Date. 16 June '25
Page No.

1) DBMS

- 1) Data
raw bytes
can be in form of images, float, int etc
unorganized.

- Information
processed data is information.
Information helps in decision making.
When data gets a meaning, it is information.

- 1) Database
electronic system that stores the data.
It can be disk or anything.

- 1) DBMS
1) Data base Management system.
Made of two things.
- 1) Database
- Management system → It provide functionality to modify data.
add, update, delete.

- Need for DBMS
DBMS helps us with file system flaws as.

Date.
Page No.

i) Data redundancy / inconsistency.
If one thing gets changed or updated then it is reflected to every place. This is called consistency.

storing same data on multiple places causes duplicates. This is redundancy.

ii) Accessible.
Data can be accessed efficiently and quickly by a simple query.

iii) Atomicity.
either a yes or no. Their is nothing in between e.g. credit and debit.

iv) Security
The security is maintained. Their are methods to grant or take access of data.

v) concurrent access.
It means multiple queries can run together without any trouble.

vi) Isolation of data.
data can be in different formats. It leads to data manipulation issues.

Date.
Page No.

vii) Integrity
Their are multiple constraints that help in functionality. e.g balance can not be below 10,000.

- Abstraction
It is a technique used to hide data from different uses.
It follows three level architecture.

i) External level
highest level of abstraction.
also called view schema.
This is used by end users.
Their are multiple schema for different user groups
These schema are called subschema.

eg. view for logistics team ≠ view for customer servic

Security measures can also be implemented.

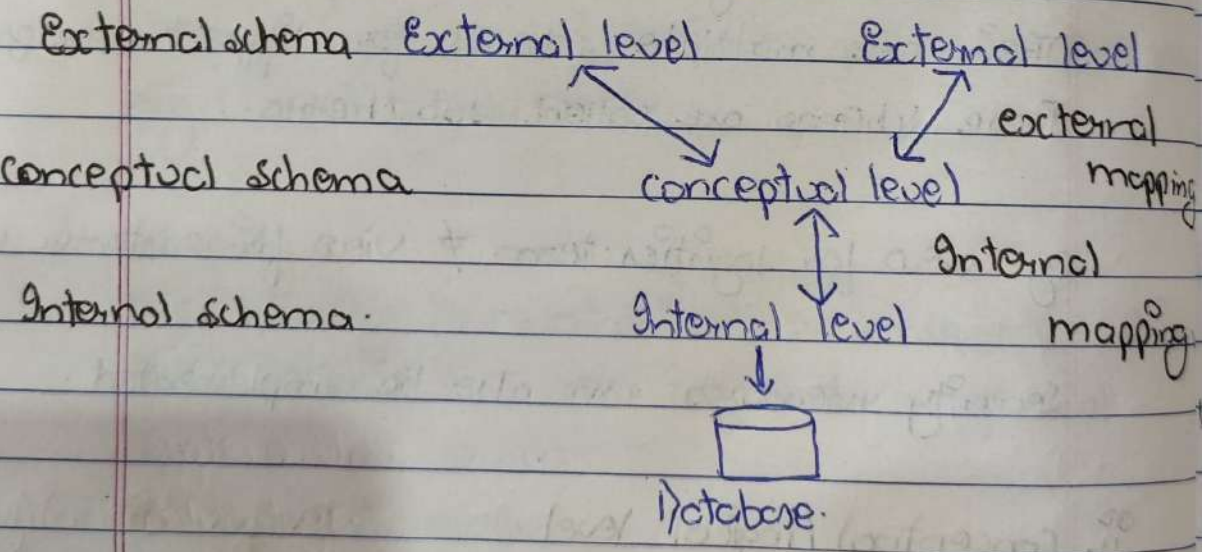
ii) Conceptual / logical level
It is also called logical schema.
It tells us what data and its relationship.
It does not focus on how data is stored.
It is used by DBA.

Date.
Page No.

iii) Internal level / Physical level
It is the place where data is stored.
It tells us how data is stored.
change in physical level does not affect logical level
It has physical schema that describe physical storage structure.
It goal is to make data access efficient.

Note converting data from physical to logical level is called mapping of data. Internal mapping.

Architecture



- Schema
Blueprint in DBMS that defines how data is organized in database.
It describes tables, fields, column, views etc.

In short, logical structure of database.

Date.
Page No.

- Instance
Total data points stored in database at a point of time is Instance of DB.

i) DBA
i) Database Administrator.
person who has the control of the system
It has some functions like:-
1. schema definition.
2. Storage structure and access method
3. Authorization control
4. Routine maintenence
   - backups
   - security patches
   - Upgrades.

- DBMS system Architecture

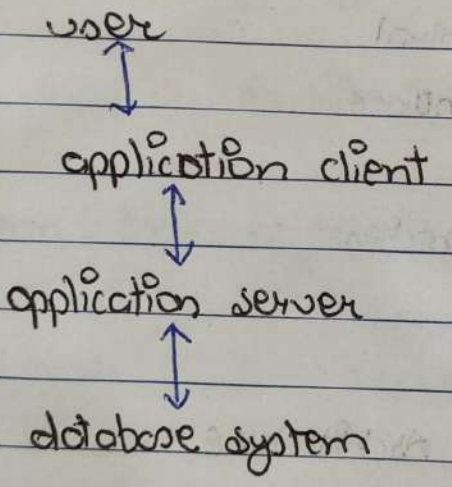
i) Tier 1 Architecture.
It states that client, server and Database, all three are stored in single place.
e.g. while learning sql

ii) Tier 2 Architecture.
It has two components.
Application sides sends query to the database system.

Date.
Page No.

user can directly access the database.
API like ODBC and JDBC are used.

ii) Tier 3 Architecture
It has three parts.
client machine has frontend and communicate to application server.
Application server acts as a mediator between client side and database system.



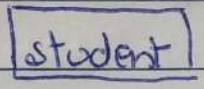
These architecture are used for large scale system like amazon.
Advantage.
Scalability -> multiple client and application server can be set up.
Integrity -> It does not affect security as client can not connect to database directly.

Date. 17 June '25
Page No.

1) ata models
These are the models that defines the structure of data, relationships, constraints.

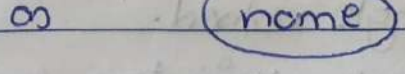
ER Model
It stands for entity relationship model

entity
It is any real world object that has some attributes of itself.
e.g-> student is an entity.
It is denoted as.



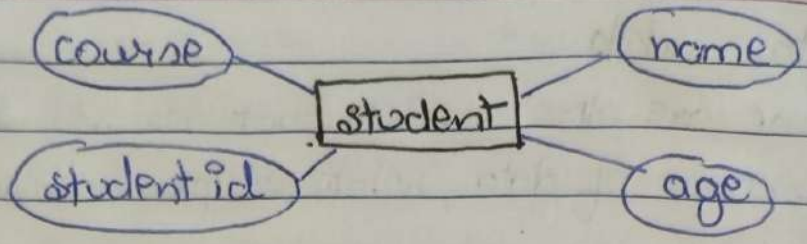
entity set
This is a collection of similar type of entities.
eg. There can be multiple students.

Attributes.
These are the features of the entity. e.g. student has several attributes ie, studentname, age, id etc.
These are different for different students.

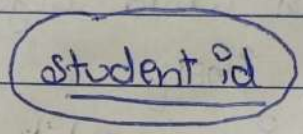
These are denoted as 

Date.
Page No.

∴



Note The attributes used to uniquely define an entity is called primary key.
e.g studentid is primary key
It is denoted as



Primary key is unique for all and can not be null.

Types of Attributes.

- Simple
These are single attributes that can not be further broken down.
e.g. Adhaar number.

- composite
These can be further broken down.
eg name broken into firstname, lastname.

- single valued.
These attributes can have only one value.

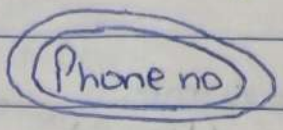
Date.
Page No.

e.g studentid.

- Multivalued

These can have multiple values for one entity.

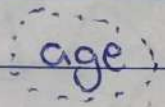
e.g. phone no.



- Derived.

These values can be derived from other attributes.

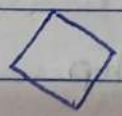
e.g Age derived from date of birth.



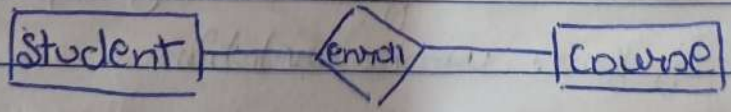
Relationships

These are used to define how two entities are related to each other.

It is denoted by.



e.g.



Strong entity

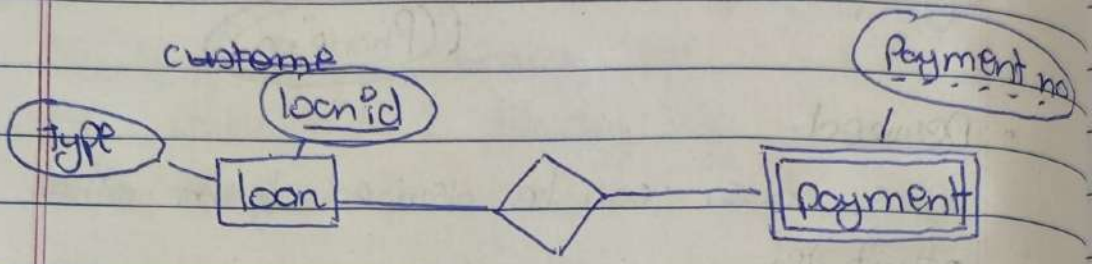
These entities can be uniquely defined.

They are independent.

e.g. student can be defined by studentid

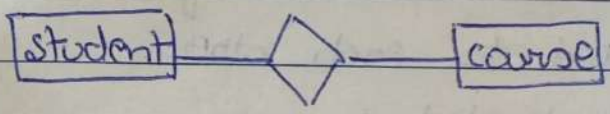
Date.
Page No.

weak entity.
These are dependent on other entities for existence.
e.g.

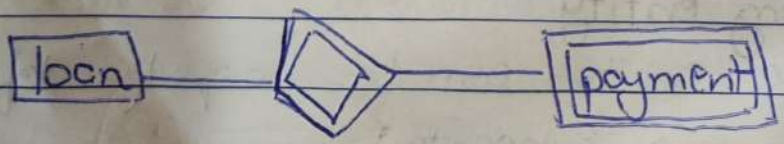


Here, payment depend on loan.
Therefore, they are defined as double rectangle.

strong Relationship
Bothe entities exist independently.
e.g.



weak relationship
When one of the entity depends on other, then it forms weak relation.
e.g.

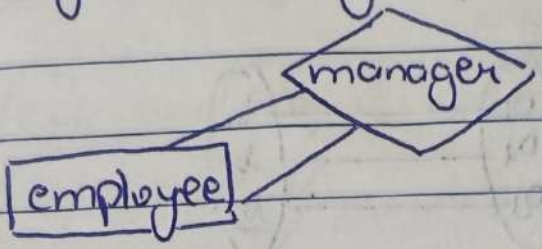


It is denoted by double rhombus.

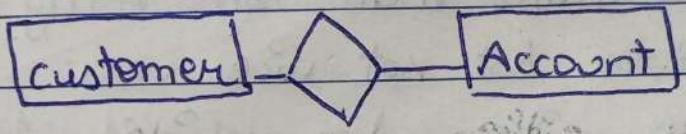
Date.
Page No.

Types of Relationship

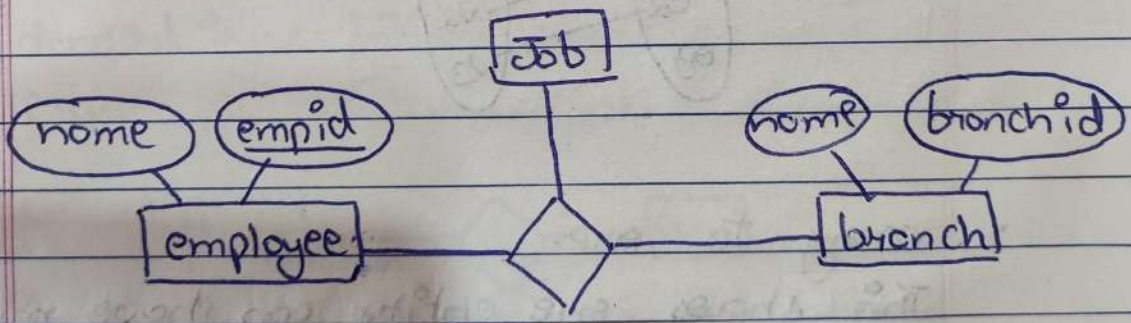
- Unary relation
It has only one entity.
e.g.



- Binary Relation.
It has two entity.
e.g.



- Ternary relation.
It has three entities.

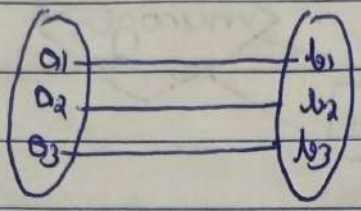


Relationship constraints

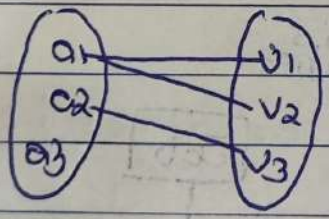
1. Mapping cardinality.
This tells us how many entitys in an entity set are connected to no. of entities of other entity set.

Date.
Page No.

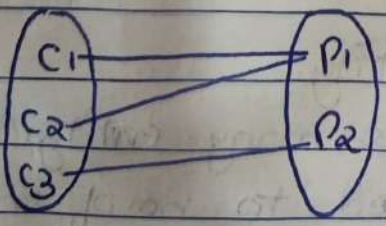
- one to one.
This entity that one entity with exactly one entity.
e.g. citizes has adhaar.



- one to many
This states that one entity can have multip values but not vice versa.
e.g. citizen has vehicle.

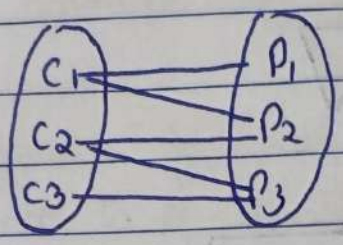


- many to one.
This shaes one entity can have multiple values but not vice versa.
e.g. course by profesor.



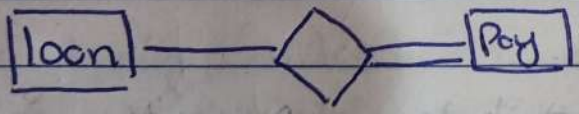
Date.
Page No.

- many to many.
n entities of A can have relation to n entities of B.
e.g. customer buy product

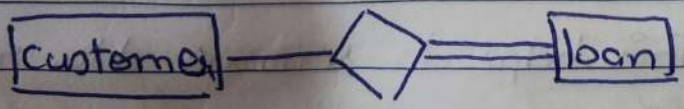


⁂ Participation constraint.
It shows us the minimum cardinality constraint.

- Total participation.
This happens when all entities of entity set participate.
denoted by ═══
e.g.

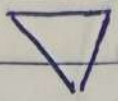


- partial participation.
This shows that not all entities participate.
e.g.

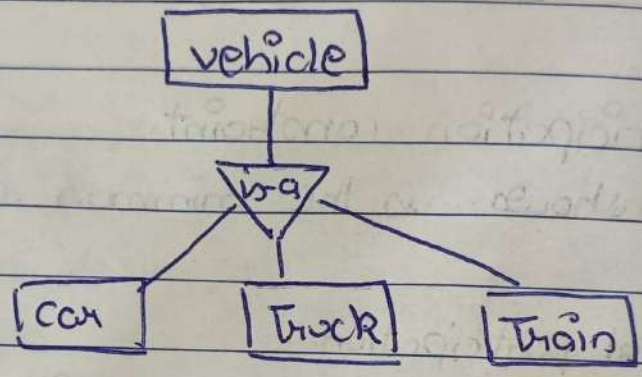


Date.
Page No.

- Specialization
Top-down approach.
We break down larger entities into smaller.
It has 'is-a' relation.
It is denoted



e.g.



why?
To remove redundancy.
It is like inheritance.

Scenario.
Some attributes are comman in above examp
for all three entities like speed.
∴ we remove speed from them and add it
to vehicle.

This helps to remove data redundancy.

Date.
Page No.

- Generalization
Bottom-up approach
We group sub entities into larger entities.
It has 'is-a' relation.
It is opposite of specialization.

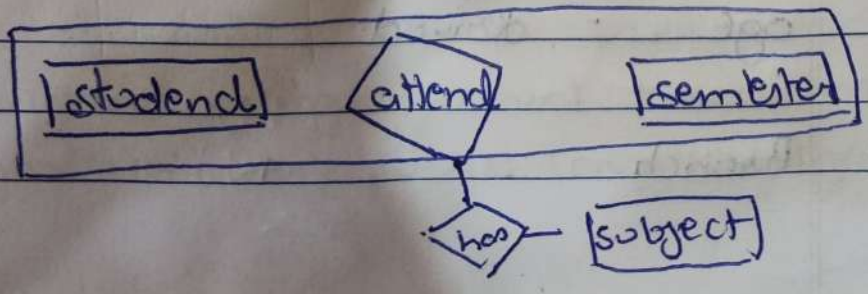
Note ER diagram for both are same, just the difference is of thinking.

if smaller group first then generalization., else specialization.

- Attribute inheritance.
Both specialization and generalization have it.
common attributes are inherited to through parent entity into child entity.

- participation inheritance.
If parent entity involved in a relation then its child also gets involved.

- Aggregation
combining entities relation to a higher entity
This introduces abstraction.



Date.
Page No.

How to draw er diagram.

There are three steps to draw er diagram

- identify the entities
- identify Attributes and their types.
- idetify relationship and constraints
  - Mapping
  - Participation.

1. entities

Banking System

1. enties

customer, branch, employees, saving acc, current account, loan, payment.

2. Attribute and types.

- customer

name - ~~multiva~~ composite
cust id - Primary key, single
address - composite
phone - multivalue
DOB - single
age - derived.

- Branch

Date.
Page No.

branch id - Primary key
assets
liabilities
name - single
city - single

- employees
name - composite
emp_id - Primary key
Phone - single
year of service - derived.
dependent name - multivalued
start date - single.

- saving acc -
account no - Primary key
balance - single
daily withdrawal limit - single
interest - single.

- current acc
account no - Primary key
balance - single
per transaction charge - single.

Note: current and savings account have some common attributes so we can generalize them.

Date.
Page No.

- generalized entity - Account
  account no - account no
  balance - single.

- loan
  loan id - Primary key
  amount - single.

- payment
  payment number.
  payment date
  amount.

Note loan and payment will have weak relationship.

3. Relationships.

- Customer deposit account
  M : N

- customer has loan
  M : N

- loan has payment
  1 : N

Date.
Page No.

- employee manage employee.
  N : 1

loan originated branch
N : 1
=

- Relational Model
  storing the data in the form of tables.

entity are converted to table names.
Attributes converted to columns.

row/tuple define a single entity.
column/Attribute contain a domain specific information.

Note degree of table is no. of columns
cardinality is no. of rows.

steps to draw a Database system.

i) draw er model
ii) convert to relational model
iii) implement RDBMS to relational model

RDBMS : MySQL, Microsoft access, Oracle

Date.
Page No.

- Properties
  unique name for all tables.
  unique name for all columns in a table
  sequence of columns does not matter.
  table must follow consistency constraints.
  Values should be atomic
  each row should be unique.

Keys

- Super key → All the combination of columns that can be used as primary key.
  e.g.

custid	name	email	contact

Super key :- cust id, [name email],
[name, contact], [email, contact]
etc.

- Candidate key.
  subset of super key.
  cannot be null
  do not have columns with redundant type.
  e.g → cust id, [email contact]

Date.
Page No.

∴ we removed name as it is redundant.

- Primary key
  It is a candidate key with least no. of attribute.
  It can not be null.
  It is unique.

  e.g cust id.

Note every Primary key is a candidate key but every candidate key is not a primary key.

- Alternate key
  All combination of candidate key left after chosing primary key.
  ∴ {CK} - PK = AK.

- Foreign key
  It is a primary key that is used to join two tables
  every foreign key is a primary key of other table.

  The table whose primary key is used as foreign key is called Parent/referenced table.

Date.
Page No.

The table who takes primary key as foreign key is called child / referencing table.

- composite key
  Primary key formed by at least 2 attributes

- compound key
  Primary key formed by 2 foreign keys.

- surrogate key
  used to join two tables where nothing can be used in place of PK.
  usually holds an integer.
  can be used as Primary key.

Integrity constraints
These are important to maintain consistency in database when CRUD operations are performed.

- Domain constraints
  To have a check on domain specific column
  e.g. id should be int, age ≥ 18

- entity constraint
  every table should have a PK.
  PK != null

Date.
Page No.

- Referential constraint
- insert constraint
cannot insert in child relation table if not present in parent relation table.

- delete constraint
cannot delete from parent relation table if entry present in child relation table.

Can we not delete then from parent?
Yes we can by using On delete cascade.
This delete the entry related to parent table and child table at once.

e.g. if we delete cust id = 3 from parent relation then it gets deleted from child relation.

can the foreign key be ~~zero~~ Null?
Yes Yes, by using on delete Null.
This do not deletes entry in child table instead set the foreign key value to Null.

Key constraints
These are the constraints that are used to manipulate the data.

Date.
Page No.

i) Not null
ensures that value in a column is not n
By default value of a column is null.

cust_id int NOT NULL.

ii) Unique.
ensure every value is unique in the colum
Their can be multiple unique constraints.

cust_id int,
unique (id)

iii) default
This defines the default value of a column
it is added if no values is specified.

iv) Check.
It is used to apply conditional constraints
e.g.

age int,
check age ≥ 18

v) Primary key.
Set of attributes or single attribute that
uniquely define a table.
Their can only be one Primary key.

Date. 18 June '25
Page No.

vi) Foreign key.
It is used to join two tables.
The Foreign key must be primary key of other entity set.

e.g. Primary key (order ID),
Foreign key (cust_id) Referencing customer (cust ID)

Transformation from ER To RM

i) strong entity
convert to a seperate table.

ii) weak entity
convert to a table.
introduce FK ie primary key of strong entity.
Primary key of this table is FK + PK.

iii) composite attribute.
create seperate table.
All attributes included.
seperate column for every possible attribute
eg → address : streetno, streetname, city, state.

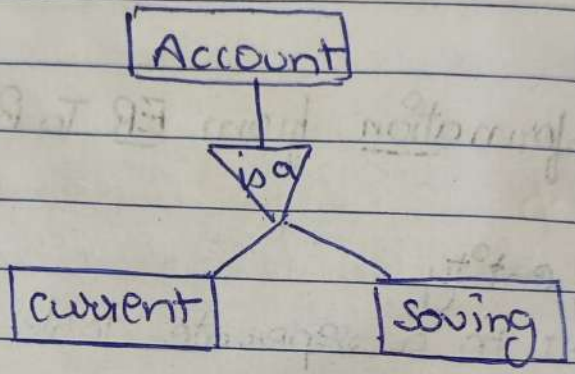
iv) multivalued attribute.
create new table with same name as attribi

Date.
Page No.

introduce FK ie primary key of main table + PK attribute is PK.

ii) Generalization

- Method 1
seperate tables for all entity.
eg. if



Then three tables.
account : acc_no, bal
curr_acc : acc_no, ...
sav_acc : acc_no, ...

- Method 2
create only two tables for lower entity.
curr_acc : acc_no, bal, ...
sav_acc : acc_no, bal, ...

Problem with method 2.
- if an account is both savings and current then redundancy.

Date.
Page No.

- if the account is neither current nor saving then no way to fetch.

vi) Aggregation
convert the relation into table. Take look on fig of aggregation.
All the primary keys of all entities become attribute.
The PK is combination of all attributes ie a composite key.

manages (emp_id, manager_id, Jobid, branchid)

PK → combination of all attributes.

vii) Unary relation.
add another colaam, the FK ie the primary key of same table.
This is for 1 : N

- 1 : 1
same like 1 : N
but primary key is only one attribute ie PK itself.

- M : N
second table of relationship.

Date.
Page No.

include both id ie both are primary key of main table.
These both will act as FK.

Here the PK will be combination of both FK. compound key.

Normalization

Functional dependency.
When one attribute can be determined by other then it is functional dependency.

e.g. $A \rightarrow B$

$\{emp\_id\} \rightarrow \{emp\_name\}$

Here A is determiner
B is dependent

Types.

- Trivial

It states that if $A \rightarrow B$ has Trivial FD if $B \subseteq A$.

$\{emp\_id, emp\_name\} \rightarrow \{emp\_name\}$

Date.
Page No.

- Non Trivial
  it states if $A \rightarrow B$ but $B \not\subset A$
  e.g.
  $\{emp\_id\} \rightarrow \{emp\_add\}$.

Armstrong's Axioms Roles

- Reflexive
  if A is set of attribute and $B \subseteq A$ then
  $A \rightarrow B$.

e.g. $A \rightarrow \{a, b, c, d\}$, $B = \{b, c\}$

$\therefore A \rightarrow B$.

- Augmentation.
  if $A \rightarrow B$ then

  $Az \rightarrow Bz$ is also true.

- Transitivity.
  if $A \rightarrow B$ and $B \rightarrow C$ then. $A \rightarrow C$

Date.
Page No.

Normalization

It is a technique to remove the redundancy in the data to make it Efficient.

Why Normalization?

To remove redundancy.

What if not normalization?

if we do not normalize data the we can encounter anomalies.

- insertion.

We cannot insert data until the all data is fetched.

- deletion

cannot delete data properly.

Deletion of data may cause loss of other data

- updation.

Data updation will require repeating steps again and again.

It also increase size of database.

∴ Decompose tables until SRP is achieved.

Single responsibity Principle.

Date.
Page No.

Types

1NF
It states that their should only atomic values no multivalued attributes allowed.

2NF
table should be in 1st normal form.
Their should not be partial dependency.
all non prime attributes should be fully dependent on prime attribute.

Partial dependency
non prime attribute cannot be determined by part of prime attribute.
eg. if R1 {A, B, C, D}

AB → primary key, then

AB → C ✓
B → C x partial dependency.

3NF
must be in 2NF
No transitive dependency.
non prime cannot determine non prime.

Date.
Page No.

e.g if A(A, B, C) then,

A → B ✓

B → C ✗

- BCNF

Boyce - codd Normal form.

Must be 3NF

prime attribute cannot be on right side.

if A → B then, A must be a super key.

Non prime cannot find prime, or non prime.

Acid Properties And Transactions

Transaction

Amount of work done that includes multiple steps executed in a sequence.

This work should either be successful or not successful.

eg.

A |1000| B |2000|

if A wants to send 100 ₹ to B then

Date.
Page No.

Read (A) → A = A - 100 → write (A) → 900
Read (B) → B = B + 100 → writte (B) → 3100

∴ When all these steps are completed, we say the transaction is successful.

Their are few properties that ensures integrity.

- Atomicity.
Transaction should either be completed or not completed. If system crashes in middle then option to roll back to old state.

- Consistency.
The consistency should be maintained. Therefore the balance sum of A and B should be equal to amount before transaction and after transactio.

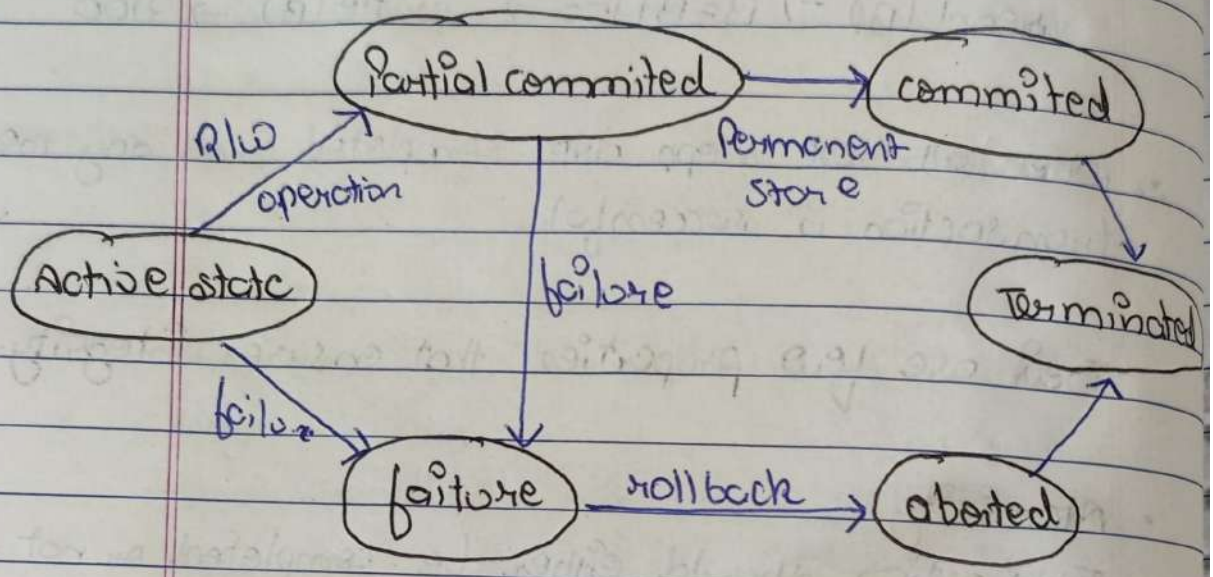
∴ (A+B) before T_1 = (A+B) After T_1

- Isolation.
Their are N number of transaction.
All transactions should be executed one after other.

- Durability.
changes in DB should be permanent.

Date.
Page No.

Transaction states



- Active state.
  First state of transaction
  R/W operations perform.
  if executed then move to Partial commited
  If not then failure.

- Partial committed
  Here the changes are stored in buffer of main memory.
  If changes permanent then to committed state else to failed state.

- Committed state.
  Here changes are mod permanent and then move to terminated state.

- failed state

Date.
Page No.

This is reached if any error is encountered.
Then the steps are rollbacked to original state.

- aborted state
If roll back is achieved then aborted state is reached.

- Terminated state
It is reached either after committed or aborted state.

How to implement Atomicity?

- Shadow copy scheme:
We create a copy of data and perform operations on the copy.
While the db pointer still points on to the old data.
If all the operations are successfully performed then db pointer points to new copy.

If error then new copy is destroyed.

- for next transaction the new copy act as the old copy.

Date.
Page No.

Log recovery Method

In this method we write logs side by side with each step.
logs are written before the execution of steps

- deffered db method
we commit afterwards in this method.
e.g.

read (A)	<to start>
A = A - 50	<to, A, 950>
write (A)	<to, B, 2050>
	<commit>

if transaction fail before commit then logs are destroyed.
if after commit then these are used to redo

- Immediate db method.
In this method we directly write on db and logs are written like

<to start>
<to, A, 1000, 950>
<to, B, 2000, 2050>

Here old value is also stored to recover in case of error.

Date. 19 June '25
Page No.

Indexing

It is a technique to optimize the searching in the table.
It is used to increase the efficiency.

We create an inder file that has a sorted key or non key attributes and entries are divided into batches.

Note Inder file is always sorted.
It is not necessary that disk should be sorted only on PK. non prime attributes can also be used.

Types

- Primary indexing.
It states disk should be sorted on any attribute whether primary or non prime.
eg sorted on basis of roll no.

Index file

1	B_1
11	B_2
21	B_3

Date.
Page No.

Dense index

When every unique value is entered in the index file then it is called dense index.
Also known as clustering index.

clustering index

It is also a primary index but here non prime attribute is used as index.

Sparse index

When only one value is listed in index table representing other values of a block it is called sparse index.
It is usually used with primary attribute

- Multi-level index
  Indexes with two or more levels.
  It is done when index table gets too large
  It is sparse index.

- Secondary index (non-clustering index)
  It is used on unsorted data.
  It is a dense index example.
  It can be done on prime or non prime attribute.
  If values are not unique then we can maintain linked list of base pointers

Date. 20 June '25
Page No.

Index file

1 B1 → B3 → B19
2
3

It is helpful as index file is always sorted.
We cannot apply Binary search on data file but can apply on index file.

Non SQL (Not only SQL)
It stores data other than in tables.
One famous example is JSON format.
Mongo DB uses NoSQL.

Why NoSQL?

- Flexible schema
It provide flexible schema as it does not have any structure.
We can add information that is not complete.

- Horizontal scaling (scale out)
It has important feature of horizontal scaling, ie dividing the data and storing it in multiple servers.

Note SQL use Vertical scaling (scale up)

Date.
Page No.

- Vertical scaling
When upgrades are made on single system to increase the space.
Hardware, RAM are upgraded.

- Fast data retrieval
It helps in fast data accen as all the data can be stored in single file, ie no normalizing Insert and read operations are fast.

∴ Delete and update operations are time taking

Note SQL can use Horizontal scaling but it is not practical.

- High availaibility
data is stored in multiple server due to its auto replication feature. Thus if one server crashes then other server can be used.

- Cloud
More suitable for cloud servers

- caching Mechanism
stores frequently accessed data in a fast acces storage.
e.g chats in social media app.

Date.
Page No.

When to use Nosql?
Huge amount of data
fast development
When scale out feature is required
storage of unstructured data

Types

i) key-value
They have key value pairs. One pair of key and value.
e.g cards in amazon, flipkart

ii) C-store
store data in coleum format

| Dev | Kashipur | 30 | Matt | Pune | 20 | → row store

| Deve | Matt | Kashipur | Pune | 30 | 20 | → coleum store

This is suitable for analytics
Insertion is slow but data access is fast

iii) Document based
MongoDB is an example.
Similar to json format
Multiple values can be stored

Date.
Page No.

e.g e-commerce platform.

5) Graph based.
It is suitable for storing relationships without using joins.
Uses graph structure to store data.
e.g Fb friends structure.

i) Disadvantages of NoSQL

- Data redundancy :. storage high.
- Slow delete and update and costly

Note: MongoDB support acid properties.

Types of Database

- Relational databases.
- NoSQL databases.
- Object oriented Databases.
  based on object oriented programming.
  Used when establishing relationship gets too complex.
  It supports encapsulation, inheritance etc.
  stores information as a class and object is used to access it.

Date.
Page No.

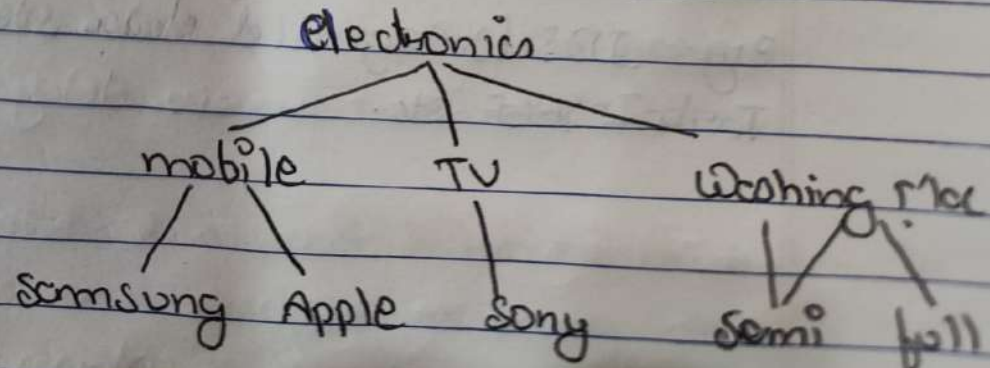
Advantage
Data retrieval is fast.
Can handle complex data relations.
Friendly with real world problems.

Disadvantage
Highly complex
Less community support
Does not support view like relational database.

e.g ObjectDB, GemStone etc.

- Hierarchical Databases.
Tree like structures.
Used when top-down data retrieval is required specific use case.
One child can only have one parent.

e.g. Electronics shop.



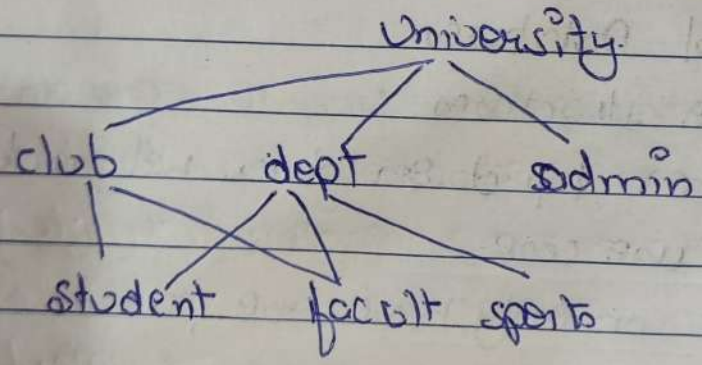
It is easy to use.

Date.
Page No.

easy to use delete and insert.

Disadvantage
not flexible; can only have one parent.
not much community support.

- Network Databases.
Extension to Hierarchal databases.
can have more than one parent.
Graph like structure.
e.g.



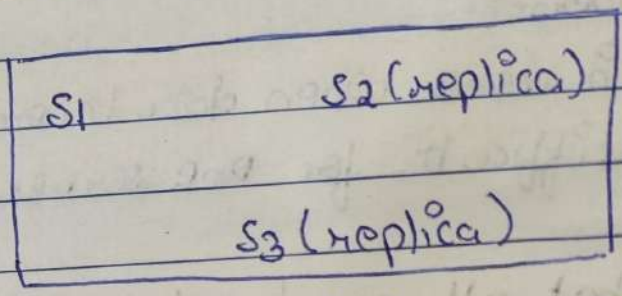
These can handle comple relations.
Not much commity support.
e.g. IDS (Integrated data store), IDMS
TurboIMAGE etc.

Date.
Page No.

clustering

It is also called replica set.
It is a process of combining more than one server that connects a single database.

e.g. There is a server S1 with info about 1000 users, then



clustering.

∴ All three servers contain same information.

Advantage

- Data redundancy
  if one system crashes then the request can be fulfilled by other server.
  It also provid a layer of abstraction.

- load balancing.
  We introduce a load balancer in order to balance the load of incoming request
  It also improves the scalability.

Date.
Page No.

- High availability.
  Due to clustering, the website is always working and request can be fulfilled instantly.

- Partioning
  It is a procen to divide data into multiple servers.
  It is done when data becomes huge and it is difficult for one server to manage it.

What all can be done for optimization?

- Scale up.
  vertical scaling can be done
  - but it increases the cost.

- clustering.
  - It is managed by a main server and replica sets.
  updates are first on main server then to replica sets that causes delay.

- Portioning
  It allows two types-
  - vertical partitioning - column wise
  - horizontal partioning - row wise.

Date.
Page No.

Distributed database.
single logical database spread across multiple servers and logically interconnected by network.

- Sharding
It is a technique to implement horizontal partition
It requires a routing layer to help in sending request to appropriate shards (server).

Advantage
Scalability
Availability.

Disadvantage.
increase complexity cause of routing layer.
Non-uniformity
Not suitable for analytical queries like aggregatio

Note Advantages of Partitioning
Parallelism
Manageability
Reduce cost of vertical scaling
Availability.
Performance.

Date.
Page No.

CAP Theorum

It is one of most important concept in Distrib. databases.

- Consistency
  value of a record must be same across all nodes.

- Availability
  The system should remain operational all the time.

- Partition tolerance.
  When partition occurs, the communication b/w the nodes is lost.
  If system is partition tolerant then it does not fail even after partion.

Note CAP theorum states that distributed system can only provide two of three properties at a time.
∴ their is a trade off between consistency and availability when there a partition.

CA Database: These provide both consistency and availability but their is no partition.
eg. MySQL, PostgreSQL.

Date.
Page No.

CP Database: Consistency, Partition tolerance.
When a partition occurs, the system turns of inconsistent node that contradicts the availability.
e.g. mongo DB., Banking System.

AP database: Availability, partition tolerance.
When partition occurs, these system allows both nodes to run which leads to loss of consistency.

Though, these system ensure consistency after the partition is fixed and nodes get sync.

Eg → Cassandra, Dynamo DB.
App like FB

Master slave Architecture

Their is a master database and many slave databases.
Master DB handles all write operations.
Slave DB handles all read operations.

Once write is performed in master DB, then it is replicated to slave DB.

How replication takes place?

Date.
Page No.

- Asynchronus
This means that if write query is processed in master db at $t=0$, then it will be replicated to slave db in some time.

Let's say it updates in slave 1 at $t=10$ and slave 2 at $t=12$.

∴ if any read query arrives between $t=0$ - $t=$ that then it will show old value.

- Synchronus.
This states that after the value is updated in all the slave DB. only after that the request is accepted.

Advantages
Scale out read request
Availability
parallelism
backup if master DB fails.