DBMS
(Data Base Management System)

Data base

→ Data base is a collection of interrelated data which is used to insect, delete and retrive the data efficiently.

→ It is used to organize the data in table form schema views and report etc.

char → only alphabates
varchar → both numericals alphabates

→ DBMS is a software, which is used to create and manage the data base.

Ex :- MYSQL, ORACLE etc. MS access

→ MYSQL, ORACLE are very popular commercial database, which is used in different application.

→ DBMS provides an interface to various operation like data base creation, storing data, updating data, creating a table in the data base.

→ Its provides the protection and security to the database in the case multiple use as it also maintaing data consistency.

Data

The data refers one type of raw facts i.e. processed so that it doesn't proper objective.

Ex :- Student name etc.

→ For user understanding on verbal way two types for language specification to be defined.

Quantitative

→ When the users expressing all over the terms in numerical format i.e. called quantitative data.

→ It mainly represent integer format or decimal format.

Ex:- 121, 13.24 etc.

Qualitative

→ When the users expressing all over the terms in characters.

By combination of both numeric, symbolic functionlitied, specific key, or any alphabatic characters i.e. called Qualitative data.

Data base

→ It is the collection of relevant data that contains information relevant for particular organisation. So that it gives a proper objective.

→ It represent some accept of the real world design and co-pre-related with data for specific purpose.

Reductancy

→ In file processing system each users maintain separate file for different application the same information is duplicated in multiple files.

→ This reductancy in defining and storing data result in wastage memory space.

→ The data reductancy also occurs when the file contains similar records on same location that times also the reductancy occurs.

Inconsistancy

→ As the file containing various copies of same data that may no longer agree that means the existance period is very low when it contains some copies on same location.

→ In this case the file to be erased automatically.

Data access very difficulties

→ As you know without operating system the users can't work very easily.
→ Incase of file based system if the user wants to access a particular record by using an unit field that can't be possible, so developers or users first write down the access program by using unit field then the particular record to be accessed.

Integrity Program

→ The integrity problem to be occure when the file based system to be used because the data value stored in the data base must satisfied certain type-condition so that easily all types of updation to be occur

Atomicity Problem

→ The file based system the file may be containing unprocessed or view data so that the users cannot be find out any specific objective.
→ But the database system the atomicity nature may be occure that means all the data to be processed in a format and gives an accurate objective or input.

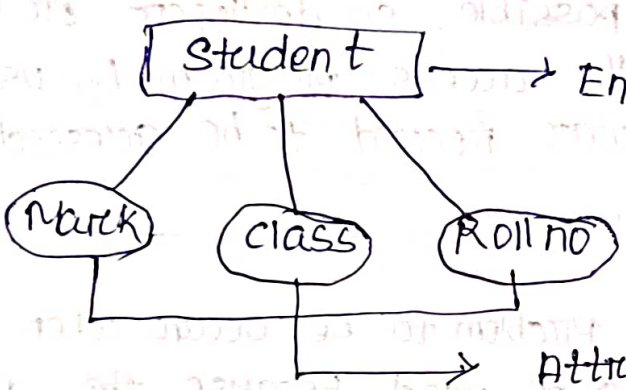
Securcity

→ In case of file based system any user can access and manupulate old data, which is stored in between file, Because there is no securcity mechanism to be used.
→ So that in DBMS one authentication (identify) mechanism to be produced for data secure at the time of unauthorised users access.

VIEW OF DATA

Data abstraction

Data abstraction refers to the supression of data organisation and storage details and also so the relationship between different data attributed.



→ A major purpose of a DBMS is to provide users with an abstract view of the data view of the data that means it doesnot describes the detail background of the database.

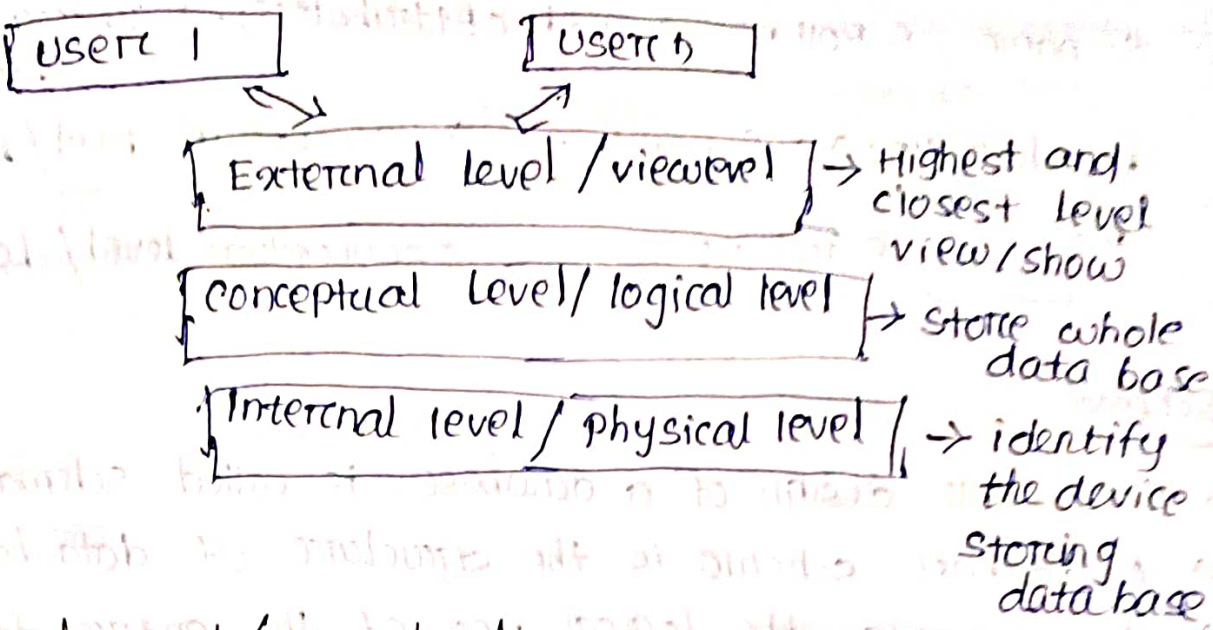
→ To archive data abstraction three level architecture proposed.

(i) External level / view level
(ii) conceptual level / logical level
(iii) internal level / physical level

→ The primary goal is to separate the users application and the physical data base.

→ The architecture to be developed by ANSI in 1975.

→ The three level architecture divided into 3 types.



① External level / view level

→ The view level of the architecture of DBMS is the level closest to the user.

→ It is the highest level of the base abstract.

→ The level of abstraction that describes only a part of the entire data base i.e. schema.

② Logical and conceptual level :-

It is the middle level of the database abstraction.

→ It describes the logical structure of whole database.

→ It describes what types of data are stored in the database and what relationship exist among those data.

③ Internal level / physical level

It is the lowest level of data abstraction, it describes how the data are actually stored by using which device.

→ It describes the physical storage structure of database.

Student → Database name

S name	S - Roll no	S - address
X	10	ctc
Y	11	RKL
Z	12	BBSR

S-name, S Roll no, S add (Attributes) → External level / View level

varchar(20) S name → conceptual level / Logical level

Int S roll no → conceptual level / Logical level

Schema

→ The overall design of a database is called schema.
→ A database schema is the structure of data base.
→ It represents the logical view of the entire database.
→ A schema contains objects like table, different keys, views, coloumns, data types, etc

Database Instance

→ The content of data which is stored in the data base at a particular moment of time is called an instance of the data base.

Database state

→ The content of the database
→ Schema is also called intension and state is also called extension.

Data Independence

→ The ability to modify a schema defination is one level without affecting a schema defination in the next higher level is called data Independence.

→ There are two levels of data Independence

1. physical data Independence
2. Logical data Independence

① Physical data Independence

→ The physical data independence refers to the ability to modify the schema followed at physical level without affecting the schema followed at a conceptual level.

② Logical data Independence

It refers to the ability to modify the logical schema without causing any changes in application programmer view program.

→ Logical data Independency is more difficult to archieve than the physical data Independence.

Data base Language

A data base system provide 3 language

(I) DDL (Data defination language)
(II) DML (Data Manipulation language)
(III) DBM (Data base administrator)

① DDL

→ It stands for data defination language.

→ The DDL is used to specify the data base schema.

→ The result of DDL compilation is stored in special file, is called data dictionary.

→ The result of DDL is also known as Meta data or system catalogues.

Varchar 2 → latest version of varchar

Ex :- create table, drop table, rename table.

(11) DML

→ It stands for Data manipulation Language.

→ Data manupulation language to be used for to allow the users to access or manupulate the data to be stored in the data base.

→ Data manupulation of new information into the data base.

→ The deletion of information from the database.

→ The modification of information stored in the data base.

Ex :- Select, Insert, delete, Modify, retrive

Program

```
Create table student
(
name varchar (20)
roll-no number (5)
address varchar (25)
mark number (7)
course varchar (30)
phone-no number (10));
desc student ; (view for table)
```

name	roll no	address	mark	course	Phone-no

(iii) DCL

→ It stands for data control language.

→ It is used to control the transaction, it also grants priviledge to others users (permission)

Ex :- Save, Revoke, count, grant, etc.

① CREATE :- This command is used to create table in the data base

Syntax

create + table < table name > ;

② DROP :- This command is used to remove table from the database.

Syntax

DROP table < table name > ;

③ RENAME :- This command is used to rename the table

Syntax

Rename < table name > ;

④ SELECT :- This command is used to select data from a database.

Syntax

Select < col-name >, col -name 2 > - - - from < table name > ;

⑤ select* → Syntax

select* From < table name > ;

⑥ Insect → This command is used to delete the data from a table.

Syntax :-

Insect into < table - name >

(u datatype - ———) ;

(7) Delete → This command is used to delete the data from a table.

Alter → The Alter table statement changes, the structure an existing table.

→ It is possible to add / drop a colomn modify the datatype of a coloumn, rename a coloumn or a table.

Syntax :- Alter TABLE <table-name> ADD <new datatype;

Alter TABLE <table-name> DROP<existing

(8) UPDATE :- This command is used to update the existing data within a table.

Syntax :-
update <table-name> SET<col-name>=value;

(9) COMMIT :- This command is used to ~~upd~~ save the work done.

(10) To display the table command - Desc<table-name>;

DBA

→ It stands for database administrator.

→ DBA is a person or group of person responsible for overall control the database that means the imitiation and final state of database to being carried act by DBA.

→ The objective of DBA to be given below, deciding the information content of the database i.e. identifying the entities of interest to the interprise and the information to be recorded about those

entities of interest to the DDL schema.

→ Deciding the storage structures and the memory space that means with the help of DBA to the end users to be known what type of storage devices to be used for data storing and also specify how much memory to be required for particular data storage before creating the data base.

→ This storage structure to be initialized by using physical schema.

→ With users i.e. that ensures that the data they required is a available and to write the external schema by using DDL language.

→ Defining a strategy for backup and recovery if the data to be hault or damaged due to some software and hardware interrupt in the middle part.

→ Manifesting performance and responding to change in requirement all things in manulation to be carried out with the help of DBA.

→ DBMS is a software which is used for controlling the database, access that means all manupulation easily carried at by DBMS software.

Ex :- Insecting, Deleting, updating etc.

Structure of DBMS

The interesting schema of the database execution to be performed logically in an authentication manner. So it goes to different stages.

Database users

→ Database users are persons who worked in a database.

There are 4 different types of system users:

1. Application programmers.
2. sophisticated users
3. Specialised or DBA users
4. Native users

① Database Application programmers (How to create and write database)

→ They are computer professionals who interact with the system through DMA by embadded with call some application specific purpose software.

→ Those users to be initiated any software their which all the data to be stored to system this given software to be created using any host languages (C, C++, FORTAN, COBOL)

② sophisticated users

→ They interact with the database system without application interface, that means those users to be direct interact with system database by using any database language.

③ Native users

→ Their unsophisticated users who interact with the system by involving one of the permanent application program that have been written previously.

④ Specialised users or DBA users (How to create a data base
How to access the database)

→ They are sophisticate users who writes specialised database application that don't feet

into the traditional data processing from work.

→ Those users to be store the abstract knowledge of database which is required database identified like database schema, memory states and relationship between data schema.

→ The function of the DBMS internally carried act by using two methodology.

① Query processor component
② storage management.

Data base user and administrator

→ A person who has central control of both data and program access those, that is called a database administrator.

→ Database administrator include the following function

① schema defination
② storage structure and access method
③ storage and physical organizational modification.
④ Greating of authorisation for data access
⑤ Rating maintenence

who knows the query - sophisticated

Advantages of DBMS

1 → control database reductancy

→ All the data is stored in one place and that recorded in the database.

→ Hence the reductancy in the database.

2→ Data sharing

→ DBMS allows users with authority to share the data in database with multiple users.

3 → Reduce time

4 → Easy maintenence

5 → The centralized nature of the database helps the easy maintenence of the data.

5 → Back up

6 → Multiple user interface

Disadvantage

1→ cost of software and hardware.

2→ size

→ a large amount of storage size is required to run DBMS.

3→ complexity

→ DBMS adds an additional layer of complexity to the data.

4→ Higher impact of failure

Transaction management

→ A transaction is a collection of operation that performs a single logical function in a database application

→ Each transaction is a unit of both atomicity and consistency.

Atomicity

The transaction is ~~a collection~~ entirely done or not at all i.e. all or none requirement is called atomicity.

Consistency

→ The correctness must be equal. If is essentially that execution preserve the consistency of the database i.e. correctness is called consistency.

Schema and Instances

Instances

→ Database change occure over time as information is inserted and deleted.

→ The collection of information stored in the database at a particulare moment is called instance of the base.

Schema

→ The overcall design of the database is called schema.

→ A schema is a collection of database objects schema, objects and logical structure created by user to contains or reference their data.

→ schema object includes structure like table, views and index.

→ The relational database, the schema defines the table, the fields, relationship, views, packages, procedures, Functions, Queries and other elements.

→ A database schema is a collection of meta data that describes the relations in a database.

→ A schema can be simply describe as the layout of a database or the blueprint of the outlines the way i.e. organized into tables.

→ In general schema can be catagorised by 2 types :

① Logical schema

② Physical schema

① Logical schema

It is consult with exploting the data structure offered by DBMS in order to make the schema understandable to the computer.

② Physical schema

It deals with the manner in which the conceptual database shall get represented in the computer as stored database it can usually be changed easily without affecting application program.

Sub schema

A sub schema is a subject of a schema an inherites the same property that a set has subschema refers to an application program. view of the data item types and recorded types.

Data Model I.M.P

→ A data model to be used for representing the real database structure in the format of blue print.

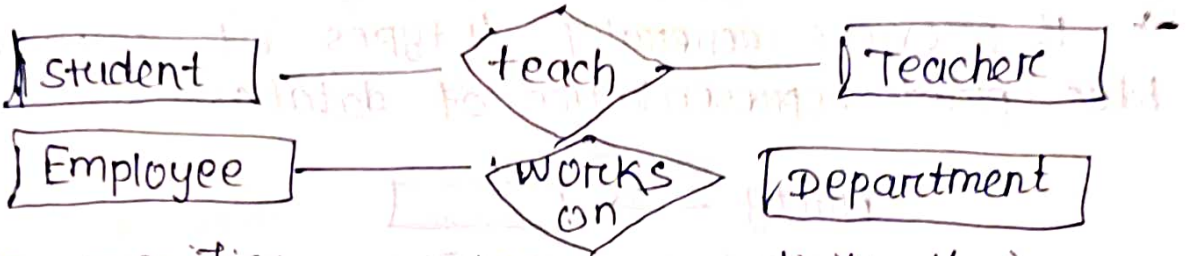
→ It is the collection of conceptual tools for describing the relationship between the data value and view tool of collection the schema value to representing the pictural notation.

→ The data model is divided into 3 parts
① object based logical model
② Record based logical model
③ physical based data model.

① object based logical model

It is used to describing data in logical and view level that means by using this methodology the user to be represent the blue print structure and its relationship between the given example

Ex:-



→ It specifies constraints explicity. It is again classified into 4 types
① Entity relationship model
② object oriented model
③ Simating data model
④ Functional data model

① Entity Relationship model

→ This model is used to describe the relationship among physical data item.

→ This model consisting of a collection basic object called entity and relationship among these objects.

→ An entity is a thing or object in the real world that is ~~design~~ g distinguishible from other objects.

→ Entities are represented by collection of no. of attributes and methods.

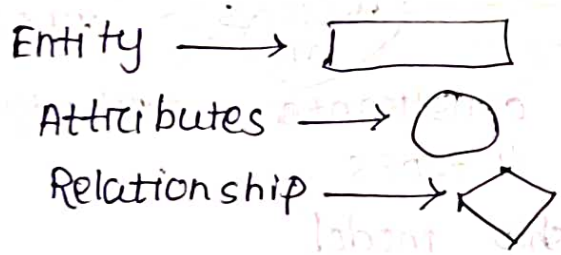
→ The attributes means same properties of the given entity which is created by user.

→ The set of all entities of same type is known as entity set.

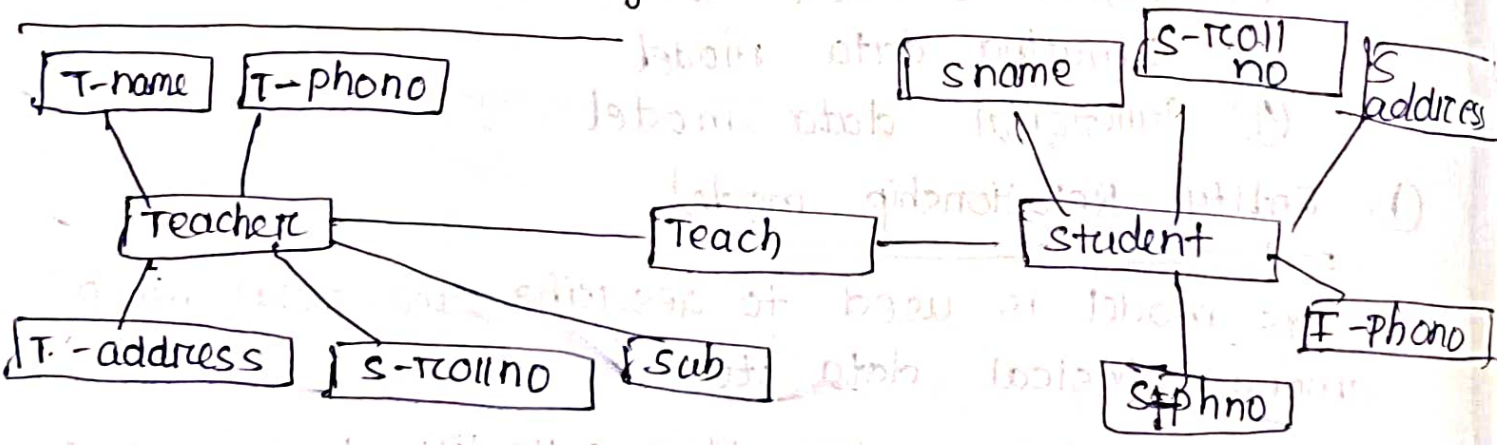
→ The ER-diagram (Entity relationship diagram) also provides for association mechanism to create the relationship between one object to another object.

→ It also provides some constraints condition to assign on blue print representation.

→ It provides generally 4 types of components for blue print representation of database.



Student teacher ER-diagram



② Object oriented model

→ The object oriented model contain all the attributes and method to be represented by using alogrithm procedure.

→ The object oriented model is based on collection of object and it also describe the

relationship between objects that means the object oriented model providing communication mechanism to different objects.

→ The message to be transfer to the ~~other~~ other object with the help of method.

Ex :- Student ⟶ Entity

Attributes ⟶ S-name, S-roll no, S-add

Method ⟶ S-result, (S-roll no, S-id)

Teacher ⟶ Entity

Attributes ⟶ T-name, T-add, S-roll no

Method ⟶ T-id.

③ ~~Sensen~~
Semanting model

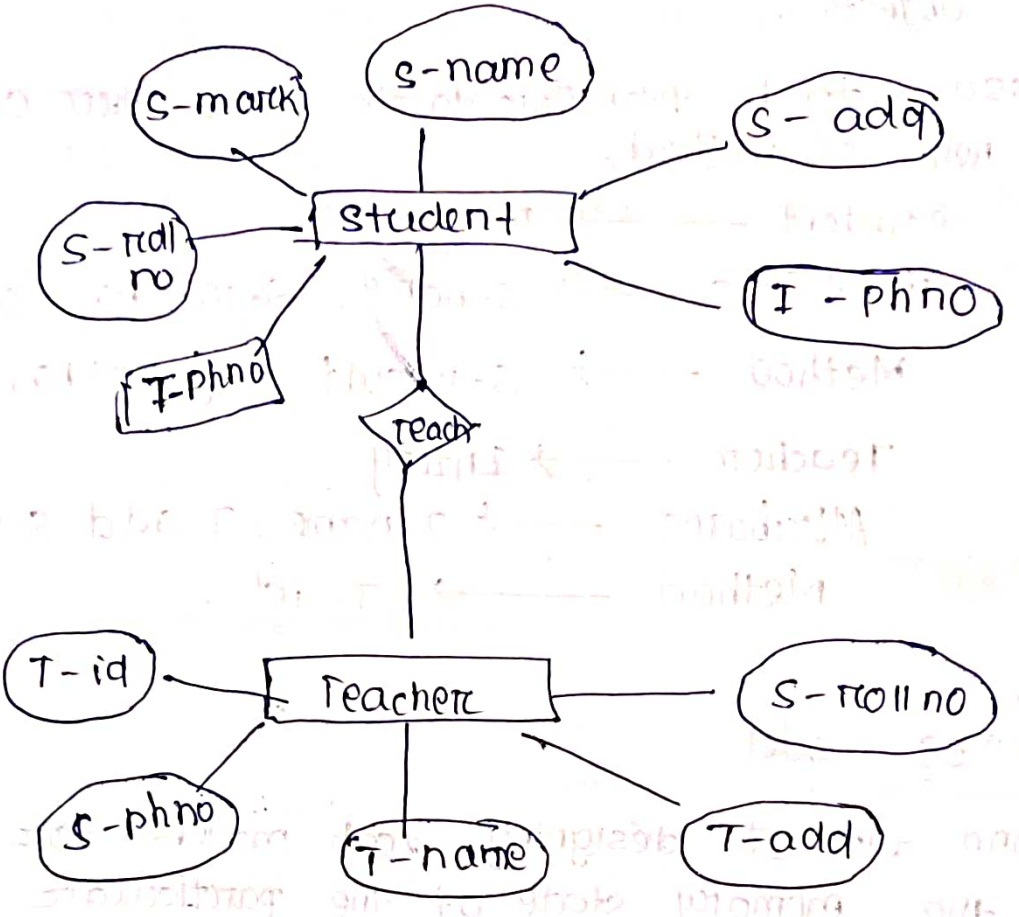
It is one type of designing tools models for calculating the memory state of the particular database.

→ Simanting refers to some rules and regulating to provided when the designed phase to be initiated.

→ It is the main designing tool for database creation all the designing to be represented on the tree structure format.

→ In this case all the attributes this relationship and also it logical data structure to be defined.

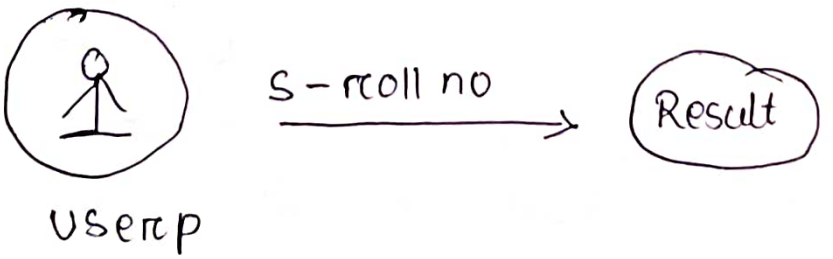
Tree



④ Functional data model

→ Functional data model only provides some specific rules for representing the main objective of the database that mean in this case the detail internal structure can't be modelated only methods to be describe.

→ All the data models to be represented by using interaction structure of the UML certified modulated language.



② Record based model

The record based model fetches all the schema and subschema data and also sometimes fetches the data and als value from the existing database.

→ In this data model considering the logical structure of a database.

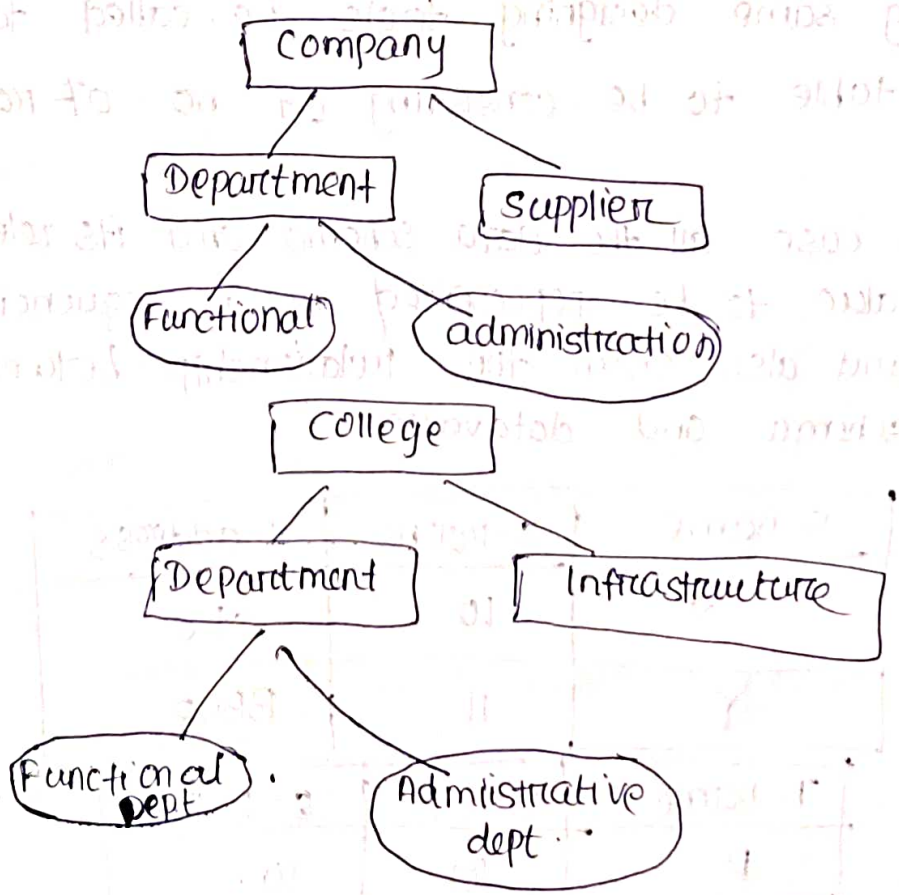
→ It again broadly classified into 3 types

① Hierarchical data model
② Network data model
③ Relational data model

① Hierarchial data model

In this case the entities are represented in hierarchial fashion that reason are parent node is given and other entities derive from the given parent class.

→ In hierarchial model only show up about the information and its relationship.



② Network data model :-

→ This is an extension of hierachial model.
→ In this model data is organised more like a graph and are allowed to have more than the parent node.

→ In this database model data is more related as more relationship established in the database model also as the data is more related.

→ Hence accesing the data is also easier Fast.

→ The database model was used to map many to many data relationship.

→ This was the most widely used database model before relational model was introduce.

③ Relational model

In relational model all the data to be represented by using same designing tools. i.e. called table.

→ that table to be consisting of no - of rows and columns.

→ In this case all the data schema and its relevant data value to be represented in a sequencial was and also show the relationship between the data schema and datavalue.

Ex :-

S-name	S-roll no	S-address
X	10	ctc
Y	11	BBSR

T-name	T-id	S-rollno
P	101	10
Q	102	11

Physical based data model :-

→ The physical model only show up the internal logical information by using some designing tools.

→ This tools indicates which type of data the user uploaded / downloaded and also show up the memory states.

Database system structure and architecture

① Entity relationship model :-

→ The ER-model is based on perception of real life things and data value about the object.

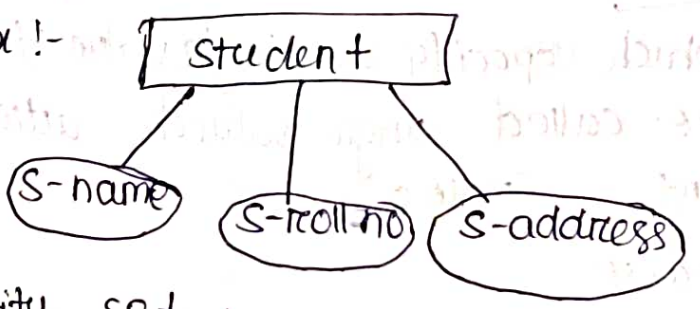
→ The ER model consisting of 4- components such as

1 → Entity
2 → Attributes
3 → Key attribute
4 → Mapping relationship

① Entity :-

→ An entity is a thing or object in the real life world. i.e. distinguishble from another object.

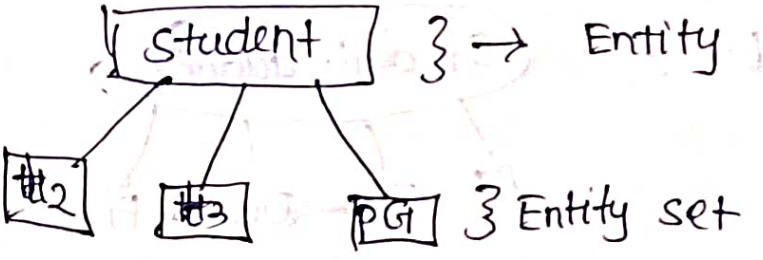
Ex :-



② Entity set :-

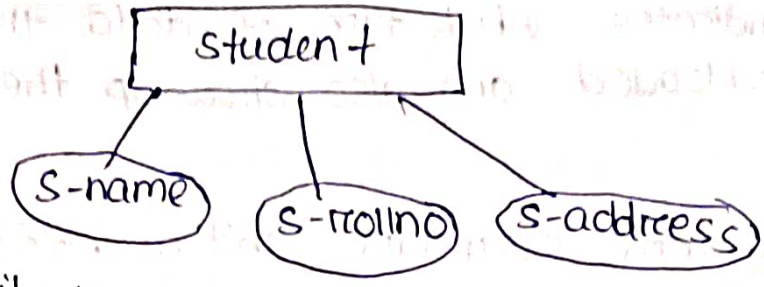
→ An entity set is the collection of entities of similar type.

Ex :-



③ Attributes

→ The keyword attributes defines same properties of the given entity for identification.



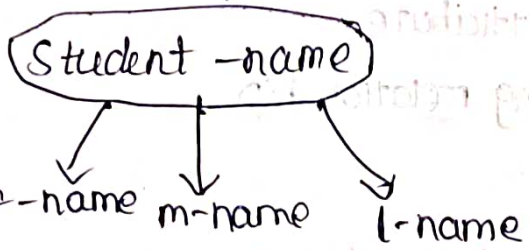
Types of attributes

① simple attribute

→ Those attribute value can't be subdivided is called simple attribute.

② composite attribute

→ Those attribute value can be subdivided into multiple path i.e. called composite attribute.

Ex :- 

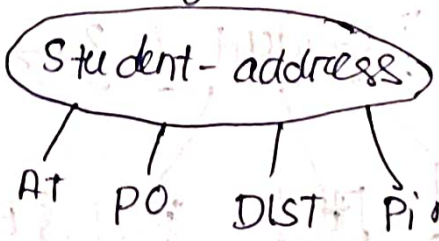
③ Single valued attribute

→ Those attribute which specify a single value for a particular entity i.e. called single valued attributes.

Ex :- student — rollno

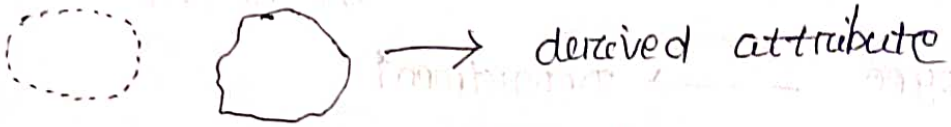
④ Multi valued attribute

→ Those attribute which specify multi values for a particulare entity i.e. called multi value entity.

Ex :- 

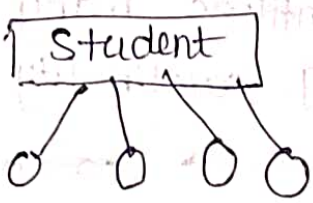
(5) Derived attributes

The value of derive attributes to be created from existing from attribute in the given entity.



(6) Null attribute

If the programmer can't assign any value or attribute on a given entity that times it is called a Null attributes.



(4) Mapping Relationship

Relationship

→ A relationship is an association among several relationship to different entities

→ The relationship to be carried out all database to be consisting information of the every database key. which is uniquely identify. that means any one database needs to create an association to another database, that time the 1st database must be hold primary key attribute of the 2nd data base.

Mapping cardinality :-

The mapping cardinality to be used for view of efficient relationship between one database to another database.

→ so that there are 4 cardinality view of

→ one to one
→ one to many
→ Many to one
→ Many to many

One-to-one relationship

In this case one data base table to be linked another database table or relation each attribute one to one-maping inciates.

-formal that times one-to-one

Ex !- employee ⟶ Department

Department ⟶ HOD

One-to-many relationship

In this case if one database table entity to linked another ~~and~~ database table entities many times of that time one to many maping cardinality intiated.

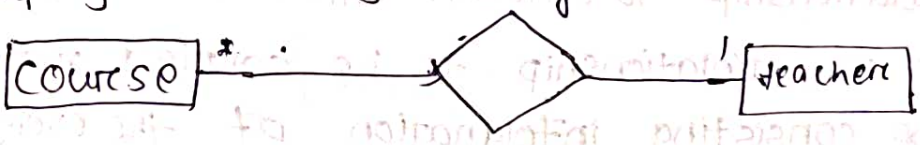
Ex !- teacher !⟶ ∞ course

Customer !⟶ ∞ Product

Many to one-relationship

In this case if the 1st database entity set to be linked with another database entity, that times than many to one maping cardinality arranges.

Ex !-

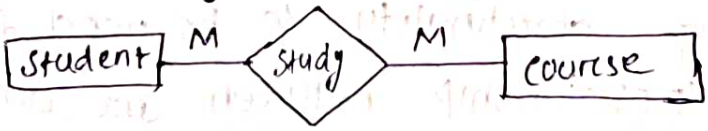


Student ∞ ! teacher

Many to many relationship

In this case if the 1st database entity set to linked with another database entity set i.e. called time of many to many mapping cardinality

Ex !-



Key attribute :-

→ The key attribute to be used for identifying the particular Database field so that E-coded (Entity coded) provides different types of entity elements.

→ Such as

- → super key
- → Candidate key
- → Primary key
- → Secondary key
- → Foreign key

① Super key :-

→ If the database record to be identify by consisting of one or more attributes for uniquely identification that key is called super key.

Ex :- Database ⟶ student

attribute ⟶ roll no, address, name, mark

superkey (s roll no, s-add)

(s-roll no, s-name)

(s-add, s-name)

② candidate key :-

It is minimal of superkey a candidate key is an attribute or set attribute that uniquely identify the record.

Ex :- candidate key (s-roll, s-add, s-name)

③ Primary key :-

It is a key to be intiated on combination of null and require key information that means all the field of primary key elements are not creating duplicate value.

Ex!- Primary key (CS-roll no)

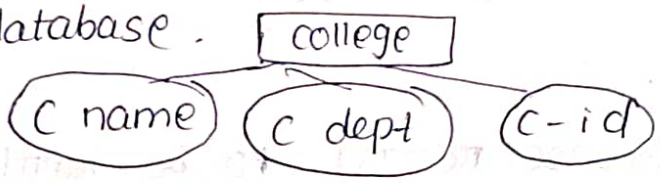
(4) Secondary key !-

→ The secondary key to be used when any weak ~~easity~~ entity set to create relationship to another entities.

→ The secondary key cannot to be candidate key.

→ Always the database user to be checked the secondary key depends upon other sub schema which is exist on database.

Ex!-



C dept - secondary key

(5) Foreign key !-

→ If any the foreign key to be initiated when more than one database created.

→ The foreign key terms appears when any one database consisting of a key field which is primary of the another database.

Ex!- Student - S-name, S-rollno, S-add, S-mark (Primary key)

Teacher - T-name, T-id, S-rollno (Primary key)

S-roll no - Foreign key

Unit-2

Database design theory & Normalization

Database design theory

-> Each relation schema consist of a no. of attributes & the relational database schema consist of a number of relation schema.

-> we have assumed that attribute are grouped to form a relation schema by using common sense of a database designer are by mapping a database schema design from a conceptual deeta model.

Such as :- ER or EER model

-> These model makes the designer identify entity types and relationship types and their respective attributes which leads to natural & logical grouping of the attributes & relation.

-> There are two ~~logical~~ label at which we can discuss the goodness of relation schemas.

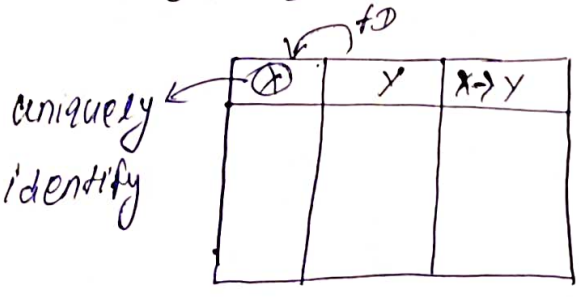
-> The logical / conceptual level, how useres interrupt the relation schema & mining of their attributes.

-> The implementation or physical storage level how the in a tuples base relation are stored & updated.

functional dependencies :-

-> A functional dependancy A->B a relation holds if two tuples having same value for attribute B.

-> It is denoted by A->B where A is called fully dependant on B.



→ It is a set of constraints bet'n two attributes in a relation.

Emp-id	Emp-name	Surname
S_1	Smith	G
S_2	Punam	P
S_3	Smith	M

Types of functional dependencies

① Trival functional dependencies
② Non-Trival functional dependencies.

① Trival functional dependencies

→ A→B has trival functional dependencies of B is a subset of A.

→ The following dependencies are also trival like

A→A , B→B

Ex→ Consider a table with two columns emp_id & emp_name.

→ Emp-id is a trival functional dependencies as employee_id is a subset of (emp_id, Emp_name).

Emp_id → Emp_id

Emp_name → emp_name

Emp_id , Emp_name → Emp_id

Emp_id	E-name	Dept
101	A	marketing
102	B	Delivery
103	C	Return

(ii) NON - trivial functional dependencies :-

-> A->B is a non-trivial functional dependencies if B is not a subset of A.

-> When A intersection B is null then A determines B is called as complete non-trivial functional dependencies.

$$\frac{\text{Emp-id , Emp-name}}{A} \rightarrow \frac{\text{Dept}}{B}$$

-> A->B is a non-trivial functional dependencies

Emp-id	emp-name	dept
101	A	marketing
102	B	delivery
103	C	Return

Inference rules :-

(i) Reflexivity :- If $x \supseteq y$ then $x \rightarrow y$

(ii) Augementation :- If $x \rightarrow y$, then $xz \rightarrow yz$ for any z

(iii) Transitivity :- If $x \rightarrow y$ and $y \rightarrow z$ then $x \rightarrow z$

In addition to above axioms some additional rules for computing closure set of functional dependency are as follows.

(iv) Union :- If $x \rightarrow y$ and $x \rightarrow z$ then $x \rightarrow yz$

(v) Decomposition :- If $x \rightarrow yz$, then $x \rightarrow y$ and $x \rightarrow z$

I.m.p / Normalization :-

-> The same piece of data is held in two separate place in small table.

-> Normalization divides the large table into smaller & ones then using relationship.

-> The normal forem is used to reduce reductancy from the database table.

Reductant data	Normalization
-> It can be determine from other data in the database.	-> Aims to reduce data reductancy.
-> leads to various problem insert anomalies, delete anomalies, update anomalies	-> Reductancy is expressed in term of dependancy.
	-> Norma forems are defined that do not have certain type of dependancy.

① Insert anomalies

-> Attribute can't insert with the presence of other attribute.

-> we tried to insert data in record that data exust at all.

② Delete anomalies :-

-> Exist when certain delete attributes are lost because of deletion of one attribute

:- delete from std=2;

③ Update anomalies :-

-> Partial update because of data inconsistancy.

:- Update student set std names = Amrit

where std = 4

Types of Normalization :-

① 1Nf -> first Normal form (atomic value)
② 2Nf -> second Normal form
③ 3Nf -> third Normal form
④ BCNf -> Boyce codd's Normal form
⑤ 4Nf -> fourth Normal form
⑥ 5Nf -> fifth Normal form

① 1Nf (first Normal form)

-> A relation is in first normal form if it contains an atomic value.

-> It eliminates repeated groups.

-> The only attribute values allowed by 1Nf are single atomic values.

Name	Roll no.	course
A	1	C/C++
B	2	Java
C	3	DBMS

Name	Roll no.	course
A	1	C
B	2	Java
C	3	DBMS
D	4	C++

② 2Nf (second Normal form)

-> A relation will be 2nd Nf if it is in 1st Nf and all Nonkey attribute are fully functionally depended on the primary key.

-> Eliminates partial functional dependency.

-> Partial dependency means that a non prime attribute is functionally dependent on part of a candidate key.

Prime attribute :- An attribute, which is a part of the candidate-key is known as Prime attribute. ~~Non-Prime attribute~~

Non prime attribute :- An attribute, which is not a part of the candidate key is known as Non-prime attribute.

customer ID	store ID	location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Baleswore
4	3	Mumbai

customer ID	store ID
1	1
1	3
2	1
3	2
4	3

Store id	Location
1	Delhi
2	Bangole
3	mumbai

Primary key- customer i'D
Store i'D

Non key attribute :-
location

3rd Nf (3rd Normal form)

→ A relation will be no transitive dependancy exist.

→ Transitive dependancy means indirect relation and function dependancy.

→ Eliminates transitive dependancy.

A → B

B → C

Indirect A → C Transitive dependancy occures so it is Eliminates.

function dependancy

④ Boyce codd's Normal form (BCNF)

→ Boyce codd's Normal form is a hyhere version of the third normal form.

→ This from deals with certain type of anamoly that is not handled by 3NF.

→ A 3NF table which does not have multiple overlapping candidate key is said to be in BCNF.

Note :- L.H.S of each functionaly dependecies should be candidate key or super key.

S roll no.	S name	voter ID	Age
1	Ravi	K038	20
2	vishnu	Z127	21
3	John	K706	23
4	mona	T135	21
5	.Seba	L305	29

valid function dependencies

S roll no → name

S roll no → Age

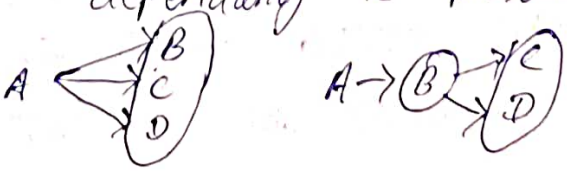
voter ID → name

voter ID → Age

⑤ 4NF (fourth Normal form) :-

→ A relation will be BCNF it is in 4NF from & has no multivalued dependancy.

→ Eleminate multivalued dependancy if there are 3 attribute atleast in a relation & two are dependancy and then multivalued dependancy is present.



⑥ 5NF (fifth Normal form) :-

→ A relation is in 5th Normal from is also known as Project join normal form.

→ It is in 4NF doesn't contain any joint dependancy joining should be loss less decompositions.

→ Its eliminate joint dependancy.

Joind dependancy :-

→ Let R be a relation schema & $R_1, R_2, \ldots, R_n$ be the decomposition of $(R_1, R_2, \ldots, R_n)$ if and only if.

$R[A,B,C,D]$ → original table

$R[A,B]$ $R_2[C,D]$

Joind

$R \bowtie R_1 = [A,B,C,D]$

R

faculty	subject	commitee
MUNU	DBMS	Placemnt
MUNU	Java	Placement
MUNU	C	Placement
MUNU	DBMS	Scholarship
MUNU	Java	Scholarship
MUNU	C	Scholarship

R_1

faculty	subject
MUNU.	DBMS.
MUNU	Java
MUNU	C

R_2

subject	committees
DBMS	Placement
C	Placement
Java	Placement
DBMS	Scholarship
C	Scholarship
Java	Scholarship

faculty	subject	committees
MUNU	DBMS	Placement
MUNU	DBMS	Scholarship
MUNU	Java	Placement
MUNU	Java	Scholarship
MUNU	C	Placement
MUNU	C	Scholarship

$R_1 + R_2$ = original table → 5NF

→ The given table is not in 4NF & 5NF 1st we convert it in 4NF with converting in two sub table

→ To convert it in 5NF we join both table & tables if it gives the result same as original table then it is 5NF otherwise its not 5NF

Multivalued dependancy

→ fore a dependancy, if fore a single value of A, multiple values of B exists, then the table may have multi-values dependency.

→ Also, a table should have atleast 3 colaums fore it to have a multi-valued dependancy.

→ The multivalued dependancy is denoted by →→

Ex:-

→ fore a relation R(A,B,C), if there is a multivalued dependancy between A & B, then B & C should be independent of each other.

→ If all these conditions are true fore any relation (table), it is said to have multivalued dependancy.

S-id	course	skill
1	C c++	English German
2	Java	English french

S-id →→ course
S-id →→ skill

S-id	course	skill
1	C	English
1	c++	German
1	C	German
1	c++	English
2	Java	English
2	Java	french

Unit-3 DBMS 23.12.21

Relational Model :-

- Relational model represents the database as a collection of relations.
- Each relation resembles a table of values or, to some extent, a flat file of records.
- It is called a flat file because each record has a simple, linear or flat structure.
- The relational data model was first introduced by Ted Codd of IBM Research in 1970 in a classic paper and it attracted immediate attention due to its simplicity and mathematical foundation.

When a rel^n is thought of as a table values, each row in the table represents collection of related data values. And the row is called a tuple.

A column header is called an attribute and the table is called a relation.

The data type describing the types of values that can appear in each column is represented by a domain of possible values.

Domain represented as $dom(A_i)$ → Attribute

e.g. Student

Name	Roll-no	Mark	Add
Ajit	7	232	etc
Bikash	8	451	Rkl
Nilu	6	154	BBSR

→ Tuple

↓ Domain

The degree of a relation is the number of attributes present in a relation.

Characteristics of Relations :-

① Ordering of Tuples in a Relation -

→ A relation is defined as a set of tuples. Tuples in a rel^n don't have any particular order.
→ A relation is not sensitive to the ordering of tuples.
→ However, in a file, records are physically stored on disk, so there always is an order among the records. This ordering indicates 1st, 2nd, ith and last in the file.
→ Similarly, when we display a rel^n as a table, the rows are displayed in a certain order.
→ Many tuple orders can be specified on the same rel^n.
e.g. tuples in the student rel^n could be ordered by values of Name, SSN, Age or other attribute.

② Ordering of values within a Tuple -

→ According to the preceding definition of rel^n, an n-tuple is an ordered list of n-values; so the ordering of values in a tuple and hence of attribute in a rel^n schema is important.
→ However, at a more abstract level, the order of attributes and their values is not important as long as the correspondence between attributes and values is maintained.

③ Values and NULLs in the tuples -

→ Each value in a tuple is an atomic value i.e. it is not divisible into components within the framework of the basic relational model.
→ Hence, composite and multivalued attributes are not allowed.
→ This model is sometimes called flat-related

→ An important concept is that of NULL values, which are used to represent the values of attributes that may be unknown or may not apply to a tuple. A special value called NULL, is used in these cases.

For e.g.

Some student tuples have NULL for their office phones.

27.12.21

Relational Model Constraints :-

→ There are generally many restrictions or constraints on the actual values in a database state.

→ Every rel^n has some cond^s that must hold for it to be a valid rel^n known as Relational Constraints.

→ There are three main constraints.

1. Key constraints
2. Domain Constraints
3. Integrity constraints

Domain Constraints :-

It specify the atomic values on attribute for each entity otherwise it is violated.

e.g. For STUDENT table

<'Anil', 25, 539, etc> is valid but

STUDENT

Name	Roll-no	Mark	Addr

<'Anil', 'Anil', 25, 539, etc> not valid because Name attribute value is repeated it is not atomic.

Key Constraints :-

→ Key Attribute contain unique value for each record in a relation and it can't duplicate.

→ There is no two tuples in a rel^n having same comb^n of values i.e. $t_1[S] \neq t_2[S]$

-> The primary key of a rel^n should be not NULL.
-> Primary key may be a single attribute or comb^n of more than one attribute.

e.g. In STUDENT table, std is used as key attribute because it is unique for each students.

Types of key Constraints :-

① Primary key :- It is the first key which is used to identify one and only one instance of an entity uniquely.

-> An entity can contain multiple keys ~~as we saw in~~ And the key which is most suitable from those lists become a primary key.
-> In EMPLOYEE table id, license-number and passport number can be primary key since they are unique.
-> For each entity, selection of the primary key is based on requirement and developers.
-> There can be only one primary key in a table.
-> Primary key can't have the same values repeating for any row.
-> The primary key put on a column or set of column not allow them to have any null values or any duplicates.
-> Any value in the primary key can't be changed by any foreign keys.

② Super key :-

Super key is the set of all the keys

① Domain Constraint :-

Domain Constraints can be defined as the definition of a valid set of values for an attribute.

→ The Datatype of Domain includes string, characters, integer, time, date, currency etc.

→ The value of the attribute must be available in the corresponding domain.

e.g.

id	Name	Semester	Age	
1000	Ram	1st	17	(valid)
1004	Mira	6th	A	(invalid bcz Age is integer type)

② Integrity Constraint :-

Integrity Constraints are a set of rules. It is used to maintain the quality of information.

It is of 2 types.

① Entity Integrity Constraint

② Referential Integrity Constraint.

Entity Integrity Constraint –

The entity integrity constraint states that primary key value can't be null.

→ This is because the primary key value is used to identify individual rows in rel" and if the primary key has null value, then we can't identify those rows.

→ A table can contain null value other than a primary key field.

e.g.

emp-id	emp-Name	Salary	
123	Ram	30,000	(Valid)
NULL	Mira	27,000	(Invalid)

Referential Integrity Constraint —

→ A Referential integrity constraint is specified bet^n two tables.

→ In the referential integrity constraints if a foreign key in table-1 refers to the primary key of table-2, then every value of the foreign key in table-1 must be null or be available in table-2.

Table-1

emp-id	Name	Age	D-no
1	Ram	23	11
2	Hari	26	13
3	Mira	40	24

(foreign key: D-no)

Rel^{n}ship

Table-2

D-no	D-loc
11	Mumbai
13	Delhi

(Primary key: D-no)

→ Not allowed as D-no 24 is not defined as a primary key of table-2 and in table-1 D-no is a foreign key defined.

③ Key Constraints :-

Keys are the entity set that is used to identify an entity within its entity set uniquely.

→ An entity set can have multiple keys but all of which one key will be the primary key.

→ Key attribute contain unique value for each record in a rel^n and it can't be duplicated.

Different Types of Keys :-

① Primary key -

→ Primary key may be a single attribute or composite of more than one attribute.

→ There is no two tuples in a rel^n having same comb^n of values i.e. tuple $t_1 \neq$ tuple t_2

i.e. $t_1[S] \neq t_2[S_2]$

→ The primary key of a rel? should be not NULL.
→ The primary key is the most important key in the database. There can be only one primary key in a table.
→ The primary key contains unique values.

e.g.
In employee-table employee-Id is a primary key.

② Super key -
→ A super key is a set of one or more keys that are used to identify data or records uniquely in a database table.
→ It includes only those fields that have unique values.

e.g.
emp-designation, employee-Id

③ Alternate key -
→ The alternate key can be an alternative or a candidate for the primary key when needed but it is not a primary key.
→ An alternate key is a function of all candidate keys except the primary key.

e.g.
Phone-Number.

→ If a table has more than one candidate key then one of them is a primary key and rest of all are alternate keys.

④ Candidate key -
A candidate key is a set of one or multiple columns in a database table.
→ It can identify a record uniquely just like a primary key. These are other unique columns that can become a primary key.
→ There can be any number of candidate keys that

can be used in place of the primary key if required

e.g.

⑤ Composite key -

This is a combination of one or more columns that can uniquely identify the records in a table.

-> The composite key can be a combination of prim and candidate keys.

e.g.

emp-Id and emp-panNumber

⑥ Foreign key -

-> A Foreign key can be a common key in two database tables.

-> Using a foreign key we can identify records fro multiple tables.

-> It accepts duplicate values as well as null value

-> e.g.

A Company table present a emp-Id column and same column present in employee table.

03.01.22.

Update Operation and Constraints violation : –

① Insert Operation –

→ It is used to insert a single or multiple records in a table.

→ There are 2 method for data insertion.

① No need to specify the column name, user need only their values.

Syntax – INSERT INTO tablename
VALUES (value1, value2 ....);

② Specify both column name and values.

Syntax – INSERT INTO tablename (column1, ...)
VALUES (value1, value2 .....);

● violation –

Student

Name	Roll	Mark	Dno
A	1	50	100
B	2	40	101
C	3	45	100

Dept

Dnum	Dname
100	BCA
101	ITM

INSERT INTO Student

① ('Ram Ch Jena', 2, 50, 101) (Domain constraints violated)

② ('Ram', 1, 50, 101) (Key " ")

③ ('Hari, NULL, 55, 100) (Entity Integrity " ")

④ ('Ram, 1, 50, 103) (Referential Integrity " ")

② UPDATE Operation –

→ Update operation is used to change the values of one or more attributes in a tuple of some rel^n R.

→ Update operation may violate primary key constraints and referential Integrity constraints.

e.g.

① Update Dno of student having Roll 1 to 103
(violate referential integrity constraints)

② Update Roll of student having Roll 2 to 1
(key constraints violated)

③ Update Roll of student having Roll 2 to NULL
(Entity Integrity constraints)

Delete Operation –

-> Delete operation used to delete the tuples.

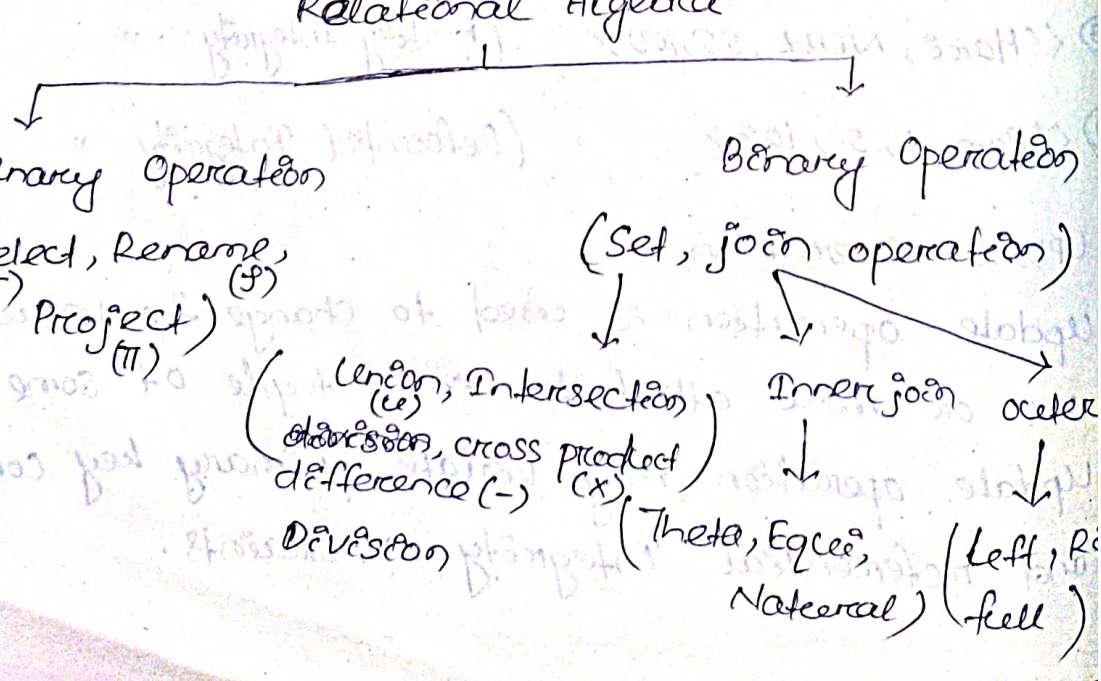
-> Delete may violate only referential integrity.

-> If the primary key values of the tuple being deleted is referenced from other tuples in the database.

e.g.

Delete the tuple from Dept where Dnum = 101;
(Referential integrity constraints)

The Relational Algebra and Relational Calculus :-



Relational Algebra –

→ Relational algebra is a procedural query language, which takes instances of rel's as input and yields instances of rel's as output.

→ It uses operators to perform queries.

→ Relational algebra is performed recursively on a rel^n and intermediate results are also considered relations.

* Unary Relational Operations –
  1. SELECT (σ)
  2. PROJECT (π)
  3. RENAME (ρ)

* Binary Relational Operations –
  1. UNION (∪)
  2. INTERSECTION
  3. DIFFERENCE (−)
  4. CARTESIAN PRODUCT (×)

* Additional Relational Operations –
  1. JOIN
  2. DIVISION
  3. OUTER JOINS
  4. OUTER UNION
  5. AGGREGATE FUNCTIONS (SUM, COUNT, AVG, MIN, MAX)

Employee

FNAME	LNAME	SSN	SALARY	DNO
A	B	123	3000	1
AB	C	456	2052	2
HB	KJ	152	6251	3
SS	kk	789	5645	1
PP	MM	526	7945	2
DD	LL	452	2584	1

Dept

Dname	Dnum	MGRSSN
CSC	1	123
BCA	2	526
BBA	3	152

04.01.22

Select Operation :- (σ)

→ The SELECT operation is used to select a subset of the tuples from a rel^n based on a selection cond^n.

→ Selection cond^n acts as a filter it keeps the tuples which are satisfying the cond^n and discarded the other tuples.

→ The selection visualize as horizontal partition of tuples into 2 sets.

→ Syntax - σ (R)

<sel^n cond^n>

↓

<attribute name> <comparison operator> <constraints>

↓

(<, >, <=, >=, =, !=)

Here σ → selection predicate

R → Relation

Condition is propositional logic formula which may use connectors like and, or, and not.

e.g. Select the EMPLOYEE tuples whose department number is 2;

σ (EMPLOYEE)

DNO = 2

Properties :-

→ The SELECT operation produces a rel^n s that has the same schema as R.

→ SELECT (σ) is commutative.

σ (R) = σ (R)

<cond^{n}1> <cond^{n}2> <cond^{n}2> <cond^{n}1>

e.g. $\left(\sigma\ (EMP)\right)_{(salary >= 300)}(DNO = 2) = \left(\sigma\ (EMP)\right)_{(DNO = 2)}(salary >= 300)$

→ A cascade of SELECT operations may be replaced by a single selection with a conjunction (AND, OR) of all conds.

→ The number of tuples in the result of a SELECT is less than or equal to the number of tuples in rel R.

Project Operation :- (Π)

→ Project Operation is used to project only a certain set of attributes of a rel.

→ It will only project or show the columns asked for, and also remove duplicate data from the columns.

→ General form $\Pi(R)$, where Π = <attribute list> R =

→ Projection visualize as column partition of tuple.

e.g. Select Fname, Lname, Salary of the EMP;

Π (EMP)
(Fname, Lname, Salary)

Properties –

→ The number of tuples in result of projection is always less than or equal to number of tuples in R.

→ If the list of attributes includes a key of R, then the number of tuples in the result of PROJECT is equal to the number of tuples in R.

→ PROJECT is not commutative.

$$\left(\Pi_{\text{<list1>}}(\text{EMP})\right)_{\text{<list2>}} \neq \left(\Pi_{\text{<list2>}}(\text{EMP})\right)_{\text{<list1>}}$$

RENAME (ρ) :-

Several relational algebra operations can apply one after the other.

① Either can create the operations as a single relational algebra expression or by nesting the operations;

② One operation can apply at a time and create intermediate result relations and must give names to the relations that hold the intermediate results.

eg. Retrieve the Fname, Lname and Salary of all employees who work in Dno=2;

$$R \leftarrow \pi_{Fname, Salary Lname,} \left(\begin{matrix} \sigma (EMP) \\ Dno=2 \end{matrix} \right)$$

→ In some cases user want to rename the attributes of a rel^n or the rel^n name or both i.e.

- when a query requires multiple operations.
- Necessary in some cases (JOIN operation)

→ e.g. $\rho_S(B1, B2, \ldots Bn)(R)$ changes the rel^n name to S and col^n attribute names to B1, B2, ... Bn

e.g. $R(Fname, Lname, sal) \leftarrow \pi_{Fname, Lname, Salary} \left(\begin{matrix} \sigma (EMP) \\ Dno=2 \end{matrix} \right)$

Relational Algebra Operations from Set Theory :-

UNION :- (∪)

→ The union operation produces the tuples that are in either R_1 or R_2 or both.

→ It is denoted by $R_1 \cup R_2$

→ Duplicate tuples are eliminated.

→ R_1 and R_2 must have same no. of attributes.

→ Each pair of corresponding attributes must be type compatible.

e.g. Retrieve the SSN of all employees who either work in Dno=1 or salary >4000

$$R_1 \leftarrow \prod_{SSN} (\sigma_{Dno=1} (EMP))$$

$$R_2 \leftarrow \prod_{SSN} (\sigma_{salary>4000} (EMP))$$

$$R_3 \leftarrow R_1 \cup R_2$$

output

SSN
456
526
152
789
526

INTERSECTION (∩) :

→ The intersection operation produces the tuples that are in both R_1 and R_2.

→ The attribute names in the result will be same as attribute names in R_1.

→ It is denoted by $R_1 \cap R_2$.

→ The two operand rel's R_1 & R_2 must be "type compatible".

e.g. Retrive the SSN of all employees who works in Dno=1 and salary >4000

$$R_4 \leftarrow R_1 \cap R_2$$

output

SSN
526

DIFFERENCE (−) :-

→ It denoted by $R_1 - R_2$

→ The result of $R_1 - R_2$ is a rel^n that includes all the tuples that are in R_1 but not in R_2.

→ The attribute names in the result will be same as the attribute names in R_1.

→ The two operand rel's R_1 and R_2 must be "type compatible".

e.g. $R_3 \leftarrow R_1 - R_2$

output

SSN
152
789

5) Division (÷) :-

→ Division is a binary operation written as $R \div S$.

→ Division is not implemented directly in SQL.

→ The result consists of the restrictions of tuples in R to the attribute names unique to R i.e. in the header of R but not in the header of S, for which it holds that all their combinations with tuples in S are present in R.

e.g.

Completed	
Sted	Task
Fred	Database1
Fred	Database2

DBProject
Task
Database1
Database2

Completed ÷ DBProject
Sted
Fred
Sarah

05.01.22

CARTESIAN (OR CROSS) PRODUCT operation:-

→ This operation is used to combine tuples from two relations in a combinatorial fashion.

→ Denoted by R x S.

→ Result of a rel^n Q with degree n+m attributes

→ The resulting relation state has one tuple for each combination of tuples - one from R and one from S.

→ Hence, if R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then R x S will have $n_R * n_S$ tuples.

→ The two operands do not have to be "type compatible".

R

FNAME	LNAME	SALARY
AB	C	2052
PP	MM	7945

S

FNAME	DNUM
AB	1
PP	2

R x S

FNAME	LNAME	SALARY	FNAME	DNUM
AB	C	2052	AB	1
AB	C	2052	PP	2
PP	MM	7945	AB	1
PP	MM	7945	PP	2

Join Operation (⋈) :-

→ The sequence of cartesian product followed by select is used quite commonly to identify and select related tuples from two relations.

→ A special operation, called join combines this sequence into a single operation.

→ This operation is very important for any relational database with more than a single rel^n, because it allows us combine related tuples from various relations.

→ The general form $R \bowtie S$
<join cond^n>

Types of Join Operation —

① Inner Joins

② Outer Joins

Inner Joins —

① Theta join —

→ Theta join combines tuples from different rel provided they satisfy the theta cond^n.

→ Denoted by θ.

→ Such that the attributes don't have anything in common, that is $R \cap S = \phi$

→ Theta join can use all kinds of comparison operators (>, <, >=, <=, =, != etc)

Model	Carprice
CarA	20,000
CarB	40,000
CarC	50,000

Model	Boatprice
Boat1	10,000
Boat2	40,000
Boat3	60,000

Car ⋈ Boat
(Carprice >= Boatprice)

CarA	20,000	Boat1	10,000
CarB	40,000	Boat1	10,000
CarB	40,000	Boat2	40,000
CarC	50,000	Boat1	10,000
CarC	50,000	Boat2	40,000

Equijoin :-

→ When Theta join uses only equality comparison operator, it is said to be equijoin.

e.g. Car ⋈ Boat
(carprice = Boatprice)

CarB	40,000	Boat2	40,000

Natural Join :-

→ It is denoted by *

→ Natural join requires that the two join attributes, or each pair of corresponding join attributes, have the same name in both rel's.

→ If this is not the case, a renaming operation is applied first.

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

Dept

DeptName	Manager
Finance	George
Sales	Harriet
Production	Charles

Employee * Dept

Name	EmpId	DeptName	Manager
Harry	3415	Finance	George
Sally	2241	Sales	Harriet
George	3401	Finance	George
Harriet	2202	Sales	Harriet

Outer joins :-

→ It is the extension of join operation and it deal with missing information.

→ In NATURAL ~~JOIN~~ JOIN and EQUIJOIN tuples formed by combining matching tuples in the two operands and without a matching tuple are eliminated from the join result.

→ Tuples with null in the join attributes are also eliminated.

Left outer join :-

It keeps every tuple in the left rel^n R in R ⟕ S if no matching found in S, then the attributes of S in the join result are filled with NULL values.

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales

Dept

DeptName	Manager
Sales	Harriet
Production	charles

Employee ⟕ Dept

Name	EmpId	DeptName	Manager
Harry	3415	Finance	NULL
Sally	2241	Sales	Harriet

Right outer join :-

→ Right outer join operation keeps every tuple in the right relation S in the result of R ⟕ S.

→ It also includes the matched values from left table but if there is no matching in both tables, it returns NULL.

→ It is denoted by R ⟖ S.

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales

Dept	
DeptName	Manager
Sales	Harriet
Production	Charles

Employee ⟖ S

Name	EmpId	DeptName	Manager
Sally	2241	Sales	Harriet
w	w	Production	charles

Full outer join :-

-> It is denoted by R ⟗ S.

-> It keeps all tuples in both left and right rel'S when no matching tuples are found, then NULL values created.

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Tim	1123	Executive

DeptName	Manager
Sales	Harriet
Production	Charles

Name	EmpId	DeptName	Manager
Harry	3415	Finance	w
Sally	2241	Sales	Harriet
George	3401	Finance	w
Harriet	2202	Sales	Harriet
Tim	1123	Executive	w

* Cross product with selection operator takes more time and more memory compare to join operation so join operation is more reliable than cross product.

* Relational Algebra :- It tells how to retrieved data.
Relational Calculus tells what to be retrieved.

06.01.22

Relational Calculus :-

→ A relational calculus expression creates a new relation, which is specified in terms of variable that range over rows of stored database rel^n (tuple calculus) and or over columns of stored relations (domain calculus).

→ In this expression, there is no order of operations to specify how to retrieve the query result. It specifies only what information the result should contain.

→ Relational Calculus is a nonprocedural language.

→

Tuple Relational Calculus :-

→ It is based on specifying a number of tuple variables.

→ Each tuple variable usually ranges over a particular database relation, meaning that the variable may take as its value any individual tuple from that relation.

→ A simple tuple relational calculus is of the form

$\{t \mid COND(t)\}$ where t = Tuple variable

$COND(t)$ = Conditional expression

e.g. To find the first and last names of all employees whose salary is above $50,000.

$\{t.FNAME, t.LNAME \mid EMP(t) \text{ AND } t.SALARY > 50000\}$

Projection $\pi(FNAME, LNAME)$ — It specifies that the range rel^n of tuple variable t is EMP. — Selection cond^n

Domain Relational Calculus :-

→ Domain Calculus is equivalent to tuple calculus and to relational algebra.

→ The language QBE (Query-By-Example) that is related to domain calculus was developed almost concurrently to SQL at IBM Research, Yorktown Heights, New York

→ Domain Calculus differs from tuple calculus in the type of variables used in formulas rather than having variables range over tuples, the variables range over single values from domains of attributes.

→ To form a rel^n of degree n for a query result, we must have n of these domain variables one of each attribute.

→ The Domain Calculus is of the form

$$\{x_1, x_2, \ldots, x_n \mid COND(x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+m})\}$$

↓ Domain variables that range over domains (of attributes)

↓ is a cond^n or formula of the domain relational Calculus.

e.g: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

$$\{uv \mid (\exists q)(\exists r)(\exists s)(\exists t)(\exists w)(\exists x)(\exists y)(\exists z)$$
$$EMP(qrstuvwxyz) \text{ and } q = \text{'John'} \text{ and } r = \text{'B'} \text{ and } s = \text{'Smith'})\}$$

* q, r, s, ... z variables and u, v are free.

↓ for BDATE ↓ ADDRESS

In tuple Calculus :-

* Tuple variable is written in lowercase.
* A tuple calculus query is in the form

$$\{t \mid COND(t)\}$$

↓ conditional expression (if, if), cond^n or formula

↓ Name of rel^n ↓ made up of predicate calculus atom.

1. $R(t_i)$ → tuple variable ranges over Rel^n R. ↓ $R(t_i)$
2. $t_i.A_i$ op $t_j.B_j$ → Comparison operator; → attribute of Rel^n S; → Attribute of Rel^n R; ↳ tuple variable of Rel^n R; → variable of Rel^n S
3. $t_i.A$ op C → Constant

eg: Retrieve the salary and address of employee whose fname is John and lname is Smith.

$\{ t.Salary, t.add \mid emp(t) \text{ and } t.fname = John \text{ and } t.lname = Smith \}$

e.g. Display the fname and lname of employee who manages Dept no. 1

$\{ t.fname, t.lname \mid emp(t) \text{ and } (\exists d) dept(d) \text{ and } d.dnum = 1 \text{ and } d.MGR\_SSN = t.SSN \}$

* In Domain Calculus :-

$\{x_1, x_2 \ldots x_n \mid COND(x_1, x_2 \ldots x_{n+1}, x_{n+2}, \ldots x_{n+m})\}$

$x_1, x_2 \ldots$ → domain variable

COND → Cond^n or formula made of wff atoms

1. $R(x_1, x_2, x_3 \ldots x_j)$ → Name of the rel^n; x_j → Domain variable
2. x_i op x_j → op: Comparison operator ($<, >, <=, >=, !=, =$); x_i, x_j: Domain variable
3. x_i op c or c op x_j → x_i: Domain variable; c: Constant

u v w x y
Emp(Ename, SSN, add, DOJ, Dn
Dept(Dnumber, Dname, Dlocati
l m n

e.g. Display the employee information.

$\{uvwxy \mid EMP(uvwxy)\}$

e.g. Display the Ename, address where SSN is 1234

$\{uw \mid (\exists v) emp(uvwxy) \text{ and } v = 1234\}$

e.g. Display the employee address, DOJ, Dname who is working for Dno = 5

$\{wxm \mid (\exists y)(\exists l)(EMP(uvwxy) \text{ and } Dept(lmn) \text{ and } y = 5 \text{ and } y = l\}$

e.g.

Display the fname, lname of employee who manages Dept 1.

① $\Pi_{fname, lname} \left(\sigma_{(EMP \times Dept)\ Dno = 1 \text{ and } MGR\text{-}SSN = SSN} \right)$

② $\Pi_{fname, lname} \left(\sigma_{(EMP \bowtie Dept)\ Dno = 1 \text{ and } MGR\text{-}SSN = SSN} \right)$

17.01.22 Unit-3

SQL :- (structered queery language)

-> It is pronounced as S-Q-L or sometimes as See-Quell.
-> This database language is mainly designed for maintaing the data in relational database management systems.
-> It is a special tool used by data professionals for handling structered data.
-> It is also designed for stream processing in RDSMS.
-> This queery language became the standard of ANSI in the year of 1986 and ISO in the year of 1987.

Why SQL ?

-> SQL is widely used in data science and analytics.
-> The basic use of SQL for data professionals and SQL users is to insert, update and delete the data from the relational database.
-> It also helps to describe the structered data.
-> It allows SQL users to create, drop and manipulate the database and its tables.
-> It also helps in creating the view, stored procedure and functions in the relational database.
-> It allows to define the data and modify that stored data in the relational database.
-> It allows SQL users to set the permissions or constraints on table columns, views and stored procedure.

History of SQL :-

-> The IBM researchers Raymond Boyce and Donald Chamberlin developed the SEQUEL (structered English queery Language) after learning from the paper published by E.F codd.
-> They both developed the SQL at the San Jose

Research laboratory of IBM corporation in 1970.
→ SQL was based on RDBMS.
→ Relational Software Inc, known as Oracle corporation, introduced Oracle V2 in June 1979, which is the first implementation of SQL language.
→ Head-office of SQL is at pune, (Bangalore).

Advantages of SQL :- 20.01.22

1. No programming needed -
→ SQL doesn't require a large number of coding lines for managing the database systems.
→ We can easily access and maintain the database by using SQL. So SQL is user-friendly.
2. High-Speed Query Processing -
→ A large amount of data is accessed quickly and efficiently from the database using SQL.
→ Insertion, deletion and updation operations on data are also performed in less time.
3. Standardized Language -
→ SQL follows a long-established standards of ISO and ANSI, which offer a uniform platform across the globe to all its users.
4. Portability -
→ SQL easily used in desktop computers, laptops, tablets and even smartphones.
→ It also used other applications according to user's requirement.
5. Interactive language -
→ We use this language for communicating with the database because it is a simple query language.
→ This language is also used for receiving the

answers to complex queries in a few seconds.

6. More than one Data view –

→ SQL also helps in making the multiple views of the database structures for the different database users.

Disadvantages of SQL :-

1. Cost – The operation cost of some SQL versions is high. That's why some programmers can't use SQL.
2. Interface is Complex – Interface of structured query language is difficult, which makes it difficult for SQL users to use and manage it.
3. Partial Database control –
   The business rules are hidden. So, the data professionals and users who are using this SQL can't have full database control.

Some SQL Commands :-

The SQL commands help in creating and managing the database. The most common SQL commands are –

① CREATE Command
② UPDATE "
③ DELETE "
④ SELECT "
⑤ DROP "
⑥ INSERT "

CREATE Command – This command helps in creating the new database, new table, table view and other objects of the database.

UPDATE Command – This command helps in updating or changing the stored data in the database.

DELETE Command – This command helps in removing or erasing the saved records from the database table

→ It erase single or multiple tuples from the tables of the database.

SELECT Command - This helps in accessing the single or multiple rows from one or multiple tables of the database. We use this command with WHERE cla

DROP Command - This helps in deleting the entire table, table view and other objects from database.

INSERT Command - This helps in inserting the data or records into the database tables. We can ins the records of single or multiple rows of the tabl

SQL Syntax :-

→ The syntax of SQL is a unique set of rules & guidelines, which is not case-sensitive.

→ Its syntax is defined and maintained by the ISO and ANSI standards.

→ We can write the keywords of SQL in both uppercase and lowercase. But uppercase improves the readability of SQL query.

→ SQL statements or syntax dependent on text lines. We can place a single SQL statement on one or more text lines.

→ We can perform most of action in a database with SQL statements.

→ SQL syntax depends on relational algebra and tup relational calculus.

SQL Statement :-

SQL statements tell the database what operation we want to perform on the structured data an what information we would like to access from the database.

→ The statements are very simple and easy to use and understand.

e.g:

SELECT "column-name" FROM "table-name";

→ Each SQL statement begins with keyword and ends with semicolon (;).

→ Semicolon is used for seperating multiple SQL statements.

Most Important SQL Commands and Statements :-

① Select Statement :-

This SQL statement reads the data from the SQL database and shows it as the output to the database user.

Syntax :

1. SELECT column-name1, column-name2, ..., column-nameN
2. [FROM table-name]
3. [WHERE cond^n]
4. [ORDER BY order-column-name1 [ASC/DESC], ...];

e.g. SELECT emp-ID, salary FROM employee-details WHERE salary=10000 ORDER BY Last-name.

② Update Statement :-

This SQL statement changes or modifies the stored data in the SQL database.

Syntax :

UPDATE table-name SET column-name1 = new-value1, ...
column-nameN = new-valueN WHERE cond^n;

e.g.

UPDATE employee-details SET salary=10000 WHERE emp-ID=10;

③ DELETE Statement :-

This SQL statement deletes the stored data from the SQL database.

Syntax :

DELETE FROM table-name WHERE cond^n;

e.g.

DELETE FROM employee-details WHERE F_Name='Suresh';

④ CREATE TABLE Statement –

This SQL statement creates the new table in SQL database.

Syntax:

```
CREATE TABLE table-name
    ( column-name1 data-type [column1 constraint(s)],
      column-name2 data-type [column2 constraint(s)],
      .
      .
      .
      column-nameN
      PRIMARY KEY );
```

e.g.

```
CREATE TABLE employee-details (
    emp_ID NUMBER (4) NOT NULL,
    F_Name VARCHAR (30),
    Salary Money,
    city VARCHAR (30),
    PRIMARY KEY (emp_ID));
```

⑤ ALTER TABLE Statement –

This statement adds, deletes and modifies the columns of the table in SQL database.

Syntax:

```
ALTER TABLE table-name ADD column-name datatype[(size)];
                       MODIFY
                       (RENAME)
                       DROP COLUMN column-name;
```

⑥ DROP TABLE Statement –

Syntax:

```
DROP TABLE [IF EXISTS] table-name1, table-name2...;
                       table-nameN;
```

e.g.

```
DROP TABLE employee-details;
```

→ This statement deletes the table and structure, views, permissions and triggers associated with that table.

⑦ CREATE DATABASE Statement –
This statement creates the new database on the DBMS.
Syntax :
CREATE DATABASE database-name;
e.g.
CREATE DATABASE Company;

⑧ DROP DATABASE statement –
This statement deletes the existing database with all the data tables and views from the DBMS.
Syntax : DROP DATABASE database-name;
e.g. : DROP DATABASE Company;

⑨ INSERT INTO Statement –
This statement inserts the data or records in the existing table of SQL database.
Syntax : INSERT INTO table-name (column-name1 . . . column-nameN)
VALUES (value 1, value 2 . . . . valueN);
e.g.
INSERT INTO employee-details (emp-ID, F-Name, Salary)
VALUES (101, Akhil, 40000);
For multiple records in a single query –
Syntax :
INSERT INTO table-name (column-name1 . . . column-nameN)
VALUES (value1, . . . valueN), (value1 . . . valueN);
e.g
INSERT INTO employee-details (empID, F-Name, Salary)
VALUES (101, Akhil, 10000), (101, John, 50,000) . . . . . ;

⑩ TRUNCATE TABLE Statement –
This statement deletes all the stored records from the table of the SQL database.
Syntax : TRUNCATE TABLE table-name;
e.g. TRUNCATE TABLE employee-details;

⑪ DESCRIBE statement –
This statement tells something about the specified table or view in the query.
Syntax : DESCRIBE table-name; e.g. DESCRIBE employee-dt

(12) DISTINCT clause -

This statement shows the distinct values from the specified columns of the database table.

-> This statement used with SELECT keyword.

Syntax :

SELECT DISTINCT column_name1, ... FROM table_name;

e.g.

SELECT DISTINCT city, salary FROM employee details;

(13) COMMIT statement -

This statement saves the changes permanently, which are done in the transaction of SQL database.

Syntax : COMMIT

e.g. DELECT FROM employee_detail WHERE salary = 3000;

COMMIT;

(14) ROLLBACK statement -

This statement undo the transactions and operations which are not yet saved to SQL database.

syntax : ROLLBACK

e.g. DELECT FROM employee_details WHERE city = Mumbai;

ROLLBACK;

(15) CREATE INDEX statement -

Create new index in SQL database

Syntax

CREATE INDEX index_name ON table_name (column1 ... columnN);

e.g.

CREATE INDEX idx_F_Name ON employee_details(F_Name);

(16) DROP INDEX statement -

Deletes the existing index of SQL database table.

Syn

DROP INDEX index_name;

27.01.22

SQL DATA TYPES :-

Data types are used to represent the nature of the data that can be stored in the database table.

Data types are ① String Data types
② Numeric Data types
③ Date and time Data types

MYSQL String Data Types -

① CHAR(size) :- It is used to specify a fixed length string that can contain numbers, letters & special letters.
→ Its size can be 0 to 255 characters. Default is 1.

② VARCHAR(size) : specify a variable length string that can contain numbers, letters & special characters.
→ Its size can be from 0 to 65535 characters.

③ BINARY(size) : It equal to CHAR() but stores binary byte strings. Default size is 1

④ VARBINARY(size) : It equal to VARCHAR() but stores binary byte strings. Its size parameter specifies the max^m column length in bytes.

⑤ TEXT(size) : It holds a string that can contain a max^m length of 255 characters.

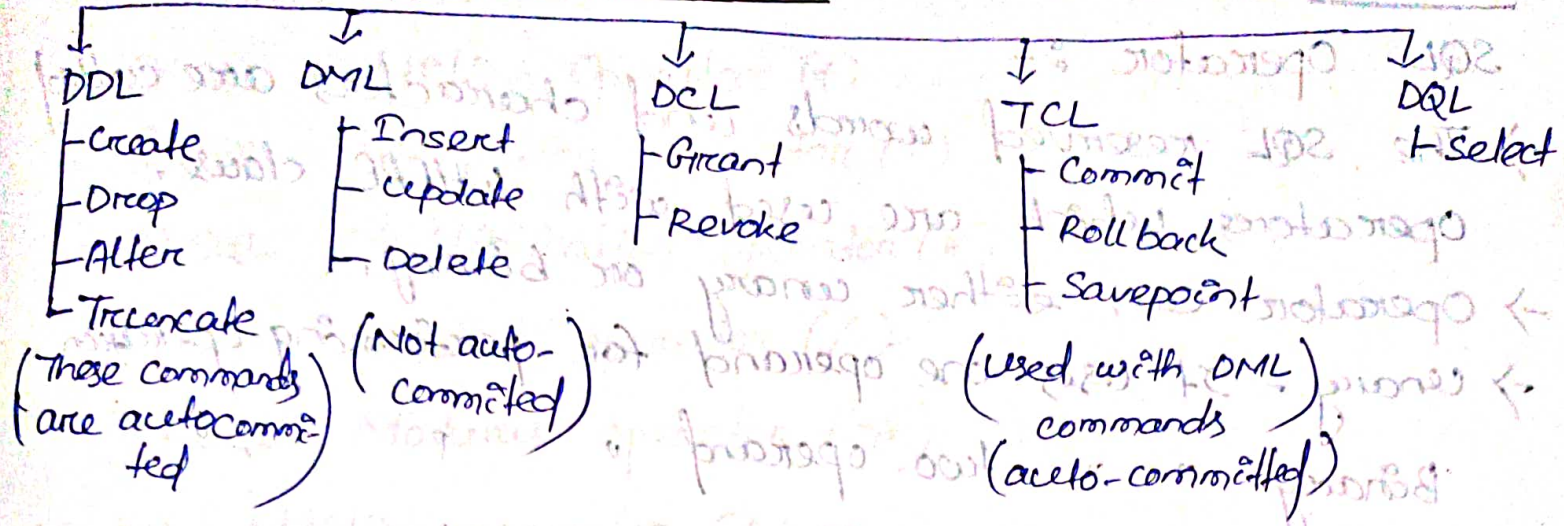
⑥ TINYTEXT : It holds a string with a max^m length of 255 characters.

⑦ MEDIUMTEXT : It holds a string with a max^m length of 16,777,215.

⑧ LONGTEXT : It holds a string with a max^m length of 4,294,967,295 characters.

⑨ ENUM(val1, ...) : It is used when a string object having only one value, chosen from a list of values. It contains 65535 values in an ENUM list.

SQL COMMAND



* Grant – It is used to give user access privileges to a database.

e.g.

```
GRANT SELECT, UPDATE ON MY-TABLE TO SOME-USER, ANOTHER USER;
```

* Revoke :- It is used to take back permissions from the user.

e.g.

```
REVOKE SELECT, UPDADE ON MY-TABLE FROM USER1, USER2;
```

* Savepoint : It is used to roll the transaction back to a certain point without rolling back the entire transaction

Syn SAVEPOINT SAVEPOINT-NAME;

SQL Process :-

29.01.22

SQL Operator :-

→ The SQL reserved words and characters are called operators, which are used with WHERE clause.

→ Operator can either unary or binary.

→ Unary - It uses one operand for performing operation.

Binary - " " two operand " "

Symbols	Operators
**	Exponentiation operator
+, -	Identity operator, Negation
*, /	Multiplication, Division
+, -, ||	Addition, Subtraction, Concatenation
<, >, >=, <=, !=, =, IN, IS NULL, LIKE, BETWEEN	Comparison
NOT	Logical negation
&& or AND	Conjunction
OR	Inclusion

Types of Operator :-

① SQL Arithmetic Operator
② SQL Comparison "
③ SQL Logical "
④ SQL Set "
⑤ SQL Bitwise "
⑥ SQL unary "

SQL Arithmetic Operator -

→ This operator perform mathematical operation on the numerical data.

Arithmetic Operators are —:

① SQL Addition Operator (+)

② SQL subtraction operator (-)

③ SQL Multiplication operator (*)

④ SQL Division operator (/)

⑤ SQL Modulus operator (%)

① SQL Addition operator (+) —

In SQL, we can easily add the numerical values of two columns of the same table by specifying both the col° names as 1st and 2nd operand.

Syntax SELECT operand1 + operand2;

e.g. SELECT emp-Salary + emp-Bonus as emp-Total Salary FROM employee;

② SQL Subtraction operator (-) —

e.g. SELECT Emp-Salary - Penalty as emp-Total Salary FROM employee;

③ SQL Multiplication operator (*) —

e.g. SELECT emp-Salary * 2 as emp-New-Salary FROM employee;

④ SQL Division operator (/) —

e.g. SELECT emp-Salary / 2 as emp-New-Salary FROM employee;

⑤ SQL Modulus operator (%) —

e.g. SELECT operand1 % operand2 as Remainder FROM employee;

SQL Comparison Operators :-

The comparison operators in SQL compare two different data of SQL tables and check whether they are same, greater and lesser.

→ These operators are used with WHERE clause.

Comparison operators are -

① SQL Equal Operator (=)
② SQL Not Equal " (!=)
③ SQL Greater than " (>)
④ SQL Greater than equals to operator (>=)
⑤ SQL Less than operator (<)
⑥ SQL Less than equals to operator (<=)

① SQL equal operator (=) :-

→ The equal operator in SQL shows only data that matches the specified value in the query.

e.g. SELECT * FROM employee WHERE emp-salary=3000;

② SQL not equal operator (!=) :-

→ This operator in SQL shows only those data that do not match the query's specified value.

e.g. SELECT * FROM employee WHERE emp-salary != 4500;

③ SQL Greater than (>) operater :-

→ This operator shows only those data which are greater than the value of the right-hand operand.

e.g. SELECT * FROM employee WHERE emp-Id > 202.

④ SQL Greater than equals to (>=) operator :-

→ This operator shows those data from the table which are greater than and equal to the value of right-hand operand.

e.g. SELECT * FROM employee WHERE emp_Id >= 202;

⑤ SQL less than (<) operator :-

→ This operator shows only those data from the database tables which are less than the value of right-side operand.

e.g. SELECT * FROM employee WHERE emp_Id < 204;

⑥ SQL less than equals to operator (<=) :-

→ This operator shows those data from the table which are lesser and equal to the value of the right-side operand.

e.g. SELECT * FROM employee WHERE emp_Id <= 203;

SQL Logical Operators :-

→ The Logical operators in SQL perform the Boolean operations, which give two results True and False.

→ These operators provide True value if both operands match the logical cond^n.

① SQL ALL operator
② SQL AND "
③ " OR "
④ " BETWEEN
⑤ IN
⑥ NOT
⑦ ANY
⑧ LIKE

① SQL ALL operator —

The ALL operator in SQL compares the specified value to all the values of a column from the sub-query in the SQL database.

-> This operator used with SELECT, HAVING & WHERE.

Syntax:

SELECT col°name1, columnname2, ... FROM table-name WHERE column Comparison-operator ALL (SELECT column FROM tablename);

eg: SELECT emp-Id, emp-name FROM employee WHERE emp-salary > ALL (SELECT emp-salary FROM employee WHERE city = Jaipur);

② SQL: AND operator :-

-> This operator show the record from the table if all the cond°s separated by AND operator evaluated to True.

Syntax:

SELECT col°1 ..... col°N FROM table-name WHERE cond° AND cond°2 ..... AND cond°N;

e.g. SELECT * FROM employee WHERE emp-salary = 2500 AND city = 'Delhi';

③ SQL OR operator :- (like AND)

Syntax:

SELECT col°1, ... col°N FROM table-name WHERE cond°1 OR cond°2 ..... OR cond°N;

e.g. SELECT * FROM employee WHERE emp-salary = 2500 OR city = 'Delhi';

④ SQL BETWEEN Operator :-

-> This operator in SQL shows the record with the range mentioned in the SQL query.

-> If there is no value in the given range, then it shows NULL value.

Syntax:

SELECT col°1, col°2 .... col°N FROM table-name WHERE col° BETWEEN value1 and value2;

e.g. SELECT * FROM employee WHERE emp-Salary BETWEEN 3000 AND 4500;

⑤ SQL IN operator —

→ The IN operator in SQL allows database users to specify two or more values in a WHERE clause.

→ It minimizes the requirement of multiple OR cond^n's.

Syntax :

SELECT col^n 1 .... col^n N FROM tablename WHERE col^n name IN (list of values);

e.g. SELECT * FROM employee WHERE emp-Id IN (202, 203, 205);

⑥ SQL NOT operator —

→ It shows the record from the table if the cond^n evaluates to false. It used WHERE clause.

Syntax :

SELECT col^n 1, col^n 2, ... col^n N FROM tablename WHERE NOT cond^n;

e.g. SELECT * FROM employee WHERE NOT emp-city = 'Delhi';

⑦ SQL ANY operator —

→ It shows the records when any of the values returned by the sub-query meet the cond^n.

→ The ANY operator must match at least one record in the inner query and must be preceded by any SQL comparison operator.

Syntax :

SELECT col^n 1, col^n 2 ... col^n N FROM tablename WHERE col^n comparison operator ANY (SELECT col^n FROM tablename WHERE cond^n (s));

(8) SQL LIKE operator –

-> It shows those records from the table which match with the specified given queery.

-> It uses (%) sign.

-> It uses WHERE clause with SELECT, UPDATE, DELETE statement.

Syntax:

SELECT col^n1, ... col^nN FROM table-name WHERE col^name LIKE pattern;

eg SELECT * FROM employee WHERE emp-Name LIKE 'S%';

[%S, S%Y, %S%]

03.02.22.

SQL Union Operator :–

-> The SQL union operator combines the result of two or more SELECT statements and provide single output.

-> The Data type and number of col^n must same for each SELECT statement.

-> This operator doesn't show duplicate records.

Syntax:

SELECT col^n1, col^n2, ... col^nN FROM table-name1 [WHERE cond^n]

UNION

SELECT col^n1, col^n2, ... col^nN FROM table-name2 [WHERE cond]

eg. SELECT emp-Id, emp-Name FROM employee1;

UNION

SELECT emp-Id, emp-Name FROM employee2;

SQL Union ALL Operator :–

-> It is same as UNION operator, but it also shows the same record.

SQL Intersect Operator –

-> This operator shows the common record from two or more SELECT statements.

-> The Data type and number of col's must be same for each SELECT statements.

SQL Minus Operator –

-> This operator combines the result of two or more SELECT statements and shows only the results from the first data set.

SQL Unary Operators :–

① SQL Unary +ve operator

② " " -ve "

③ Bitwise NOT operator

SQL Unary +ve operator — (+)

SELECT + (col^n 1) + (col^n 2) + ... + (col^n N) FROM table-name
[WHERE cond^n];

SELECT + emp-Salary employee;

SQL Unary -ve operator — (-)

SELECT - empsalary employee WHERE empcity = 'Delhi';

SQL Bitwise NOT operator —

-> The SQL Bitwise NOT operator provides the one's complement of the single numeric operand.

-> e.g. 001100 ⟶ 110011

SELECT ~ (col^n 1), ~ (col^n 2), ... ~ (col^n N) FROM tablename
WHERE cond^n ;

SELECT ~ Stu-Marks employee;

SQL Bitwise operators :-

→ This operators perform the bit operation on the integer values.

① Bitwise AND (&)

② Bitwise OR (|)

Bitwise AND (&) -

→ The Bitwise AND operator performs the logical AND operation on the given integer values.

SELECT col^n1 & col^n2 .... & col^nN FROM tablename

WHERE cond^n's ;

→ When we use the Bitwise AND, then SQL converts the values of both col^n in binary format, AND operation performed on converted bits.

→ SQL converts the resultant bits into decimal format.

Bitwise OR (|) - (same &)

Unit-4 18.02.22

Introduction to Transaction Processing concepts and theory

Transaction Processing Systems are systems with large databases and hundreds of concurrent users executing database transactions.

e.g. Airline reservations, banking, credit card processing, online retail purchasing, stock markets, supermarket checkouts and many other applications.

A Database System is 2 types according to the number of user who can use the system concurrently.

① Single-user System
② Multi-user system

Single-user system :-
A DBMS is single-user if at most one user at a time can use the system.

→ Single-user DBMSs are mostly restricted.
→ e.g. Personal Computer

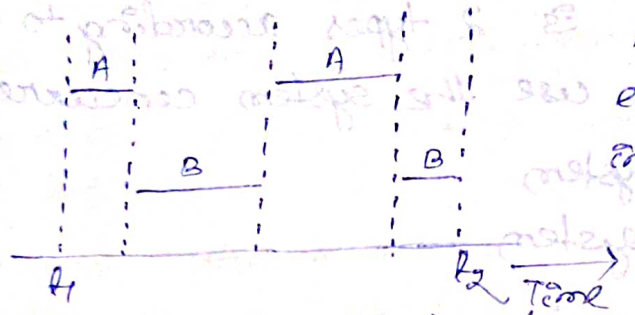
Multiuser System :-
A DBMS is multi-user if many (more than one) users can use the system and hence access the database at a time.

→ Most DBMSs are multiuser.
→ e.g. A airline reservation system is used by hundred of travel agents, Database system used in banks, insurance agencies, stock exchanges, supermarkets and many other applications.

* Multiusers can access database and use computer systems simultaneously because of the concept of multiprogramming, which allows the operating system to execute multiple programs at the

-> A single CPU can only execute at most one process at a time. But in Multiprogramming operating systems execute some commands from one process, then suspend that process and execute some commands from the next process and so on. A process is resumed at the point where it was suspended when ever it gets turn to use the CPU again.

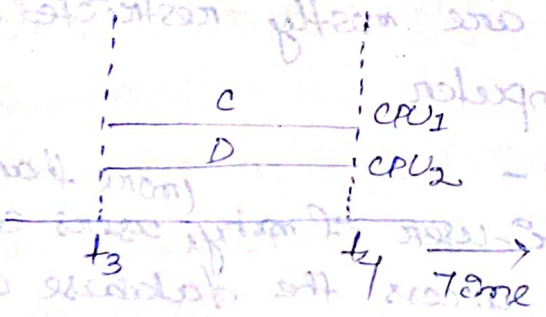
-> Hence, concurrent execution of process is actually interleaved.



Here A and B, executing concurrently in an interleaved fashion.

-> If the computer system has multiple hardware processors (CPUs), parallel processing of multiple process is possible.

e.g.



Here C and D in parallel processing.

Transaction Read and Write Operation :-

A transaction is an executing program that forms a logical unit of database processing.

-> A transaction includes one or more database access operations includes insertion, deletion, modification etc.

-> If the database operations in a transaction do not update the database but only retrieve data, the transaction is called a read-only transaction; otherwise known as read-write transaction.

The basic database access operations that a transaction can include —

① read(X) : Reads a database item named X into a program variable.

② Write(X) : Write the value of program variable X into the database item named X.

Why Concurrency Control IS Needed :—

Several problems can occur when concurrent transactions execute in an uncontrolled manner.

→ If we can't control concurrent transaction then we face some problems.

① Lost of Update Problem :-

This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database items incorrect:

Suppose T_1 and T_2 are submitted at approximately the same time. Suppose their operations interleaved then the final value of X is incorrect because T_2 reads the value of X before T_1 changes in the database and hence updated value resulting from T_1 is lost.

T_1	T_2
read(X);	
X = X - N;	
	read(X);
	X = X + M;
write(X);	
read(Y);	
	write(X);
Y = Y + N;	
write(Y);	

Here, item X has an incorrect value because its update by T_1 is lost.

e.g. X = 80, N = 5, M = 4

then X = 79 (final)

If interleaving occurs then X = 84 because update in T_1 was lost.

The temporary Update or Dirty Read Problem :-
This problem occurs when one transaction updates a database item and then the transaction fails for some reason.

e.g. T_1 updates item X and then fails before completion, so the system must change X back to its original value. Before it can do so, transaction T_2 reads the temporary value of X, which will not be recorded permanently in the database because of failure of T_1.
The value of item X that is read by T_2 is called dirty data.

T_1	T_2
read(X);	
X = X - N;	
write(X);	
	read(X);
	X = X + M;
	write(X);
read(Y);	

T_1 fails and change the value of X back to its old value and T_2 has read the temporary incorrect value of X.

The Incorrect Summary Problem :-
If one transaction is calculating an aggregate summary func^n on a number of database items while other transactions are updating some of these items, the aggregate func^n may calculate some values before they are updated and others after they are updated.

e.g. Suppose T_3 calculating the total no. of reservations on all flights while T_1 is executing. If interleaving operation occurs then T_3 reads the value of ~~X~~ after ~~N~~ ~~S~~ before those N seats have added to it.

T_1	T_3
	sum = 0
	read (A);
	sum = sum + A;
read (X);	
X = X - N;	
write (X);	
	read (X);
	sum = sum + X;
	read (Y);
	sum = sum + Y;
read (Y);	
Y = Y + N;	
write (Y);	

T_3 reads X after N is subtracted and reads Y before N is added so a wrong summary is result.

19.02.22

Why Recovery is Needed :-

Whenever a transaction is submitted to a DBMS for execution, the system is responsible for making sure that either all the operations in the transaction are completed successfully and their effect is recorded permanently in the database, or that the transaction doesn't have any effect on the database or any other transactions.

→ The DBMS must not permit some operations of a transaction T to be applied to the database while other operations of T is not. This basically may happen if a transaction fails after executing some of its operations but before executing all of them.

Types of failures :- Failures are generally classified as transaction, system and media failures. There are several possible reasons for a transaction to fail in the middle of the exe tion

1. A Computer Failure :- (System Crash)
A hardware, software or network error occurs in the computer system during transaction execution.
→ Hardware crashes are usually media failures.
→ e.g. main memory failure.

2. A Transaction or system error :-
Some operation in the transaction may cause it to fail, such as integer overflow or division by zero.
→ Transaction failure also occur because of erroneous parameter values or logical programming error.

3. Local errors or exception conditions detected by the transaction :-
During transaction execution, certain cond's may occur that necessitate cancelation of transaction.
e.g. An insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawl, to be canceled.
→ This exception could be programmed in the transaction itself and in such a case would not be considered as a transaction failure.

4. Concurrency control enforcement :-

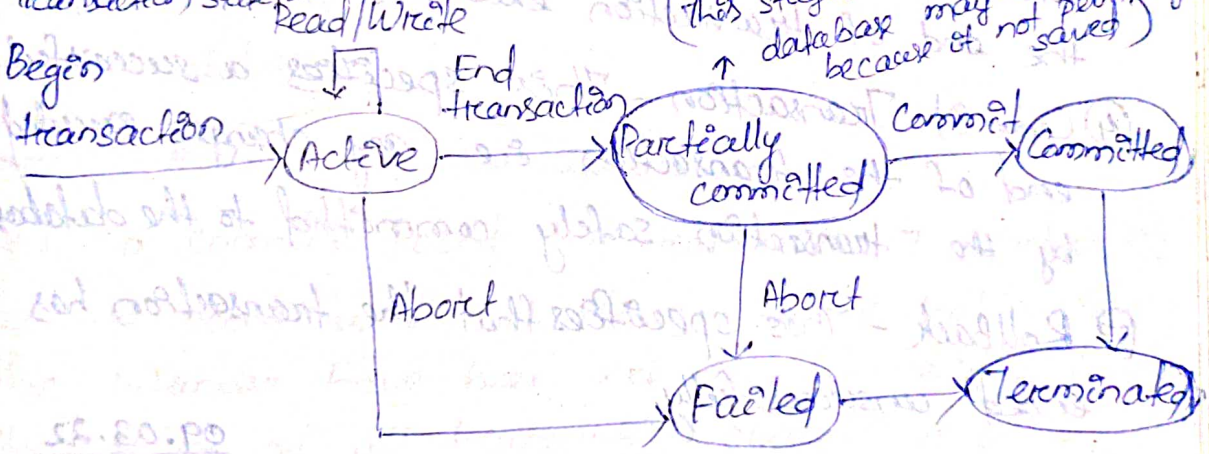
5. Disk Failure :-
Some disk blocks may lose their data because of a disk read/write head crash. This may happen during a read or write operation of the transaction.

6. Physical problems and catastrophes :-
This refers to an endless list of problems that includes power or air conditioning failure, fire, theft, sabotage, overwriting disks or tapes by mistake and mounting of a wrong tape by the operator.

* When a failure 1 and 4 occurs, the system must keep sufficient information to quickly recover from the failure.
* Disk or catastrophic failure don't happen frequently, if they do occur, recovery is a major task.

LQ Transaction and System Concepts :-

1. Transaction state :-



A transaction goes into an active state immediately after it starts execution, where it can execute its Read and write operations.

→ When the transaction ends, it moves to the partially committed state. This point, some recovery protocols need to ensure that a system failure will not result in an inability to record the changes of the transaction permanently.

→ Once this check is successful, the transaction reached its commit point and enters the committed state.

→ When a transaction is committed, it has concluded its execution successfully and all changes must be recorded permanently in the database, even if a system failure occurs.

→ A transaction can go to the failed state if one of the checks fail.

→ The transaction terminated state corresponds to the transaction leaving the system.

Different Transaction State —

① Begin Transaction – This marks the beginning of transaction execution.

② ~~End~~ Read or Write Transaction – This specifies the Read or write operations on the database.

③ End Transaction – This specifies that Read and Write transaction operations have ended and marks the end of transaction execution.

④ Commit Transaction – This specifies a successful end of the transaction i.e. any changes executed by the transaction safely committed to the database.

⑤ RollBack – This specifies that the transaction has ended unsuccessfully.

09.03.22

② The System Log —

Log : The log keeps track of all transaction operations that affect the values of database items.

→ This information may be needed to permit recovery from transaction failures.

→ The log is kept on disk, so it is not affected by any type of failure except for disk or catastrophic failures.

→ In addition, the log is periodically backed up to archival storage to guard against such catastrophic failures.

Types of log record :-

1. [start_Transaction, T] : Records that transaction T has started execution.
2. [Read-item, T, X] : Records that transaction T has read the value of database item X.
3. [write_item, T, X, Old value to new value] : Records that transaction T has changed the value of database item X from oldvalue to newvalue.
4. [Abort, T] : Records that transaction T has been aborted.
5. [Commit, T] : Records that transaction T has completed successfully and affirms that its effect can be committed to the database.

Commit Point of a Transaction :-

Def^n a commit Point : A transaction T reaches its commit point when all its operations that access the database have been executed successfully and the effect of all transaction operations on the database have been recorded on the log.

→ Beyond the commit point, the transaction is said to be committed and its effect must be permanently recorded to the database.

→ The transaction then writes a commit record [commit, T] into the log.

RollBack of transaction : Needed for transactions that have a [start_transaction, T] entry into the log but no commit entry [commit, T] into the Log.

Redoing transaction : Transaction that have written their commit entry in the log must also have recorded all their write operations in the log; otherwise they would not be committed, so their effect on the database can be redone from the log entries. (Notice that the log file must be kept on disk)

→ At the time of a system crash, only the log entries that have been written back to disk are considered in the recovery process because the contents of main memory may be lost.

Force writing a log : Before a transaction reaches its commit point, any portion of the log that has not been written to the disk yet must now be written to the disk.

→ This process is called force-writing the log file before committing a transaction.

12.03.22

Characterizing Schedules based on Recoverability :-

When transactions are executing concurrently in an interleaving fashion, then the order of execution of operation from the various transaction is known as schedule.

e.g.

T_1	T_2
read(A)	
A = A + 5	
	read(B)
	B = B + 10
	write(B)
write(A)	

Schedule

$T_1 \rightarrow T_2 \rightarrow T_1 \rightarrow T_2$

→ Two operation in a schedule said to be conflict if they satisfy all three of following conditions.

① They belong to different transaction

② They accessing same data-item.

③ Atleast one of operation is a write-item(x).

e.g. Let $r_1(x)$ and $w_2(x)$ are conflict because they are belong to different transaction, accessing the same data-item and one operation is write operation.

$r_2(x)$ and $w_1(x)$, $w_1(x)$ and $w_2(x)$ are may be conflict.

But $r_1(x)$ and $r_2(x)$, $w_2(x)$ and $w_2(x)$, $r_1(x)$ and $w_1(x)$ are not conflict because they do not satisfy all the 3 cond's.

→ Based on Recoverability the schedule is categories into 2 types. ① Recoverable schedule

② Non-Recoverable schedule.

S.Q Recoverable Schedule :-

It ensures that once a transaction T is committed it should never be necessary to rollback T. This type of schedules known as recoverable schedule.

Non-recoverable schedule :-

The schedule that don't satisfy the recoverable schedule cond^{n} known as Non-recoverable schedule.

* A schedule S is recoverable if no transaction T in S commits until all transaction T' that have written an item that T reads have committed.

Q. Cascading Rollback :-

In recoverable schedule no committed transaction ever needs to be rollback. But it is possible for a cascading rollback to occur, where an uncommitted transaction has to be roll back because it read an item from the transaction that failed.

Q. Cascadless Schedule :-

A schedule is said to be cascadless or to avoiding cascade rollback, if every transaction in a schedule reads only item that were written by committed transaction.

Strict Schedules :-

A schedule in which a transaction can neither read or write an item X until the last transaction that wrote X has committed.

16.03.22

Characterizing Schedules Based on Serializability :-

Serializability - It is the concept of executing transaction without interleaving.

→ The type of schedule characterize based on serializability and that are consider correct when concurrent transaction are executing.

→ e.g. Two transaction T_1 & T_2 submit to DBMS and if no interleaving of operations is permitted, there are only two possible outcomes are arises.

1. Execute all the operations of transaction T_1 (in sequence) followed by all the operations of transaction T_2.

2. Execute all the operations of transaction T_2 (in sequence) followed by all the operations of transaction T_1.

3. The concept of serializability of schedule is used to identify which schedules are correct when transaction execution have interleaving of their operations in the schedules.

e.g.

(a)

T_1	T_2
read_item(X);	
X = X - N;	
write_item(X);	
read_item(Y);	
Y = Y + N;	
write_item(Y);	
	read_item(X);
	X = X + M;
	write_item(X);

(b)

T_1	T_2
	read_item(X);
	X = X + M;
	write_item(X);
read_item(X);	
X = X - N;	
write_item(X);	
read_item(Y);	
Y = Y + N;	
write_item(Y);	

(c)

T_1	T_2
read_item(X);	
X = X - N;	
write_item(X);	
	read_item(X);
	X = X + M;
	write_item(X);
read_item(Y);	
Y = Y + N;	
write_item(Y);	

(a) T_1 followed by T_2

(b) Serial schedule B: T_2 followed by T_1

(c) Two nonserial schedule C and D with interleaving of operations.

Types of schedule based on Serializability :-

There are 3 different schedule based on serializability.

① Serial Schedule (interleaving not allow)
② Non-serial schedule (interleaving allow)
③ Conflict Schedule

1. Serial Schedule :-

The schedule are called serial schedule, when the operations of each transaction are executed consecutively without any interleaved operation, from the other transaction.

2. Non-Serial Schedule :-

A schedule is said to be non-serial when each sequence of operation interleaves from the two transaction.

e.g.

S1

T_1	T_2
read(X);	
X = X+3;	
write(X);	
read(Y);	
Y = Y-3;	
write(Y);	
	read(X);
	X = X-3;
	write(X);

(Serial Schedule)

cond^n $T_1 \rightarrow T_2$

S2

T_1	T_2
read(X);	
X = X+3;	
write(X);	
	read(X);
	X = X-3;
	write(X);
read(Y);	
Y = Y-3;	
write(Y);	

Non-serial schedule

cond^n $T_1 \rightarrow T_2 \rightarrow T_1$

Serializability of a schedule :-

This concept is used to identify with schedules are correct when transaction executions have interleaving of their operations in their schedule.

23.03.22

Concurrency Control Techniques :-

The concurrency control technique used to ensure non-interference of isolation property of concurrently executing transactions.

→ The concurrency control techniques use the protocols or set of rules to ensure the serializability. These are

ⓐ Locking ⓒ Multiversion

ⓑ Timestamp ⓓ Validation / Certification

ⓐ Locking :-

The Locking technique used to lock the data item to prevent multiple transaction from accessing the item concurrently.

ⓑ Timestamp :-

A Timestamp is a unique identifier for each transaction generated by the system.

ⓒ Multiversion :-

Multiversion concurrency control protocol that used multiple version of a data item.

ⓓ Validation :-

A protocol based on the concept of validation or certification of a transaction after it executes it's operation.

Two Phase Locking technique for concurrency control :-

Lock - A lock is a variable associated with a data item that describe the status of the item with respect to possible operation that can be applied to it.

→ There is one lock for each data item in the database and the locks are used as a means of synchronising the access by concurrent transactions to the database item.

The two problem associated with the use of Locks are.
ⓐ Dead Lock ⓑ Starvation.

25.03.22

Types of Locking :-

→ Several types of Locks are used in concurrency control.
① Binary lock & ② Shared / Exclusive Lock

Binary Lock :- (read/write Lock)-
Binary Locks are simple, but are also too restrictive for database concurrency control purposes. So are not used in practice.

→ A Binary lock can have two states or values, locked (1) and unlocked (0).
→ A distinct lock is associated with each database item X.
→ If the value of the lock on X is 1, item X can't be accessed by a database operation that requests the item.
→ If the value of the lock on X is 0, the item can be accessed when requested, and the lock value is changed to 1.
→ The current value of lock associated with item X as lock (X).
→ Two operation lock-item and unlock-item are used with binary locking. If lock(X) = 1, the transaction is forced to wait. If lock(X) = 0, it is set to 1 and transaction is allowed to access item X.
→ When the transaction is through using the item, it issues an unlock-item (X) operation, which sets LOCK(X) back to 0 so that X may be accessed by other transactions.

Algorithm for binary lock —

```
lock_item(X):
B: if LOCK(X) = 0
       then LOCK(X) ← 1
   else
       begin
       wait (until LOCK(X) = 0
       and the lock manager wakes up the transaction)
       goto B
       end;

unlock_item(X):
   LOCK(X) ← 0;
   if any transactions are waiting
       then wakeup one of the waiting transactions;
```

Rules for binary Lock :—

1. A transaction T must issue the operation lock_item(X) before any read_item(X) or write_item(X) operations are performed in T.
2. A transaction T must issue the operation unlock_item(X) after all read_item(X) and write_item(X) operations are completed in T.
3. A transaction T will not issue a lock_item(X) operation if it already holds the lock on item X.
4. A transaction T will not issue an unlock_item(X) operation unless it already holds the lock on item X.

Shared / Exclusive (or Read/Write) Locks :-
It provide more general capabilities and are used in practical database locking schemes.

→ In this, we should allow several transactions to access the same item X if they all access X for reading purpose only. This is because read operations on the same item by different transactions are not conflicting.

→ However, if a transaction is to write an item X, it must have exclusive access to X.

→ For this purpose, a different type lock called multiple-mode lock is used. And this scheme is called shared/exclusive or read/write locks.

→ There are 3 locking schemes :-
1. read-lock(X)
2. write-lock(X)
3. unlock(X)

→ A read-locked item is also called shared-locked because other transactions are allowed to read the item. And write-locked item is called exclusive-locked because a single transaction exclusively holds the lock on the item.

→ A Lock associated with an item X, Lock(X) has three possible states:
1. If Lock(X) = write-locked, the value of locking-transaction is a single transaction that holds the exclusive lock on X.
2. If Lock(X) = read-locked, the value of locking transaction is a list of one or more transactions that hold the shared lock on X.

Algorithm for shared / Exclusive Lock :-

read_lock(X):

B: if LOCK(X) = "unlocked"
then begin LOCK(X) ← "read-locked";
no-of-reads(X) ← 1
end
else if LOCK(X) = "read-locked"
then no-of-reads(X) ← no-of-reads(X)+1
else begin
wait (until LOCK(X) = "unlocked"
and the lock manager wakes up the transaction);
goto B
end;

write_lock(X):

B: if LOCK(X) = "unlocked"
then LOCK(X) ← "write-locked"
else begin
wait (until LOCK(X) = "unlocked"
and the lock manager wakes up the transaction);
goto B
end;

unlock(X):

if LOCK(X) = "write-locked"
then begin LOCK(X) ← "unlocked";
wakeup one of the waiting transactions, if any
end
else if LOCK(X) = "read-locked"
then begin
no-of-reads(X) ← no-of-reads(X)-1;
if no-of-reads(X)=0
then begin LOCK(X) = "unlocked";
wakeup one of the waiting transactions, if any
end
end;

Rules for shared / Exclusive Locks —

1. A transaction T must issue the operation read-lock(X) or write-lock(X) before any read-item(X) operation is performed in T.
2. A transaction T must issue the operation write-lock(X) before any write-item(X) operation is performed in T.
3. A transaction T must issue the operation unlock(X) after all read-item(X) and write-item(X) operations are completed in T.
4. A transaction T will not issue a read-lock(X) operation if it already holds ~~a read lock or~~ a write lock on item X. This rule may be relaxed.
5. A transaction T will not issue a write-lock(X) operation if it already holds a read lock or write lock on item X.
6. A transaction T will not issue an unlock(X) operation unless it already holds a read lock or a write lock on item X.

26.03.22

Conversion of Lock :-

Lock Conversion means the transaction that already holds a lock on item X is allowed under certain conditions to convert the lock from one locked state to another.

e.g. It is possible for a transaction T to issue a read-lock(X) and then later to upgrade the lock by issuing a write-lock(X) operation.

→ If T is the only transaction holding a read lock on X at the time it issues the write-lock(X) operation, the lock can be upgraded; otherwise, the transaction must wait.

→ It is also possible for a transaction T to issue a write-lock(x) and then later to downgrade the lock by issuing a read-lock(x) operation.

Guaranteeing Serializability by Two-Phase Locking:

A Transaction is said to follow the two-phase locking protocol if all locking operation i.e. write-lock, read-lock precede the first unlock operation in the transaction. And such a transaction is devided onto two phases. ① Expanding or Growing → or first phase ② shrinking or second phase

① Expanding Phase :-

During this phase new locks on items can be acquired but none can be released.

→ If Lock conversion is allowed, then upgrading of locks must be done during the expanding phase.

② Shrinking phase :-

During this phase existing locks can be released but no new locks can be acquired.

→ If Lock conversion is allowed, then downgrading of locks must be done in the shrinking phase.

→ Hence, a read-lock(x) operation that downgrades an already held write lock on x can appear only in the shrinking phase.

e.g.

T_1
read-lock(y);
read-item(y);
write-lock(x);
unlock(y)
read-item(x);
x = x+y;
write-item(x);

If (T_1 & T_2) follow the two-phase locking protocol.

* They can produce a deadlock.

```
T2
read-lock(X);
read-item(X);
write-lock(X);
unlock(X)
read-item(Y);
Y=X+Y;
write-item(Y);
unlock(Y);
```

-> If every transaction in a schedule follows the two-phase locking protocol, the schedule is guaranteed to be serializable, obviating the need to test for serializability of schedules.

Basic, Conservative, Strict and Rigorous Two-phase Locking :-

There are a number of variations of two-phase locking. These are

① Basic Two-phase Locking Protocol.
② Conservative " "
③ Strict " "
④ Rigorous " "

Basic Two-phase Locking Protocol :-

Let assume X, Y, Z be the three data item required by transaction T. In basic Two-phase locking protocol X must remain locked by T until all items that the transaction needs to read or write have been locked only then can X be released by T.

Conservative (static) Two-phase locking protocol :-

In this protocol a transaction to lock all the items it access before the transaction begins execution

by predeclaring its read-set and write-set.

→ If any of the predeclared items needed cannot be locked, the transaction doesn't lock any item instead it waits until all the items are available for locking.

→ Conservative is a deadlock-free protocol.

Strict Two-Phase Locking Protocol :-

In this variation, a transaction T doesn't release any of its exclusive locks until after it commits or aborts.

→ Hence, no other transaction can read or write an item that is written by T unless T has committed, leading to a strict schedule for recoverability.

→ Strict 2PL is not deadlock-free.

Rigorous Two-Phase Locking Protocol :-

A more restrictive variation of strict 2PL is rigorous 2PL, which also guarantees strict schedules.

→ In this variation, a transaction T does not release any of its locks until after it commits or aborts.

Difference bet^n Conservative and rigorous 2PL :-

30.03.22.

Dealing with DeadLock and Starvation :-

DeadLock occurs when each transaction T in a set of two or more transaction is waiting for some item i.e. lock by some other transaction T' in the set. Hence each transaction in the set is on a waiting queue, waiting for one of the other transaction in the set to release the lock on an item.

DeadLock Prevention Protocol :-

DeadLock can be prevented by using a deadlock prevention protocol. There are no. of deadlock prevention protocol such as.

① Conservative 2 phase locking protocol
② Limit the concurrency execution
③ Transaction Timestamp

Conservative 2PL :-

The Conservative 2PL protocol requires that every transaction lock all the items it needs in advance And if any of the items can't be obtained, none of the items are locked. Rather, the transaction waits and then tries again to lock all the items it needs.

Limit the Concurrency execution :-

The 2nd protocol is to limit the concurrency execution i.e. it involves ordering the items in the database and making sure that a transaction that needs several items will lock them according to that order.

Transaction Timestamp :-

The 3rd technique to avoid deadlock is transaction timestamp TS(T), which is unique identifier assign to each transaction to avoid deadlock.

→ In transaction timestamp, the older timestamp has

the smaller timestamp value. And the two schema that prevent the deadlock are ① wait die
② wound die

Wait die :-
If $TS(T_i) < TS(T_j)$ then (T_i is older than T_j), T_i is allowed to wait; otherwise (T_i is younger than T_j) then abort T_i and restart it later with the same timestamp.

Wound die :-
If $TS(T_i) < TS(T_j)$ then
(T_i is older than T_j) then abort T_j and restart it later with the same timestamp
Otherwise (T_i younger than T_j).
So T_j is allowed to wait

Starvation :-
This problem may occur when a transaction can't proceed for an indefinite period of time while other transactions in the system continue normally and this occurs if the waiting schema for the locked item is unfair, and giving priority to some transactions over others and this can be solved by using first come first serve queue i.e. the transaction are enabled to lock an item in the order in which they originally requested the lock.

e.g. of deadlock

T_1	T_2
read-lock(X)	
read-item(Y)	
	read-lock(X);
	read-item(X);
write-lock(X)	
	write-lock(Y);

Here T_1 & T_2 both are deadlocked because neither T_1 nor T_2 nor any other transaction can access items X and Y.

~~Concurrency control based on timestamp order :-~~

Limit the concurrency execution -

On this the programmer is aware of the chosen order of items, which is also not practical on database context.

Q2

Concurrency Control Based on Timestamp ordering :-

Basic Timestamp ordering protocol —

→ unique value assign to every transaction.

→ Tells the order (when they enter into system)

TS(T) = Timestamp values of a transaction

Read-TS (RTS) = Last (Lastest) transaction which performed Read successfully.

Write-TS (WTS) = Last (Lastest) transaction member which performed write successfully.

10:00	10:10	10:15
T_1	T_2	T_3
100	200	300
Older	Younger	Youngest

TST (T_i)

Read-TS

10	20	30
T_1	T_2	T_3
Read(A)		
	Read(A)	
		Read(A)

Read-TS (A) = 30

Write-TS

10	20	30
T_1	T_2	T_3
write(A)		
	write(A)	
		write(A)

write-TS (A) = 20

Basic TS :- The transaction which come first will complete first.

Rules :-

(a) If WTS (A) > TS (T_i), Rollback T_i

(b) Otherwise execute R (A) operations.

Set RTS (A) = Max {RTS (A), TS (T)}

② Transaction T_i issues write(A) operation.

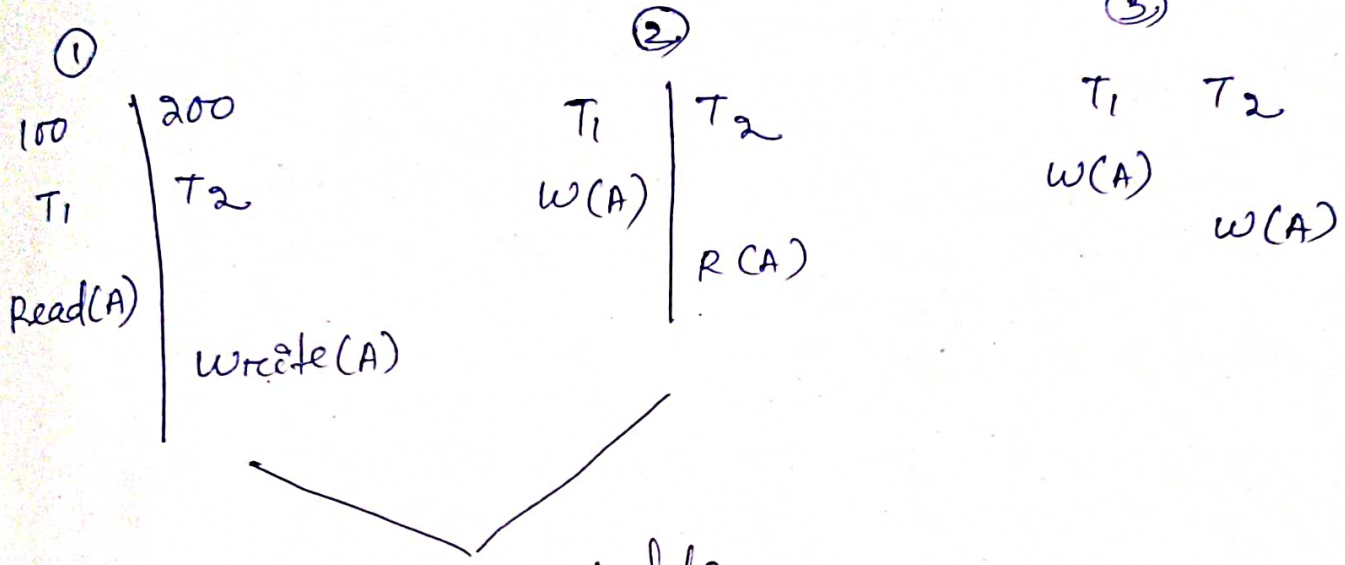
(a) if $RTS(A) > TS(T_i)$ then Rollback T_i

(b) if $WTS(A) > TS(T_i)$ then Rollback T_i

(c) Otherwise execute write(A) operation

set $WTS(A) = TS(T_i)$

① ② ③



Parallel schedule
(Multiple transaction rule)