University of Windsor

00 Background

Dr. Kalyani Selvarajah
School of Computer Science
University of Windsor

Advanced Database Topics
COMP 8157 01/02
Winter 2022

Outlines

Introduction to Databases

Database Environment

Database Architectures

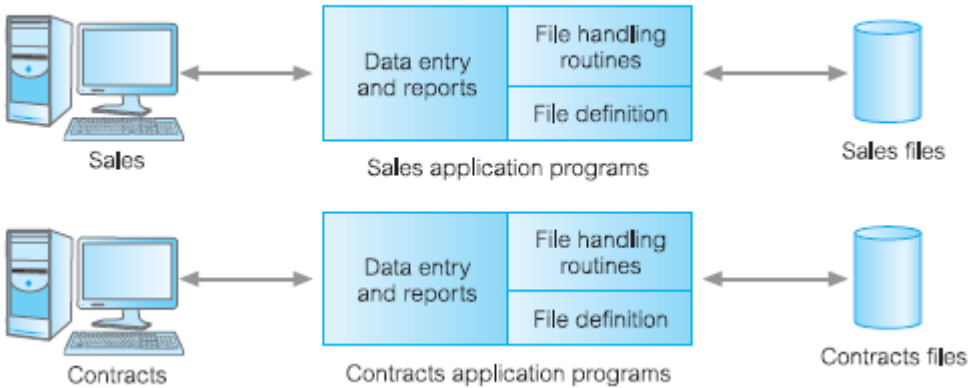
Examples of Database Applications

- Purchases from the supermarket
- Purchases using your credit card
- Booking a holiday at the travel agents
- Using the local library
- Taking out insurance
- Renting a video
- Using the Internet
- Studying at university

File-Based Systems

- Collection of application programs that perform services for the end users (e.g. reports).
- Each program defines and manages its own data.

File-Based Processing



Sales Files

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)

Contracts Files

Lease (leaseNo, propertyNo, clientNo, rent, paymentMethod, deposit, paid, rentStart, rentFinish, duration)
PropertyForRent (propertyNo, street, city, postcode, rent)
Client (clientNo, fName, lName, address, telNo)

Limitations of File-Based Approach

- Separation and isolation of data
  - Each program maintains its own set of data.
  - Users of one program may be unaware of potentially useful data held by other programs.

- Duplication of data
  - Same data is held by different programs.
  - Wasted space and potentially different values and/or different formats for the same item.

Limitations of File-Based Approach

- Data dependence
  - File structure is defined in the program code.
- Incompatible file formats
  - Programs are written in different languages, and so cannot easily access each other's files.
- Fixed Queries/Proliferation of application programs
  - Programs are written to satisfy particular functions.
  - Any new requirement needs a new program.

Database Approach

- Arose because:
  - Definition of data was embedded in application programs, rather than being stored separately and independently.
  - No control over access and manipulation of data beyond that imposed by application programs.

- Result:
  - the database and Database Management System (DBMS).

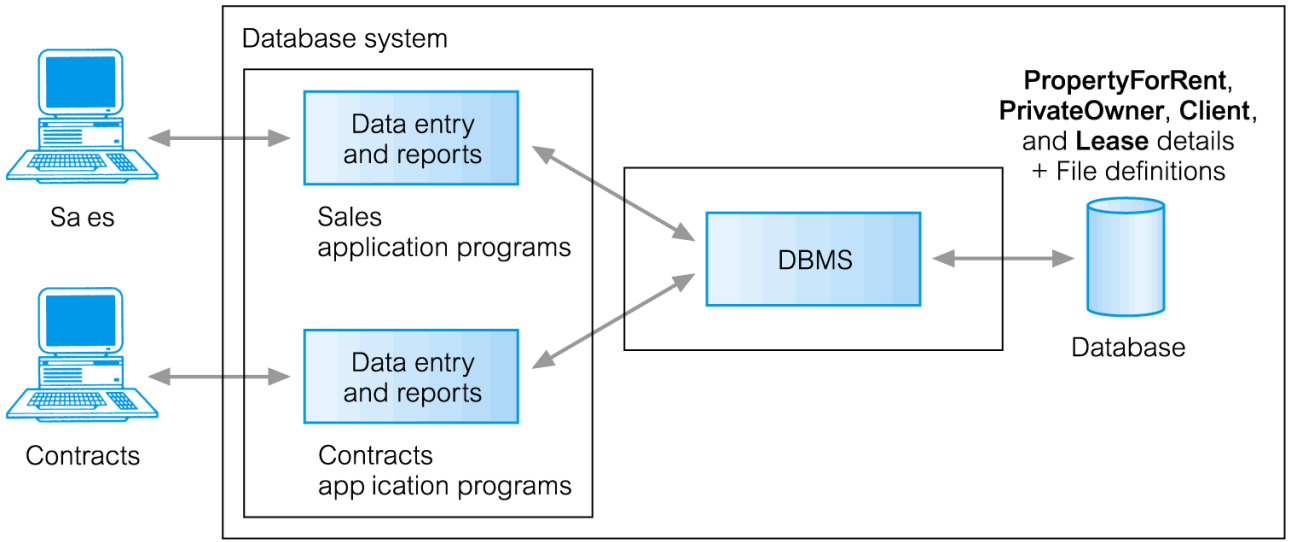
Database

- Shared collection of logically related data (and a description of this data), designed to meet the information needs of an organization.
- System catalog (metadata) provides description of data to enable program–data independence.
- Logically related data comprises entities, attributes, and relationships of an organization's information.

Database Management System (DBMS)

- A software system that enables users to define, create, maintain, and control access to the database.
- (Database) application program: a computer program that interacts with database by issuing an appropriate request (SQL statement) to the DBMS.

Database Management System (DBMS)



PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)
Lease (leaseNo, propertyNo, clientNo, paymentMethod, deposit, paid, rentStart, rentF nish)

Database Approach

- Data definition language (DDL).
  - Permits specification of data types, structures and any data constraints.
  - All specifications are stored in the database.

- Data manipulation language (DML).
  - General enquiry facility (query language) of the data.

Database Approach

- Controlled access to database may include:
  - a security system
  - an integrity system
  - a concurrency control system
  - a recovery control system
  - a user-accessible catalog.

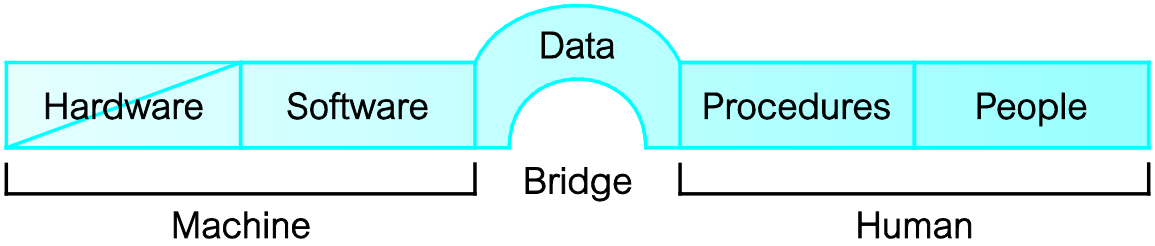
Views

- Allows each user to have his or her own view of the database.
- A view is essentially some subset of the database.

Views - Benefits

- Reduce complexity
- Provide a level of security
- Provide a mechanism to customize the appearance of the database
- Present a consistent, unchanging picture of the structure of the database, even if the underlying database is changed

Components of DBMS Environment



Components of DBMS Environment

- Hardware
  - Can range from a PC to a network of computers.
- Software
  - DBMS, operating system, network software (if necessary) and also the application programs.
- Data
  - Used by the organization and a description of this data called the schema.

Components of DBMS Environment

- Procedures
  - Instructions and rules that should be applied to the design and use of the database and DBMS.
- People

Roles in the Database Environment

- Data Administrator (DA)
- Database Administrator (DBA)
- Database Designers (Logical and Physical)
- Application Programmers
- End Users (naive and sophisticated)

History of Database Systems

- First-generation
  - Hierarchical and Network
- Second generation
  - Relational
- Third generation
  - Object-Relational
  - Object-Oriented

Advantages of DBMSs

- Control of data redundancy
- Data consistency
- More information from the same amount of data
- Sharing of data
- Improved data integrity
- Improved security
- Enforcement of standards
- Economy of scale

Advantages of DBMSs

- Balance conflicting requirements
- Improved data accessibility and responsiveness
- Increased productivity
- Improved maintenance through data independence
- Increased concurrency
- Improved backup and recovery services

Disadvantages of DBMSs

- Complexity
- Size
- Cost of DBMS
- Additional hardware costs
- Cost of conversion
- Performance
- Higher impact of a failure

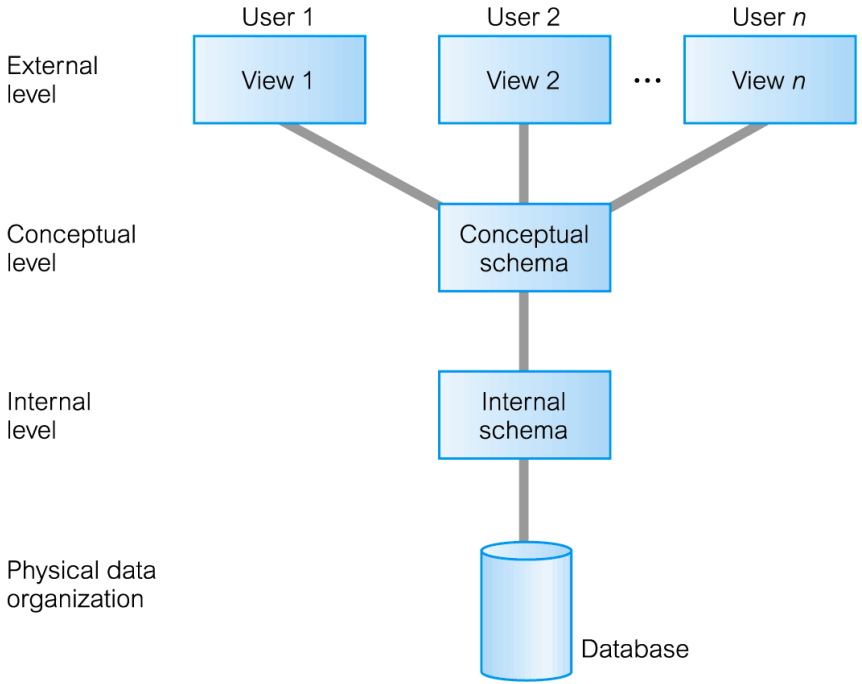
Objectives of Three-Level Architecture

- All users should be able to access same data.
- A user's view is immune to changes made in other views.
- Users should not need to know physical database storage details.

Objectives of Three-Level Architecture

- DBA should be able to change database storage structures without affecting the users' views.
- Internal structure of database should be unaffected by changes to physical aspects of storage.
- DBA should be able to change conceptual structure of database without affecting all users.

ANSI-SPARC Three-Level Architecture



ANSI-SPARC Three-Level Architecture

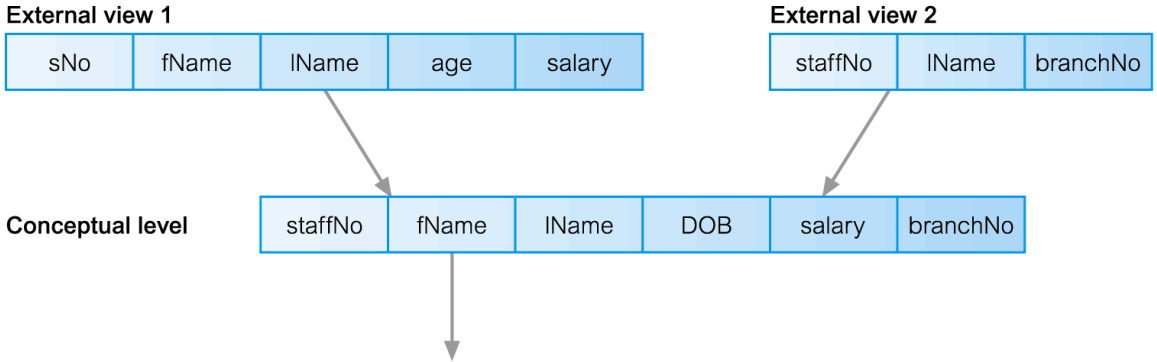
- External Level
  - Users' view of the database.
  - Describes that part of database that is relevant to a particular user.

- Conceptual Level
  - Community view of the database.
  - Describes what data is stored in database and relationships among the data.

ANSI-SPARC Three-Level Architecture

- Internal Level
  - Physical representation of the database on the computer.
  - Describes how the data is stored in the database.

Differences between Three Levels of ANSI-SPARC Architecture



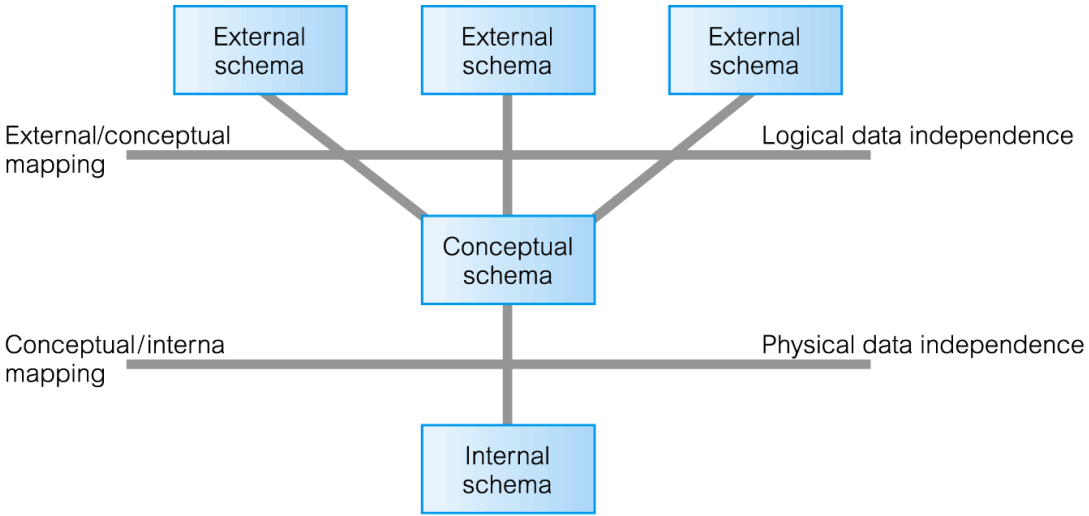
Data Independence

- Logical Data Independence
  - Refers to immunity of external schemas to changes in conceptual schema.
  - Conceptual schema changes (e.g. addition/removal of entities).
  - Should not require changes to external schema or rewrites of application programs.

Data Independence

- Physical Data Independence
  - Refers to immunity of conceptual schema to changes in the internal schema.
  - Internal schema changes (e.g. using different file organizations, storage structures/devices).
  - Should not require change to conceptual or external schemas.

Data Independence and the ANSI-SPARC Three-Level Architecture



Database Languages

- Data Definition Language (DDL)
  - Allows the DBA or user to describe and name entities, attributes, and relationships required for the application
  - plus any associated integrity and security constraints.

Database Languages

- Data Manipulation Language (DML)
  - Provides basic data manipulation operations on data held in the database.
- Procedural DML
  - allows user to tell system exactly how to manipulate data.
- Non-Procedural DML
  - allows user to state what data is needed rather than how it is to be retrieved.
- Fourth Generation Languages (4GLs)

Data Model

- Integrated collection of concepts for describing data, relationships between data, and constraints on the data in an organization.

- Data Model comprises:
  - a structural part;
  - a manipulative part;
  - possibly a set of integrity rules.

Data Model

- Purpose
  - To represent data in an understandable way.

- Categories of data models include:
  - Object-based
  - Record-based
  - Physical.

Data Models

- Object-Based Data Models
  - Entity-Relationship
  - Semantic
  - Functional
  - Object-Oriented.
- Record-Based Data Models
  - Relational Data Model
  - Network Data Model
  - Hierarchical Data Model.
- Physical Data Models

Relational Data Model

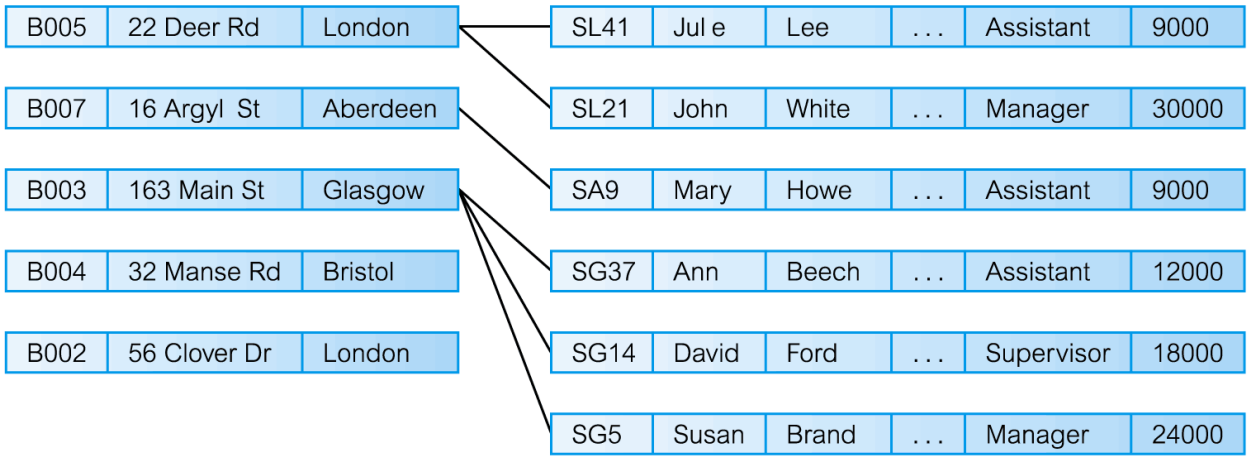
Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

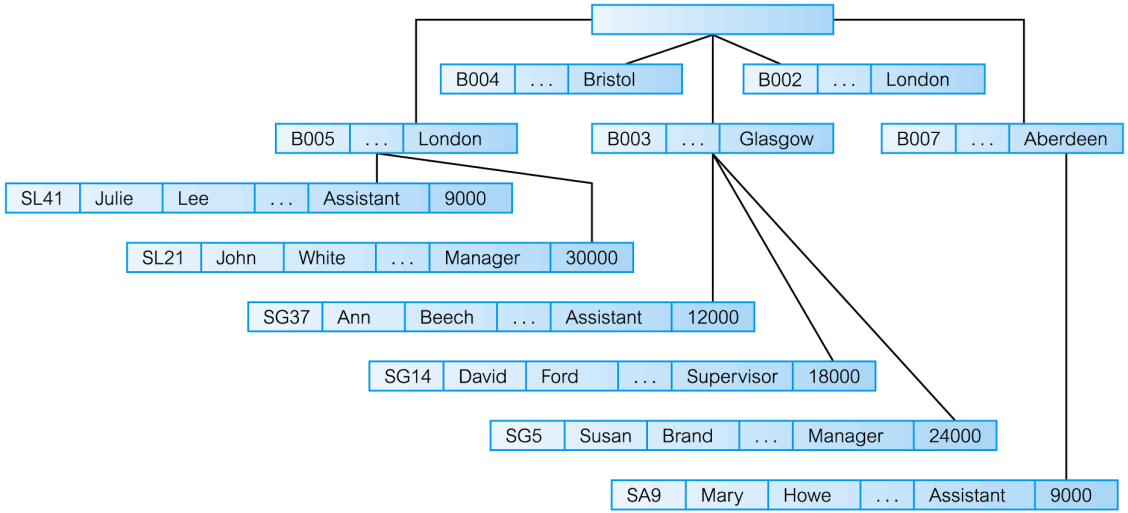
Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Network Data Model



Hierarchical Data Model



Conceptual Modeling

- Conceptual schema is the core of a system supporting all user views.
- Should be complete and accurate representation of an organization's data requirements.

- Conceptual modeling is process of developing a model of information use that is independent of implementation details.
- Result is a conceptual data model.

Functions of a DBMS

- Data Storage, Retrieval, and Update.
- A User-Accessible Catalog.
- Transaction Support.
- Concurrency Control Services.
- Recovery Services.

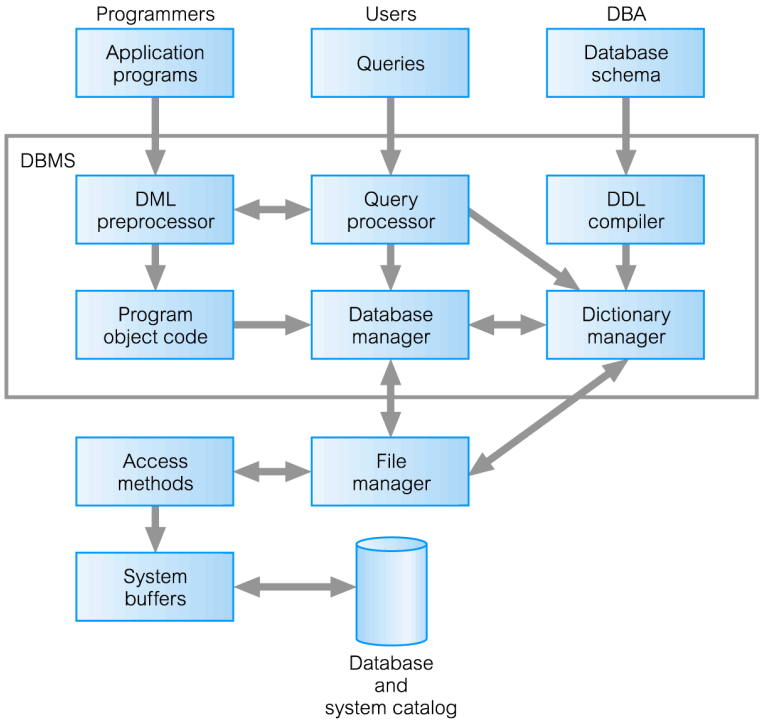
Functions of a DBMS

- Authorization Services.
- Support for Data Communication.
- Integrity Services.
- Services to Promote Data Independence.
- Utility Services.

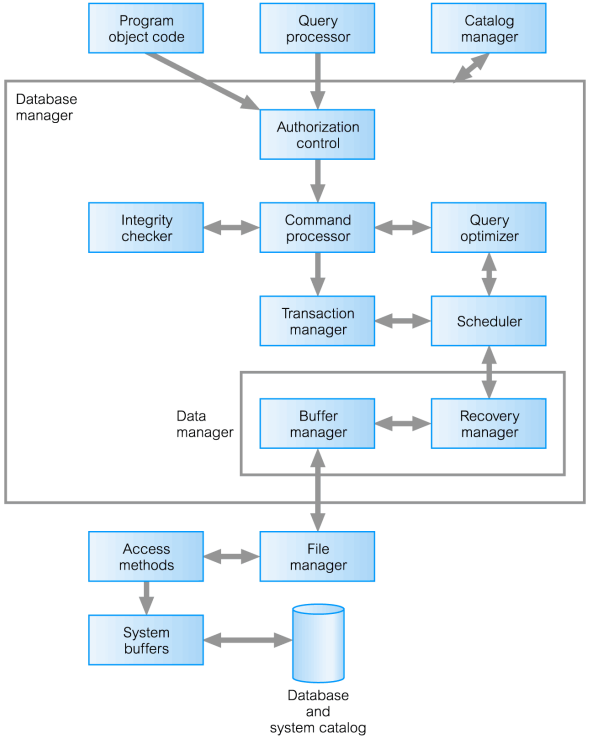
System Catalog

- Repository of information (metadata) describing the data in the database.
- One of the fundamental components of DBMS.
- Typically stores:
  - names, types, and sizes of data items;
  - constraints on the data;
  - names of authorized users;
  - data items accessible by a user and the type of access;
  - usage statistics.

Components of a DBMS



Components of Database Manager

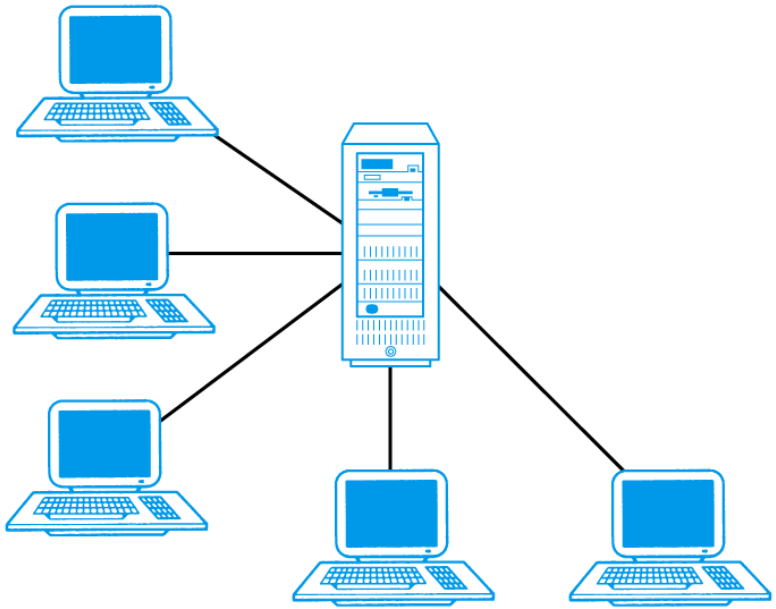


Multi-User DBMS Architectures

- Teleprocessing
- File-server
- Client-server

Teleprocessing

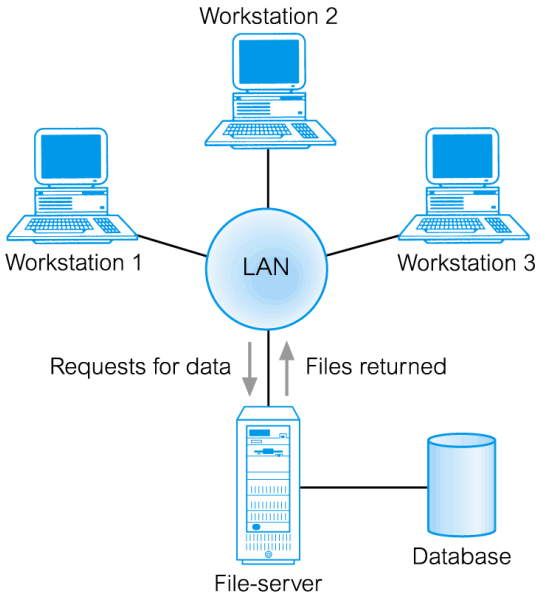
- Traditional architecture.
- Single mainframe with a number of terminals attached.
- Trend is now towards downsizing.



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.

File-Server Architecture

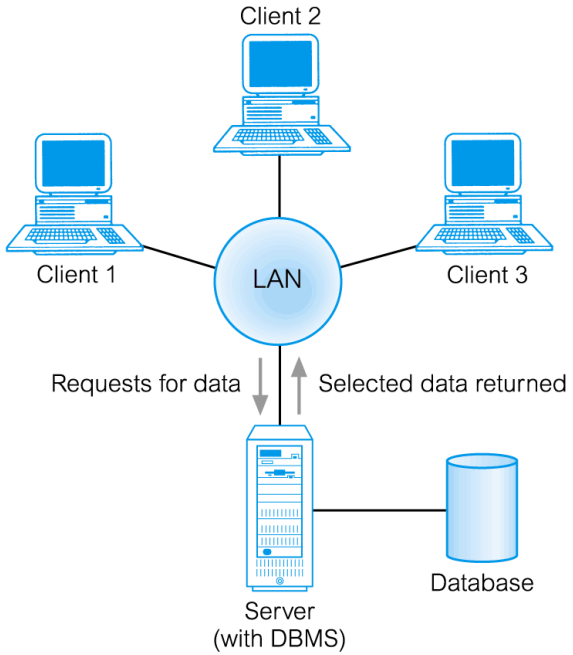


Traditional Two-Tier Client-Server

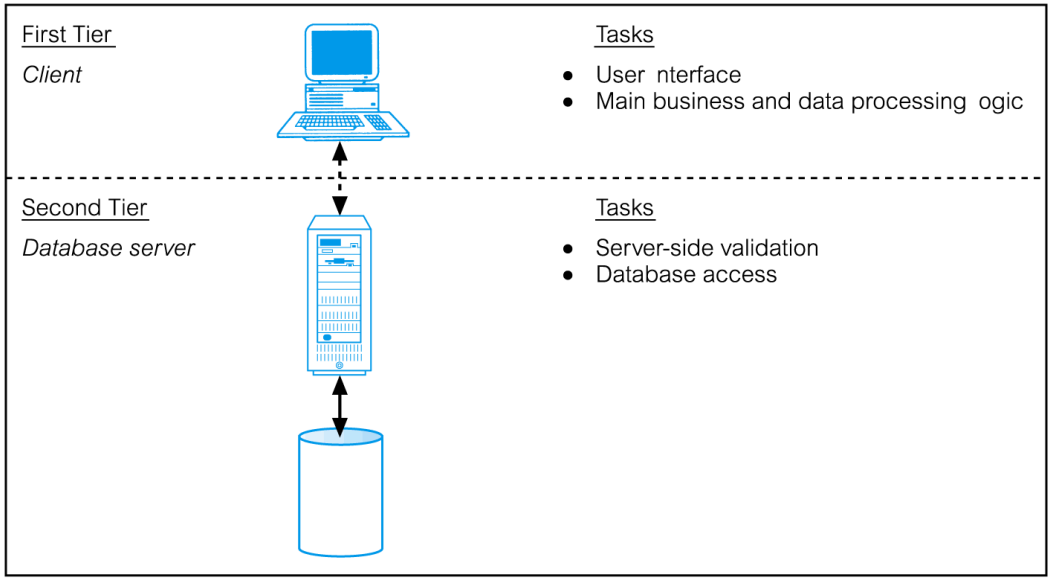
- Client (tier 1) manages user interface and runs applications.
- Server (tier 2) holds database and DBMS.

- 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



Traditional Two-Tier Client-Server



Three-Tier Client-Server

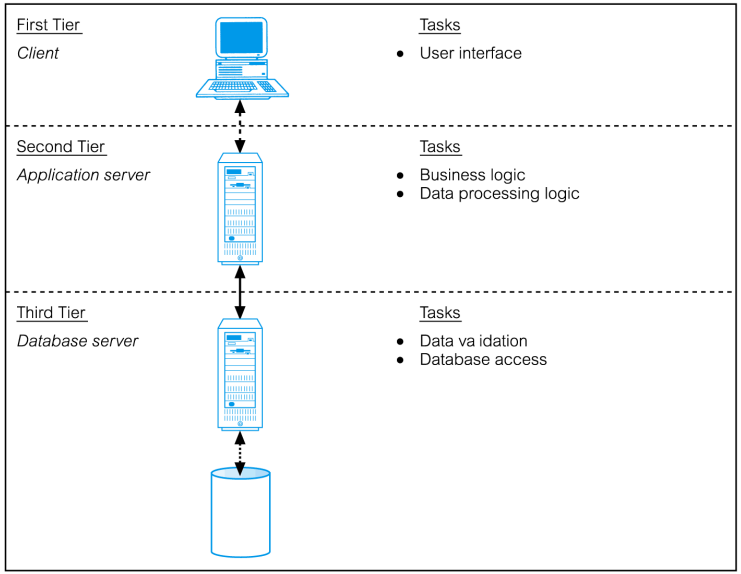
- Client side presented two problems preventing true scalability:
  - 'Fat' client, requiring considerable resources on client's computer to run effectively.
  - Significant client side administration overhead.

- 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.

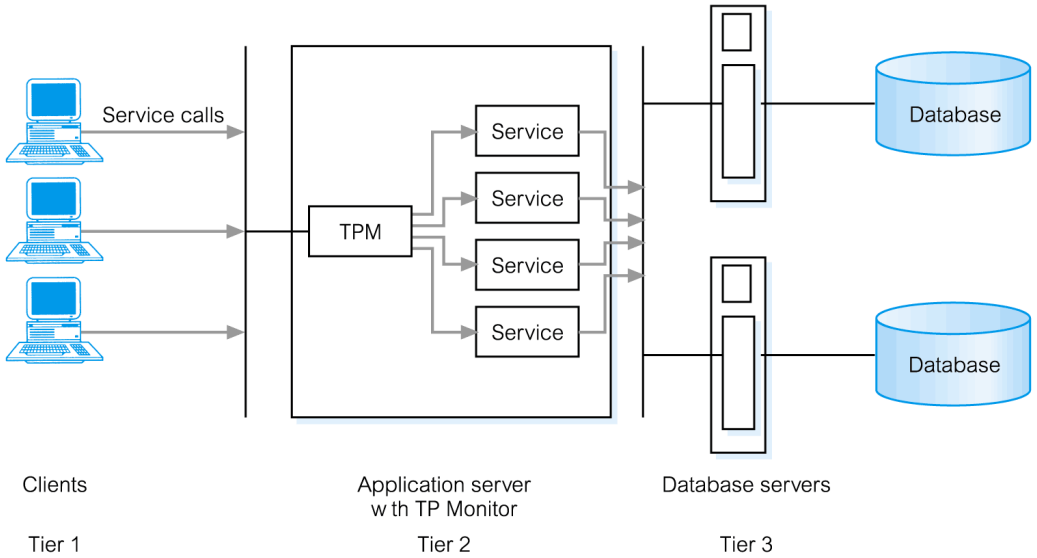
Three-Tier Client-Server



Transaction Processing Monitors

- Program that controls data transfer between clients and servers in order to provide a consistent environment, particularly for Online Transaction Processing (OLTP).

TPM as middle tier of 3-tier client-server



Multi-user DBMS Architectures

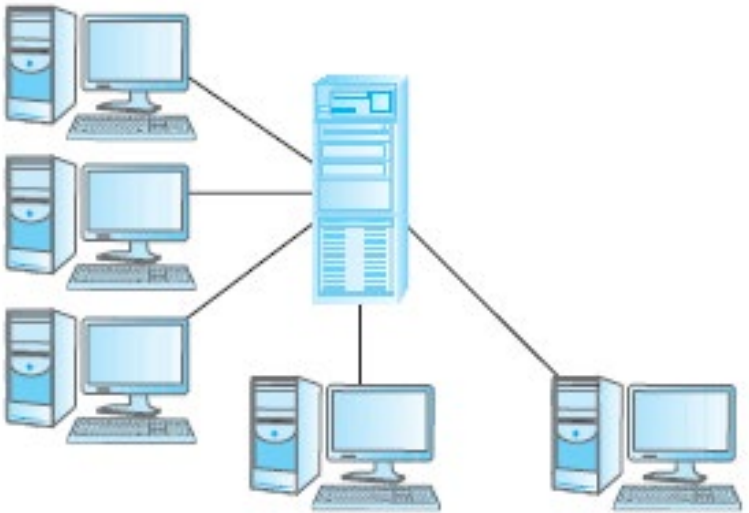
- The common architectures that are used to implement multi-user database management systems:
  - Teleprocessing
  - File-Server
  - Client-Server

File-Server

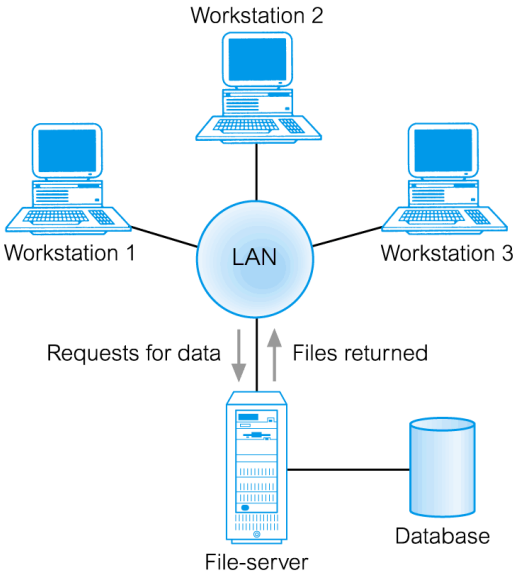
- File-server is connected to several workstations across a network.
- Database resides on file-server.
- DBMS and applications run on each workstation.
- Disadvantages include:
  - Significant network traffic.
  - Copy of DBMS on each workstation.
  - Concurrency, recovery and integrity control more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



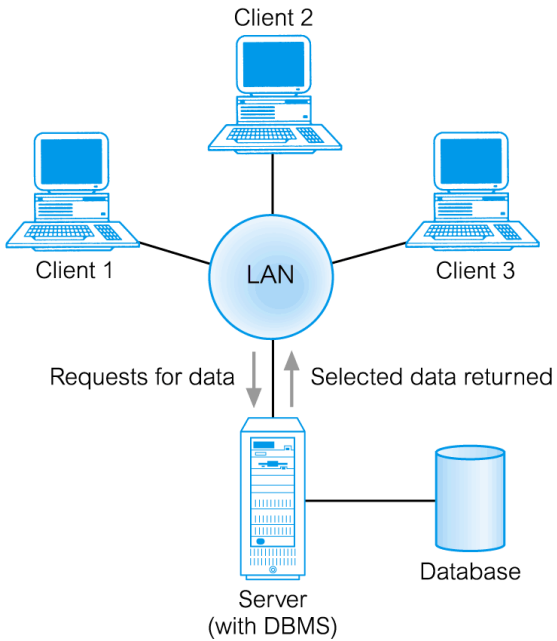
- In a file-server environment, the processing is distributed about the network, typically a local area network (LAN).

Traditional Two-Tier Client-Server

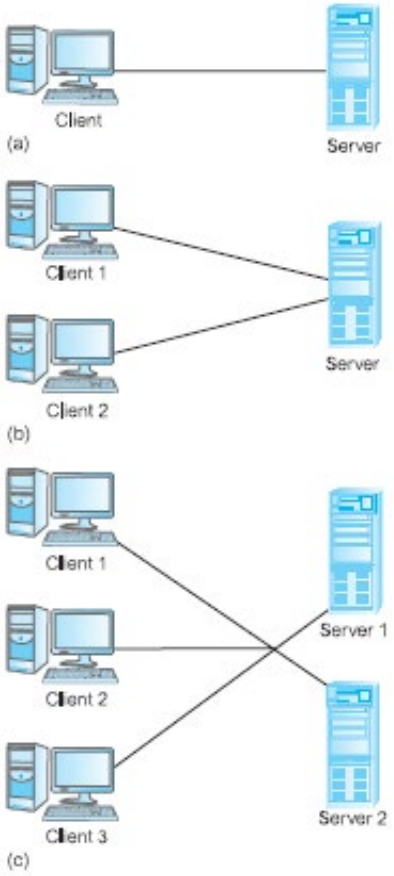
- Client (tier 1) manages user interface and runs applications.
- Server (tier 2) holds database and DBMS.

- Advantages include:
  - wider access to existing databases;
  - increased performance;
  - possible reduction in hardware costs;
  - reduction in communication costs;
  - increased consistency.

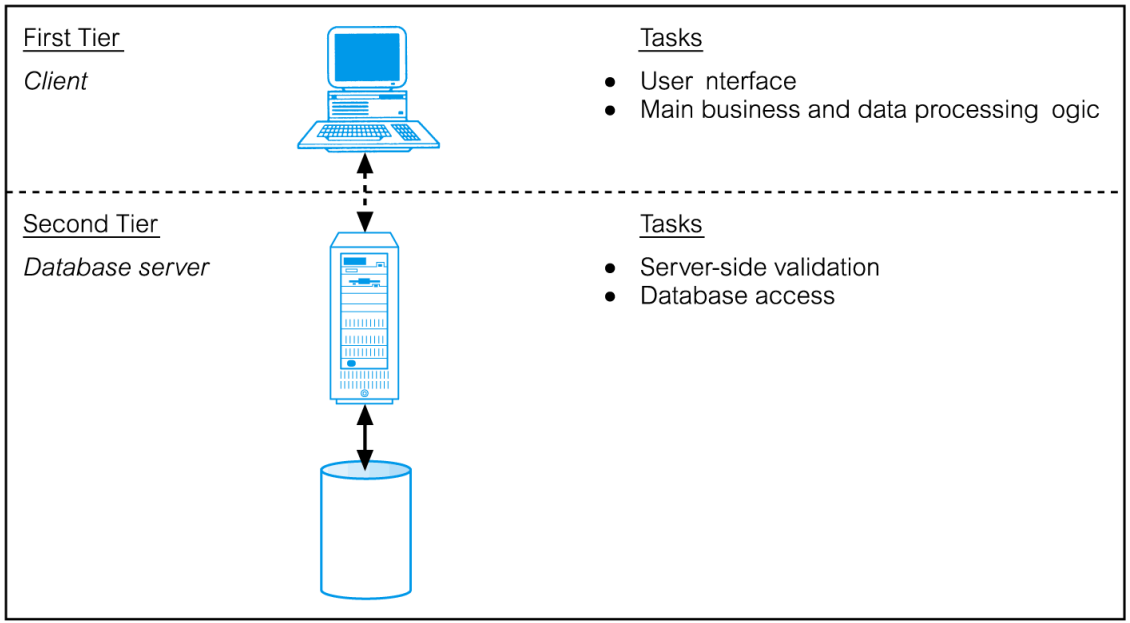
Traditional Two-Tier Client-Server



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.

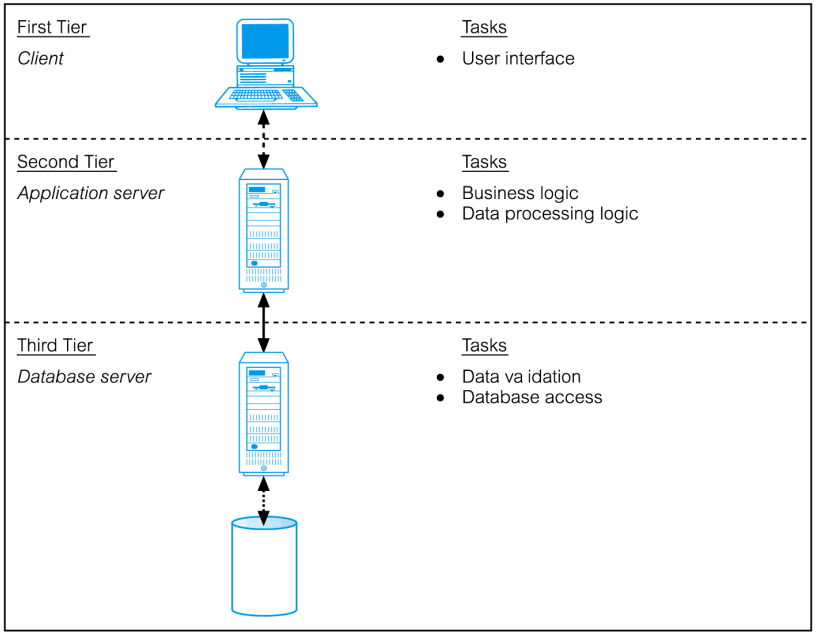
- Client side presented two problems preventing true scalability:
  - ‘Fat’ client, requiring considerable resources on client’s computer to run effectively.
  - Significant client side administration overhead.

- By 1995, three layers proposed, each potentially running on a different platform.

Three-Tier Client-Server

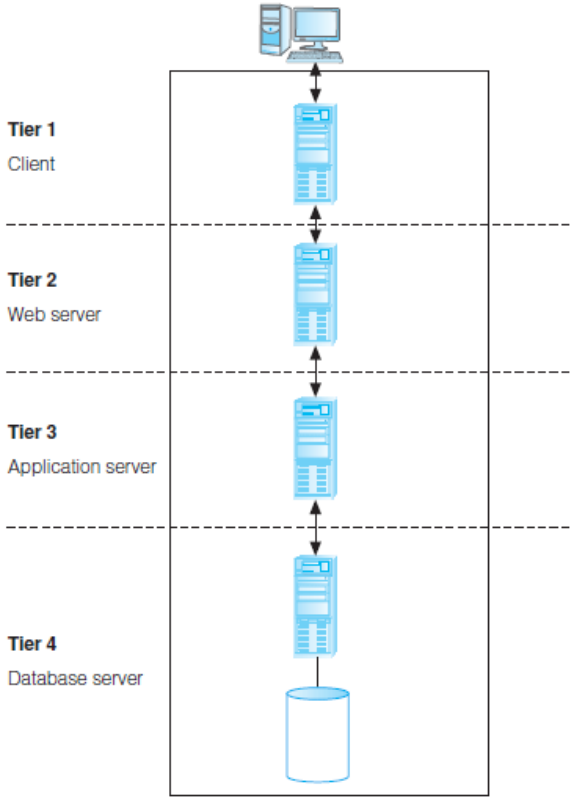
- Advantages:
  - ‘Thin’ client, requiring less expensive hardware.
  - Application maintenance centralized.
  - Easier to modify or replace one tier without affecting others.
  - Separating business logic from database functions makes it easier to implement load balancing.
  - Maps quite naturally to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

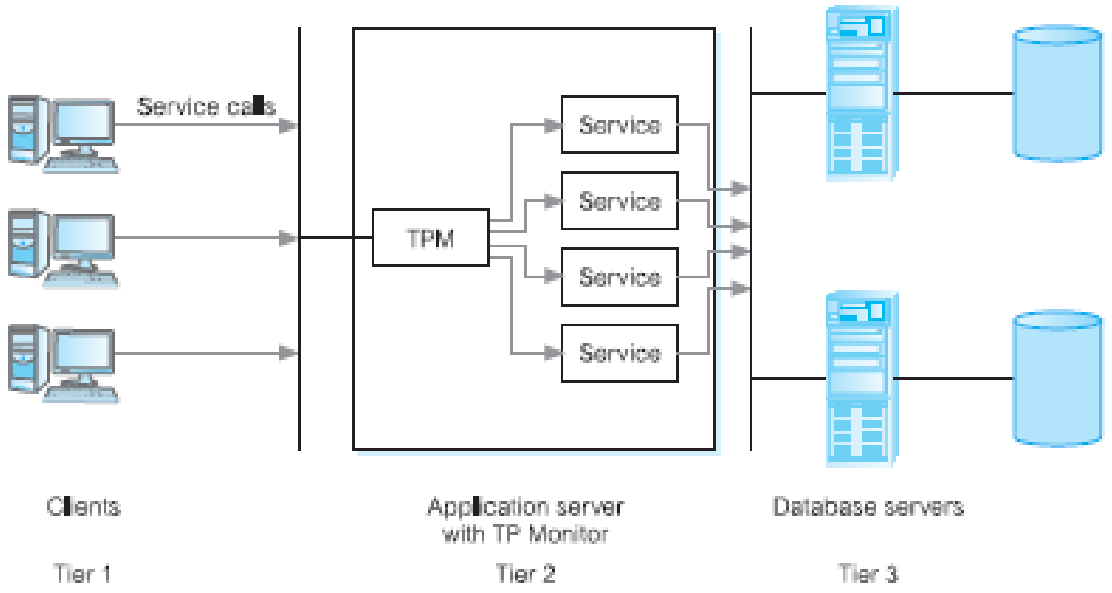
- The National Institute of Standards and Technology (NIST) provided a definition.

- 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 (0LTP).

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



Web Services and Service-Oriented Architectures

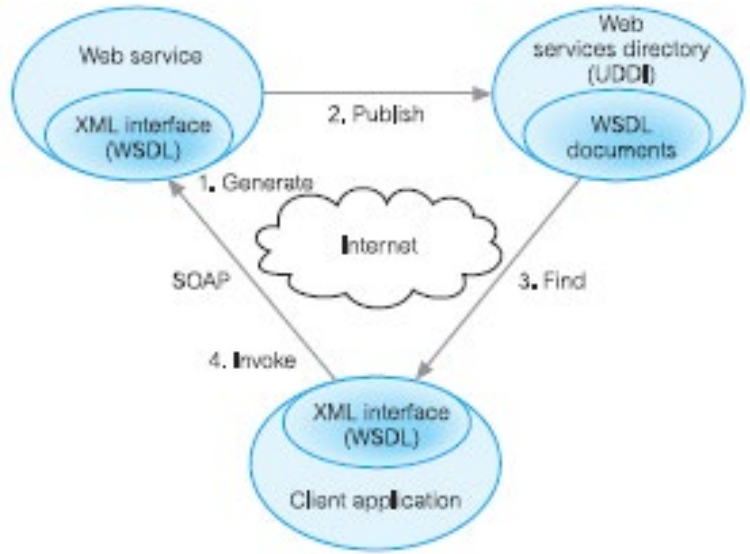
- Web service is a software system designed to support interoperable machine-to-web service machine interaction over a network.
- Web services share business logic, data, and processes through a programmatic interface across a network.
- Developers can add the Web service to a Web page (or an executable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

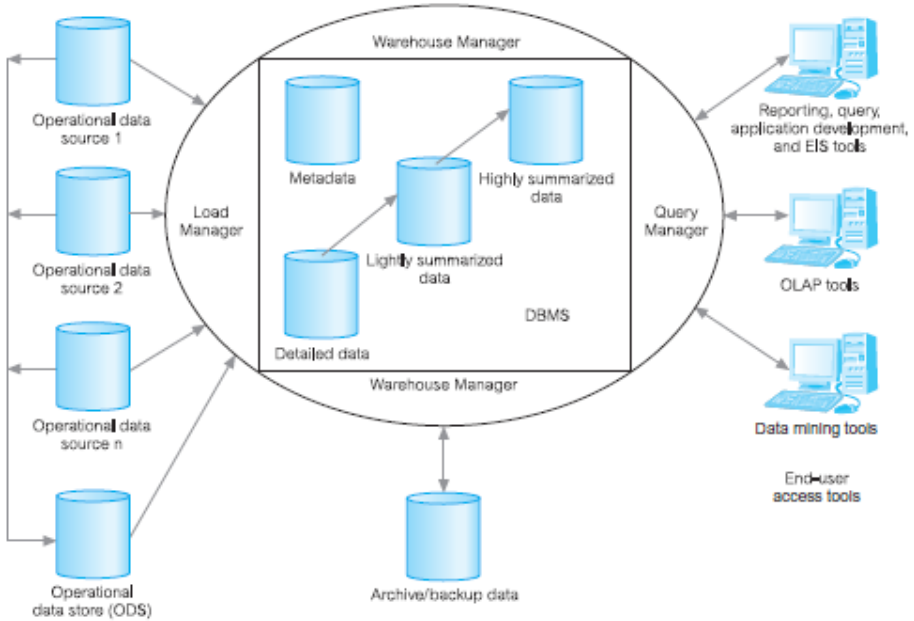
- A distributed DBMS is the software system that permits the management of the distributed database and makes the distribution transparent to users.

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.

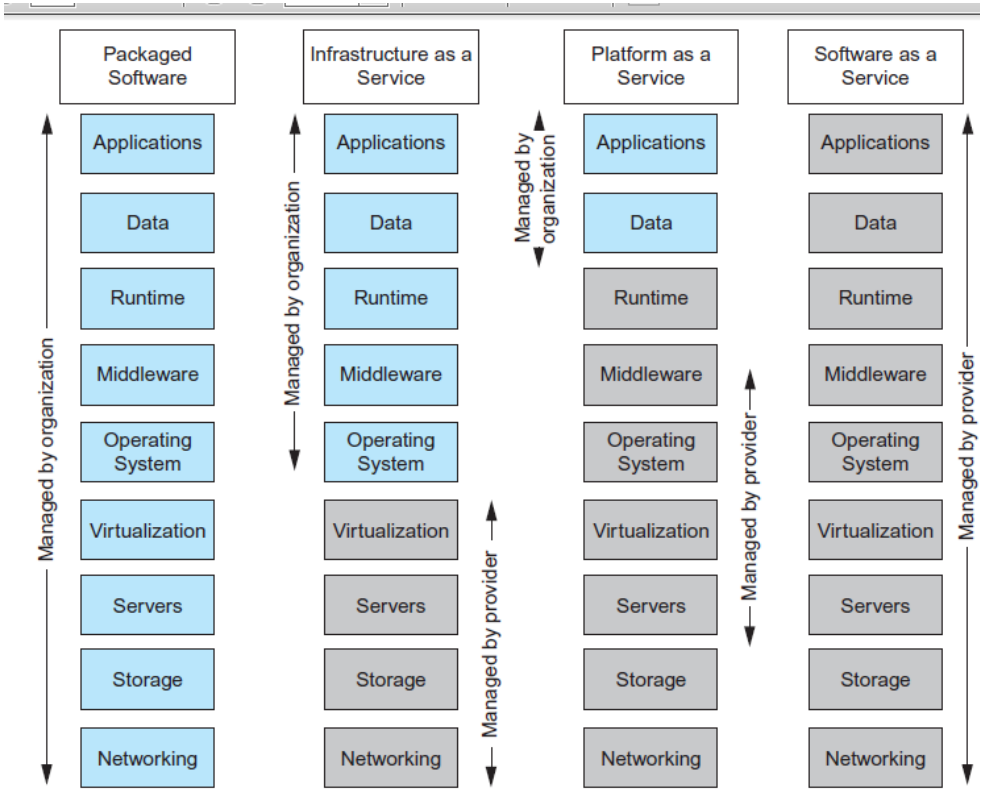
Cloud Computing – Service Models

- 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

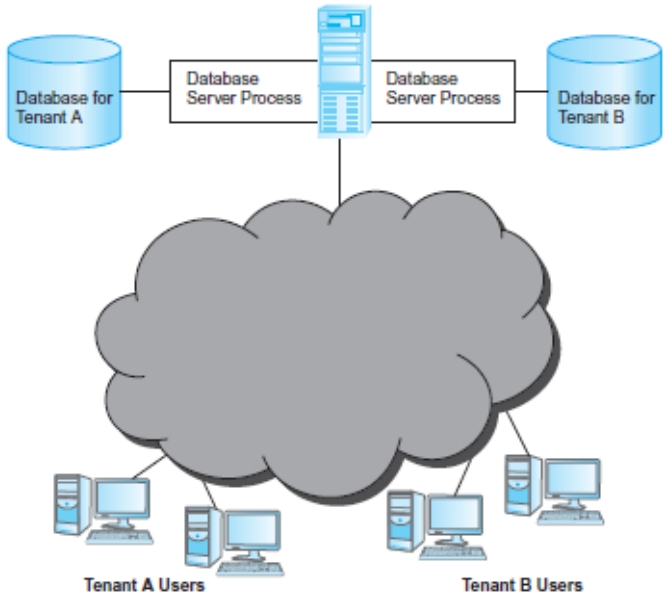
- DBaaS
  - Offers full database functionality to application developers.
  - Provides a management layer that provides continuous monitoring and configuring of the database to optimized scaling, high availability, multi-tenancy (that is, serving multiple client organizations), and effective resource 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.)

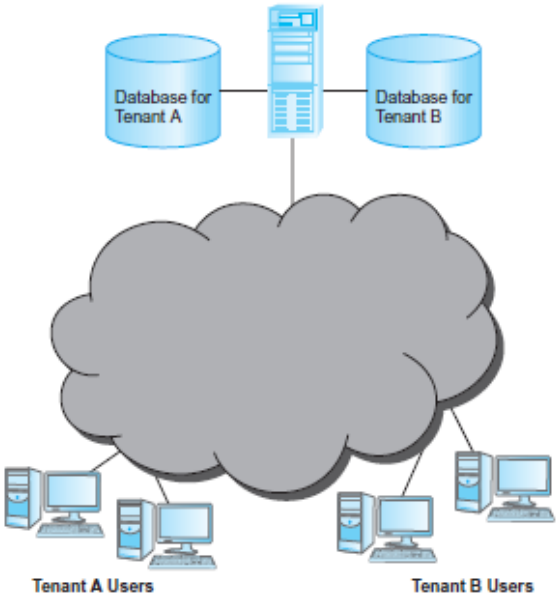
Cloud-based database solutions

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



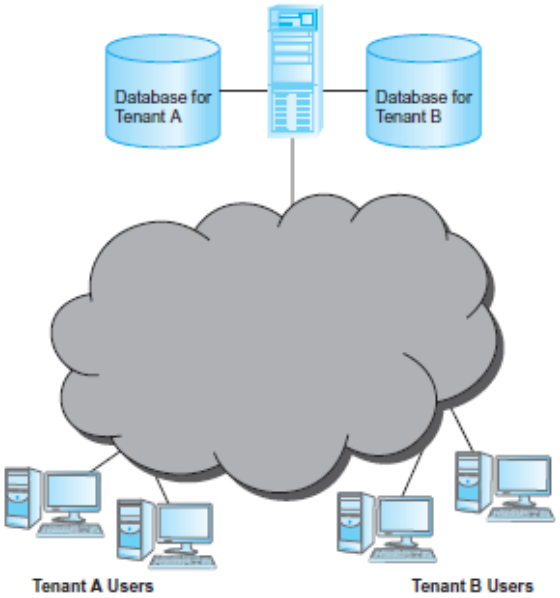
Cloud-based database solutions

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



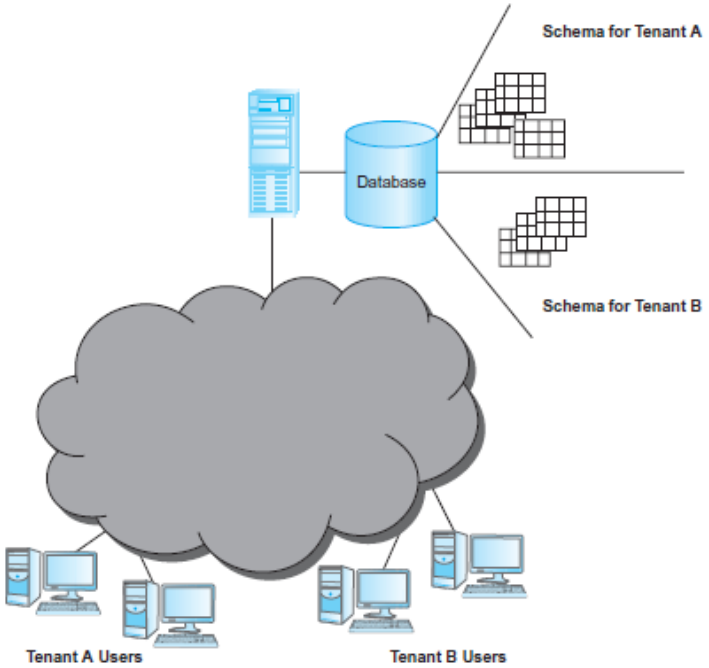
Cloud-based database solutions

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



Cloud-based database solutions

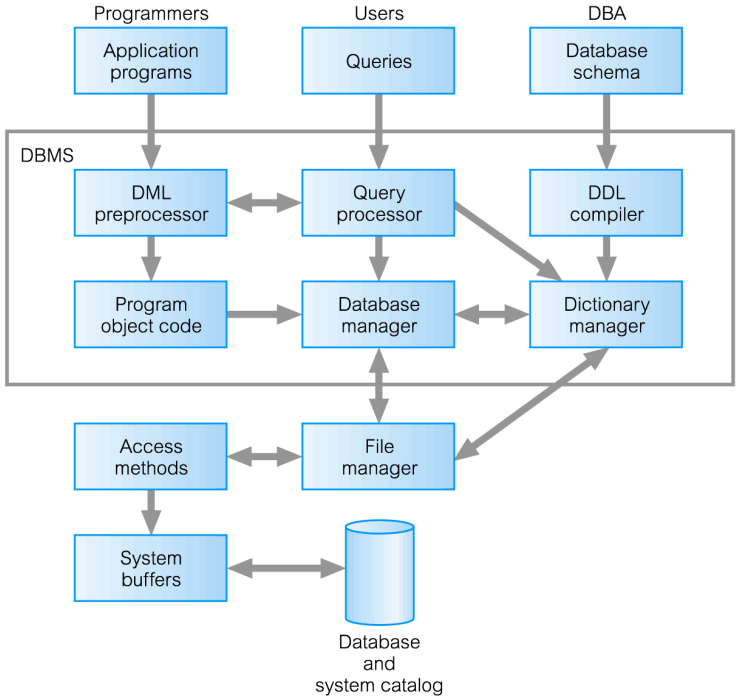
- Multi-tenant cloud database—shared database, separate schema architecture.



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.

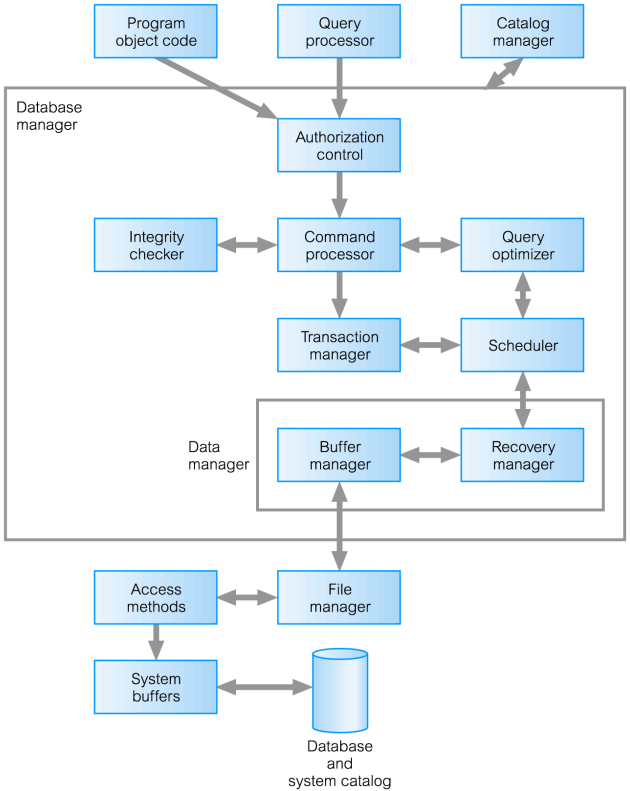
Components of a DBMS (Continued)

- 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.