Chapter 3

Database Architectures and the Web Transparencies

Database Environment - Objectives

- The meaning of the client–server architecture and the advantages of this type of architecture for a DBMS.
- The difference between two-tier, three-tier and *n*-tier client–server architectures.
- About cloud computing and data as a service (DaaS) and database as a service (DBaaS).
- Software components of a DBMS.

Database Environment - Objectives

- The purpose of a Web service and the technological standards used to develop a Web service.
- The meaning of service-oriented architecture (SOA).
- The difference between distributed DBMSs, and distributed processing.
- The architecture of a data warehouse.
- About cloud computing and cloud databases.
- The software components of a DBMS.

Multi-user DBMS Architectures

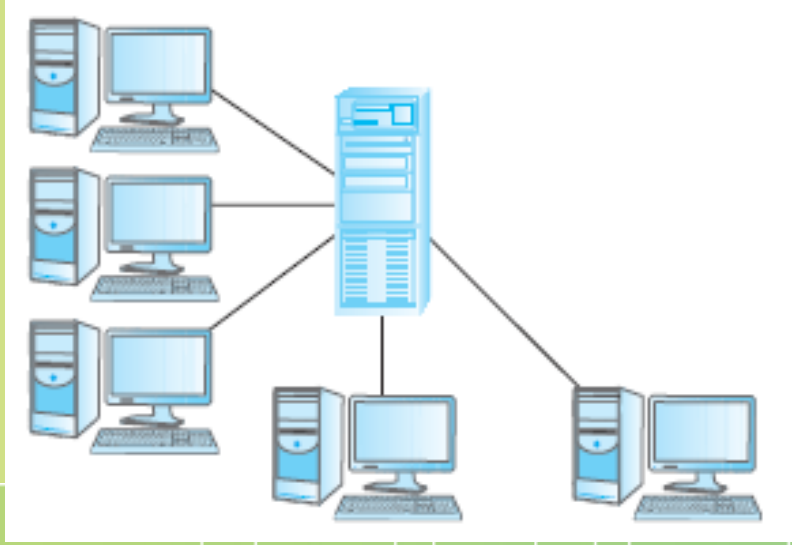
- The common architectures that are used to implement multi-user database management systems:
  - Teleprocessing
  - File-Server
  - Client-Server

File-Server

- File-server is connected to several workstations across a network.
- Database resides on file-server.
- DBMS and applications run on each workstation.
- Disadvantages include:
  - Significant network traffic.
  - Copy of DBMS on each workstation.
  - Concurrency, recovery and integrity control more complex.

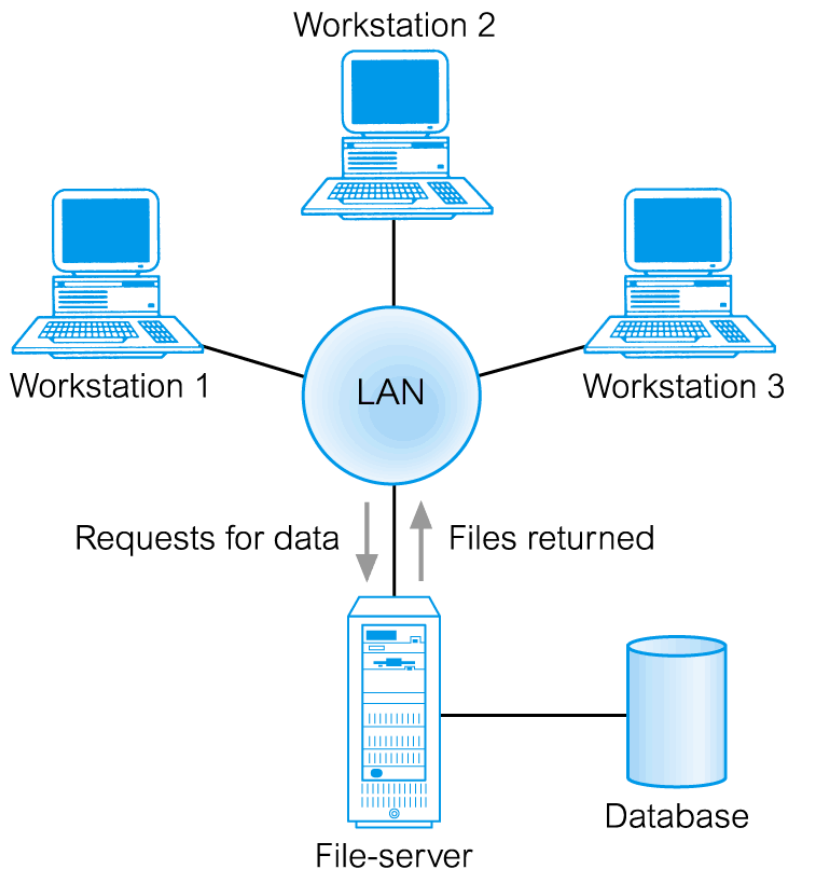
Teleprocessing

- One computer with a single CPU and a number of terminals.
- Processing performed within the same physical computer. User terminals are typically “dumb”, incapable of functioning on their own, and cabled to the central computer.



File-Server Architecture

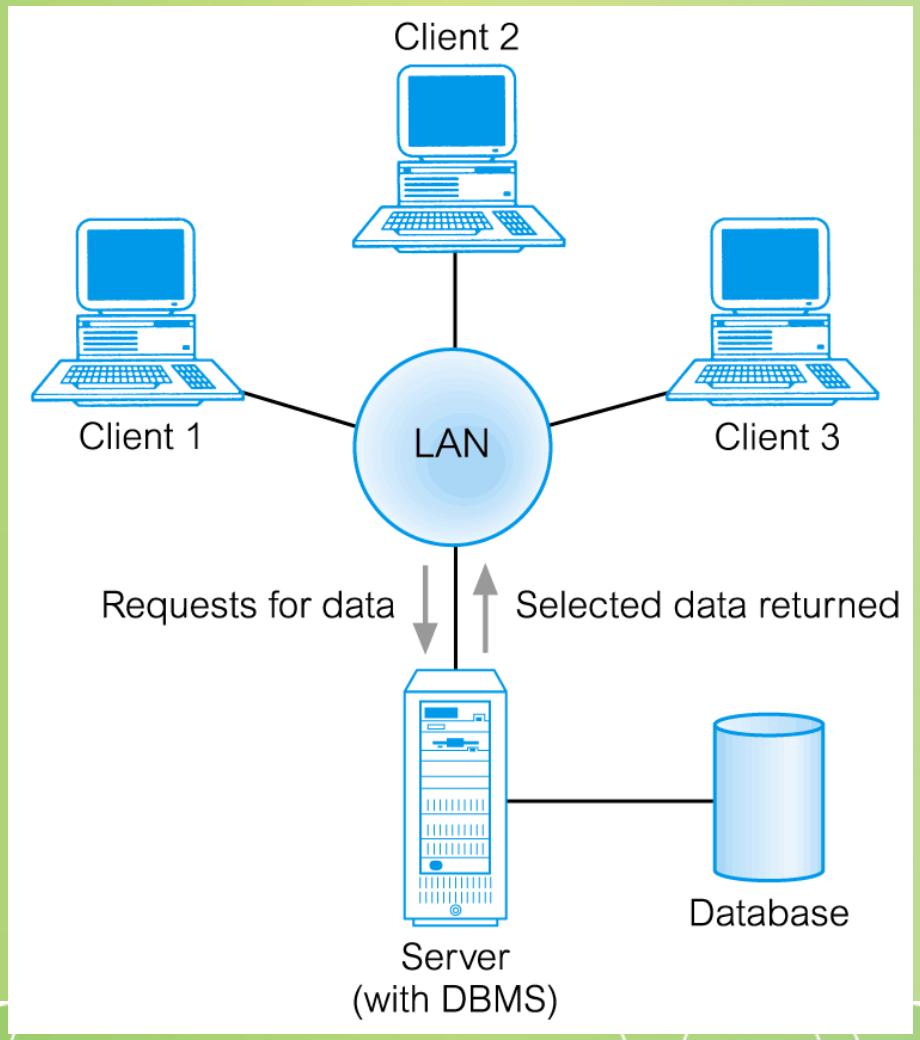
- **In a file-server environment, the processing is distributed about the network, typically a local area network (LAN).**



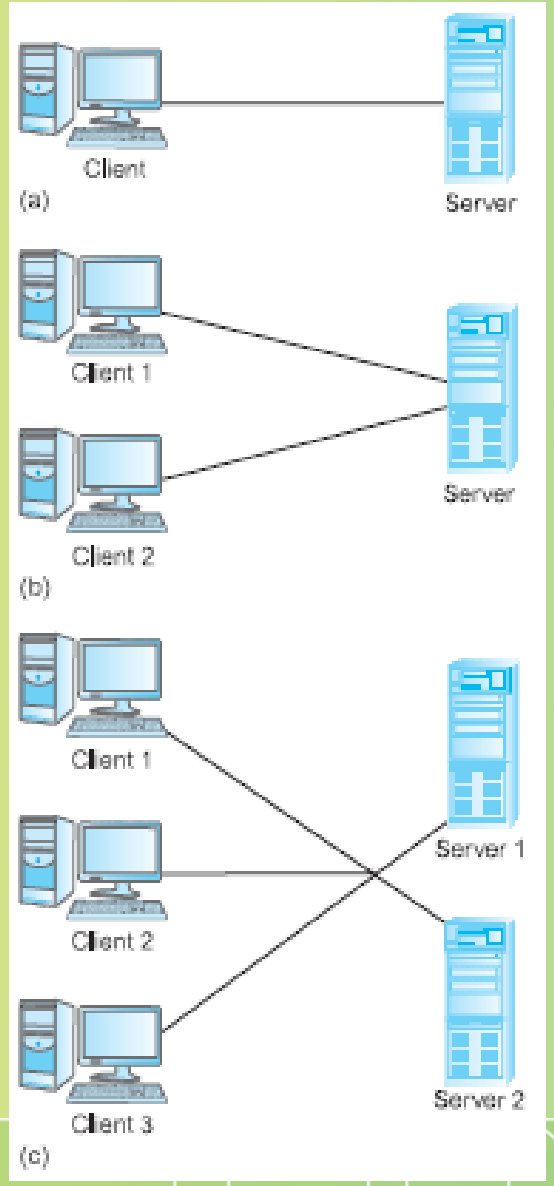
Traditional Two-Tier Client-Server

- Client (tier 1) manages user interface and runs applications.
- Server (tier 2) holds database and DBMS.
- Advantages include:
  - wider access to existing databases;
  - increased performance;
  - possible reduction in hardware costs;
  - reduction in communication costs;
  - increased consistency.

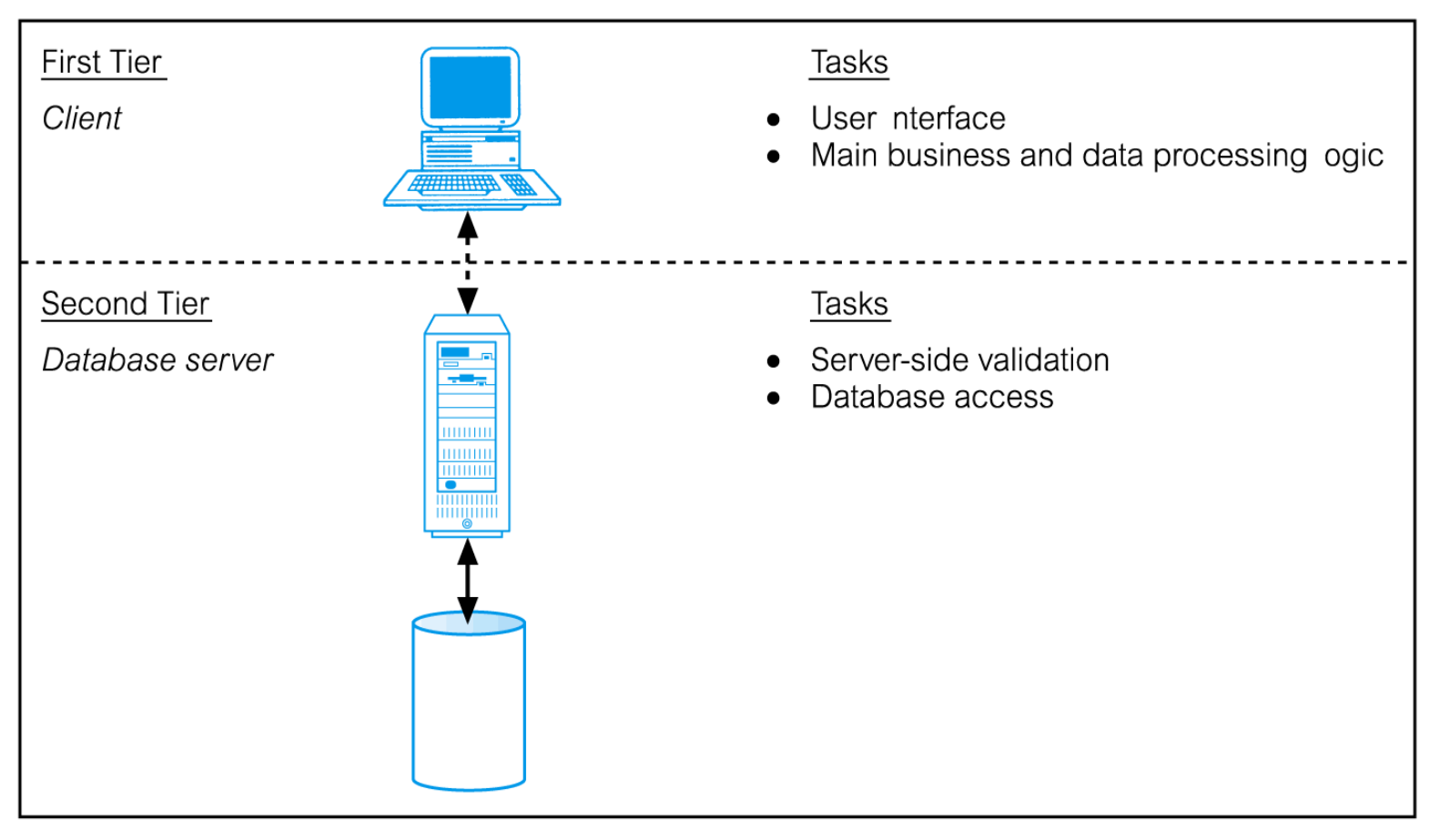
Traditional Two-Tier Client-Server



Alternative Client-Server Topologies



Traditional Two-Tier Client-Server



Summary of Client-Server Functions

CLIENT	SERVER
Manages the user interface	Accepts and processes database requests from clients
Accepts and checks syntax of user input	Checks authorization
Processes application logic	Ensures integrity constraints not violated
Generates database requests and transmits to server	Performs query/update processing and transmits response to client
Passes response back to user	Maintains system catalog Provides concurrent database access Provides recovery control

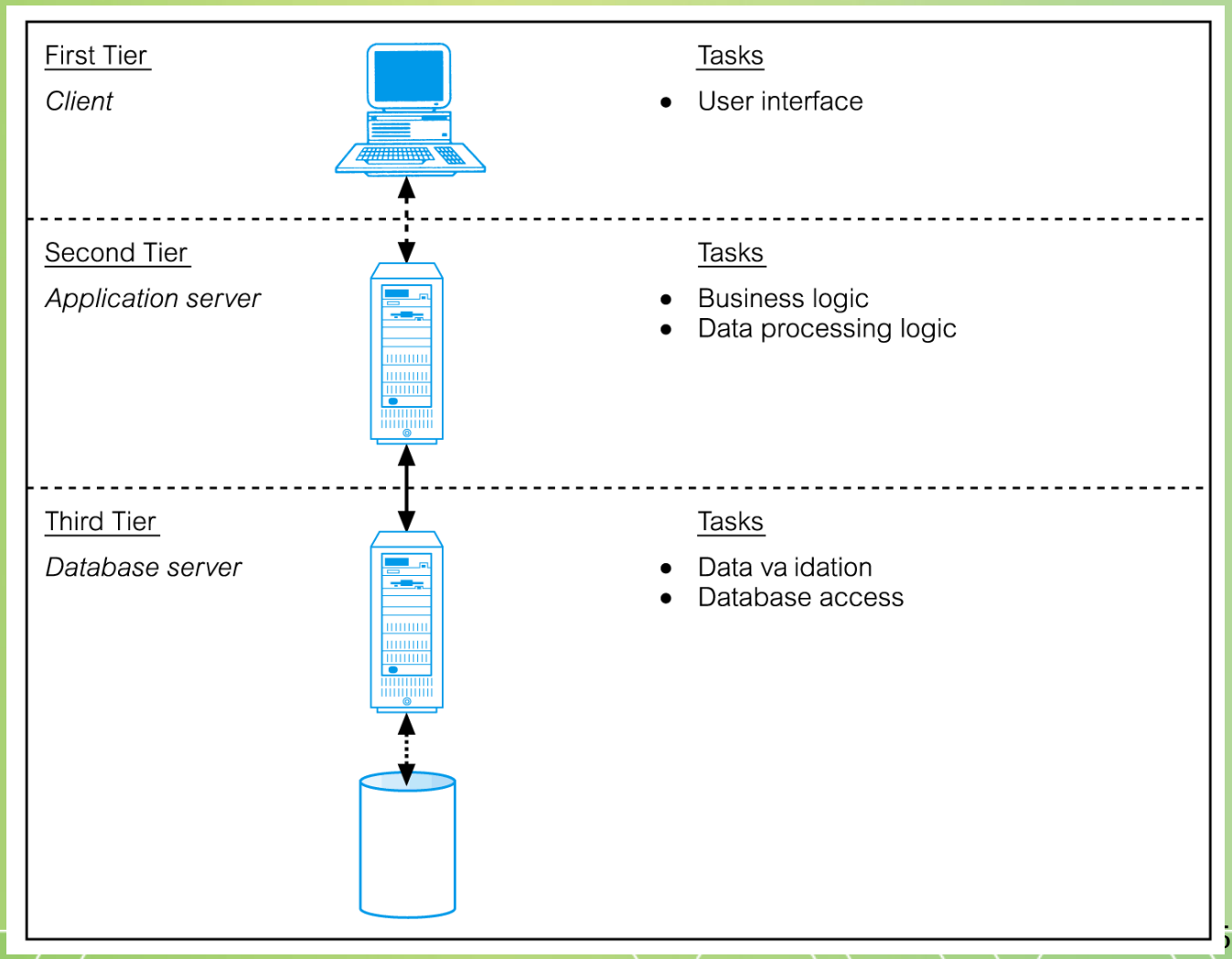
Three-Tier Client-Server

- The need for enterprise scalability challenged the traditional two-tier client–server model.
- Client side presented two problems preventing true scalability:
  - 'Fat' client, requiring considerable resources on client's computer to run effectively.
  - Significant client side administration overhead.
- By 1995, three layers proposed, each potentially running on a different platform.

Three-Tier Client-Server

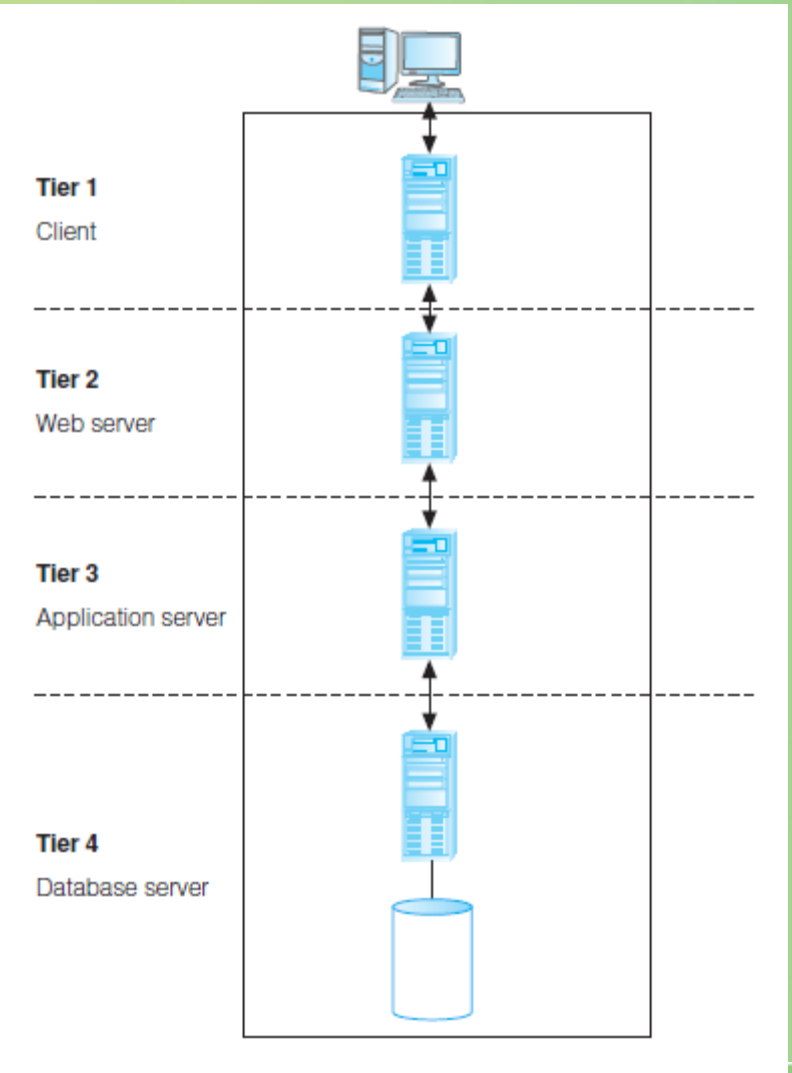
- **Advantages:**
  - **‘Thin’ client, requiring less expensive hardware.**
  - **Application maintenance centralized.**
  - **Easier to modify or replace one tier without affecting others.**
  - **Separating business logic from database functions makes it easier to implement load balancing.**
  - **Maps quite naturally to Web environment.**

Three-Tier Client-Server



n-Tier Client-Server (e.g. 4-Tier)

- The three-tier architecture can be expanded to *n* tiers, with additional tiers providing more flexibility and scalability.
- Applications servers host API to expose business logic and business processes for use by other applications.



Middleware

- Middleware is a generic term used to describe software that mediates with other software and allows for communication between disparate applications in a heterogeneous system.

- The need for middleware arises when distributed systems become too complex to manage efficiently without a common interface.

Cloud Computing

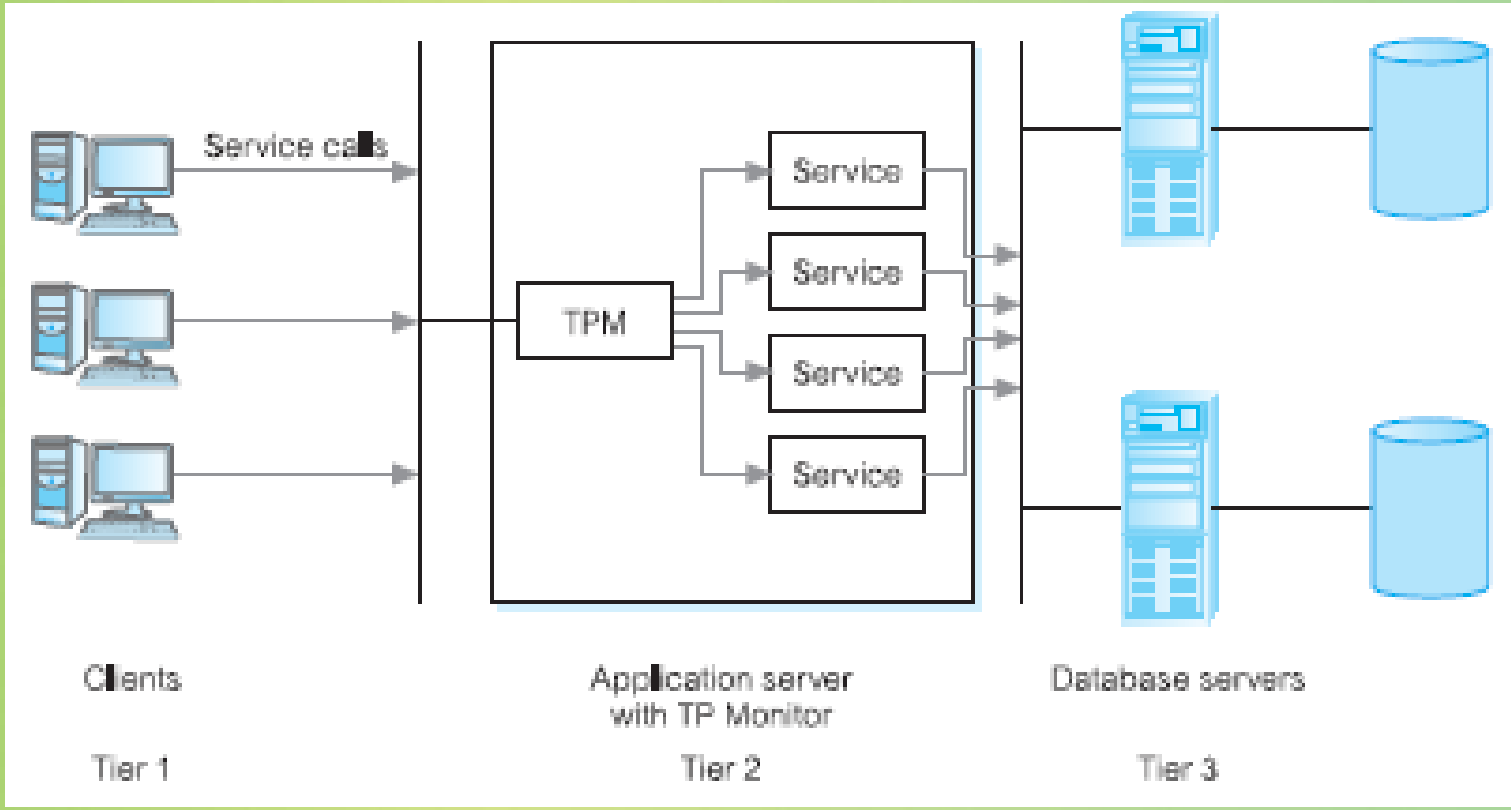
- The National Institute of Standards and Technology (NIST) provided a definition.

- Defined as "A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Transaction Processing Monitors

- **TP monitor is a program that controls data transfer between clients and servers in order to provide a consistent environment, particularly for online transaction processing (OLTP).**

Transaction Processing Monitor as middle tier of 3-tier client-server



Web Services and Service-Oriented Architectures

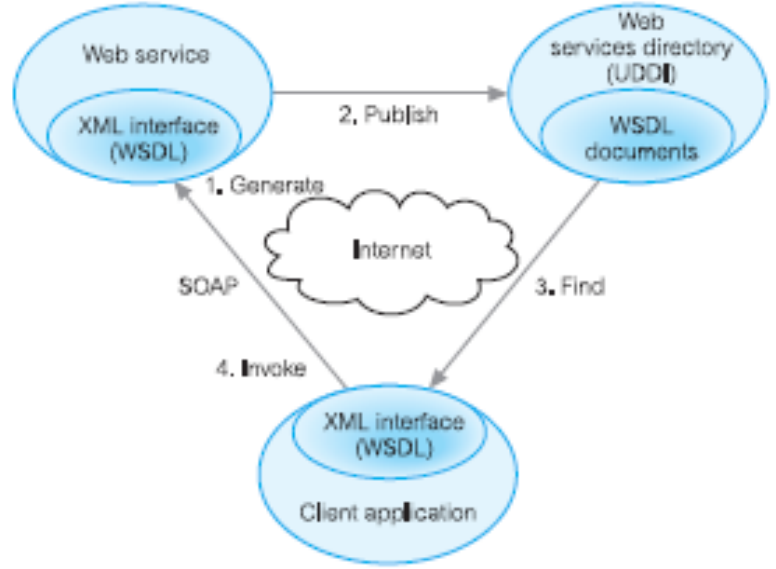
- Web service is a software system designed to support interoperable machine-to-web service machine interaction over a network.
- Web services share business logic, data, and processes through a programmatic interface across a network.
- Developers can add the Web service to a Web page (or an executable program) to offer specific functionality to users.

Web Services and Service-Oriented Architectures

- Web services approach uses accepted technologies and standards, such as:
  - XML (extensible Markup Language).
  - SOAP (Simple Object Access Protocol) is a communication protocol for exchanging structured information over the Internet and uses a message format based on XML. It is both platform- and language-independent.
  - WSDL (Web Services Description Language) protocol, again based on XML, is used to describe and locate a Web service.

Web Services and Service-Oriented Architectures

- UDDI (Universal Discovery, Description, and Integration) protocol is a platform independent, XML-based registry for businesses to list themselves on the Internet.



Service-Oriented Architectures (SOA)

- A business-centric software architecture for building applications that implement business processes as sets of services published at a granularity relevant to the service consumer. Services can be invoked, published, and discovered, and are abstracted away from the implementation using a single standards-based form of interface.

Distributed DBMSs

- A distributed database is a logically interrelated collection of shared data (and a description of this data), physically distributed over a computer network.

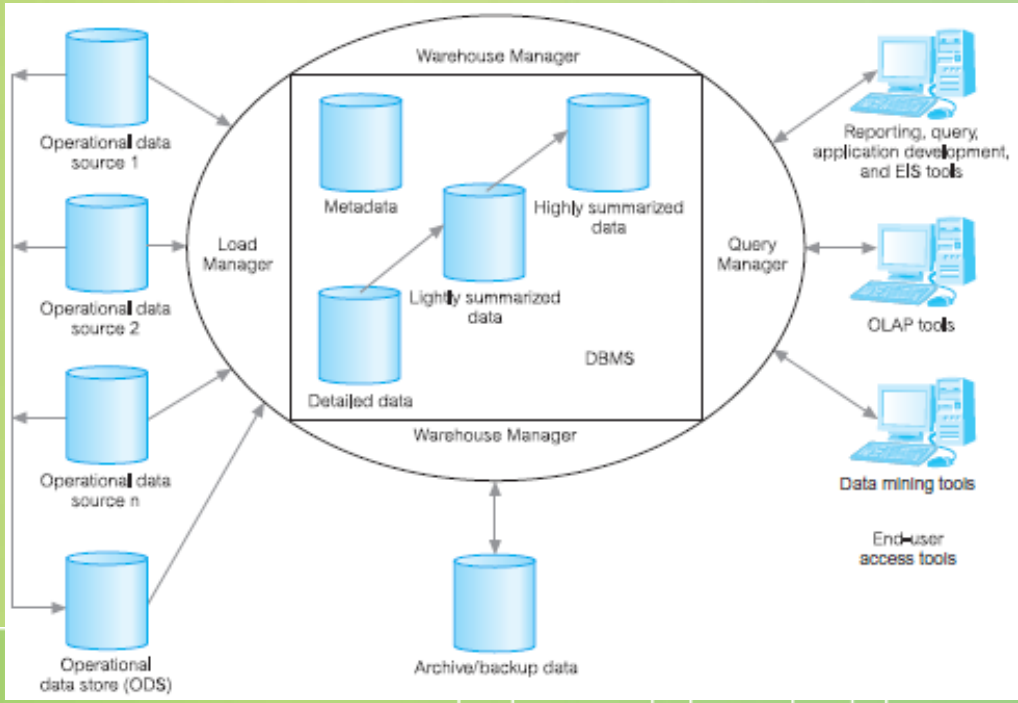
- A distributed DBMS is the software system that permits the management of the distributed database and makes the distribution transparent to users.

Distributed DBMSs

- A DDBMS consists of a single logical database split into a number *of fragments.*
- Each fragment is stored on one or more computers *(replicas)* under the control of a separate DBMS, with the computers connected by a network.
- Each site is capable of independently processing user requests that require access to local data (that is, each site has some degree of local autonomy) and is also capable of processing data stored on other computers in the network.

Data Warehousing

- **A data warehouse was deemed the solution to meet the requirements of a system capable of supporting decision making, receiving data from multiple operational data sources.**



Cloud Computing

- The National Institute of Standards and Technology (NIST) provided a definition.

- Defined as "A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Cloud Computing – Key Characteristics

- ***On-demand self-service***
  - **Consumers can obtain, configure and deploy cloud services without help from provider.**
- ***Broad network access***
  - **Accessible from anywhere, from any standardized platform (e.g. desktop computers, laptops, mobile devices).**

Cloud Computing – Key Characteristics

- ***Resource pooling***
  - **Provider's computing resources are pooled to serve multiple consumers, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. Examples of resources include storage, processing, memory, and network bandwidth.**

Cloud Computing – Key Characteristics

- ***Rapid elasticity***
  - **Provider's capacity caters for customer's spikes in demand and reduces risk of outages and service interruptions. Capacity can be automated to scale rapidly based on demand.**
- ***Measured service***
  - **Provider uses a metering capability to measure usage of service (e.g. storage, processing, bandwidth, and active user accounts).**

Cloud Computing – Service Models

- ***Software as a Service (SaaS):***
  - **Software and data hosted on cloud. Accessed through using thin client interface (e.g. web browser). Consumer may be offered limited user specific application configuration settings.**
  - **Examples include Salesforce.com sales management applications, NetSuite's integrated business management software, Google's Gmail and Cornerstone OnDemand.**

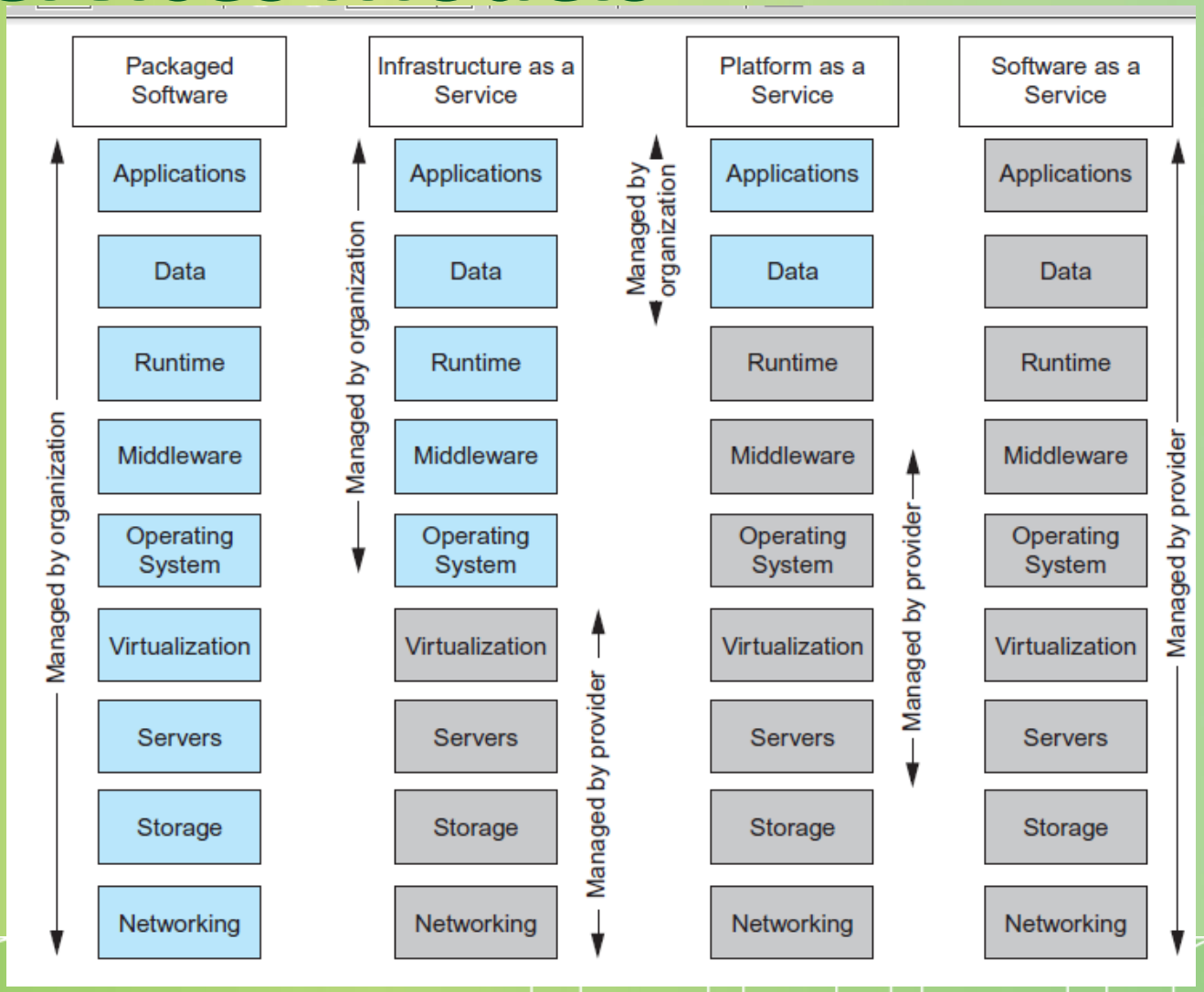
Cloud Computing – Service Models

- ***Platform as a Service (PaaS)***
  - **Allows creation of web applications without buying/maintaining the software and underlying infrastructure. Provider manages the infrastructure including network, servers, OS and storage, while customer controls deployment of applications and possibly configuration.**
  - **Examples include Salesforce.com's Force.com, Google's App Engine, and Microsoft's Azure.**

Cloud Computing – Service Models

- *Infrastructure as a Service (IaaS)*
  - **Provider's offer servers, storage, network and operating systems – typically a platform virtualization environment – to consumers as an on-demand service, in a single bundle and billed according to usage.**
  - **A popular use of IaaS is in hosting websites. Examples Amazon's Elastic Compute Cloud (EC2), Rackspace and GoGrid.**

Cloud Computing – Comparison of Services Models



Benefits of Cloud Computing

- ***Cost-Reduction*: Avoid up-front capital expenditure.**
- ***Scalability/Agility*: Organisations set up resources on an as-needs basis.**
- ***Improved Security*: Providers can devote expertise & resources to security; not affordable by customer.**
- ***Improved Reliability*: Providers can devote expertise & resources on reliability of systems; not affordable by customer.**
- ***Access to new technologies*: Through use of provider's systems, customers may access latest technology.**

Benefits of Cloud Computing

- ***Faster development*: Provider's platforms can provide many of the core services to accelerate development cycle.**
- ***Large scale prototyping/load testing*: Providers have the resources to enable this.**
- ***More flexible working practices*: Staff can access files using mobile devices.**
- ***Increased competitiveness*: Allows organizations to focus on their core competencies rather than their IT infrastructures.**

Risks of Cloud Computing

- ***Network Dependency*: Power outages, bandwidth issues and service interruptions.**
- ***System Dependency*: Customer's dependency on availability and reliability of provider's systems.**
- ***Cloud Provider Dependency*: Provider could became insolvent or acquired by competitor, resulting in the service suddenly terminating.**
- ***Lack of control*: Customers unable to deploy technical or organisational measures to safeguard the data. May result in reduced availability, integrity, confidentiality, intervenability and isolation.**
- ***Lack of information on processing transparency***

Cloud-based database solutions

- As a type of Software as a Service (SaaS), cloud-based database solutions fall into two basic categories:
  - Data as a Service (DaaS) and
  - Database as a Service (DBaaS).

- Key difference between the two options is mainly how the data is managed.

Cloud-based database solutions

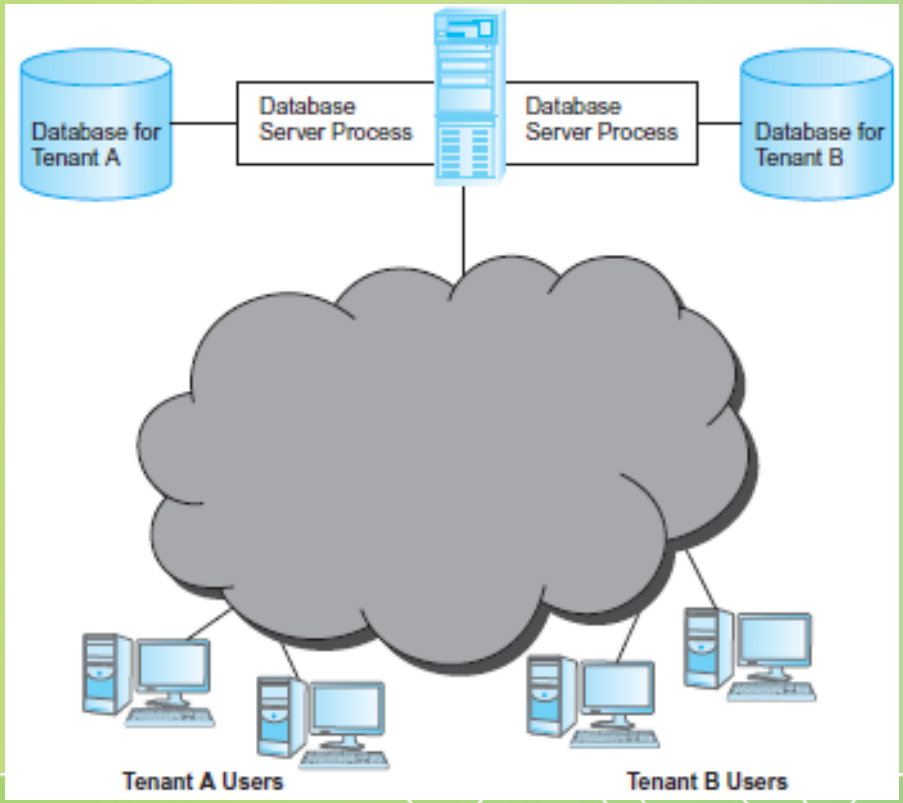
- DBaaS
  - Offers full database functionality to application developers.
  - Provides a management layer that provides continuous monitoring and configuring of the database to optimized scaling, high availability, multi-tenancy (that is, serving multiple client organizations), and effective resource allocation in the cloud, thereby sparing the developer from ongoing database administration tasks.

Cloud-based database solutions

- DaaS:
  - Services enables data definition in the cloud and subsequently querying.
  - Does not implement typical DBMS interfaces (e.g. SQL) but instead data is accessed via common APIs.
  - Enables organization with valuable data to offer access to others. Examples Urban Mapping (geography data service), Xignite (financial data service) and Hoovers (business data service.)

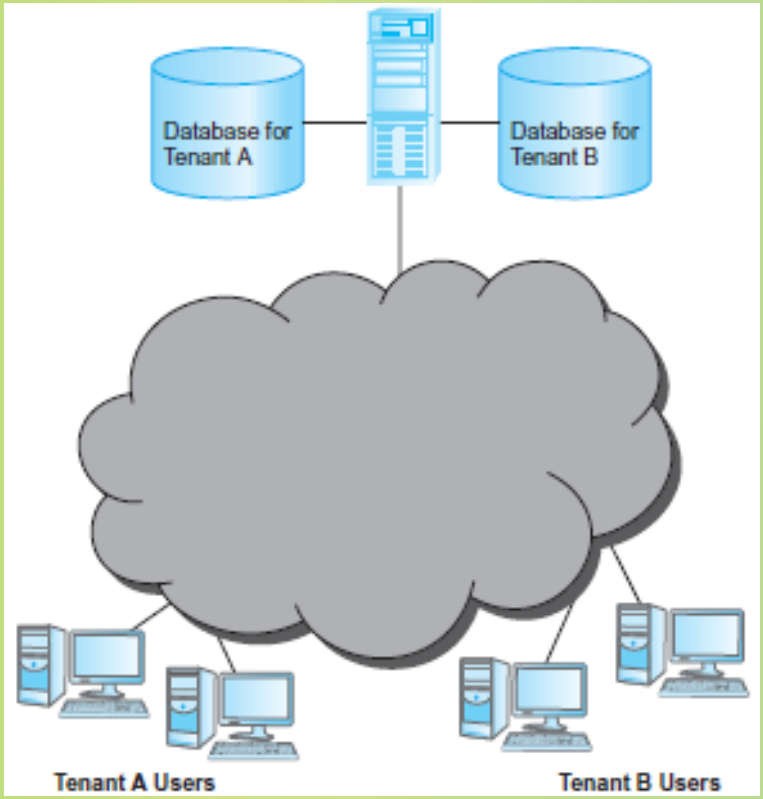
Cloud-based database solutions

- **Multi-tenant cloud database-shared server, separate database server process architecture.**



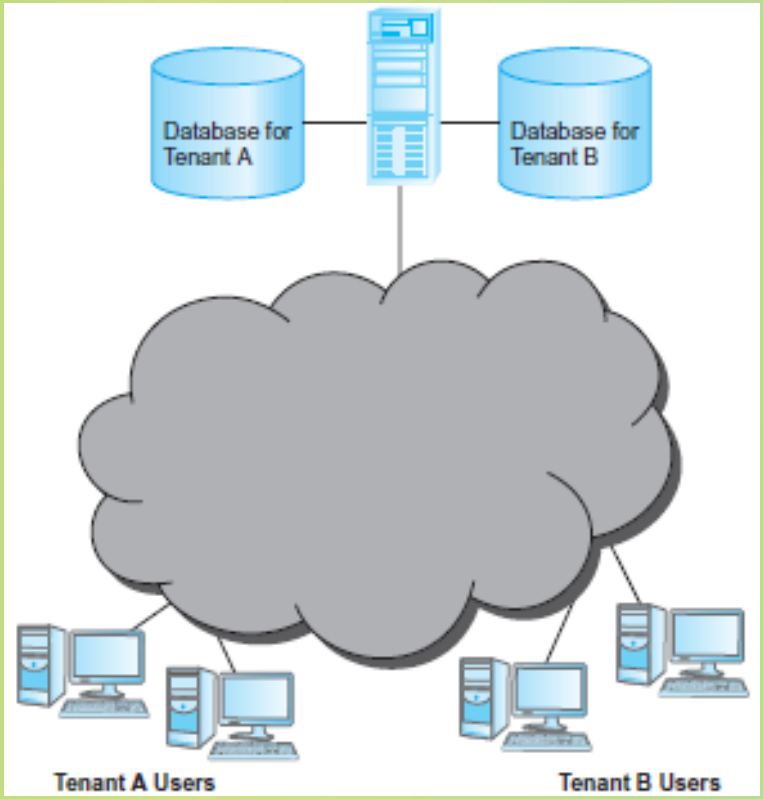
Cloud-based database solutions

- **Multi-tenant cloud database-shared DBMS server, separate databases.**



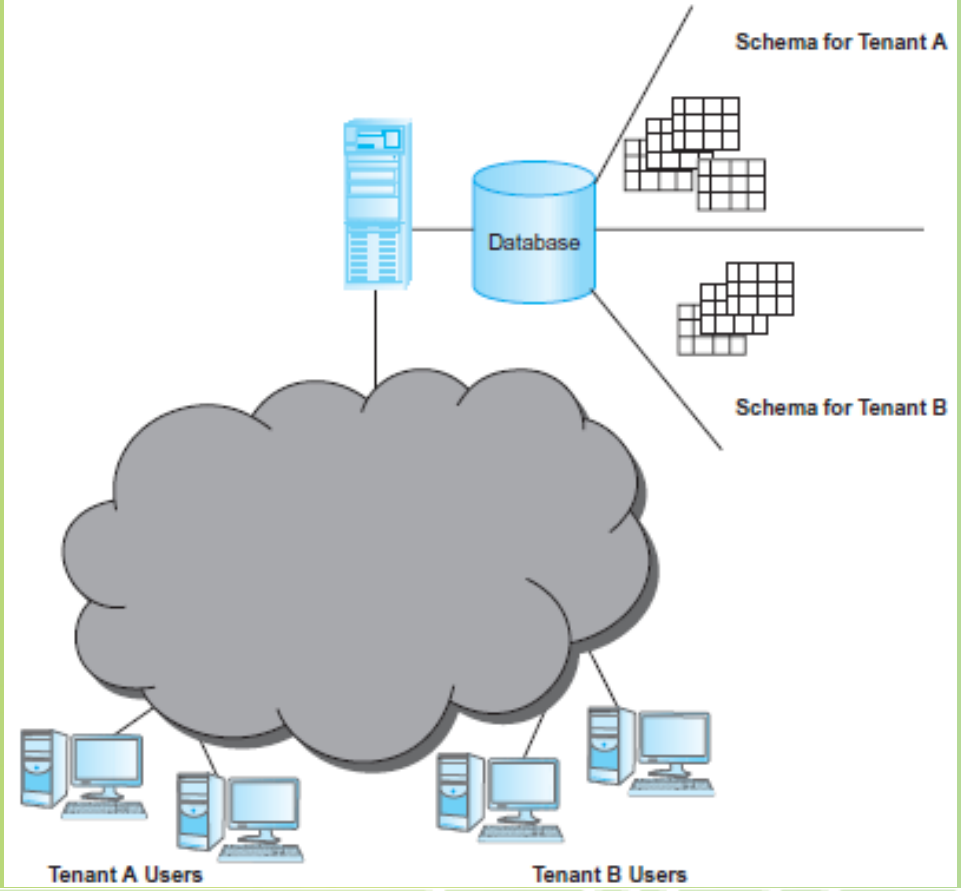
Cloud-based database solutions

- **Multi-tenant cloud database-shared DBMS server, separate databases.**



Cloud-based database solutions

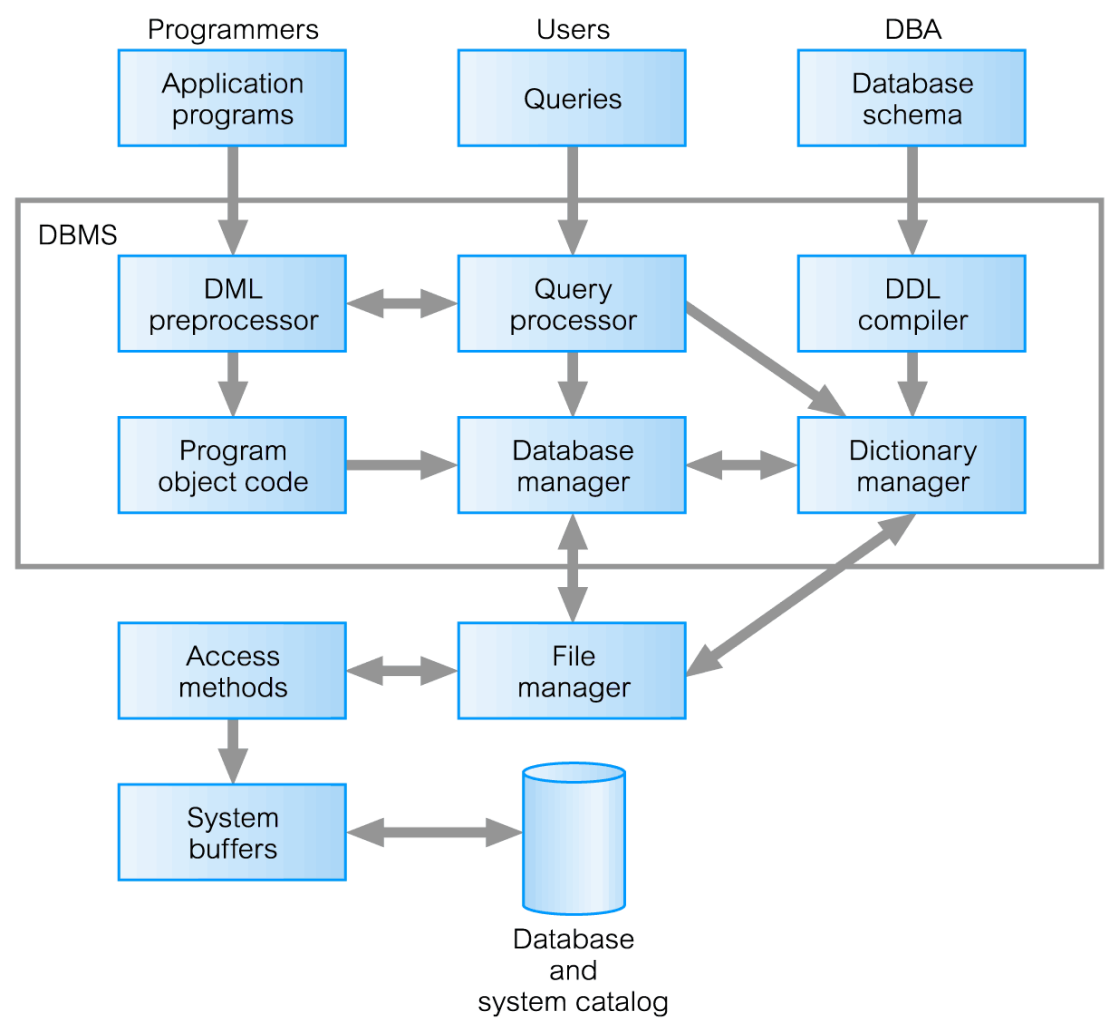
- **Multi-tenant cloud database–shared database, separate schema architecture.**



Components of a DBMS

- A DBMS is partitioned into several software components (or *modules),* each of which is assigned a specific operation. As stated previously, some of the functions of the DBMS are supported by the underlying operating system.
- The DBMS interfaces with other software components, such as user queries and access methods (file management techniques for storing and retrieving data records).

Components of a DBMS



Components of a DBMS (Continued)

- *Query processor* is a major DBMS component that transforms queries into a series of low-level instructions directed to the database manager.
- *Database manager (DM)* interfaces with user-submitted application programs and queries. The DM examines the external and conceptual schemas to determine what conceptual records are required to satisfy the request. The DM then places a call to the file manager to perform the request.

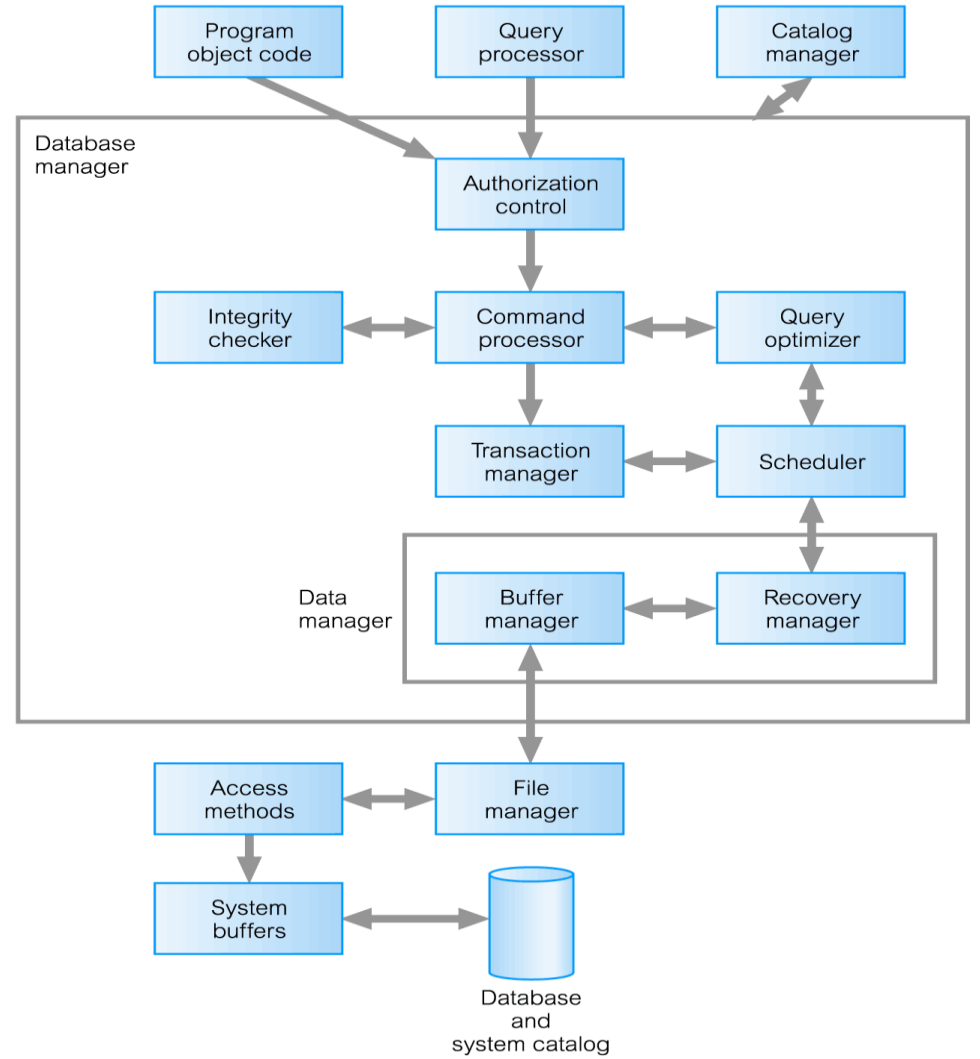
Components of a DBMS (Continued)

- *File manager* manipulates the underlying storage files and manages the allocation of storage space on disk. It establishes and maintains the list of structures and indexes defined in the internal schema.
- *DML preprocessor* converts DML statements embedded in an application program into standard function calls in the host language. The DML preprocessor must interact with the query processor to generate the appropriate code.

Components of a DBMS (Continued)

- *DDL compiler* converts DDL statements into a set of tables containing metadata. These tables are then stored in the system catalog while control information is stored in data file headers.
- *Catalog manager* manages access to and maintains the system catalog. The system catalog is accessed by most DBMS components.

Components of Database Manager (DM)



Components of the Database Manager

- ***Authorization control*** **to confirm whether the user has the necessary permission to carry out the required operation.**
- ***Command processor*** **on confirmation of user authority, control is passed to the command processor.**
- ***Integrity checker*** **ensures that requested operation satisfies all necessary integrity constraints (e.g. key constraints) for an operation that changes the database.**

Components of the Database Manager (Continued)

- *Query optimizer* determines an optimal strategy for the query execution.
- *Transaction manager* performs the required processing of operations that it receives from transactions.
- *Scheduler* ensures that concurrent operations on the database proceed without conflicting with one another. It controls the relative order in which transaction operations are executed.

Components of the Database Manager (Continued)

- ***Recovery manager*** **ensures that the database remains in a consistent state in the presence of failures. It is responsible for transaction commit and abort.**
- ***Buffer manager*** **responsible for the transfer of data between main memory and secondary storage, such as disk and tape.**
- **The recovery manager and the buffer manager also known as (aka) the *data manager.* The buffer manager aka the *cache manager.***