18/8/25

UNIT - I

- A database management system (DBMS) is a collection of interrelated data & set of programs to acces those data.
- The collection of data is referred to as the database which contains information related to an enterprise.
- The goal of DBMS is to provide a way to store & retrieve database info that is both convinient & efficient.
- DB systems are designed to manage large bodies of Info.

- Applications of database system -

Databases are widely used such as follows:-

a) Enterprise Information -

- Sales : for customer, product & purchase info.
- Accounting : for payments, recieipts, account balances, assets and etc.
- Human resources : for info abt employees, salaries, payrolls and benefits.

b) Banking & finance -

- Banking : for customer info, accounts, loans & transaction
- credit card transaction : for purchase on credit cards &

generation of monthly stmts.

- finance - for storing info abt holdings, sales & purchases of financial instruments such as stocks & bonds.

c) Universities - for student info, online registration and grades.

d) Airlines - for reservations & schedule info. Airlines were among 1st to use databases in geographically distributed manner.

e) Telecommunications - for keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards & storing info abt communication networks

- Purpose of databases -

a) Data sharing across locations :-
A DDS allows users & applications at different sites to access and share data as if it were stored in single location.

b) Transparency :-
Even though data is stored across multiple sites, the DB provides location transparency (users dont need to know where data physically resides)

c) Improved reliability & Availability -

By storing copies of data (replication) at different sites, DDS ensures that if one site fails, data can still be accessed from another.

d) Performance Enhancement -

Data can be placed closer to site where it is most frequently accessed, reducing response time.

e) Scalability -

New sites (database/servers) can be added easily without disrupting existing system.

f) Autonomy of local databases -

Each site can maintain level of local control over its database while still being part of distributed system.

g) Cost Effectiveness -

Instead of relying on one large central computer, DDS can use multiple smaller, cheaper computers connected over the network.

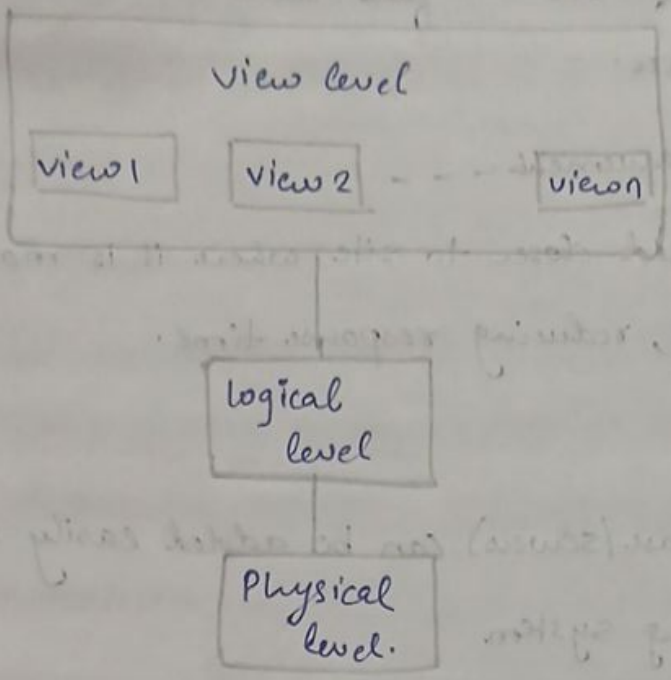
- Views of Data -

A DB system is a collection of interrelated data & set of programs that allows users to access and modify these data.

A major purpose of DB system is to provide users with abstract view of data; i.e., system hides certain details

of how data are stored and maintained.

2) Data Abstraction -



Physical level -

The lowest level of abstraction describes how data are actually stored.

logical level -

The next higher level of abstraction describes what data are stored in database, and what relationships exist among those data. This is refered to as physical data independence. DB administrators who must decide what info to keep in database, use logical level of abstraction

View level -

The highest level of abstraction describes only part of entire database. Even though logical level uses simpler

structures, complexity remains because variety of information stored in large database.

b) Instances and schemas-

Databases change over time as info is inserted & deleted. The collection of info stored in database at a particular moment is called as instance.

The overall design of database is called the database schema.

The physical schema describes database design at the physical level,

while logical schema describes the database design at the logical level.

A DB may also have several schemas at view levels Sometimes called subschemas, describes different views of database.

c) Data Models-

A collection of conceptual tools for describing data, data relationships, semantics & consistency constraints. A data model provides way to describing design of database at the physical, logical & view levels.

- Relational model-

uses collection of tables to represent both data & the relationships among those data. Each table has the multiple columns & each column has a unique name.

Tables are also as relations.

The relational model is example of record based model.

- Entity relationship Model-

Uses collection of basic objects called entities & relationships among these objects. An entity is a thing/object in real world.

- Object based data model-

object oriented programming (Java, C++, or, C#) has become dominant s/w development methodology with notations of encapsulation, methods (functions) and the object identity.

- Semi Structured data model-

Permits specification of data where individual data items of same type may have different sets of attributes. The XML i.e; extensible markup language is used to represent semi structured data.

- Historically network data model & the hierarchical data model preceded the relational data model.

- Database languages-

A database system provides a data definition language to specify the database schema and a data manipulation language to express database queries & updates.

Data Manipulation Language-

- A language that enables users to query or manipulate data. The types of query are as follows:-

a) Retrieval of info
b) Insertion of new info
c) Deletion of info
d) Modification of info.

Retrieval of data (SELECT)-

- used to fetch data from DB table
- Example:-

```
SELECT name, age FROM students WHERE age > 18;
```

Insertion of data (INSERT)-

- Adds new records (rows) into a table.
- Example:-

```
INSERT INTO Students (id, name, age) VALUES (1, 'Rahul',20
```

Deletion of data (DELETE)-

- Remove existing records from table.
- Example:-

```
DELETE FROM students WHERE id = 1;
```

Modification of data (UPDATE)

- changes values of existing records in table.
- Example:-

```
UPDATE Students SET age = 21 WHERE id = 1;
```

features of DML-

- works only on data not on DB structure.
- Can be of two types :-

  Procedual DML - specifies how to get data

  Non-procedual DML - specifies what data is required.
- Helps in maintaining & manipulating real time info. in application.

Data Defination Language-

- A part of SQL used to define, create, modify and delete the structure of database objects such as table, schema, indexes & views. This deals with DB schema not the actual data.

commands of DDL -

a) CREATE -

used to create new database objects (Tables, views, indexes).

Example :-

```
CREATE TABLE students (
    id INT Primary key,
    name VARCHAR (50);
    age INT.
);
```

b) ALTER -

~~Deletes~~ used to modify existing DB object.

Example :-

ALTER TABLE students ADD column email VARCHAR(100);

c) DROP -

Deletes an existing database object permanently

Example :-

DROP Table students;

d) TRUNCATE -

Remove all records from table but keeps the structure.

Example :-

TRUNCATE TABLE students;

e) RENAME -

used to rename a database object:

Example :-

RENAME TABLE students to learners;

Features of DDL :-

- Defines schema & structure of DB.
- Automatically commits changes cannot be rolled back in most cases.
- Mostly executed by database administrators during setup.

Relational Databases-

A relational DB is based on relational model & uses a collection of tables to represent both data & relationships among those data.

It also includes DML and DDL.

Tables -

Each Table has multiple columns and each column has a unique name

The relational model is an example of record based model. Record based models are so named because the database is structured in fixed format records of several types. Each table contains records of a particular type. Each record type defines fixed no. of fields / attributes. The columns of table correspond to attributes of record type.

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci	75000
98345	Kim	Elec. Eng	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000

Instructor table

76543 Singh finance 80000

Department table

dept_name	building	budget
comp. sci	Taylor	10 0000
Biology	watson	90000
Elec. Eng	Taylor	85000
Music	Packard	80000
finance	Painter	120000
History	Painter	50000
Physics	watson	70,000

Data Manipulation lang-

The SQL query lang is non procedural. A query takes as input several tables (possibly only one) and always return a single table.

Eg:-

```
Select instructor.name
from instructor
where instructor.dept_name = 'History';
```

This query specifies those rows from table instructor where dept_name is History must be retrieved & name attribute of these rows must be displayed.

Data defination language-

SQL provides a rich DDL that allows one to define tables integrity constraints, assertions etc.

Example:-

```
CREATE table department
( dept_name   char(20),
  building    char(15),
  budget      numeric(12, 2));
```

Database Access from Application Programs.

SQL is not as powerful as universal turing machine there are some computations that are possible using general purpose programming lang but not possible using SQL.

SQL also does not support actions such as I/p from users, output to displays or communication over network. Such computations & actions must be written in host lang such as C, C++ or Java.

Application programs are programs that are used to interact with the database in fashion.

Database Design-

A DDS stores data across multiple sites connected by a network. The goal of database design in DDS is to make system:-

- Efficient (fast access to data)
- Reliable (no data loss)
- Transparent (users feel like its one single DB not many)

Steps in DB design :-

a) Requirement Analysis-

- collect user and application needs.
- Identify what data is frequently accessed & where it should be located.

b) Conceptual Design-

- creates global schema using ER model or UML
- Represents whole system as if it were centralized.

c) Fragmentation-

- Breaks large relations (tables) into smaller pieces for the distribution.
  - Horizontal fragmentation - Divide rows.
  - Vertical fragmentation - Divide columns.
  - Hybrid fragmentation - combination of both.

d) Replication -

- Stores copies of fragments at multiple sites.
  - Full Replication - Every site has full copy.
  - Partial Replication - only frequently used fragments are copied.
  - No Replication - Each fragment stored at only one site.

e) Allocation -

- Decide at which site each fragment / replica will be stored
- factors :-
  - frequency of access
  - communication costs.
  - Reliability requirements.

f) Schema Integration -

- integrate local schemas into global schema.
- Ensure naming consistency, avoid conflicts & provide a unified view.

g) Transparency requirements -

- A good DDS design ensures:
  - location transparency - users dont need to know where data is stored.
  - Replication transparency - users dont see multiple copies.
  - Fragmentation transparency - users query as if its a single table

Data Storage & Querying-

The DB system is divided into two components such as

a) Data Storage Manager

b) Query Manager.

- Data Storage Manager-

- Also known as database control system.
- Generally a program that provides interface how data stored and queries recieved.
- Helps to maintain integrity & consistency of database by applying constraints.
- Translates various DML stmts into low level commands.

i) Authorization & Integrity Manager-

The main purpose is to ensure the satisfaction of integrity constraints and checks the authority of users to acces info.

ii) Transaction Manager-

To ensure that even after system failures, the database should remain in a uniform state.

iii) File Manager-

To manage allocation of space on disk storage.

iv) Buffer Manager-

To fetch data from disk storage into the memory mgmt

- Query Manager
Software modules contains the three components as follows:-

i) DDL Interpreter-
- Data definition language interpreter.
- used to build & modify structure of tables & other objects in tables.
- Interprets DDL stmts & records the definition in the data dictionary.

ii) DML compiler-
- Data Manipulation language compiler.
- used for inserting, updating & deleting data in a database.
- Also performs query optimization.

iii) Query Evaluation Engine-
- Executes low level language instructions, generated by DML compiler.

Transaction Management-

Transactions are set of operations used to perform a logical set of work. A transaction usually means that data in database has changed.

Properties of transaction -

Transaction mgmt in DDB ensures that database transactions are executed reliably & follows ACID Properties Atomicity, consistency, Isolation and Durability. These Principles help maintain data integrity even during failures.

for transaction to be performed in DBMS, it must follows several properties called ACID properties.

A : Atomicity

C : consistency

I : Isolation

D : Durability.

a) Atomicity :-

- States that all operations of transaction takes place at once if not the transactions are aborted.
- There is no midway i.e., transaction cannot occur partially.
- Involves two operations

  abort

  commit

b) Consistency -

- The rules that keep the database accurate & consistent are followed before and after transaction.

- When a transaction is completed it leaves database either as it was before or in new stable state.

c) Isolation-

- It shows that data which is used at time of execution of transaction cannot be used by second transaction until the first one is completed.

d) Durability-

- used to indicate peformance of databases consistent state. It states that transaction made the permanent changes.

Atomicity - A transaction is all or nothing

Consistency - A transaction must bring database from one valid state to another valid state.

Isolation - Multiple transactions happening at same time must not interfere with each other.

Durability - once transaction is committed, it remains permanent, even if there's system crash or power failure.

Data Mining and Analysis :-

-Data Mining

Refers to discovering useful patterns, trends and knowledge from large sets of data.

In DDS, data is spread across multiple sites / locations.

Data Mining in DDS involves:-

a) Data collection & integration - combing data from the different sites.

b) Preprocessing - cleaning, filtering, removing duplicates, handling missing values.

c) Pattern discovery - applying algorithms (classification, clustering, association rules etc).

d) Knowledge sharing - Sending results back to user.

-Data Analysis :-

This is refers to examining & interpreting the mixed data to make decisions.

In DDS, analysis has some unique challenges :-

- Heterogeneity - Data may be in different formats.
- Distribution - Data stored across multiple sites, so queries must be optimized.
- Scalability - Handling very large amounts of data.

Techniques used:-

a) Query optimization - finding most efficient way to retrieve distributed data.

b) Data aggregation - collecting summary info

c) Statistical & predictive analysis - Applying models like regression, forecasting.

d) OLAP in DDS - for multi-dimensional analysis.

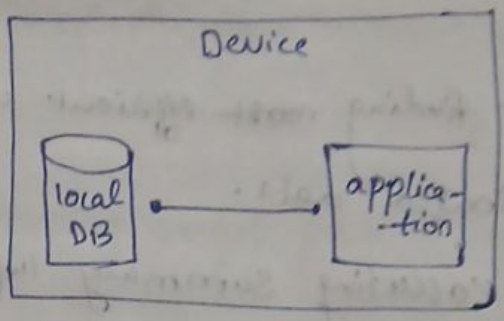
Database Architecture -

This defines how users interact with database to read, write or update information.

The types of DBMS architecture as follows:-

a) 1-tier

- user works directly with database on same system
- Means client, server and database are all in one application.
- common example is Microsoft Excel.
  (Everything from user interface to logic & data storage happens on same device.)
- The setup is simple & easy to use, making it ideal ~~and~~ for personal & standalone applications



Advantages:-

- Simple Architecture - Simple to setup, as only single machine is required to maintain it.

Cost Effective - No additional hardware is required for implementing

Easy to implement - can be easily deployed and used in small projects.

Di's advantages:-

limited to single user - only one person can use the application at a time.

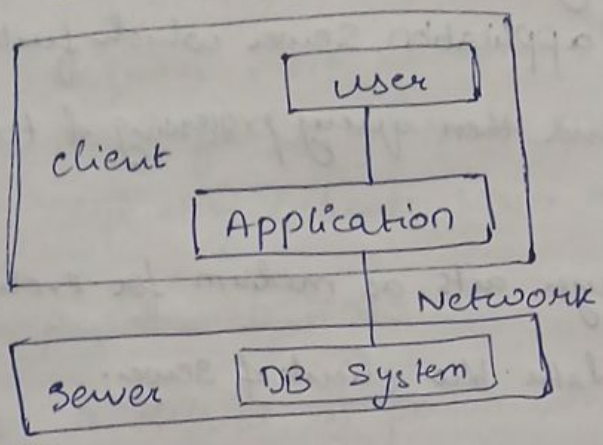
Poor security - Since everything is on same machine, if someone gets access to system they can access both data & application easily.

No centralized control - Data is stored locally, so there's no central DB.

Hard to share data - Sharing data btw users is difficult because everything is stored on one computer.

b) 2-tier-

- Similar to basic client server model.
- Application at client end directly communicates with database on server side.
- Server side is responsible for providing query processing & transaction mgmt functionalities.



client (Tier 1): This is the user interface that users interact with.

Database layer (Tier 2): The DB server stores all the records

Advantages:-

Easy to access - Makes easy access to the DB which makes fast retrieval.

Scalable - we can scale DB easily, by adding clients / upgrading hardware.

Disadvantages:-

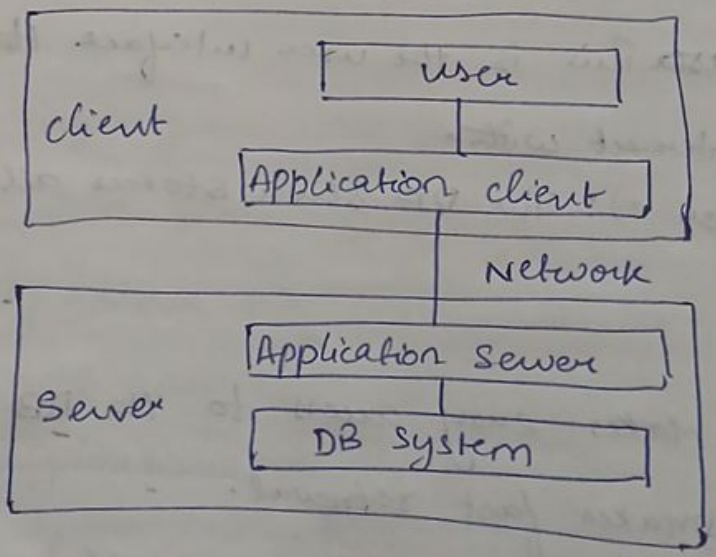
Security Issues:- clients connect directly to DB, which can make system more vulnerable to attacks or data leaks.

Tight coupling - client & server are closely linked. If DB changes, client application often needs to be updated too.

c) 3 - tier -

In this, there is another layer b/w client & server. The client doesn't directly communicate with server. Instead, it interacts with application server which further communicates with DB system and then query processing & transaction mgmt takes place.

This intermediate layer acts as medium for exchange of partially processed data b/w client & server.



Advantages :-

Enhanced scalability - Scalability is enhanced due to distributed deployment of application servers.

Data integrity - Since there is middle layer b/w client & server, data corruption can be avoided.

Security - Prevents direct interaction of client with server

Disadvantages -

Difficult to interact : It becomes difficult for this sort of interaction to take place due to presence of middle layers.

Slower response time :- Since request passes through an extra layer, it may take more time to get response compared to 2 tier.

Higher cost - Requires more hardware, s/w and skilled people for setting up & maintaining 3 layers seperately (client, server and DB).

Database users & administrators -

Database user is defined as person who interacts with data daily for updating, reading and modifying the given data.

There are 4 different types of DB System users:

a) Naive users :-

- Also known as Parametric End users.
- Are the unsophiscated users who lacks the DBMS knowledge but frequently use database applications in their daily life to get the desired results.
- Example :- Railway ticket booking users.

b) Application Programmers -

- Also known as system analysts or s/w engineers
- who write backend code for application programs.
- They use languages such as C, visual Basic, COBOL etc to design test & maintain ~~prog~~ programs.

c) Sophiscated users -

- Are like engineers or analysts interact directly with DB using SQL queries.
- They dont write full application code but use ~~the~~ query processor to anes and manipulate data.

d) Specialized users -

- Are sophiscated users who write specialized database application that doesnt fit into traditional data-processing framework.

Database Administrator - (DBA)

A person who has such central control over system is called database administrator. The functions of DBA includes :-

a) Schema definition - DBA creates original DB schema by executing set of data definition stmts in DDL.

b) Storage structure & anes method definition.

c) Schema & physical organization modification :- DBA carries out changes to schema & physical organization to reflect changing needs of organization.

d) Granting of authorization for data aness :- By granting different types of authorization, the DBA can regulate which parts of DB various users can aness.

e) Routine Maintenance :-

Examples of routine maintenance activities are :

- Periodically backing up the DB.
- Ensuring enough free disk space is available for normal operations.
- Monitoring jobs running on DB & ensuring that performance is not degraded

Relational Model -

- This model organizes data using tables (relations) consisting of rows and columns.
- The relational model represents how data is stored & managed in the relational databases, where data is organized into tables each known as relation.
- Each row of table represents an entity or record;
  Each column of table represents particular attribute of that entity.

Terms in relational model:-

a) Attribute - properties that define an entity.

b) Relation Schema - defines structure of relation & represents name of relation with its attributes.

c) Tuple - represents row in a relation; each tuple contains set of attribute values that describe particular entity.

d) Relation Instance - set of tuples of relation at particular instance of time is called relation instance.

e) Degree - No. of attributes in relation is known as degree of relation.

f) Cardinality - No. of tuples in relation is known as cardinality.

g) NULL values - value which is not known or unavailable.
Represented by NULL.

Advantages:-

a) RM is simple & easy to use in comparision to other langs

b) RM is more flexible than any other RM

c) RM is more accurate in relational data model.

d) Its ~~pp~~ betta to perform operations in RM.

e) Data is more accurate in relational data model.

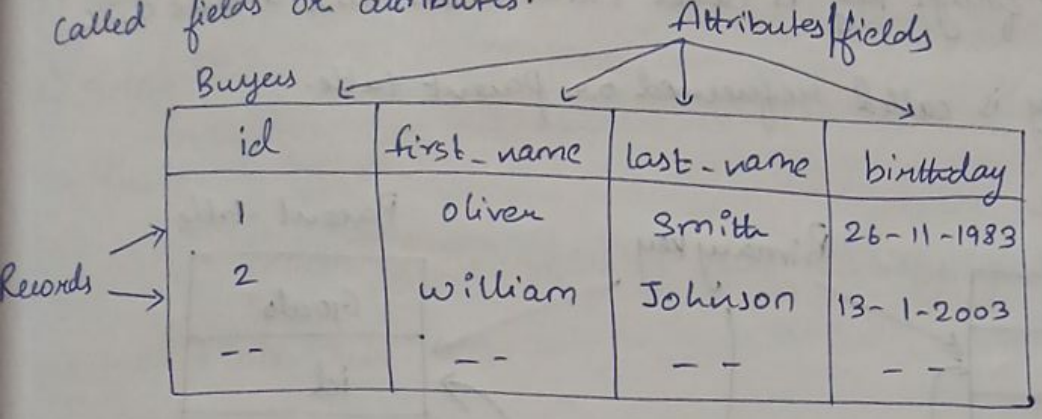
Disadvantages:-

a) RM can experience performance issues with very large DB

b) Extensive use of normalization can result in complex queries & slower performance.

Structure of relational databases:-

- A relational database consists of collection of tables, each of which is assigned a unique name.
- The table consists of rows called records and columns called fields or attributes.



- In each table, each column has predetermined datatype:-

for eg:-

- VARCHAR (string datatype)
- INTEGER (numeric datatype)
- DATETIME (date & time datatype)

and others.

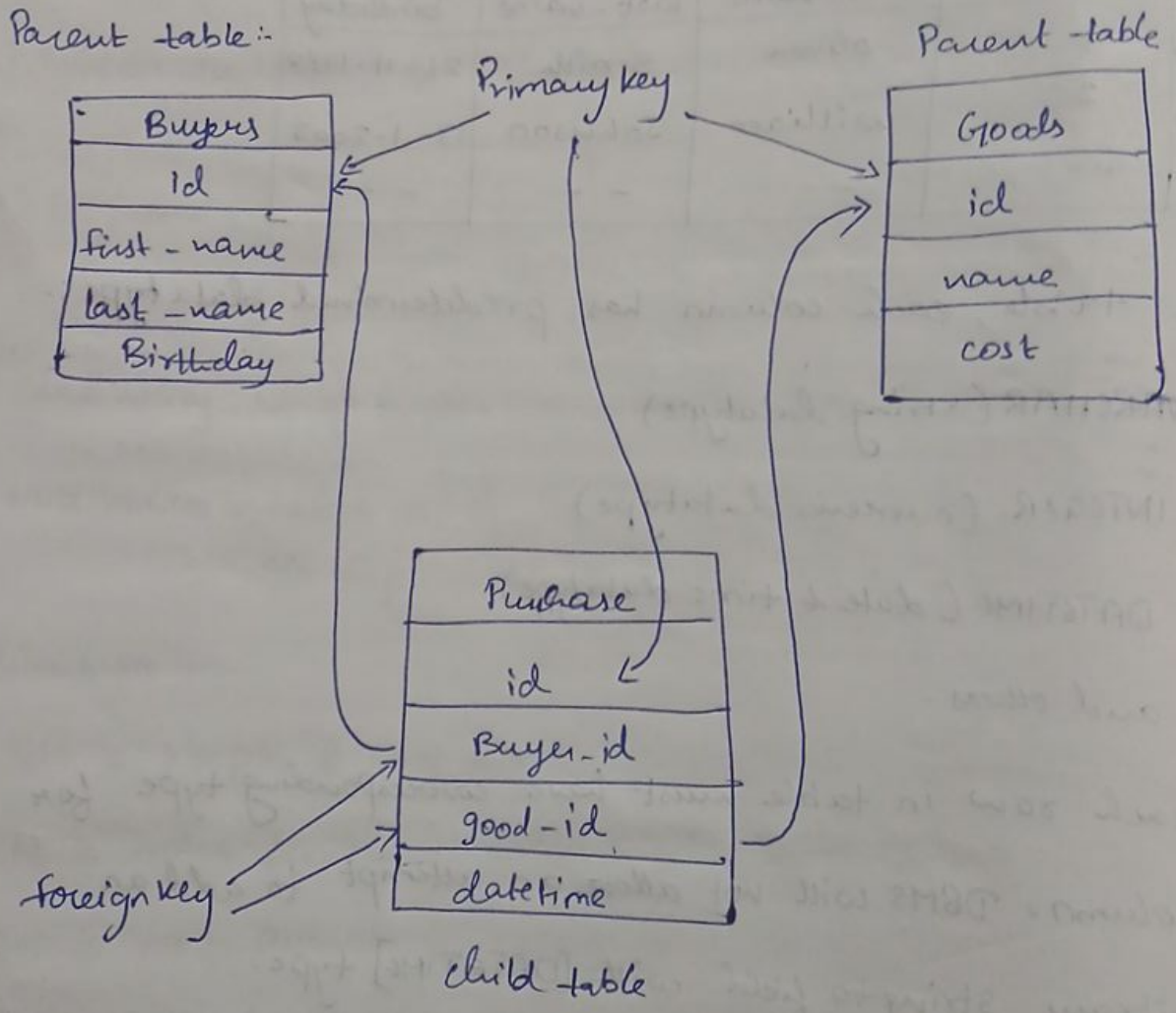
- And each row in table must have corresponding type for each column. DBMS will not allow an attempt to add an arbitrary string to field with [DATETIME] type.
- Primary key-

A key field is a field whose values uniquely identifies a record in the table.

- foreign key :-

A foreign key is field in one table that refers to the primary key in the another table.

- Table with foreign key is called child-table and a table with primary key is called referenced or parent table.



Fundamental Relational-Algebra operations-

Relational algebra is procedural query language in DB systems that uses set of operations to manipulate relations (tables). The operations are as follows :-

a) selection -

- Denoted by σ
- Selects rows from relation that satisfy given condition.
- works on rows.
- Notation :

$$\sigma_{\text{condition}}(\text{relation})$$

b) Projection -

- Denoted by Π
- Selects certain columns from a relation.
- works on columns.
- Notation :

$$\Pi_{\text{attribute-list}}(\text{Relation})$$

c) Union -

- Denoted by $\cup$
- combines tuples from two relations, removing duplicates.
- works on two relations with same set of attributes.
- Notation :

$$R \cup S$$

d) Set Difference -

- Denoted by -
- finds tuples in one relation but not in other.
- works on two union compatible relations
- Notation :-

$$R - S$$

e) Cartesian Product -

- Denoted by x
- combines each tuple of one relation with every tuple of another.
- works on Any two relation
- Notation :-

$$R \times S$$

f) Rename -

- Denoted by ρ
- changes name of a relation or its attributes.
- works on any relation.
- Notation :-

$$\rho(\text{new-name}, \text{attribute-list})(\text{Relation}).$$

Additional relational algebra operations -

- Also known as derived operations.
- Types of operations :-

a) Intersection -

- Denoted by ∩.
- Gives tuples present in both relations.
- Notation :

  $R \cap S$

b) Join -

- Denoted by ⋈.
- combines related tuples from two ~~tuples~~ relations based on condition.
- Notation :

  $R \bowtie_{Condition} S$

- Types of Join :

  Theta Join - uses a condition with "<, >, ≤, ≥, =".

  Equi Join - condition uses only "=".

  Natural Join - automatically matches attributes with same name.

c) Division -

- Denoted by ÷
- Returns tuples from one relation that are associated with all tuples of another relation.
- Notation :

  $R \div S$

d) Assignment -

- Denoted by ←
- Stores result of an expression in new relation
- Notation:

temp ← σ(age > 18)(student).

e) Outer joins -

- Variant of join that preserves tuples even if no match is found
- Types:

left outer join :-

Denoted by ⟕

Keeps all tuples from left relation.

Right outer join :-

Denoted by ⟖

Keeps all tuples from right relation.

full outer join :-

Denoted by ⟗

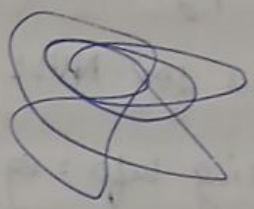
Keeps all tuples from both relation

Extended Relational-algebra operations-

These are not part of primitive algebra but are extensions added to support grouping, aggregation & more

a) Aggregate functions -

- Perform calculations on set of values.



- functions:
  SUM, AVG, MIN, MAX, COUNT

- Notation:
  attributes agg (Relation)

b) Generalized projection-

- Extends projection by allowing arithmetic expressions and renaming.

c) outer union-

- Extends union by allowing relations with different sets of attributes.

d) Extended Division -

- Similar to division but often used with aggregation/grouping.
- finds entities related to all items in another set.

NULL Values -

- NULL means missing, unknown or inapplicable value in a database.
- It is not zero, not empty string, not blank space, simply means "value not available".
- Reasons for NULL values
  - Missing info : eg. phone no not yet provided
  - Not application : eg. spouse name for an unmarried person
  - unknown at present : eg, marks of student not yet entered.

- Properties of NULL -

- NULL is special marker, not data value.
- Any arithmetic / comparision with NULL results in UNKNOWN

  eg :- (5 + NULL) = NULL

  (age ≠ NULL) → UNKNOWN

- Handling NULL -
  - IS NULL / IS NOT NULL is used to check for NULL values.

    eg :-

    SELECT * FROM Student WHERE PHONE is NULL;

  - Aggregate functions in SQL handle NULL by ignoring them except (COUNT)

    eg :-

    AVG (salary) ignores NULL salaries.

Problems with NULL-

- Makes query results harder to interpret.
- Complicates comparision operations.
- Can affect aggregate results.

Modifications of databases :-

Modifications of databases means operations that change the actual data stored in the database. There are 3 main types of modifications in relational databases.

a) Insertion - (adding tuples)
- Adding new tuples (rows) into table.
- Example :-

```
INSERT INTO Student (id, name, age) VALUES
                    (1, 'Amit', 20);
```

b) Deletion - (removing tuples)
- Removing tuples (rows) from table.
- Example :-

```
DELETE FROM Student WHERE id = 1;
```

c) Update (Modification) - (Modifying attribute values).
- Changing values of existing tuples (rows).
- Example -

```
UPDATE Student SET age = 21 WHERE id = 1;
```

History of DB Systems -

1. File Based Systems (Before 1960's)
   - Data stored in flat files.
   - No standard access methods.
2. Hierarchical & Network DB's (1960's - early 1970's)
   - Developed to manage large business data.
   - Hierarchical model (1968)
     - Data organized in tree structure (parent - child)
   - Network model (1971)
     - Data represented as records with links (graph structure)
3. Relational DB's (1970's - 1980's).
   - Proposed by E.F. Codd in 1970
   - Data represented in tables.
   - Based on relational algebra & calculus.
   - SQL developed in 1970's.
4. Object oriented & object relational DB's (1990's)
   - To support complex data (images, multimedia)
   - Integrated features of objects with relational systems.
5. NO SQL DB's (2000's - Present)
   - Needed for big data, web applications & unstructured data.
   - Schema less, highly scalable, handle distributed data.
6. New Trends (Present - Future).
   - New SQL - combines Scalability of NoSQL - In Memory DB's
   - Cloud DB's - AWS RDS, Azure SQL - AI integrated DB's

20/8/25

UNIT - II

Query Processing refers to range of activities involved in extracting data from a database. These activities include parsing & translation of queries, a variety of query-optimizing transformations and actual evaluation of queries. (converts high level query to low level execution plan)

The steps involved in processing a query

a) Parsing and Translation

b) optimization

c) Evaluation.

Parsing & Translation -

- The query is parsed (check for syntax & semantics)
- DBMS translates it into an internal representation

Optimization - (Picks the cheapest / best plan)

- Several equivalent relational algebra expressions are possible
- The optimizer chooses most efficient strategy based on cost estimation.

Evaluation - (The chosen plan is run by the engine)

- The chosen execution plan is run by the evaluation engine
- Data is retrieved from storage & results are returned to user.

Flowchart of Query processing :-

Query → Parser & Translator → optimizer → Execution → o/p.

Measures of Query cost -

When DBMS chooses how to run a query, it estimates cost of different query plans.

The main measures are as follows :-

a) Disk Access Cost (I/O cost) - (Time to read/write data from disk)
- Most important measure.
- Refers to no. of disk block transfers needed.
- Since disk access is much slower than CPU, this dominates cost.

b) CPU Cost - (Time for processing)
- Time taken for operations like parsing, comparisions, joins, sorting, hashing.
- usually smaller than I/O cost, but important for large in-memory operations.

c) Communication cost - (Data Transfer time in distributed systems)
- Cost of transferring data across network.
- very high compared to CPU/disk, so it must be minimized.

d2) Memory usage - (RAM needed during execution)

- Amount of RAM needed for query execution
- High memory usage may slow down other processes.

Selection operation -

- This selection operation is used in relational algebra
- To filter rows (tuples) of relation (table) based on the given condition.
- Produces a horizontal subset of relation because it only reduces rows not columns.
- Notation :-

$$\sigma_{condition}(Relation)$$

where; σ → Symbol for selection

condition → predicate (eg: Age > 18)

R → Relation (table).

- Example :-

Relation : Student (ID, Name, Age, Marks)

Query : find students with marks > 80

$$\sigma_{marks > 80}(Student)$$

O/P : Only those student rows where marks > 80 are returned.

- Reduces rows, not attributes :- O/P has fewer or equal rows than I/P, but same no. of columns.
- use in Query processing :-
  - Acts as filter to retrieve only required tuples
  - Often combined with projection (π) for final result query

Sorting-

- Sorting means arranging records of table in the ascending/descending order based on one or more attributes / columns.
- This is often used in query processing, especially for ORDER BY, grouping and duplicate elimination.
- The most common method is external sort merge.
- Phases of external sort merge.

a) Run Generation :-
- Also known as sorting phase.
- Breaks relation into chunks that fit into memory.
- After this step, we have many small sorted runs on disk.

b) Merge ~~sort~~ phase-
- Repeatedly merge runs until single sorted file remains.
- uses N-way merging : read one block from each run into memory, pick smallest tuple, O/P it and replace it with next tuple from same run.

- After all merge passes - final sorted relation.

Join operation -

- A join operation combines related tuples (rows) from two tables into a single table.
- It is based on common attribute.
- Joins are very common in SQL queries.
- Types of joins -

a) Theta Join -

- General form of join
- combines tuples from two relations R and S whenever they satisfy a condition θ.
- Notation :-

$$R \bowtie_\theta S$$

b) Equi Join -

- special case of theta join where condition is equality (=).
- Removes matching tuples where attribute values are equal

c) Natural Join -

- A type of equijoin, but automatically joins on all attributes with same name.
- Removes duplicate columns.
- Denoted by $\bowtie$.

d) Outer join -

- unlike natural / equi-join, outer joins also keep unmatched tuples by filling them with NULL.
  - left outer join : keeps all tuples from left relation
  - Right outer join : keeps all tuples from right relation
  - Full outer join : keeps all tuples from both relations.

e) Semi Join -

- Returns tuples from one relation that match with tuples in the other relation.
- Only attributes of R are kept.

Other operations -

a) Duplicate Elimination :-

- In SQL, by default, queries may produce duplicates
- Eg :-

```
SELECT DeptID FROM Student;
```

- To remove duplicates → use DISTINCT

b) Projection :-

- Selecting only certain columns of the table
- Eg :-

```
SELECT name, DeptID FROM Student;
```

c) Set operations (Union, Intersection, Difference).

- SQL supports set operations : UNION, ~~INTERSECT~~ INTERSECT, EXCEPT

- Eg :-

```
SELECT name FROM Student
UNION
SELECT name FROM alumni;
```

d) Outer Joins :-

- Outer joins keeps unmatched tuples too, filling with NULLS

- Types :-

  Left outer join : keeps all tuples from left relation

  Right outer join : keeps all tuples from right relation.

  full outer join : keeps all tuples from both sides.

e) Aggregation :-

- SQL provides aggregate functions : SUM, AVG, COUNT, MAX, MIN

- Eg :-

```
SELECT DeptId, AVG(Salary)
FROM Employee
GROUPBY DeptId;
```

Evaluation of expressions -

Meaning :-

- In DBMS, queries are written in high level languages
- But internally system must evaluate these queries efficiently.
- The process of finding best way to compute a relational algebra expression is called expression evaluation.

Expression Tree :-

- Queries are represented using expression tree
- leaves → relations
- Internal nodes → relational operators

Steps in evaluation :-

- Parse query - convert SQL to relational algebra
- Generate expression tree - logical representation
- Choose evaluation strategy - Decide which algorithms / operators to use
- Execute - perform operations in order (usually bottom-up from leaves)

Techniques of evaluation.

- Materialization : compute each operation step by step and store results in temporary tables.
- Pipelining : Pass results of one operation directly as input to the next.

Query optimization-

- This is the process of selecting most efficient query execution plan from multiple logically equivalent expressions of query.
- Goal :- Minimizes query cost (I/O, CPU, memory, response time)
- Types of query optimization:-

a) Heuristic (Rule-based) optimization-

- uses predefined rules / hemistics to rearrange queries for efficiency.
- Example rules:-
  - Performs selections early to reduce size of intermediate results.
  - Performs projections early.
  - Replace cartesian product + selection with join.

b) cost based optimization-

- Estimates cost of different execution strategies.
- Chooses one with minimum estimated cost.
- uses statistics (table size, index, selectivity of condition)

- steps in query optimization-

- Parsing: convert SQL into relational algebra
- Local optimization: Apply algebraic expression rules.
- Physical optimization: select best algorithms

- Plan generation : produce a query execution plan
- Execution : DBMS runs chosen plan.

- Benefits -
  - Reduces query execution time
  - Saves disk I/O and CPU cycles
  - Improves overall performance of DBMS.

Transformation of relational expressions -

- A relational algebra expression can be written in many equivalent forms.
- Transforming one expression into another equivalent form is called transformation of relational expression. (Same result but faster).
- Purpose :- To find a more efficient query execution plan.
- Why Transform :-
  - Reduce size of intermediate results.
  - Minimize cost.
  - Enable better use of indexes & joins
- Transformation rules :-
  - Cascade of selections

    A selection with multiple conditions can be broken into sequence of simple selections

- Commutativity of selection:-
  - order of multiple selections does not matter.
- Cascade of projections:-
  Multiple projections can be merged into one.
- Commutativity of selection & projection:-
  A projection & selection can be interchanged but projection must include all attributes used in selection condition.
- Commutativity of join:-
  order of relations in join does not matter.
- Associativity of join:-
  Grouping of joins does ut matter.
- Cartesian Product + selection = Join.
  A cartesian Product followed by selection is equivalent to a join.

Process of transforming SQL query into relational algebra operations:-

- SQL is high level, declarative language - user specifies what data is needed.
- DBMS converts SQL into relational algebra expressions.
- This transformation is 1st step in query processing.

- Steps in Transformation process:-

Step 1: Pasing & syntax analysis

- SQL query is checked for syntax errors.
- A parse tree is created, representing structure of SQL stmt.

Step 2: conversion to relational algebra

- Parse tree is translated into relational algebra expression.
- Main SQL clauses are:-
  - SELECT : Projection
  - FROM : relation
  - WHERE : Selection
  - JOIN : join
  - GROUP BY : Grouping
  - UNION / INTERSECT / EXCEPT : Set operation.

Step 3:- Expression tree formation

The expression is represented as an operator tree:-

Leaves → base relations

Internal nodes → relational operators.

Step 4:- Query optimization

Using transformation rules, Relation algebra expression is rewritten into an equivalent but more effient form.

- SQL query:

```
SELECT name
FROM Student
WHERE age > 18 AND dept = 'CS';
```

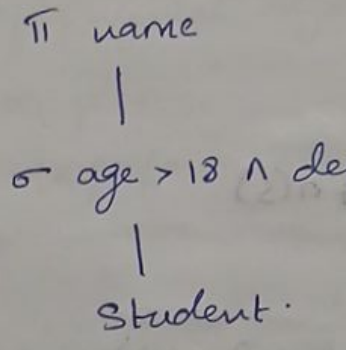
Relational algebra conversion :-

FROM student → Relation Student

WHERE age > 18 and dept = 'CS' → $\sigma_{age > 18 \wedge dept = 'CS'}(student)$

SELECT name → $\pi_{name}(\sigma_{age > 18 \wedge dept = 'CS'}(student))$

Expression tree :-



Estimating statistics of expression results -

- When optimizing queries, DBMS needs to estimate size of results produced by relational operations.
- These estimates help in choosing the best query plan. (since smaller intermediate results → faster query)

- Statistics Maintained by DBMS:-

$n(R)$ → No. of tuples in table R.

$V(R, A)$ → No of distinct values of attribute A in table R

$b(R)$ → No of data blocks used by table R

$T(R)$ → Average size of tuple in R.

- Estimation for different operations:-

a) selection (σ)

Query :- σ condition (R)

b) Projection (Π)

Query :- Π A(R)

c) Union ($R \cup S$)

Max size : $n(R) + n(S)$

d) Intersection ($R \cap S$)

Max size : $\min(n(R), n(S))$

e) Join ($R \bowtie S$):-

for condition : $R.A = S.B$

f) Cartesian Product ($R \times S$):-

Estimated size :- $n(R) + n(S)$

Choice of Evaluation plans :-

- A single query can be executed in multiple ways.
- Each way of executing is called an evaluation plan.
- DBMS must choose best plan → one with lowest estimation cost (I/O, CPU, memory).
- Types of evaluation plans :-

a) Logical evaluation plans :-

- Specifies what operations to perform
- Derived directly from SQL → relational algebra expressions
- Independent of physical algorithms.

b) Physical evaluation plan -

- Specifies how to perform operations.
- Example :-

Selection → Index scan or linear scan

Join → nested loop join, hash join, Sort merge join

- Factors in choosing a plan :-

- Size of relations
- Availability of indexes
- Join algorithms
- Cost estimates
- Statistics about relations.

Materialized views -

- A view in SQL is virtual table derived from query
- Normally, a view doesn't store data, it just runs query each time.
- A materialized view stores result of query physically in the database.
- use of materialized views :-
  - Improve query performance
  - useful for complex queries
  - support data warehouses & decision support systems.
- features :-
  - Stored physically on disk
  - Must be refreshed when underlying base tables change
  - can be indexed for fast access.
- Refreshing materialized views :-

  There are two ways -
  - Immediate refresh - updated automatically whenever base tables change.
  - Deferred refresh - updated at specific times (daily/weekly
- Advantages :-
  - Faster query performance.
  - Reduces computation cost for repeated queries.
  - useful in OLAP / data warehouse environments.

- Disadvantages :-

- Needs extra storage space.
- Requires maintenance
- May become outdated if not refreshed frequently.

Benefits of materialized views :-

a) Improved query performance -

Results are precomputed & stored, so queries run much faster.

b) Reduced computation cost -

Expensive operations are not recomputed each time

c) Efficient for repeated queries -

Commonly used queries can directly access the stored result instead of re-executing base query.

d) Support for data warehousing & OLAP -

widely used in decision support system where analytical queries over large datasets needs quick responses.

e) Can be indexed -

Since materialized views are physically stored, indexes can be created on them to further speed up access.

f) Reduces load on base tables -

Instead of repeatedly scanning large base tables, queries

can hit the smaller, precomputed view.

9) flexibility in refresh-

can be refreshed immediately / deferred balancing performance & freshness.

23/8/25

UNIT - III

Parallel Systems -

- Parallel systems are systems where multiple processors (or nodes) work simultaneously on a query / task by dividing the ~~workflow~~ workload, to achieve faster response time & higher throughput.

- Goals -
  - Improve performance
  - Increase throughput
  - Provide scalability
  - Ensure fault tolerance.

- Types of parallelism :-

a) Inter - query parallelism

Multiple queries run in parallel.

eg :- Query A runs on node 1,
Query B runs on node 2.

b) Intra - query parallelism

A single query is divided into parts & executed in parallel.

eg :- A SELECT query split into multiple sub - queries.

c) Intra-operation parallelism:-

A single operation is parallelized

eg:- Table scan divided across multiple processors.

d) Inter-operation parallelism:-

Different operations in query executed in parallel.

eg:- ~~while~~ while scanning one table, join with another can start.

=Architecture of parallel systems:-

a) Shared Memory -

All processors share same memory & disk.

b) Shared disk -

Each processor has its own memory, but disks are shared.

c) Shared Nothing -

Each processor has own memory + own disk

communication only via network.

- Benefits :-

a) faster query execution

b) can handle very large DB's.

c) Better resource utilization

d) Improved fault tolerance.

- Steps :-

a) Partitioning data.

b) Distributing tasks to nodes

c) Executing in parallel

d) collecting & merging results.

Speedup and Scaleup:-

Speedup:-

- The ratio of time taken to execute a task on a single processor to the time taken on parallel system with multiple processors.
- shows how much faster a system gets when more processors are added.
- formula:

$$\text{speedup (S)} = \frac{T_s}{T_p}$$

where T_s refers to time taken using single processor

T_p refers to time taken using parallel (large) system.

- Types of speedup:-

a) linear speedup:-

If adding N processors make task N time faster

b) sub linear speedup:-

speedup < N, due to overheads.

c) super linear speedup:-

speedup > N, happens rarely.

- Eg:-

If task takes 100 seconds on 1 processor and 25 seconds on 4 processors; $S = \frac{100}{25} = 4$.

Scaleup :-

- Refers to ability of parallel system to handle proportionally larger problems / workloads as system size increases. (Handling larger tasks with more resources).

- Goal :-

Keep performance constant when both workload & resources increase ~~together~~.

= -Types of scaleup-

a) Strong scaleup -

Same task executes faster as system resources increase.

b) Weak scaleup -

As workload increases, adding processors allow system to maintain same response time.

- Eg :-

Double data + Double processors → Same query time.

A good parallel system should achieve near-linear speedup & good scaleup.

- Running given task in less time by increasing degree of parallelism is called speedup.

- Handling larger tasks by increasing degree of parallelism is called scaleup.

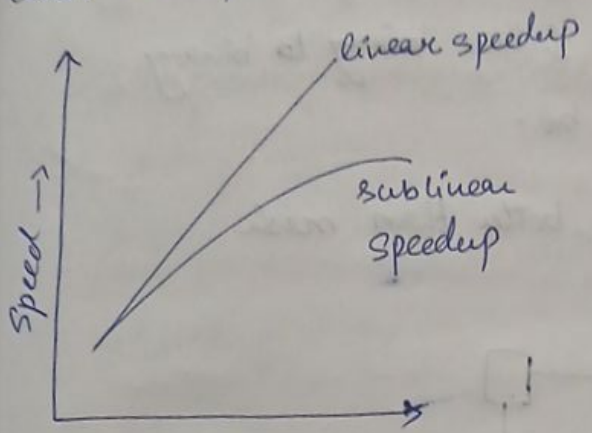


fig:- speedup

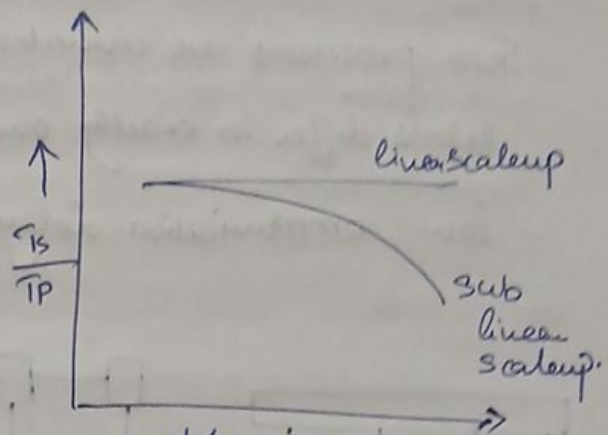


fig:- scaleup.

Interconnection Networks -

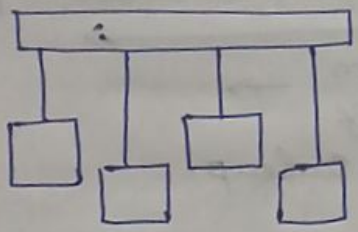
Parallel systems consists of set of components (processor, memory and disks) that can communicate with each other via interconnection network.

a) Bus -
- All components are connected to single communication bus
- Easy to implement, works well for small no. of processors.

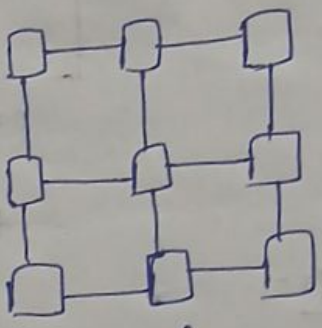
b) Mesh -
- Processors are arranged as nodes in a grid.
- Each node is connected to its adjacent neighbors (4 in 2D, 6 in 3D)

c) hyper cube -

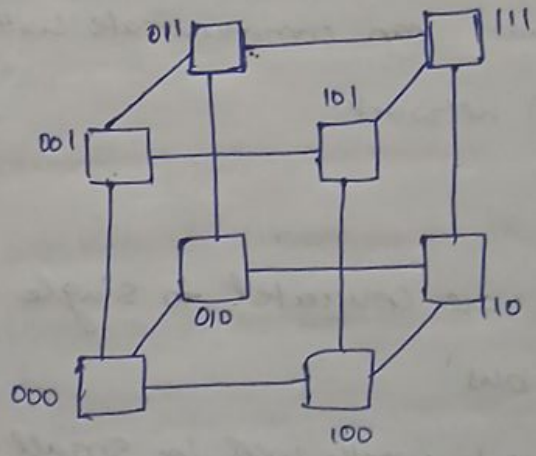
- Processors are labeled with binary numbers.
- Two processors are connected if their ~~binary~~ binary labels differ in exactly one bit.
- low communication delay, better than mesh.



Bus



Mesh.



hypercube

Parallel database architecture -

There are several architecture models for the parallel machines.

There are four main architectures as follows:-

a) Shared memory:-

- All processors share common memory & communicate through it.
- Each processor can directly read/write data in the shared memory.
- Adv: easy communication, faster access.

b) Shared disk:-

- All processors share same disk storage but each have its own private memory.
- Processors communicate indirectly via the disk.
- Adv: If one processor fails, others can still access the data.

c) Shared nothing:

- Each processor has its own private memory and own private disk.
- Processors communicate over network.
- Adv: scales very well since no resource is shared.

d) Hierarchical (hybrid):

- combination of above models.
- Provides flexibility & better fault tolerance.

These architectures improve speed, scalability & fault tolerance in parallel DB systems.

Parallel databases:-

- A type of database systems that uses multiple processors and disks working in parallel to process queries more efficiently and faster.

- Main goals:-
  - Speedup - Run queries faster by dividing work among processors.
  - Scaleup - Handle large databases/more users by adding more processors.
  - High throughput - Execute many queries at same time.

Parallel Database Architectures:-

- Shared Memory - All processors share one memory.
- Shared disk - processors have private memory but share same disk.
- Shared nothing - Each processor has its own memory and disk.
- Hierarchical - A hybrid/combination of above models.

I/O Parallelism -

- This reduces the time to read/write relations from disk by partitioning data across multiple disks & accessing them in parallel. [Performing disk operations in parallel using multiple disks, so that data can be fetched/stored faster.]
- This is usually done by ~~byyyy~~ horizontal partitioning
- Partitioning techniques :-

a) Round robin partitioning -

- Tuples are assigned to disks in circular manner
- Eg :- 1st tuple → Disk 0
  2nd tuple → Disk 1
  3rd tuple → Disk 2 then repeat.
- Ensures even distribution of tuples.
- Good for scanning entire relation sequentially.

b) Hash partitioning -

- A hash function is applied on one/more attributes.
- The result decides which disk gets the tuple.
- Eg :-
  If hash(emp_id) mod 3 = 0 → Disk 0
  = 1 → Disk 1
  = 2 → Disk 2
- Good for point queries.

c) Range Partitioning -

- Tuples are divided based on value ranges of attribute and stored on different disks.
- Example with Salary :-

  Salary < 10,000 → Disk 0

  Salary 10,000 - 50,000 → Disk 1

  Salary > 50,000 → Disk 2.

- Good for range queries.

Interquery parallelism :-

- Interquery parallelism means executing multiple queries at the same time in parallel, by assigning each query to a different processor.

- Example :-

Suppose a database recieves 3 queries at once :-

Query 1 : find all employees with salary > 50K

Query 2 : count total number of departments

Query 3 : list all customers in delhi

- In Interquery parallelism ; query1 runs on processor 1, query 2 runs on processor 2, query3 runs on processor 3 (in parallel)

Advantages :-

- Increase throughput
- Reduces waiting time for users when many arrive together.

Disadvantages :-

- No single query becomes faster, because each query is still executed by only one processor.
- only useful when there are many users/queries.

Intraquery Parallelism -

- Intraquery parallelism means executing a single query in parallel by dividing it into sub operations that can run simultaneously on different processors.
- Example :-

  Suppose database recieves one query :-

  find average salary of employees in each department.

  Processor 1 scans partition of employee table.

  Processor 2 scans another partition.

  Both processors compute partial results, then merge to get final average.

- Types of intraquery parallelism :-

  - Intra-operation parallelism - parallelize single operation.
  - Inter-operation Parallelism - Execute different operations of same query in parallel.

Advantages:-

- Speeds up execution of single large query.
- Efficient use of multiple processors.

Disadvantages:-

- High communication & synchronization overhead
- load balancing issues may arise.

Interoperation parallelism-

- Interoperation parallelism is type of interquery parallelism in which different operations of single query are executed simultaneously on multiple processors.
- Example:-

```
SELECT SUM(Salary)
FROM Employees
WHERE department = 'IT';
```

Operations involved:-

Selection : Department = 'IT'

Projection : Salary column

Aggregation : SUM

Processor 1 performs selection

processor 2 performs projection on already selected tuples

Process 3 performs aggregation. as soon as partial data is ready

This allows operations to overlap rather than executing strictly one after the other.

- Advantages:-
  - Faster query execution due to overlapping operations.
  - Efficient use of multiple processors in parallel DB systems.
  - can handle complex queries effectively

- Disadvantages:-
  - Some operations cannot start until their I/P is available.
  - Require careful coordination b/w operations.

Intraoperation Parallelism-

- A type of intraquery parallelism where single operation of query is divided & executed simultaneously on multiple processors.
- Goal: reduce the execution time of that operation:-
- Example:-

```
SELECT * FROM Employees
WHERE Salary > 50000;
```

Operation:

Selection (Salary > 50000)

Intra operation parallelism:-

If employees table has 1000 rows & 4 processors:-

Processor 1 checks rows 1-250

Processor 2 checks rows 251-500

Processor 3 checks rows 501-750

Processor 4 checks rows 751-1000.

All processors work simultaneously to complete selection faster.

Advantages:-

- Reduces execution time of complex operations.
- Efficient use of multiple processors.

Disadvantages:-

- overhead of splitting & merging data across processors.
- Some processors may finish earlier, leaving others idle.
- Implementation is more complex than sequential execution.

Design of parallel systems-

Design of parallel system refers to how system is architected to allow multiple processors to work together efficiently to solve computational / database tasks simultaneously.

Goals of design :-

a) Maximize speed up - Reduce total execution time

b) Scalability - Handle large datasets by adding more processors

c) Efficient resource utilization - Avoid idle processors -

d) Reliability - Ensure system works correctly even with processor failures.

Components of parallel system :-

a) Processors : perform computations

b) Memory can be :

Shared memory : All processors access same memory

Distributed memory : Each processor has its own memory.

c) Interconnection network :

connects processors & memory ; determines speed of data transfer.

Types :- Bus, ring, mesh, hypercube.

d) operating system :

Schedule tasks, manages communication, handles synchronization.

Design approaches:-

a) Shared memory systems -

- Processors share global memory.
- Easier programming but memory contention can occur.

b) Distributed memory systems -

- Each processor has its own memory.
- Requires message passing, scales well to large systems.

c) Hybrid systems -

- Combines shared & distributed memory features.

Performance Metrics :-

a) Speed up (S):

$$S = \frac{\text{Time of 1 processor}}{\text{Time of P processor.}}$$

b) Efficiency (E):

$$E = \frac{S}{P}$$

c) Scalability:

Ability to maintain efficiency as number of processors increases.

24/8/25

UNIT - IV

Distributed databases :-

- A distributed database is collection of multiple logically interrelated databases that are spread across different locations but connected through a computer network.
(a db that looks like one single db to user but in reality the data is stored at multiple places connected by network)

- Key characteristics -

- Distribution of data : data is stored in fragments across different sites.
- Transparency : users dont need to know where data is stored.
  - location transparency - users query without knowing site location.
  - Replication transparency - users dont worry about copy.
  - fragmentation transparency - Data fragments are hidden from users.
- Autonomy - each site can operate independently.
- Continuous availability - If one site fails other site work.
- Network based - Sites communicate via computer network.

- Architecture of DDS -

- Single site DBMS : centralized database
- Multi database systems : Different independent db, loosely connected.
- Distributed DB system : Appears as one DB, but actually distributed.

Reference architecture of DDB -

- A distributed database system (DDBS) is built on the top of a distributed database (DDB) which looks like single logical database to users but is actually spread across multiple sites.
- The reference architecture defines layers, components, and transparency features needed for a proper DDB.
- Layers of DDB reference architecture.

It is usually explained in 3 main layers :-

a) External layer (User layer) -

- Provides user views of DB.
- Hides distribution details - user feels they are accessing a single centralized DB
- Supports application programs, query interfaces, forms & reports.

b) Conceptual layer (Global schema)-

- Defines logical structure of entire distributed DB.
- Specifies entities, relationships, constraints & global schema mapping
- Achieves data distribution transparency.

c) Internal layer (fragmentation & allocation layer)-

- Deals with data fragmentation (horizontal, vertical, hybrid)
- Handles data replication & allocation to different sites.
- Includes local schema for each site.
- Responsible for query optimization, transaction mgmt, concurrency control.

- Main components :-

a) Global schema - Single logical schema of distributed DB.

b) Fragmentation & allocation schema - rules for how data is partitioned / replicated.

c) Local schemas - describes each sites' physical DB.

d) DDBMS s/w - provides integration, transparency & communication

- Types of transparency achieved :

A good DB reference architecture supports :-

a) Distribution transparency (users dont know where data is stored).

b) Replication transparency (users dont know if data is duplicated)

c) Fragmentation transparency (users dont know if data is split into fragments).

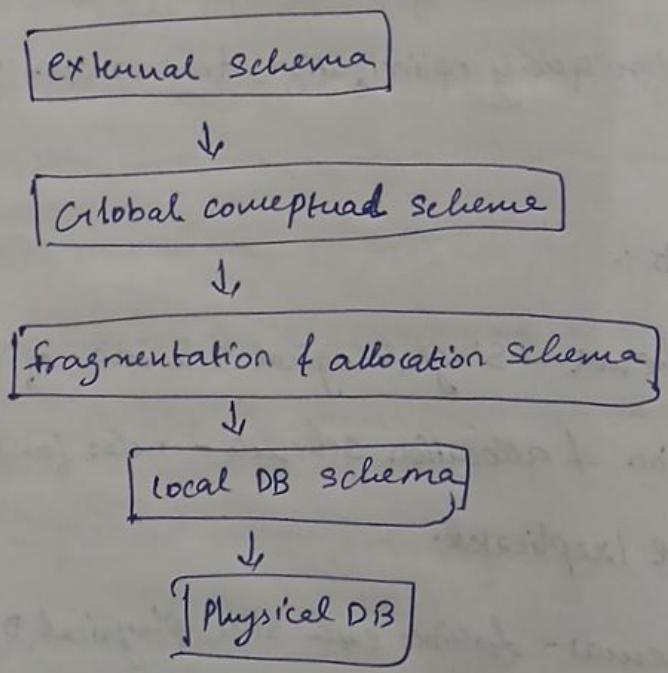
d) location transparency (users dont need to specify site)

e) Transaction transparency (ensures ACID across sites)

ACID - Atomicity, Consistency, Isolation, durability.

~~Types of fragmentation~~

- Reference Architecture of DDB



fragmentation -

- The process of dividing a relation (table) into smaller parts (fragments) so they can be stored across multiple sites in a distributed DB.

- Each fragment must be a logical subset of DB & together they should allow reconstruction of original relation without data loss

- Goal :

Improve locality of access. (data closer to where its needed)

Increase performance and parallelism

Maintain transparency. (users don't know about fragmentation).

- Types of fragmentation :-

a) Horizontal fragmentation -

Splits relation row wise (tuples / records)

Each fragment contains a subset of rows, but all columns.

Done using selection operator (σ) with conditions.

Eg :-

Employee table split by department.

fragment 1 :- Employees of dept = "Sales"

fragment 2 :- Employees of dept = "HR"

Adv - queries accessing specific groups of tuples run locally.

b) vertical fragmentation -

Splits relation column wise (attributes)

Each fragment contains subset of columns + the primary key.

Done using a projection operation (Π).

Eg :-

Employee table.

fragment 1 :- (EmpId, name, Dept)

fragment 2 :- (EmpId, salary, address)

Adv - queries needing only some attributes can avoid scanning

unnecessary columns.

c) Hybrid (Mixed) fragmentation:-

- Combination of horizontal + vertical.
- first apply horizontal fragmentation, then vertical (or vice versa)
- Provides fine grained control of distribution.

Eg:-

Split employee by dept (horizontal) & then split each dept's fragment into personal info vs job info (vertical).

d) Derived fragmentation:-

- used when relation is fragmented based on fragmentation of another relation (because of join dependency).
- Ensures related tuples are co-located.
- Eg:- If dept is horizontally fragmented by region, then employee must also be fragmented acc. to dept no to maintain join locality.

- Properties of good fragmentation:-

- Completeness - All data must be included across fragments.
- Reconstruction - original relation can be reconstructed using union (horizontal) or join (vertical)
- Disjointness - No redundant data (except in replication)

Distributed transparency for read only applications-

- Distributed transparency is the ability of distributed database system to hide details of data distribution (location, fragmentation, replication) from the user. for the user, the system looks like a single centralized db, even though data is spread across sites.

- for read-only applications -

In read-only applications (where users only query the DB but donot perform updates), distributed transparency play a very important role.

a) location transparency -

users donot need to know where the Data resides.

query :-

```
SELECT * FROM employee WHERE dept = "HR";
```

works the same whether employee table rows are stored in Delhi, Mumbai or Chennai sites.

The DDBMS automatically locates data & fetches it.

b) fragmentation transparency -

If data is horizontally or vertically fragmented, the system automatically reconstructs the original relation (table).

Eg:-

employee table split by dept across sites.

user query retrieves all employees - DBMS performs a union of all fragments.

c) Replication transparency -

If copies of same data exist at multiple sites, the system chooses the nearest / most efficient copy for query execution.

In read-only applications, consistency mgmt is easier since no updates are made.

d) Transaction transparency - (Read only case) :-

Ensures that even read only queries follow ACID properties (mainly consistency & Isolation).

Since no updates are performed, concurrency control overhead is much less.

Multiple users can read from different replicas simultaneously without conflict.

- Advantages for read only application :-

Performance - Queries can be answered from closest replica

Parallelism - Queries can run in parallel across different sites.

fault tolerance - If one site is unavailable, another replica can serve the query.

user simplicity - user is unaware of fragmentation, replication or location → just queries the database like a single system.

Distribution transparency for update applications -

- Distribution transparency means the user (application) interacts with the distributed database as if it were a single centralized database, without worrying about where data is stored, how it is fragmented or how many copies exist.
- for update applications, this transparency is essentially important because updates must be consistent, reliable & independent of distribution.
- Types of distribution transparency for update applications :-

a) fragmentation transparency :-

- User updates DB without knowing if table is fragmented.
- The system ensures that update is applied to correct fragments.
- Eg :- If an employee record is horizontally fragmented by dept, updating salary requires system to find the right fragment automatically.

b) Replication transparency :-

- user updates data without knowing if multiple copies (replicas) exist.
- The system ensures that all replicas are updated consistently
- Eg :- If employee data is replicated at 3 sites, an update query must modify all 3 copies / ensure conflict resolutions

c) Location transparency:-

- Users does not need to specify physical location/site of data.
- update requests are sent to logical database name, and the system routes them to correct site.
- Eg:- UPDATE Employee Set Salary = 5000 where empid = 101;
  will automatically go to site where empid = 101 exists.

d) Transaction transparency:-

- Ensures that updates follow ACID properties across distributed sites.
- Even if update involves multiple fragments/replicas the system guarantees atomicity & consistency.
- Eg:- Salary update in one site + corresponding dept update in another must both succeed or both roll back.

Distributed Database query primitives -

- In Distributed database system, users query/update the database as if it were centralized. But internally, the system uses query primitives to perform operations across different sites.
- Query primitives are basic operations that a distributed DBMS supports to retrieve or update data stored at multiple sites.

- Types of aues primitives

a) Remote aues primitive (RAP)- (local read/write)

- used to aues data stored at remote site
- Allows read/write operations without user knowing the location.
- Eg:- Read (Emp_Record) from Site B while user queries from site A.

b) local aues Primitive (LAP)- (Remote read)

- used to aues data stored at same(local) site.
- Eg:- Searching employee records in local database at Sik A.

c) Remote update Primitive (RUP)- (Remote update)

- used to update data at ~~the~~ remote site.
- Ensures replication/consistency is maintained.
- Eg:- update Salary = 50000 at site B triggered from Site A.

d) Transaction control primitives (TCP) - Commit/rollback in distributed setting)

- Manage distribute transactions aues multiple Sites.
- Ensure ACID properties.
- Eg:- Begin transaction, commit, Rollback using Two phase Commit (2PC) protocol.

e) communication primitive - (Messaging blw sites)

- Handle message passing blw sites for query execution and transaction coordination.
- Eg :- Sending query request, recieving result set, coordination messages in 2PC.

Integrity constraints in DDB - [Rules that make sure data is correct/consistent in distributed DB]

- Integrity constraints are rules that ensure the accuracy, consistency, and validity of data in a DB.

In DDB, enforcing constraints is harder because data is fragmented, replicated and spread across sites.

- Types of integrity constraints :-

a) Domain constraints - [values must be valid]

values of an attribute must be from a valid domain (eg : age must be integer > 0)

Enforced locally at each site.

b) Entity integrity - [Primary key cannot be null]

Every relation must have a primary key that is not null.

Even if table is fragmented across sites, entity integrity must hold.

c) Referential Integrity - [foreign key must match a primary key in another table.]

- A foreign key in one relation must reference a valid primary key in another.

problem in DDB - the two relations may be at different sites

Requires coordination b/w sites to check constraints.

d) unique constraints - [NO duplicates allowed]

Attribute values must be unique across the entire distributed system.

eg:- Employee ID must not duplicate across sites.

Requires a global check before inserting new records.

e) check constraints / General constraints:- [Data must satisfy a condition]

Any condition defined by user (eg :- Salary > 3000)

Must be verified at the site where update occurs, and sometime globally

f) key constraints in replication -

If data is replicated across sites, all copies must respect the same constraints.

updates must be propogated consistently to avoid violation.

Challenges in DDB -

a) Data fragmentation - constraints may involve fragments stored at different sites.

b) Data replication - updates must be consistent across all replicas

c) communication cost: checking referential / uniqueness constraints across sites is expensive.

d) concurrency: simultaneous updates at different sites may cause violations.

A framework for distributed database design-

- Designing a distributed database is more complex than centralized DB because data is spread across multiple sites. A framework gives a step by step process to design such a system.

-Steps in framework-

a) Requirement analysis:-

- Identify what users / applications need.
- Decide what data, which queries / transactions and performance goals.

b) Conceptual DB design:-

- Create global schema that represents entire DB without distribution details.

c) Fragmentation design:

- Decide how to split tables

  Horizontal fragmentation

  Vertical fragmentation

  Hybrid fragmentation

d) Allocation design -

- Decide where to store each fragment.
- options:
  - Store each fragment at one site.
  - Replicate fragments at multiple sites.
  - Consider site workload, communication cost & reliability.

e) Replication design -

- Decide which fragments need copies at multiple sites.
- Balance availability & performance vs update overhead.

f) local schema design -

- for each site, map fragments into a local DB schema

g) validation -

- check correctness of design:
  - completeness
  - Reconstruction
  - Disjointness

h) implementation -

- Define communication, concurrency control & transaction mgmt mechanisms
- Ensure distributed ~~trans~~ transparency for users.

The design of database fragmentation-

- fragmentation refers to breaking a relation into smaller parts for storage at different sites. The design of fragmentation is about deciding how to split relations so that queries run efficiently & data consistency is maintained.

- steps in designing fragmentation.

a) Choose fragmentation type :-

AKSHITA

Horizontal fragmentation - split by rows

Vertical fragmentation - split by columns.

Hybrid fragmentation - Mix of both.

b) Define fragmentation rules :-

use selection predicates for horizontal fragmentation

use attribute grouping for vertical fragmentation

c) check fragmentation correctness :-

Every good fragmentation must satisfy :-

- Completeness - All data in original relation must appear in fragments
- Reconstruction - original relation can be rebuilt
- Disjointness - Tuples / attributes should not overlap.

d) Decide fragment allocation-

- After designing fragments, assign them to sites
- criteria : acess frequency, locality of queries, communication cost

Evaluate fragmentation strategies impact on db performance -

- fragmentation refers to dividing a relation into smaller pieces. Different strategies affect query performance, storage & communication cost in distributed db.

- horizontal fragmentation:

Split rows based on condition.

Adv:

Queries that access local data only are faster.

Multiple sites can process different fragments at same time

Storage is balanced among sites.

Disadv:

Queries that need to scan all rows across sites become slower.

Join across fragments may increase network cost.

- vertical fragmentation:

Split columns into fragments with a common key.

Adv:

Reduce size of tuples

Improve performance for queries that access only subset of attributes.

Sensitive attributes can be kept at restricted sites.

Disadv:-

Queries needing all attributes require join operations

If queries frequently need whole tuple, performance suffers

- Hybrid fragmentation:

combination of both horizontal + vertical fragmentation

Adv:-

can give best of both worlds if designed properly.

Queries that are both location specific & attribute specific perform better

Disadv:

Complex to design & maintain.

Reconstruction may require both Union + Join -> higher cost

- General impacts of performance.

locality of access - performance improves if data is fragmented based on query patterns.

communication overhead - Performance drops if queries need data from multiple fragments/sites.

Parallelism - Performance improves because fragments can be processed simultaneously

update cost - If fragments are replicated, updates take longer.

Difference blw centralized & Distributed database systems-

Feature	centralized DB	Distributed DB.
location	Entire DB stored at one single site	DB is spread across multiple sites connected by network.
Control	Managed by single DBMS at one location	controlled by a distributed DBMS that coordinates multiple sites.
Availability	If the central site fails → whole system fails.	failure of one site doesn't stop the entire DB system.
Performance	Good for small/medium systems, but bottleneck if many users.	Better performance due to parallelism & locality of data.
Scalability	Difficult to scale	Highly scalable
Communication Cost	low, since everything is in one place	Higher, since queries may need data from multiple sites
Data Sharing	limited, all users must connect to the central site	users can access local site data quickly & share globally if needed.
Security	Easier to enforce security	More complex, since multiple sites need consistent security policies.

Redundancy/ replication	usually no redundancy	Supports replication, which improves reliability but adds update cost.

Centralized DB -> Simple, one site, but single point of failure

Distributed DB -> Data spread across sites, better availability & scalability, but more complex.

1/9/25

UNIT - V

Transformation of global queries to fragment queries -

- In a distributed DB, data is fragmented (horizontally, vatically or hybrid) & stored at diffuent sites. when a global query is written, it refers to global schema (as if DB is centralized). But, the system must transform it into fragment queries that run on correct sites.

- steps in transformation :-

a) Parsing global query -

Gobal query is first written in SQL (on global schema)

eg :-

```
SELECT name FROM Employees WHERE dept = 'HR';
```

b) Mapping to fragments -

The system checks the fragmentation rules.

Suppose employee table is horizontally fragmented.

Emp_hyderabad (Dept in hyderabad)

Emp_Chennai (Dept in Chennai)

Emp_Delhi (Dept in Delhi)

c) Generate fragment queries -

The global query is rewritten as queries on each fragment.

Such as follows :-

~~SELECT name FROM employees WHERE dept~~

SELECT name from emp-hyderabad WHERE dept = 'HR';

SELECT name FROM emp-chennai WHERE dept = 'HR';

SELECT name FROM emp-delhi WHERE dept = 'HR';

d) Execute locally -

Each fragment query runs ~~as~~ at its respective site.

e) combine results -

The local results are sent back & combined (via union or join) to produce the final global results.

Equivalence transformations for queries -

- Equivalence transformations are rules that convert one query into another without changing its final result.
- They are mainly used in query optimization in distributed databases.
- Since data is fragmented & distributed, we try to reduce communication cost, computation time & data transfer by rewriting queries into more efficient forms.
- Types of equivalence transformations :-

a) commutativity - order of operations can be change without affecting result.

eg:- $R \bowtie S = S \bowtie R$

b) Associativity - Grouping of operations can be changed.

eg:- $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

c) Selection pushdown - Move selections closer to base relations to reduce data early.

eg:- $\sigma\theta(R \bowtie S) = (\sigma\theta(R) \bowtie S)$

d) Projection pushdown - Apply projection (Π) to early to remove unnecessary attributes.

eg:- $\Pi A (R \bowtie S) = (\Pi A1(R)) \bowtie (\Pi A2(S))$

e) Join with selection - Combine selection with join if condition relates to both relations.

eg:- $\sigma R.A = S.B(R \times S) = R \bowtie$

f) Cascade of selections - Break down combined selections into smaller one

eg:-

$\sigma\theta1 \wedge \theta2(R) = \sigma\theta1(\sigma\theta2(R))$.

Importance -

a) Helps optimize queries in DDS.

b) Reduces comp communication cost, data transfer & execution time

c) Ensures queries are executed efficiently while returning same result.

Distributed grouping & aggregate function evaluation:-

- In distributed database systems, queries often use the aggregate functions like:-

  sum, count, avg, min, max, group by.

Since data is stored across multiple sites, aggregate evaluation must be done efficiently to reduce the communication cost.

- Steps for distributed aggregate evaluation:-

a) local computation at each site:

Each site compute partial aggregates on its fragment.

eg: Site 1 → Sum = 200

Site 2 → Sum = 300

b) Transfer partial results :-

Only partial results are sent to a central site.

c) Global aggregation:-

The central site combines partial results to get final answer.

eg: -Global Sum = 200+300 = 500.

- Grouping in Distributed system:-

If query has group by, grouping can also be done locally first.

eg:-

Query - SELECT dept, SUM(salary) FROM employee GROUP BY dept;

- Each site groups employees by department locally.
- Then sends only grouped results to coordinator.
- Coordinator merges groups & computes final aggregates.

- Aggregate functions & how they work in DDS:-

SUM/COUNT/AVG :- Compute partial sums/counts locally, then merge.

AVG = (Global SUM) ÷ (Global count).

MIN/MAX = Take local min/max, then global min/max.

- Advantages:-

- Reduces data transfer.
- Improves query performance
- Makes system more scalable.

Parametric queries -

- A parametric query is a query that contains parameters instead of fixed values
- The parameter values are supplied at execution time, not when query is written.
- common in database & especially in distributed database systems to improve performance & security.

Example:-

Instead of writing:-

```
SELECT * FROM employees WHERE dept = 'Sales';
```

we write:-

```
SELECT * FROM employee WHERE dept = ?;
```

Here, ? is a parameter → actual value (eg:- "Sales", "HR") is given at runtime.

Uses in DDS

a) Reusability - Same query can be executed with different values

b) optimization - Query is parsed & optimized once, only parameters change

c) security - protects against SQL injection.

d) Efficiency - saves communication & compilation cost in the distributed systems.

Access control Models -

Access control databases define who can access what data & how. There are 3 main models:-

a) Discretionary access control (DAC)

- Access is based on owners discretion.
- User who owns a resource can grant or revoke access to others.

b) Mandatory access control (MAC) -

- Access is based on the security levels.
- users and data objects are assigned labels
- user can access data only if clearance level ≥ data level.

c) Role - based access control (RBAC) -

- Access is given based on roles instead of individuals.
- users are assigned roles & roles have permissions.
- eg :- Admin, manager, employee.
- widely used in modern enterprises.

DAC - Owner control access

MAC - Access based on security levels.

RBAC - Access based on roles.

Database Security -

In Distributed database system (DDS), data is stored across multiple sites connected by a network. This increases security challenges because data travels over networks & is stored at different locations.

Key aspects of security in DDS :-

a) Access control -

Ensures only authorized users can access data.

Uses DAC, MAC, RBAC models.

b) Authorization & authentication -

verifying user identity.

Deciding what actions they are allowed to perform.

c) Encryption -

protects data during storage & transmission over networks

prevents eavesdropping & tampering.

d) Data Integrity -

Ensures data is accurate, consistent & not modified by unauthorized users.

uses checksums, hashing & transaction control.

e) Audit & Monitoring -

keeps logs of all user activities.

Helps in detecting intrusions / misuse

f) Replication Security -

In DDS, data is replicated at multiple sites

Security policies must ensure replicas are equally protected

g) failure & Recovery Security -

Backup & recovery mechanisms ensure data is not lost or corrupted in case of system failures or attacks

- Challenges in DDS security -

- Multiple sites : harder to maintain uniform security policies.
- Network based communication : vulnerable to interception.
- More attack surface due to distribution.

Join Queries -

- A query that combines rows from two/more tables based on a related column btw them. It is used to merge data that is spread across different relations.
- Types of join :-

a) INNER JOIN :

Returns only rows when there is a match in both tables.

b) LEFT JOIN :

Returns all rows from left table, and matching rows from right table.

If no match → NULL is shown.

c) RIGHT JOIN :

Opposite to left join.

Returns all rows from right table and matching rows from left table.

d) FULL OUTER JOIN :

Returns all rows from both tables.

Non matching rows filled with NULL.

e) CROSS JOIN-

Returns all combinations of rows from both tables.

eg:- If tableA has 3 rows and table B has 4 rows → result = 12 rows

f) SELF JOIN-

A table is joined with itself.

eg:- finding employees and their managers in same employee table.

Join in Distributed databases.

Join may happen blw tables stored at different sites.

optimization is required to reduce data transfer

General Queries-

A query is a request to retrieve / manipulate data stored in the database.

Types of general queries:-

a) Data retrieval queries (SELECT)-

used to fetch data from one / more tables

eg:-

SELECT name, salary FROM employee WHERE dept = 'Sales';

b) Data Modification queries-

- INSERT - Add new records.

INSERT INTO Employee (id, name, Salary) VALUES (1, 'John', 50000);

- UPDATE - Modify existing records.

UPDATE Employee SET Salary = 6000 WHERE id=1;

- DELETE - Remove records.

DELETE FROM Employee WHERE id=1;

c) DDL queries (Structure definition)

- CREATE - Create new table of database

CREATE TABLE department (dept-id INT, dept-name VARCHAR(50));

- ALTER - Modify table structure

- DROP - Delete table/database

d) Aggregate queries -

use fimctions like SUM, COUNT, AVG, MIN, MAX with GROUPBY.

SELECT dept, AVG(salary) FROM Employee GROUPBY dept;

e) Join queries -

Combine data from multiple tables (Inner JOIN, OUTER JOIN, etc)

UNIT-I

1) comparision blw file based system and applications.

filed-Based System	Applications in DDS
- Data Stored in seperate files, usually at one Location	- Data stored in databases distrib across multiple sites
- Acces through file-handling routines, no query language	- Acces through SQL & distributed query processing
- High data redundancy across files.	- Controlled redundancy with replication Strategies.
- consistency is hard to maintain	- Strong mechanisms ensure global consistency.
- limited security, mainly file permissions.	- Advanced security with authentication & authorization
- Poor data sharing, manual file transfer needed	- Easy sharing, multiple users & applications can acess concurrently
- Not Scalable, adding new files is difficult	- Highly scalable, new sites and databases can be added.
- Little or no concurrency control	- Full concurrency control with locks, timestamps etc

- Suitable for simple, standalone applications	- Suitable for complex, mission-critical applications like banking, telecom, e-commerce
- Low fault tolerance, failures may cause data loss	- High fault tolerance with replication & recovery techniques.

2) object, • semi structured databases -

object oriented databases -

- Stores data in form of objects (like object oriented programming)
- Each object has attributes & methods (data & functions)
- Supports complex data types like images, audio, video.
- provides inheritance, encapsulation and polymorphism.
- Example: Storing a "Student" object with properties (name, roll number) and methods (calculate GPA)

Semi-Structured databases -

- Data does not follow a strict schema like relational databases
- Structure can be irregular / flexible.
- Represented using XML, JSON
- Suitable when data format is not fixed / varies frequently
- Example: web data, sensor data, emails.

3) TCL in database languages -

- TCL stands for transaction control language, which is a subset of SQL used to manage transactions in a database
- A transaction is logical unit of work that must be executed fully or not at all (follow acid properties)

- Commands of TCL -

a) commit -

- Save all the changes made by transaction permanently into the database
- eg :-

```
COMMIT;
```

b) ROLLBACK -

- undoes the changes made by the current transaction (before commit)
- eg :-

```
ROLLBACK;
```

c) SAVEPOINT -

- creates checkpoint within a transaction.
- you can roll back to that point without affecting the entire transaction
- eg :-

```
SAVEPOINT SP1;
ROLLBACK TO SP1;
```

UNIT-III

1) Difference b/w interquery & intraquery.

Interquery [Many queries run at once]	Intraquery [One query is split & run in Parallel]
→ Executes multiple queries in parallel	→ Executes a single query in parallel.
→ Each query is independent of others	→ A single query is divided into sub tasks that run simultaneously
→ Improves throughput	→ Improves response time.
→ eg:- Running salary update query & product search query at same time	→ eg:- Splitting a large join query into smaller multiple join operations across sites
→ useful in multi-user environment where many queries arrive together.	→ useful for complex queries involving joins, aggregations/ large datasets.
→ Coarse grained parallelism	→ Fine grained parallelism.

2) Difference between intraoperation and interoperation

Intra operation	Inter operation.
→ Parallelism within a single operation	→ Parallelism across different operations in a query.
→ Breaks one operation into smaller tasks & run them in parallel	→ Executes multiple operations of same query at sametime.
→ Improves performance of costly operations like joins and aggregations	→ Exploits pipelining / independent execution of different operators
→ Eg: partitioning table scan across multiple processors	→ Eg: performing a join operation while also doing aggregation on another part of query
→ fine grained parallelism	→ Coarse grained parallelism.

Intra operation : parallelism inside one operation

Inter operation : parallelism between multiple operations.