DATABASE MANAGEMENT SYSTEMS
(Open Elective – II)

III-B.Tech- II Sem | **L T P C**
Subject Code: 17CS3204OE | **3 0 0 3**

Course Outcomes: Upon completion of the course, the students will be able to

1. design databases using E-R model
2. construct database using relational model
3. formulate SQL queries to interact with database
4. make use of transaction control commands
5. apply normalization on database to eliminate redundancy

Unit-I

Introduction to Database Systems: Introduction and applications of DBMS, Purpose of data base, History of database, Database architecture - Abstraction Levels, Data Independence, Database Languages, Database users and DBA.
Introduction to Database Design: Database Design Process, Data Models, ER Diagrams - Entities, Attributes, Relationships, Constraints, keys, Generalization, Specialization, Aggregation, Conceptual design with the E-R model for large Enterprise.

Unit-II

Relational Model: Introduction to the relational model, Integrity constraints over relations, Enforcing integrity constraints, Querying relational data, Logical database design: E-R to relational, Introduction to views, Destroying/altering tables and views.

Unit-III

Part-A: SQL Basics: DDL, DML, DCL, structure – creation, alteration, defining constraints – Primary key, foreign key, unique, not null, check, in operator.

Part-B: Functions: Aggregate functions, Built-in functions - numeric, date, string functions, set operations.

Unit-IV

Sub-queries: Introduction, correlated sub-queries, use of group by, having, order by, join and its types, Exist, Any, All, view and its types.
Transaction control commands: ACID properties, concurrency control, Commit, Rollback, save point, cursors, stored procedures, Triggers.

Unit-V

Normalization: Introduction, Normal forms - 1NF, 2NF, 3NF, BCNF, 4NF and 5NF, concept of De-normalization and practical problems based on these forms.

Textbooks:

1. Raghurama Krishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, TMH.
2. Abraham Silberschatz, Henry F.Korth, S.Sudarshan, Database System Concepts, 6th Edn, TMH.

UNIT-1

Introduction to Database System & Design

Introduction to Database Systems:-

* **Data:-** Data is a raw and unorganized fact that required to be processed to make it meaningful. Data can be simple at the same time unorganized unless it is organized.

(or)

Any raw / fact material that can be recorded.

Ex:- Data is always interpreted, by a human or machine, to drive meaning. So, data is meaningless. Data contains numbers, statements and characters in a raw form.

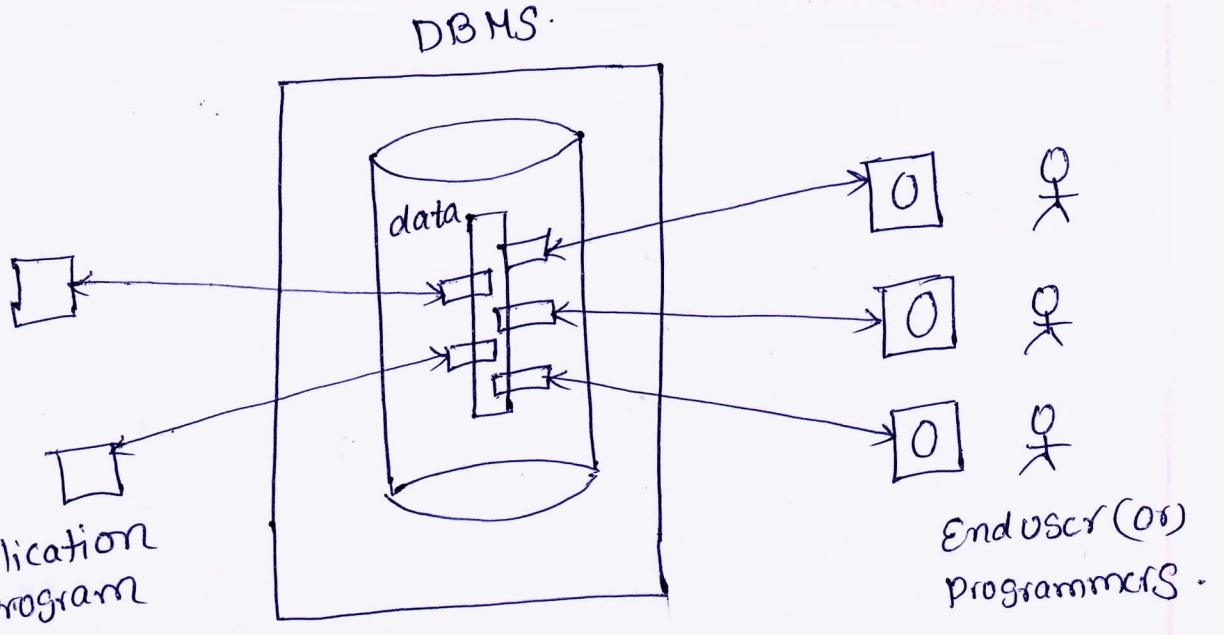
Ex:- Text, number, images, videos, speech.

* **Information:-** Information is a set of data which is processed in a meaningful way according to the given requirement. Information is processed, structured, or presented in a given context to make it meaningful and useful.

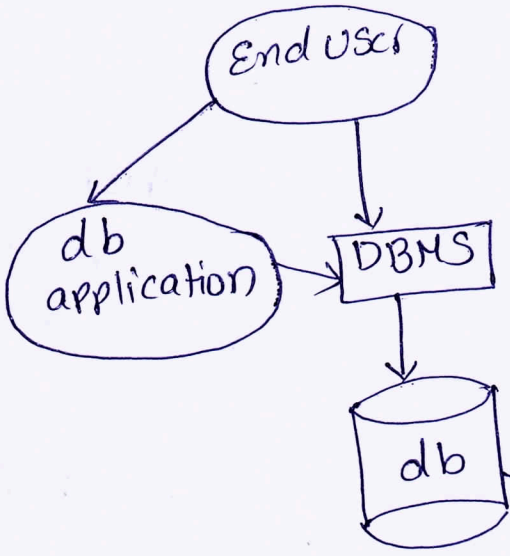
Ex:- The average score of a class or of the entire school is information that can be drived from the given data.

* **Data Base:-** Database is a collection of related data organised in a way that data can be easily accessed, managed and updated.

In diagramatic representation:-



Ex:- C, C++, Java.



∴ End users developing the database application with the help of DBMS software and ~~the~~ store the data in database.

There are different types of data bases:-

→ Traditional database :- Text, numbers.
→ Multimedia database :- Rockets, Satellites, images.
→ Real time database :- Marketing, Banking.
→ Data Ware house :- It is a one kind of database. Data is going to be very huge and historical.

* DBMS:- DBMS is a collection of interrelated / related data and a set of programs to access and store those data in an easy and efficient manner.

(or)

DBMS is a software that allows creation, difinition and manipulation of database, allowing users to store, process and analyse data easily.

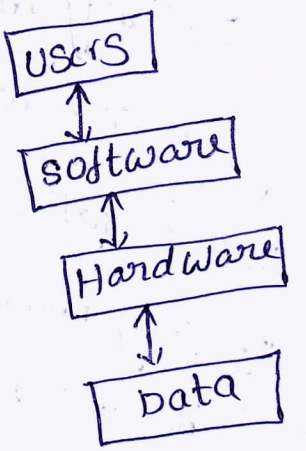
• DBMS also provides protection and security to the databases. It also maintains data consistency in case of multiple users.

Examples of popular DBMS used these days:

→ MySQL
→ Oracle
→ SQL Server
→ IBM DB2
→ postgre SQL
→ Amazon Simple DB (cloud based).

* Components of DBMS :-

The components of DBMS are (1) Users (2) Software (3) Hardware (4) Data.



(1) Users:- Users is also called as Application programmers, End users, database administrator.

(2) Software:- It controls the organization, storage, management and retrieval of data in database.

(3) Hardware:- Hardware of a system can range from PC to a network of components. It also include various storage devices like hard disk, I/O devices like monitor, printer.

(4) Data:- Data stored in database includy, numerical data, non numerical data (or) logical data.

Building Block of DB:-

We can store the data in database used in three building blocks

→ columns/ fieldy
→ rows/ tuples/ record.
→ Tables.

Purpose of Database:-

Previously data is stored in files.

→ Drawbacks of using file systems to store data:

1) Data redundancy & inconsistency:- multiple file formats, duplication of information in different files.

2) Difficulty in access data:- Need to write a new program to carry out each new task.

3) Data isolation:- multiple files and formats.

4) Integrity problems:- Integrity constraints (E.g:- account balance >0) become part of program code.
→ Hard to add new constraints or change existing ones.

5) Security problems.

5) Atomicity of updates:- Failure may leave database in an inconsistent state with partial updates carried out.

E.g:- Transfer of funds from one account to another should either complete or not happen at all.

6) Concurrent access by multiple users:-

→ concurrent accessed needed for performance

→ uncontrolled concurrent accesses can lead to inconsistencies

Eg:- Two people reading a balance and updating it at the same time.

→ Database systems offer solutions to all the above problems.

* Differences b/w file system and Database system:

	File system	Database
1. Definition	A process that manages how and where data on a storage disk is stored, accessed and managed	An organized collection of data can be easily accessed, managed and updated.
2. Data consistency	High data inconsistency	Maintains data consistency
3. Structure	Structure is simple	Structure is complex
4. Data sharing	Data sharing is hard	Data sharing is easy
5. Redundancy	High Redundancy	Low redundancy
6. Security	less secure	More secure
7. Backup and Recovery	No backup and Recovery process	There is a backup and recovery

* Advantages of database Management system: -

1) Data Independence

2) Efficient data access: Database can access the data efficiently and faster and more secure

3) data integrity & security

4) Data administration: Everything will be control of the administrator. only the person who are having the username and password, he can only modify the data which is presented in the database.

5) Concurrent access & Crash Recovery: - something will happen in the database we can recovery on the data in database. DB can crash recovery is possible.

6) Reduce application develop time and maintainance need.

* Disadvantages of database Management System: -

→ The overhead cost of using DBMS.

- High initial investment in hardware, software and training.
- cost of defining & processing data.
- overhead for security, concurrency control, recovery
- complexity
- Except MySQL, which is open source, licensed DBMS's are generally costly
- They are large in size.

Characteristics of Database Management System:-

1) Data stored into Tables:- Data is never directly stored into the database. Data is stored into tables, created inside the database. DBMS also allows to have relationships between tables which makes the data more meaningful and connected. You can easily understand what type of data is stored where by looking at all the tables created in a database.

2) Reduced Redundancy:- In the modern world hard drives are very cheap, but earlier when hard drives were too expensive, unnecessary repetition of data in database was a big problem. But DBMS follows Normalisation which divides the data in such a way that repetition is minimum.

3) Data consistency:- Data that is being continuously updated and added, maintaining the consistency of data can become a challenge. But DBMS handles it all by itself.

4) Support Multiple User and Concurrent Access:- DBMS allows multiple users to work on it (update, insert, delete data) at the same time and still manages to maintain the data consistency.

5) Query Language:- DBMS provides users with a simple query language, using which data can be easily fetched, inserted, deleted and updated in a database.

6) Security:- The DBMS also takes care of the security of data, protecting the data from unauthorized access. DBMS, we can create user accounts with different access permissions, using which can easily secure our data by restricting user access.

7) DBMS supports transactions, which allows us to better handle and manage data integrity in real world applications where multi-threading is extensively used.

* Applications Of DBMS:-

1. Banking:- customer information, Accounts loans and banking transactions.

2. Airlines:- Reservation, cancellation and schedule information and also staff and passenger information.

3. Universities:- Student & course information and registration and results with grades.

4. Credit card transactions:- Purchase on credit cards and generation of monthly bills.

5. Telecommunications:- Information of communication network, keeping records of calls made, generating balance on prepaid calling cards.

6. Finance:- For storing information about ~~holidays~~ holdings, sales and purchase of financial instruments such as stocks and bonds

7. Sales:- For customer, product and purchase information

8. Manufacturing:- For management of supply chain and for tracking production of items in factories, inventories of items in ware houses / stores and order for items.

9. Human Resource:- For information about Employees, salaries, payroll taxes and benefits.

* History of Database:-

1950's and early 1960's:-

→ Data Processing using magnetic tapes for storage
→ Tapes provided only sequential access
→ punched cards for input

Late 1960s and 1970s:-

→ Hard disks allowed direct access to data.
→ Hierarchical and network data models in widespread use
  • IBM's DL/I (Data Language one)
  • CODASYL's DBTG (Data Base Task Group) model the basis of current DBMSs.
→ Ted codd defines the relational data model
  • IBM Research develops System R prototype
  • UC Berkely develops Ingres prototype
→ Entity - Relationship Model for database design

1980's:-

- Research relational prototypes into commercial systems
- DB2 from IBM is the first DBMS product based on the relational model.
- Oracle and microsoft SQL server are the most prominent commercial DBMS Products based on the relational model.
- SQL becomes industrial standard
- Parallel and distributed database systems
- Object-oriented database systems (OODBMS)

Late 1990's:-

- Large decision support and data-mining applications
- Large multi-terabyte data warehouses
- Emergence of web-commerce.

Early 2000's:-

→ XML and XQuery standards
→ Automated database administration

Later 2000's:-

- Web databases (semi-structured data, XML, complex data types)
- cloud computing
- Giant data storage systems (Google Big Table, Yahoo PNuts, Amazon web services, ...:)
- Advanced databases (mainly non-relational (e.g; graph-based, text-based) but advanced relational model)

* View of Data:-

The database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.

The major purpose of a database system is to provide users with an abstract view of the data. i.e; the system hides certain details of how the data are stored and maintained.

Data Abstraction:-

The system to be usable, it must retrieve data efficiently The need for efficiency has led designers to use complex data structures to represent data in the database.

There are 3 different types of data abstraction levels.

1) view level (External level) 2) physical level (internal / storage level) 3) logical level. (conceptual level)

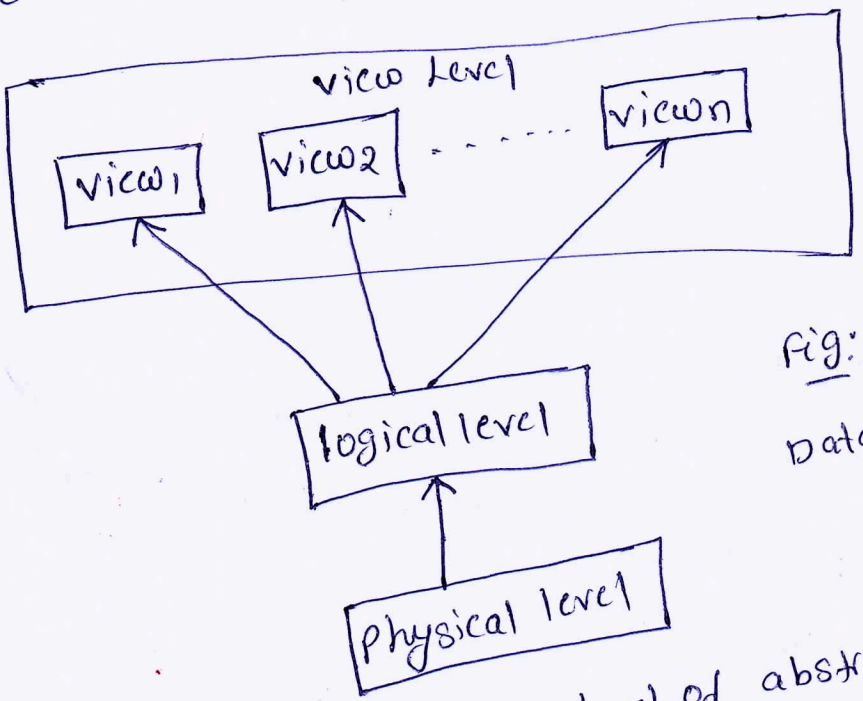


fig:- Three levels of data abstraction

1) view level:- The highest level of abstraction describes only part of the entire database

→ The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

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

3) logical level:- The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data.

logical level thus describes the entire database in terms of a small no. of relatively simple structures.

Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.

Ex:- We are storing customer information in a customer table.

physical level:- records can be described as block of storage in memory (bytes, gigabytes)

logical level:- These records can be described as fields and attributes along with their data types, relationship among each other can be logicaly implemented.

view level:- User just interact with system with the help of GUI and Enter the details of the screen, they are not aware of how the data is stored and what data is stored.

* Data Independence:-

Instances:- The collection of information stored in the database at a particular moment is called an instance of the database.

Schema:- The overall design of the database is called the database schema. schemas are changed infrequently.

There are two different types of schemas.

physical schema:- It describes the database design at the physical level.

logical schema:- It describes the database design at the logical level.

Data Independence:- It is the capacity of changing the schema at one level without affecting the other level.

→ DB system are designed in multilayer

→ DB system stores data about data, known as meta data.

→ Meta data follows a layered architecture.

1) logical data Independence:-

→ It stores information about how data is managed inside.

→ Ability to change logical schema without changing the external or application program.

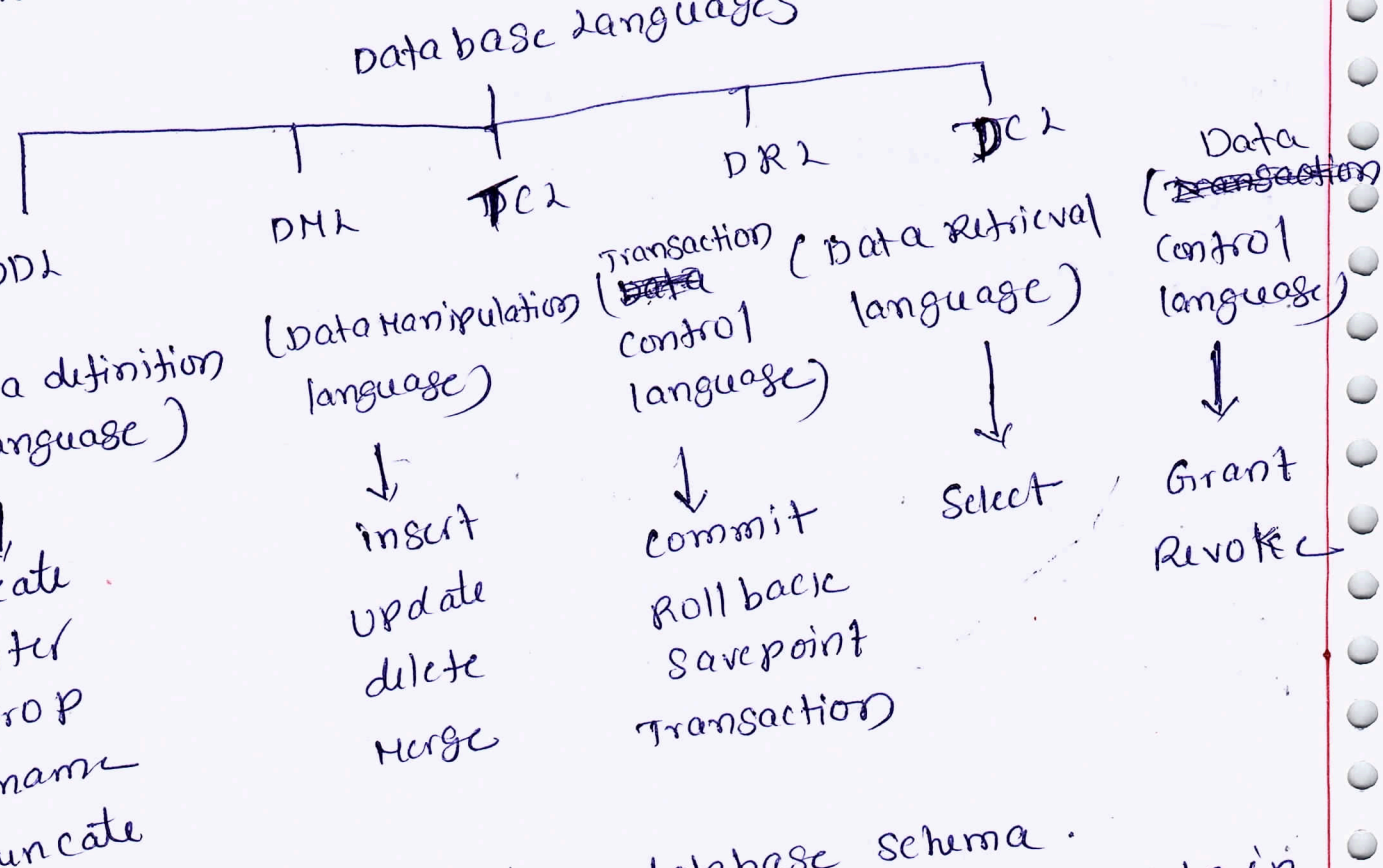
2) physical data Independence:- Ability to change the physical data without impacting the logical data / external

Ex:- Change in internal schema different file organization, storage devices, structure should be possible without affecting the conceptual / External schema.

* Data base Languages:-

Database language is a language used to communicate with the database. These languages are used to store, read and update database. There are many database languages but commonly used language is SQL.

SQL is a structured Query language used to communicate with the database.



DDL:- Used for specifying database schema.

DML:- DML is used for accessing and manipulating data in a database

DML is of two types:

→ Procedural DML:- Procedural deals with "what and how" data are needed.

Ex:- relational algebra.

→ Declarative DML(Or) Non Procedural:- It deals with "what" data are needed without specifying how to get those data.

Ex:- relational calculus (tuple and domain)

* Database Architecture:-

The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs. Database systems can be centralized or client-server, where one server machine executes work on behalf of multiple client machines. Database systems can also be designed to exploit parallel computer architecture.

Most users of a database system today are not present at the site of the database system, but connect to it through a network. We can therefore differentiate between client machines on which remote database users work, and server machines on which the database system runs.

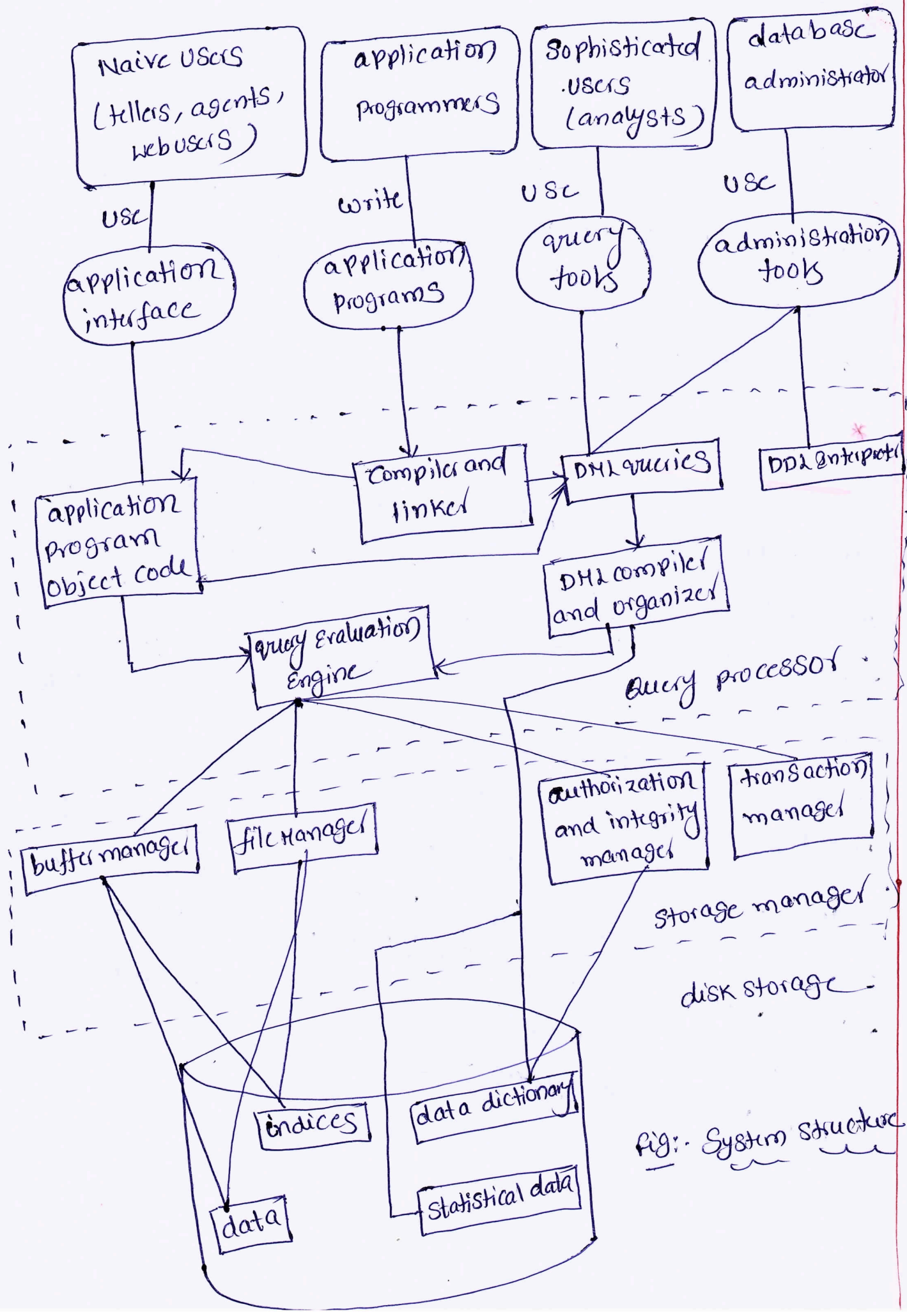
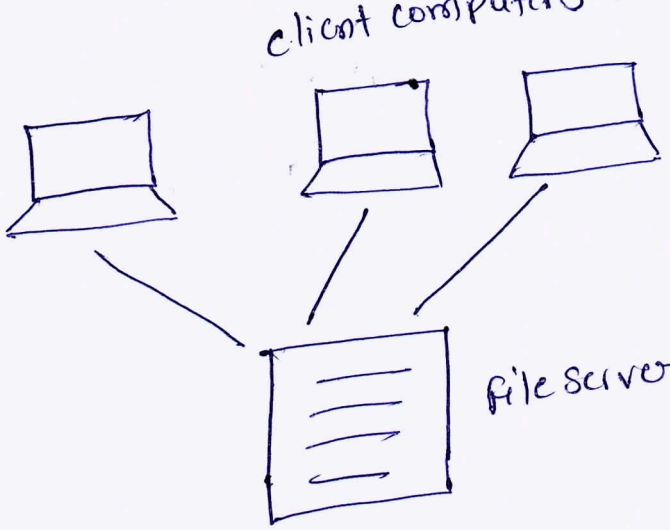


fig:- System structure

→ DBMS can be seen as either single tier or multitier.

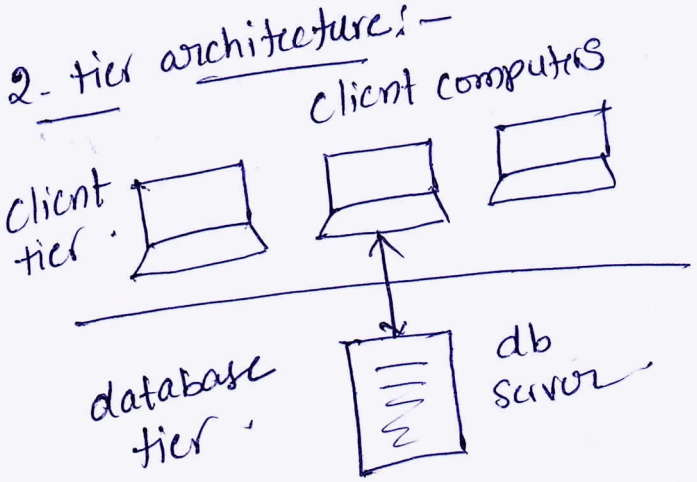
→ An n tier architecture divides the whole system into related but independent n modules.

1- tier architecture:-

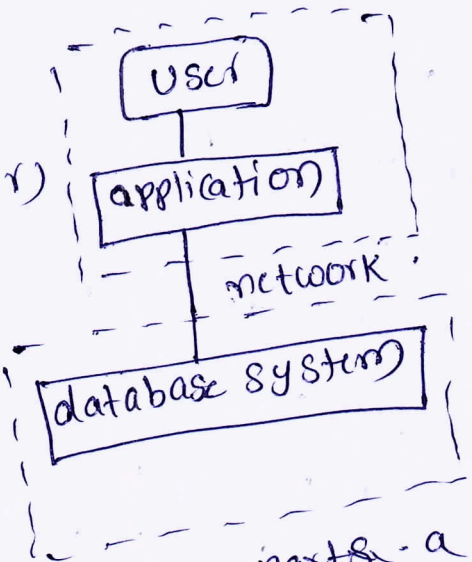


→ User interface
→ presentation service
→ Application service

2- tier architecture:-



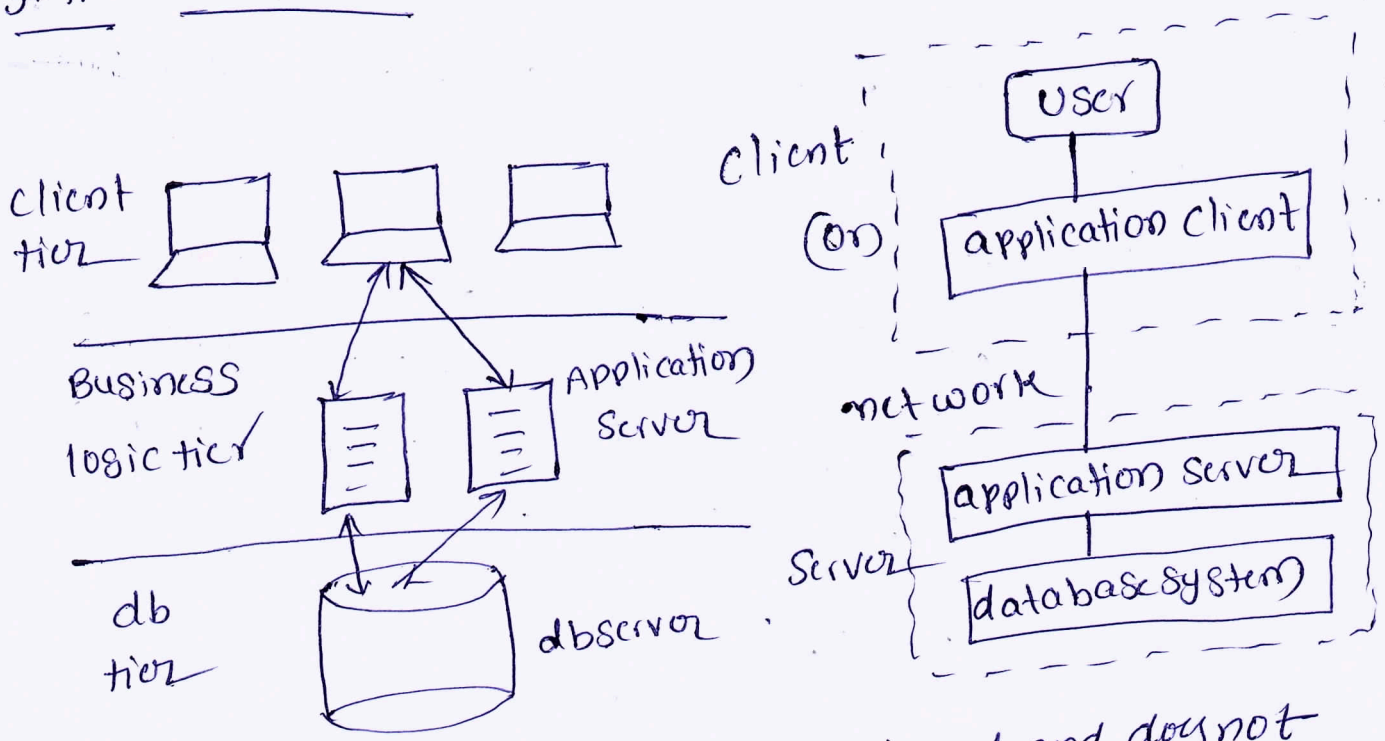
(or)



→ The application is partitioned into two or ~~three~~ parts - a component that resides at the client machine, which invokes database system functionality at the server machine through query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server.

→ It is a client server architecture
→ Direct communication
→ Run faster.

3-tier architecture:-



→ The client machine acts as merely a front end and does not contain any direct database calls. The client end communicates with an application server, usually through a forms interface. The application server in turn communicates with a database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide Web.

→ It separate tiers from each other based on complexity of the users & how the users & how they we the data present db.

→ It is a web based application.

→ Now a days we are using cloud computing.

There are 3 layers — client layer
business layer
data layer.

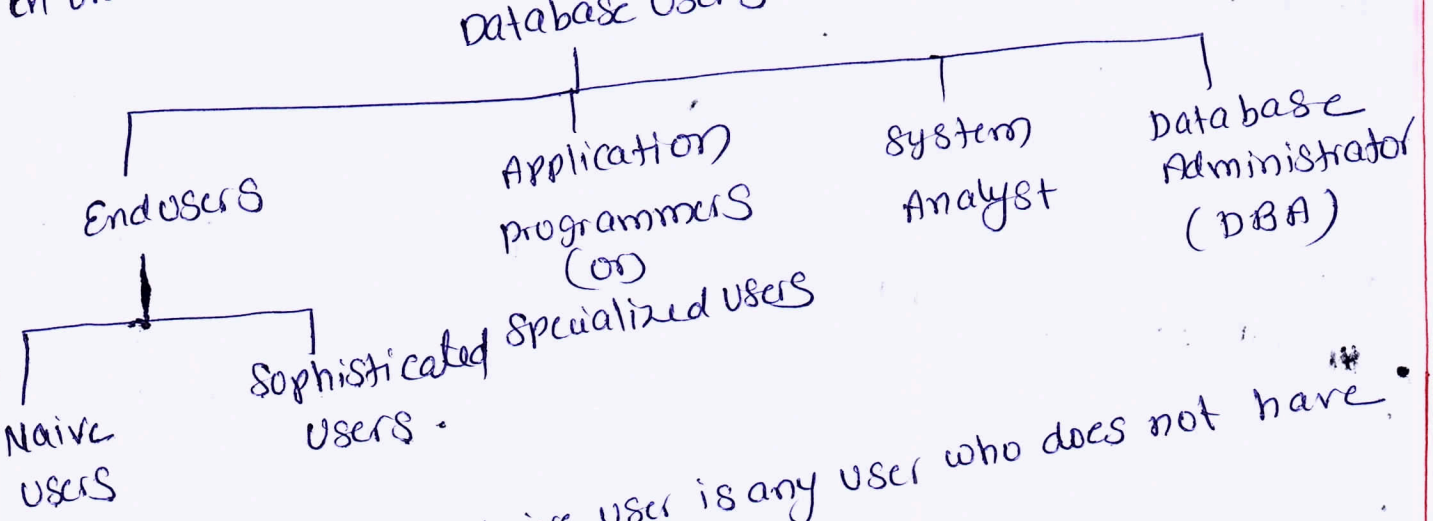
→ Database (Data) tier:- DB resides along with its query processing language

→ Application (Middle) tier:-
This presents an abstract view of db.
→ It acts as mediator between user and database
→ User (presentation) tier:- End user operate on this tier & they know nothing about any existence of db beyond this layer.

* Database users and Administrators:-

* Database users:-

There are no. of database users who interact with the database in order to access & update the database.



(i) Naive users:- Naive user is any user who does not have any knowledge about database.
Their task is to use the developed application (bank management system, library management system)

Ex:- owner of the bookstore who enters the details of various books in the database by using an appropriate application program.
- clerical staff in any library.

(ii) Sophisticated users:- sophisticated users have great knowledge of query language. so they use database query language to access information from the database to meet their complicated requirements.

Ex:- users such as a business analyst, scientist interact with the database without writing any application programs.

(2) Application programmers / specialized users:- specialized users writes application programs that uses the database.
- Application programs can be written in any high level programming language like C#, Java, .Net
- Application programs are made according to user requirements so that they can create, delete, access & update information in db.
- specialized users interact with the DBMS through data manipulation language

Ex:- specialized users write applications such as computer aided design systems, knowledge base & expert systems that store data having complex data types.
Application programmers develop user interface for data processing applications such as banking, payroll management.

(3) System Analyst:-

System Analyst is responsible for the design, structure & property of database.

- System analysts determine the requirement of the end users to create a solution for their business.

* Database Administrator:-

DBMS is to have central control of both the data and the programs that access those data. A person who has such central control over the system is called a database administrator (DBA).

→ Schema definition:- The DBA creates the original database schema by executing a set of data definition statements in the DDL.

→ Storage structure and access-method definition.

→ 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 someone attempts to access the data in the system.

→ Routine maintenance:- Backup database, tapes, onto remote servers - free disk space is available for normal operations.
Upgrading disk space.

→ Monitoring job running on the database and ensuring that performance.

DBA has many responsibilities:

→ Installing and upgrading the DBMS servers
→ Design and implementation
→ performance tuning
→ Migrate database servers
→ Backup and Recovery
→ Security
→ Documentation.

* Introduction to Database Design:-

* Database Design process:-

The database design process can be divided into six steps, The ER model is most relevant to the first three steps.

1. Requirement Analysis:- The very first step in designing a database application is to understand what data is to be stored in the database, what applications must be built on top of it, and what operations are most frequent and subject to performance requirements. In other words, we must find out what the users want from the database. This is usually an informal process that involves discussions with user groups, a study of the current operating environment and how it is expected to change, analysis of

any available documentation on existing applications that are expected to be replaced or complemented by the database.

2. Conceptual database Design:- The information gathered in the requirement analysis step is used to develop a high-level description of the data to be stored in the database, along with the constraints known to hold over this data. The ER model is one of several high-level, or semantic, data models used in database design. The goal is to create a simple description of the data that closely matches how users and developers think of the data.

3. logical Database Design:- We must choose a DBMS to implement our database design, and convert the conceptual database design into a database schema in the data model of the choosen DBMS. The logical design step is to convert an ER schema into a relational database schema.

4. Schema Refinement:- The fourth step in database design is to analyze the collection of relations in our relational database schema to identify potential problems, and to refine it.

5. physical Database Design:- We consider typical expected workloads that our database must support and further refine the database design to ensure that it meets desired performance criteria. This step may simply involve building indexes on some tables and clustering some tables or it may involve a substantial redesign of parts of the database schema obtained from the earlier design steps.

6. Physical Database Design:-

6. Application and Security Design:- Any Software Project that involves a DBMS must consider aspects of the application that go beyond the database itself. Design methodologies like UML try to address the complete software design and development cycle. We must identify the entities and processes involved in the application. We must describe the role of each entity in every process that is reflected in some application task, as part of a complete workflow for that task.

* Data Models:-

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. While the Relational Model is the most widely used database model.

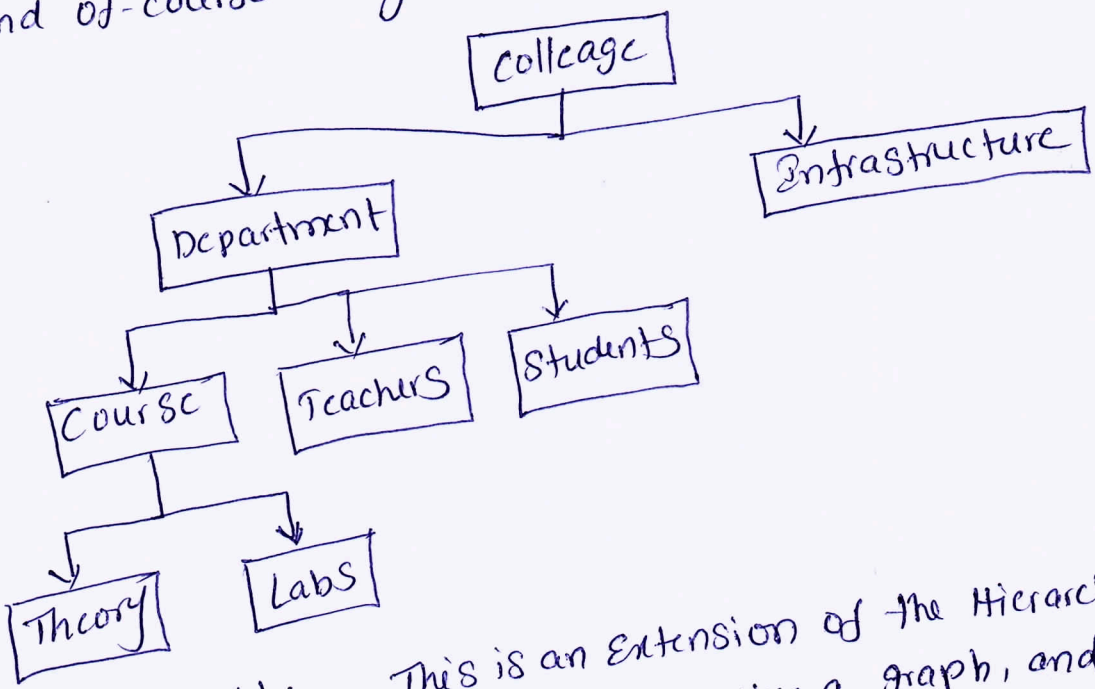
1) Hierarchical Model
2) Network Model
3) Entity - relationship Model
4) Relational Model.

1) Hierarchical Model:- This database modul organises 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. In this model, a child node will only have a single parent node.

This model efficiently describes many real-world relationships like index of a book, recipes.

In hierarchical model, data is organised into tree-like structure with one-to-many relationship between two different types of data.

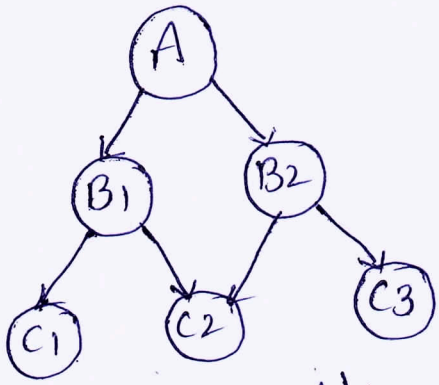
Ex:- one department can have many courses, many professors and of-course many students.



Network Model:- This is an Extension of the Hierarchical model. In this model data is organised, 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. 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.

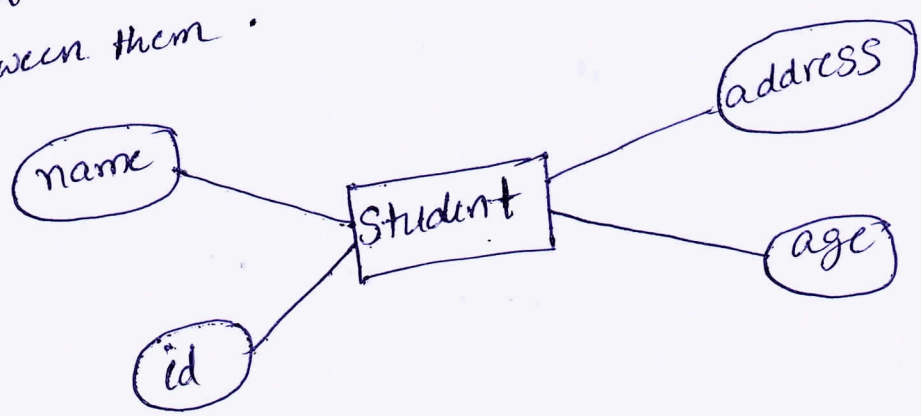


3) Entity - relationship Model:-

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. This model is good to design a database, which can be turned into tables in relational model.

Ex:- If we have to design a school database, then student will be an entity with attributes name, age, address. As address is generally complex, it can be another Entity with attributes street name, pincode, city and there will be a relationship between them.



4) 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

Student_id	name	age
1	A	17
2	B	18
3	C	17
4	D	18

Subject_id	name	teacher
1	Java	Mr.J
2	C++	Miss C
3	C#	Mr.C Hash
4	Php	Mr.PHP

Student_id	Subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

* ER Diagrams :- (Entity Relationship Diagram)

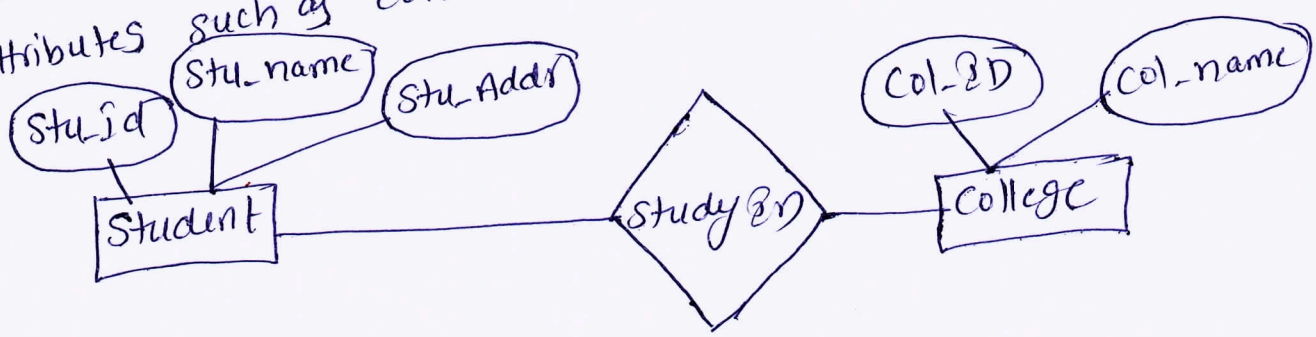
An Entity-relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can be implemented as a database. The main components of E-R model are: Entity set and relationship set.

What is an Entity Relationship Diagram (ER Diagram)?

An ER Diagram shows the relationship among entity sets. An Entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, ER Diagram shows the complete logical structure of a database

Simple ER Diagram:-

We have two entities student and colleage and their relationship. The relationship between student and college is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as stu_id, stu_name & stu_Addr and college entity has attributes such as col_ID & col_Name.



Rectangle: Represents Entity sets →

Ellipses: Attributes →

Diamonds: Relationship set →

Lines: They link attributes to Entity sets and Entity sets to Relationship set →

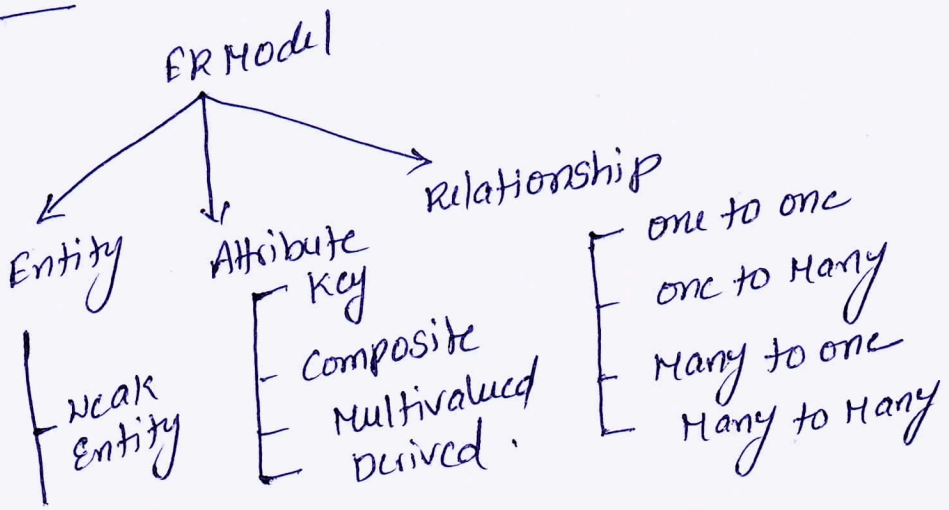
Double Ellipses: Multivalued Attributes →

Dashed Ellipses: Derived Attributes →

Double Rectangles: Weak Entity sets →

Double Lines: Total participation of an Entity in a relationship set →

Components of a ER Diagram



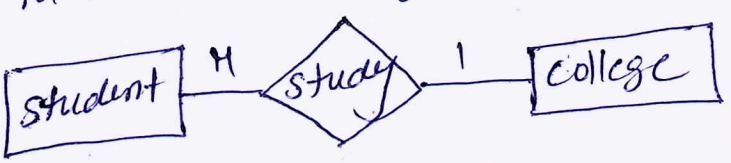
ER diagram has three main components:

1. Entity 2. Attribute 3. Relationship

1. Entity:- An Entity is an object or component of data.

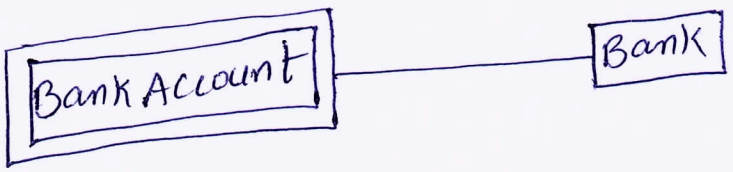
An Entity is represented as rectangle in an ER diagram

Ex:- We have two Entities Student and college and these two Entities have many to one relationship as many students study in a single college.



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.

Ex:- A bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.

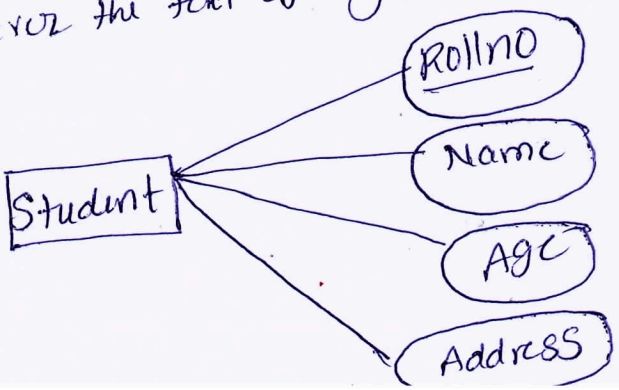


Attributes:-

An attribute describes the property of an entity. An attribute is represented as oval in an ER diagram. There are four types of attributes:

1) Key attribute 2) Composite attribute 3. Multivalued attribute
4) Derived attribute.

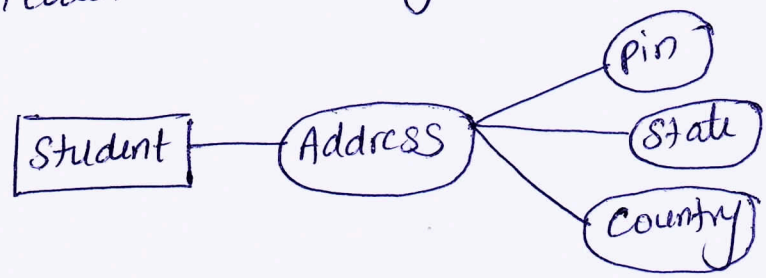
1) Key attribute:- A key attribute can uniquely identify an entity from an entity set. For example, student roll number can uniquely identify a student from a set of students. Key attribute is represented by oval same as other attributes however the text of key attribute is underlined.



2) Composite attribute:- An attribute that is a combination of other attributes is known as composite attribute.

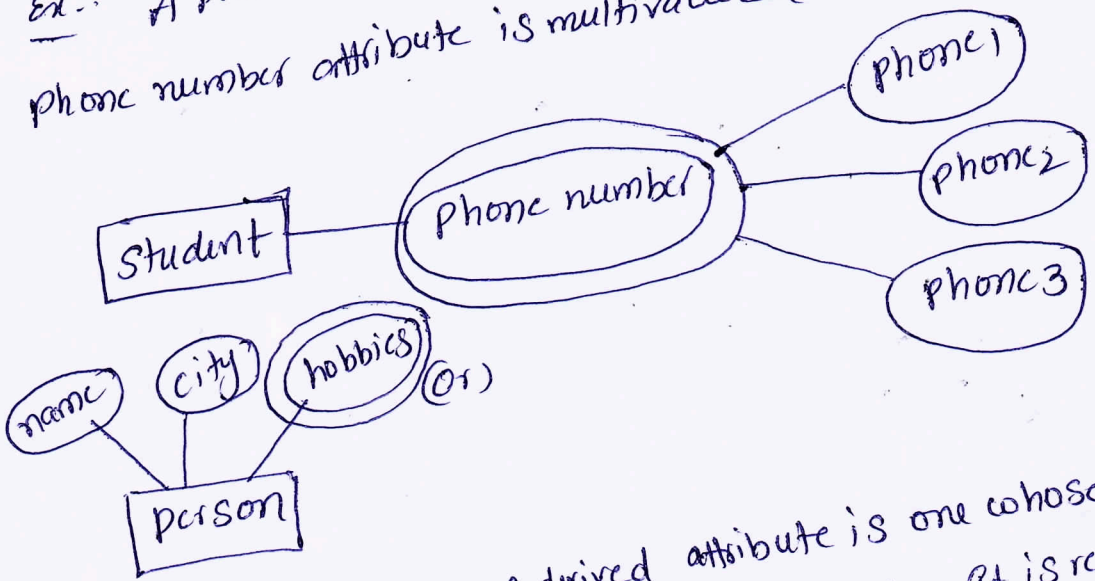
Ex:- In student entity, the student address is a composite attribute as an address is composed of other attributes such as pincode, state, country.

∴ Here address is a composite attribute



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.

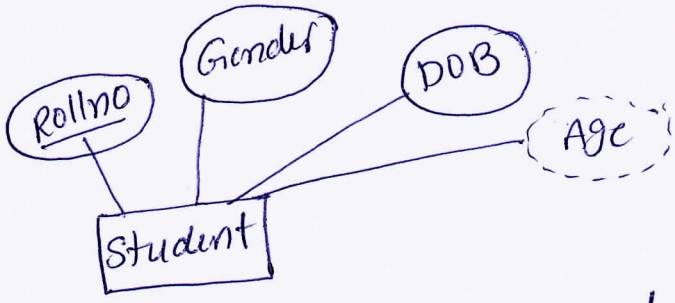
Ex:- A person can have more than one phone numbers so the phone number attribute is multivalued.



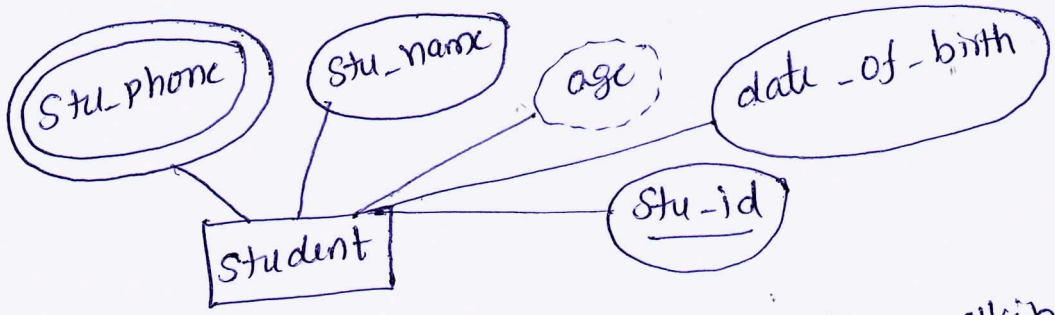
4. Derived attribute:- A drived attribute is one whose value is dynamic and drived from another attribute. It is represented by dashed oval in an ER diagram.

Ex:- person age is a derived attribute as it changes over time and can be drived from another attribute (Date of birth)

Ex:-

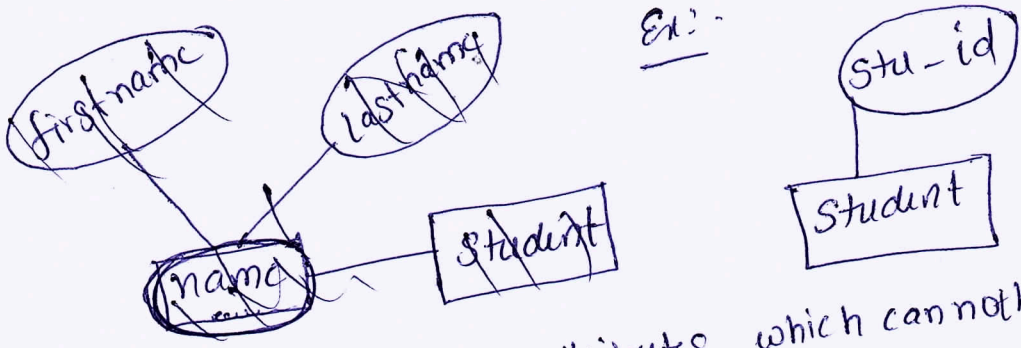


E-R diagram with multivalued and derived attribute



5) Single valued attributes:- Single-valued attribute is an attribute that can have only a single values:

Ex:-



6) simple / atomic attributes:- The attributes which cannot be further divided are called as simple / atomic attributes

Ex:- The Entity like age, marital status can not be sub divided and are simple attributes.

7) Stored attributes:- Attribute that cannot be drived from other attributes are called as stored attributes.

* Entity Set :-

An entity set is a collection of similar types of entities. An Entity set may contain entities with attribute sharing similar values.

Ex:- A student set may contain all the students of a school; like wise a Teachers set may contain all the teachers of a school from all facilities. Entity sets need not be disjoint.

* Relationships :-

Def:- A relationship is an association among two or more Entities.

Ex:- An Employee works_at a department, a student enrolls in a course. Here, works_at and Enrolls are called relationships.

Relationship set :- A set of relationships of similar type is called a relationship set. A relationship set can be thought of as a set of n-tuples:

$\{(e_1, \ldots, e_n) \mid e_1 \in E_1, \ldots, e_n \in E_n\}$

Each n-tuple denotes a relationship involving n entities e_1 through e_n, where entity e_i is in entity set E_i

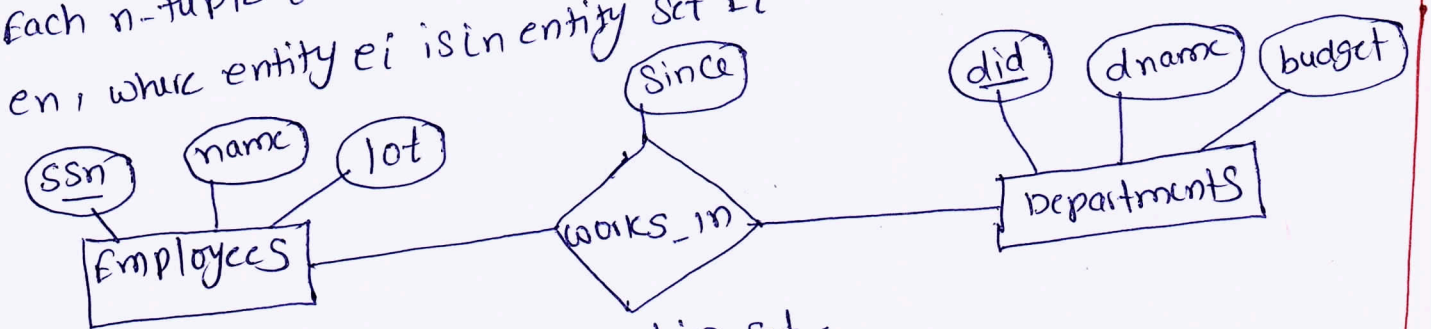


Fig:- The works_in Relationship set.

A relationship too can have attributes. These attributes are called descriptive attributes.

Degree of Relationship:-

The no. of participating entities in a relationship defines the degree of the relationship.

→ Binary = degree 2

→ Ternary = degree 3

→ n-ary = degree.

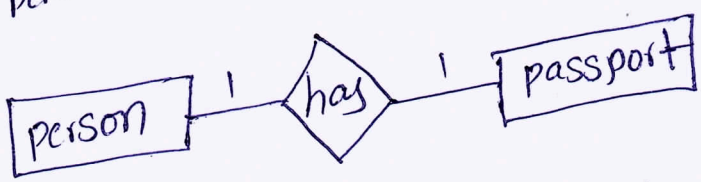
Cardinality:- Defines the numerical attributes of the relationship between two entities or entity sets.

Different types of cardinal relationships are:

1) one to one 2) one to many 3) many to one 4) Many to Many

1) one to one Relationship:- When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship. (i.e; 1:1)

Ex:. A person has only one passport and a passport is given to one person.

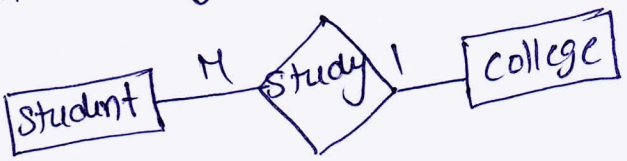


2) One to Many Relationship:- When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship. i.e; 1:M.

Ex:- A customer can place many orders but a order can not be placed by many customers.

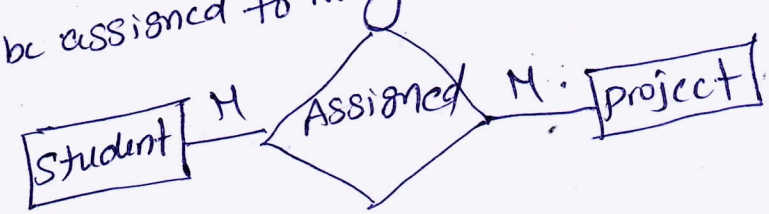
3) Many to one Relationship:- When more than one instances of an Entity is associated with a single instance of another entity then it is called many to one relationship. i.e; M:1

Ex:- Many students can study in a single college but a student can not study in many colleges at the same time.



4) Many to Many Relationship:- When more than one instances of an entity is associated with more than one instances of another Entity then it is called many to many relationship. i.e; M:M.

Ex:- A Student can be assigned to many projects and a project can be assigned to many students.



* Constraints in DBMS:- (Relational model)

Constraints enforce limits to the data or type of data that can be inserted / updated / deleted from a table. The whole purpose of constraints is to maintain the data integrity during an update / delete / insert into a table.

→ Relational constraints are the restrictions imposed on the database contents and operations.

→ They ensure the correctness of data in the database.

Types of constraints:- 5 different types of relational constraints.

1) Domain constraint
2) Tuple uniqueness constraint
3) key constraint
4) Entity Integrity constraint
5) Referential Integrity constraint

1) Domain constraint:-

→ Domain constraint defines the domain or set of values for an attribute.

→ It specifies that the value taken by the attribute must be the atomic value from its domain.

Ex:-

STU_ID	Name	Age
S001	Akshay	20
S002	Abhishek	21
S003	shashank	20
S004	rahul	A

∴ Here value 'A' is not allowed since only integer values can be taken by the age attribute.

2) Tuple uniqueness constraint:- Tuple uniqueness constraint specifies that all the tuples must be necessarily unique in any relation.

Ex:-

1)

Stu-ID	Name	Age
S001	Akshay	20
S002	Abhishek	21
S003	Shashank	20
S004	Rahul	20

∴ This relation satisfies the tuple uniqueness constraint since here all the tuples are unique.

2)

Stu-ID	Name	Age
S001	Akshay	20
S001	Akshay	20
S003	Shashank	20
S004	Rahul	20

∴ This relation doy not satisfy the tuple uniqueness constraint Since here all the tuples are not unique.

3) Key constraint:- Key constraint specifies that in any relation

→ All the values of primary key must be unique

→ The value of primary key must not be null

Stu-ID	Name	Age
S001	Akshay	20
S001	Abhishek	21
S003	Shashank	20
S004	Rahul	20

∴ This relation doy not satisfy the key constraint as here all the values of primary key are not unique.

4) Entity integrity constraint:- Entity integrity constraint specifies that no attribute of primary key must contain a null value in any relation.

→ This is because the presence of null value in the primary key violates the uniqueness property.

Stu-ID	Name	age
5001	Akshay	20
8002	Abhishek	21
8003	Shashank	20
	Rahul	20

∴ This relation does not satisfy the Entity integrity constraint as here the primary key contains a null value.

5. Referential Integrity Constraint:- This constraint is enforced when a foreign key references the primary key of a relation. It specifies that all the values taken by the foreign key must either be available in the relation of the primary key or be null

6. NOT NULL
7. UNIQUE
8. DEFAULT
9. CHECK.
10. MAPPING constraints.

* Keys:-

A DBMS key is an attribute or set of an attribute which helps you to identify a row (tuple) in a relation (table). They allow you to find the relation between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table.

(or)

Key plays an important role in relational database. It is used for identifying unique rows from table. It also establishes relationship among tables.

Ex:-

(1)

Serial NO	Item Name	Quantity	Price
1	cashew nuts	1 kg	800
2	Almonds	1 kg	900
3	pine nuts	1 kg	1000

Field.

Record.

In the above data item, each column is a field and each row is a record.

(2)

Employee ID	firstname	lastname
11	Andrew	Johnson
22	Tom	Wood
33	Alex	Hale

In the above example, Employee ID is a primary key because it uniquely identifies an employee record. In this table, no other employee can have the same Employee ID.

Why We need a key:-

→ Keys help you to identify any row of data in a table. In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated. Key ensure that you can uniquely identify a table record despite these challenges.

→ Allows you to Establish a relationship between and identify the relation between tables.

→ To Enforce identity and Integrity in the relationship

Various keys in Database Management System

1) Super key 2) primary key 3) candidate key 4) Alternate key
5) foreign key 6) compound key 7) composite key.

1) **Super key:-** A super key is a group of single or multiple keys which identifies rows in a table. A super key may have additional attributes that are not needed for uniqueness identification.

Example:-

EmpSSN	EmpNum	Empname
9812345098	AB05	Shown
9876512345	AB06	Roslyn
1999378990	AB07	James

In the above-given Example, EmpSSN and EmpNum name are super keys.

2) **Primary key:-** A column or group of columns in a table which helps us to uniquely identifies every row in that table is called a primary key. This DBMS can't be a duplicate. The same value can't appear more than once in the table.

Rules:- 1) Two rows can't have the same primary key value.
2) It must for every row to have a primary key value.
3) The primary key field cannot be null
4) The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

Ex:-

Stu_ID	First Name	lastName
11	Tom	price
12	NICK	wright
13	Dana	Natan

∴ Stu_ID is a primary key.

3) Alternate key:- All the keys which are not primary key are called an alternate key. It is a candidate key which is currently not the primary key. However, A table may have single or multiple choices for the primary key.

Ex:- StudID, RollNO, Email are qualified to become a primary key. But since StudID is the primary key, RollNO, Email becomes the alternative key.

StudID	RollNO	firstName	lastName	Email
1	11	Tom	price	abc@gmail.com
2	12	NICK	wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

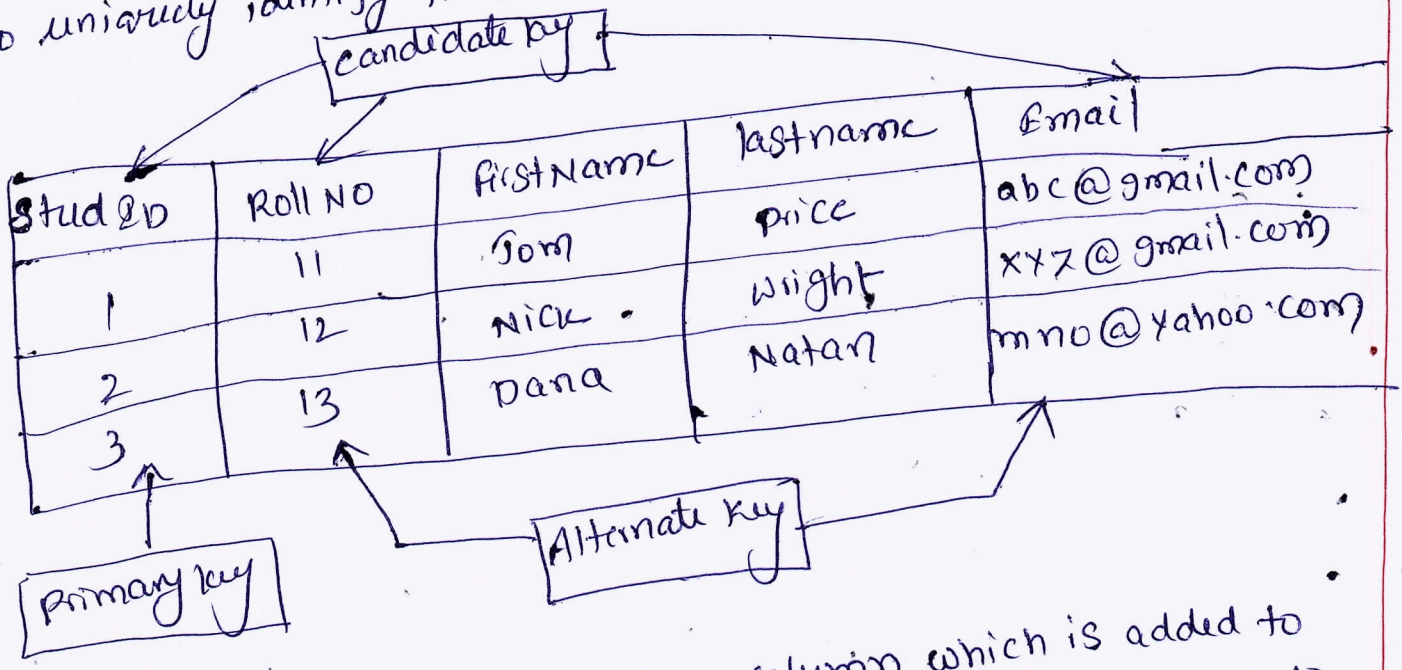
4) Candidate key:- A super key with no repeated attribute is called candidate key. The minimal set of attribute which can uniquely identify a tuple is known as candidate key.
The primary key should be selected from the candidate keys.

Every table must have atleast a single candidate key.

Rules:- 1) It must contain unique values.
2) candidate key may have multiple attributes
3) Must not contain null values.
4) It should contain minimum fields to ensure uniqueness

5) uniquely identify each record in a table:

Ex:- Stu ID, Roll NO, and Email are candidate keys which helps us to uniquely identify the student record in the table -



5) Foreign key:- A foreign key is a column which is added to create a relationship with another table. foreign keys help us to maintain data integrity and also allows navigation between two different instances of an entity. Every relationship in the model needs to be supported by a foreign key.

Ex:-

Emp-ID	Name	Address
1	Rani	Delhi
2	Jay	Mum.
3	veeru	Kol

DEP-ID	Dname	Emp-ID
1	IT	1
2	HR	1
3	Admin	2

6) Compound key:- compound key has many fields which allow you to uniquely recognize a specific record. It is possible that each column may be not unique by itself within the database. When combined with the other column or columns the combination of composite keys become unique.

Ex:-

OrderNO	Product ID	ProductName	Quantity
B005	TAP10245 9	Mouse	5
B005	DKT321573	USB	10
B005	OHG446789	LCD Monitor	20
B004	DKT321573	USB	15
B002	OHG446789	Laser printer	3

OrderNO and Product ID can't be a primary key as it does not uniquely identify a record. A compound key of Order ID and Product ID could be used as it uniquely identified each record.

7) Composite key:- A key which has multiple attributes to uniquely identify rows in a table is called a composite key. The difference between compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or may be not a part of the foreign key.

* Differences between Primary key and Foreign key

Primary key	Foreign key
1. primary key is a column or combination of columns that uniquely defines a row in a table of a relational database	1. Foreign key is an attribute of table reference as primary key in another table.
2. Primary keys enforce Entity integrity by uniquely identifying entity instances	2. Foreign keys enforce referential integrity by completing an association between two entities
3. primary key is Unique key	3. Foreign key always refers to primary key
4. can not be NULL	4. can be NULL

* The Enhanced ER Model (or) Class Hierarchy: -

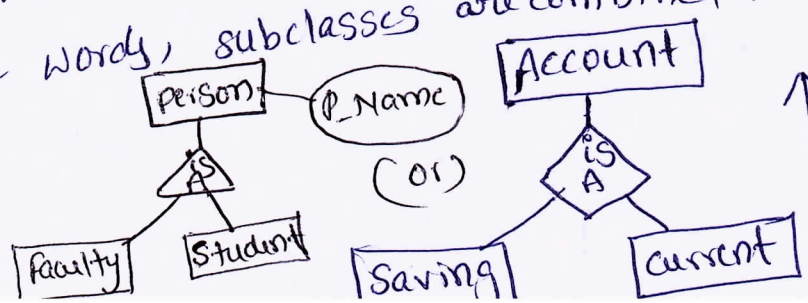
The complexity of data increased in the late 1980's. It become more and more difficult to use the traditional ER model for database modelling. Hence some improvements or enhancements were made to the existing ER Model to make it able to handle the complex applications better.

Hence, as part of the Enhanced ER Model, along with other improvements, three new concepts were added to the existing ER Model, they were:

1) Generalization
2) Specialization
3) Aggregation

1) Generalization:-

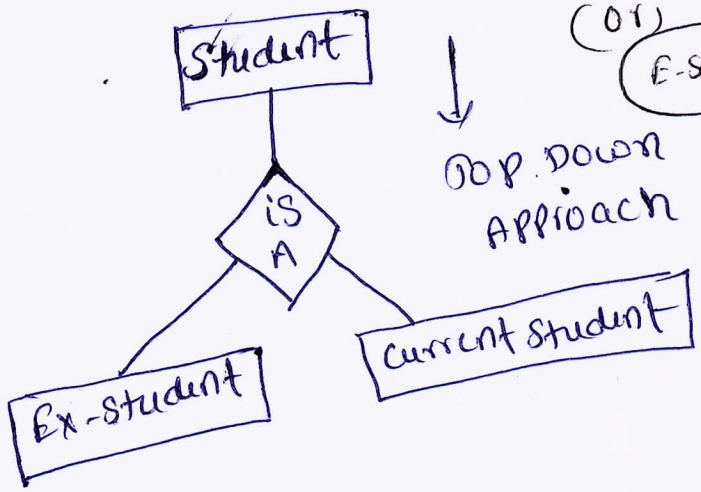
Generalization is a bottom-up approach in which two lower level Entities combine to form a higher level Entity. In Generalization, the higher level Entity can also combine with other lower level Entities to make further higher level entity. It's more like superclass and subclass system, but the only difference is the approach, which is bottom-up. Hence, Entities are combined to form a more generalized Entity, in other words, subclasses are combined to form a super-class.



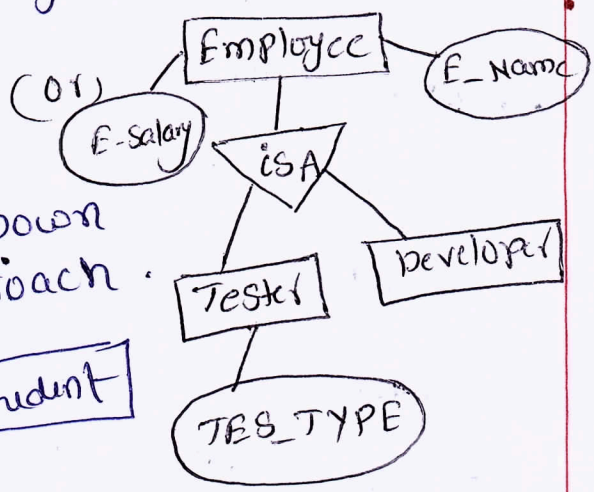
Ex: Saving and current account types entities can be generalized and an entity with name Account can be created, which covers both.

* Specialization:-

Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, a higher-level entity may not have any lower-level entity sets, it's possible

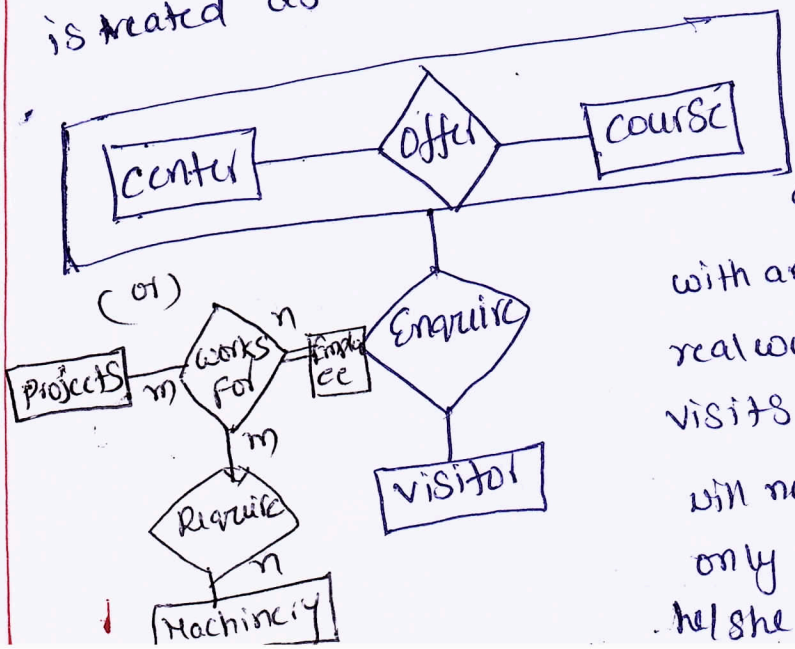


Top Down Approach.



* Aggregation:-

Aggregation is a process when relation between two entities is treated as a single entity.



In the above diagram, the relationship between center and course together, is acting as an entity, which is in relationship with another Entity visitor. Now in real world, if a visitor or a student visits a coaching center, he/she will never Enquire about the center only or just about the course, rather he/she will ask Enquire about both.

* Conceptual design with the E-R model for large Enterprise

* Conceptual Design using the ER Model: -

Developing an ER diagram presents several choices:

Design choices: -

→ should a concept be modled as an Entity or an attribute?

→ should a concept be modled as an Entity or a relationship?

→ what are the relationship sets as their participating Entity sets?

Should we use binary or ternary relationships?

→ should we use aggregation?

Constraints in the ER Model: -

→ A lot of data semantics can be captured

→ But some constraints cannot be captured in ER diagrams

Need for further refining the schema: -

- Relational schema obtained from ER diagram is a good first step. But ER design subjective & can't express certain constraints: So this relational schema may need refinement.

Entity Vs. Attribute: -

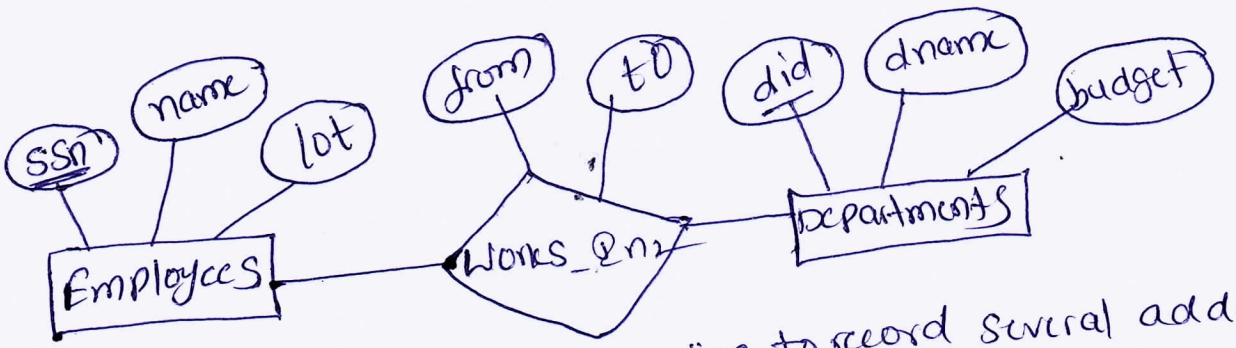
→ Should address be an attribute of Employees or an Entity

→ Depends upon how we want to use address information and the semantics of the data:

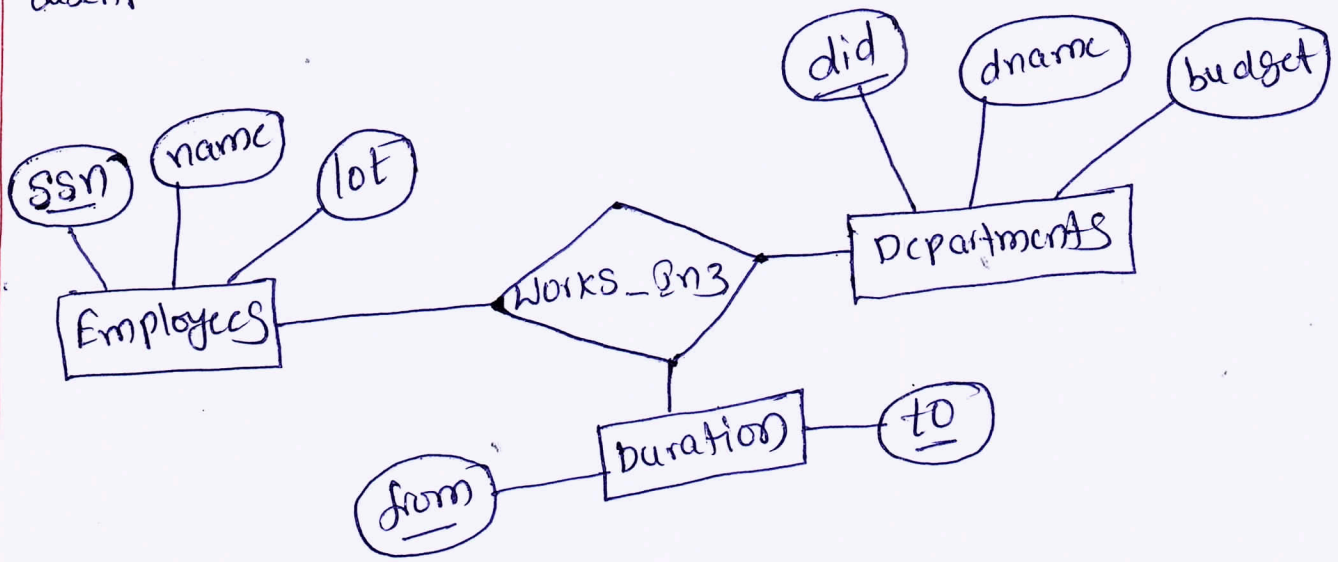
- If we have several addresses per employee, address must be an Entity (since attributes cannot be set-valued).

→ If the structure (city, street) is important, address must be modled as an Entity (since attribute values are atomic)

→ Works_In2 doy not allow an employee to work in a department for two or more periods.



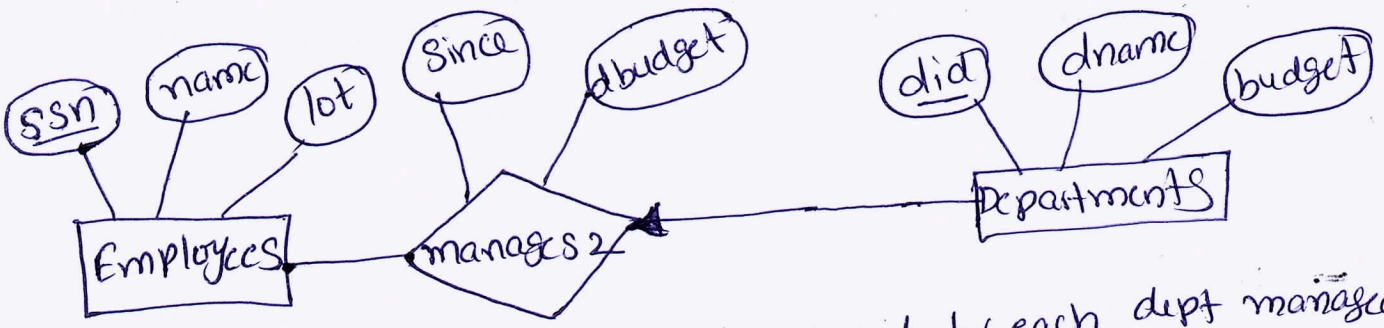
→ Similar to the problem of wanting to record several addresses for an Employee: We want to record several values of the descriptive attributes for each instance of this relationship.



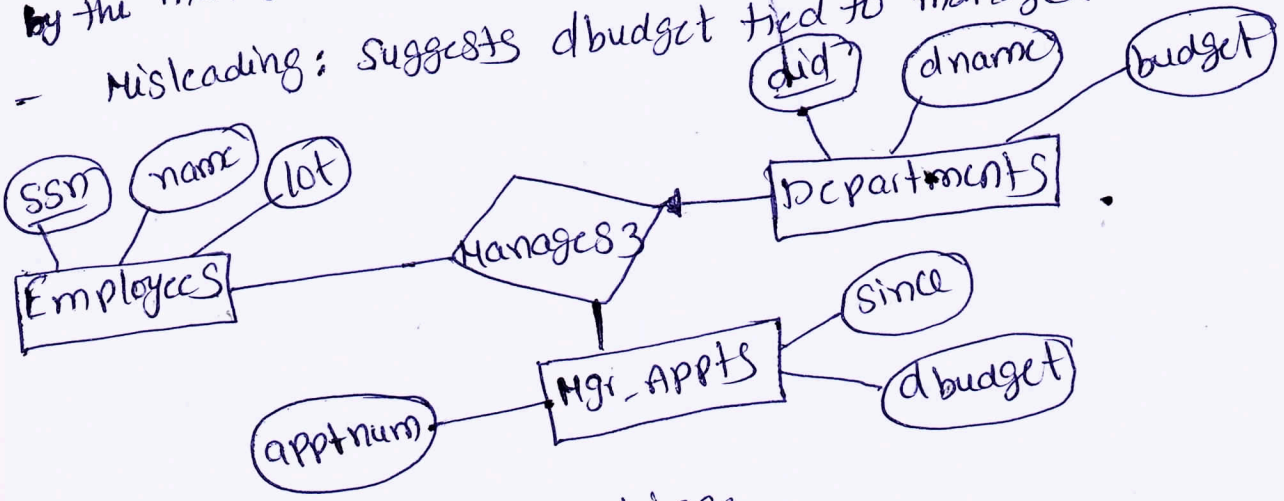
Entity vs Relationship: —

→ First ER diagram OK if a manager gets a separate discretionary budget for each dept.

→ What if a manager gets a discretionary budget that covers all managed dpts?



- Redundancy of dbudget, which is stored for each dept managed by the manager.
- Misleading: Suggests dbudget tied to managed dept



Binary vs Ternary Relationships:-

An employee can own several policies, each policy can be owned by several Employees, and each dependent can be covered by several policies.

→ A policy cannot be owned jointly by two or more employees

→ Every policy must be owned by some employee

→ Dependents is a weak entity set, and each dependent entity is uniquely identified by taking pname in conjunction with the policyid of a policy entity.

The first requirement suggests that we impose a key constraint on Policies with respect to Covers, but this constraint has the unintended side effect that a policy can cover only one dependent.

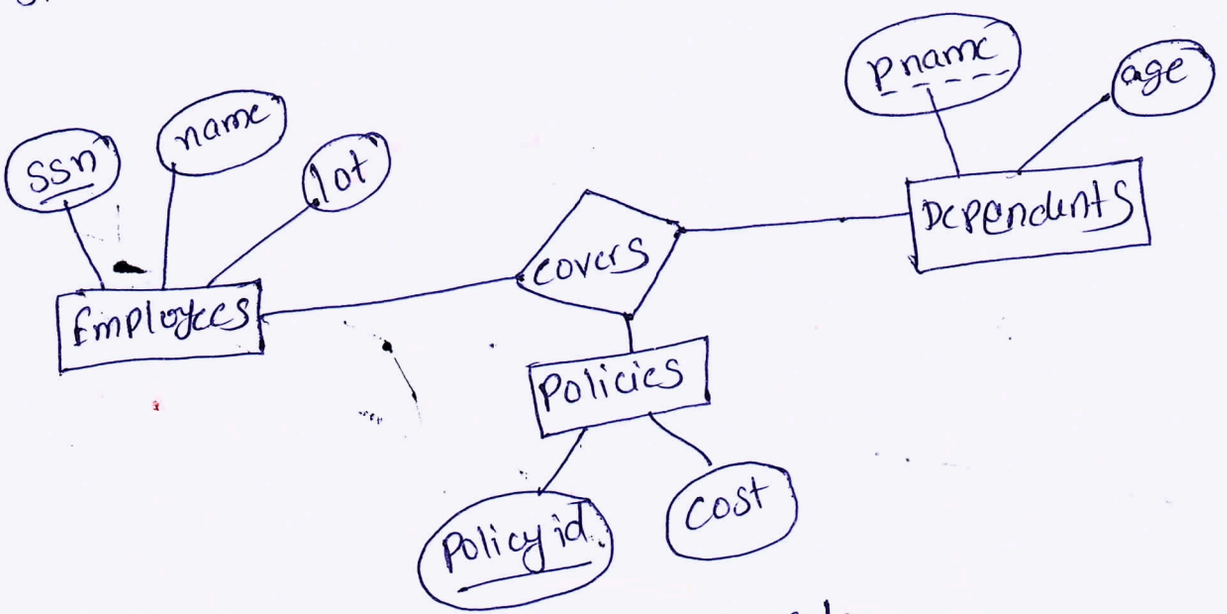


Fig: Policies as an Entity set

The second requirement suggests that we impose a total participation constraint on Policies. This solution is acceptable if each policy covers at least one dependent

The third requirement forces us to introduce an identifying relationship that is binary

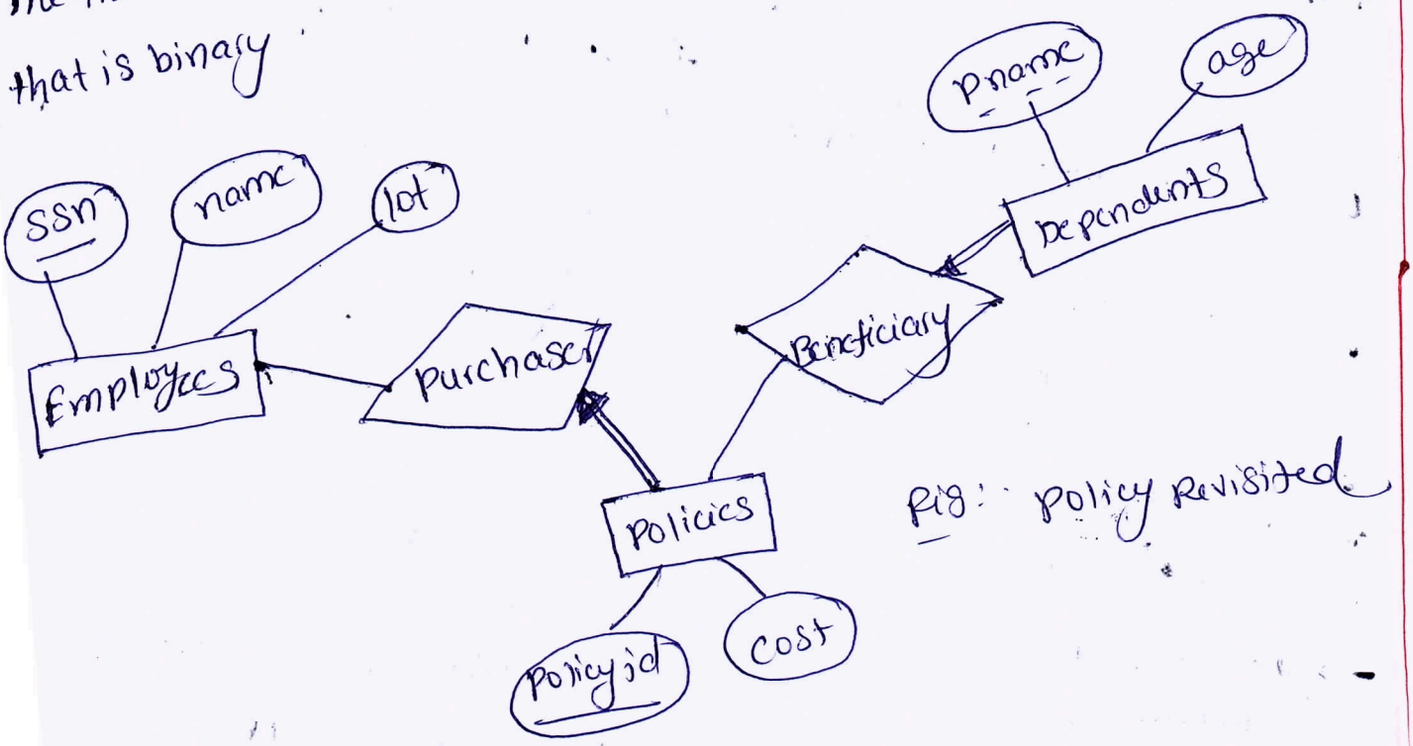
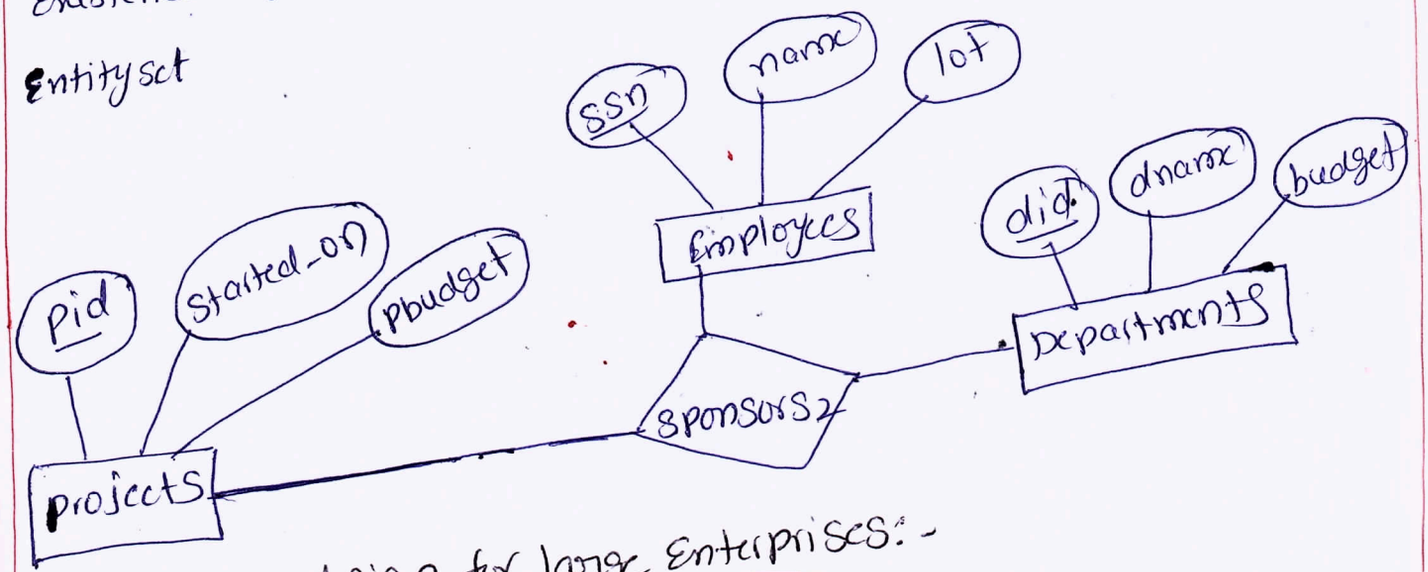


Fig: Policy Revisited

Aggregation vs Ternary Relationships :-

Aggregation or a ternary relationship is mainly determined by the Existence of a relationship that relates a relationship set to an Entity set



* Conceptual design for large Enterprises :-

→ The process of conceptual design consists of more than just describing small fragments of the application in terms of ER diagrams.

→ For a large Enterprise, the design may require the efforts of more than one designer and span data and application code used by a number of user groups.

→ An important aspect of the design process is the methodology used to structure the development of the overall design and to Ensure that the design takes into account all user requirements and is consistent.

→ The usual approach is that the requirements of various user grs are considered, any conflicting requirements are somehow resolv and a single set of global requirements is generated at the end of th requirements analysis phase

→ To integrate multiple conceptual schemas, we must estabi correspondences between entities, relationships and attrib and we must resolve numerous kinds of conflicts.

* Constraints in ER:-

An E-R Enterprise schema may define certain constraints to which the contents of a database must conform. Every relation has some condition).

1) Mapping constraints (cardinalities):-

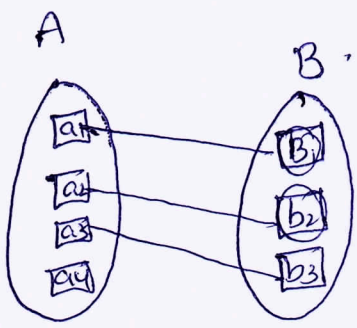
→ Mapping ~~cons~~ cardinalities or cardinality ratios, express the number of Entities to which another Entity can be associated via a relationship set

→ Mapping cardinalities are most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than two entity sets.

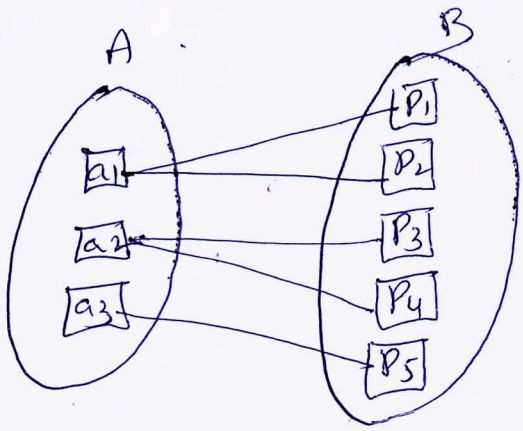
→ We shall concentrate on only binary relationship sets

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

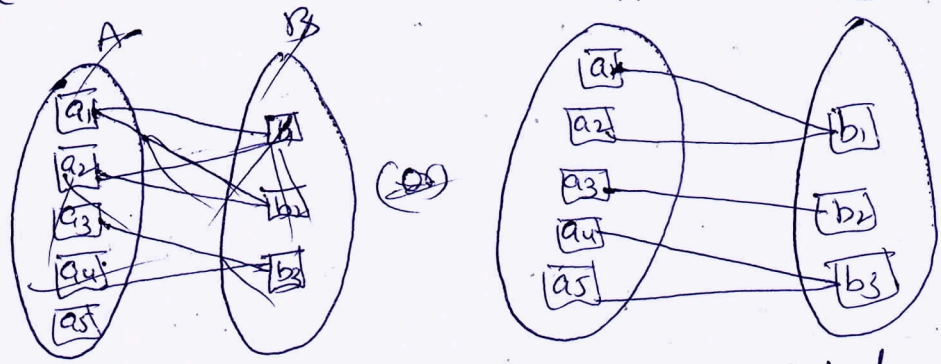
→ one to one:- An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



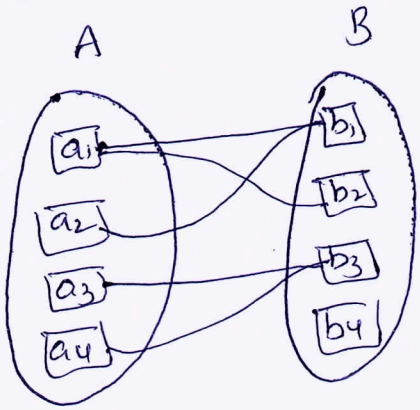
One-to-many:- An entity in A is associated with any number (zero or more) of entities in B. An entity in B, however, can be associated with at most one entity in A.



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 (zero or more) of entities in A.



Many-to-many:- An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.



2) participation constraints:-

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

→ If only some entities in E participate in relationships in R, the participation of ~~A in the~~ Entity set E in relationship R is said to be partial.

→ The participation of B in the relationship set is total while the participation of A in the relationship set is partial. The participation of both A and B in the relationship set are total.

3) key constraints:-

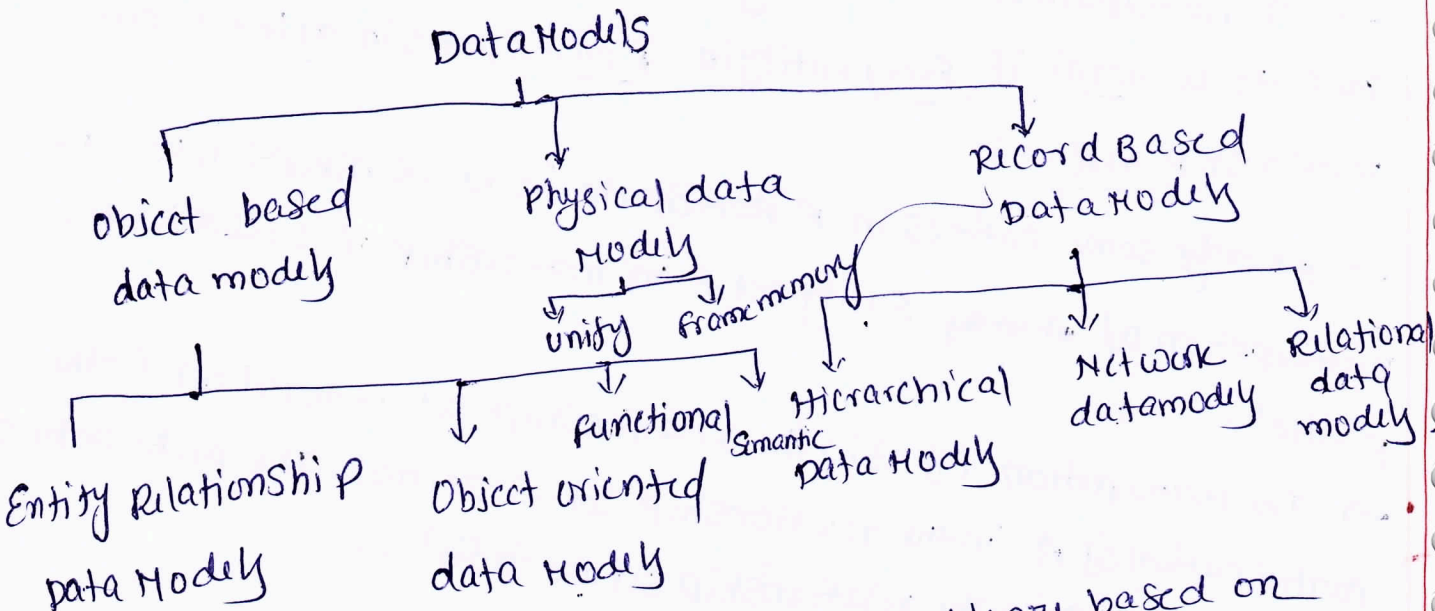
→ The values of the attribute values of an entity must be such that they can uniquely identify the entity. In other words, no two entities in an entity set are allowed to have exactly the same value for all attributes.

→ The notion of a key for a relation schema applies directly to entity sets. That is, a key for an entity is a set of attributes that suffice to distinguish entities from each other.

→ The concepts of superkey, candidate key, and primary key are applicable to entitysets just as they are applicable to relationschemas.

→ The primary key of an entity set allows us to distinguish among the various entities of the set. Let R be a relationship set involving entitysets $E_1, E_2 \ldots E_n$

* Data Models:-



Record Based Data Model Ex:- These data models are based on application and user levels of data. They are modled considering the logical structure of the objects in the database. This data model defines the actual relationship between the data in the Entities.

Physical data Model:- A physical data model is a database specific model that represents relational data objects (Ex: tables, columns, primary and foreign key) and their relationships. A physical data model can be used to generate DDL commands / statements which can then be deployed to a database server.

Object based data models:- An object data model is a data model based on object-oriented programming, associating methods (procedures) with objects that can benefit from class hierarchies. "Objects" are levels of abstraction that include attributes and behavior.

* Database System structure Explanation :-

Database System Structure consists of:

→ Query processor
→ Storage Manager
→ Disk Storage
→ outermost is user level (Naive, Sophisticated users, Application programmer, Analyst)

→ Query processor :- It translates statements in a query language into low-level instructions the database manager understands. The query processor simplifies and facilitates access to data. The query processor includes the following component.

1) DDL Interpreter :- The DDL interpreter interprets DDL statements and records the definition in the data dictionary.

2) DML compiler :- The DML compiler translates DML statements in a query language into an Evaluation plan consisting of low-level instructions that the query Evaluation engine understands. The DML compiler also performs query optimization, which is it picks the lowest cost evaluation plan from among the alternatives

3) Query Evaluation Engine :- Query evaluation Engine executes low level instructions generated by the DML compiler.

4) DML precompiled converts DML statements Embedded in an application program to normal procedure calls in a host language. The precompiled interacts with the query processor.

5) DDL Compiler :- Converts DDL statements to a set of tables containing meta data stored in a data dictionary.

Storage Manager:- The storage manager is important because database typically require a large amount of storage space. So it is very important efficient use of storage, and to minimize the movement of data to and from disk.

A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and the queries submitted to the system. The storage manager is responsible for the interaction with the file manager.

The storage manager translates the various DML statements into low level file system commands. The storage manager is responsible for storing, retrieving and updating data in the database. The storage manager components:

→ Authorization and Integrity Manager:- Authorization and Integrity Manager tests for the satisfaction of integrity constraints and checks the authority of users to access data.

→ Transaction Manager:- Transaction Manager ensures that the database remains in a consistent state and allowing concurrent transactions to proceed without conflicting.

→ File Manager:- The file manager manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

→ Buffer Manager:- The Buffer manager is responsible for fetching the data from disk storage into main memory and deciding what data to cache in main memory.

The storage manager implements the following data structures as part of the physical system implementation.

Data file, Data Dictionary, Indices, Datafiles Stores the database itself. The Data dictionary stores Meta data about the structure of database, in particular the schema of the database. Indices provide fast access to data items.

Data Files:- Store the database itself.

Data Dictionary:- stores information about the structure of the database. It is used heavily. Great Emphasis should be placed on developing a good design and efficient implementation of the dictionary.

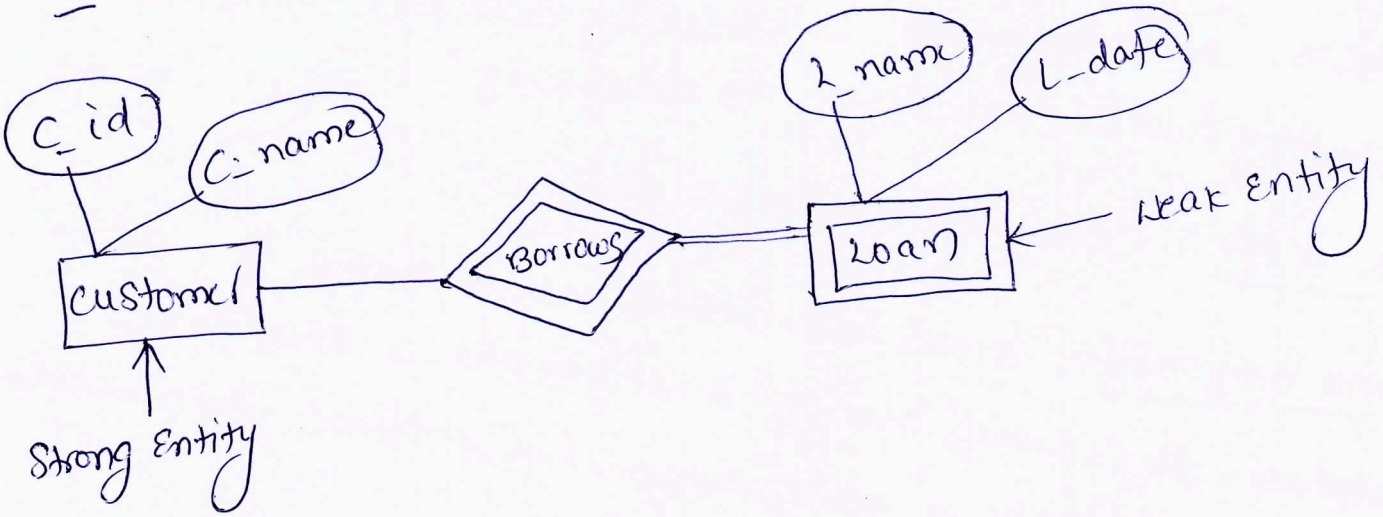
Indices:- Provide fast access to data items holding particular value.

* Difference b/w Strong Entity and Weak Entity:-

Strong Entity:- Strong entity is not dependent of any other entity in schema. Strong entity always has primary key. Strong entity is represented by single rectangle. Two strong Entities relationship is represented by single diamond. Various strong entities together makes the strong entity set.

Weak Entity:- Weak entity is depend on strong entity to Ensure that existance of weak entity. Like strong entity. Weak Entity does not have any primary key, weak entity is represented by double rectangle. The relation between one strong and one weak entity is represented by double diamond.

Ex:-


C_id
C_name
Customer
Borrows
L_name
L_date
Loan
Weak Entity
Strong Entity