RDBMS :- Introduction To RDBMS

* **Data :-** Data is raw facts. From facts we mean that does not convey any meaning. Data generally consists of numbers and text. In addition, data also includes multimedia files such as an image, audio, files etc.

* **Information :-** The manipulated and processed data is called information. It conveys proper meaning and is used in decision making.

* **Knowledge :-** Organized and meaningful information used purposefully is known as knowledge. The sum of information and experience which can be used for a purpose is called knowledge.

* Difference between data and information.

Data	Information
1. The raw facts and figures is called data.	1. Meaningful and processed data is called Information.
2. Sets of words is called data	2. Words arranged in proper manner is called information
3. Data when organized can be used to represent useful information.	3. Useful information helps people in making decisions.
4. ~~Helps in decision making~~	5. ~~Doesnot help~~

* Qualities of an Information :-

1) It must be meaningful.
2) It must be accurate & clear
3) It should convey to the point and relevant one.
4) It should confirm and correct the previous knowledge.
5) It should reach at proper time.

* Database :- Collection of related relations is called database.

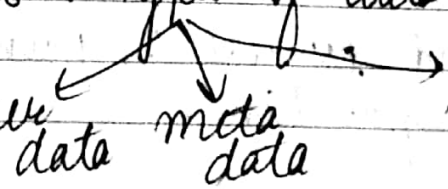
* Features operations of Data in Database :-
  → Insertion → Deletion
  → Updation → Selection

* Record :- Collection of related data items.
* Table :- Collection of related records. It is also called relations
* Attributes :- The columns in relation are called attributes.
* Domain :- The columns in relation are called domain. It is also called attributes & ~~domain~~ fields.
* Tuples :- The rows in relation are called tuples. It is also called records.

Database System :- A computer based record keeping system. Its purpose is to maintain information, that is relevant to the organization and helps in decision making.

Components of Database :-
→ Data
→ Hardware
→ Software
→ users

Data :- It acts as a bridge between machine and users. Types of data



properties of data → consistent + valid
shared

classmate
Date
Page

Hardware :- The hardware consists of secondary storage devices such as magnetic disks, optical disks etc. (physical parts)

Software :- It consists of DBMS which acts as a bridge between the user and the database. The DBMS softwares includes Oracle, Mysql server, DB2 etc.
Foxbase, Sybase, Ingress
(UNIX)

Users :- Users take information from database to fulfile their primary business responsibilities.

- Types of users :-
  - → Database Administrators (DBA)
  - → Database Designers
  - → End Users
  - → Application Programmers
  - → Data Entity Operators :- Proficient typists and are respos. for adding large data in database

DBA :- DBA is a person or a group of person who is responsible for management of database.
It includes defining database structure, storage structure etc. It helps in maintaining backups and recoveries. Grant and Revoke are used in it.

Database Designers :- These are the users who design a database. They are responsible for identifying the data to be stored in the database.

End Users :- End users are the people who interact with the database through applications or utilities.
Classification :-
(i) Casual :- Access occasionally by entering queries (not aware of presence of database system)
(ii) Naive :- Access data through application programmers (aware of different systems)
(iii) Stand-alone / On-line :- They maintain personal databases by using ready made program packages.

Application programmers :- They are responsible for writting application programs that use the database. These are written in general purpose programming languages like C, C++, JAVA etc.

DBMS :- It stands for database management system. It is the appropriate package of software that manages database.

Database languages :-
Four types of database languages :-
(i) DDL
(ii) DML
(iii) DCL
(iv) TCL

(i) DDL :- It stands for database definition language. These languages are used to define the implementation details of the database schemas, which are hidden by users.
Eg :- create, ~~access~~ ~~delete~~,

(ii) DML :- It stands for database Manipulation language. It is used to modify, delete and manipulate the data.
Eg :- modify, ~~delete~~ update, retrieve, insert, delete

(iii) DCL :- ~~Data~~ It stands for data control language. It controls the access to data.
Eg :- GRANT, REVOKE

(iv) TCL :- It stands for transaction control language. It helps when transaction held in database
Eg :- COMMIT, SAVEPOINT, ROLLBACK.

Data files :- It contains data of database

Data dictionary :- It is the description of data in a database

Functions of DBA :-

1) Storage structure and access :- The DBA decides how the data is stored and represented in a database.
2) Assisting application programmers :- It provides assistance to application programmers to develop an application.
3) Approving data access :- It determines which user needs access to which part of data.
4) Backup and Recovery :- It ensures this periodically backing up the database.
In case of any failure or virus attack database is recovered from this backup.

Mapping is supplied by DBA, Responsible for maintaining integrity of database (data contained in database must be accurate)

Advantages of DBMS :-

1. Controlling Data Redundancy :- (Duplication, Wastage of storage space, errors coz of same data, wastage of time)
2. Elimination of Inconsistency
3. Provides backup & recovery
4. Cost of maintaining & developing is lower
5. Concurrency control :- Provides concurrent access & ensures the correctness of data.

Shared data, Integrity of data, Security, Conflict Resolution, Data Independency

Metadata :- Data about the data.

File system V/s Database

Parameters	Database	File
1. Definition	Database is the collection of related data at the centralized location	A data file is a collection of related records stored on a storage medium such as a hard disk or optical disc.
2. Files	Tables are files	Files are files
3. Redundancy	Less	More
4. Inconsistancy	Less	More
5. Size	Large	Small
6. Sharing of data	Easier due to centralized system	complex as file is not centralized
7. Backup & recovery	possible	Not possible
8. Complexity	Higher	Simple
9. Cost	Expensive	Cheap
10. Expertise	Needed	not needed

****

Instances :- State of database at any point of time
Collection of information stored in Database at a particular time. Also called extension of DB.

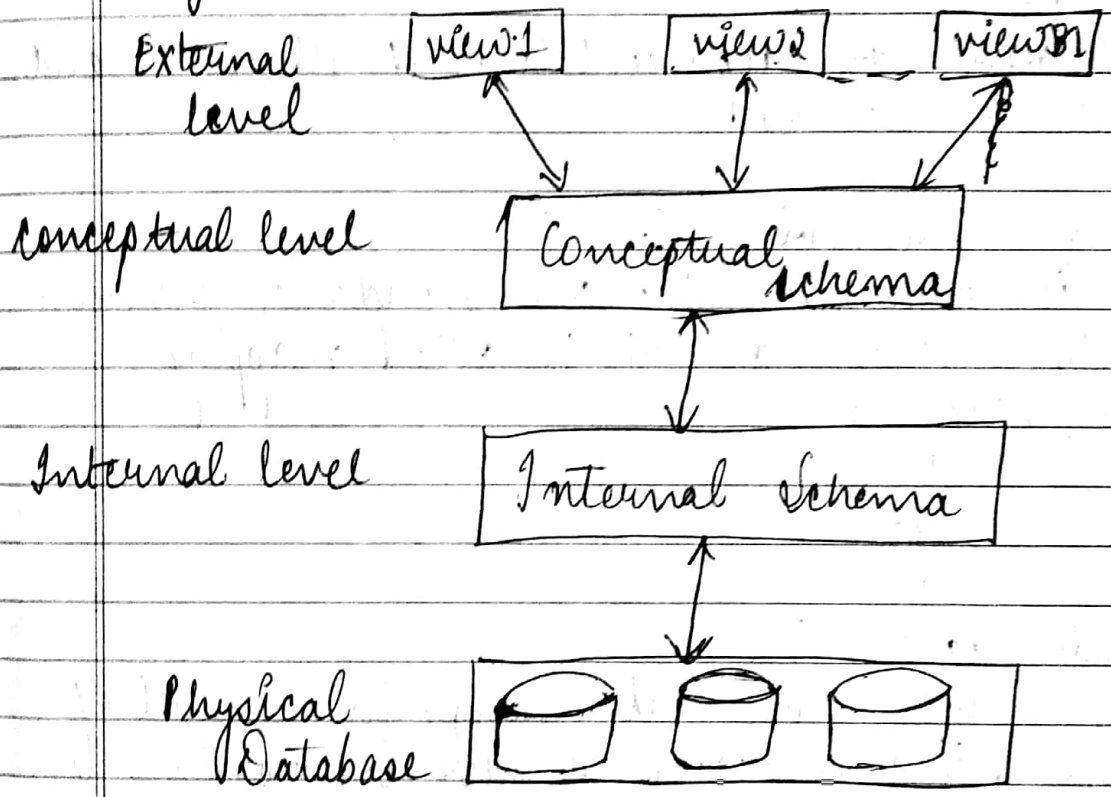
Database schema :- Overall design of database
→ Outline or plane that describes the records & relationships existing at perticular level. eg.
→ Also called intension of database.

Three level Architecture (TLA) / 3-level Architecture

DBMS architecture is a framework where DBMS structure is described.
Its aim is designing of DBMS, to provide users with an attractive view of the data by hiding certain details of how data is stored and maintained.

Levels
→ External level (Physical level)
→ Conceptual level (Logical level)
→ Internal level (Storage view)

Diagram :-



External level :-
→ Highest level
→ Describes user view
→ Only that portions are described which are relevant to user
→ Describes external schema written in DDL

Derives Object in external view from object in Conceptual view

One conceptual view represents entire database

Conceptual level :- → Derives objects in conceptual view from internal view.

→ Describes logic structure of whole database
→ Middle level
→ Describes database entities, attributes and relationships together with constrains.
→ It is defined using DDL

Internal level :-
→ Lowest level
→ Describes how data is stored and data structures and access methods to be used by database.
→ Holds information of data
→ Written in DDL
→ Tells representation of attributes and sequence of records

Mapping between views :-
DBMS is responsible for correspondance between three types of schema. This is called mapping.
Types of Mapping
→ Conceptual/Internal → record from physical database
→ External/Conceptual → relationship of external concep. view in external view of conceptual view

Data Independence :-
→ It insures that if we make changes in any level of database, it will change the above information automatically and no need of changes further.
Types :-
* Logical → conceptual schema can be changed without affecting existing external schema
* Physical → physical storage can be changed without affecting conceptual schema
Capacity to change schema at one level without changing the schema at the next higher level.

classmate
Date
Page

RDBMS

Data Models

Data Model :- It is the specifications which describes how data is to be arranged for a specific purpose. It tells information in a database.

Importance of data models :-
- → Faster understanding of organization
- → communication with application programmers and end users
- → A good data model provides basis of a valid, maintainable application.
- → Detailed enough data model is used by designers to build physical database.

Types of data Model :-
- → Object based (conceptual) :- Define concepts, describes data
- → Record based (Representation) :-
- → Physical

Object based types
E-R, Object Oriented, Semantic & Functional

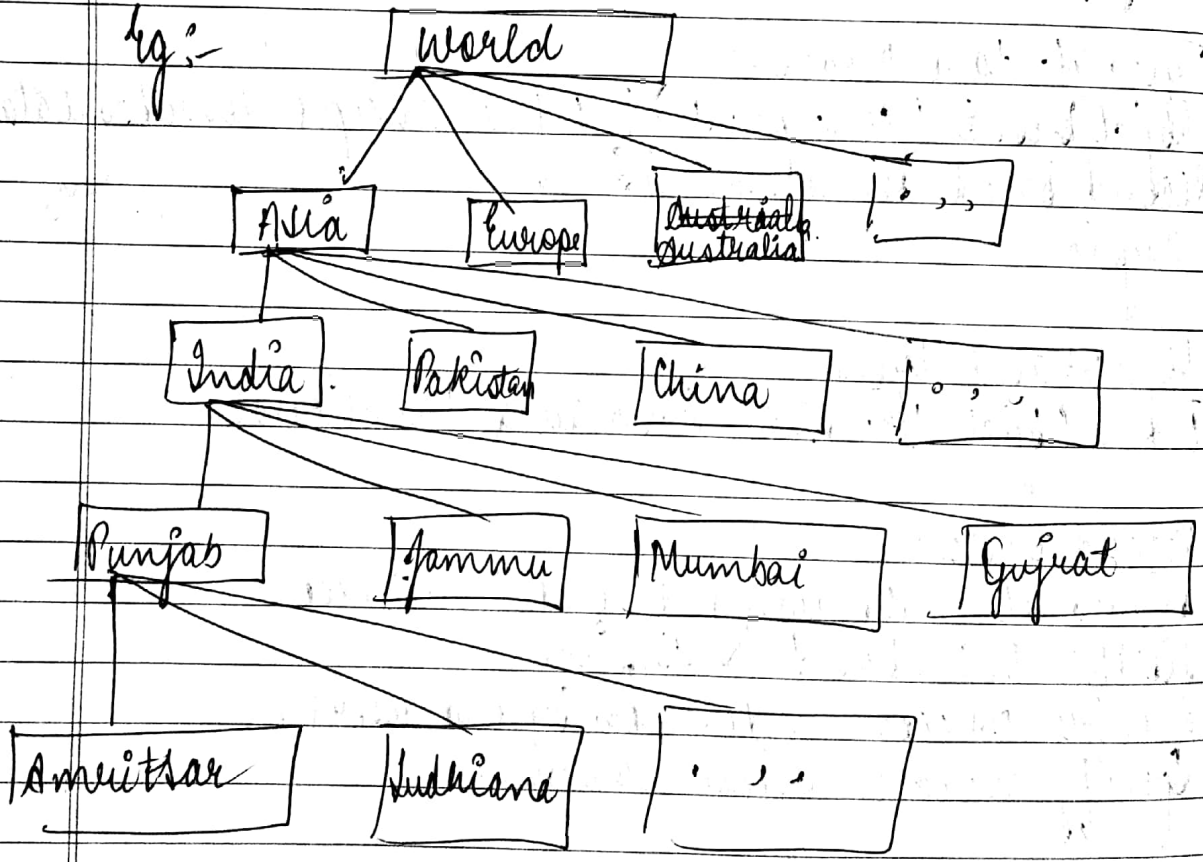
Record based
- → Represents data using record structures
- → Lie b/w Object based & Physical
- → Provide concepts understood by end users.
- → Specify overall logical structure.

Types :-
→ Hierarchical
→ Network
→ Relational

Physical data Models :-
→ Provide concepts that describes the details of how the data is stored in computer.

Hierarchical Model :-
→ One of the oldest database models
→ Organizes records in tree structure. (Parent-child)
→ Employs two main concepts :-
* Record
* Parent child relationship (1 : N relationship)

Eg :-



Operations On hierarchical model :-

1. Insertion
2. Deletion
3. Updation
4. Retrieval

Advantages :- records are in

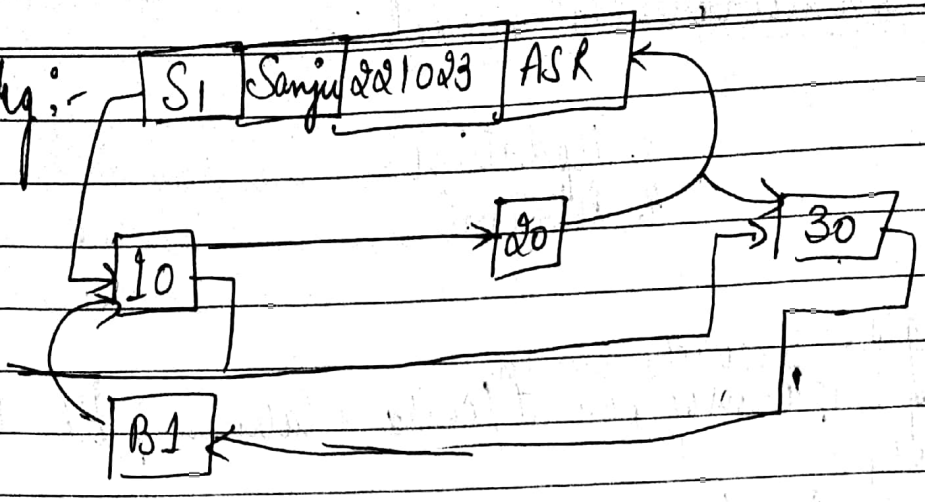
(i) Simplicity :- (Form of parent/child relationship like tree)
(ii) Data Security :-
(iii) Efficiency :- (One to many relationships)
(iv) Handle transactions easily :- (Efficient to handle large transactions)

Disadvantages :-

(i) ~~Exo~~ Complexity :- (physical links are difficult to expand)
(ii) Inflexibility :- changes in record type results in very complex system management tasks.
(iii) Lacks DDL and DML.
(iv) Management of Database :- (New relation and nodes results in complexity.)
(v) Lack of Querying facilities :- (Lack of declarative querying facilities and need for navigation of pointers to access needed information.

Network Model :-

→ Handle many to many (M:N) relationship.
→ Permits child record having more than one parent.
→ Two basic data structures :-
* Records
* Sets

Eg:- 

Operations
1.) Insertion
2.) Deletion
3.) Updation
4.) Retrieval

Advantages:-
1.) Eliminate data Redundancy
2.) Lesser storage requirements
3.) Better performance
4.) Easy access of data
5.) Data Independence

Dissadvantages:-
1.) Complexity
2.) Difficult in querring data
3.) Lack of structural Independence

Relationship Model:-
→ Most popular developments in database
→ Absolute separation of logical view & the physical view of data
→ Components:-
* Structural components:- (set of tables (relations), set of domains (data structures))
* Set of rules for maintaining integrity of database
*

Relational Model terminologies :-

(i) Relation :- Set of tuples having same attributes.
(ii) Attributes :- Smallest unit of data in relational model.
(iii) Tuple :- row or record represents a collection of information about a single row of a relation.
(iv) Domain :- Set of values from which the actual values are chosen.
(v) Cardinality :- Total number of rows or tuples in a table.
(vi) Degree :- Total number of columns or attributes in a relation.
(vii) Keys :- An attribute that can uniquely identify any particular tuple (row) within a relation (table).

Types :-

(i) Primary :- Set of attributes of table which can uniquely identify each tuple in a relation.
(ii) Foreign key :- attribute or set of attributes in relation that references a primary key of some other table.
(iii) Candidate key :- Super key with no repeated attributes
(iv) Super key :- attributes or set of attributes that uniquely identify tuples of relation.
(v) Alternate key :- keys which are not primary key.
(vi) Composite key :- key having multiple attributes to uniquely identify rows in a table
(vii) Artificial key :- key created using arbitrarily assigned data.

Query :- Find all shopkeepers who orders for a particular book B2.
Ans :- select shopkeepers from tablename where book = B2 ;

Advantages :-

(i) Simplicity :- (Easy & simple to design)
(ii) Data Independance
(iii) Recovery capability :- (makes possible for a high level query language like SQL)
(iv) Matured technology :- (Useful for representing most of the real world objects & relationships b/w them.
(v) Ability to easily ~~handle~~ take advantage of new hardware technology.

Disadvantages :-

(i) Limited ability to deal with binary large objects like images, spreadsheets, e-mail etc.
(ii) Mapping objects to relational database is difficult
(iii) Data Integrity is difficult
(iv) Not suitable for huge databases.

Insertion anomaly :- Insertion of child record without parent record.

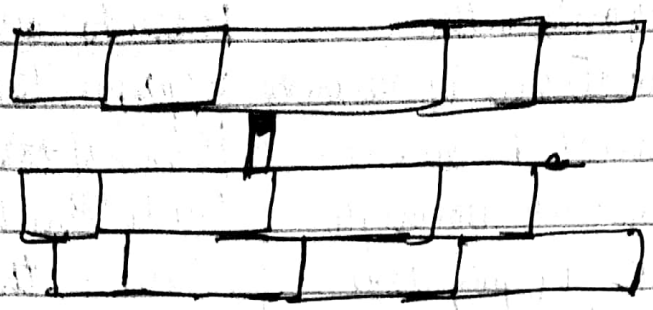
classmate
Date
Page

Difference between Hierarchical, Network & Relational Data Models

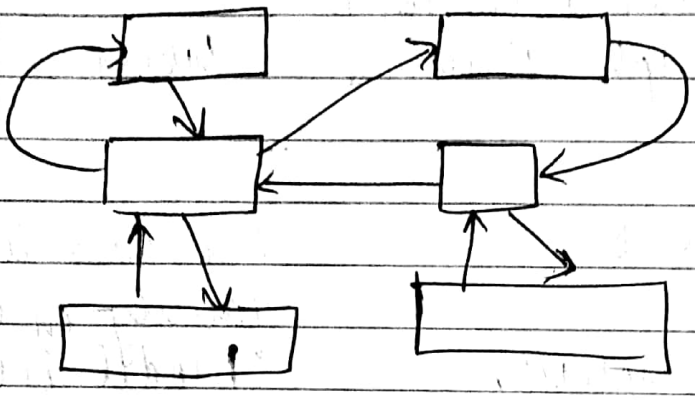
	Hierarchical	Network	Relational
1.)	Organizes the records in a tree structure	Organizes the collection of records connected to one another through links or pointers	Records are represented in the form of table and relationships between the tables.
2.)	Only one to one (1:1) & one to many (1:N) relationships	One to One (1:1), One to many (1:N) & Many to One (M:N)	All relations can be implimented
3.)	Organizes records in tree structure	Graph structure	tables
4)	Lack of declarative querrying facilities	Lack.	Provides the facility in SQL
5.)	Complexity makes database design difficult	Complexity increases the burden on the programmer for database	Model is very simple to understand
6)	Insertion of child record without parent record is not possible	Possible	Possible
7.)	Inconsistancy due to multiple occurance of child record	Single occurances of records	Due to normalizat redundancy of data is removed
8)	less data independence.	Partial	Data Independence

Basic structures :-

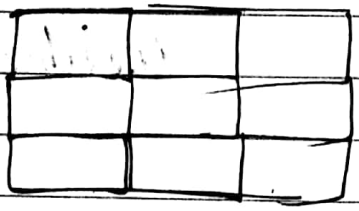
Hierarchical data Model :-



Network data Model :-



Relational data Model :-



RDBMS

ENTITY - RELATIONSHIP Model

E-R Model :- E-R model stands for entity relationship model. It is high level description of the data and the relationships among the data, rather than how data is stored.
It provides basic understanding of the nature of data and how data is used by the enterprise. It is independent of any hardware platform specially DBMS.

→ It is atomic and can't be break down.
→ It represents real world item
→ The basic object about information is stored.
→ It is an instance of entity type.

Book | Person | Employee

Representation of Entities

Features :-

1. Used for representing E-R model, can be easily converted into relations (tables) in a relational model
2. The ER model used for the purpose of good database design by the database developer so as to use that data model in various DBMS
3. Is top down approach to database design
4. Provides a standard and logical way of visualizing the data.
5. Gives precise understanding of the nature of data.

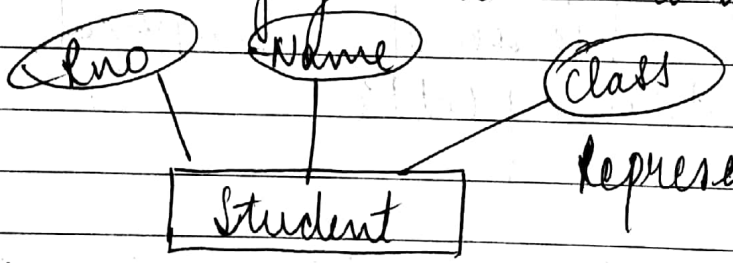
E-R Model Terminology :-

1.) **Entity :-** Collection of entities that have same attributes but different values.
An entity is instance of entity types.
Eg: RON, 231, BCA,
All entities has same attributes, so they are entity type student.
(no primary key) Entity set → payment (weak) → pay numb (a_1, a_2), date (d_1, d_2), amount

2.) **Attributes :-** Properties or characteristics of entities. (ID, 20, 20, no)

Key Features :-

→ Attributes of an entity should be unique.
→ The set of permitted values for each attribute is Domain
→ In E-R model diagram, it is represented by ellipse attached to a relevant entity by a line labelled with entity name.



Representation of Attributes.

→ Attributes should uniquely identify the entity.

3.) **Entity set :-** An entity set is a collection of all instances of a particular entity type at any point of time in the database.

Types of attributes :-

(i) **Single attribute :-** It is known as atomic attribute
→ An attribute which cannot be further subdivided into smaller components.

(ii) Composite attribute :- Composite attributes is an attribute that can be further divided. The value of composite attribute is obtained by the concatenation of values of its sub parts.
Eg :- Attribute 'Name' can be composed of first_name, middle_name & last_name.

(iii) Single-valued attribute :- A single-valued attribute is the one that one that has only a single value for the Rollno attribute.

(iv) Multi-valued attribute :- A multi-valued attribute is an attribute that holds multiple values for a particular single entity. A multi-valued attribute may have lower and upper limits. Eg :- Phone no.

(v) Derived attribute :- A derived attribute whose value is derived or calculated from the value of the related attribute.

4.) Relationships :- An association between two or more entities is called a relationship.
A relationship is indicated by a "verb", connecting two or more entities. It is represented by diamond shape symbols.
Eg :- Employees are assigned a project.
Student attends class.

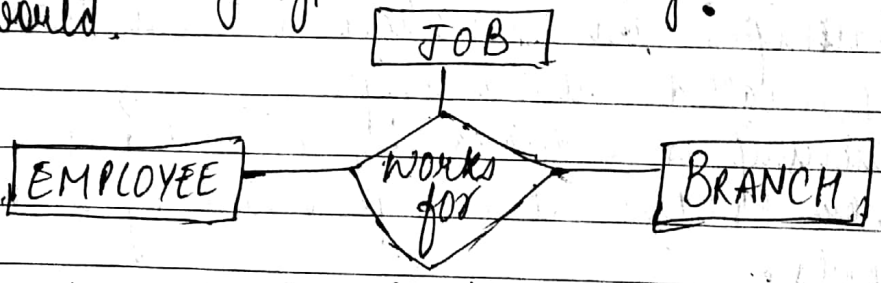
Relationships are identified by :-
> Degree > Cardinality > Connectivity > Direction

Degree of a Relationship:-
It is the number of entities associated with relationship.
Some common types are:-

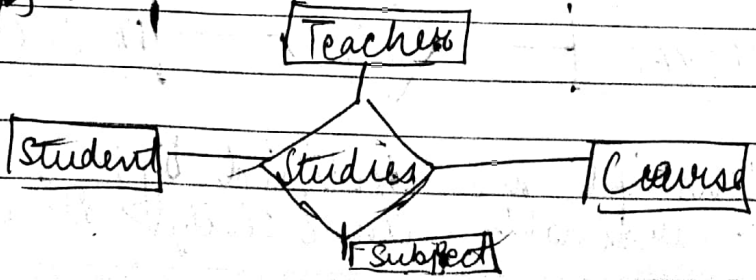
1.) Unary Relationship:- When the association exist within a single entity type it is referred as unary relationship. Degree = 1.
Eg:- For taking a particular subject in a course there is a condition to take a particular subject you have to take another subject. Like ...

2.) Binary Relationship:- When the associations exists between two entity types. Degree of binary relationship = 2.
These type of relationship are most common in real world.
Eg:- An employee is assigned a project.

3.) Ternary Relationship:- When associations exists b/w three entity types. It is rarely used in real world.



4.) Quaternary Relationship:- When association exists among four entities type. It is rearly used & is decomposed into one or more binary relationships.



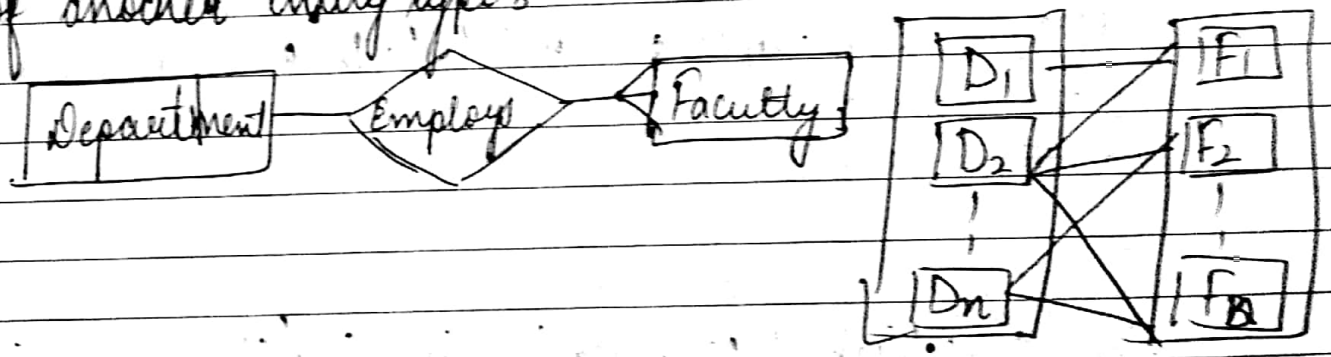
Connectivity of relationships :- It means how many instances of one entity are associated with how many instances of other entity in a relationship.
→ It describes the mapping of associated entity instances in the relationship.
→ Values of connectivity are either one or many.

→ One to One (1:1) :- It occurs when an instance of entity is associated with exactly one instance of another entity type and vice versa.

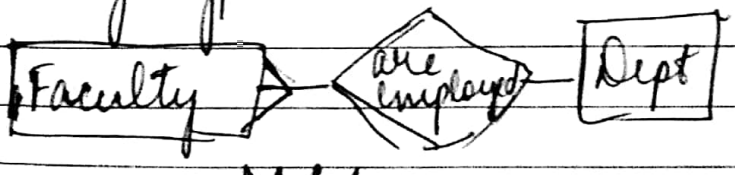
Eg :-

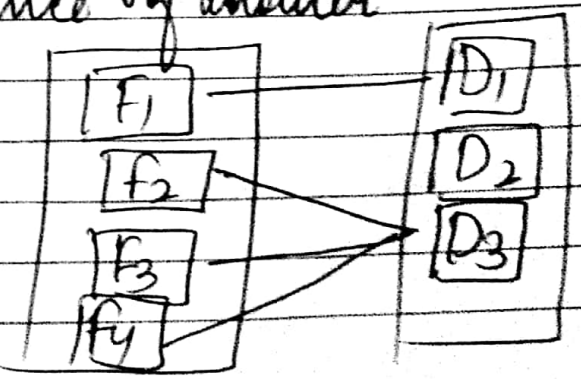
→ One to many (1:N) :- It occurs when any one instance of entity type A is associated with any number of instance of another entity type.

Eg :-



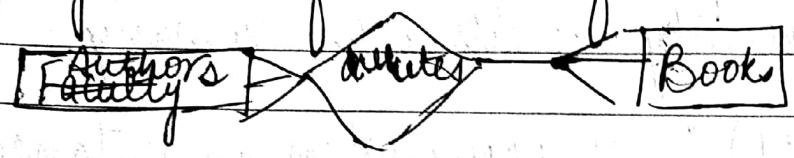
→ Many to One (M:1) This relationship occurs when any N number of instances of entity type "A" is associated with almost one instance of another entity type B.

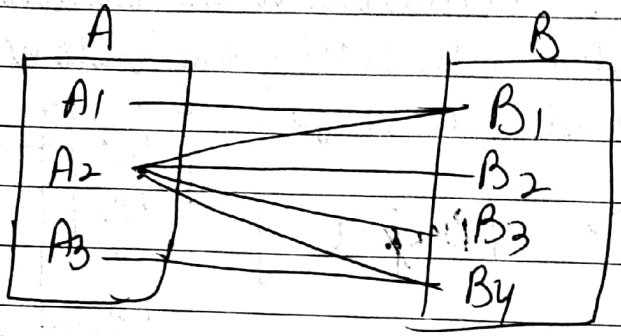




Many to Many (M:N) :- This relationship occurs when an instance of entity type A is associated with any number of instances of entity type B. and instances of B is also associated with any number of instances of A.

Eg :-





Cardinality of Relationship :-

Cardinality of relationship quantifies the relationship between entities by measuring how many instances of one entity type are related to a single instance of another.

In other words

Direction of cardinality :-

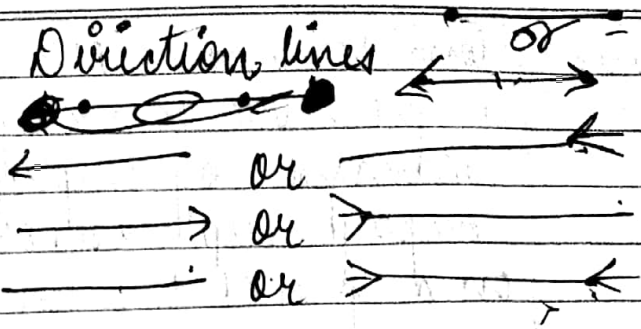
The direction of cardinality is lined (1) connecting the two entities related to each other by a relationship. The entity from which the relationship starts is the parent entity and the entity where relationship ends is called child entity.

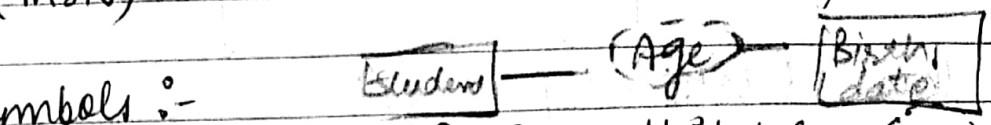
classmate
Date
Page

Relationship
→ One to One (1:1)
→ One to Many (1:N)
→ Many to One (N:1)
→ Many to Many (M:N).

Various E-R symbols :-
* Entity
* Weak Entity
* Relationship
* Identifying relationship
* Attribute
* Multivalued attribute

Direction lines





* Derived attribute
* Composite attribute
* Total participation
* Primary key
* Cardinality ratio 1:N

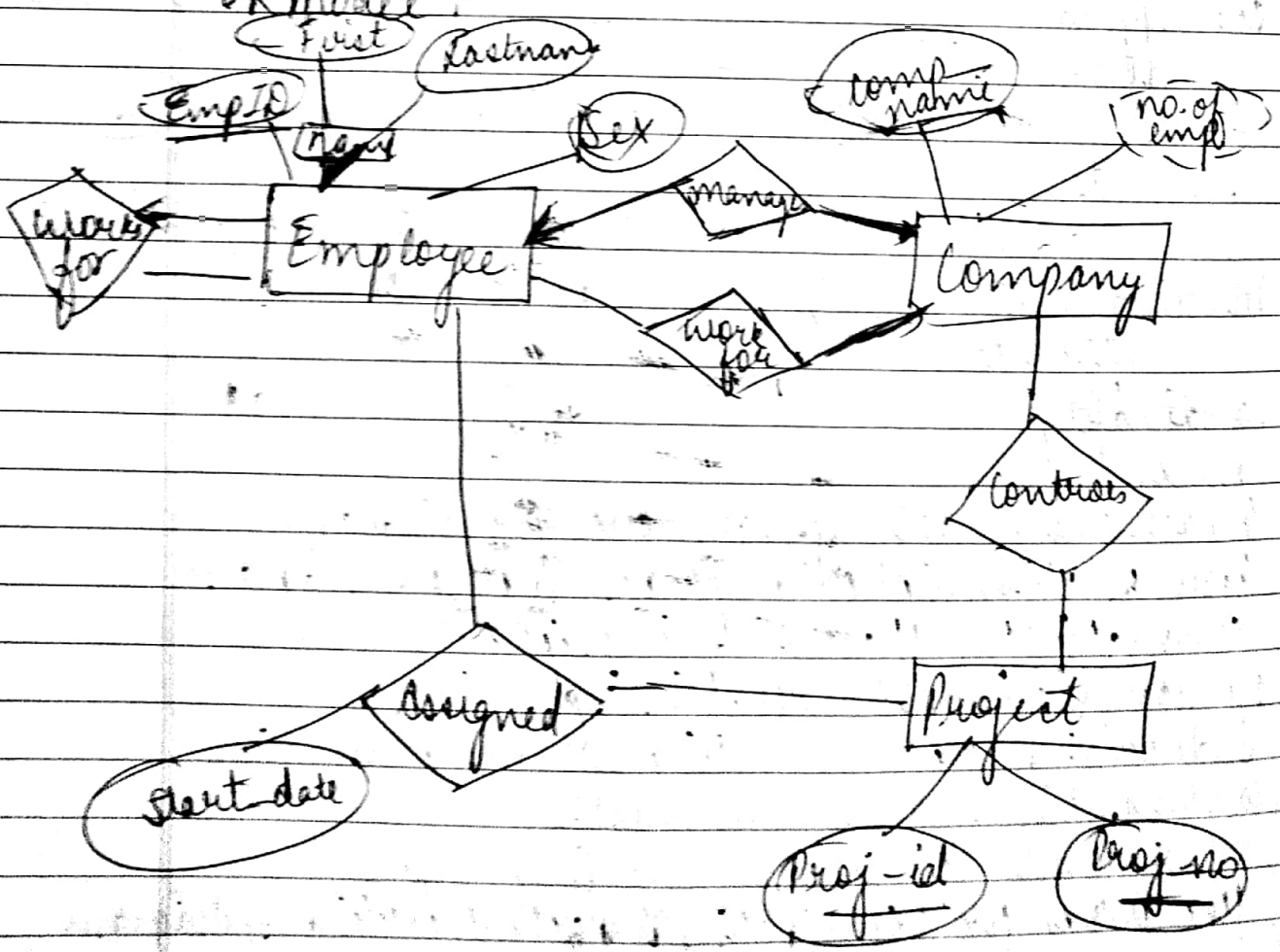
→ person can have more than phone nos.

~~Cardinality~~ Rules :-
1. Each entity only appears once per diagram
2. Each relationship only appears once per diagram
3. Connectivities of all relationships.
4. No connections b/w relationships
5. Indications for all existing optionality conditions
6. Cardinalities for all relationships
7. All composite attributes should be expanded
8. Use colors for highlighting important portion of diagram

Ques:- Consider a case of a software company that handles projects. The company controls projects which are managed by employees of the company. An employee supervises other employees and works on a project for hourly basis.

Ans:- Relationship holds:-
Employee works for a company
company controls projects
company manages employee
Employee works for employee
Employee is assigned project

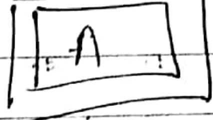
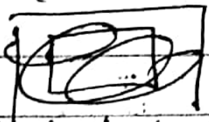
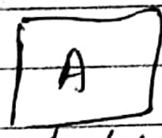
ER model:-



Weak entity :- The entity set that doesn't has sufficient attributes to form a primary key and relies on another for its identification is called weak entity set.

Strong entity :- Entity that has primary key.

Difference between strong entity set and weak entity set

	Weak entity set	Strong entity set
1.	It does not have sufficient attributes to form a primary key	It has an attribute to form a primary key.
2.	The number of weak entity set is called subordinate entity	The number of strong entity set is called dominant entity set
3.	It is represented by double rectangle	It is represented by single rectangle
4.	Its primary key is combination of partial key & primary key of strong entity set	Its primary key is chosen from one of its attributes which uniquely identifies its members.
5	Symbol 	Symbol  

* *

E-ER Model:-

It stands for enhanced ER model. ER modeling concepts were no longer sufficient to represent complex application such as data warehousing, Geographic Information system, data mining etc. So enhanced ER model was developed to represent complex applications.

Specialization:-

It is the process of defining one or more subclass entity sets, from the superclass entity set, based on some unique attribute, specific to subclass entity set.

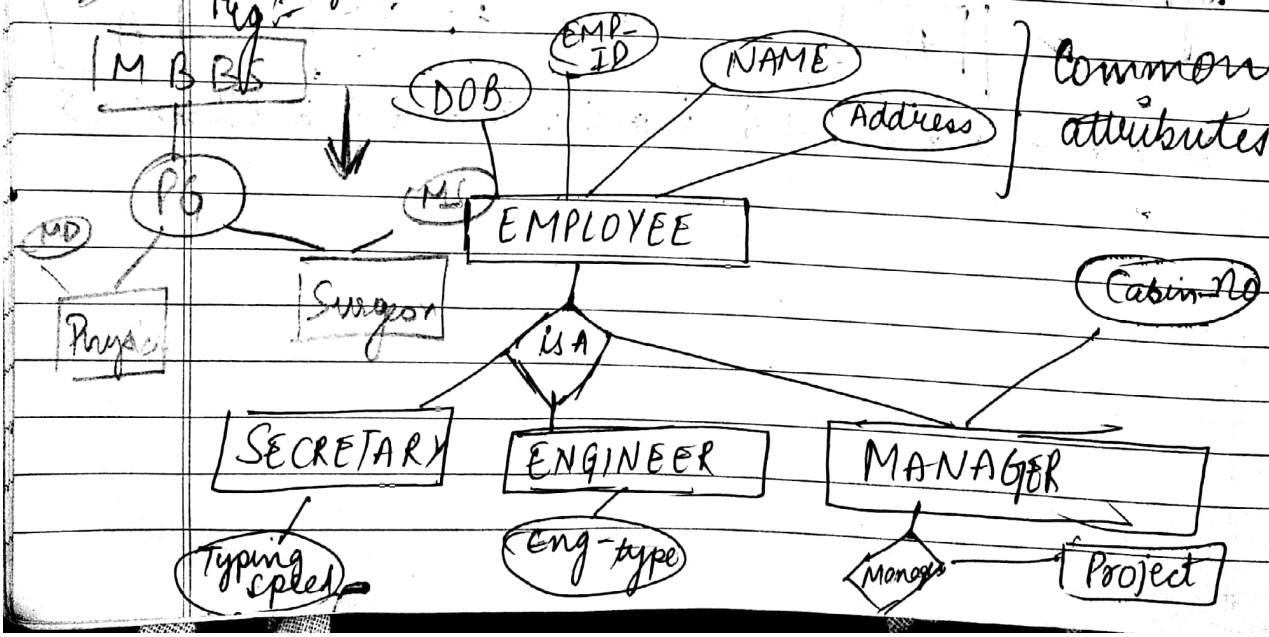
→ It is represented by a triangular component labelled 'is A', 'is a' is referred to as a superclass/subclass relationship.

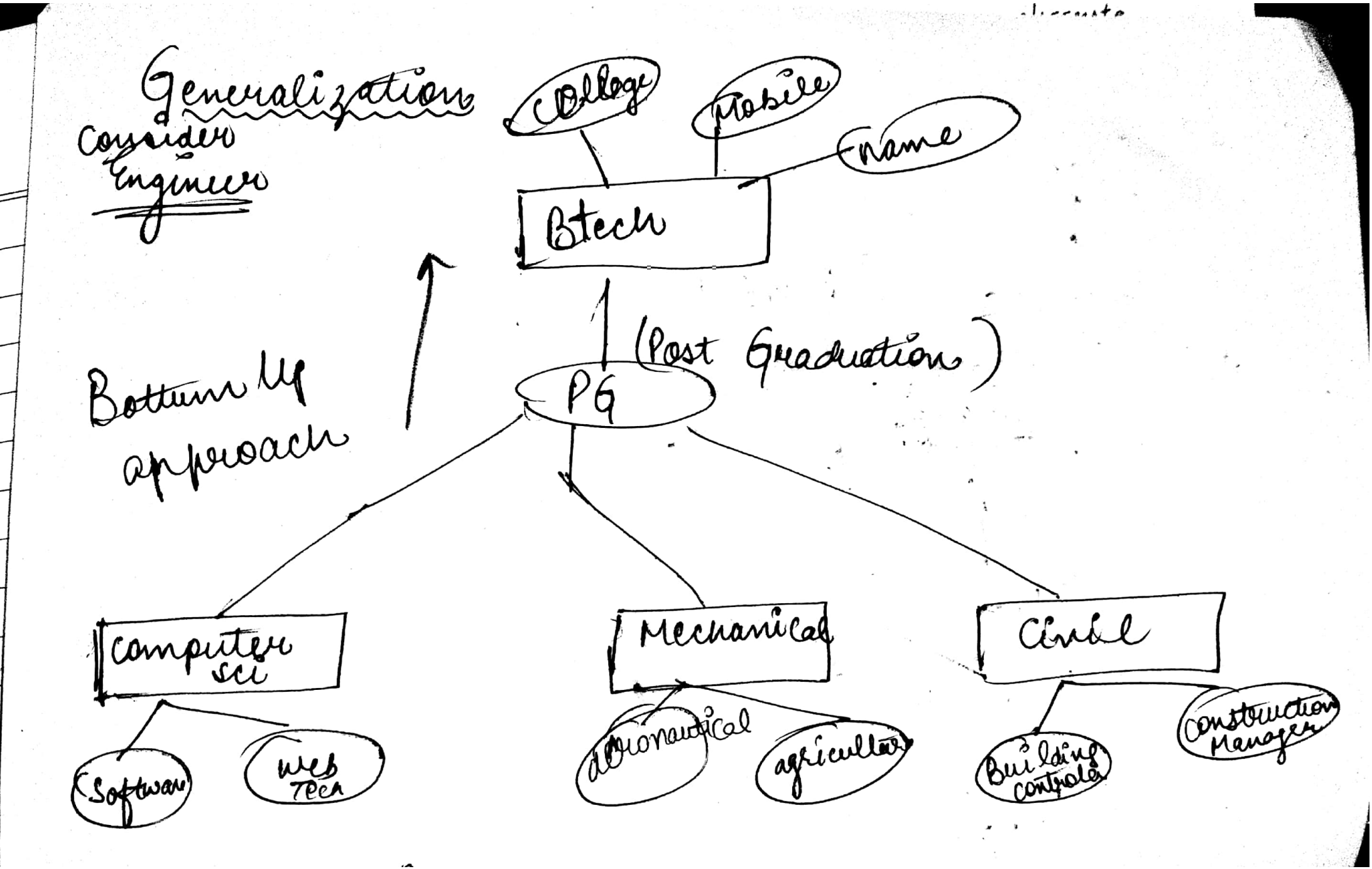
→ It is a top down process.

→ Specialization exists only where there exists specific or unique attributes of the subclass entity set.

→ If there is no unique attribute of subclass entity set, then there is no need of ~~specification~~ specialization.

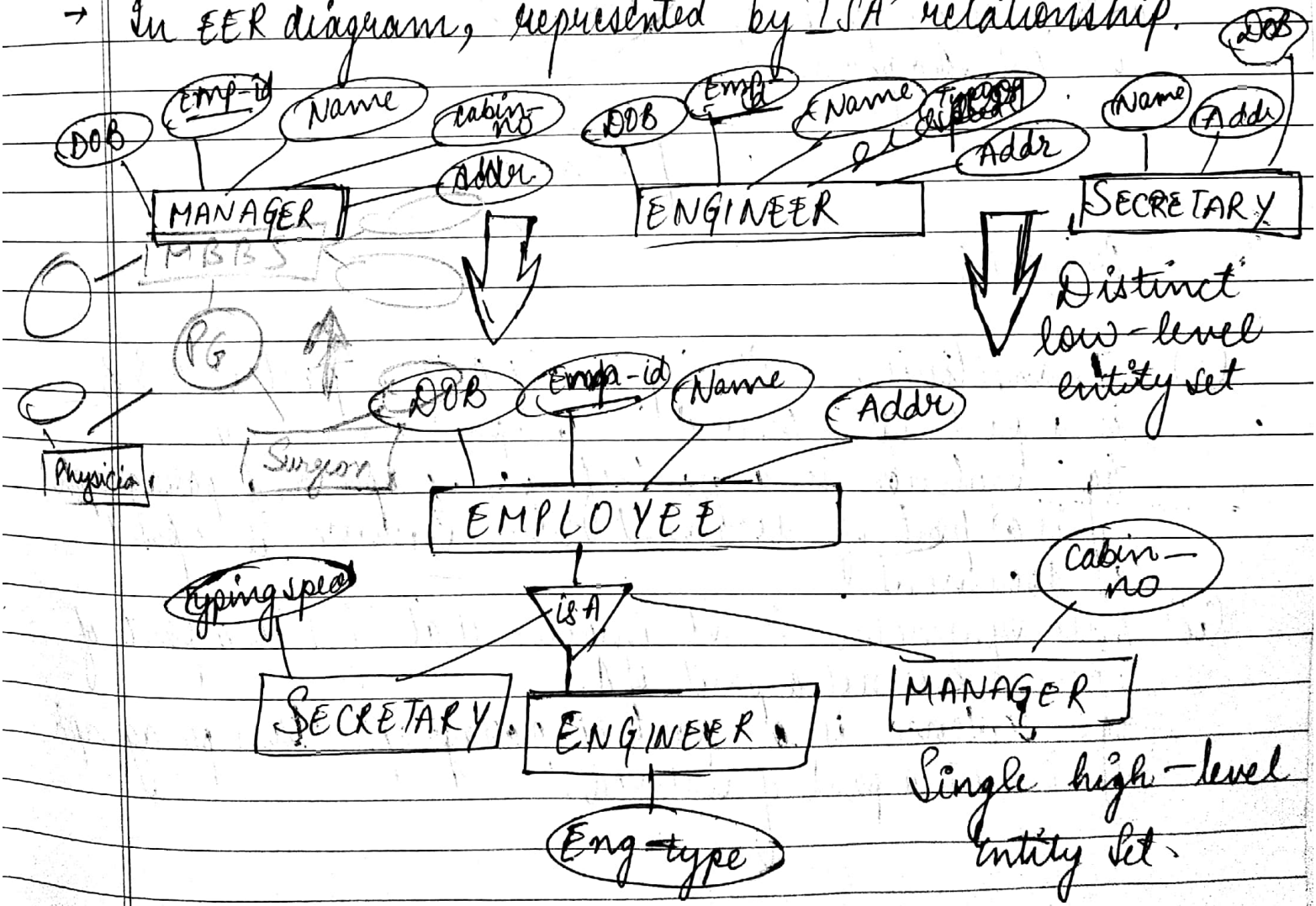
Eg:-



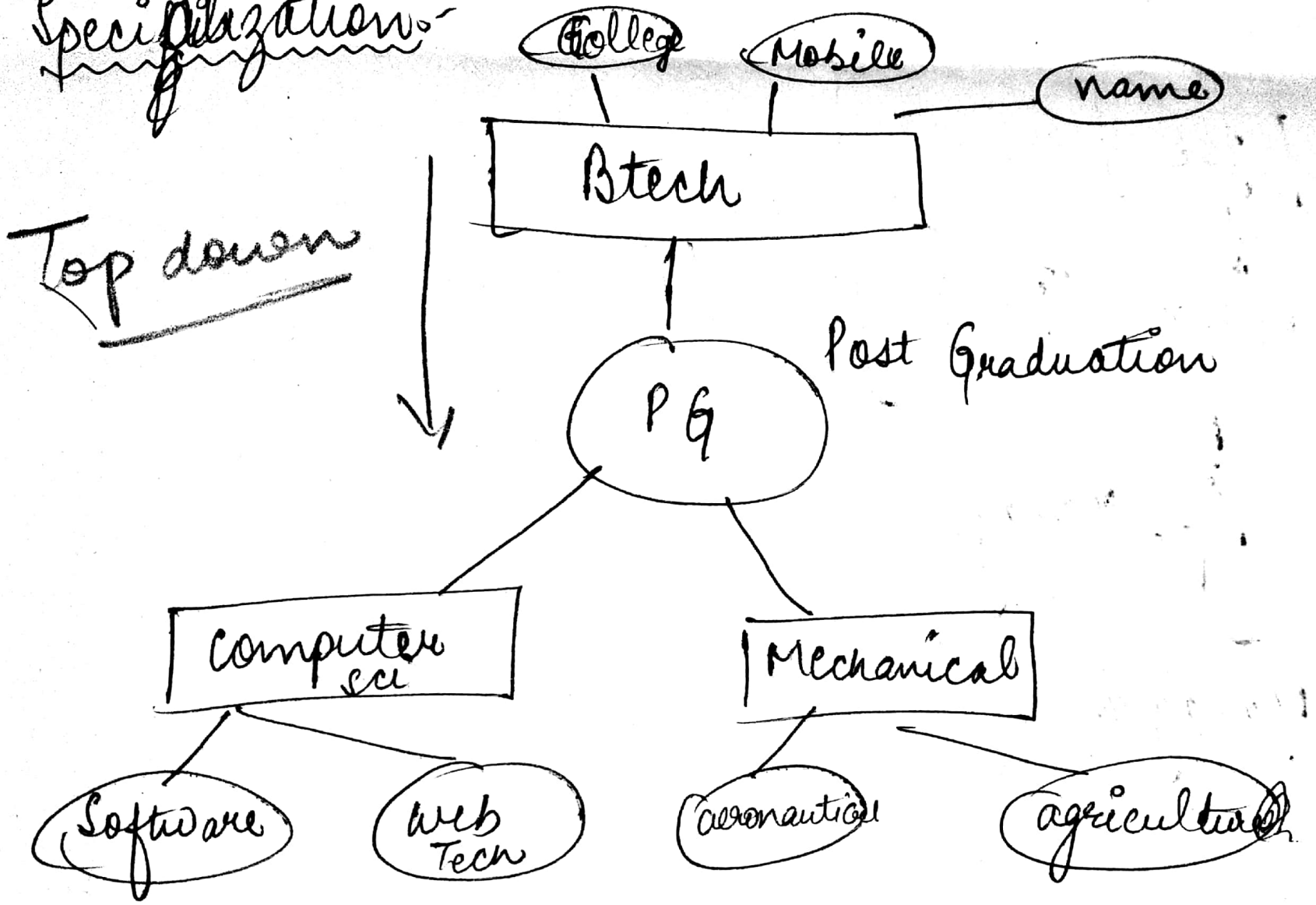


→ EMPLOYEE - Super class entity set
→ SECRETARY, ENGINEER, MANAGER → Sub class Entity set
Their unique attribute is called "local attributes".

2.) Generalization :- (used to create a general class)
→ It is the reverse of specialization
→ When lower level entity sets share the same attributes we generate single higher level entity set based on these common attributes & relationships.
→ It is a bottom up approach.
→ It is exists only when the distinct low level entity sets have some common attributes and same relationship types.
→ Shared attributes are not repeated
→ In EER diagram, represented by 'ISA' relationship.

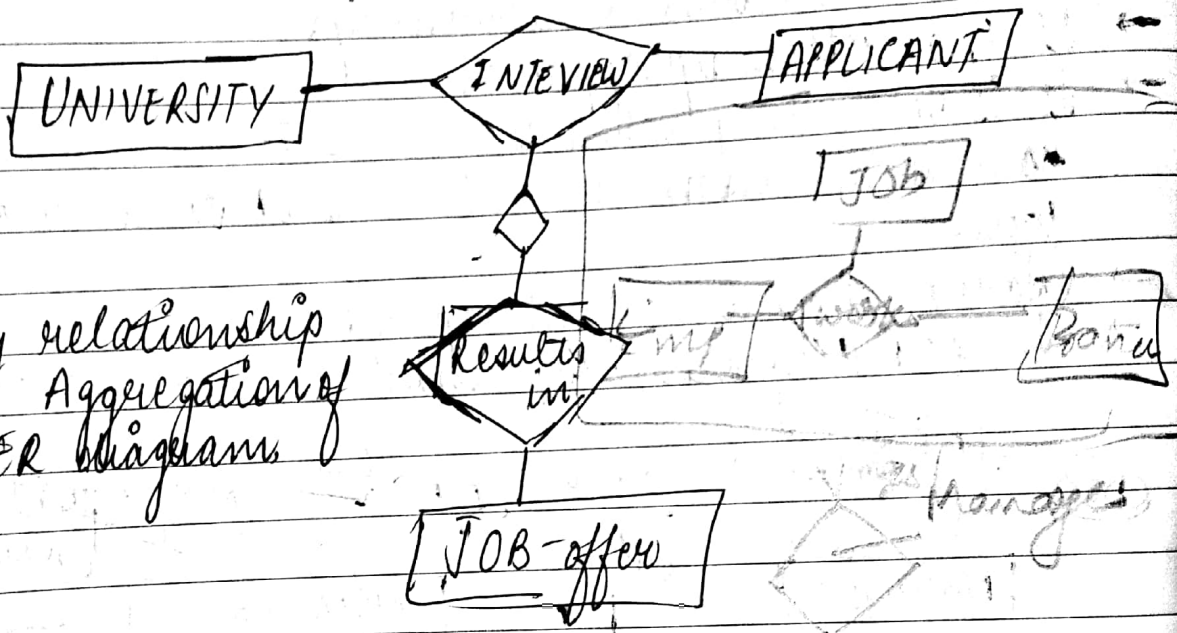


Specialization:-



(3) Aggregation :- One of the main limitation of ER model is that it cannot express relationships among relationships. To represent relationship among relationship, we combine the entity sets & their relationship to form a higher level entity set. This is called Aggregation.

> In EER diagram, the aggregation is represented by a small diamond



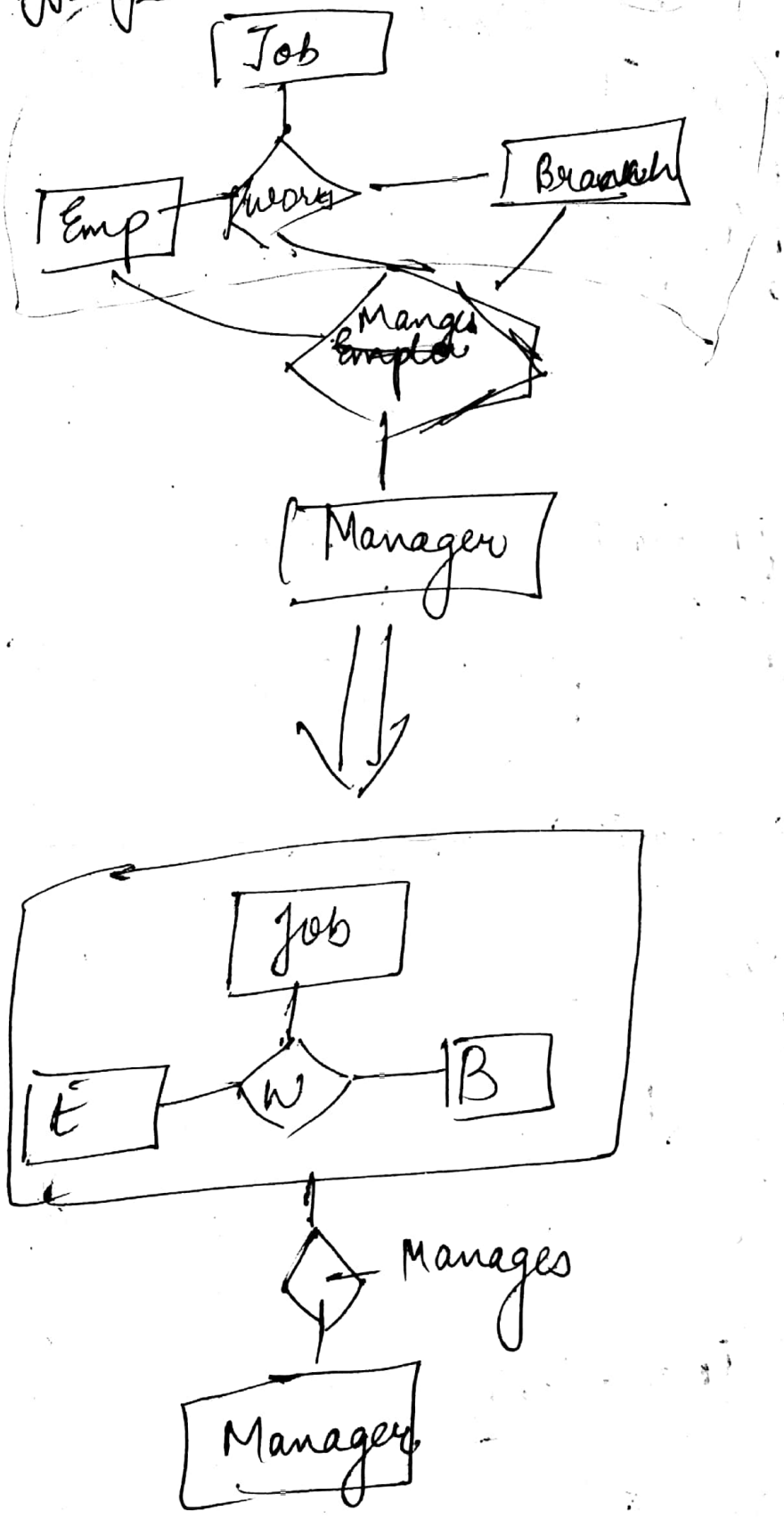
Ternary relationship after Aggregation of ER diagram

Properties :-

(i) Transitivity :- It implies if A is a part of B & B is a part of C then A is a part of C.
eg :- Sentence is a part of paragraph and paragraph is a part of document then sentence is a part of document.

(ii) Anti Symmetric :- It implies that if A is a part of B then B is not a part of A.
eg :- Sentence is a part of paragraph then paragraph is not a part of sentence.

Aggregation :-



RDBMS

RELATIONAL DATABASE MANAGEMENT SYSTEM

Relational data structure :- Uniform data structure type is called relation

Relational data Integrity rules :- A set of rules that guard to consistency of any database at every moment in time.

Relational data Manipulation :- A set of operators that transform relations into some other relations.

Relational Data structure :-

1) Relation :- Everything in a relational database is stored in the form of a relations. A relation is a set of tuples that has same number and type of attributes.

Properties :-

(i) Each relations has a unique name.
(ii) A relation doesnot contain duplicate values
(iii) A tuples of relation has no specific order
(iv) Order of attributes in relation is irrelevant
(v) Each cell in relation contains exactly one value.

2) Attributes :- An attribute of a relation is a characteristic of the relation that helps in interpreting meaning of data items in each tuple (row)

Properties :-
i) Every attribute of relation must be a name
ii) No two attributes of same table can have same name
iii) Null values are permitted for the attributes.
iv) Attribute that uniquely identify each tuple of a relation is referred as primary key.
v) Constraints can be applied on values of attributes

3) Tuple :- The row of data items in a relation corresponds to a record and is known as the tuple of a relation.
Properties :-
(i) Order of tuples are irrelevant
(ii) All tuples of relation have same format & same number of entries.
(iii) The attributes that can uniquely identify each tuple is known as the primary key

4) Data item :- The smallest unit of data in the relation is the individual data item.
cell :- Intersection of rows and columns.
Properties :-
(i) Data items are atomic.
(ii) All data items for an attribute should be drawn from the same domain.
(iii) All data items in an attribute should be of the same type.

5) Degree :- The total number of columns that comprises a relation is known as degree.
Unary relation :- Relation with 1 attribute
Ternary relation :- Relation with 3 attributes

(6) Cardinality :- Total number of tuples at any one time.

(7) Domain :- It refers to the possible values each attribute can contain.

Properties :-

(8) Keys :-

Unique Identification :- No duplication of tuples.

Irreducable :- The attributes which forms the key cannot be further decomposed

Types of keys :-

1.) Candidate key, Primary key & Super key.

Eg :-

Properties of keys :-

(i) Candidate key :-

→ It cannot contain null value

→ A table must have at least 1 candidate key

→ All attributes of a relation can acts as an candidate key

→ Uniqueness and irreducibility.

(ii) Properties of Super key :-

→ Every relation should have atleast one super key

→ A candidate key is a subset of super key

→ A candidate key is irreducible super key

→ All attributes of a table taken together form a super key

(iii) Primary key :-

→ A relation can contain only one primary key

→ A primary key is the minimum super key

→ It cannot have duplicate values

→ The values should not be null.

(iv) Foreign key :-
→ Attributes have same domain as the set of attributes defined for the primary key of another table.
→ Data values of a relation can be either null or must match the primary key.
→ There may exists more than one foreign key in a table.
→ Primary key and foreign key can exists in the same relation (table)
→ A foreign key is not necessarily the primary key or part of the primary key in the relation.

(v) Composite key :-
→ A composite key cannot contain a null value
→ It is irreducible
→ It is part of the candidate key

(vi) Artificial key :-
→ Data values are numbered in serial order
→ Used to identify uniquely relations.

Concept of Null :-
Null represents a value of an attribute that may correspond to missing information.

Difference between RDBMS and DBMS :-

	RDBMS	DBMS
1.	It stands for relational database Management System.	It stands for database Management System.
2.	It can maintain many users at a time	It cannot maintain many users
3.	It stores data in the form of relation or tables	It stores data in format of trees, graphs and also in tables.
4.	It ~~dove~~ supports Objects	It doesnot support objects
5.	It is Client/Server based System.	It is not based on Client/Server system.

RDBMS :-

RELATIONAL ALGEBRA AND RELATIONAL CALCULUS

Basic set oriented operations :-

(1) UNION (∪) :- It contains rows from both relations and remove duplicates.

Eg :-

P

ID	Name
501	Amit
503	Anand
504	Kapil

Q

ID	Name
503	Anand
506	Amitabh
510	Madhav

R = P ∪ Q

ID	Name
501	Amit
503	Anand
504	Kapil
506	Amitabh
510	Madhav

Properties :-

(i) Commutativity :- It means that result of (P ∪ Q) is same as that of (Q ∪ P)

(ii) Associativity :- It means that P ∪ (Q ∪ S) = (P ∪ Q) ∪ S

(2) INTERSECTION ($\cap$) :- It contains rows that are common to both relations.

$R = P \cap Q$ =

ID	Name
503	Anand
~~503~~	

Properties :-
(i) Commutativity :- Result of $P \cap Q$ is same as that of $Q \cap P$
(ii) Associativity :- $P \cap (Q \cap S) = (P \cap Q) \cap S$.

(3) Difference (−) :- The difference of two relations results in a new relation that contains those tuples that occur in the first relation but not in second

$R = P - Q$

ID	Name
501	Amit
504	Kapil

Properties :-
(i) Input relations must contain union
(ii) $P - Q \neq Q - P$ (Non-commutative)
(iii) $P - (Q - S) \neq (P - Q) - S$ (non associative).

(4.) Cartesian Product (X) :- Combination of two tuples.

$R = P \times Q$ =

ID	Names
501	Amit
503	Anand
503	Anand
504	Kapil
506	Amitabh
510	Madhav

Properties :-

(i) The relations need not to be union necessarily.
(ii) The result table may hold duplicate values.
(iii) The degree of resultant is equal to sum of the degrees of all the relations.
Degree of R = deg(P) + deg(Q)
(iv) The total no. of tuples is equal to product of tuples in both relations (tables).
(v) Commutativity
(vi) Associativity

~~Special relational operations :-~~

Joins :-

A join operation combines two or more relations to form a new relation on the basis of atleast one common attribute present in participating relations.

Properties :-

(i) Join operations are commutative.
(ii) Rows having null joins attributes, don't appear in the resultants relation.
(iii) More than two tables can be joined and it increases complexity.
(iv) Joins are most commonly used when there exists a relationship b/w relations (tables)
(v) These can be combined with cross-product to

Inner join / Natural join :- It produces the same resulting relation as that on performing equi join with a difference that instead of two identical attributes.

Aggregate functions :-
→ Performed on a collection of values on the database and returns only a single value.
→ It includes
COUNT, AVERAGE, MINIMUM, MAXIMUM, SUM.

Natural/Inner join :- The resultant relation contains combinations of all the tuples in both the relations.

Outer join :- When matching and non-matching tuples appear in resultant relation.

Types of Outer join :
Left Outer Right Outer full outer.

Left Outer :- It results in a relation that contains all the tuples from both the relations satisfying the join condition. Denoted by ⟕.

Right Outer :- It results in a relation that contains all the tuples in the right relation not satisfying the join condition are also concatenated with tuples. Denoted by ⟖.

Queries :-

EMP

EMP_ID	Name	Salary	Dept_ID
101	Anurag	12000	01
103	Aushul	1000	01
106	Ankur	15000	02
107	Manav	8000	03

DEPT

Dept_NO	Dname
01	Finance
02	Marketing
03	Packing

PROJECT

Proj_NO	Pname	DPno
P1	Database	01
P2	Banking	02
P3	OS	03
P4	Website	01

Query 1:- Find the name and salary of all the employees.
→ Select Name, Salary from Emp;

(2.) To find the name, Employee Id, salary of all employees who work for 'finance' department
→ Select name, Emp id, salary from Emp, Dept where Dept.Dname = "Finance";

(3.) Find the project number and controlling department name and employees working in that project.
→ ~~Select proj_no, Dname from project, dept where~~
→ Select proj_no, Dname, Ename from project, Dept, Emp where proj.Depno = Emp.Depno;

(4) Find the details of employees working on the "website" project.

→ Select * from emp, ~~Dept~~, project where project.pname = "website" ~~and emp.Empid = Dept.Deptno and project. Deptno = Dept.Deptno~~;

(5) Obtain the employee names of all the employees having salary between 10000 and 20000.

→ Select ename from emp where salary between 10000 and 20000;

(6) Retrieve the employee names who either work in department number 01 or are assigned 'OS' project.

→ Select ename from emp, Dept, project where deptno = '01' or pname = 'OS';

(7) Retrieve the employee names who work on all the projects controlled by department number 10.

→ Select ename from emp, project where ~~Dept~~ emp.deptId = project.deptno and project.deptno = '10';

(8) Find the name of all the departments who have not been assigned project.

→ Select dname from dept, project where Dept.Deptno != project.Deptno;

Queries in tuple calculus:-

(1.) Get Ename, Emp_Id of the employees who have been assigned project whose project_Id = 'P101'.

→ Select ename, Emp_Id from emp where project Id = 'P101';

(2.) Get list of employees working on either 'P101' or 'P102' projects.

→ Select ename from emp where project_id = 'P101' or project_id = 'P102';

(3.) Get the list of employees names working on both the projects 'P101' and 'P102'.

→ Select ename from emp where project_Id = 'P101' and Project_id = 'P102';

(4.) Get the names of the employees who don't work on project 'P503'.

→ Select ename from emp where projectid != 'P503';

(5.)

RDBMS

NORMALIZATION

A large database defined as a single relation may result in duplication of data. This repetition of data results in

(i) Making relations (tables) very large
(ii) Difficulty in maintaining and updating data as it would involved searching of many records in the relation
(iii) Wastage and poor utilization of disk space and resources
(iv) Likelihood of errors and inconsistencies increases.

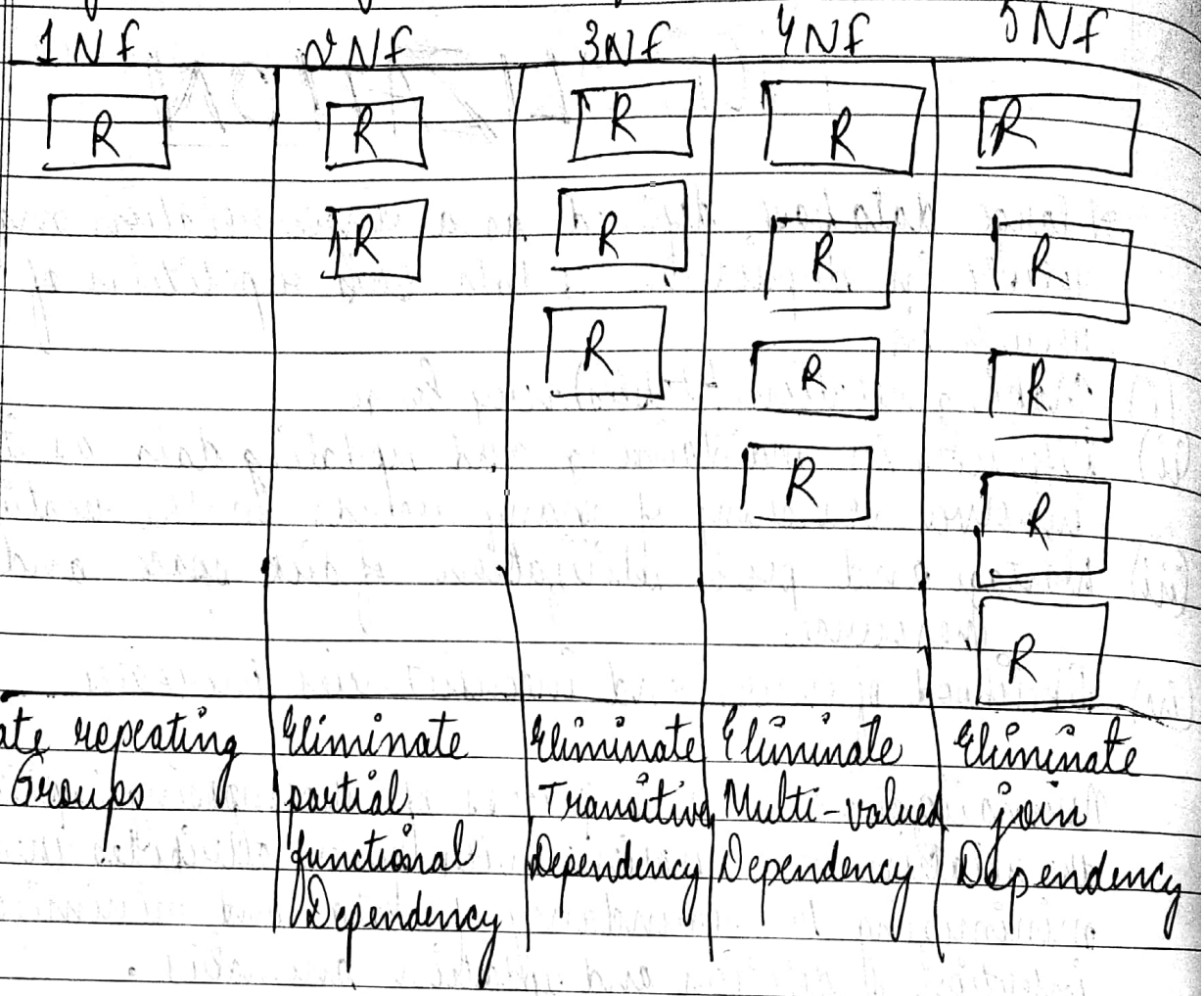
Normalization :- It is process of decomposing (splitting) the relation into relations with fewer attributes there by minimizing the redundancy of data and minimizing insertion, deletion and updation anomalies.
Normalization can be defined as step by step reversible process ~~all desirable properties~~ of transforming and unnormalized relation into relations with progressively simpler structures. No information is lost in this process.

Normal forms :- The series of stages on which normalization works is called normal forms.

Types of normal forms :- 1NF, 2NF, 3NF, 4NF, 5NF.
Where NF stands for normal forms.

The first of three normal forms ie 1NF, 2NF, 3NF were proposed by Dr. E.F Codd. Later on, a stronger definition of 3NF known as BCNF (Boyce-Codd Normal form)

Diagram showing Normal forms :-



Need of Normalization :-

→ Normalization consists of series of guidelines that help to guide you in creating a good database structure

→ While working with relational database systems, it was discovered that unnormalized relations presented certain problems whenever updations were made in them, these problems are called anomalies.

→ To create a formal framework for analyzing relation schemas based on their keys and on the functional dependencies among their attributes.

→ To obtain powerful relational retrieval algorithms based on a collection of primitive relational

operators.
→ To free relations from undesirable insertion, update and deletion anomalies
→ To reduce the need for restructuring the relations as new data types are introduced.

Data Modification anomaly :-
(i) Insert Anomaly
(ii) Deletion Anomaly
(iii) Updation Anomaly

Insert anomaly :- When one cannot insert a new row (tuple) into a table (Relation)
Delete anomaly :- When the deletion of data results in unintended loss of some other important data
Updation Anomaly :- It occurs when a single update of a single data value requires multiple rows of data to be updated.

Advantages of Normalization :-
→ Minimizes data redundancy
→ Greater overall database organization
→ Data consistency within a database
→ Much more flexible database design
→ Enforces concept of referential integrity

Disadvantages :-
→ Database can't build before knowing the needs of user.
→ Higher the normal form of a relation i.e 4NF, 5NF, lower the performance
→ Normalizing relations of higher degree leads to more consumption of time and difficult process

→ Careless decomposition leads to bad design of database.

First Normal Form (1NF)

Rules :-

(i) No duplicate rows ie all primary key attributes are defined

(ii) No repeating groups in the relation. ie. each row/column intersection contains one and only one value, not a set of values.

Eg :-

St_Id	St_Name	RECORD	
		Subject	Marks
2407	Ranjit	Computer	72
		Economics	65
		Maths	79
2408	Anurag	English	63
		Punjabi	75
		Economics	89

Primary key

Unnormalized relation

(iii) All primary key attributes are defined

Ways to achieve 1st Normal form :-

(i) Method 1 :-

Remove repeating values from ~~column~~ table and converted to a flat relation. Student 1 by repeating the pair (St_Id, St_Name) for every entry and remove attribute record with subordinate attributes Subject_Rec and Marks_Rec.

Stu ID → St_Name →
→ Subject
→ Marks

Student_table

St_Id	Std_Name	Subject_Ref	Marks
2407	Ranjit	Computer	72
2407	Ranjit	Eco	65
2407	Ranjit	maths	79
2408	Anurag	English	63
2408	Anurag	PHY	75
2408	Anurag	Eco	89

Disadvantage :- Wastage of space to keep repeated values.

(2) Student table → decomposed into 2 tables Stu_Details & Stu_PERF

Stu_DETAILS

Std_Id	St_name
2407	Ranjit
2408	Anurag

Stu_Perf

Std_Id	Subj	Marks
2407	Comp	72
2407	Eco	65
2407	Maths	79
2408	English	63
2408	Phy	75
2408	Eco	89

idea of decomposing :- keep different types of info into their seperate relations as INF disallows multivalued attribute that are composite in nature.

→ Relation is decomposed acc. to foll rules :-

(i) One relation consists of the P key of the original table & non-repeating attributes of the original table

(ii) The other relation consists of copy of the P key of original table & all the repeating attributes of the original table.

For a relation to be INF, each set of repeating groups

should appear in its own table and every table should have a 1° key.

Functional Dependency :-
Basis for first 3NF's

Functional dependencies are the consequence of the interrelationships among attributes of a relation represented by some link or association.

An attribute Y of a relation R is said to be functionally dependent on attribute X of a relation R iff for each value of x in R has associated with it only one value of Y in R at any given time represented by $X \rightarrow Y$.

X ⟶ Y
X : determinant
Y : determined

Determinant is an attribute in a relation which can uniquely determines the value and other attributes.

example :- Student table

Primary key	Roll no	S_name	course	S_phone	city
	2405	Jaskaran	MCA	2225070	Amritsar
	3162	Harpreet	BSc	2155517	Ludhiana
	2789	Ravneet	B Tech	2150707	Jalandhar
	4162	Jaspreet	M.Tech.	3190009	Amritsar

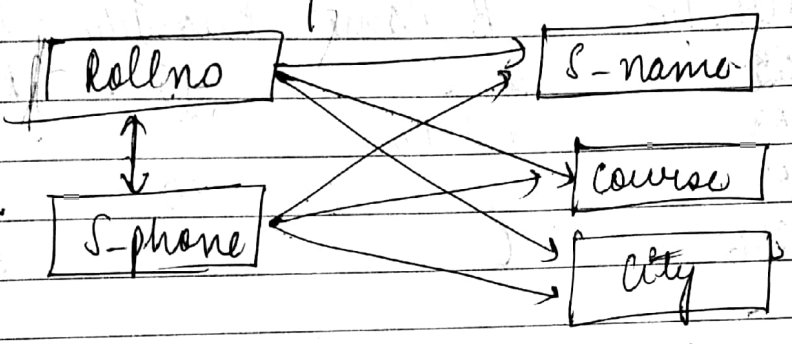
In this table, Roll no → primary key

S_name, Course, S_phone, city are functionally dependendent are 1° key Roll_no because corresponding to each stude rollno then exists one value of each attribute.

Roll no → S_name
roll no → course
but converse is not true
ie. St_name ↛ course

Roll no → S_phone.
Rollno → city.

FD's can be represented as :-



Dependencies :-

Rollno → S_name
Roll no → city
Rollno → course
Rollno → S_phone

S-phone → S-name.
S-phone → Rollno
S_phone → course
S-phone → city

Features

→ If an attribute acts as a 1° key then all the col's in the relation must be functionally dependent on 1°key attribute.
→ If X → Y holds in relation R then Y → X may not or may hold.

Fully Functional Dependency :-

→ applies to table with composite keys

→ Attribute Y in relation R is fully functionally dependent on an attribute X of relation R if it is functionally dependent on X and not on any subset of X

It means when 1°key is composite then other non-key attributes of the relation must be identified by the entire key and not by some of the attributes that make up the key.

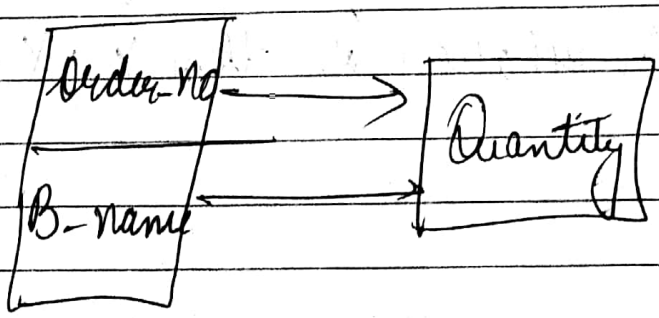
Eg :- ORDER-BOOK

Order_No	B_Name	Quantity	B_price
4252	C	15	Rs 175
4253	Database	20	Rs 225
4154	IT	30	Rs 200
4256	C	50	Rs 175

Here 1° key is composed of 2 attributes (Order_no, B_name) & Quantity is FD on 1°key & not any subset of it (i.e Order_no alone or B_name alone)

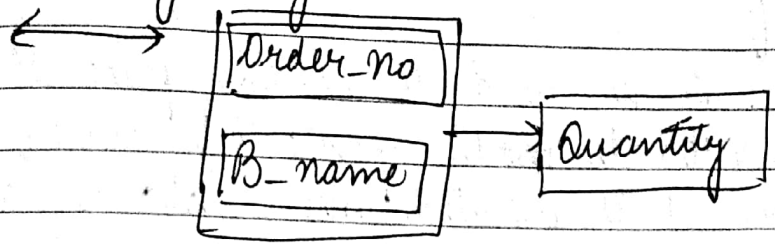
Quantity is not FD on either of two because corresponding to each order_no or B_name there are multiple values of for the Quantity attribute.

(Order_no, B_name) → Quantity



But price is not ffd on 1°key because B_price is FD on B_name

dependency diagram



A	B
1	2
3	4
2	4
2	5

3 → 4
2 → 4

But 2 < 5

Incorrect

1 NF was concerned with the structure of representing relations 2NF is concerned with eliminating redundancy in these relations.

Anomalies in 1NF :-

→ Insertion :-

consider

ORDER_Book relation

Order_No	B_name	Quantity	price
4252	C	15	175
4253	Database	30	925
4154	IT	20	200
4256	C	15	175

We can't insert a book until some order is placed for it coz in relation 1°key is composed of 2 attributes Order_no & B_name. So, acc. to principle of entity Integrity rule Order_No & B_name can't have null values. This occurs Insertion anomaly

Order No	B_name	Quantity	B_price
Null	photoshop	Null	120

Invalid

2. Deletion :- If we want to delete order 4154 then whole info of that book will be deleted
loss of data

3. Updation :- If we need to modify price of any book then we need to ~~modi~~ update its price at every point where it is inserted when data is large

Second Normal form :-
A relation is in 2NF iff both conditions hold simultaneously
→ Relation should be in 1NF
→ Every non key attribute should by FFD on 1°key

if the relation in 1NF consists of only one attribute in 1°key then the relation is said to be 2NF.

Consider ORDER_BOOK relation
1°Key → (Order_No, B_name), non-key att (Quantity & B_price)
Here we have
(Order_No, B_name) → Quantity
B_name → Bprice
(Order_No, Bname) -/→ B_price

So to be in 2NF we have to eliminate partial key dependency on the 1°key. The reduction of 1NF to 2NF consists of splitting 1NF relations into appropriate relations such that every non-key attribute are FFD on 1°key of resp. relation

FFD :- (Order_No, B_name) → Quantity
FD :- (B_name) → B_price

ORDER-BOOK

Order_No	B_Name	Quantity	B_price
4253	C	15	175
4253	Database	20	225
4154	IT	30	200
4256	C	50	175

Order

Order No	B_name	Quantity
4253	C	15
4253	Database	20
4254	IT	30
4256	C	50

Book-Info

B_name	B_price
C	175
Database	225
IT	200

Use of 2NF to remove anomalies of 1NF
→ Insertion anomaly
In 1NF insertion of book wasn't possible without order no. But in 2NF we can insert a book without order_no.

B_name	B_price
C	175
Database	225
IT	200
photoshop	120

'Book_name' acts as a 1°key in the Book_info relation and not the composite key 'B_name' & 'Order_No'.

→ Deletion anomaly :- Deletion is easy coze of seperate orders.

→ Updation anomaly :- performing an updation in a relation 1NF is easier than in INF as we have seperate relations for various attributes.

Anomalies in 1NF

STUDENT relation

*Key Stu_Id	Stu_Name	Teach_Id	Teach_Name	Teach_Qual
2523	Anurag	T001	Navathe	PHD
3712	Pankaj	T004	Date	MTech
4096	Anshu	T001	Navathe	PHD
2716	Harman	T004	Date	MTech
1768	Gagan	T009	Desai	M.Tech

→ Insertion anomaly :- Insertion of teacher info without student info is not possible as here Stu_Id is 1° key and having NULL value in it is against the principle of EI Rule.

→ Deletion anomaly :- If any student data is deleted the data of teacher is also deleted along with it, loss of data.

→ Updation :- Updating any teacher qualification needs to be updated at every record. So in case of huge database it will be big problem and leads to inconsistency of data.

CONCLUDE:- In 3NF, we have data manipulation properties or, Hence bringing a relation to 3NF wouldn't terminate logical database design. Further transformation are required to diminate these anomalies.

Transitive Dependency / Indirect dependency

Suppose relation R exists with attributes A, B, C. A is a key and b, c are non-key attributes.
Lets assume the foll dependency holds
$A \to B$ but $B \not\to A$ (B is FD on A)
$B \to C$
then C is transitive dependent on A, $A \to C$

$$\left.\begin{array}{l} A \to B \\ B \to C \end{array}\right\} \Rightarrow A \to C$$

Consider STUDENT relation
Here, the foll FD's occur
Stu_Id → Stu_Name
Teach_Id → Teach_Name
Stu_Id → Teach_Id
Teach_Id → Teach_Qual

Stu_Id → Teach_Id
Teach_Id → Teach_Name
Teach_Id → Teach_Qual

Stu_Id → Teach_Name , Stu_Id → Teach_Qual

Third Normal Form :- 3NF

A relation is said to be 3NF if and only if holds :-

(i) Relation is in 2NF

(ii) Non-key attribute should not be transitively functionally dependent on 1° key

A relation is in 3NF if it is in 2NF and every non-key attribute is directly dependent on 1° key

Main purpose of 3NF is to remove the transitive dependency which is the main cause of anomalies in 2NF.

STUDENT

Stu_Id	Stu_Name	Teach_Id	Teach_Name	Teach_Qual
9523	Harpreet	T001	Shweta	PHD
3712	Jaskaran	T004	Rajni	M Tech
4096	Jaspreet	T001	Shweta	PHD
9716	Ravneet	T004	Rajni	M Tech
1768	Manmeet	T009	Desai	M.Tech

Source table

→ Create a new table from the above, which contains all the original attributes but without those attributes that are transitively dependent on the primary key

→ Other one display those attributes which are dependent on primary key.

Example :- In STUDENT relation, the non-key attributes Teach_Name &

Teach_Qual are dependent on the primary key StdId

STU_TEACH

Stu_Id	Stu_Name	Teach_Id
2523	Harpreet	T001
3712	Jaskaran	T004
4096	Jaspreet	T001
2716	Parneet	T004
1768	Manmeet	T009

Teacher

Teach_Id	Teach_Name	Qual
T001	Shweta	PHD
T004	Rajni	M Tech
T009	Debai	M Tech

Anomalies

→ ~~Logic~~ Insertion :- Insertion of info of teacher is possible even if student info is not assigned to that teacher.

→ Deletion :- If student info is deleted and then there will be no loss to the info of teacher.

→ Updation :- Updating the teacher record will be easy if the teacher's info is maintained in second table seperately.

~~Boyce~~ - Codd

Boyce - Codd Normal Form (BCNF)

Any relation can be BCNF if it doesn't have:
(i) Multiple candidate keys
(ii) Where multiple candidate keys are composite.
(iii) Multiple candidate keys are overlapped i.e. for more candidate keys share common attribute.

Determinent: An attribute on which some of the attributes are FD.
So relation can only be in BCNF iff. every determinent is a candidate key.

Eg:- Consider:

Stu-Id	Sname	Course	Grade
5705	Rajesh	Computer	A
5901	Kapil	computer	C
5658	Sunil	DSA	A
5705	Rajesh	French	A
5816	Ankit	Dance	B
5658	Sunil	Yoga	B

Primary key

Candidate keys:- Sname, Course

(Sname, Course) → Grade
Sname → Stu_Id
(Stu-Id, Course) → Grade.
Stu_Id → SName

Grade is not in 3NF coz no transitive dependency but not in BCNF because

(a) multiple candidate keys (Sname, course) & (Stu Id, course)
(b) candidate keys consists of more than 1 attribute (S_Name, Course) & (Stu Id, course)
(c) Both candidate keys share a common attribute 'Course'

Anomalies

1) Insertion :- Student info cannot be ~~regi~~ entered if he/she has not registered any course

2) Updation :- Sname, Stu Id are repeated Multiple times so any change in 1 will update all

3) Deletion :- If a student with drawes from all courses, the info of student is completely lost.

GRADE-IT

Sname	Stu Id

Course.

Stu Id	Course	Grade

Comparison B/w 3NF & BCNF

→ A table in BCNF is also in 3NF but converse is not true
→ During updation loss of data can be there in BCNF
→ BCNF is stronger definition to 3NF and has no link with 1NF & 2NF
→ In BCNF the relation has multiple composite candidate keys and two or more candidate keys share a common attribute.
⇒ A relation can be normalised both with 3NF and BCNF if relation consists of only single candidate key

Multivalued dependency :-

It disallowes an attribute in a tuple to have set of values. In case of ~~some~~ functional dependency in a relation $A \rightarrow B$ relates one value of A to one value of B while in multi-valued dependency represented as $A \twoheadrightarrow B$ defines a relationship in which set of values in B determines a single value in A.

Eg:- Table Emp

Ename	Child	Salary	Grade
Ranjit	sunny	25000	2001
Ranjit	sunny	30000	2001
Pankaj	Akshay	15000	2002
Pankaj	Akshay	25000	2003
Pankaj	Rupa	15000	2002
Pankaj	Rupa	25000	2003

Child is multivalued of Ename. (multiple values of child corresponding to empname).

Ename $\twoheadrightarrow$ Child

$Child_{Ranjit\ 25000,2001} = Child_{Ranjit,30000,2001} = \{sunny\}$

If we delete a record of {Pankaj, Akshay, 25000 2003} then it resulted in failure of multivalued dependency coz.

$Child_{Akshay} \neq Child_{Roopa}$

And left side results in both akshay & roopa but right only results in {Rupa}

Therefore Ename →→ child does not hold.

Properties :-

→ In a relation to maintain a multivalued dependency, it must have atleast three attributes ie A →→ B holds iff A →→ C also holds in R(A, B, C)

→ The attributes giving rise to the multivalued facts must be independent of each other

Fourth Normal Form (4NF)

A relation R is in 4NF if it is in BCNF (or 3NF) and it contains no multivalued dependencies.

Eg :- EMPLOYEE

Empname	Equipment	Language
Anurag	PC	English
Anurag	PC	French
Anurag	Mainframe	English
Anurag	Mainframe	French
Kapil	PC	English
Kapil	PC	French
Kapil	PC	Japanese

Empname	Equip
Anurag	PC
Anurag	Mainframe
Kapil	PC

Empname	Language

Join Dependency

Join dependency states that after a relation has been decomposed into multiple smaller relations, it must be capable of being joined again on common keys to form the original relation.

Eg:-

R1

Factory	Component
GM	Engine
GM	Gear box
Honda	Engine

R2

Component	Project
Engine	MPC
Gearbox	125A
Engine	125A

R4

Factory	Component	Project
GM	Engine	MPC
GM	Engine	125A
~~Honda~~ GM	Gearbox	125A
Honda	Engine	MPC
Honda	Engine	125A

R3

Factory	Project
GM	MPC
GM	125A
Honda	125A

Factory	Component	Project
GM	Engine	MPC
GM	Gearbox	125A
Honda	Engine	125A
GM	Engine	125A

Fifth Normal Form (5NF) :-

Also known as project join normal form

Projection of join dependency conditions :-

→ R should be in 4NF

→ It cannot be further non-loss decomposed.

Agent	Company	Product
Suneet	ABC	Nut
Suneet	ABC	Screw
Suneet	CDE	Bolt
Raj	ABC	Bolt

P1

Agent	Company
Suneet	ABC
Suneet	CDE
Raj	ABC

P2

Agent	Product name
Suneet	Nut
Suneet	Bolt
Suneet	Screw
Raj	Bolt

3NF

Stu#	Stuname	Course	sub#	Subname	Teacher	Room	Result
1507	Pankaj	comp	100	CS1	Anurag	10	Pass
			200	DBSW	Kapil	16	Comp
			300	SP	Anurag	10	Pass
2101	Shweta	Acct	100	CS1	Anurag	10	Pass
			400	CS2	Anurag	10	Pass
3562	Ranjit	comp	450	CO	Kapil	16	Comp
			300	SP	Anurag	10	Pass
			325	BF	Sumit	16	Pass
1942	Jagan	comp	400	CS2	Anurag	10	Fail
			450	CO	Kapil	16	Fail
			300	SP	Anurag	10	Pass
			200	DBSW	Kapil	16	Pass

Results (in 1NF)

Stu#	StuName	course	sub#	subname	Teacher	Room	Result
1507	Pankaj	comp	100	CS1	Anurag	10	Pass
1507	Pankaj	comp	200	DBSW	Kapil	16	Comp
1507	Pankaj	comp	300	SP	Anurag	10	Pass
2101	Shweta	Acct	100	CS1	Anurag	10	Pass
2101	Shweta	Acct	400	CS2	Anurag	10	Pass
3562	Ranjit	comp	450	CO	Kapil	16	Comp
3562	Ranjit	comp	300	SP	Anurag	10	Pass
3562	Ranjit	comp	325	BF	Sumit	16	Pass
1942	Jagan	comp	400	CS2	Anurag	10	Fail
1942	Jagan	comp	450	CO	Kapil	16	Fail
1942	Jagan	comp	300	SP	Anurag	10	Pass
1942	Jagan	comp	200	DBSW	Kapil	16	Pass

Res — Studen — Subject

Res :

Stu#	Sub#	Result
1507	100	Pass
1507	200	comp
1507	300	Pass
0101	100	Pass
0101	400	Pass
3562	450	comp
3562	300	Pass
3562	325	Pass
1942	400	Fail
1942	450	Fail
1942	300	Pass
1942	200	Pass

Student

Stu#	Stname	course
1507	Pankaj	comp
0101	Shweta	Acct
3562	Ranjit	comp
1942	Jagan	comp

Subject

Sub#	Subname	Teacher	Room
100	CS1	Anurag	10
200	OBSW	Kapil	16
300	SP	Anurag	10
400	CS2	Anurag	10
450	CO	Kapil	16
325	BF	Sumit	10

Sub

Sub#	Subname	Teacher
100	CS1	Anurag
200	OBSW	Kapil
300	SP	Anurag
400	CS2	Anurag
450	CO	Kapil
325	BF	Sumit

Teachers

Teacher	Room
Anurag	10
Kapil	16
Sumit	38

R. DBMS

TRANSACTION MANAGEMENT AND CONCURRENCY CONTROL

Transaction :- A transaction is a collection of operations that form a single logical unit of work.

operations performed during transaction :-

(i) Adding (inserting) some data
(ii) Modifying the existing data (Updation)
(iii) Accessing existing data or a combination of these.

For any transaction to be completed, all operations in a transaction must be executed properly. If any operation fails to execute, the entire transaction is aborted.

A transaction is a logical unit that can either be entirely executed or entirely aborted.

Eg :- Banking system, telephone bills / electricity bills

Example :- Suppose you need to pay telephone bills using online bill payments.

Steps :- Switch to appropriate link of site of that company. If you want to pay ₹900, you will instruct the bank to debit ₹900 from your account and credit same to the BSNL account using various operations.

This transaction will work on following operations :-
→ Subtracting bill amount of ₹900 from bank account of customer using UPDATE operation
→ Adding bill amount in BSNL account using UPDATE function

In above case both the operations must be executed or both should not be executed. It is unacceptable that customer account is debited and BSNL account is not credited. In that case customer money can be lost.
Incomplete transaction results in inconsistent database state, so in such a case DBMS rolls back the database to a previous consistent state.

ACID properties of transaction :-

(i) Atomicity :- A transaction is said to be atomic if all the operations of transaction run to completion. Incase of any failure the transaction is aborted.
Features :-
→ Either all operation in a transaction are executed or none of them is executed.
→ Atomicity is maintained in presence of disk failure, CPU failure, database software failures etc.
→ Atomicity is maintained in case of deadlocks.
→ It can be turned off at system as well as sessional level.

(ii) Consistancy :- A transaction is said to possess consistency property if the database is in consistent state before

the start of the transaction and after the completion of transaction.

(iii) Isolation :- It is particularly useful when there are concurrent transactions (2 transactions executing simult.) The concurrent transactions are those transactions when multiple users access and update the shared database at the same time.
This property ensures that when multiple transactions are executing at the same time, the data used by one transaction cannot be used by the other transactions until the first one has completed.

(iv) Durability :- It ensures that once the transaction is commits, its effects on database cannot be changed or lost even in case of any failure, system crash etc.
It is responsible for recovery subsystem of DBMS.

Implementation of ACID properties :-

Atomicity :- Suppose before executing of transaction T, the balances in the account A and account B be ₹5000 and ₹8000 resp.
Now assume during execution a failure occurs and it prevents T from completing its execution successfully.
The failure occurs after write(A) and before execution of write(B) operation.
Thus values in A and B are ₹4250 and ₹8000 and ₹750 that are drawn from A are lost coz of failure.
Now database is in inconsistence state because of partial completion of transaction.

Consistency :- consider the same amounts in account A and B ie ₹5000 and ₹8000 resp. After the execution of our transaction the amount becomes ₹4250 and ₹8750 resp. So in both cases sum of amount in A and B is ₹13000 which satisfy consistency property.

Isolation :

Now consider two transactions T1 and T2 executing concurrently. The transactions T1, that transfers amount of ₹750 from A to B. And T2 transfers 20% of amount from A to B

Initially balances are ₹5000 and ₹8000.

Consistent state

T1	T2	Status
Read(A)		₹5000
A' := A - 750		₹4250
Write(A)		Store to A
	Read(A)	₹4250
	temp := A * 0.2	₹850
	A' := A - temp	₹3400
	Write A	Store to A
Read(B)		₹8000
B' := B + 750		₹8750
Write(B)		Store to B.
	Read(B)	₹8750
	B' := B + temp	₹9600
	Write(B)	Store to B

Inconsistent State

T1	T2	Status
Read(A)		₹5000
A' := A - 750		₹4250
	Read(A)	₹5000
	temp := A * 0.2	₹1000
	A := A - temp	₹4000
	Write(A)	Store to A = 4000
	Read(B)	₹8000
Write(A)		Store to A = 4250
Read(B)		₹8000
B' := B + 750		₹8750
Write(B)		Store to B = 8750
	B' := B + temp	₹9750
	Write(B)	Store to B = 9750

Durability :-
→ The updates carried out by the transaction has been written to the disk is before the transaction is completed.
→ These updates are sufficient for database to reconstruct after database is started after failure.

States of transaction :-
→ Active :- The transaction is said to be active when the first operation of the transaction starts executing and it remains active until all operations in a transactions are completed.
→ Partially committed :- A transaction goes in a partially committed state if all the operations of the transaction are completed successfully ~~and it enters the state to committed state where~~
→ Failed :- A transaction is said to be in failed state after the system determines that the transaction can no longer proceed with its normal execution because of hardware or logical errors.
→ Aboarted :- A transaction is said to be aboarted when it is rolled back.
→ Committed :- Once all changes are successfully made in a transaction. It is said to be committed.

Concurrency control :-
Large computer systems usually allow multiple ~~part~~ transactions to run concurrently ie, at same time. In case of large computer systems when multiple users share database, each user that has rights to access database, perform it by using transaction. So, these concurrent running transactions may interfere into one another transactions to make database

Benifits → [→ Helps in reducing waiting time
→ Improved throughput & resource utilization]

inconsistent.
One of the possible solution of this problem is that ~~transaction~~ transaction run serially. But executing transaction serially may take a lot of time (1 at a time).
A transaction consists of number of steps. Some of them may involve input/output activity and other may involve CPU activity. Both these activities are independent on each other. can be made to run parallely. This allows multiple transactions to run concurrently.
→ As a result the operations of one transaction is in progress, the other transaction may be running in CPU. Similarly, the other transactions switch between I/O operations and CPU operations. This process is similar to multiprogramming ie. as a result no. of transaction per unit time increases which results in increase in CPU utilization.
→ When multiple transactions are executing concurrently, some of them may be long transactions and some may be short transactions. If transactions are executing serially, then short transaction have to wait for long transactions to be executed, that results in unpredictable ~~results~~ delays in running a transaction.

Need of concurrency control:-
The main objective of developing a database is to allow many concurrent users to access the database. Concurrent access doesn't cause any problem as long as multiple users read from a database, the problems arrises when user try to

update a database.
There are three different ways in which users can overlap :-
→ Lost Update Problem (WW conflicts)
→ Dirty Read or temporary update problems (WR conflicts)
→ Incorrect Analysis problem or unrepeatable reads (RW conflicts)

Lost Update Problem :- (WW conflict)
When multiple transactions are running concurrently, the lost update problem occurs when data is being updated by one transaction and is being overwritten by other transaction.
Consider two transactions T1 and T2 running concurrently and both updating bank balance of account A by increasing previous balance by ₹200.
Initial Balance be ₹1000

If transactions are executing serially ie. the second transaction cannot be executed until first one is finished.

Seq no	Order	Description
1	T1: Read(A)	A = 1000
2	T1: A := A + 200	A = 1000 + 200
3	T1: Write (A)	A = 1200 Account updated
4	T2: Read(A)	A = 1200
5	T2: A := A + 200	A = 1200 + 200
6	T2: Write (A)	Account updated A = 1400

actual

The balance of A is properly update ie ₹400 (200+200) and total amount becomes ₹1400 resp.

If transactions are not executed serially, the data gets lost.

eg:-

seq no	Order	Description
01	T1: Read(A)	A = ₹1000
02	T2: Read(A)	A = ₹1000
03	T1: A: A+200	A = A+200 ⇒ A = 1000+200
04	T2: A: A+200	A = A+200 ⇒ A = 1000+200
05	T1: Write(A)	A = 1200 Account updated
06	T2: Write(A)	A = 1200 Account updated

Update is lost

When transactions are not executed serially:-
→ Various operations gets eg overlaps with each other
→ The updates have been lost.

Dirty Read problem :-
This problem occurs whenever multiple transaction run concurrently and one transaction is allowed to see the intermediate results of another before it is committed.
In other words when one transaction updates the data6 item and then the transaction aborts due to some reasons but the updated item is accessed by another transaction before it is changed back to its original value.

Consider two transactions T1 and T2. Suppose balance of account A = 1000 and increase it by 200.

Seq No	Order	Description
01	T1: Read(A)	A = 1000
02	T1: A := A+200	A = 1000+200
03	T1: Write(A)	A = 1200 account updated
04	T2: Read(A)	A = 1200
05	T1: Aborted	Transaction T1 rolls back to original value.
06	T2: A := A+200	A = 1200+200 1400
07	T2: Write(A)	Account updated A = 1400

(Roll back to Initial State.) but it should be 1200

T1 aborted, but
T2 read the data before it aborted and which is not stored in database permanently because of failure of T1. So value which is read by T2 is meaningless.

Inconsistent analysis problem :- Unrepeatable read (R-W conflict)
When multiple transactions are running concurrently the unrepeatable read problem occurs when a transaction reads an item twice and item is updated by another transaction between the two reads, so transaction recieves different values for two reads.

T1: Read (A)
T2: update (A)
T1: Read (A)

eg:

T1	T2	Desc
Read(A)		A = 1000
	R(A)	1000
	A = A+200	
	W(A)	3000
R(A)		3000

Schedulers & Locks

Schedule :- The sequence of instructions that specify the chronological order in which instructions of concurrent transactions are executed.

The actual order of execution of statements is called schedule. (The sequence of operations performed by transaction)

Consider two transactions T1 and T2 such that T1 transfers ₹100 from account A to B & T2 transfers ₹200 from account B to C. Let the initial balances in accounts be 1000, 2000, 3000 resp.

Set of operations

T1		T2	
Read (A)	1000	Read (B)	2100
A := A - 100		B := B - 200	
Write (A)	900	Write (B)	1900
Read (B)	2000	Read (C)	3000
B := B + 100		C := C + 200	
Write (B)	2100	Write (C)	3200

Initially A = 1000, B = 2000, C = 3000.

classmate
Date
Page

Possible Schedules

Schedule 1

A=1000	T1:	Read(A)
	T1:	A := A-100
A=900	T1:	Write (A)
B=2000	T1:	Read (B)
	T1:	B := B+100
B=2100	T1:	Write (B)
B=2100	T2:	Read (B)
	T2:	B := B-200
B=1900	T2:	Write (B)
C=3000	T2:	Read (C)
	T2:	C := C+200
C=3200	T2:	Write (C)

Serial Schedules

Schedule 2

A=1000	T2:	Read(A)
	T1:	A := A-100
A=900	T1:	Write (A)
B=2000	T2:	Read (B)
	T2:	B := B+200
B=1800	T2:	Write (B)
B=1800	T1:	Read (B)
	T1:	B := B+100
B=1900	T1:	Write (B)
C=3000	T2:	Read (C)
	T2:	C := C+200
C=3200	T2:	Write (C)

Schedule 3

A=1000	T1:	Read(A)
	T1:	A := A-100
B=2000	T2:	Read (B)
	T2:	B := B-200
A=900	T1:	Write (A)
B=2000	T2:	Read (B)
	T1:	B := B+100
B=1800	T1:	Write (B)
C=3000	T2:	Read (C)
	T2:	C := C+200
C=3200	T2:	Write (C)
	T2:	Write (B)
B=1900		

non Serial Schedules

Types of Schedules :-

Serial non-serial

Serial schedule :- A schedule is said to be serial if for every transaction involved in the schedule all the operations of each transaction are executed sequentially without any interference of other transactions

In A serial schedule each transaction is executed continuously and second transaction is executed after the completion of first one.

The order of execution is important in this type of schedule.

→ If some transactions are long then other transaction have to wait before complition of operations of that

transaction.

→ If transaction waits for I/O operation, CPU processor cannot switch to another transaction resulting in wastage of CPU processing time.

Non-Serial schedule:- A schedule is said to be non-serial if if the operations of two or more transactions execute concurrently the operations overlap. If the operations of transaction are not properly interleaved then they may results in problem like lost update, dirty read etc

When more than one transactions are running at a time, their operations may be in conflict with each other in schedule if they satisfy following conditions:-

(i) Operations should be of different transactions.
(ii) Operations access same data items
(iii) At least one of the operations must be a write operation.

Complete schedule:- A schedule is said to be complete schedule if it satisfies the following conditions:-

→ Operations of schedule must match operations of transaction.
→ Each transaction in a schedule consists of commit, or abort operations as the last operation.
→ For any two conflicting operations one or 2 transaction must occur
→ Order of operations in a schedule must be same as that of operations in a transactions.

Serializibility :- The concept which is used to ensure the correctness of a non-serial schedule.
A non serial schedule is known as serializable schedule if it produces same result as that of serial schedule.

Result Equivalent :- Two schedules are said to be result equivalent if they result in same final state of the database.

Eg :-

S1	S2
Read (X)	Read (X)
X := X + 20	X := X * 10
Write (X)	Write (X)

For X=100, both produces same result i.e. X = 120 but X = 200, the results for both is different ie X=2~~40~~ & X=~~220~~

Locks :- The notion of locks helps to convert a schedule into a serializable schedule. A lock can be placed by transaction on a resource that is it desires to use it anymore, it should unlock the resource so that other transactions can use it.

T1	T2
lock Bx	lock By
lock By	lock Bz
Read Bx	Read By
Bx := Bx - 50	By := By - 60
update Bx	Update By
Unlock Bx	Unlock By
Read By	Read Bz
By := By + 50	Bz := Bz + 60
update By	Update Bz
Unlock By	unlock Bz

Types of locks :-

- Binary locks
- Shared locks
- Exclusive locks.

Read/write locking

- Shared locks (read) :- This should be used by a transaction which shall perform an update operation on a resource. (S)
- Exclusive locks (write) :- This should be used by a transaction which shall perform an update operation by on a resource. No other transaction can place either a shared lock or an exclusive lock on a resource that has been acquired in an exclusive mode (X).

Consider a situation where there exists two banking accounts A & B. Initial balances be ₹1000 & ₹900 resp. consider transaction T1 which is sum of balances of A and B and T2 which transfered ₹200 from A to B

T1	T2
Lock X(A)	Load X (Sum)
Read (A)	Sum = 0
A := A - 200	Lock S(A)
Write (A)	Read A
Unlock (A)	Sum := Sum + A
Lock X (B)	Unlock (A)
Read (B)	Lock S (B)
B := B + 200	Read (B)
Write (B)	Sum := Sum + B
Unlock (B)	Write B.

Schedule :-

T2: lock (Sum)
T2: Sum = 0
T2: lock S(A)
T2: Read (A)
T2: Sum := Sum + A
T2: Unlock (A)
T1: lock X(A)
T1: Read (A)
T1: A := A - 200
T1: Write (A)
T1: Unlock (A)
T1: lock X (B)
T1: Read (B)
T1: B := B + 200
T1: Write (B)
T1: Unlock (B)
T1: lock S(B)
T1: ~~Read~~ S (B)
T2: Sum := Sum + B
T2: Write (Sum)
T2: Unlock (B)
T2: Unlock (Sum)

Binary lock :-

→ Two valued - locked & Unlocked
→ Two basic operations
(i) lock (A)
(ii) Unlock (A)
→ A transaction requests access to a data item by first locking the data item using lock ( ) operation and if any other transaction tries to access same data, it is forced to wait.
→ Once transaction is executed it automatically unlocks operations

Rules :-

(i) lock ( ) :- operation must issued by the transaction before any update operation like read ( ), write ( )
(ii) Unlock ( ) :- Must be issued by the transaction after read and write operation

Eg :-

T1	T2
Read (A)	Read (A)
A := A + 200	A := A + 300
Write (A)	Write (A)

Schedule

T1: Read (A)
T2: Read (A)
T1: A := A + 200
T1: Write (A)
T2: A := A + 300
T2: Write (A)

Two phase locking (2PL) :-

Requires lock & Unlocks in 2 phase

A transaction is said to follow the two phase locking protocol if all locking operations precede the first unlock operation in the transaction. Each transaction in 2PL issues lock and unlock request in two phases :-

Growing phase, Shrinking phase

The growing phase is also known as the expanding or first phase. In this phase transaction requires all the locks needed but cannot release any lock. (No. of locks increases)

The Shrinking phase is also known as second phase or the contracting phase. In this phase transaction releases existing lock and cannot require new lock. (no. of locks decreases)

Both are monotonic.

Eg:- T3

```
Lock X(A)
Read(A)
A := A - 200
Write(A)
Lock X(B)
Unlock(A)
Read(B)
B := B + 200
Write(B)
Unlock(B)
```

Any additional lock can be aquired at any time (Growing)

T4

```
Lock X(Sum)
Sum := 0
Lock S(A)
Read(A)
Sum := Sum + A
Lock S(B)
Read(B)
Sum := Sum + B
Write(Sum)
Unlock(A)
Unlock(B)
Unlock(Sum)
```

Once lock is released no additional lock is required (Shrinking)

Round (4.8)

4.8

DATABASE RECOVERY [04-Feb-14]

Why recovery is needed ?-
→ Database is centralized facility for many organizations
→ Several types of failure occurs coz it is accessed by many organisations.
→ Therefore several failures are bound to occur.
→ So it is important to restore database by recovery techniques.
→ Recovery is reqd. to protect database from any data loss and inconsistencies.
→ Recovery manager ensures the atomicity and durability property

Types of failures :-

1) System crashes :-
→ occur due to hardware, software and network error during execution of the transaction.
→ caused loss of content of volatile storage. But contents of non volatile store aren't effected.
→ In case of system crash we need to reboot system.
→ Cause : spontaneous shutting down of system due to power supply

2) System Error :-
→ These errors occurs when some of the operation in transaction may cause it to fail such as division by zero, integer overflow etc.
→ It causes computer system to enter deadlock state.
→ To avoid this, transaction should be rolled back

3) Disk failure :-
→ Occurs due to disk R/W head crash, power distribution, an error in transfer operation
→ Results :- loss of vital info, causes :- abnormal transmission
→ To avoid this, one should provide multiple copies of data on the other disk or magnetic tapes.

4) Logical errors :-
→ Occurs during execution when certain logical errors may occur that causes transaction failure

(5) Concurrency control enforcement :-
→ When multiple transactions are running at same time operation of transaction overlaps.
→ To make database consistent one transaction should be aborted.

6) Natural and physical disaster :- Floods, Tornados etc.

~~7) Natural and physical disaster :-~~

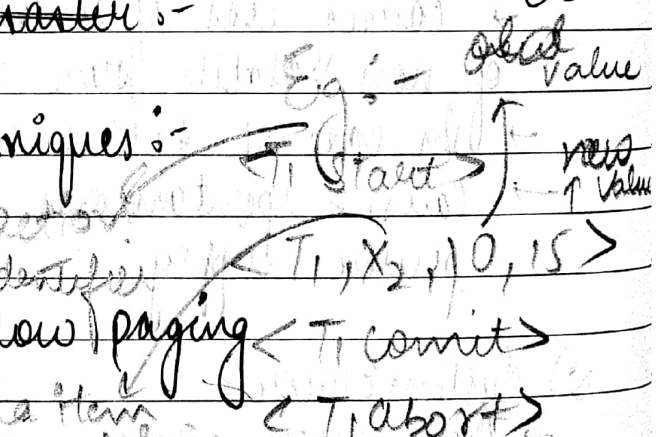
Database Recovery techniques :-

Log based Recovery techniques (keep backup of data on magnetic disk etc)

Shadow paging

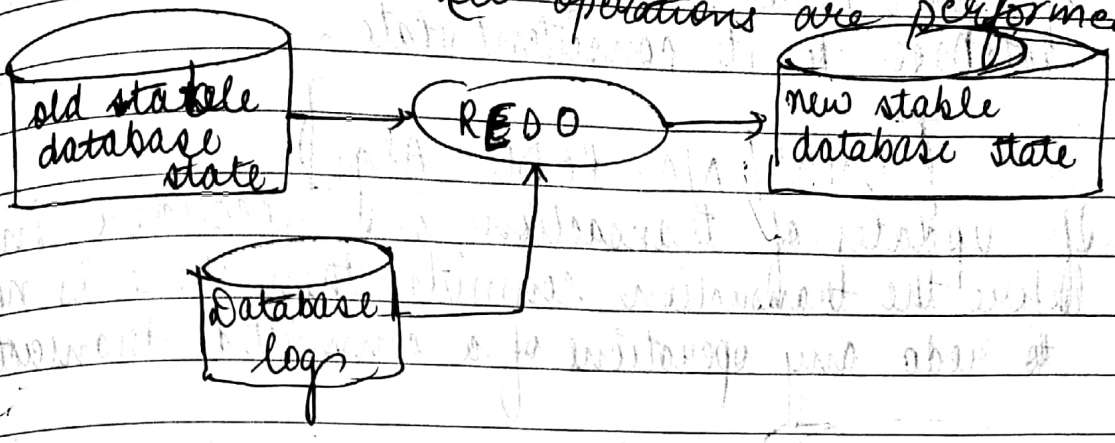
Deffered update

Immediate update (Recording database modification)



Deffered update :-

→ When we uses this technique, all update operations of the transaction to the database are deferred or postpone until the transaction enters partially committed state. (all operations are performed)



→ Before reaching commit point, all transition updates are recorded in buffers. During commit, the updates are first recorded into logs and is force written to disk. Updates are recorded in DB using info from logs.

→ Transaction operations do not immediately update the physical DB.

No Undo / Redo Algo

→ Used when transactions are short and each transaction changes only few data items in the database.

ac

If transaction fails before reaching its commit point, there is no need to undo any operation because transaction has not changed the database anyway.

→ Recovery using deffered is much easier

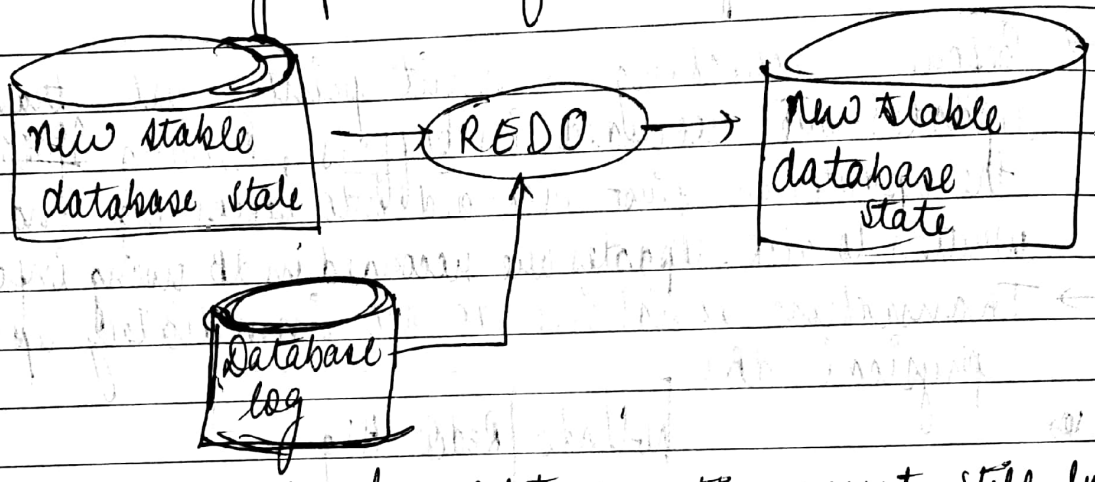
Immediate update :

→ Database is immediately updated by transaction operations during the execution of transaction before it reaches commit point.

→ In case of transaction abort, before it reaches commit point, a Rollback or ~~Undo~~ UNDO operation needs to be done to restore the database to its consistent state.

[UNDO/NO-REDO Algo]

→ If updates of transaction are recorded in DB before the transaction commits, then there is no need to redo any operations of a committed transaction.



→ The change made by update operation must still be recorded in log before the changes are recorded in the database using write ahead logging (WAL) technique so that we can recover case of failure.

→ Also known as backward recovery.

(Every write operation stored in log
(Undo effect (Rollback) Transaction backwards

Two categories

Undo/No Redo (All changes recorded in database before transaction commits)

Undo/Redo (Allows to commit before all changes are written in DB)

shadow page table never changes, current page table may change.

Directory

Current page table

Shadow page table

Both pages are identical during start of transaction

Shadow paging :-

→ In this scheme database is considered to be made up of fixed size logical units of storage called pages.

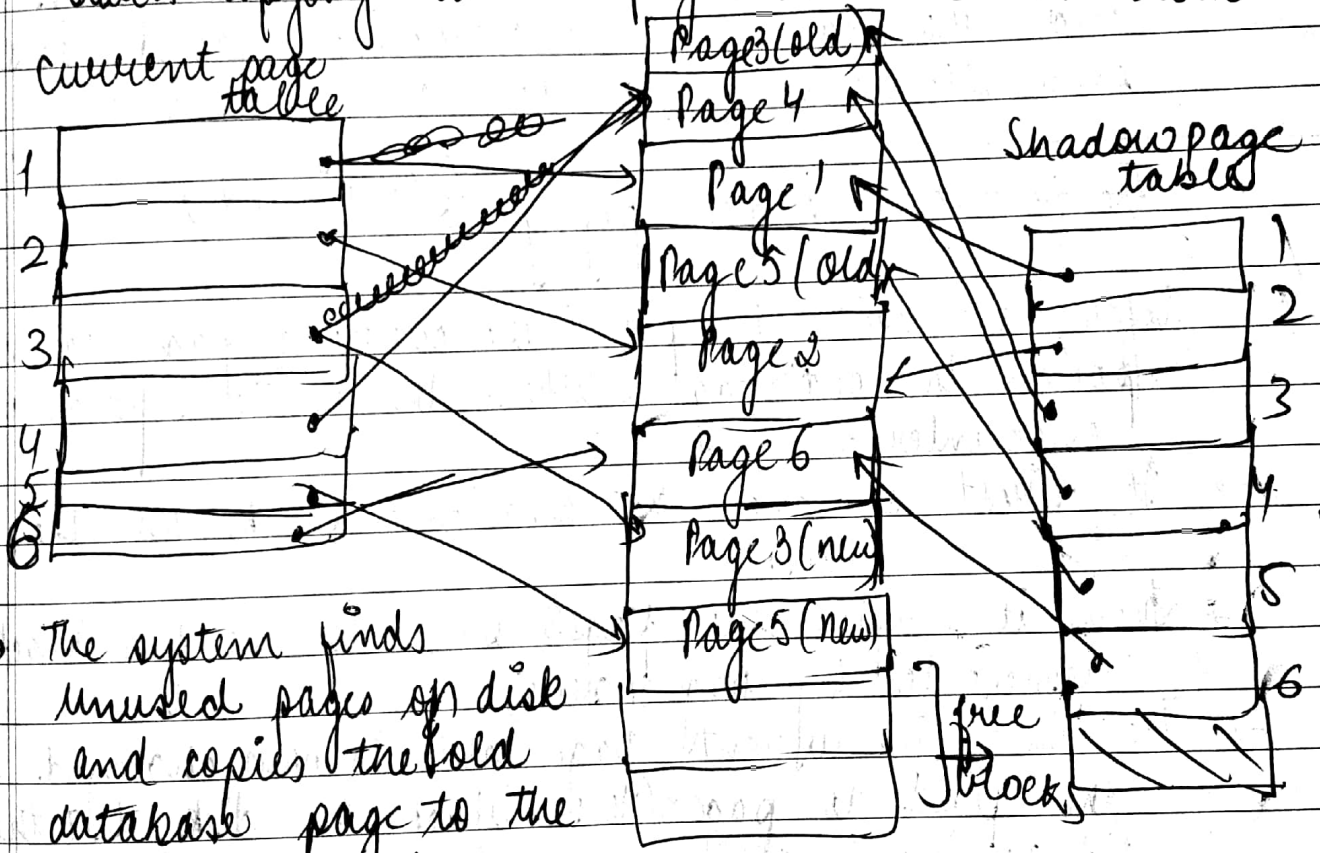
→ Pages are mapped into physical blocks storage by means of a page table with one entry for each logical page of database.

→ Consider a large table with n entries,

→ This scheme use 2 tables → current page, shadow page

→ The shadow page is populated when the transaction starts copying current page. Database Disk blocks

when transaction commits system writes current page to non-volatile storage and current page becomes shadow page



→ The system finds unused pages on disk and copies the old database page to the new database page.

→ The shadow page continuous to point old unchanged copy.

→ If transaction executes :- current page becomes shadow page.

→ If transaction fails :- New page are discarded & shadow page becomes current page

~~The figure~~ Shadow paging (Continued)

Before we write in page 3 the current page table points the old page 3. When the write operation is encountered.

→ It first searches the available block on the disk

→ Then it copies the page 3 block to the free block represented by page 3 (new)

→ The entry in current page table is now changed to the point to page 3 (new) block on disk, but the old version is still referenced by the shadow directory because it is not modified

→ The changes are propagated to the page 3 (new) block pointed to by the current page table.

~~Commit operations~~ :-

Advantages of shadow paging :-

→ Require fewer disk accesses than the log based method of recovery under certain circumstances

→ Recovery from system crash using scheme is relatively inexpensive and faster.

→ NO UNDO / NO REDO OPERATIONS are reqd during recovery using this technique.

Disadvantages of :-

→ Since the updated database pages changes location on the disk so this makes it ~~different~~ difficult to keep database pages closer together on disk without complex storage

→ Whenever transaction commits, the original version of the changed blocks pointed to by the shadow.

→ The commit of a single transaction using shadow Paging requires multiple blocks to be output such as actual data blocks.

→ It is hard to extend shadow paging to allow multiple transactions to execute concurrently because it requires additional book keeping and in such a scheme some logging scheme is used.

Checkpoints :-

A checkpoint operation, performed periodically, copies log information onto stable storage. The information and operations performed at each checkpoint :-

→ A start-up checkpoint along with time and date of the checkpoint is written to the log on a stable storage device.

→ All log info from buffers in variable storage is copied to the log on the stable storage.

→ All database updates from buffer is copied to log.

→ At the end of checkpoint record is written and the address of the checkpoint record is shared on a file accessible to the recovery routine.

RELATIONAL QUERRY LANGUAGE

Sql :- Structured Query language

Basic Sql :-

Sql is component of oracle which is used to execute sql statements.

Sql (*) plus is also used for developing scripts for controlling indexes, the tables and other database objects.

It offers consistent environment for executing the scripts regardless of the OS in which it resides.

Sql commands

- CREATE
- REPLACE
- APPEND

1. Display whole content of table employee.
   select * from emp;

2. If you want to save it into the file
   save Emp.sql created file emp.sql.

3. Type the command to return a description of the Dept table
   Desc dept;

4. List all the columns of emp and dept table.
Select * from emp;
Select * from dept;

5. Display all employees from the emp table.
Select ename from emp;

6. Display all employees whose salary is less than 2500.
Select ename from emp where salary < 2500;

7. Display all the employees who are "clerks" and whose name starts with letter "S".
Select ename from emp where job = "clerks" and ename like 'S%';

8. Display all the employees who were hired before the year 1975 and who belong to deptno 20.
Select ename from emp where hiredate < 1975 and dept=20;

9. Write a Sql query that will return the name and job of all employees, ordered alphabetically by name.
Select ename, job from emp order by ename.

10. Display name and salary of all employees whose names end with letter R and who do not earn a commission
Select ename, sal from emp where ename like '%R' and comm is NULL;

11. Write a query to display unique job codes from emp table
Select distinct (job) from emp;

12. Write a query to display employees belonging to deptno : 10, 20, 30
Select ename from emp where dept in (10,20,30)

Uses of SQL :-
→ It can execute queries against a database
→ It can retrieve data from database
→ It can insert, update & delete records in database
→ It can create new databases, tables
→ It can store procedures in database, create views in it.

13. Write a query to count different countries in table customer
- Select count(distinct(country)) from customers;

14. ~~Se~~ Write a query to select whole content of table customers who lives in Germany and city as Berlin
Select * from customers where country = 'Germany' AND city = 'Berlin';

15. city can either be berlin or ~~Germany~~ München
Select * from customers where ~~country~~ city = 'Germany' Berlin
OR city = 'Berlin München';

16. ~~Select~~ Write a query to display the content of customers who does not belong to Germany
Select * from customers where NOT country = 'Germany';

17

CustomerID	CustomerName	ContactName	Score
89	White Clover	Matti	90
90	Markets	Kale	80
91	Wolski	Zbyszek.	95

(a) Insert another row into table Customer
Insert into Customer (CustomerID, CustomerName, ContactName) values (92, 'Aman', 'Kali', 80);

(b) Change the customer Name from Markets to Smith
Update customer set customername = 'Smith' where customer name = 'Markets';

(c) Deleting a record of customer ID = 90
Delete from customer where customerID = 90;

(d) Delete all records.
Delete from customers;

(e) Select first 3 records from customer table.
Select top 3 * from Customer;

(f) Display highest and lowest score from CustomerID Table
Select min (score) from customer;
Select max (score) from customer;

(g) Display total no. of records enter
Select count (customerID) from customer;

(h) Display avg score of customer
Select avg (score) from customers;

(i) Display total score.
Select sum(score) from total;

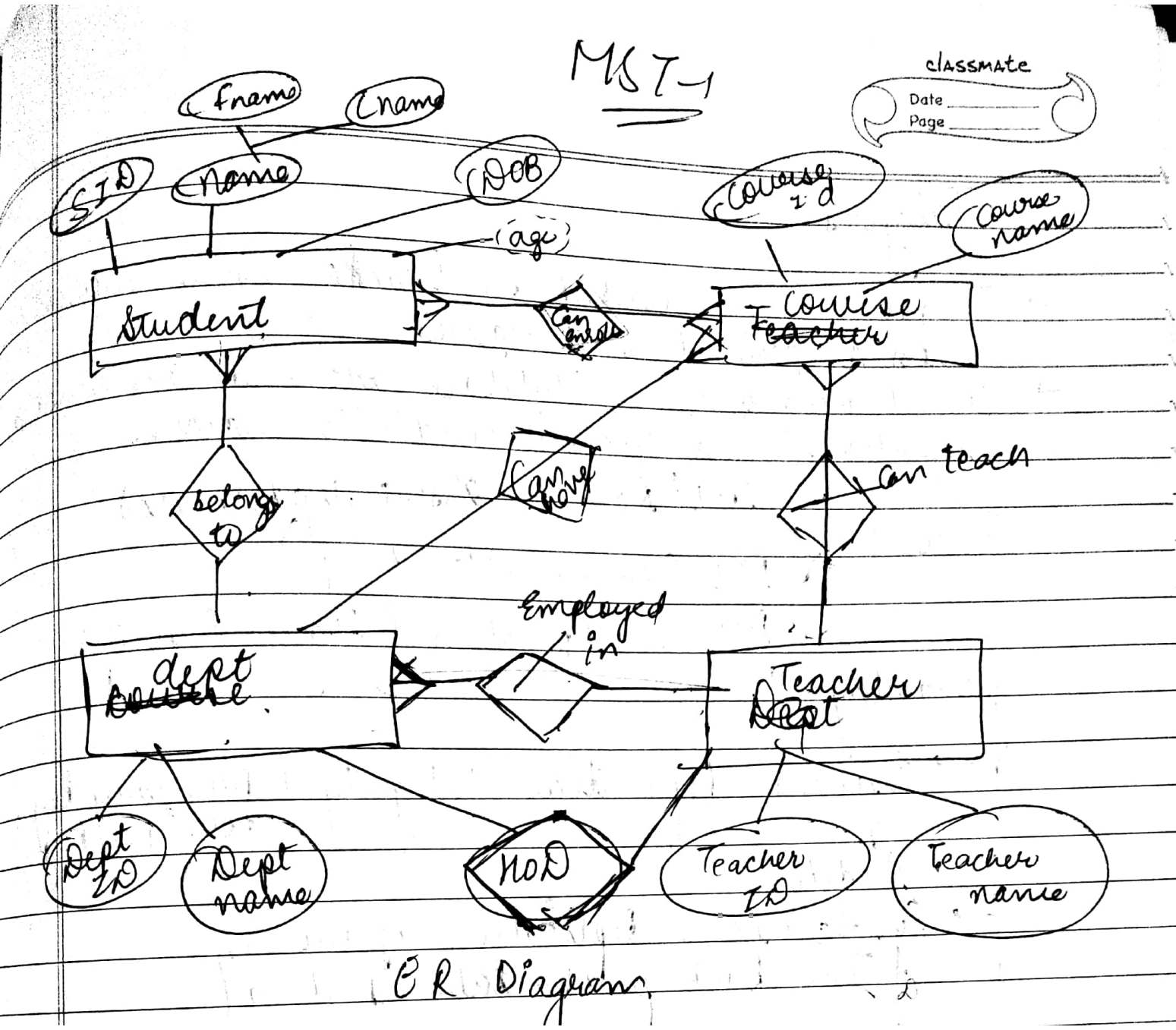
Wild characters :-

* → Represents 0 or more characters
? → Represents single character
^ → Represents any character not in bracket
% → Represents 0 or more characters

(j) Display customer ID ~~where~~ of that customer where score ~~total~~ is more than 85 and less than 90
Select customer ID from customer where ~~customer~~ ID between 85 and 90;

(k) Display customer ID where score is not b/w 80 & 90
select * customer ID from customer where score NOT BETWEEN 85 and 90;

MST-1



ER Diagram

Triggers

→ Triggers are invoked by Oracle Engine automatically when a specified event occurs

→ Stored programs in database and are invoked repeatedly when specific condition matches.

Events :- DML (Delete, Insert, Update etc)
DDL (Create, Alter, Drop)
Database Operation (Login, Logout)

Advantages :-

→ Enforces Referential Integrity (primary key, foreign key)
→ Auditing
→ Synchronous Replication of tables [Time backup of table data]
→ Event logging [Emp → Log]

Syntax :-

Create or Replace Trigger trigger_name
trigger_timing (BEFORE / AFTER)
event1 or event2 (Insert / Update / Delete)
On table_name
PL/SQL Block
/

Update previous Trigger if of same name exists.

Example :-

Create or Replace Trigger sal_change
Before delete or insert or update on Emp
For each row
Declare
i number;
Begin
if new.sal - old.sal
insert into Emp (valdiff)
end; values(i)

name = Execution of event

Trigger Type = Statement / Row

Statement :- Executes Only Once for triggering event
Row :- Executes Once for each row effected by triggering event.

Views

Table → Emp < 10 - view10
20 - view20

→ Virtual Table
→ View is a saved sql query

create view

Syntax :-
create or replace
view view_name
as
select col names
from table_name
where condition

Eg :-
create or replace
view view10 as
select * from emp
where depid = 10;

Drop view

Drop view view_name;

Views are used when we have to restrict data depending on type of user you are giving access.

PROCEDURES & Cursors

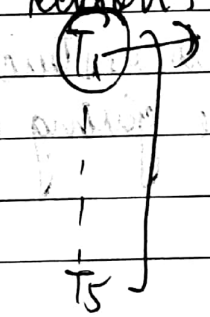
Create Procedure

```
Create or Replace Procedure
  raise_sal (e_no emp.empno %Type)
  is
  BEGIN
     update Emp
     set sal = sal + 1000
     where empno = e_no;
  End raise_sal;
```

Starvation

A transaction is starved if it cannot proceed for an indefinite period of time while other transactions in the system continue normally.

Reason:



select & abort & Rollback by deadlock prevention algo.

If it happened too many times then T1 will be starved

T1 → Modify timestamp

→ Increase priority of transaction (jese hoka execution complete ho jaye)

```
T1
lock S(A)
read (A)
lock X(B)
If A=0 then B=B+1
write (B)
unlock(B)
unlock(A)
```

Internal / External

Database Threats :-

→ Data Tampering :-

Threats :- Software damaged [security damage]

Data Tempering :- Data send from one side to another- dusri side pe change kr dega.

Evasdropping :- 2 ppl communicate, 3rd person sitting wala case.

Falsifying User Identities :- Fake identities

Password related threats :- same password case

Unauthorised access to Data :-

Lack of accountability :- Large data wala case, no checking regularly

Defence Mechanisms

→ Physical Security :- Data should be kept in storage devices, user identification & passwords should be kept confidential etc

→ Human Factors :- Large scale organisation (Notes)

→ Operating System :- Security of operating system is dependent upon os

→ DBMS Security :- Database security

Ques 1:- Create the table "student" with the following columns and constraints :-

Name Not Null constraint Name name_null
Roll no Primary key.
Class
DOB
Marks between 0 and 100
section valid sections are "A", "B", "C", "D"
Grade valid grades A B C D.
Grade A Marks >= 80
Grade B marks >= 60 but less then 80.
Grade C marks >= 40 but less then 60.
Grade D marks < 40.
Sex valid entries "M", "F"
City Not Null.

Ans:-
```
Create table Student(
name varchar2(20) constraint name_null Not Null,
Roll no number(10) Primary key,
Class varchar2(20),
DOB Varchar2(20),
Marks number(20) check_marks between (0 and 8),
section varchar2(20) check_section in ("A", "B", "C", "D"),
Grade varchar2(20) check_grade in ("A", "B", "C", "D"),
Sex varchar2(10) check_sex in ("M", "F"),
City varchar2(10) Not Null);
```

Ques 2 (a) Add a new column "hobbies" of type Varchar2 size 80
Alter table student add (hobbies varchar2(80));
(b) Add a constraint on "marks" field so that marks should not be more than 100. Name this constraint as "mark_check".
Alter table student add check marks <=100.00;
(c) Drop the not null constraint from "name" column.
~~Drop to~~ Alter table student drop not Null;
(d) Modify column width of class column by reducing it by 10 when should this work?
~~Update student set~~ Alter table student modify class varchar2(10);
(e) Increase the column width of column "name" to 40.
Alter table ~~cost~~ student modify name varchar2(20);
(f) Drop the column "hobbies" from the table
Alter table student drop Hobbies;
(g) Ques 3 (a) List all students who secured grade A and who are born before 1993
Select name from student where grade = 'A' and DOb < 1993;
(b) List all details of students with roll nos: 11, 21, 35, 45, 75 and 85.
Select * from students where roll in (11, 21, 35, 45, 75, 85);
(c) List all females students who are from "Delhi & whose name does not end with letter "k"
Select name from student where sex = "M" and city = "Delhi" and name not like "%k";
(d) Alter table student ~~modi~~ set marks = marks + 0.05 marks where sex = "M";
(e) Desc student;
(f) Delete * from table where marks < 40;

PL/SQL fundamentals

PL/SQL
Procedured language / Structured Query language
→ Extended version of SQL
→ Superset of SQL
→ Eliminates restrictions of SQL
→ Adds control structures

Difference between SQL and PL/SQL

SQL	PL/SQL
1. Execution of single query or command at a time	Execution of block of code at a time
2. Stands for structured Query language	Stands for Procedural language / Structured Query language
3. In SQL we write queries or commands using DDL, DML statements	In PL/SQL you can write block of code that has procedures, functions, packages or variables etc
4. We can modify, add, delete or manipulate data in database using SQL	We can create applications or server pages that displays the information obtained from SQL in a proper format.
5 It uses semi colon for execution of each statement	It does not uses.
6. SQL does not support PL/SQL statements	PL/SQL support SQL block.

Advantages of PL/SQL:-

→ Supports declaration and manipulation of obj. types
→ External functions and procedural calls allowed
→ Built in package library
→ Triggers :- Prog. stored :- Database executed immediately before / after INSERT, UPDATE & DELETE commands.
→ Cursors :- named for workspace to execute SQL commands
→

Elements of PL/SQL:-

→ Operators, Indicators & punctuations
→ Identifiers → literals, comments, expressions.
→ Datatypes & Declarations

Operators :- +, −, *, /, **.

Expression oper :- := , ... (range) (1...4), || (concate)

LIKE, NULL, BETWEEN, IN, AND, OR, !=.

Identifiers :- Named conventions used for var, const,

group by

Ques 1:- List various categories of operators

Ans:- Various operators are:-
=, < >(not equal to), >, <, >=, <=, BETWEEN, LIKE, IN

Ques 2:- List various categories of functions in Oracle Sql.

Ans:- → Single row functions
(i) Number functions :- abs(n), ceil(n), floor(n), exp(n) etc.
(ii) Character functions :- ASCII, chr, concat(m,n), etc.
(iii) Date functions :- sysdate, last_day, trunc etc
→ Multiple row functions
(i) Count(*)
(ii) distinct
(iii) sum etc.

Ques 3:- List and explain the aggregate functions

Ans:- Aggregate functions are multiple row functions. Some of the aggregate functions are listed below:-
(i) COUNT
(ii) MIN
(iii) MAX
(iv) AVG
(v) SUM
(vi) STD DEV
(vii) VARIANCE

Ques 4:- Following queries in dual table
(i) Arithmetic functions :-
→ CEIL
```
Select ceil(2.2) from dual;
```
→ MOD
```
Select MOD(5,2) from dual;
```
→ SIGN
```
Select sign(-3) from dual;
```

→ trunc
Select trunc (43.826,2) from dual;
→ ROUND
Select round (-43.282,1) from dual;
→ ABS
Select abs (23.12) from dual;
→ SQRT.
Select sqrt (64) from dual;
→ FLOOR
Select floor (2.2) from dual;
→ ~~SIGN~~ POWER
Select power(2,3) from dual;

(ii) Character functions :-
→ CHR
Select chr(95) from dual;
→ ASCII
Select ASCII('A') from dual;
→ INTCAP
Select INTCAP('aayush') from dual;
→ CONCAT
Select concat ("abc","abc") from dual;
→ LPAD
Select ename, lpad(ename,10,*) from dual;
→ RPAD
Select ename, rpad(ename,10,'L') from dual;
→ LTRIM
Select ~~LTRIM~~ (aman, a) from dual;
→ RTRIM
Select RTRIM(aman, a) from dual;
→ SUBSTR
Select ('aman','a',2) from dual;

Ques:- create a table "Employee" with the following columns and constraints.

Emp no	Varchar2(6) Primary key
Emp Name	varchar2(30) not null
Dob	date
city	varchar2(15) default = "Amritsar"
Basic	Number(7,2)

Ans:- create table Employee (Emp no varchar2(6), Emp name varchar2(30) not null, Dob date(), city varchar2(15) Default 'Amritsar', Basic Number(7,2));

Ques 2:- Create a table "Qualification" with the following columns and constraints:

Emp no	varchar2(6)
Proj_Name	varchar2(30)
doa	date
doc	date

Ans:- create table Qualification (Emp no varchar2(6), Proj-Name varchar2(30), doa date(), doc date());

Ques 3:- Alter the above tables as follows:-

(a) In the "Employee" table, apply the NOT NULL constraint to dob column.

Alter table employee ~~add not Null(dob);~~ add Not Null

(b) In the "Employee" table, apply not null constraint to empname column

Alter table employee add not null (emp Name);

(c) In the "Qualification" table make Empno column as the foreign key with its primary key as Empno column from "Employee" table.

(b) In the "Qualification" table, apply the NOT NULL constraint to doc and doa columns.

Alter table Qualification add NOT Null (doc, doa);

Ques6) Answer the following queries :-

(a) Display all employees who were born on a "Sunday".

Select Empname from Employee where dob = 'Sunday';

(b) Display all project details for the projects completed in less than 1 year.

select * from "Qualification" where doa < 1;

(c) Display all employees who where born in "Aug 1974".

Select Empname from Employee where dob = 'Aug 1974';

(d) Display all employees whose empno starts with letter "A", name ends with letter "R" and who are born on 3rd day of week before 1995

Select ename from Employee where ename like 'A%R', ~~dob < 1995~~ dob like '%1995';

(e) Update the city column for all the employee, whose empno starts with letter "S" or "M", from "Amritsar" to "Delhi".

Update Employee set city = "Delhi" where city = 'Amritsar' and empname like '%S' or '%M';

→ LENGTH
Select ename, length(ename) from dual;
→ LOWER
Select lower('AAYUSH') from dual;
→ UPPER
Select upper('aayush') from dual;
(iii) Date functions:-
→ SYSDATE
Select sysdate from dual;
→ MONTHS_BETWEEN
Select hiredate, months_between(sysdate, hiredate) from ~~emp~~ dual;
→ LAST_DAY
Select sysdate, last_day(sysdate) from dual;
→ NEXT_DAY
Select sysdate, next_day(sysdate, 'sunday') from dual;
→ ADD_MONTHS
Select sysdate, add_months(sysdate, 2), ADD_months(sysdate, -12) from dual;
→ TO_CHAR
Select sal, to_char(sal, '$9999') from emp/dual;
→ TO_DATE
Select to_date('12109l2003', 'dd/mm/yyyy') from dual;
→ TRUNC
Select trunc(sysdate, 'MON') from dual;
→ ROUND
Select hiredate, round(hiredate, 'mon') from emp/dual;
(iv) Aggregate functions:-
→ COUNT
Select count(*) from dual;
→ SUM
Select SUM(salary) from employee/dual;
→ AVG
Select avg(salary) from employee/dual;

→ MAX
Select max(marks) from student / dual;
→ MIN
Select min(marks) from student / dual;
→ STDDEV
Select stddev(sal) from emp;
→ VAR
Select variance(sal), variance(comm) from emp;

v) General functions :-

→ GREATEST
Select greatest('A', 'B', 'C') from dual;
→ LEAST
Select least('A', 'B', 'C') from dual;
→ NVL
Select (name, comm, NVL(to_char(comm))) from ~~emp~~ dual;
→ UID
Select UID from dual;
→ USER
Select user from dual;

Tutorial sheet-1 CSE-1

Ques1:- What is Sql and PLSQL?

Ans:- SQL is the standard language to query a database PLSQL basically stands for "Procedural language extensions to SQL. This is the extension of structured Query language (SQL). that is used in Oracle.
SQL executes the single query at a time whereas PLSQL executes block of code at once.

Ques2:- Write a complete syntax for select statement?

Ans:- Select <select_list> from <tablelist> [where <condition>]
[group by <column 1>, <column 2> --- <column n>]
[Having <condition>]
[order by <expression>];

Ques3:- Explain the order by clause and its usage?

Ans:- Order by clause comes after the from clause. The order by clause allows you to sort the result set based on one or more columns in diff orders: ascending & descending you put the columns name that you want to use to sort after the order by clause followed by the ASC to DESC keyword.

Ques4:- List various Oracle datatypes?

Ans:- Character - char, varchar, varchar2, long, row etc.
number - number etc.
Date
LOB's (large objects)

Queries

1. Type the command to return a description of the dept table
   Desc dept;

2. List all the columns of Empand Dept table
   select * from emp;
   select * from dept;

3. Display all the employees from the emp table.
   select ename from emp;

4. Display all the employees who are "clerks" and whose name starts with letter 'S'.
   select ename from emp where job = "clerk" and ename like 'S%';

5. Display all the employees whose salary is less than 2500.
   select ename, from emp where sal < 2500;

6. Display all employees who are hired before the year 1975 and who belongs to deptno 20.
   select ename from emp where hiredate < 1975 and dept = 20; 1/2

7. Write a SQL query that will return the employee number and name of all clerks, ordered by employee number.
   select ename, eno from emp order by eno;

8. Write a sql query that will return the name and job of all employees, ordered alphabetically by name.
   Select ename, job from emp order by ename;

9. Display name and salary of all employees whose names end with letter "R" and who do not earn a commissio
   select ename, sal from emp where ename like '%R' and comm is NULL;

ple. a query to display unique job codes from emp

Select ~~ename~~ distinct job from emp;

Write a query to display employees belonging to deptno: 10, 20, 30

Select ename from emp where dept in (10, 20, 30);