UNIT 1 DATABASE SYSTEM ARCHITECTURE

Database: collection of interrelated data

DBMS: set of programs that are used to store, retrieve and manipulate the data in database, in order to produce meaningful information

DBMS: stores, retrieves, updates and manages database

Q) What are benefits of well-designed database architecture?

Database Architecture: set of rules, specifications and processes that dictate how data is stored and accessed

Benefits of well-designed database architecture:

- **Performance:** fast and efficient data access
- **Scalability:** can handle increasing amount of data and users
- **Data Integrity:** accuracy and consistency
- **Security:** protecting sensitive data from unauthorized access
- **Maintainability:** easy to update and modify DB

Q) What is concurrency control? What is need of concurrency control?

Concurrency Control: "it is the process of managing simultaneous execution of transactions like queries, updates, inserts, deletes in multiprocessing database system without interfering each other's operations**"**

Concurrency control is a technique used in database systems to manage the execution of multiple transactions on the same data at the same time.

Need/purpose of concurrency control:

- **Preserve database consistency:** during concurrent execution

- **Minimize transaction size:** smaller transactions interact less likely with other transactions
- **Run resource-intensive operations:** schedule long running operations after peak hours
- **Limit transaction operations:** transaction should accomplish one task- and include related statements
- **Data isolation:** isolation among conflicting transactions
- **Access resources in consistent order:** avoid blocking of resources by transactions
- **Control over database:** to co-ordinate concurrent accesses
- **Minimize resource access time:** keep resources open for access for minimum limited time
- **Resolve conflicts:**
- **Avoid dirty data read:** if dB is updated but not committed- users read old data
- **Transaction reading dirty data is called 'dirty read'**

Q) What are different concurrency control protocols?

- A concurrency control protocol ensures serializability of concurrent transactions
- **Lock based protocol**
- **Timestamp based protocols**
- **Granting of lock**
- **Two phase locking protocol**
- Locking protocols are used commercially

Q) Explain Lock Based Protocol

"Set of rules followed by all transactions while requesting and releasing locks"

- **Lock:**
- Lock is a variable associated with each data item
- Lock indicates weather an operation can be applied on data item
- Locks are granted and released by lock manager
- Data structure for lock manager is lock table
- Manipulating value of lock is called locking
- **2 types of locks**
    - **Shared lock/read lock:** if transaction T_i obtained shared mode lock on A → T_i can read but cannot write A
    - **Exclusive lock/write lock:** if transaction T_i obtained exclusive mode lock on A → T_i can both read and write A
- **Lock granularity:** size of data item that is locked
- It's a lock table unit
- Locking takes place at levels like- DB level, table level, page level, row level, attribute levels
- **Compatibility function and matrix**
- Lock compatibility determines weather lock can be acquired by multiple transactions at the same time
- Represented by compatibility matrix

Mode(B↓,A→)	S(Shared Mode)	X(Exclusive Mode)
S	True	False
X	False	True

- **Conflict serializability:** locking protocol ensures conflict serializability if for all legal schedules the associated relation is acyclic

- **Starvation of locks:** when a transaction waits indefinitely for lock it is called as transaction starvation
- This happens for large transactions locking many data items

Q) Differentiate between shared lock and exclusive lock

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.

Lock Granularity:

Q) Explain granting of locks

- **"**When a transaction requests a lock on data item in a particular mode and no other transaction has a lock on the same data item in conflicting mode – the lock is granted**"**
- **E.g.**
- Transaction T_2 has shared mode lock on data Q
- T_1 requests exclusive-mode lock on Q
- T_1 has to wait till T_2 releases the lock
- Meanwhile T_3 requests shared-mode lock on Q
- Lock request is compatible
- So shared-lock may be granted to both T_2 and T_3
- If T_2 releases the lock T_1 has to wait for T_3
- If only shared lock is being asked on Q, T_1 may never get a lock and get starved
- **How to avoid STARVATION:**

- Lock is granted to a transaction if
  - No other transaction holding lock on the same data item that conflicts with request
  - And no other transaction is **waiting** for lock for same data item

Q) What are 2 phase locking protocols?

- It is a lock-based concurrency control technique
- Ensures serializability
- **"**2PL is a protocol of controlling concurrent processing in which all locking operations precede the first unlocking operation"
- **Each transaction issues lock and unlock request in 2 phases**

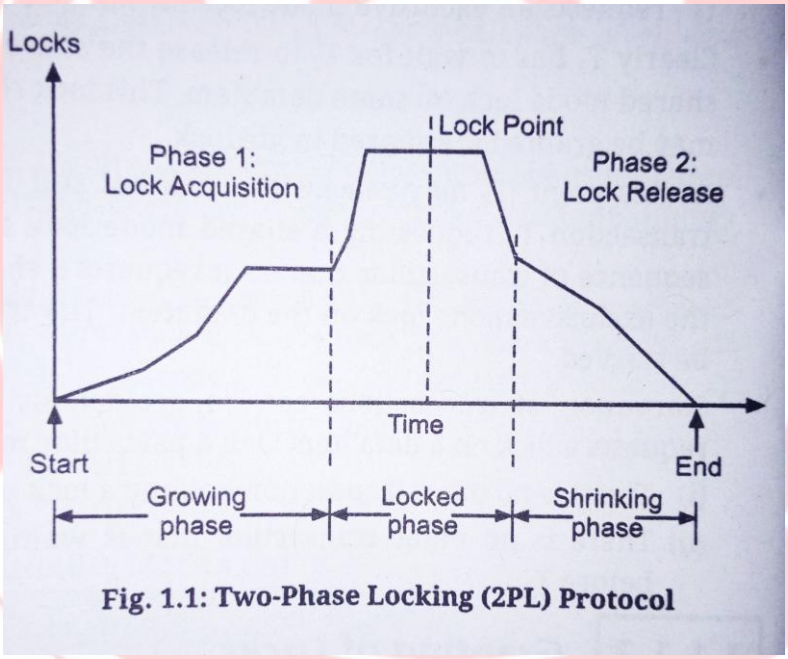

Fig. 1.1: Two-Phase Locking (2PL) Protocol

- **Growing phase:**
- Transaction acquires all required locks without unlocking any data
- Once all locks are acquired – transaction is in locked phase
- **Shrinking phase:**
- Transaction releases all locks and cannot obtain any new lock

- **Lock point:** point in the schedule where transaction has obtained its final lock is called lock point of the transaction
- Transactions are ordered according to lock points
- This gives serializability ordering for transaction

- **Example:** Following two transactions are two phase transactions.

T_1	T_2
lock - X (B);	lock-S (A);
read (B);	read (A);
B: = B – 50;	unlock (A);
write (B);	lock-S (B);
Unlock (B);	read (B);
lock – X(A);	Unlock (B);
read (A);	display (A + B);
A: = A + 50;	
write (A);	
Unlock (A).	

Fig. 1.2: Transaction T_1

Fig. 1.3: Transaction T_2

- Unlock transaction do not need to appear at the end of transaction

Q) Enlist different variations on Two-Phase locking protocols?

- Strict 2 phase locking protocol:
- Rigorous 2 phase locking protocol:
- 2 phase locking with lock conversion:

Q) What are different types of database models?

- Centralized database system architecture:
- Client server database system architecture:

Q) Explain centralized database system with its advantages and disadvantages.

- 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

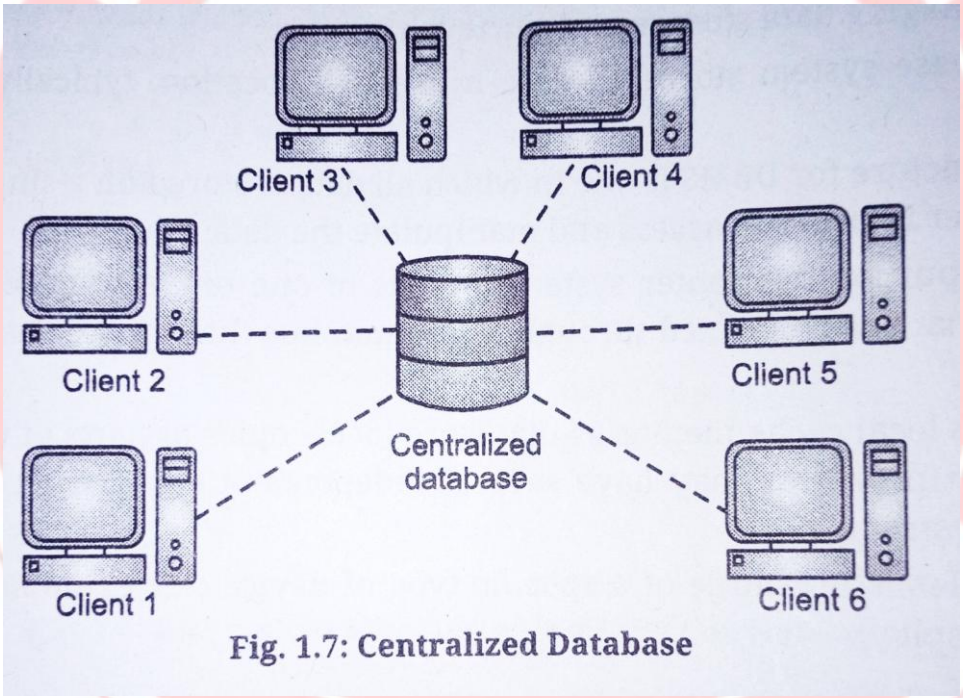

Fig. 1.7: Centralized Database

- **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

Q) Explain 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**

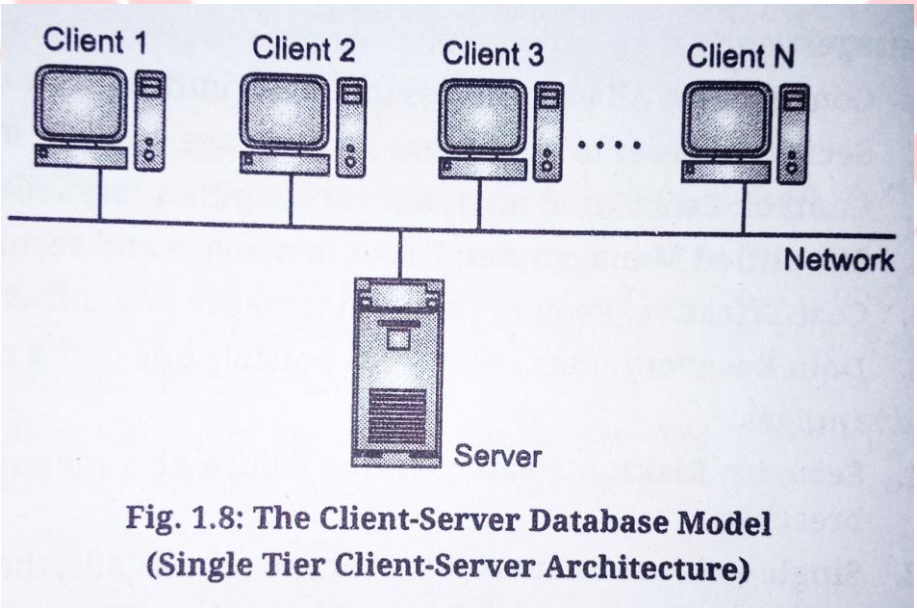

Fig. 1.8: The Client-Server Database Model
(Single Tier Client-Server Architecture)

- 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++

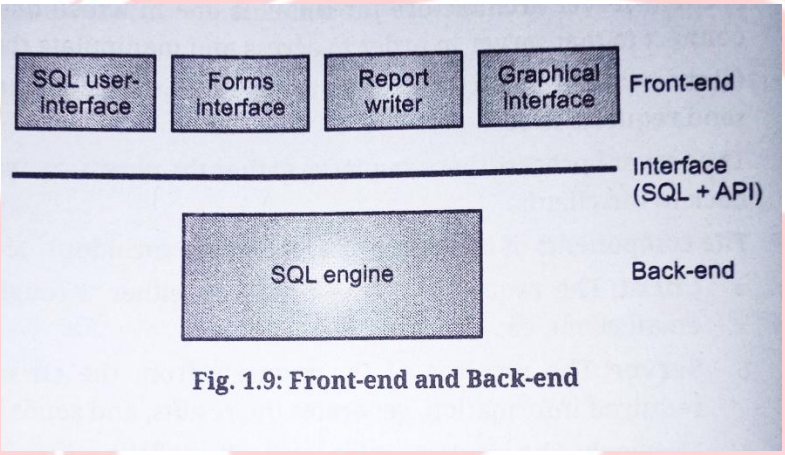

Fig. 1.9: Front-end and Back-end

- 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

Q) 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

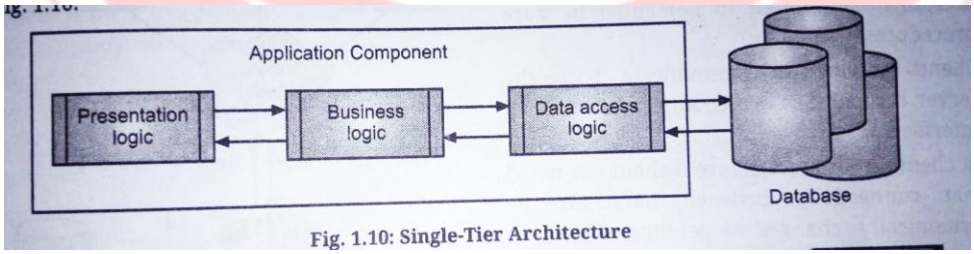

Fig. 1.10: Single-Tier Architecture

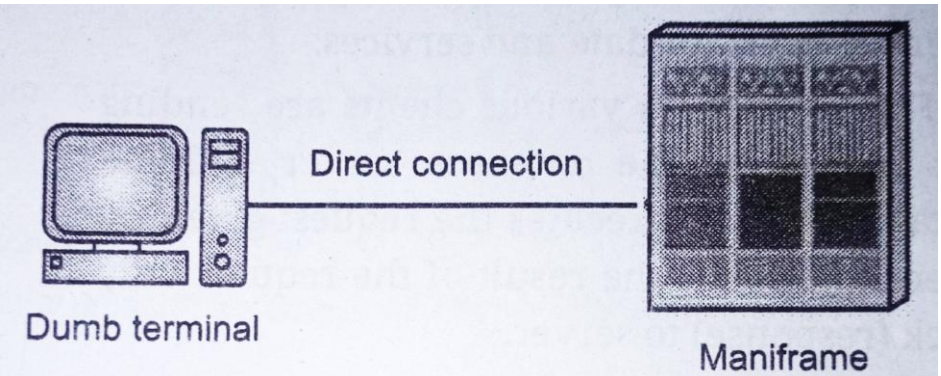

Fig. 1.11: Example of Single-Tier Architecture

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
-

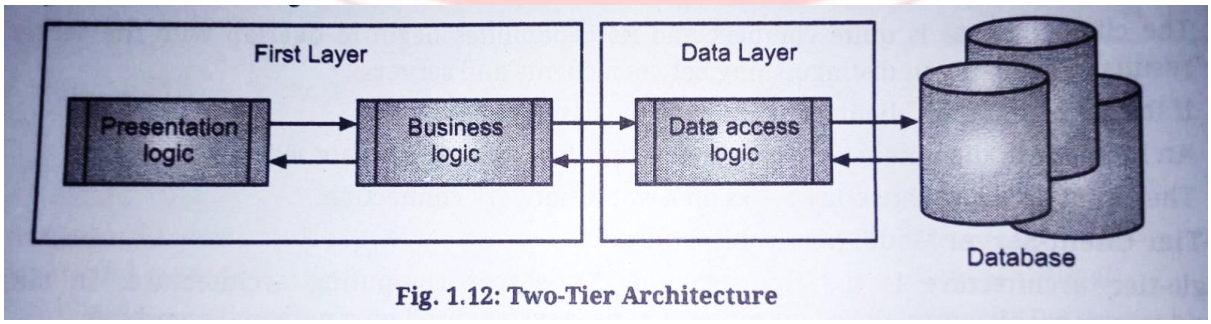

Fig. 1.12: Two-Tier Architecture

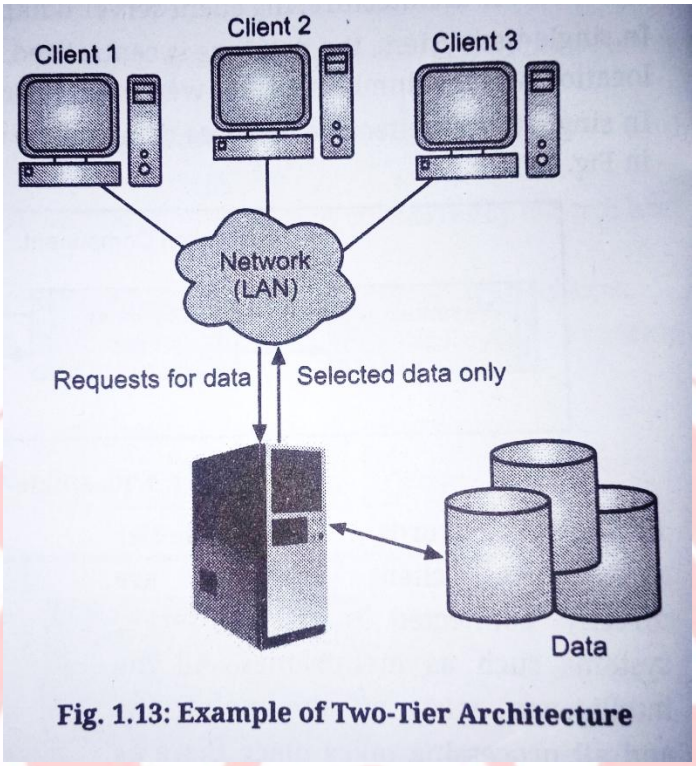

Fig. 1.13: Example of Two-Tier Architecture

- 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

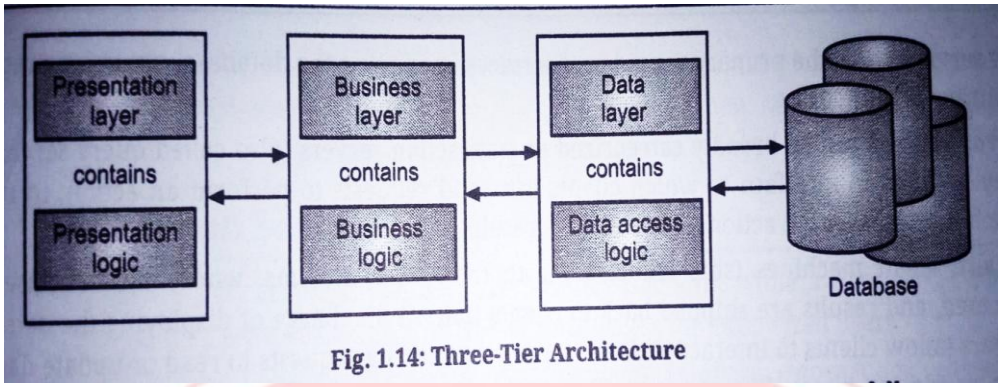

Fig. 1.14: Three-Tier Architecture

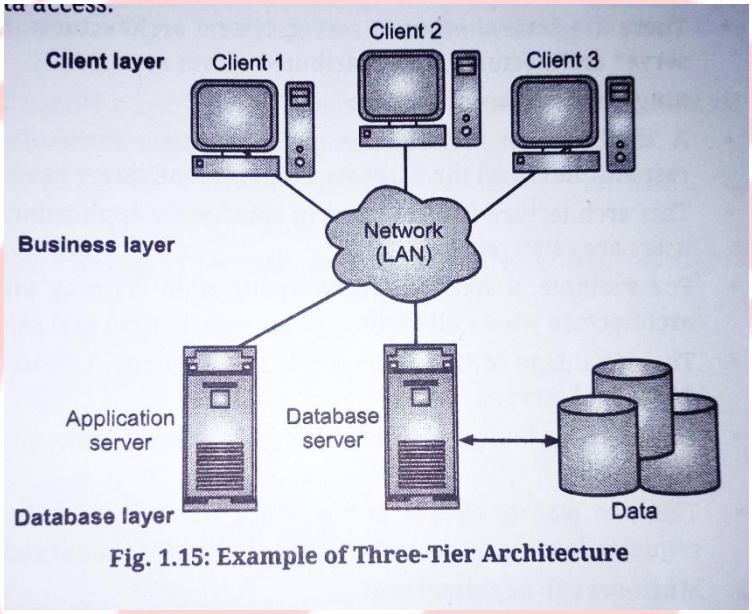

Fig. 1.15: Example of Three-Tier Architecture

- **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

Q) Explain server system architecture with its types. (single server, multi-server, distributed server)

- System centered around central server that performs core processing tasks for clients or users
- Server- primary provider of services-managing DB, processing requests, ensuring data integrity
- **Servers: -** transaction servers/query system server- responds to client's query
- **Client machine: -** ships transaction to the server
- **3 types of server system architecture:**
  1. **single-server architecture:**
     - simplest form
     - one server responsible for- DB management, Query processing, transaction handling
     - used in small scale applications with low volume of data and users
     - e.g. small business or organization
     - **advantages:**
     - it is a simple system

- no need for complex configuration
- no need of additional servers
- **disadvantages:**
- if amount of data increases server becomes bottleneck
- slower processing leads to server down issue

2. **multi-server architecture:**

- multiple servers- to distribute load
- servers handle tasks- DB storage, query processing, transaction management
- one server fails- another server takes over
- e.g. e-commerce platform→one server- manages products catalog, other handles orders, third handles payment
- system works efficiently even if number of users and data increases
- reliable- one server failure does not impact entire system

3. **distributed server architecture:**

- servers are distributed across different physical locations
- **Used in large-**scale applications
- Provides high availability, fault tolerance and load balancing
- e.g. e-commerce accesses from any country
- allows data replication
- data copies are stored at different servers

Q) 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

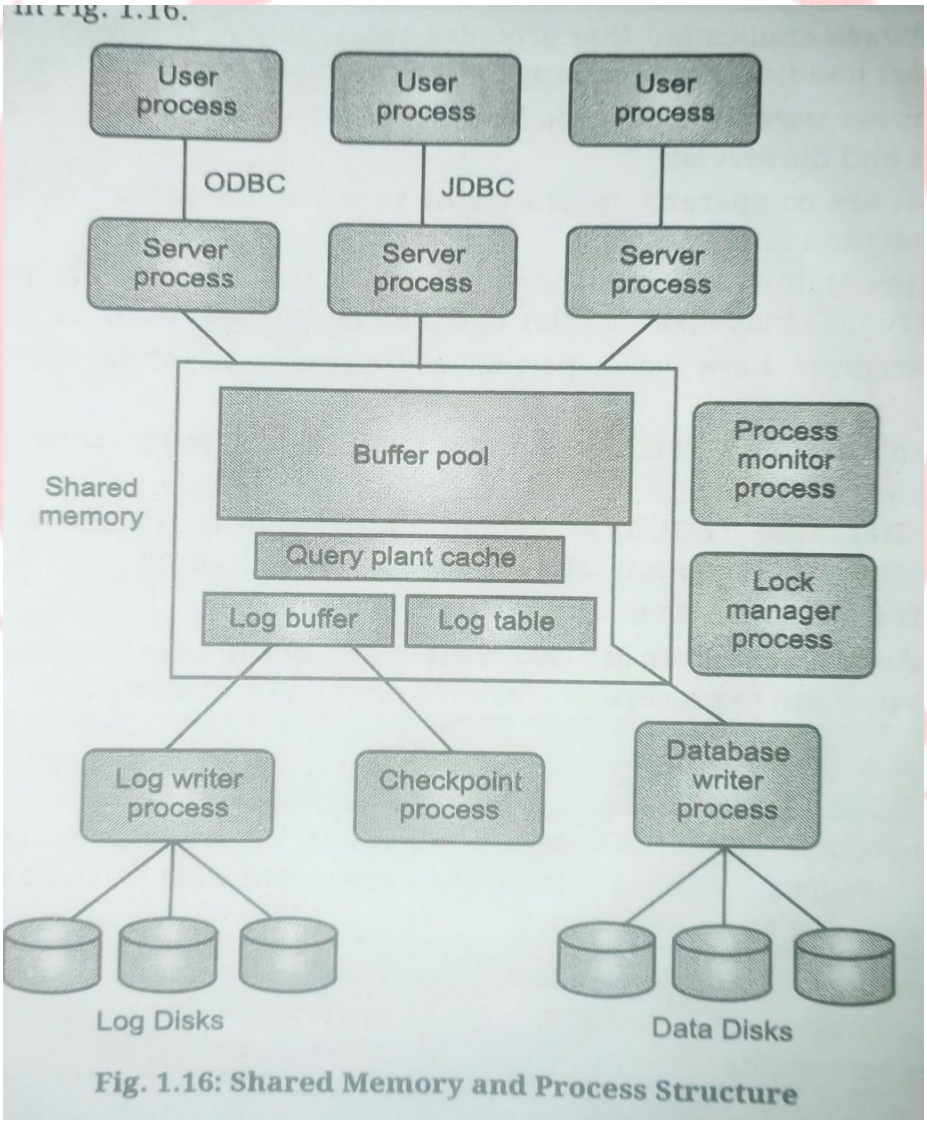

Fig. 1.16: Shared Memory and Process Structure

- 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

Q) What is Data Server?

- Focuses on storing and managing database – provides access to authorized users
- "A computer or software application that provides access to and manages database or data storage system"
- Data servers- handles client requests to – retrieve store and manipulate data
- Used in LAN (high speed connection between client and server)
- Client performs processing of data
- Data server → popular in object-oriented DB system
    - ObjectDB: NoSQL for Java Programming (with ACID compliance)

- Zope Object Database (ZODB): Python oriented DB(with transparency and persistency in web applications)

Q) Differentiate between transaction server and data server.

each with its own approach to processing client

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.

Contd...

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.