DATABASE MANAGEMENT SYSTEM

Chapter 8
Database Technology

Bikash Khadka Shah

MCSE, MSDA, OCP, RHCE, RHCVA, CCNA, CEH

9841766620 | 9801076620

Database Technology

- Database Technology refers to the collection of techniques, methodologies, and tools used to store, manage, and manipulate data in an organized manner.
- The core idea behind database technology is to create an internal representation (model) of the external world of interest, enabling systems to store data in a structured way that accurately reflects real-world entities and relationships.

Key Features of Database Technology:

- Internal Representation:
  - Databases model the external world by representing real-world entities and their relationships through structured data formats, such as tables in relational databases, documents in NoSQL databases, or nodes and edges in graph databases. For example, in an airline reservation system, the database might internally represent flights, aircraft, passengers, and reservations using tables or documents that correspond to these entities.
- Consistency with External Reality:
  - One of the primary goals of database technology is to ensure that the internal model remains consistent with the actual state of the external world. This involves keeping the data accurate, up-to-date, and reflective of real-world changes. This consistency is maintained through various mechanisms, such as data integrity constraints, transactions, and real-time updates.

CORE TECHNOLOGY

- Information management / processing
- Data analysis / statistics
- Data visualization / presentation
- Multimedia and hypermedia
- Office and document systems
- Business processes, workflow, CSCW (computer-supported cooperative work)

- Modern database systems indeed rely heavily on a sophisticated infrastructure to function effectively. This infrastructure is essential for the deployment, management, and utilization of databases in various sectors.

1. Networks (LAN and WAN)

- LAN (Local Area Network):
  - A LAN connects computers and devices within a limited geographical area, such as a single building or campus. It facilitates high-speed data transfer between the database server and clients, ensuring quick access to database resources.
  - LANs are crucial for internal operations, where rapid, reliable access to databases is needed for daily tasks such as data entry, querying, and processing within an organization.
- WAN (Wide Area Network):
  - A WAN connects devices over a broader geographic area, such as across cities, regions, or even countries. It allows remote access to database systems, enabling distributed teams and clients to interact with centralized databases.
  - WANs are essential for enabling access to databases from multiple locations, supporting global operations, and facilitating cloud-based database services.

2. Client-Server Computing Architecture

- Client-Server Model:
  - In this architecture, the server hosts the database and provides services, while the clients are the end-user devices (such as computers, tablets, or smartphones) that request and use these services.
  - This model allows multiple clients to interact with a central database server, supporting scalability and efficient resource management. It also enables centralized control over data, security, and backups.
- Multi-Tier Architectures:
  - Often, database systems are part of a multi-tier architecture where the client interacts with an application server, which in turn interacts with the database server.
  - This separation of concerns enhances security, allows for load balancing, and makes it easier to maintain and update the system.

3. Skilled Data Analysis and Database Design

- Data Analysis:
  - **Skilled data analysts assess the needs of the organization, identify data sources, and define the data structure that best supports business objectives.**
  - **Proper data analysis ensures that the database is designed to capture, store, and retrieve data efficiently and effectively, supporting decision-making processes.**
- Database Design:
  - **Database design involves creating a detailed data model and defining the structure of the database, including tables, relationships, indexes, and constraints.**
  - **A well-designed database optimizes performance, ensures data integrity, and supports scalability. It's the foundation for reliable and efficient data management.**

4. Skilled Systems Development Methods

- Systems Development Life Cycle (SDLC):
  - The SDLC is a structured approach to system development that includes stages such as planning, analysis, design, implementation, testing, deployment, and maintenance.
  - Using a systematic approach ensures that the database system is built to meet user requirements, is tested thoroughly, and is maintained properly over its lifecycle.
- Agile Development:
  - Agile methods emphasize iterative development, where the database system is developed in small increments with continuous feedback from users.
  - Agile allows for flexibility, rapid response to changes, and closer alignment with user needs, which is particularly important in dynamic sectors.

Client/Server Systems

- Networked computing model
- Processes distributed between clients and servers
- Client – Workstation (usually a PC) that requests and uses a service
- Server – Computer (PC/mini/mainframe) that provides a service
- For DBMS, server is a database server

Application Logic in C/S Systems

Presentation Logic
- Input – keyboard/mouse
- Output – monitor/printer

GUI Interface

Processing Logic
- I/O processing
- Business rules
- Data management

Procedures, functions, programs

Storage Logic
- Data storage/retrieval

DBMS activities

Client/Server Architectures

- File Server Architecture
- Database Server Architecture
- Three-tier Architecture

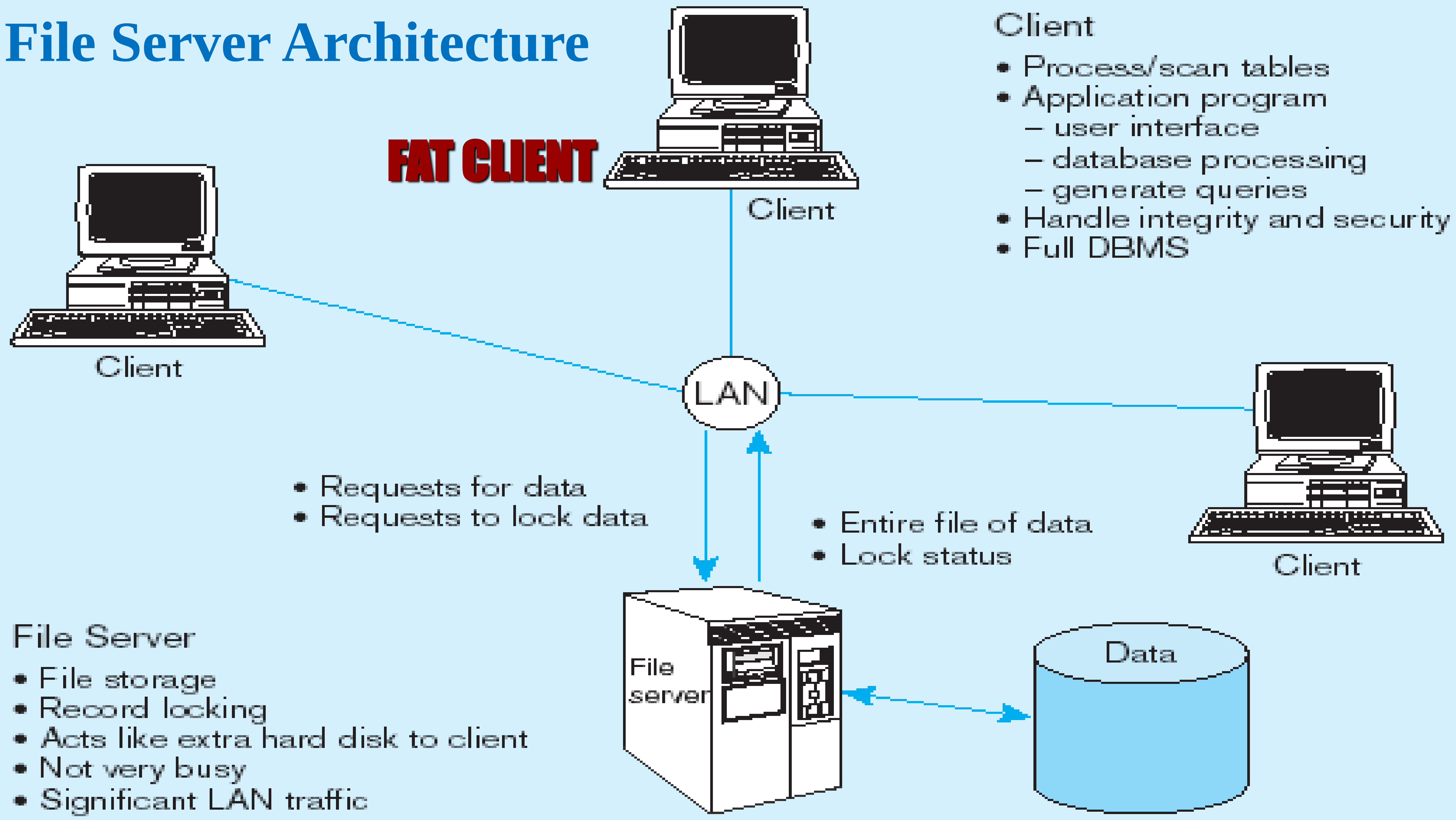
Client does extensive processing

Client does little processing

File Server Architecture: Fat Client

- All processing is done at the PC that requested the data
- Entire files are transferred from the server to the client for processing
- Problems:
    - Huge amount of data transfer on the network
    - Each client must contain full DBMS
        - Heavy resource demand on clients
        - Client DBMSs must recognize shared locks, integrity checks, etc.

File Server Architecture



Client
- Process/scan tables
- Application program
  - user interface
  - database processing
  - generate queries
- Handle integrity and security
- Full DBMS

- Requests for data
- Requests to lock data

- Entire file of data
- Lock status

File Server
- File storage
- Record locking
- Acts like extra hard disk to client
- Not very busy
- Significant LAN traffic

Two-Tier Database Server Architectures

- Client is responsible for
  - I/O processing logic
  - Some business rules logic
- Server performs all data storage and access processing

→ DBMS is only on server

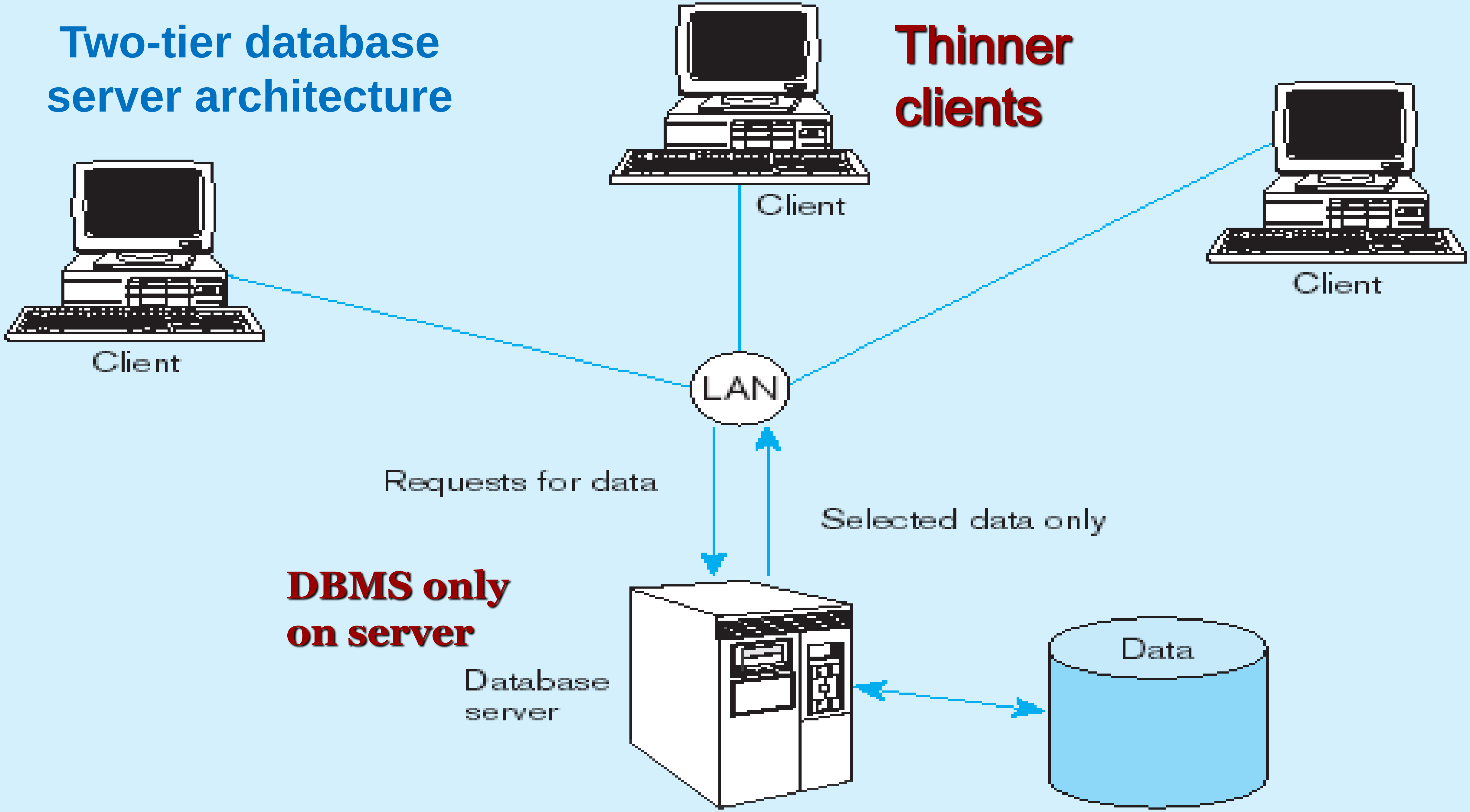
Advantages of Two-Tier Approach

- Clients do not have to be as powerful
- Greatly reduces data traffic on the network
- Improved data integrity since it is all processed centrally
- Stored procedures → DBMS code that performs some business rules done on server

Advantages of Stored Procedures

- Compiled SQL statements
- Reduced network traffic
- Improved security
- Improved data integrity
- Thinner clients

Two-tier database server architecture



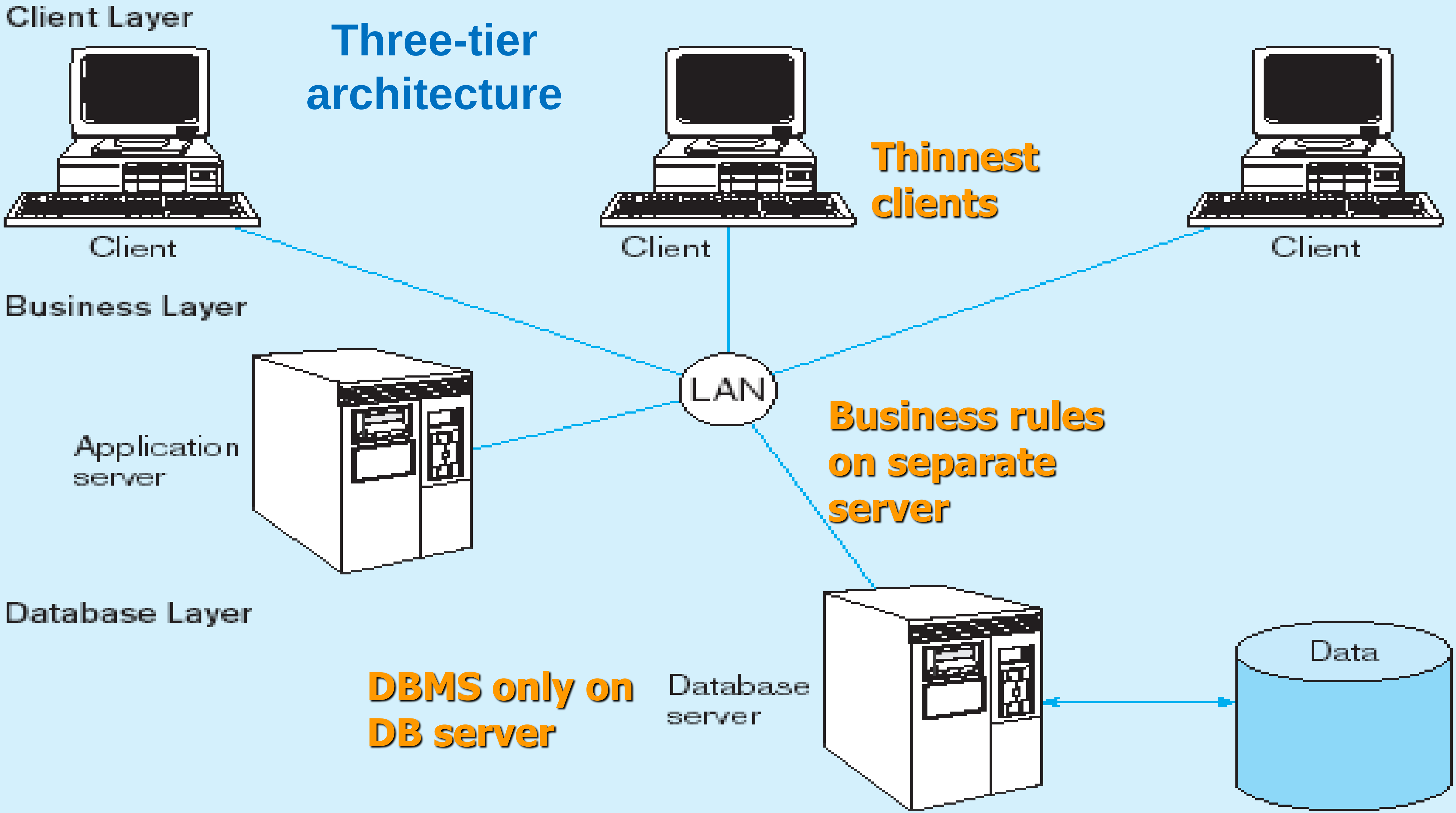
Three-Tier Architectures

Client	**GUI interface (I/O processing)**	***Browser***
Application server	**Business rules**	***Web Server***
Database server	**Data storage**	***DBMS***

Thin Client

- PC just for user interface and a little application processing. Limited or no data storage (sometimes no hard drive)

Three-tier architecture



Thinnest clients

Business rules on separate server

DBMS only on DB server

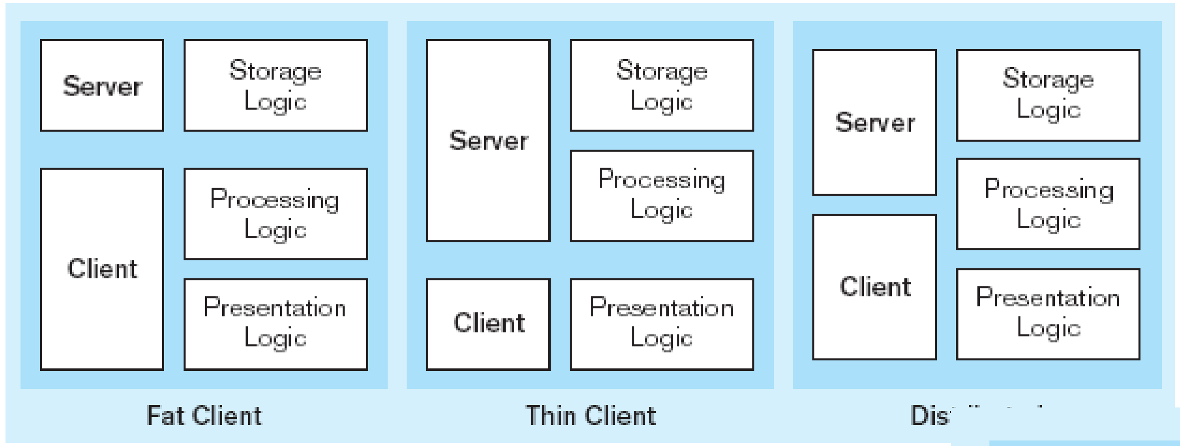
Advantages of Three-Tier Architectures

- Scalability
- Technological flexibility
- Long-term cost reduction
- Better match of systems to business needs
- Improved customer service
- Competitive advantage
- Reduced risk

Application Partitioning

- Placing portions of the application code in different locations (client vs. server)
- Advantages
  - Improved performance
  - Improved interoperability
  - Balanced workloads

Common Logic Distributions

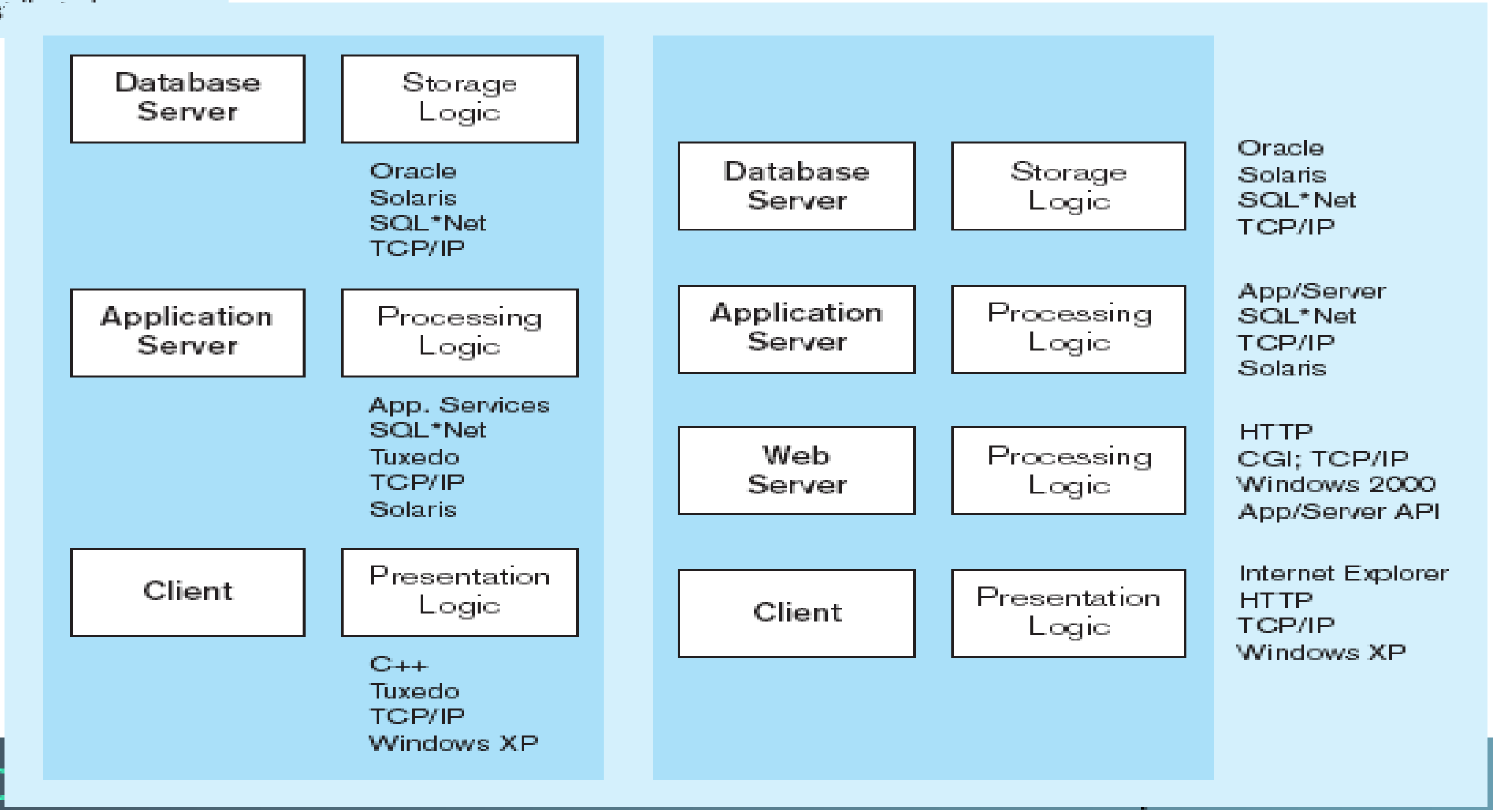


Two-tier client-server environment

Processing logic could be at client, server, or both

n-tier client-server environment

Processing logic will be at application server or Web server



Role of the Mainframe

- Mission-critical legacy systems have tended to remain on mainframes
- Distributed client/server systems tend to be used for smaller, workgroup systems
- Difficulties in moving mission critical systems from mainframe to distributed
- Rule of thumb
    - Mainframe for centralized data that does not need to be moved
    - Client for data requiring frequent user access, complex graphics, and user interface

Middleware

- Software that allows an application to **interoperate** with other software
- No need for programmer/user to understand internal processing
- Accomplished via **Application Program Interface** (API)

The *"glue"* that holds client/server applications together

Types of Middleware

- Remote Procedure Calls (RPC)
  - client makes calls to procedures running on remote computers
  - synchronous and asynchronous
- Message-Oriented Middleware (MOM)
  - asynchronous calls between the client via message queues
- Publish/Subscribe
  - push technology → server sends information to client when available
- Object Request Broker (ORB)
  - object-oriented management of communications between clients and servers
- SQL-oriented Data Access
  - middleware between applications and database servers

Database Middleware

- ODBC – Open Database Connectivity
  - Most DB vendors support this
- OLE-DB
  - Microsoft enhancement of ODBC
- JDBC – Java Database Connectivity
  - Special Java classes that allow Java applications/applets to connect to databases

Client/Server Security

- Network environment → complex security issues
- Security levels:
  - System-level password security
    - for allowing access to the system
  - Database-level password security
    - for determining access privileges to tables; read/update/insert/delete privileges
  - Secure client/server communication
    - via encryption

Keys to Successful Client-Server Implementation

- Accurate business problem analysis
- Detailed architecture analysis
- Architecture analysis before choosing tools
- Appropriate scalability
- Appropriate placement of services
- Network analysis
- Awareness of hidden costs
- Establish client/server security

Benefits of Moving to Client/Server Architecture

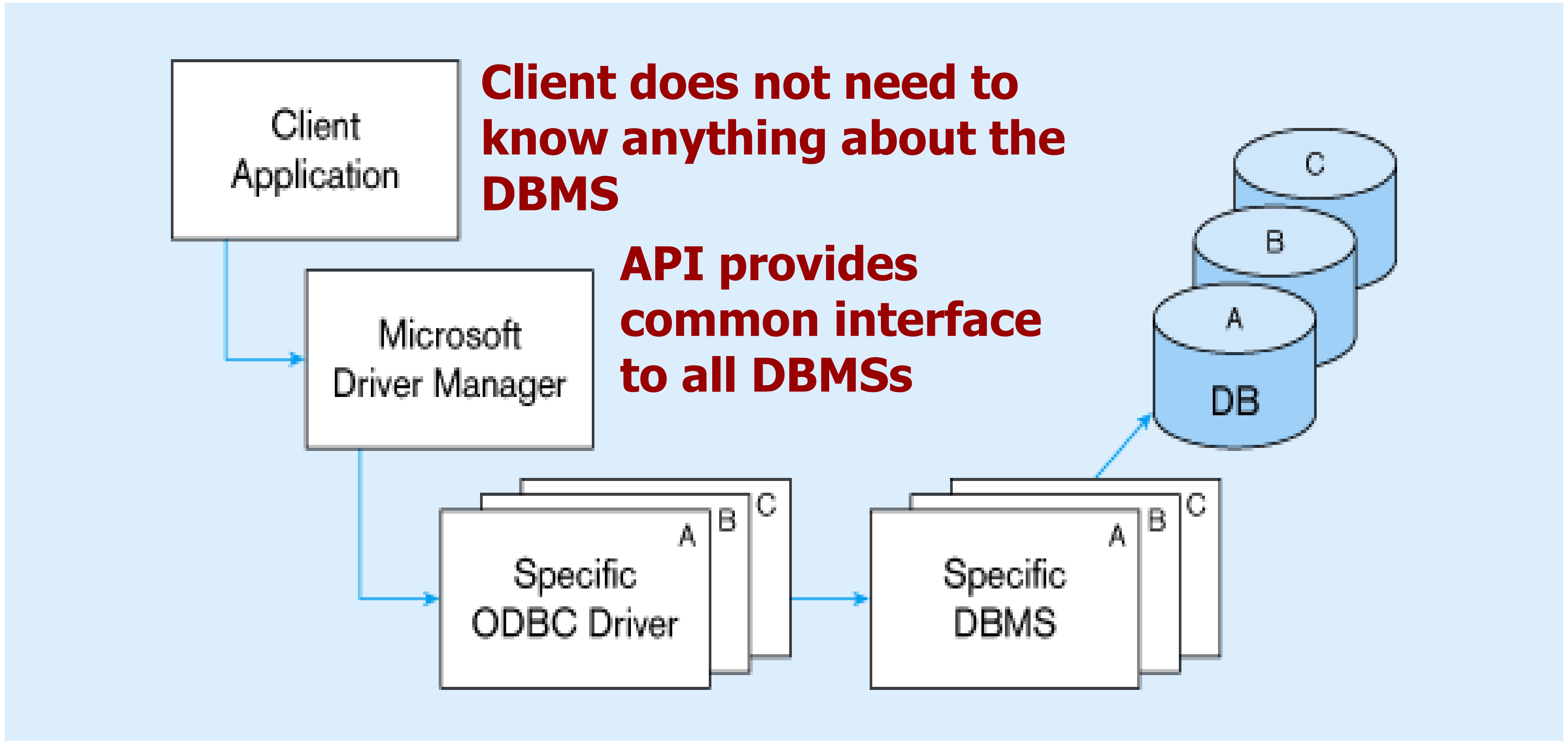
- Staged delivery of functionality speeds deployment
- GUI interfaces ease application use
- Flexibility and scalability facilitates business process reengineering
- Reduced network traffic due to increased processing at data source
- Facilitation of Web-enabled applications

Using ODBC to Link External Databases Stored on a Database Server

- Open Database Connectivity (ODBC)
  - API provides a common language for application programs to access and process SQL databases independent of the particular RDBMS that is accessed
- Required parameters:
  - ODBC driver
  - Back-end server name
  - Database name
  - User id and password
- Additional information:
  - Data source name (DSN)
  - Windows client computer name
  - Client application program's executable name

Java Database Connectivity (JDBC) is similar to ODBC–built specifically for Java applications

ODBC Architecture



Client does not need to know anything about the DBMS

API provides common interface to all DBMSs

Each DBMS has its own ODBC-compliant driver

OLTP(Online Transaction Processing)

- OLTP refers to real-time data processing systems that handle numerous small, concurrent transactions (e.g., banking, shopping, order entry).
- Initially focused on real-world exchanges like money, products, or services, OLTP now also includes digital interactions such as downloading files or triggering actions through web-connected devices.
- OLTP systems typically involve inserting, updating, or deleting small amounts of data in a database.
- These systems are critical for business operations as they track transactions, manage interactions, and support reporting and decision-making.
- It highlights how OLTP has evolved in response to digital transformation, broadening the scope of what constitutes a transaction.

OLAP (Online Analytical Processing)

- OLTP enables the real-time execution of large numbers of transactions by large numbers of people, whereas online analytical processing (OLAP) usually involves querying these transactions (also referred to as records) in a database for analytical purposes.
- OLAP helps companies extract insights from their transaction data so they can use it for making more informed decisions.
- OLAP involves querying these transactions (or records) in a database for analytical purposes, allowing companies to extract insights and make data-driven decisions.

OLTP versus OLAP

OLTP systems	OLAP systems
Enable the real-time execution of large numbers of database transactions by large numbers of people	Usually involve querying many records (even all records) in a database for analytical purposes
Require lightning-fast response times	Require response times that are orders of magnitude slower than those required by OLTP
Modify small amounts of data frequently and usually involve a balance of reads and writes	Do not modify data at all; workloads are usually read-intensive
Use indexed data to improve response times	Store data in columnar format to allow easy access to large numbers of records
Require frequent or concurrent database backups	Require far less frequent database backup
Require relatively little storage space	Typically have significant storage space requirements, because they store large amounts of historical data
Usually run simple queries involving just one or a few records	Run complex queries involving large numbers of records

OLTP Examples:

1. Banking Systems: When you withdraw money from an ATM, transfer funds, or check your balance, an OLTP system processes these real-time transactions.
2. E-commerce Websites: When a customer places an order, updates their cart, or cancels a purchase, the OLTP system handles these concurrent transactions.
3. Ticket Booking Systems: Airline or event ticket booking platforms process seat reservations, cancellations, and payments using OLTP.
4. Point of Sale (POS) Systems: When a cashier scans products and processes payments, the POS system handles this transaction data in real time.

OLAP Examples:

1. Business Intelligence Dashboards: Analyzing sales trends over the past year to make decisions about inventory levels or marketing strategies.
2. Data Warehousing: A company runs queries to identify which products perform best in different regions and segments based on historical transaction data.
3. Financial Reporting: A finance team uses OLAP to generate reports that summarize monthly revenues, expenses, and profits from multiple stores.
4. Customer Analytics: A business might analyze customer purchase patterns, such as which products are frequently bought together, to optimize marketing efforts.

- In essence, OLTP is operational and real-time, while OLAP is analytical and retrospective, focusing on data aggregation and insights.