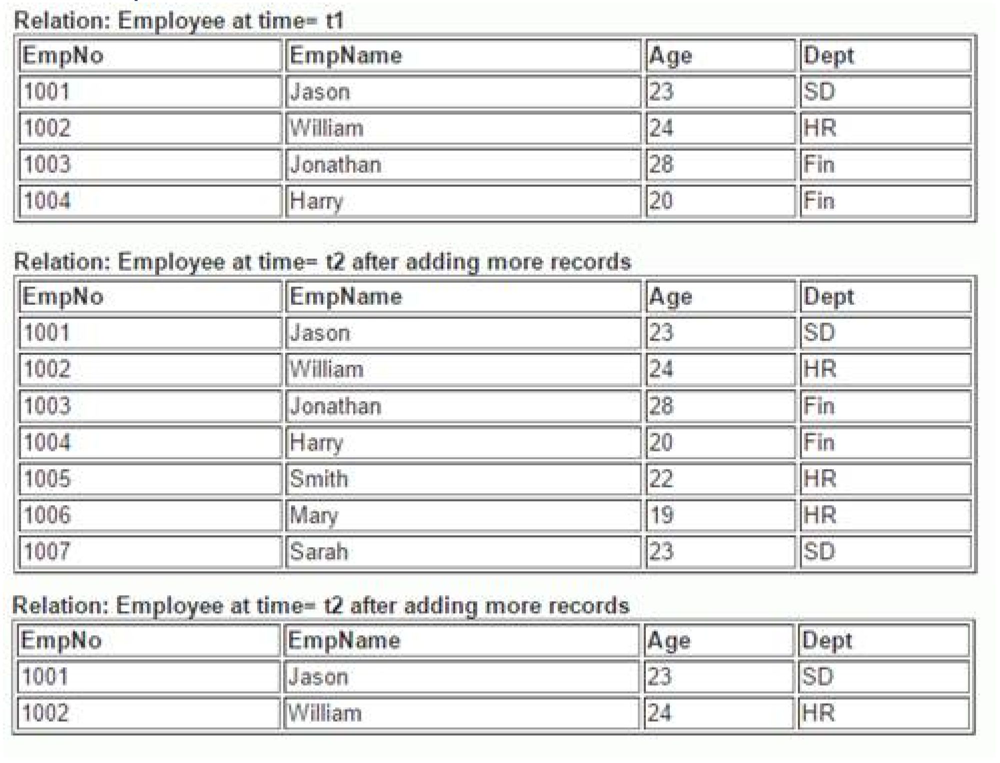
A relation in a relational database or model has 2 components -

- **Extension**
- **Intension**

Extension -

The set of tuples appearing at any instant in a relation, is called the extension of that relation. In other words, instance of Schema is the extension of a relation. The extension varies with time as instance of schema or the value in the database will change with time.

For Example :



Relation: Employee at time= t1

EmpNo	EmpName	Age	Dept
1001	Jason	23	SD
1002	William	24	HR
1003	Jonathan	28	Fin
1004	Harry	20	Fin

Relation: Employee at time= t2 after adding more records

EmpNo	EmpName	Age	Dept
1001	Jason	23	SD
1002	William	24	HR
1003	Jonathan	28	Fin
1004	Harry	20	Fin
1005	Smith	22	HR
1006	Mary	19	HR
1007	Sarah	23	SD

Relation: Employee at time= t2 after adding more records

EmpNo	EmpName	Age	Dept
1001	Jason	23	SD
1002	William	24	HR

Intension -

The intension is the schema of the relation and thus is independent of the time as it does not change once created. So, it is the permanent part of the relation and consists of -

1. **Naming Structure -** Naming Structure includes the name of the relation and the attributes of the relation.
2. **Set of Integrity Constraints -** The Integrity Constraints are divided into Integrity Rule 1 (or entity integrity rule), Integrity Rule 2 (or referential integrity rule), key constraints, domain constraints etc.

For example :
Employee(EmpNo Number(4) Not NULL, EName Char(20), Age Number(2), Dept Char(4))

Network Database or Network Model in DBMS

The network database or network model uses the plex structure as its basic data structure. A network is a directed graph consisting of nodes connected by links or directed arcs. The nodes corresponds 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.All figure shows the network model -

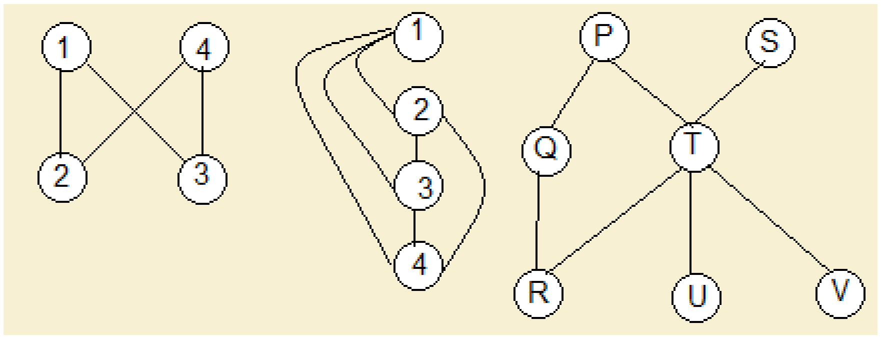


Figure 1

A diagram called as **Bachman Diagram** is used to represent a network data structure. The nodes in the network are replaced by rectangles that represent records and links are shown by lines connecting the rectangles.

A plex structure with two record types is shown / Example of Network Database :

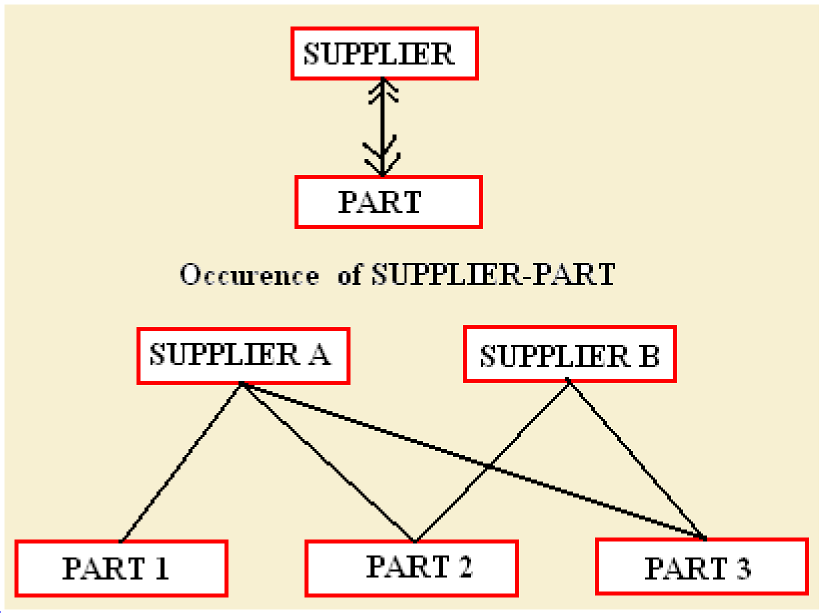
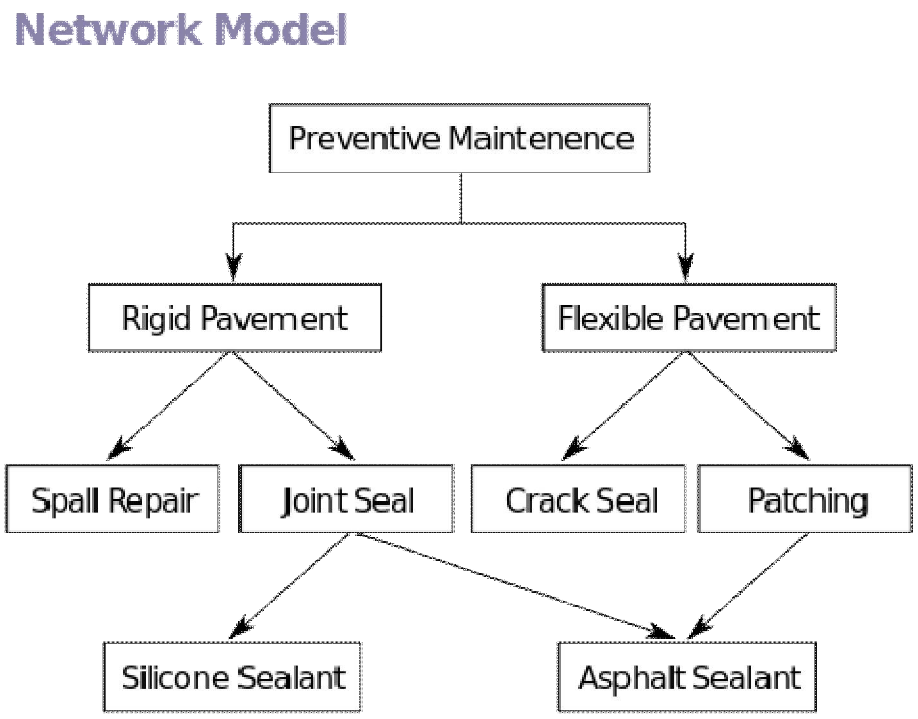


Figure 2

Another Example of Network Database is :



Operations on Network Model/Network Database

1. Insertion- In the above figure, it is clear that a new part or supplier can easily be inserted.
2. Deletion- For deletion only link is to be removed and no information will be lost. For example, to remove PART 2, we delete the connector line between suppliers.
3. Updation- Updation is also easy, for example, suppose SUPPLIER B supplies PART 1 in place of SUPPLIER 2, so, updation will be successfully done by changing the link of SUPPLIER B from PART 2 to PART 1.

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 infexible and difficult to use.
- Operation and maintenance are time consuming and expensive for large databases.

Hierarchical Model in DBMS

Hierarchical model is a data model which uses the tree as its basic structure. So, lets define the basics of the tree.

Basics of Tree :

- A tree is a data structure that consists of hierarchy of nodes with a single node, called the **root at highest level**.
- A node may have any number of children, but each child node may have only one parent node on which it is dependent. Thus the parent to child relationship in a tree is one to many relationship whereas child to parent relationship in a tree is one to one.

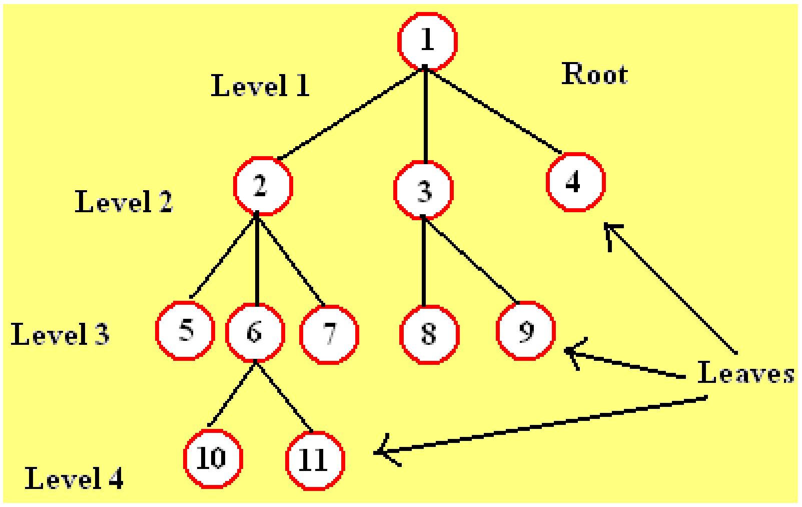


Figure 1

- In figure 1, the node at level 1 is called the root node and the nodes at that has no children are called **leaves.** For example, node 4, 5, 7, 8, 9, 10 and 11.
- Nodes that are children of the same parent are called **siblings**. For example, nodes 2, 3, 4 are siblings.
- For any node there is a single path called the **hierarchical path from the root node**. The nodes along this path are called that nodes **ancestors**.
- Similarly for a given node, any node along a path from that node to leaf is called its **descendent**.
- For example, suppose we have to find out the hierarchical path of node 10, then it will be 1→2→6→10 and the ancestors of node 10 are 1, 2 and 6.
- The **height of tree** is the number of levels on the longest hierarchical path from the root to a leaf. The above tree has a height= 4.
- A **tree is said to be balanced** if every path from the root node to a leaf has the same length.

Figure 2 shows a **balanced and an unbalanced tree**.

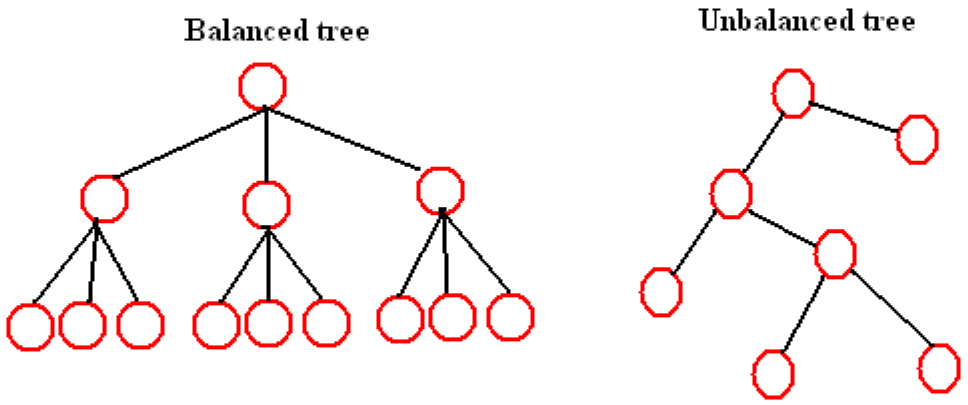


Figure 2

A **binary tree** is one in which each node has not more than two children. Figure 3 shows a binary tree

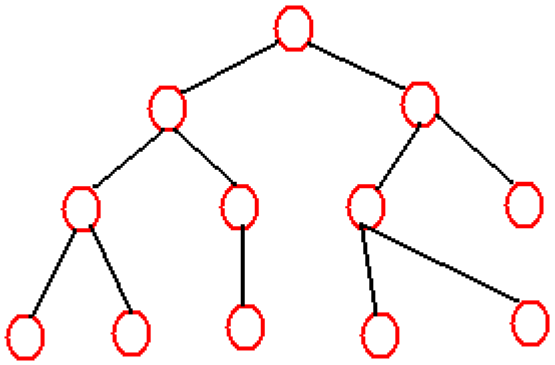


Figure 3

Example of Hierarchical Model :

- **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.

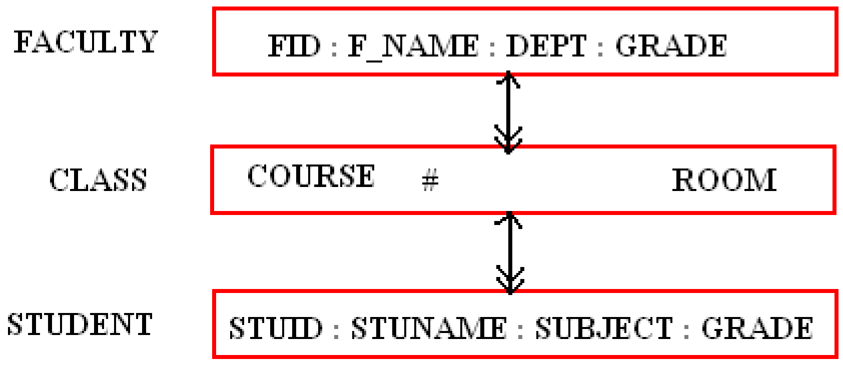

FACULTY
FID : F_NAME : DEPT : GRADE
CLASS
COURSE #
ROOM
STUDENT
STUID : STUNAME : SUBJECT : GRADE

Figure 4

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

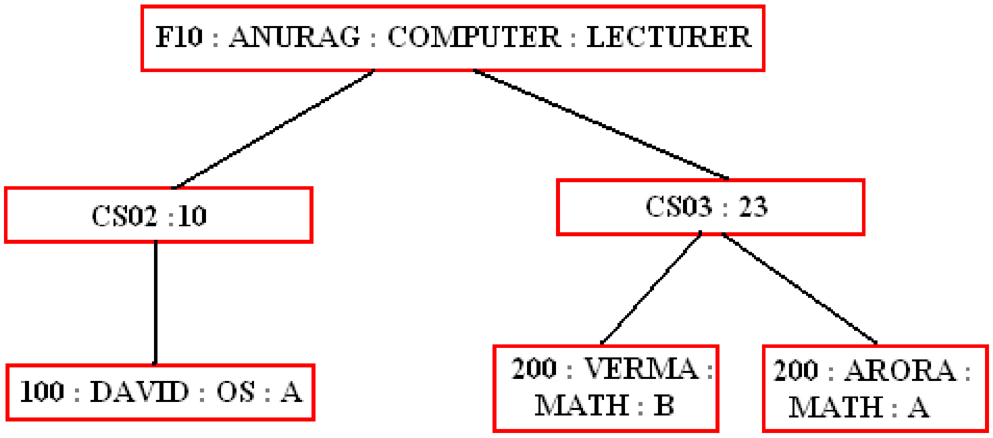
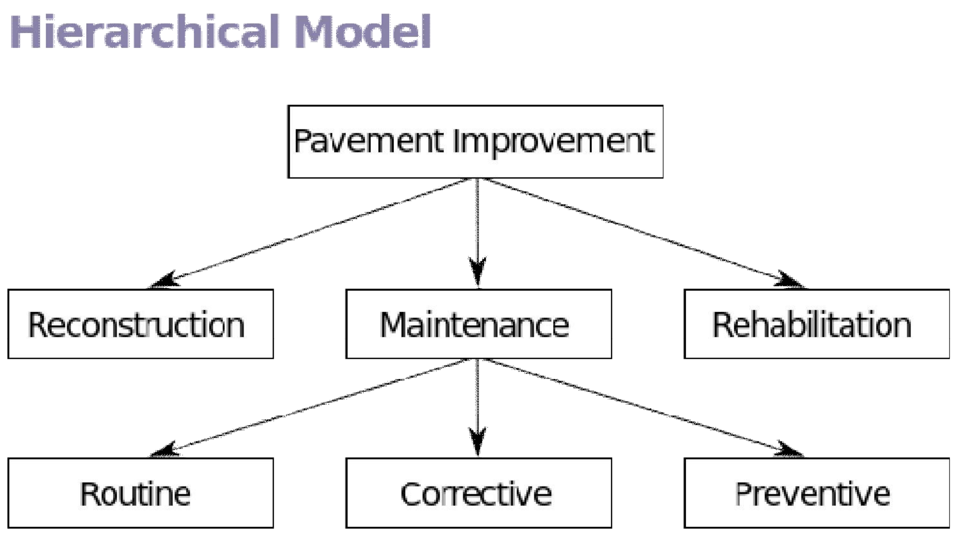


Figure 5

Another Example :



Operations on Hierarchical Model

1. **Deletion-** If CS02 is deleted, then all the students in CS02 class will be deleted. So deletion is very difficult. However deletion of leaf nodes that is students does not create difficulty in deletion.
2. **Insertion-** A new class say, CS03 may not be introduced unless some faculty is available at root level. So insertion is also difficult.
3. **Updation-** Suppose a student has changed his subject from Hindi to Sanskrit, then firstly a search is performed to find out Hindi subject and then an update is made. A search is a time consuming process here.

So these problem occurs in all the three operations.

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 accomodate the dynamic needs of an organisation
- Deletion of parent node result in deletion of child node forcefully
- Extra space is required for the storage of pointers

ER MODEL (Entity Relationship Model)

Before define ER Model, there are some terms which is require to understand ER model properly. In short, ER Model is a diagrammatic representation of entire Database tables. It represents a High Level Database Design. **For Designing the Softwares :** This phase is done in softwares by UML (Unified Modelling Language). **For Designing the Database :** ER Model is used to design database.

Definitions :

Entity : **an object (Record in the table). It a a real world object, such as person, place, thing, event, or concept about which information is recorded. For example, In a banking environment, entities are CUSTOMERS, BANK SUPPLIERS etc. Entities may be -**

- **Physical -** Can be touched(sensed) Example Fan.
- **Conceptual -** Cannot be touched. Example course offered, brand name etc.

Storing conceptual entities is a big problem.

Entity Types : is defined as a collection of entities that have the same attributes. For example, employee in a company database. As same attributes are shared by the employee entities, but such entity has its own value for each attributes.

Entity Set : Collection of similar entities of same type that share the same attribute at any point in time. For example, a Faculty set may contain all the teachers of a college, Set of all persons, Set of Companies etc. The Entity Set is also called **extension** of the Entity Type.

Attribute : It is the name of the column. An attribute gives the characteristics of the entity. It is also called as data element, data field, a field, a data item, or an elementary item. Some examples of Attributes which clear the concept of attributes –

a) A house can be described by its colour, size and surroundings
b) A bank account can be described by its bank account number, account type, account balance, account statement etc.
c) A customer of bank may be described by : name, address, customer ID number, phone number.
So, in example a) colour, size and surroundings are attributes of entity "HOUSE".
In example b) account number, type, balance and statement are attributes of entity "ACCOUNT"
In example c) name address ID number and phone number are attributes of entity "CUSTOMER"

Data Value : It is the actual data or information contained in each data element(attribute). The data element “Supplier Name” can take value as “Anil Mittal”

Difference between Entity, Attributes, and Data Values -

ENTITY	**ATTRIBUTES (OR DATA ELEMENTS)**	**VALUES**
CUSTOMER	Customer ID Number Customer Name Customer Address Customer Phone Number	985678 Ankit Mittal B-176,MC Colony, Bhiwani 57845-69821
ACCOUNT	Account Type Account Number Account Balance Account Statement	Savings 1800221133665587 125000 5000_Credit
SUPPLIER	Supplier Name Supplier Address Supplier Phone Number Supplier Bill Number	Vishal Rastogi H-456, Shiv Nagar, Bhiwani 789546256 B-896541

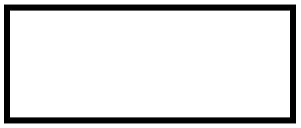
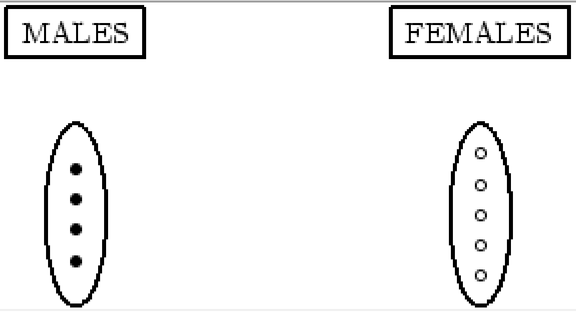
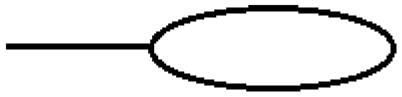
Difference Between Entity, Entity Type and Entity Sets -

Entity	**Entity Type**	**Entity Set**
Entity is a person, place, thing, event, or even a concept. It may be tangible or intangible.	An entity type defines the collection of entities that have the same attributes.	An entity set is a set of entities of the same type that share the same properties at any point in time (or) It is the collection of entities of an entity type at a point of time.
For Example, Entity is e1 or e2 or e3 or an apple where e1 e2 e3 are employees and an apple is a fruit.	For example, entity type is EMPLOYEE or Fruit	Entity set is a bucket of apple, banana, mango etc. or {e1,e2......}

Example :

Entity Type	Employee
Attributes	Name, Age, Pay
Entity Set (extension)	e1 (Rakesh, 40, 10000) e2 (Arun, 55, 8500) e3 (Sandeep, 30, 14000)

Notations Of above Definitions in ER Diagram -

	**Entity Type**
	**Entity Set**
	**Attribute**

Keys in DBMS

Keys in DBMS - CODD Rule:

Before discussing keys in DBMS, you must learn the CODD Rule. CODD Rule says that - No two tuples of the table should be the same, means one tuple should differentiate from other tuple using same attribute set.

(How we can get this?? -- By Key

What is a Key in DBMS?

The minimum no. of attributes used to differentiate all the tuples of the relation is called a key. For example in Table 1

Table 1					
SID	**CID**	**Sname**	**Marks**	**Class**	**City**
S1	C3	Ankit	98	12th	Bhiwani
S2	C4	Pooja	89	10th	Hansi
S3	C5	Komal	89	6th	Bhiwani
S4	C1	Ankit	78	6th	Ujjain

{SID} : key in Table 1 ⇒ Because It recognize the tuple uniquely.
{SID, CID} : can be the key but SID is only able to differentiate that is is a key. So, no need of CID.

Different Types of Database Keys in DBMS -

- **Simple Key**
- **Compound Key/Composite Key/Concatenated Key**
- **Primary Key**
- **Super Key**
- **Candidate Key**
- **Alternate Key/Secondary Key**
- **Foreign Key**

Let us consider **another example** to illustrate the concept of different **types of database keys in DBMS.** Let the **Table 2** be a **relation** of **EMPLOYEE**, which consists of Employee ID (**empID**), Employee Name(**Ename**), Department Number in which the Employee is working(**D.No**), Date of Birth (**DOB**), and the Employee Father's Name (**F.Name**).

Table 2					
empID	**Ename**	**D.No**	**Passport_No.**	**DOB**	**F.Name**
S1	Sunny	D1	P-45896	10/90	X
S2	Mohit	D3	P-78952	12/91	Y
S1	Sunny	D2	P-63589	11/92	Y
S3	Ajit	D1	P-98723	10/90	Z

Simple key in DBMS

If key consists of single attributes. For example,
{SID} : Simple Key ⇒ Because SID is only able to recognize the tuple uniquely.

Compound key in DBMS or Composite Key in DBMS or Concatenated Key in DBMS

If key consists of more than one attribute. In Table 2, No single attribute is able to recognize the tuple uniquely as if empID is choosen as key, then we found duplicate records for S1. So, a compound key will be choosen to identify a tuple/record uniquely of an employee.

{empID,D.NO}   : Compound key/Composite Key/Concatenated Key in Table 2
⇒ Because the key consists of two attributes empID and D.NO that recognize the tuple uniquely. Any attribute alone cannot recognize the tuple uniquely.

{empID, Ename} : cannot be the key, because anyone key alone cannot be used to differentiate.

{DOB, F.Name} : can be the key, but, any one key alone cannot be used to differentiate.

{Passport_No} : can be the key. Because each employee has its own Passport Number.

Database Primary key

One of the candidate key becomes primary key that should satisfy some properties. The properties are - .

- Primary key do not allow null values.
- Values must be distinct.

{SID}   : Primary Key in Table 1.
{empID,D.NO}   : Primary Key in Table 2.

Selection of Primary Key is a part of database optimization. Let us discuss some **constraints to select the Primary Key** -

1. Candidate key with no null values should be choosen as Primary Key.
2. Assume SID and Pno -> not null DBMS provides default index on the primary key.
3. Select *
4. from emp
   where pno=x;

   if we get results 10% only by using SID as key, and 90% from Pno, then make key which is used to access data more frequency is better choice.

5. Key with numerical values is better choice rather than choosing Key having character values.
6. Key with less number of attributes is a better choice. For example, A- NOT NULL 50% chances, integer (1 attribute, only A) BC- NOT NULL 50% chances, integer (2 attribute B & C) choose 'A' as primary key as it has only 1 attribute.
7. Key which is most frequently used to access the Database should be choosen.

Super Key in DBMS

Super key is a set of one or more than one keys that can be used to identify a record uniquely in a table. It is any combination of fields within a table that uniquely identifies each record within that table. **Primary key, Unique key, Alternate key are subset of Super Keys.**

Candidate Key in DBMS

A minimal super key is called a candidate key. There may be more than one keys in a relation such that they recognize a tuple uniquely. (i.e. each of the key have the property of the primary key). But the least combination of fields that uniquely identifies each record in the table is a Candidate Key. The least combination of fields distinguishes a candidate key from a super key. There can be multiple Candidate Keys in one table.

{SID}	: Candidate Key in Table 1
{CID}	: Candidate Key in Table 1
{empID,D.NO}	: Candidate key in Table 2
{Passport_No}	: Candidate Key in Table 2
{empID,F.Name}	: Candidate Key in Table 2

Alternate key in DBMS or Secondary key in DBMS

All candidate keys except primary key are secondary key. For example,

CID	: Alternate Key in Table 1
{Passport_No}	: Alternate Key in Table 2
{empID,F.Name}	: Alternate Key in Table 2

So, From the above understanding, the **SUPER-KEYS** will be -

{SID}	: Super Key in Table 1.
{CID}	: Super Key in Table 1.
{empID,D.NO}	: Super Key in Table 2.
{Passport_No}	: Super Key in Table 2.
{empID,F.Name}	: Super Key in Table 2.

Database Foreign Key

A foreign Key is an attribute or (combination of more than one attribute) of a relation (Table) that is the primary key of another relation (Table). In other words, If we had a table A with a primary key P that linked to a table B where P was a field in B, then X would be a foreign key in B. So, Foreign key is the set of attribute used to reference primary key and alternate key of the same table or some other table. For example,

Table 3	
D.NO	**D.Location**
D1	Rohtak
D2	Bhiwani
D3	Karnal

{D.NO} : Primary Key of Table 3
Consider the Table 3,
D.No is a foreign key in Table 2 since D.NO is a Primary Key in Table 3.

Some Important Points about Super Key

- Minimal super key is the candidate key.
- Every minimal key is super key.

Difference between primary key and alternate key or secondary key-

Primary Key	**Alternate Key or Secondary Key**
Null values not allowed	Null values Allows
Atmost 1 primary key possible	More than 1 alternate keys are possible

Questions on Super Keys and Candidate Keys using Closure

Identify Super Keys and Candidate keys :

Question 1 :
Let R(ABCDE) is a relational schema, where
$(AB)^+ = ABCDE$
$(A)^+ = ABCDE$
Is AB: Candidate Key or Not??
Solution :
AB : Not a Candidate Key, AB is only : Super Key
Question 2 :
Let R(ABCDE) is a relational Schema having FDs
$\{AB \rightarrow C, C \rightarrow D, B \rightarrow E\}$
Find out the Candidate Key ?
Solution :
(AB^+) : {ABCDE} $\Rightarrow$ super key
(A^+) : {A} $\times$
(B^+) : {EB} $\times$
$\therefore$ AB : minimal superkey $\Rightarrow$ Candidate Key. No subset of its attributes is a key.
Question 3 :
Let R(ABCDE) is a relational schema having FDs
$\{AB \rightarrow C , C \rightarrow D, B \rightarrow EA\}$

Find Out the Candidate Key ?

Solution :

(AB^+) : {ABCDE} $\Rightarrow$ Superkey

(A^+) : {A}

(B^+) : {EABCD} $\Rightarrow$ Superkey

$\Rightarrow$ B is Candidate Key.

Question 4 :

Let R(ABCDE) is a relational schema having FDs

$\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$

Find out the Candidate Key ?

Solution :

(AE^+) :{ABCDE} $\Rightarrow$ SuperKey

(A^+) : {ABCD}

(E^+) : {E}

AE : Candidate Key. No subset of its attributes is a key.

Question 5 :

Let R(ABCDEF) is a relational schema having FDs

$\{A \rightarrow BCDEF, BC \rightarrow ADEF, B \rightarrow C, D \rightarrow E\}$

Find out the Candidate Key ?

Solution:

$(A)^+$: ABCDEF $\Rightarrow$ (SuperKey)

$(BC)^+$: {BCADEF} $\Rightarrow$ (SuperKey)

$(B)^+$: {BCADEF}

$(C)^+$: {C}

{A, B} $\Leftarrow$ Candidate Key. No subset of its attributes is a key.

Question 6:

Given the following set F of functional dependencies for relation schema R = {A, B, C, D, E}.

{A -> BC, CD -> E, B -> D, E -> A}

List the candidate keys for R.

Solution :

$(A)^+$: ABCDE $\Rightarrow$ (SuperKey)

$(E)^+$: EABCD $\Rightarrow$ (SuperKey)

$(CD)^+$: CDEAB $\Rightarrow$ (SuperKey)

$(CB)^+$: CBDEA $\Rightarrow$ (SuperKey)

Any combination of attributes that includes those is a superkey.

From above , the minimal super keys are $\Rightarrow$ A, E, CD and BC.

Hence, the candidate keys are A, E, CD, BC.

Question 7:
Consider a relation R(A,B,C,D,E) with the following dependencies:
{AB-> C, CD -> E, DE -> B}
Is AB a candidate key of this relation? If not, is ABD? Explain your answer.
No. The closure of AB does not give you all of the attributes of the relation.
For ABD,
(ABD)+ = ABDCE ⇒ Super Key
(A) = {A}
(B) = {B}
(D) = {D}
⇒ ABD is a candidate key. No subset of its attributes is a key.

Question 8 :
Consider a relation with schema R(A,B,C,D) and FDs {AB -> C, C -> D, D -> A}. What are all candidate keys of R?
$(AB)^+$: ABCD ⇒ (SuperKey)
$(A)^+$: A ⇒ (Not able to determine all the attributes)
$(B)^+$: B ⇒ (not able to determine all the attributes)
$(DB)^+$: ABCD ⇒ (SuperKey)
$(CB)^+$: ABCD ⇒ (SuperKey)
$(D)^+$: DA ⇒ (Not able to determine all the attributes)
$(C)^+$: CDA ⇒ (Not able to determine all the attributes)

⇒ By calculating an attribute closure we can see the candidate keys are: AB, BC, and BD.