ADBMS

SUPER 25

1. Explain Two-tier client server database model with diagram.

Two-Tier Client-Server Model:

- Developed in 1980
- Supports form-based, user-friendly interface
- Client side – user interface, business logic→ called as fat client
- Server side - Database system services

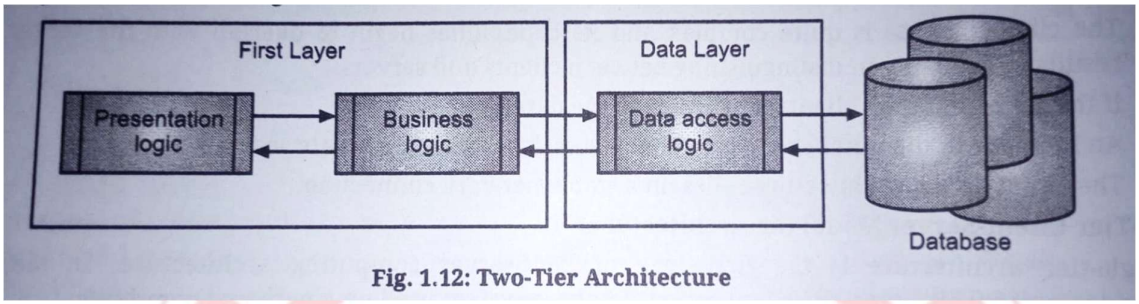


Fig. 1.12: Two-Tier Architecture

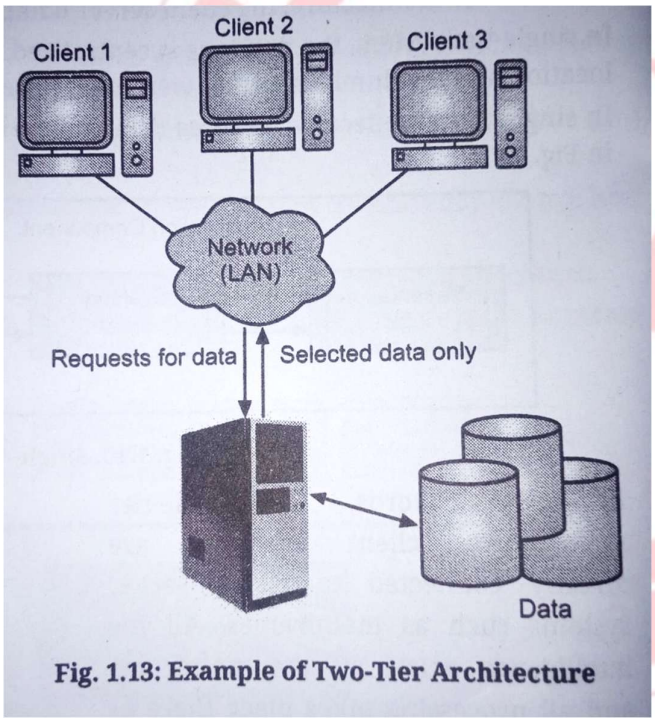


Fig. 1.13: Example of Two-Tier Architecture

- Client communicates through SQL statement

Advantages:

- Improved usability
- User friendly interface
- Less complicated
- Server can act as client for another server

Disadvantages:

- Difficult to administer
- Performance deteriorates when number of users increase

2. Explain XML document schema.

Databases have schemas, which are used to constrain what information can be stored in the database and to constrain the data types of the stored information. e the first schema-definition language included as part of the XML standard, the Document Type Definition, as well as its more recently defined replacement, XML Schema. Another XML schema definition
language called Relax NG is also in use.

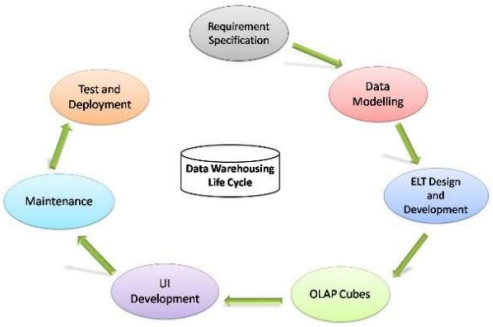
XML Schema defines a number of built-in types such as string, integer, decimal date, and boolean. In addition, it allows user-defined types;
these may be simple types with added restrictions, or complex types constructed using constructors such as complex Type and sequence

The first thing to note is that schema definitions in XML Schema are themselves specified in XML syntax, using a variety of tags defined by XML Schema. To avoid conflicts with user-defined tags, we prefix the XML Schema tag with the namespace prefix "xs:"; this prefix is
associated with the XML Schema namespace by the xmlns:xs specification in the root element:
<xs:schemaxmlns:xs="http://www.w3.org/2001/XMLSchema">
Note that any namespace prefix could be used in place of xs; thus we could replace all occurrences of "xs:" in the schema definition with "xsd:" without changing the meaning of the schema definition. All types defined by XML Schema must be prefixed by this namespace prefix.

Example:

```
<xs:schemaxmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="university" type="universityType" />
<xs:element name="department">
        <xs:complexType>
        <xs:sequence>
        <xs:element name="dept name" type="xs:string"/>
        <xs:element name="building" type="xs:string"/>
        <xs:element name="budget" type="xs:decimal"/>
        </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="course">
        <xs:element name="course id" type="xs:string"/>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="dept name" type="xs:string"/>
        <xs:element name="credits" type="xs:decimal"/>
</xs:element>
<xs:element name="instructor">
        <xs:complexType>
                <xs:sequence>
                <xs:element name="IID" type="xs:string"/>
                <xs:element name="name" type="xs:string"/>
                <xs:element name="dept name" type="xs:string"/>
                <xs:element name="salary" type="xs:decimal"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

3. Draw and explain data warehousing life cycle.



1. Requirement Gathering

- Firstly, Business analysts, the local technical lead, and the customer complete it.
- A business analyst creates a business requirement specification (BRS) document during this stage.
- Finally, after gathering the requirements, the data modeler begins to identify the dimensions, facts, and combinations depending on the needs.
- We can describe this as the data warehouse's general design.

2. Data Modelling

- This is the second step.
- This is the process of designing databases and visualizing data distribution.
- Star Schema, Snowflake Schema and Galaxy Schema are the three data models of data warehouse.
- It is the reasoning behind the storage of data in relation to other data.

3. ELT Design and development

- Thirdly a data lake can include the data that an ETL (Extract, transfer, load) tool has extracted from different source systems.
- ETL process **extract** the data from the lake. After that it **transforms** and **load** it into a data warehouse for reporting.
- A solid ETL procedure can be useful in creating a straightforward yet useful data warehouse that is beneficial at all organizational levels.
- ELT application design to fulfil specifications documents created during the analytical phase

4. OLAP Cubes

- This is the Fourth step.
- Also called **hypercube** or **multidimensional cube**.
- It is a data format that enables quick data analysis in accordance with the several aspects that characterize a business problem.
- From various data sources and file types, such as text files, excel sheets, multimedia files, etc., a data warehouse would extract information.
- The retrieved data is **modified** and **cleaned** before being placed into an OLAP server (or OLAP cube) for preliminary processing in preparation for additional analysis.

5. UI Development

- This is the Fifth step.
- A **user interface is required** for the communication between user and a computer system.
- A user interface's primary goal is to give users the ability to efficiently control the equipment or device they're using.
- There are several **tools** available on the market to aid in UI development. For those using BigQuery, excellent options include BI tools like Tableau or PowerBI.

6. Maintenance

- This is the Sixth step.
- In this step we can **update** or **change** the schema and the application domain or requirement of it.
- Systems for maintaining data warehouses must include tools for tracking changes to the schema, such as updates.

7. Test and Deployment

- This is the ultimate step.

- Finally, Business and organization evaluate data warehouses to determine implementation of necessary business problems.
- The warehouse testing requires **examining vast amounts of data.**
- Similarly, different types of data sources, including relational databases, flat files, operational data, etc., provide the data that must be compared.
- At the time of implementation, the majority of its capabilities are implemented. Additionally, both on-premises and cloud deployment options are available for the data warehouses.

4. Explain flower expressions and nested queries in Xquery.

The programming language XQuery defines **FLWOR** (pronounced 'flower') as an expression that supports iteration and binding of variables to intermediate results. **FLWOR** is an acronym: FOR, LET, WHERE, ORDER BY, RETURN.

- **For** - selects a sequence of nodes
- **Let** - binds a sequence to a variable
- **Where** - filters the nodes
- **Order by** - sorts the nodes
- **Return** - what to return (gets evaluated once for every node)

E:g:

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

The **for** clause selects all book elements under the bookstore element into a variable called $x.

The **where** clause selects only book elements with a price element with a value greater than 30.

The **order by** clause defines the sort-order. Will be sort by the title element.

The **return** clause specifies what should be returned. Here it returns the title elements.

Nested xquery can be considered the multiple nested operation from the xquery. In nested queries, a query is written inside a query. The result of inner query is used in execution of outer query For each book in the bibliography, list the title and authors, grouped
inside a result element.

E:g:

```
<results>
{
    for $b in doc("bib.xml")/bib/book
     return
<result>
            { $b/title }
            {
              for $a in $b/author
               return $a
            }
</result>
```

}
</results>

5. Explain Concurrency Control Techniques.

There are different concurrency control techniques such as:

- Lock based protocols
- Two phase Locking protocols
- Time stamp-based protocols
- Lock based protocol: To ensure serializability it requires that the data items be accessed in a mutually exclusive manner.

i.e. While one transaction is accessing a data item, no other transaction can
modify that data.
Method used to implement this requirement is to allow transaction to access a data item only if it is currently holding a lock on that item.

Locks: Lock is a data variable which is associated with a data item.
Locks help synchronize access to the database items by concurrent transactions.
All lock requests are made to the concurrency-control manager.
Transactions proceed only once the lock request is granted.
There are different types of locks:

Binary loc: A binary lock on a data item can either have locked or
unlocked states.

Shared Lock: A shared lock is also called as Read only lock.
With the shared locks data items can be shared between transactions.
Because with shared locks you will never have permission to update data on the data item.
Shared lock is denoted by S.

Exclusive Lock: With the exclusive lock a data item can be read as well as written.
This lock can't be held concurrently on the same data item.
It is denoted by X.
Exclusive lock is requested using lock-X instruction.

Two phase Locking protocol: which is also known as 2PL.
Two phase locking protocol requires that each transaction issues lock and unlock requests in two phases:

Growing phase: A transaction may obtain locks but may not release any lock.

Shrinking phase: A transaction may release locks, but may not obtain any new locks.
If the conversion is allowed, then upgrading of locks from S(A) to X(A)
happens in growing phase and the downgrade of locks from X(A) to S(A) happens in shrinking phase.
It is true that 2PL protocol offers serializability. However it does not ensure that dead locks not happen.

Time stamp-based protocols: The timestamp-based algorithm uses a timestamp to serialize the execution of concurrent transactions.
This protocol ensures that every read and write operations are executed in timestamp order.
These protocol uses the System Time or logical count as a timestamp.
The older transaction is always given priority in this method.
This is the most commonly used concurrency protocol.
E.g:
Suppose there are transactions T1, T2 and T3
T1 has entered the system at time 0010
T2 has entered the system at 0020
T3 has entered the system at 0030
Thus the priority will be given to transaction T1, then transaction T2 and then lastly to Transaction T3.

6. Explain Arrays and Multiset in SQL.

SQL supports 2 collection types→ Arrays and Multiset

Array

- It is an **ordered collection** of elements
- Array type violates 1NF(First Normal Form in DBMS)
- Array types enable us to **enhance** a field of an existing type by **putting more than one entry in to it**.
- This creates a repeating group
- Arrays are order such that each element in the array corresponds to exactly 1 ordinal position in the array
- CREATE TYPE colors_array AS VARRAY(5) OF VARCHAR(20);
- CREATE TABLE products (
- id INT,
- name VARCHAR(50),
- colors colors_array
-);
- INSERT INTO products (id, name, colors)
- VALUES (1, 'Shirt', colors_array('Red', 'Blue', 'Green'));
- SELECT colors[2] FROM products WHERE id = 1;

-- Output: Blue

Multiset

- It is an **unordered collection of elements**
- An element may occur multiple times
- We cannot reference one element in multiset because their **location** in the collection is **unknown**.
- Ex. Array and Multiset valued attributes can be defined as

```
CREATE TYPE Publisher AS (
name VARCHAR(20),
branch VARCHAR(20));

CREATE TYPE Book AS
(
title VARCHAR(20),
author_array VARCHAR(20) array [10],
pub_date date,
publisher Publisher,
keyword_set VARCHAR(20) multiset);

CREATE TABLE books of Book;
```

- CREATE TYPE fruits_multiset AS TABLE OF VARCHAR(20);
-
- CREATE TABLE fruit_basket (
- id INT,
- fruits fruits_multiset
-);
-
- INSERT INTO fruit_basket (id, fruits)
- VALUES (1, fruits_multiset('Apple', 'Banana', 'Apple'));
- This allows storing multiple instances of the same value (e.g., two Apples).

Feature	Array	Multiset

Size	Fixed	Dynamic
Duplicates	Not allowed	Allowed
Order	Maintains order	Unordered
Use Case	Structured, indexed data	Flexible, repeated values

Creating and Accessing collection values

Array:

- array[‘abc’,’pqr’,’xyz’]

Multiset:

- multiset[‘computer’,’database’,’sql’]
- we can access or update elements of an array by specifying the array index
- array[2]

7. Explain any two basic operation with Mongo DB shell OR Explain any four operation with Mongo DB with example.

The basic operations of Mongo DB are **CRUD** operations.

CRUD operations - **C**reate, **R**ead, **U**pdate & **D**elete documents.

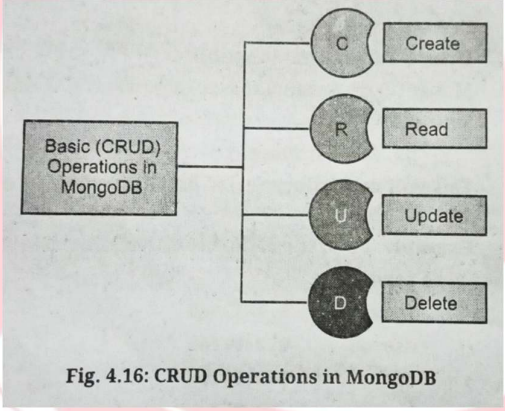


Fig. 4.16: CRUD Operations in MongoDB

1. **Create operation**

   The **create** or **insert** operations are used to add new document into the collection.

   If the collection does not exist, then the insert operation will create the collection.

   The different methods to insert document into a collection:

   - db.collection.insertone ( )
   - db.collection.insertmany ( )

   E.g:

   Db.student.insertOne

   (

   {

   Name:”Kunal”

   Age: “17”

   Status: “file pending”

   grade: “A”

}
)

2. **Read operation**

It is used to **retrieve** the documents from the collection.
The **find ( ) command** is used to queries a collection for documents or simply to retrieve the documents from the collection.
Syntax:
Db.collection.find ( )
E.g:
db.student.find ( ) – for all documents or you can retrieve specific document with the help of attributes of that document.
E.g:
db.student.find(
{
name:"Kunal"
}
)

3. **Update operation**

This operation is used to modify the existing documents in a collection.
Different methods are used for updation as
-db.collection.updateone ( )
-db.collection.updateMany ( )
-db.collection.replaceOne ( )
The Mongo DB uses the update operation for a **single collection**. One can update the all documents without specifying any criteria.
One can update specific document by providing specific criteria.
E.g: db.collaction.updateMany ( )
E.g:
db.student.updateOne (
{
Name:{$ SN: "Kunal"} **--update filter**
},
{ $ set: { status: "completed"} **--update action.**
}
)

4. **Delete operation**

- It is used to **remove documents from a collection**.
- Different **methods** to delete documents are
  -db.collection.deleteOne ( )
  -db.collection.deleteMany ( )
- The delete operation is performed on a single collection.
- To delete the specific document, you have to provide the specified criteria as per requirements.

E.g:
db.student.deleteMany ( ) or
db.student.deleteOne(
{
Name:{$SN: "Kunal"}
}
)

8. Explain basic datatypes and arrays in MongoDB.

Basic Datatypes:
Mongo DB stores document on disk in the BSON serialization format.
BSON is the binary representation of JSON documents. Hence BSON format provides more data types than JSON.

Data types supported by Mongo DB are:

- **String:** String in Mongo DB must be UTF-8 valid.
- **Integer:** Integer can be 32 bit or 64 bit depending upon server.
- **Boolean:** This type is used to store a Boolean values as True or false.
- **Double:** Used to store floating point values.
- **Min/Max keys:** This type is used to compare a value against the
- lowest and highest BSON elements.
- **Arrays:** Used to store multiple values into one key.
- **Timestamp:** Used for storing the time when a document has been modified or added.
- **Object:** This datatype is used for embedded documents
- **Null:** Used to store a null value
- **Symbol:** Used to identically convert symbol into string. This datatype is not supported by the shell. If the shell gets a symbol from the database, it will convert it into a string.
- **Date:** Used to store the current date or time in UNIX time format.
- **Object ID:** This datatype is used to store the documents ID.
- **Binary data:** Used to store binary data.
- **Code:** This is used to store Javascript code into the document.
- **Regular expression:** Used to store regular expression

Arrays: Arrays are values which can be interchangeably referred for both ordered operating as lists, stack or queues or for unordered operations as sets.
Arrays in Mongo DB are able to store different data types values.
E.g:

```
{
    "things": ["pi", 3.14]
}
```

Mongo DB enables atomic updates which helps to modify the contents of arrays.

9. Explain the association rule in data mining. Explain application of association rule mining with example.

- Association rules are if-then statements that help to show the probability of relationships between data items within large data sets in various types of databases.
- Association rule mining has a number of applications and is widely used to help discover sales correlations in transactional data or in medical data sets.
- Association rule mining, at a basic level, involves the use of machine learning models to analyze data for patterns, or co-occurrence, in a database.
- It identifies frequent if-then associations, which are called association rules.
- An association rule has two parts: an antecedent (if) and a consequent (then). An antecedent is an item found within the data. A consequent is an item found in combination with the antecedent.

Application of association rule mining:

1. Medical diagnosis.
2. Protein Sequences.
3. Fraud Detection in Credit Card Transactions.
4. Bio-Medical Literature.
5. Customer Relationship Management (CRM).
6. Census Data etc.
7. Market Basket Analysis

1) Market Basket Analysis: This is the most typical example of association mining. Data is collected using barcode scanners in most supermarkets. This database, known as the "market basket" database, consists of a large number of records on past transactions. A single record lists all the items bought by a customer in one sale. Knowing which groups are inclined towards which set of items gives these

shops the freedom to adjust the store layout and the store catalog to place the optimally concerning one another.

2) Medical Diagnosis:

Association rules in medical diagnosis can be useful for assisting physicians for curing patients. Diagnosis is not an easy process and has a scope of errors which may result in unreliable end-results. Using relational association rule mining, we can identify the probability of the occurrence of illness concerning various factors and symptoms. Further, using learning techniques, this interface can be extended by adding new symptoms and defining relationships between the new signs and the corresponding diseases.

10. Write query to execute find () function on collection: Inventory.

(i) to display all document where status equals "A" and "G".

(ii) to display all document in collection.

(iii) to display all documents where quality is less than 30 and greater than 10

Query :-

```
db.inventory.insertMany
(
        [
        {
                First_Name: "Radhika",
                Last_Name: "Sharma",
                qty: 30,
                e_mail: "radhika_sharma.123@gmail.com",
                status: "B"
        },
        {
                First_Name: "Rachel",
                Last_Name: "Miller",
                qty: "15",
                e_mail: "Rachel_Christopher.123@gmail.com",
                status: "G"
        },
        {
                First_Name: "Fathima",
                Last_Name: "Sheik",
                qty: "25",
                e_mail: "Fathima_Sheik.123@gmail.com",
                status: "A"
        }
        ]
)
i) db. inventory.find({$and : [{"status": "A"},{"status": "G"}]})
ii) db.inventory.find()
iii) db.inventory.find({$and: [{ "qty": {$lt: 30}},{ "qty": {$gt: 10}}]})
```

11. Define lock. Explain two phase locking protocol with neat example.

Locks: Lock is a data variable which is associated with a data item.

Locks help synchronize access to the database items by concurrent transactions.

All lock requests are made to the concurrency-control manager. Transactions proceed

TWO PHASE LOCKING PROTOCOL: -

Two phase Locking protocol: which is also known as 2PL. Two phase locking protocol requires that each transaction issues lock and unlock requests in two phases.

Growing phase: A transaction may obtain locks but may not release any lock.

Shrinking phase: A transaction may release locks, but may not obtain any new locks.

If the conversion is allowed, then upgrading of locks from S(A) to X(A) happens in growing phase and the downgrade of locks from X(A) to S(A) happens in shrinking phase.

It is true that 2PL protocol offers serializability. However, it does not ensure that dead locks not happen.

Example: -

	T1	T1
1	Lock-S(A)	
2		Lock-S (A)
3	Lock- X(B)	
4	---	---
5	Unlock(A)	
6		Lock- X(C)
7	Unlock(B)	
8		Unlock(A)
9		Unlock(C)

This is just a skeleton transaction that shows how unlocking and locking work with 2-PL. Note for:

Transaction T1:

- The growing Phase is from steps 1-3.
- The shrinking Phase is from steps 5-7.
- Lock Point at 3

Transaction T2:

- The growing Phase is from steps 2-6.
- The shrinking Phase is from steps 8-9.
- Lock Point at 6

12. What is a transaction? What are the states and properties of transaction?

Transaction:

"A transaction is a unit of program execution that accesses and possibly updates various data items"

- Transaction consists of operations executed between beginning and ending of transaction
- Operations performed during transaction: insert, delete, update, retrieve
- Transaction is a collection of actions – preserving system consistency, concurrency transparency and failure transparency

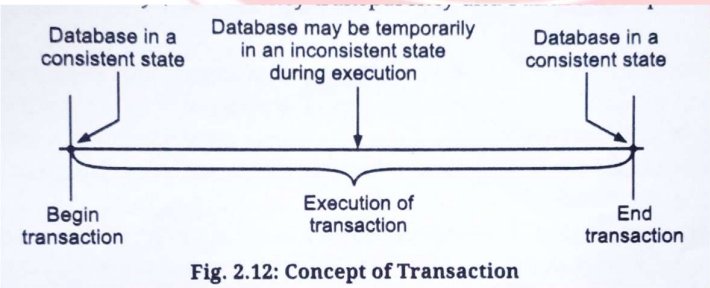


Fig. 2.12: Concept of Transaction

Operations performed in a transaction:

- **begin_transaction:** start of transaction execution
- **read_item or write_item:** main memory operation
- **end_transaction:** marker specifying end of transaction
- **commit:** signal specifying successful completion of transaction & will not be undone

- **rollback:** signal specifying transaction is unsuccessful and temporary changes in DB are undone.
- Committed transaction cannot be rolled back

Transaction states:

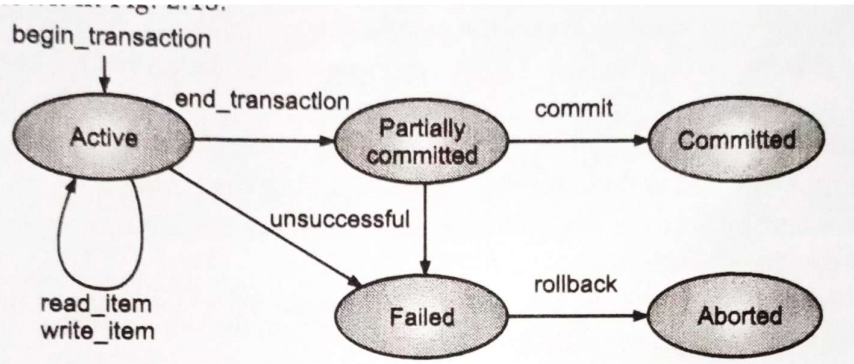


Fig. 2.13: States in Transaction

- **Active:** initial stage- transaction remains in active stage while it is executing read, write or other operations
- **Partially committed:** transaction enter in this state when the last statement has been executed
- **Committed:** transaction enters this state after successful completion of transaction and system checks have issued commit signal
- **Failed:** transaction enters failed state when normal execution cannot proceed or system checks fail
- **Aborted:** this is the state after transaction has been rolled back after failure. DB has been restored to previous state where the transaction began.

Properties of transaction:

- Transaction must maintain **ACID** properties
- **Atomicity:**
  - Transaction is an atomic unit of processing
  - Either it is performed entirely or not performed at all
  - No partial update should exist
- **Consistency:**
  - Its about correctness of transaction
  - Transaction should not adversely affect the database
- **Isolation:**
  - Transaction should be executed as if it is the only one in the system
  - No interference from other concurrent transaction running simultaneously
- **Durability:**
  - The change should be durable in database and not lost in case of any failure

13. Types of parallel database architectures:

There are 4 types of parallel database architectures

1. Shared memory architecture
2. Shared disk architecture
3. Shared nothing architecture
4. Hierarchical architecture

1. **Shared memory architecture:**

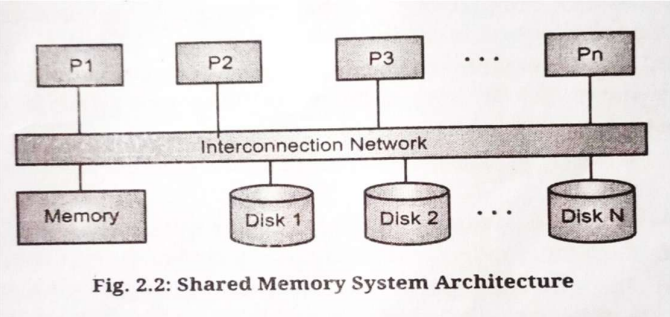


Fig. 2.2: Shared Memory System Architecture

- Multiple CPU attached to interconnection network
- Single / global main memory
- Common disk array (Storage)
- Also known as symmetric multi-processing (SMP)
- Processors can send messages using memory rights in microseconds
- Not scalable beyond 64 processors
- Ex. XPRS, DBS3 Volcano

Advantages

- Simple and provides load sharing
- Communication overhead is low
- Communication between CPUs is extremely efficient
- Provides high speed data access

Disadvantes

- Limited extensibility & low availability
- High cost
- Interconnection networks become bottleneck as number of CPUs increases
- Addition of CPU increases waiting time for their turn on bus to access memory.

2. **Shared disk architecture**

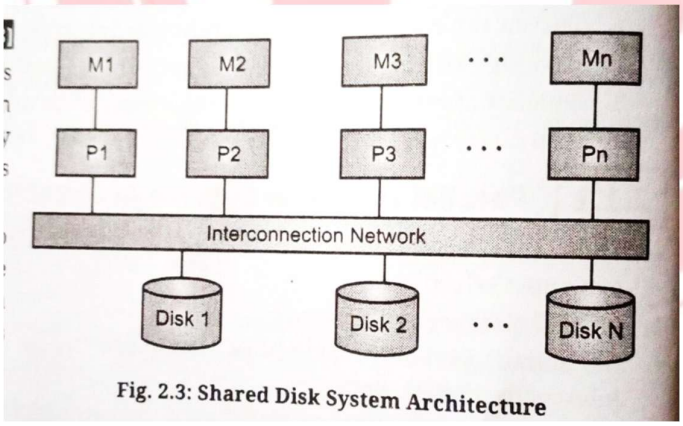


Fig. 2.3: Shared Disk System Architecture

- Multiple CPUs with own memory
- All have access to same disk
- Memory is not shared among CPUs
- Inherently centralized application
- Each CPU can assess all disks but have own private memory
- Also named as **CLUSTERS.**
- Commercial Users **DEC** (Digital Equipment Corporation) – Cluster running Rdb
- Rdb is owned by Oracle – called Oracle Rdb

Advantages

- Easy to load balance
- Low-cost solution to provide degree of fault tolerance
- Each CPU has own memory so bus is not a bottleneck
- Less cost than shared memory architecture
- High performance, high availability, high extensibility.

Disadvantages

- More complex as it requires distributed database
- Communication across processors is slower
- Scalability can lead to bottleneck

3. Shared nothing architecture

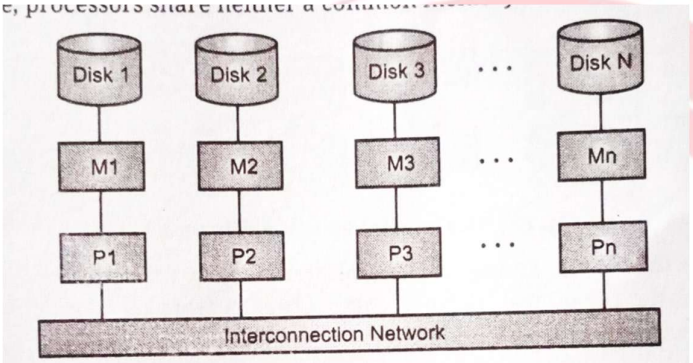


Fig. 2.4: Shared Nothing System Architecture

- Processors have own memory and disk
- NO sharing of memory and disk
- Each CPU has own copy of OS and DBMS
- Also called as massively parallel processing (MPP)
- Communication using high speed interconnection network
- Commercial user – Teradata database machine, Grace, Gama research prototype

Advantages

- Highly scalable
- Provides linear speed – up and linear scale-up
- Easily supports large number of CPUs
- Extensible
- Repication of data on multiple nodes proves high scalability

Disadvantages

- Difficult to load balance
- Requires OS capable of handling heavy amount of messaging to support inter processor
- Communication
- Higher cost of communication with non-local disk
- More complex – for large number of nodes

4. Hierarchical architecture

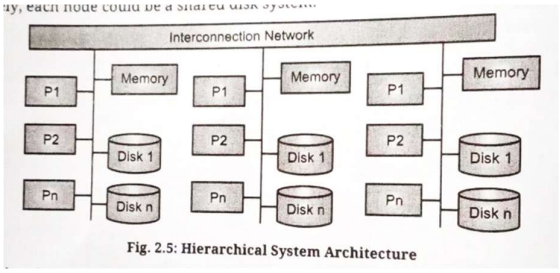


Fig. 2.5: Hierarchical System Architecture

- Hybrid (Mixed) of shared memory shard disk and shared nothing
- Combined characteristics of all 3
- Top level is shared nothing architecture
- Each node is actually a shared memory system
- Also called as NUMA- NON uniform memory architecture
- Uses local and remote memory.
- SO longer communication time

Advantages

- Improved scalability
- Minimized bottleneck
- Greater flexibility, greater performance
- Higher data availability, high extensibility

Disadvantages

- High cost
- Complex architecture

14. Explain measures of performances of parallel database system

1. **Throughput**
   - Rate at which database system can process operation within given time period
   - Number of tasks that can be completed in given time interval
2. **Response time**
   - Time taken to execute specific request
   - Amount of time to complete single task from the time it is submitted
3. **Scale up**
   - Ability to keep same performance level when workload and resources increases proportionally
   - Scaleup means handling a larger task by increasing the degree of parallelism
   - Scale up = Small system small problem elapsed time (single machine)/ Large system large problem elapsed time (parallel machine)

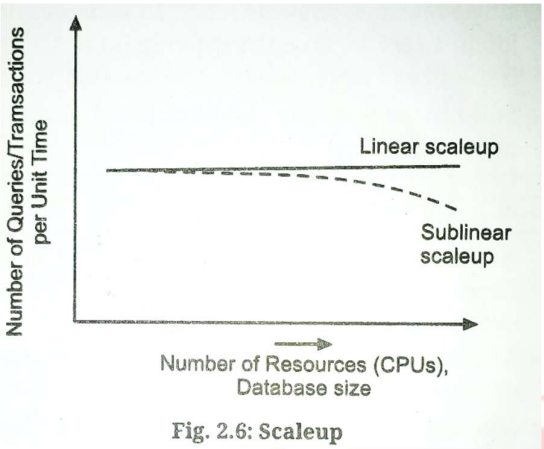


Fig. 2.6: Scaleup

a. **Linear scale up**
   - Scale up is linear if resources increase in proportion with problem size
   - It's the ability to maintain same level of performance when workload and resources are proportionally added.

b. **Sub-linear scale up**
   - If large number of problem elapsed time > small system small problem elapsed time

4. **Speed up**
   - Ability to reduce response time of large queries by adding more resources
   - Performance improvement gained as extra processing element added.
   - It's like increasing the degree of parallelism
   - It's measurement of performance of read only queries (data retrieval)
   - Speedup = small system elapsed time / large system elapsed time
   - Speed up reduces response time

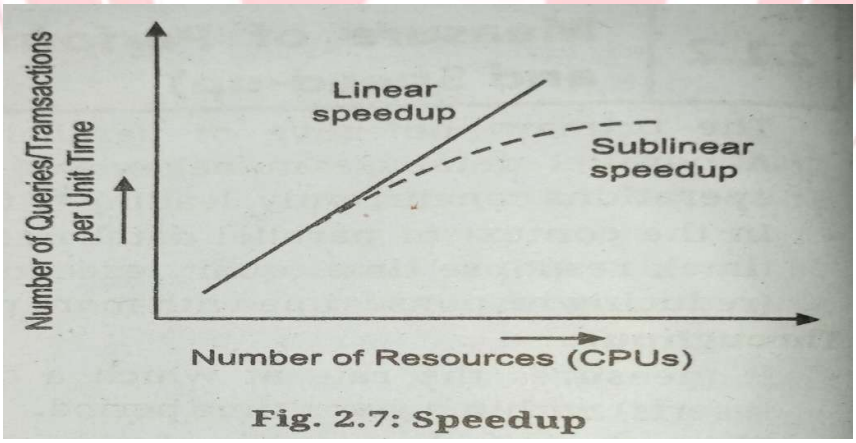


Fig. 2.7: Speedup

a. Linear speedup
   - **Linear speed = N (small system elapsed time)**
   - Single system time = 10
   - Parallel system time = 1
   - So, speed up = 10/1 = 10

b. Sub-linear speed up
   - Speedup < N

15. What is DDBMS? Explain architecture, What are different types of Distributed Database System?

DDBMS:

- Manages distributed database
- Provides mechanism to make databases transparent to users
- **"DDBMS is a centralized software system that manages DDB and provides an access mechanism that makes distribution transparent to the users"**

Features of DDBMS:

- DDBMS is used to – create, retrieve, update and delete distributed databases
- It is designed for heterogeneous database platforms
- It is used for processing large volume of data accessed by numerous users simultaneously
- DDBMS maintains confidentiality and data integrity
- It ensures data modified at any site is universally updated
- applications can access database locally and remotely

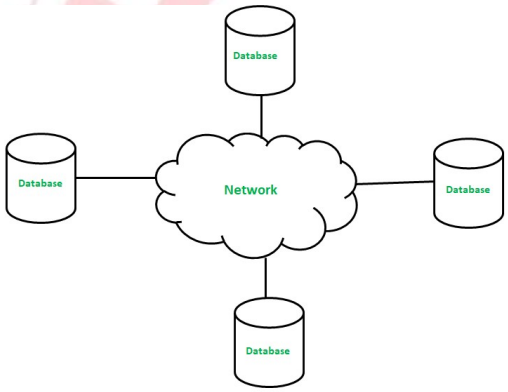
Applications in DDBMS- 2 categories:

1. **Local applications**

Requires access to local data only

2. **Global applications**

Requires access to data from remote sites in distributed system



Architecture of DDBMS

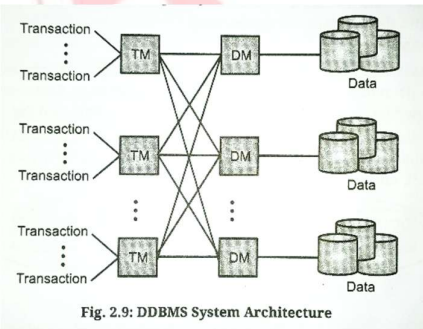


Fig. 2.9: DDBMS System Architecture

Its collection of sites interconnected by a computer network.

Components of DDBMS:

- **Computer Workstation(sites/nodes):**
  - DDBMS consists of number of computer workstations that form network system
  - Database system must be independent of computer system hardware
- **Communication Media:**
  - Communication among nodes- data transfer, information exchange→carried out by communication media

- It's very important component of DDBMS
- DDBMS must support several types of communication media

- **Transaction Processor (TP):**
  - Software component resides in each computer - responsible for receiving and processing both local and remote application data requests
  - Also known as AP-Application Processor OR TM- Transaction Manager
- **Data Processor (DP):**
  - Software component resides in each computer – stores and retrieves data located at that site
  - Also known as DM- Data Manager
  - In DDBMS- DP maybe centralized DBMS
- Each site can run both TM and DM
- TM-supervises interactions between users and DDBMS
- DM- manages actual DB

DDBMS are of 2 types:

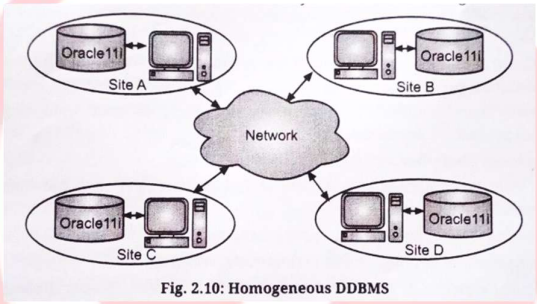
- **Homogeneous DDBMS:**
  - All sites use same DBMS product
  - All sites share common global schema
  - Run identical DBMS software
  - 

Fig. 2.10: Homogeneous DDBMS
  - DBMS softwares on each site work together in coordination
  - All sites use identical DBMS and OS
  - **Properties**:
    - Each site is aware of other sites
    - Each cooperates with other sites to process user request
    - Sites use similar software
    - Database is accessed through a single interface as if it is a single database
  - **2 types of Homogeneous DDB:**
    - **Autonomous:** each DB functions on its own independently
    - **Non-Autonomous:** data is distributed – central or master DBMS co-ordinates updates across sites

- **Heterogeneous DDBMS:**
  - Distributed database constructed by linking multiple already existing DB system together- with own schema but different database management software
  - Sites may run different DBMS product based on underlaying data model (relational, network, hierarchical and object-oriented DBMS)

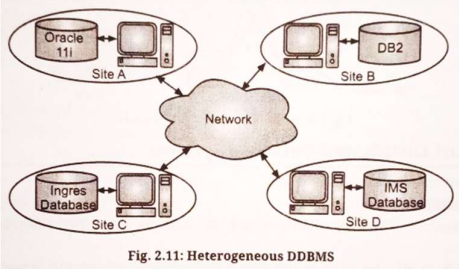


Fig. 2.11: Heterogeneous DDBMS

- Sites may not be aware about each other
- May provide limited facilities for co-operation in transaction processing
- Sites have different – OS, DBS and data models
- **Properties**:
  - Different sites use dissimilar schemas and software
  - System is composed of a variety of DBMS
  - Query processing is complex due to dissimilar schemas
  - Sites may not be aware of other sites – so less co-operation in processing user request
- **2 types of Heterogeneous Databases:**
  - **Federated**: independent in nature, integrated together to function as a single DB system
  - **Un-Federated:** DB system employ- central co-ordinating module through which DB is accessed

16. What are Components of Data Warehouse

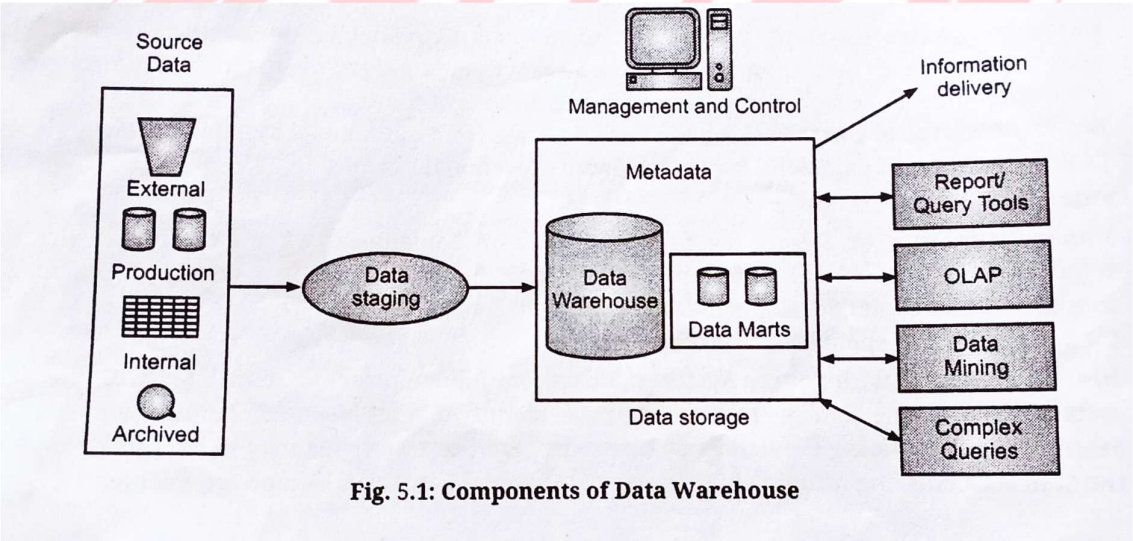


Fig. 5.1: Components of Data Warehouse

1. Source Data

This is the raw input collected from various origins:

- **Production Data:** Real-time operational data from transactional systems.
- **Internal Data:** Data generated within the organization (e.g., HR, finance).
- **Archived Data:** Historical records stored for long-term analysis.
- **External Data:** Data from outside sources like market feeds or third-party vendors.

2. Data Staging Component

Prepares data before it enters the warehouse:

- **Data Extraction:** Pulls data from source systems.
- **Data Transformation:** Cleanses and formats data to match warehouse standards.

- **Data Loading:** Inserts transformed data into the warehouse.

3. Data Storage

- Final storage area for transformed data.
- Components:
  - **Data Warehouse** – Central repository using schemas (star, snowflake, galaxy).
  - **Data Mart** – Subset of warehouse for specific departments (e.g., sales, finance).
  - **Metadata** – Info about data (source, extraction time, frequency, etc.).
  - **Data Loading** – Periodic or one-time loading from staging area.
  - **Indexing** – Partitioning and indexing for fast access.

1. Information Delivery Component

- Tools and methods to present data to users.
- Includes:
  - **Query and reporting tools**
  - **Analysis and visualization tools**
  - **Data mining tools**

2. Metadata Component

- Stores data about data which includes:
  - Logical structures
  - File paths
  - Indexes
  - Source and transformation details

3. Data Management & Control Component

- Oversees warehouse operations and services.
- Tracks data movement and interactions.
- Coordinates updates, refreshes, and purges.
- Manages metadata and controls access paths.

17. Explain Virtual warehouse

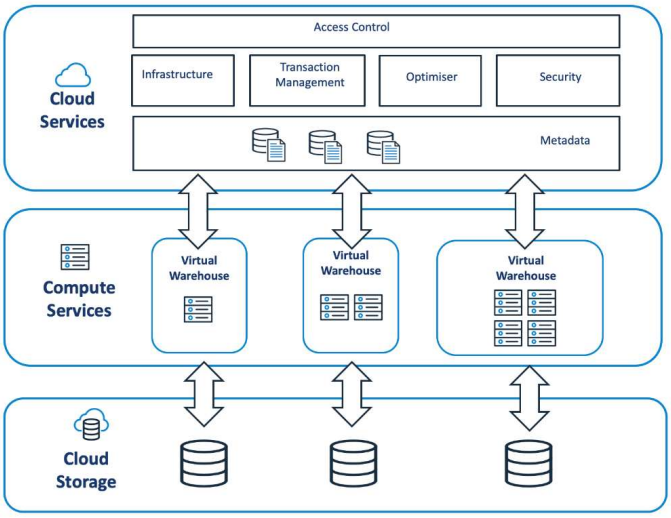
Virtual Warehouse

- A virtual warehouse is a set of views over operational databases. For efficient query processing, only some of the possible summary views may be materialized.
- **The view over an operational data warehouse is known as a virtual warehouse**.
- A virtual warehouse is a digital representation of inventory, enabling businesses →to manage stock across multiple locations or channels through software, providing real-time visibility and control, even if the physical inventory is stored in different places.
- Virtual warehousing leverages technology to track and manage inventory, providing a single, holistic view of stock levels, regardless of where it's physically stored.

Architecture of Virtual warehouse

- Data warehouse architecture is typically organized into **layered components** that streamline the flow of data from source systems to end-user analysis.
- It begins with **source data**—including internal, external, and archived information—which is **processed through the data staging** area using ETL (Extract, Transform, Load) operations.
- The **cleaned** and **transformed** data is then stored in the data storage layer, optimized for querying and analysis.
- Above this, the **information delivery layer** provides tools for reporting, dashboards, and OLAP operations.
- Supporting all layers are the metadata component, which describes the data, and the management and control component, which oversees performance, security, and scheduling.

- In modern cloud-based platforms, virtual warehouses act as scalable compute engines that interact with cloud storage and services to execute queries efficiently.



Layer	Interaction with Virtual Warehouse
Cloud Services	Manages query optimization, transaction control, security, access control, and metadata. It directs the VW on what tasks to perform.
Virtual Warehouse	Executes queries, transforms data, and handles compute-intensive operations. Multiple VWs can run in parallel for scalability.
Cloud Storage	Stores the **actual data**. VWs retrieve data from here for processing and analysis.

Benefits of Virtual warehouse

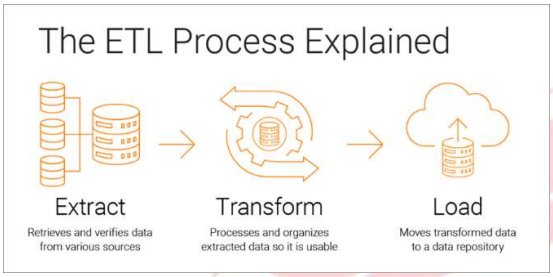
1. **Automation:** Virtual warehousing **automates** various inventory management tasks, such as stock replenishment and picking. This reduces the need for manual labour and eliminates human error.
2. **Enhanced Efficiency:** Optimized inventory management and streamlined supply chain operations.
3. **Improved Inventory Control:** Real-time visibility and tracking of inventory levels across all locations.
4. **Better Decision-Making:** Data-driven insights for informed decisions about inventory levels and replenishment times.
5. **Reduced Costs:** Improved inventory accuracy and reduced waste through better planning and management.
6. **Improved Data Analytics:** Monitoring the performance of multiple physical warehouses from a centralized view can provide valuable insights into the overall inventory health, allowing businesses to analyse trends and optimize stock accordingly.
7. **Increased Visibility:** In virtual warehousing systems, all operations are monitored digitally from one source, giving teams extensive visibility over their inventories in real-time. This helps increase accuracy and drive efficiency in warehouse operations.
8. **Reduced Storage Costs:** By managing stocks more efficiently through an automated system, companies can reduce storage costs associated with **traditional warehousing**.
9. **Increased Agility:** Virtual warehousing enables businesses to easily adapt to changing customer needs, enabling them to deliver products **quickly** and **efficiently** while providing better customer service.

Challenges of Virtual warehouse

1. **Data Security:** sensitive data need more protection
2. **Integration:** Businesses may find it difficult to integrate the necessary systems into their operations, and many companies need to hire skilled personnel to guide them through the process
3. **Initial Setup Costs:** Setting up a virtual warehouse require initial investment of time, resources and money which may be costly for some businesses.

18. Define ETL process with neat diagram

One of the key **tools** used in data warehousing is ETL (Extract, Transform, Load) tools.



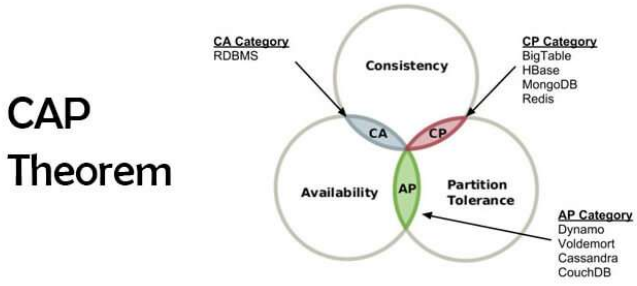
Extract-Transform-Load function involves a 5-step workflow.

1. **Data Extraction** - Collection of raw data from different sources.
2. **Data Cleaning** - Sanitisation of raw data collected from unstructured sources, flat files; removing anomalies.
3. **Data Transformation** - The raw data is cleaned, formatted, and transformed into a usable structure.
4. **Load** - The processed data is loaded into a target system, such as a data warehouse or database.
5. **Refresh** - Replaces all existing data in the target system with the latest data.

19. Explain CAP and BASE Theorems

- NoSQL follows CAP theorem → also called as Brewer Theorem
- Basic requirements of CAP theorem are:

1. **Consistency:** data should remain consistent after any kind of transaction on DB
2. **Availability:** system is always available without any down-time
3. **Partition tolerance:** system must function reliably after partitioning of server though they may not communicate with each other
   - **CAP suggests to fulfill at least 2 requirements**



1. **CA (Consistency & Availability)**
   - Nodes are always in contact and are available
   - System fails if partition occurs
2. **CP (Consistency with Partitioning)**
   - After partitioning – consistent data will be available→may not be accessible all the time
3. **AP (Availability with Partitioning)**

- System is available under partitioning
- But some of the data may be incorrect

BASE Theorem

- **Basically Available (BA):** system will be available in terms of CAP theorem
- **Soft (S) state:** indicates that→ even if no input is provided to the system- state will change over time
- **Eventual (E) Consistency:** system will attain consistency in long run→provided no input is sent to the system during that time

20. Enlist Types of NoSQL database, Explain Any two NoSQL database

There are 4 types of NoSQL database

1. Document – oriented databases
2. Graph databases
3. Column – oriented databases
4. Key – value databases

A. Document – oriented databases/Document – Store databases

- All data for a single entity can be stored as a document and documents can be stored together in collection
- Document can contain- all necessary information to describe an entity
- Documents in the collection are accessed via unique key

Data model for Document – oriented databases

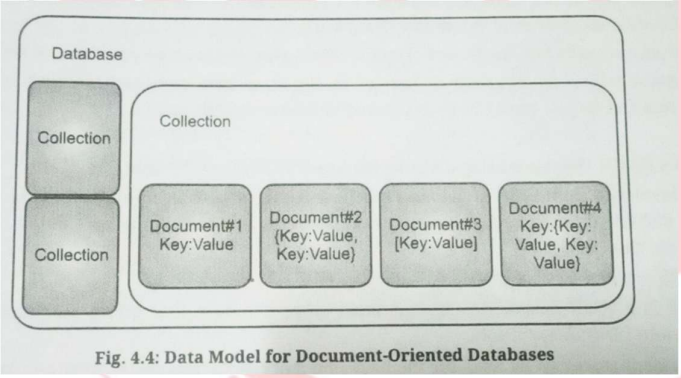


Fig. 4.4: Data Model for Document-Oriented Databases

- Unit of data is called – Document
- Table containing group of documents is called as- "collection"
- Fig. shows document-store database
- Database contains many collections
- Collection contains many documents
- each document might contain JSON document or XML document or Word document
- document database is suitable for Web based applications and applications exposing RESTful services
- (REST or Representational State Transfer is an architectural style that can be applied to web services to create and enhance properties like performance, scalability and modifiability.)

For ex.

1. **MongoDB:**
   - Open-source database
   - Used by companies and industries for wide variety of applications
   - It's an agile database→ allows schemas to change quickly as application evolve
   - Built for scalability, performance, high availability scaling→ from single server to large complex multisite architectures

2. **CouchDB:**
   - It's a database that completely embraces Web
   - Stores the data with JSON document
   - Accesses the document and query indices with web browser→via HTTP, Index, →combine and transform document with JavaScript
   - Works well in modern web and mobile apps
   - We can serve web apps directly out of CouchDB
   - Data and apps can be distributed using CouchDB's incremental replication
3. **Couchbase Server:**
   - Originally known as Membase
   - Open source, distributed, NoSQL document-oriented database→ optimized for interactive applications
   - Applications must service→ many concurrent users for creating, sorting, retrieving aggregating manipulating and presenting data
4. **MarkLogic Server:**
   - It's an enterprise NoSQL database
   - It fuses together database internals, search style indexing and application server behaviors into a unified system
   - Uses XML documents as its data model
   - Stores documents within transactional repository
   - Indexes the words and values from each of loaded documents
5. **ClusterPoint Server:**
   - DB software of high-speed storage and large-scale processing of XML and JSON data → on clusters of commodity hardware
   - Works as schema free document-oriented DBMS platform
   - With Open-source AI
   - Solves the problem of latency in Bigdata
   - Billions of documents can be searched and analyzed fast
6. **JasDB:**
   - It is a NoSQL database
   - Uses document-based storage mechanism
7. **RaptorDB:**
   - It is JSON based NoSQL document store database
   - Offers automatic hybrid bitmap indexing and LINQ query filters

B. Graph databases/ Graph – store Database

- Database is represented as graphs
- Vertices and edges→ are nodes and relations in this graph
- Graph DB works better than directed graphs
- Used to store networked data

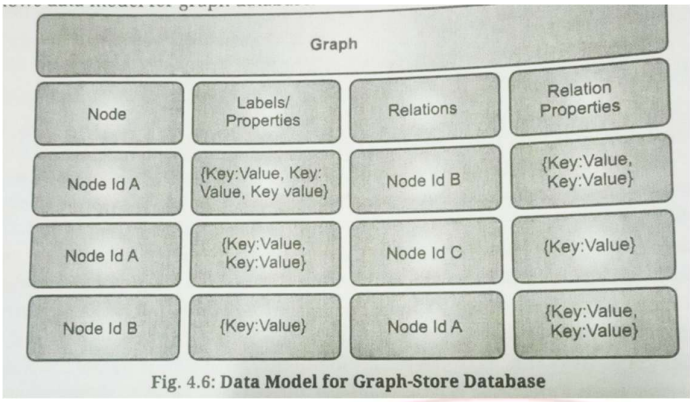


Fig. 4.6: **Data Model for Graph-Store Database**

Data Model for graph database

- Graph database is a 2-dimensional representation of graph
- Similar to tables
- Each graph contains- nodes, node properties, relation, relation properties ad columns
- There are values for each row for these columns
- Values for properties column-can have key-value pair
- Graph DB is suitable for- social media, network problems→ involving complex queries with more joins

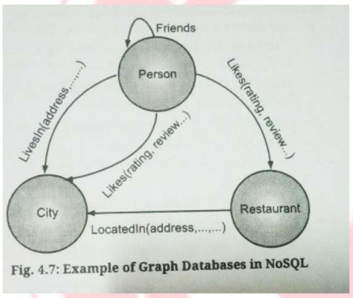


Fig. 4.7: **Example of Graph Databases in NoSQL**

For ex.

- Nodes have different type of relationship between them
- There is no limit to number and kind of relationship
- All can be represented in the same graph database

- Popular examples of graph database: -

1. **Neo-4J:**
   - Java-based open-source NoSQL graph database
   - Using Neo4J we can search social network data, connections between data are explored
   - It can solve problems that require network probing
   - Performance is high
2. **Infinite Graph:**
   - Distributed graph database implemented in Java
   - Belongs to the class of NoSQL
   - Data technologies focused on graph data structure
   - Graph data consist of objects or things(nodes) and relationship(edges) – connecting 2 or more nodes
   - Developers may use infinite graph to build web and mobile applications
3. **Allergo Graph:**
   - Modern, high performance, persistent graph database
   - Efficient memory utilization with disk-based storage
   - Can be scaled up to billions of quads → still maintaining superior performance

- Supports SPARQL, RDFS++ and prolog reasoning from numerous client apps

4. **Virtuoso:**
   - It's a universal server for middleware and database engine
   - Combines functionality of traditional RDBMS, ORDBMS, Virtual Database, RDF< XML<free-text, web development server and file server functionality→in a single system
   - It's open-source edition known as Open Link Virtuoso
5. **FlockDB:**
   - Open source distributed, fault tolerant graph data base
   - Manages wide but shallow network graphs.
   - Initially used by Twitter to store relationship between users
     - Ex. Followings and favorites
6. **VertxDB:**
   - High performance graph database
   - Supports automatic garbage collection
   - Uses HTTP protocol for requests
   - JASON for response data format
   - Its API is inspired by FUSE file system API

C. Column – oriented databases/ Column – store databases.

- Development is based on Big Table White Paper published by Google
- Different approach than traditional RDBMS
- Supports more column and have wider table
- It supports group of columns with family name → "column family "or "Super column"
- Stores data in columns within a key space
- Key space is based on unique name, value, and timestamp.

Data model for Column – store databases

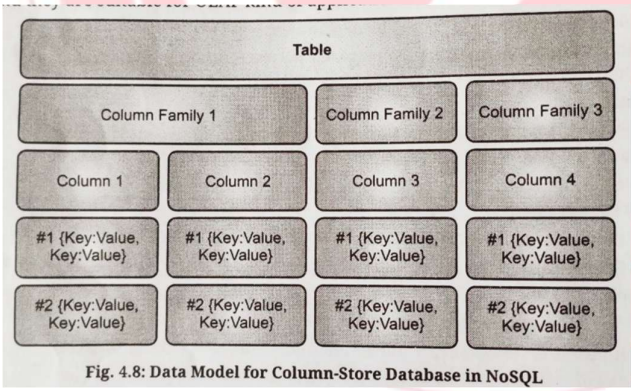


Fig. 4.8: Data Model for Column-Store Database in NoSQL

- Table contains column families
- Each column family contains many columns
- Values for column might be sparsely distributed with key – value pair
- This is an alternative to data warehousing database such as Teradata
- Suitable for OLAP (Online Analytical Processing) kind of application

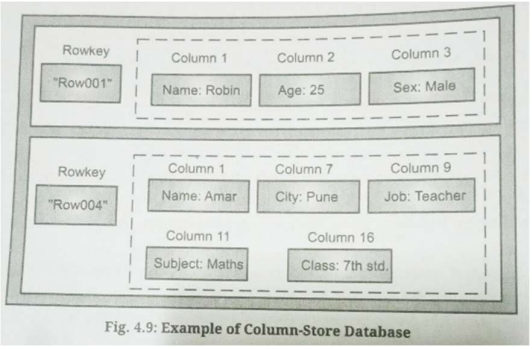


Fig. 4.9: **Example of Column-Store Database**

For ex.

1. **Hadoop/HBase:**
   - Apache HBase is – open source, distributed, versioned, non-relational database→ modelled after Googles Big Table
   - HBase provides Bigtable like capabilities on the top of Hadoop and HDFS
2. **Cassandra:**
   - Apache Cassandra DB – scalable highly available without compromising performance.
   - Linear scalability, fault tolerance
   - Makes it perfect platform for mission-critical data
   - Data is replicated at multiple data centres
3. **HyperTable:**
   - High performance, open source massively scalable DB modelled after Bigtable
4. **Accumulo:**
   - Based on Googles Big table design
   - Built on top of apache Hadoop, Zookeeper and Thrift
   - Provides scale-based access control and server-side programming mechanism
5. **Amazon SimpleDB:**
   - Highly available, flexible, non-relational data store
   - Offloads the work of database administration
   - Developer can store and query data items via web services requests
   - Using Amazon simple DB – we can focus on application development – without worrying about infrastructure, availability, maintenance, schema, index management, or performance tuning

D. Key – value databases/ Tuple – store databases

- Developed on the basis of – Dynamo whitepaper – published by Amazon
- Simplest type of NoSQL database
- Completely schema-less
- Key can point to any type of data
- Data is stored in simple key-value pair→<key>:<value>
- Key is used to retrieve the value from the table

Data model for Column – store databases

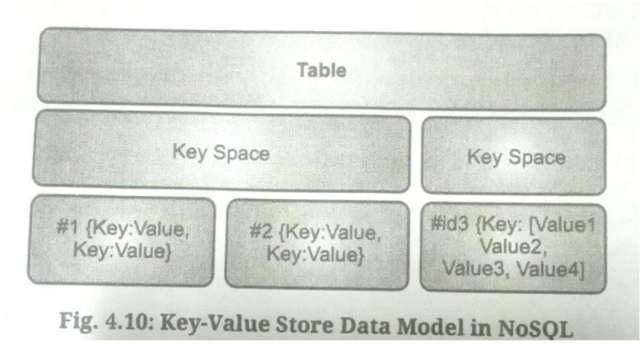


Fig. 4.10: Key-Value Store Data Model in NoSQL

- **Table:** contains many key spaces
- Key space: considered as rows
- Used for building – simple, non-complex, high available applications

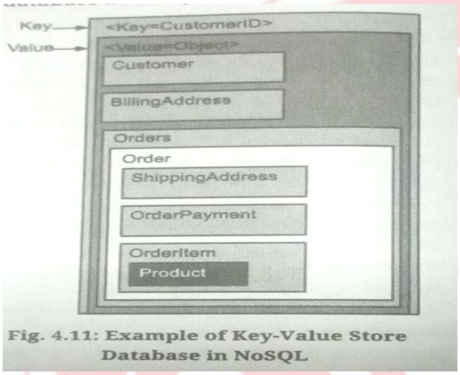


Fig. 4.11: Example of Key-Value Store Database in NoSQL

For ex.

1. **Redis:**
   - It is a NoSQL key-value data store
   - It is a data structure server
2. **Riak:**
   - It uses simple key-value model for object storage
   - Object consist of unique key and value
   - It is stored in a namespace called bucket
   - We can store- text, images, JSON/XML/HTML documents, user session data, backups, log files etc.
3. **Aerospike:**
   - World fastest, more reliable in-memory open source NoSQL DB
   - Operates with unprecedented speed
   - Wide range of applications→from web portals to universal profile stores for real-time bidding an cross-channel marketing platforms
4. **FoundationDB:**
   - Supports ACID transactions
   - High performance with NoSQL and highly scalable
   - Supports global transactions over any number of keys
5. **Amazon DynamoDB:**
   - Fast and fully managed by NoSQL DB
   - Simple and cost efficient
   - Can retrieve any amount of data and serve any amount of request traffic
   - It's a great fit for gaming, ad tech, mobile etc.
6. **Berkeley DB:**
   - BDB- it's a s/w library with high performance embedded db

- It is written in C with API bindings for C++, C#, PHP, JAVA< Perl, Python, Ruby, Tcl, Smalltalk etc.
- Supports multiple data items with single key
- It's a non-relational DB
- Supports thousands of threads and concurrent processes

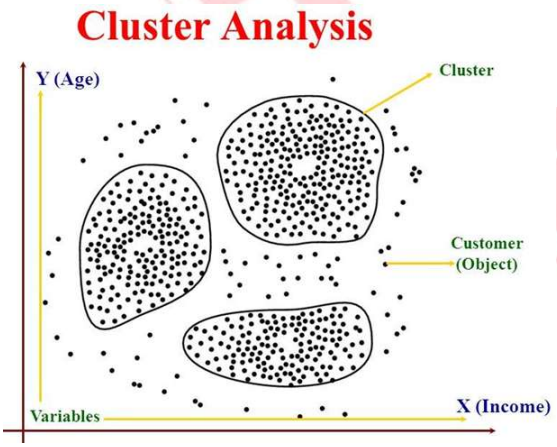
7. **Oracle NoSQL:**
   - Distributed key-value database
   - **Features :** reliability, scalability, availability
   - Data is stored on nodes
   - Uses hash value of primary key
   - Storage nodes are replicated to ensure high availability
   - Applications are written in JAVA/C API to read and write data

21. Explain Clustering Cluster analysis in Data Mining

- Clustering is an **unsupervised data mining technique** that groups similar data objects into **clusters**, so that items in the same cluster are alike, and items in different clusters are distinct.
- No predefined labels—**data is grouped based on similarity**.
- Used in market segmentation, customer profiling, image processing, and pattern recognition.
- Helps discover **hidden structures** in data without prior knowledge.

Cluster analysis

- **Clustering analysis** is an unsupervised data mining technique used to group similar data objects into clusters, where:
  - Objects **within a cluster** are highly similar.
  - Objects **between clusters** are significantly different.
  - To discover hidden patterns or natural groupings in data without predefined labels.
  - Commonly used in market segmentation, customer profiling, image processing, and anomaly detection



Clustering method in data mining:

1. **Density-Based Clustering Method:** Groups data based on regions of **high** density separated by regions of **low** density.

2. **Grid-Based Clustering Method:** Divides the data space into a grid structure and performs clustering on the grid cells.
3. **Model-Based Clustering Method:** Assumes a statistical model for each cluster and finds the best fit for the data.
4. **Constraint-Based Clustering Method:** Incorporates user-defined constraints to guide the clustering process.
5. **Partitioning Clustering Method:** Divides data into *k* clusters by optimizing a criterion like distance (e.g., K-means).
6. **Hierarchical Clustering Methods:** Builds a tree of clusters using either bottom-up (agglomerative) or top-down (divisive) approaches.

22. Discuss Architecture of the Data Lakehouse

Its architecture is layered to support scalable, secure, and efficient data operations.

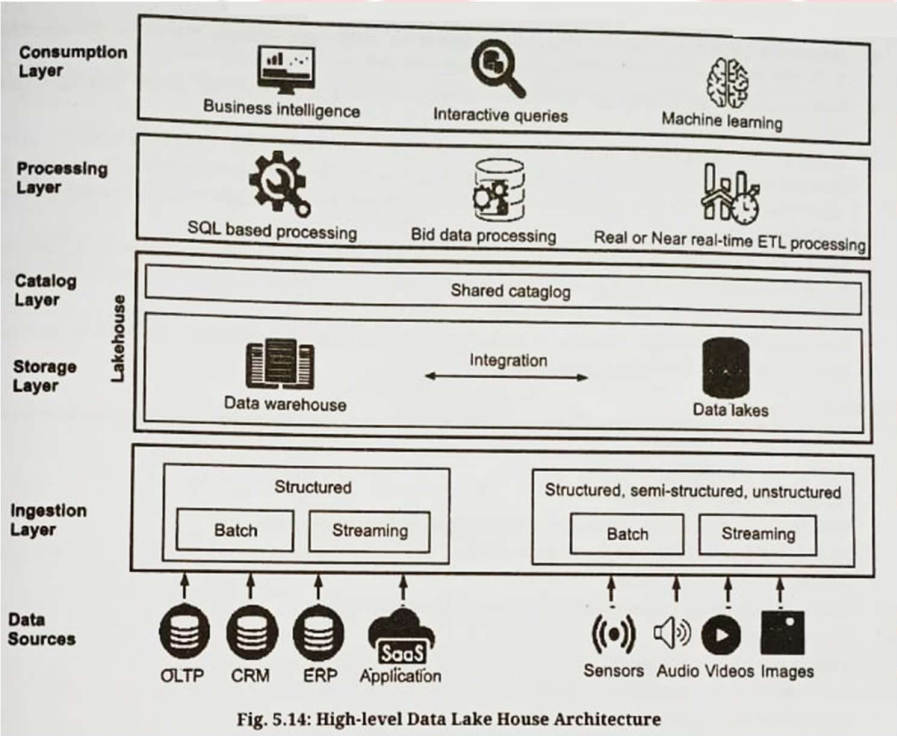


Fig. 5.14: High-level Data Lake House Architecture

1. Ingestion Layer

Components: Data Connectors, Streaming Pipelines, ETL Tools

- This layer ingests data from various sources—databases, files, IoT devices, social media, etc.
- It supports batch and real-time streaming, ensuring timely and consistent data flow.
- ETL (Extract, Transform, Load) processes clean and prepare data before storage.

2. Storage Layer

Components: Object Storage (e.g., S3, ADLS), Table Formats (e.g., Delta Lake, Apache Iceberg)

- Stores raw and processed data in open formats, enabling flexibility and scalability.
- Supports structured, semi-structured, and unstructured data.
- Table formats enable ACID transactions, schema enforcement, and time travel.

3. Catalog Layer / Metadata & Governance Layer

Components: Data Catalogues, Security Policies, Lineage Tracking

- Manages metadata about datasets—schemas, ownership, access controls.
- Ensures data governance, including compliance, auditing, and lineage tracking.
- **Tools** like Hive Metastore or Unity Catalog help organize and secure data assets.

4. Processing Layer / Compute & API Layer

Components: Processing Engines (e.g., Spark, Presto), Query Interfaces (e.g., SQL, REST APIs)

- Executes data transformations, analytics, and machine learning workflows.
- Offers interactive and batch processing capabilities.
- APIs allow developers and analysts to query and manipulate data programmatically.

5. Consumption Layer

Components: BI Tools (e.g., Power BI, Tableau), Dashboards, Applications

- Delivers data insights to end-users through visualizations, reports, and apps.
- Supports real-time dashboards, predictive models, and embedded analytics.
- Enables cross-functional access for business, data science, and engineering teams.

This layered architecture ensures that data is ingested, stored, governed, processed, and consumed efficiently—all within a unified platform.

23. What are different types of client-server architecture? (single-tier, two-tier, three-tier)

Single-Tier Client Server Model:

- Used on a personal computer
- Database is centralized
- Dumb terminals used to access DBMS
- Client terminals are directly connected to larger server system such as mainframes
- All processing takes place on the main frame

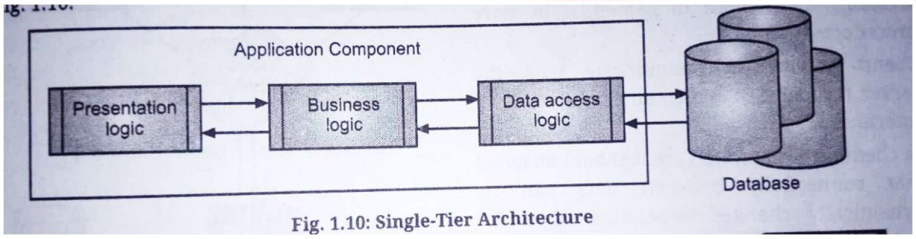


Fig. 1.10: Single-Tier Architecture

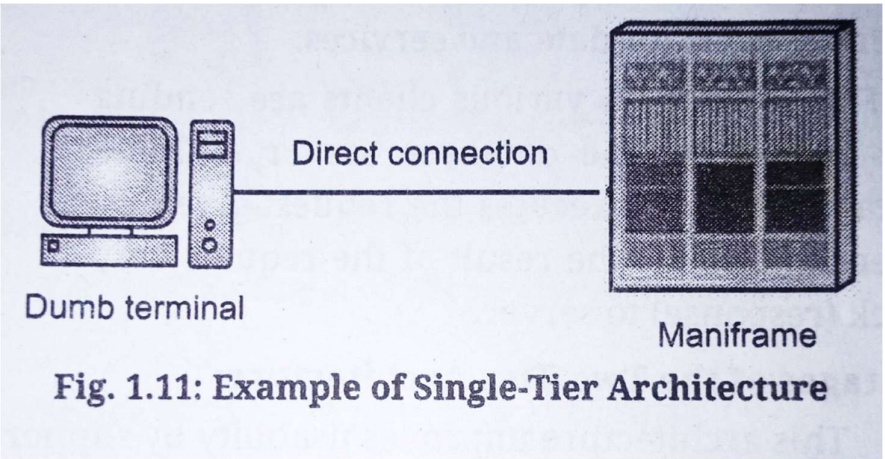


Fig. 1.11: Example of Single-Tier Architecture

Advantages:

- **Simplicity:** Easy to set-up a single machine
- **Cost effective:** No additional hardware
- **Easy implementation:** For small projects

Disadvantages:

- Limited scalability, security concerns, not suitable for multiple users

Two-Tier Client-Server Model:

- Developed in 1980
- Supports form-based, user-friendly interface
- Client side – user interface, business logic→ called as fat client
- Server side - Database system services
-

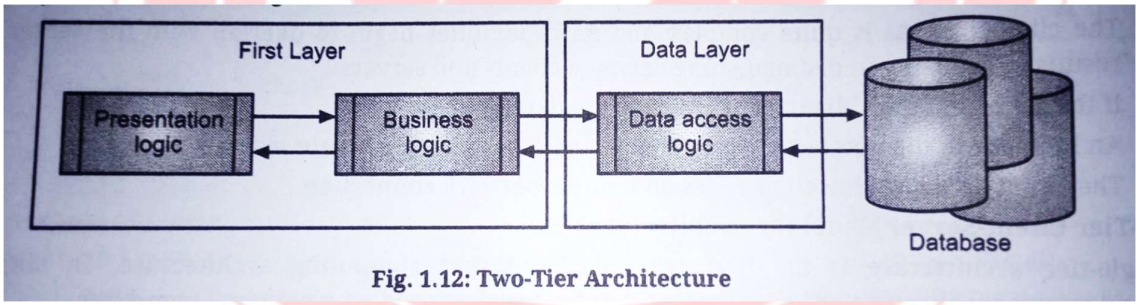


Fig. 1.12: Two-Tier Architecture

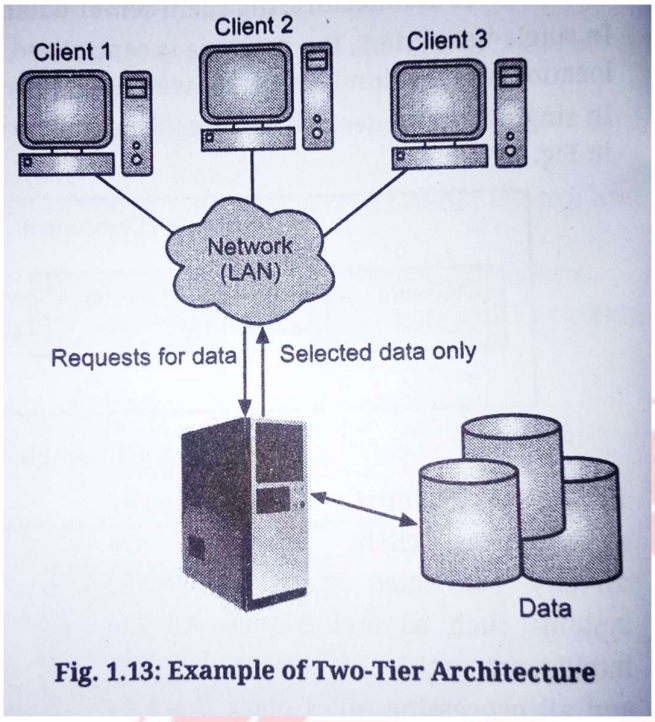


Fig. 1.13: Example of Two-Tier Architecture

- Client communicates through SQL statement

Advantages:

- Improved usability
- User friendly interface
- Less complicated
- Server can act as client for another server

Disadvantages:

- Difficult to administer
- Performance deteriorates when number of users increase

Three- Tier Client Server Model:

- Also called as multi-tier architecture
- Middle tier added between UI client and DBMS server
- Widely used design

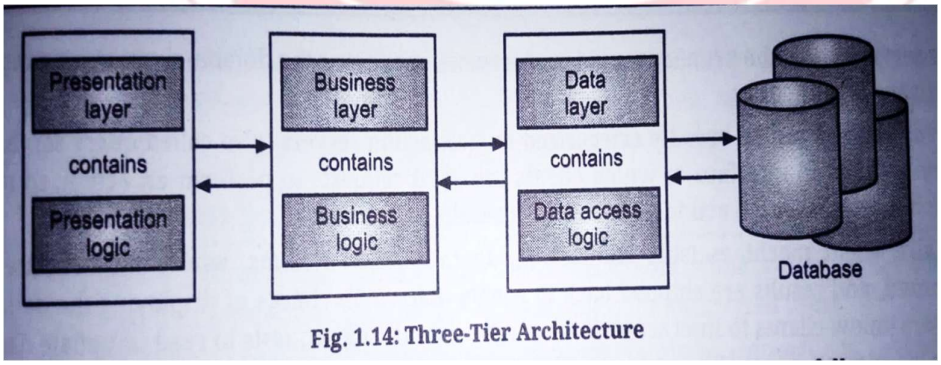


Fig. 1.14: Three-Tier Architecture

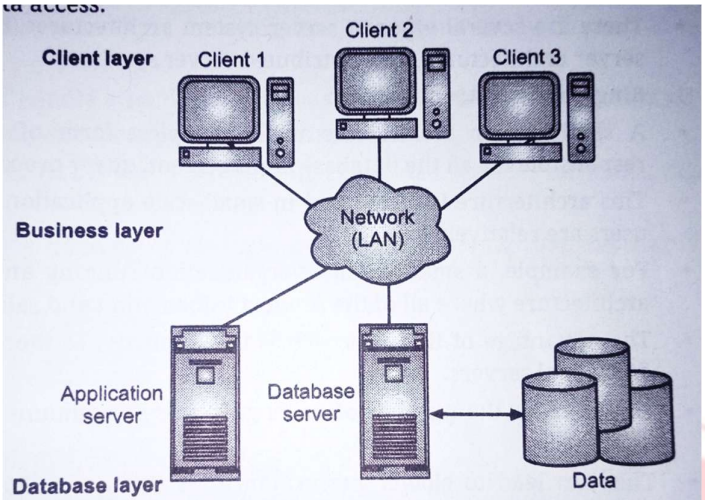


Fig. 1.15: Example of Three-Tier Architecture

- **Three interacting Tiers**
  - **Tier 1(User Interface) :** Contains presentation logic including simple control and user input validation→called as thin client
  - **Tier 2(Business Logic)** Called as application server – provides business processes logic and data access
  - **Tier 3(Database)** It has a data server – provides business data – ensures data consistency

Advantages:

- Improved performance with large number of clients
- More scalable
- Flexible in s/w implementation
- Better load balancing
- Easier to modify without affecting other tiers
- Improved customer services

Disadvantages:

- Complex and expensive
- Tier may affect performance
- Lack of compatible end user tools

24. With the help of diagram describe transaction server.

- Transaction server manages and executes database transactions
- Ensures data integrity and consistency
- Handles execution of instructions like – read, write
- **Functionality:**
- Manages flow of work related to transactions
- Ensures operations are performed in consistent and reliable manner
- E.g.
  - Bank Transfer- ensures security of funds
  - Microsoft Transaction Server (MTS)- provides services for managing transactions in distributed applications

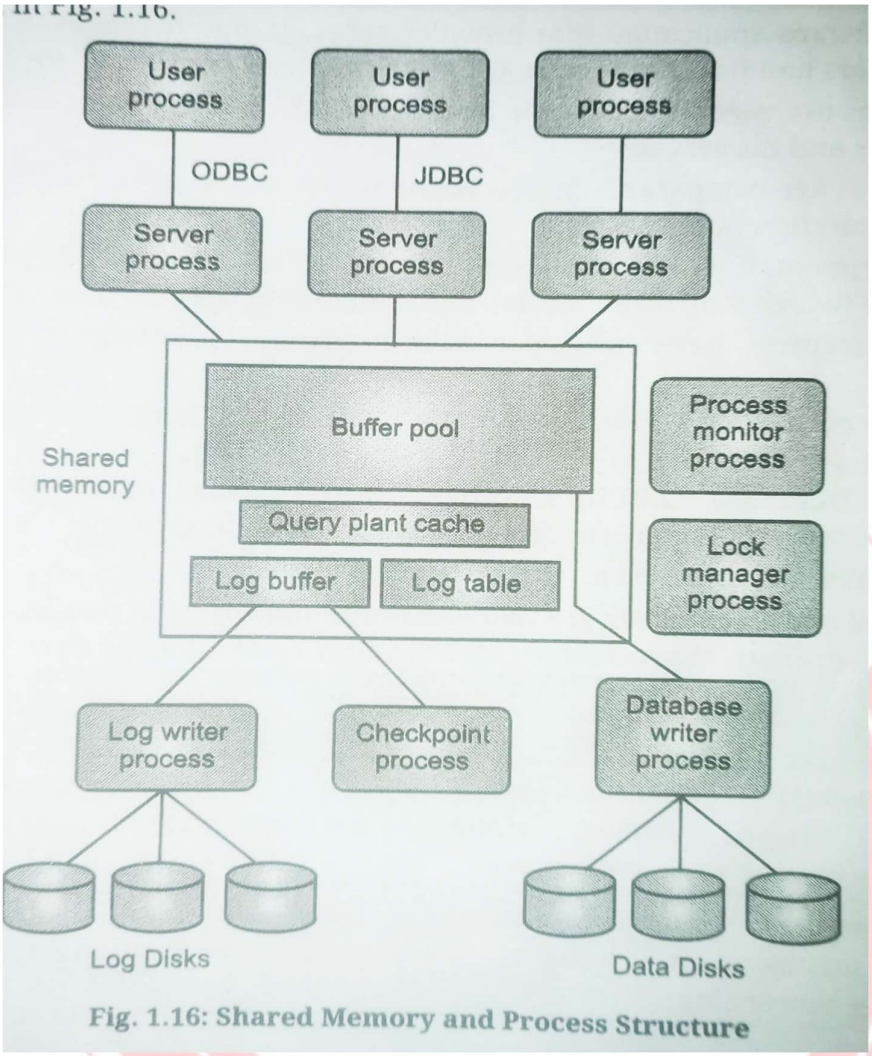


Fig. 1.16: Shared Memory and Process Structure

- Database system processes:
- Server processes- receive user queries, execute them and send result back
- Queries submitted to server processes running embedded SQL/JDBC/ODBC
- Lock manager implements- grant lock, lock release, deadlock detection
- Database writer process- gives modified buffer blocks back to disk on continuous basis
- Log writer process- outputs log records from log record buffers to stable storage
- Checkpoint process performs periodic checkpoints
- Process monitor process- monitors other processes
- Shared memory- contains all shared data- buffer pool, lock table, log buffer
- Log buffer contains log records
- Query plant cache-reused when same query is submitted again

25. What are different types of database models? Explain any one in detail

- There are two types of database system architecture:
- Centralized database system architecture &
- Client server database system architecture:

Centralized database system

- All the data is stored on a single server
- Applications interact with the server to access data

Characteristics of centralized system

- **Simplified management:** easier to manage and secure
- **Data consistency:** data consistency and integrity are maintained
- **Centralized data storage:** data stored at single location
- **Single point control:** simplifies administration and security
- **E.g.**
- **Bank database:** all customer data at central server
- **University database:** managing students records and course information on central server

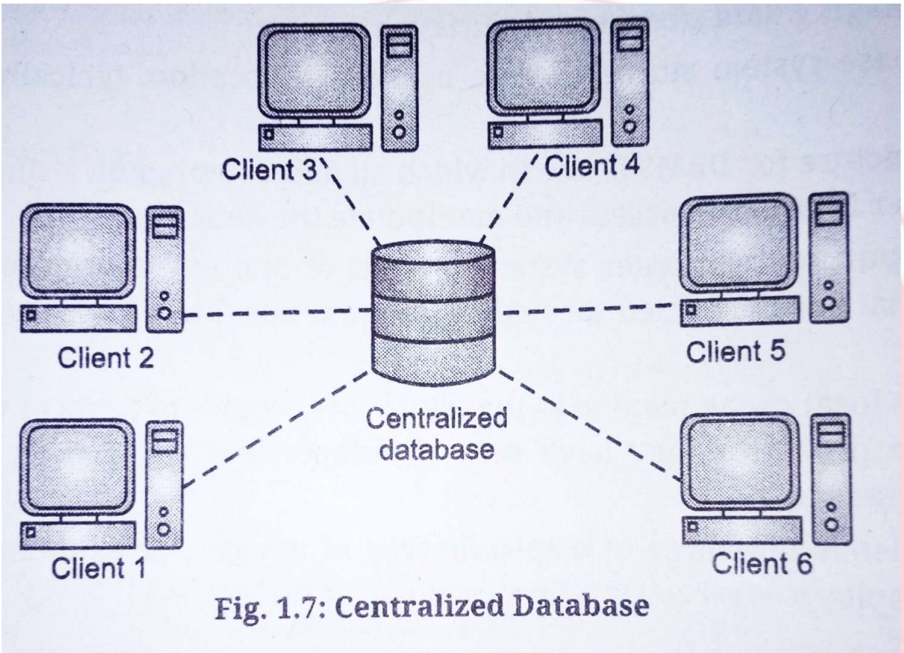


Fig. 1.7: Centralized Database

- **Advantages:**
  - **Consistency:** all users access same unified view of data
  - **Security:** easy to implement security at single point
  - **Control:** it is easier to control tasks like update, backup and maintenance
  - **Simplified management:**
  - **Cost-effective:** less maintenance cost
  - **Data recovery:** easy to recover data from single server

- **Disadvantages:**
  - **Security risks:** single point failure may lead to security breaches
  - **Single point failure:** entire system down
  - **Network performance issues:** network traffic can lead to bottleneck
  - **Scalability challenges:** difficult and expensive

Client-server database system architecture

- **Client:** requester of processes through web interface or chat client or email client
- **Server:** receiver of the request from the client.
- Processes request- gathers required information-generates results-sends them back to client
- **Network:** infrastructure connecting clients and servers

- **Architecture:**
- **3 major components – client, server, network interface**

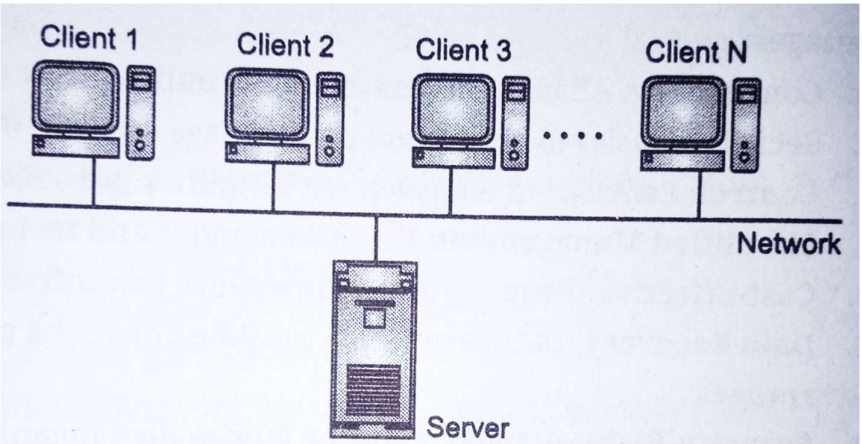


Fig. 1.8: The Client-Server Database Model (Single Tier Client-Server Architecture)

- Database functionality can be divided into backend and front end
- **Back-end:** Manages access structures, query evaluation, optimization, concurrency control & recovery.
- E.g.: SQL, MS Access, SQL Server
- **Front-end:** Consist of tools – forms, report writers, GUI facilities
  Interface between front-end and back-end →SQL or API
  E.g. VB, VC++

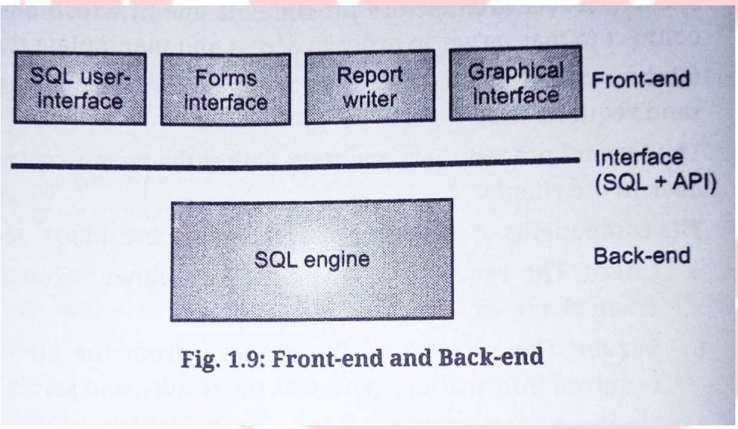


Fig. 1.9: Front-end and Back-end

- Procedure calls or queries are used to communicate between client and server
- **E.g.** Database using client base server are MySQL, ORACLE
- **Advantages:**
  - **Centralized control:** network administrator has central control over DB & its resources
  - **Simple:** to implement
  - **Increased performance:**
  - **Complex application and cost reduction:** complex application can be developed reducing their cost.
  - **High speed:**
  - **Improved data integrity:**
  - **Scalability:** Easy scaling of database and its resources
  - **Security:** Server implements security measures
- **Disadvantages:**

- Network congestion:
- Does not allow single query to span multiple servers
- Complex process
- Overlapping with server
- Confusion between client end servers
- If server fails, client loses access to DB
- Increased numbers of users and sites create security problems
- The architecture relies on stable network connection

Comparison between Shared Lock and Exclusive Lock:

Sr. No.	Shared Lock	Exclusive Locks
1	It exists when concurrent transactions are granted READ access on the basis of a common lock.	It exists when access is especially reserved for the transaction that locked the object.
2	It is issued when a transaction wants to read data from the database and no exclusive lock is held on that data item.	It is issued when a transaction wants to write (update) a data item and no locks are currently held on that data item.
3	It produces no conflict, as long as, the concurrent transactions are read only.	It is used when the potential for conflict exists.

Compare Between Transaction Server and Data Server

Sr. No.	Transaction Server	Data Server
1	Executes transactions (queries) sent by clients and returns results; manages transaction processing and concurrency.	Primarily provides access to raw data; clients are responsible for most data processing.
2	Most processing (query execution, transaction management, locking, etc.) happens on the server side.	The server mainly handles data storage and retrieval; processing happens on the client side.
3	Multiple server processes handle user transactions, manage locks, logs, and concurrency.	Server acts mainly as a data repository, with less focus on transaction management.
4	Widely used in relational DBMS for handling high volumes of queries and transactions.	Used in object-oriented DBMS or applications needing complex data manipulation at the client.
5	Clients send requests (often SQL queries) to the server, which executes them and sends back results.	Clients fetch data from the server and perform transaction logic and data manipulation locally.
6	Transaction-server architecture system includes, IBM's CICS, Microsoft COM+ etc.	Data-server architecture system includes, ObjectDB, ZODB etc.

Differentiate between Parallel Database & Distributed Database

Sr. No.	Characteristics	Parallel Database	Distributed Database
1	Nature	Parallel databases are generally homogeneous in nature.	Distributed databases may be homogeneous or heterogeneous.
2	Definition	It is software where multiple processors are used to execute and run queries in parallel.	It is software that manages multiple logically interrelated databases distributed over a computer network.
3	Geographical location	The nodes are located at geographically same location.	The nodes are usually located at geographically different locations.
4	Execution speed	Quicker/faster.	Slower.
5	Overhead	Less.	More.
6	Performance	Lower reliability and availability.	Higher reliability and availability.
7	Scope of expansion	Difficult to expand.	Easier to expand.
8	Backup	Backup at one site only.	Backup at multiple sites.
9	Consistency	Maintaining consistency is easier.	Maintaining consistency is difficult.
10	No. of Locations	Processors are tightly coupled and constitute a single database system i.e. parallel databases are centralized databases and data reside a single location.	Sites are loosely coupled and share no physical components i.e. distributed databases are geographically separated and data are distributed at several locations or sites.
11	Query optimization	More complex.	Query optimization techniques may be different at different sites and are easy to maintain.
12	Size of Database	Very large.	Relatively small.

Compare between OODBMS & ORDBMS

Basis of Comparison	Object-Oriented Database (OODBMS)	Object-Relational Database (ORDBMS)
Connection Between the Relations	Relationships are represented by references viz the object identifier (OID).	Connections between two relations are represented by foreign key attributes in one relation that reference the primary key of another relation
Data Storage Structure	It employs indexing techniques to locate disk pages that store the object. Therefore, they are able to provide persistent storage for complex-structured objects.	It does not specify any data storage structure each base relation is implemented as separate file and therefore, they are unable to provide persistent storage for complex-structured object.
Quantity of Data	Handles larger and complex data than RDBMS.	Handles comparatively simpler data.
Constrains	The constraints supported by this system vary from system to system.	It has keys, entity integrity and referential integrity.
Data Manipulation Language	The data management language is typically incorporated into a programming language such as C#, C++, etc.	There are data manipulation language such as SQL and QBE which are based on relational calculus.
Description of Stored Data	Stores data entries are described as object.	Stores data in entries is described as tables.
Examples	db4o, ObjectStore, GemStone etc.	PostgreSQL, IBM DB2 etc.
Type of Data	Object oriented database can handle different types of data.	Relational database can handle a simple type of data.
Data Storage	The data is stored in the form of objects.	Data is stored in the form of tables, which contains rows and column
Foundation	Built on object-oriented programming principles.	Extends relational databases (RDBMS) with object-oriented features.

Compares XQuery and XPath:

Sr. No.	XQuery	XPath
1	XQuery is a functional programming and query language that is used to query a group of XML data.	XPath is a XML path language that is used to select nodes from an XML document using queries.
2	XQuery is used to extract and manipulate data from either XML documents or relational databases and MS-Office documents that support an XMLdata source.	XPath is used to compute values like Strings, Numbers and Boolean types from another XML documents.
3	XQuery is represented in the form of a tree model with seven nodes namely, processing instructions, elements, document nodes, attributes, namespaces, text nodes, and comments.	XPath is represented as tree structure, navigate it by selecting different nodes.

4	XQuery supports XPath and extended relational models.	XPath is still a component of query language.
5	XQuery language helps to create syntax for new XML documents.	XPath was created to define a common syntax and behavior model for XPointer and XSLT.

Differentiate between Structured Data & Unstructured Data

Sr. No.	Features	Structured Data	Unstructured Data
1	Representation	Discrete rows and columns.	Less defined boundaries and easily addressable.
2	Storage	Rational databases or spreadsheets, etc.	Unmanaged file structure.
3	Metadata	Syntax.	Semantics.
4	Integration tools	ETL or ELT.	Batch processing or Manual data entry that involves codes.
5	Standard	SQL, ADO.NET, ODBC etc.	OpenXML, JSON, CSV etc.
6	Databases	MySQL, Oracle etc.	Hadoop, MongoDB etc.
7	Content	Text.	Text, Images, Audio, Video.
8	Nature	Not subjective and concerned with past events.	Subjective related to future events.
9	Diagram		

Differentiate between different types of database

Sr. No.	Feature	Column-Oriented	Document-Store	Key-Value Store	Graph-Store
1	Table-like schema support (columns)	Yes	No	No	Yes
2	Complete update/fetch	Yes	Yes	Yes	Yes
3	Partial update/fetch	Yes	Yes	Yes	No
4	Query/filter on value	Yes	Yes	No	Yes
5	Aggregate across rows	Yes	No	No	No
6	Relationship between entities	No	No	No	Yes
7	Cross-entry view support	No	Yes	No	No
8	Batch fetch	Yes	Yes	Yes	Yes
9	Batch update	Yes	Yes	Yes	No

Differences between SQL and NoSQL database

Sr. No.	SQL	NoSQL
1	SQL stores data in the table.	NoSQL stores data in the form of document-store, key-value-store, graph-store etc.
2	SQL provides standard platform to run complex queries.	NoSQL cannot provide any standard platform to run complex queries.
3	It supports the structured and organized data.	It supports unstructured data.
4	Strong consistency.	Dependent on the product. Few chose to provide strong consistency whereas few provide eventual consistency.
5	High level of enterprise support is provided.	Open-source model. Support through third parties or companies building the open-source products.
6	Available through easy-to-use GUI interfaces.	Querying may require programming expertise and knowledge.
7	It uses structured query language to manipulate database.	There is not declarative query language for manipulating data.
8	SQL works on ACID (Atomicity, Consistency, Isolation and Durability) properties.	NoSQL follows CAP (Consistency, Availability and Partition tolerance) theorem.
9	Data and its relationships are stored in separate tables.	NoSQL does not have any pre-defined schema.
10	Examples include Oracle, SQLite, MySQL, PostgreSQL etc.	Examples include MongoDB, Cassandra, HBase, Neo4j, Big Table etc.
11	SQL databases are primarily called as Relational Databases (RDBMS).	NoSQL database are primarily called as non-relational or distributed database.
12	SQL databases are vertically scalable.	NoSQL databases are horizontally scalable.
13	SQL databases use a powerful language "Structured Query Language (SQL)" to define and manipulate the data.	In NoSQL databases, collection of documents is used to query the data. It is also called "Unstructured Query Language (UnQL)". It varies from database to database.
14	SQL databases are best suited for complex queries.	NoSQL databases are not so good for complex queries because these are not as powerful as SQL queries.

Compare Data Mining & Data Warehouse

Sr. No.	Data Mining	Data Warehouse
1	Data mining is the process of analyzing unknown patterns of data.	A data warehouse is used to integrate data from multiple sources and then combine it into a single database.
2	Data mining techniques are applied on data warehouse in order to discover useful patterns.	It provides the organization a mechanism to store huge amount of data.
3	Data mining is the process of analyzing unknown patterns of data.	Data warehousing is the process of pooling all relevant data together.
4	Data mining is a broad set of activities used to uncover patterns, and give meaning to this data.	Data warehouse is the repository to store data.

5	As Facebook stores all the data in central aggregate database, now users can extract meaningful data patterns from it. That is, users can see the ads, get friends suggestions relevant to them, etc. This implies the data mining phase.	For example, Facebook gathers (collects) all user's data such as his/her friends, likes and messages, notifications. Etc., and then stores them into a central repository.
6	This process is always carried out after data warehousing process because it needs compiled data in order to extract useful patterns.	This process must take place before the data mining process because it compiles and organizes the data into a common database.
7	Low maintenance.	Data warehouses has high maintenance.
8	After successful initial queries, users may ask more complicated queries which would increase the workload.	Data Warehouse (DW) is complicated to implement and maintain.
9	The data mining methods are cost-effective and efficient compares to other statistical data applications.	Data warehouse's responsibility is to simplify every type of business data. Most of the work that will be done on user's part is inputting the raw data.
10	Data mining helps to generate actionable strategies built on data insights.	Once we input any information into Data warehouse system, you will unlikely to lose track of this data again. You need to conduct a quick search, helps you to find the right statistic information.
11	Data is analyzed regularly in data mining.	Data is stored periodically in DW.

Comparison between Data Warehouse, Data Lake and Data Lakehouse:

Parameters	Data Warehouse	Data Lake	Lakehouse
Data	Rational data from transactional systems, operational databases and business applications.	All data including structured, semi-structured, and unstructured.	Query every kind of data image, audio, video, and others.
Data Access	SQL only.	Open API, SQL, Python.	Open API, SQL, Python.
Data Format	Proprietary Format.	Open format.	Open Format.
Governance	Fine-grained Security.	Weak Governance and security.	Fine-grained Security.
Reliability	High with ACID Transactions.	Low Quality- Data Swamps.	High with ACID Transactions.
Performance	Fast query results using local language.	Faster query results, decoupling of computing and storage.	Faster and deeper insights without data movement.
Scalability	Scaling becomes expensive.	Low cost of scaling Regardless of the data-type.	Low cost of scaling regardless of the data-type.