Chapter 1: Introduction to Databases

A database is a collection of related data.

A database management system (DBMS) refers to the software that manages and controls access to the database.

A database application is simply a program that interacts with the database at some point in its execution.

- **Importance of Database Systems:** Database systems are vital for handling and managing data efficiently in modern organizations. They support critical applications, enhance data sharing, and streamline operations in industries like finance, healthcare, and retail.
- **Common Uses:** Examples include:
    1. **Supermarkets:** Inventory tracking and automatic reorder systems.
    2. **Credit Card Transactions:** Verification of credit limits and prevention of fraud.
    3. **Travel Agencies:** Flight and hotel reservation systems.
    4. **Libraries:** Automated borrowing, cataloguing, and notifications.
    5. **Universities:** Student enrolment, course management, and exam result tracking.
- **File-Based Systems:** A collection of application programs that perform services for the end-users, such as the production of reports. Each program defines and manages its own data.
- **Limitations of File-Based Approach:**
    1. **Separation and Isolation of Data:** Difficult to access and integrate related data.
    2. **Duplication of Data:** Redundant storage leading to inconsistency.
    3. **Data Dependence:** Changes to data format required changes in application code.
    4. **Incompatible File Formats:** Hindered integration between systems.
    5. **Fixed Queries and Application Proliferation:** Limited ability for ad hoc queries and required numerous programs.
- **Database Definition:** A database is a shared collection of logically related data designed to meet the information needs of an organization.
- **DBMS Definition:** A Database Management System (DBMS) is software that defines, creates, maintains, and controls database access.
- **An entity** is a distinct object (a person, place, thing, concept, or event) in the organization that is to be represented in the database.
- **Relationship** is an association between entities

Figure 1.6 shows an Entity– Relationship (ER) diagram for part of the DreamHome case study. It consists of:

• Six entities (the rectangles): Branch, Staff, PropertyForRent, Client, PrivateOwner, and Lease;

•Seven relationships (the names adjacent to the lines): Has, Offers, Oversees, Views, Owns, Leased By, and Holds;

•Six attributes, one for each entity: branchNo, staffNo, propertyNo, clientNo, ownerNo, and leaseNo.

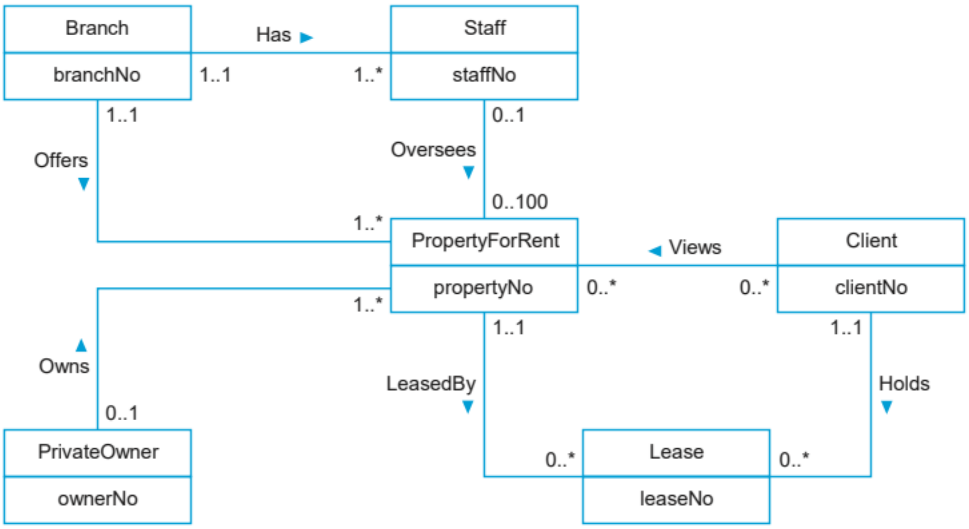


Figure 1.6 example entity–Relationship diagram.

Typically, a DBMS provides the following facilities:

- It allows users to define the database, usually through a Data Definition Language (DDL).
- It allows users to insert, update, delete, and retrieve data from the database, usually through a Data Manipulation Language (DML).
- It provides controlled access to the database. For example, it may provide:
  - a security system, which prevents unauthorized users accessing the database;
  - an integrity system, which maintains the consistency of stored data;
  - a concurrency control system, which allows shared access of the database;
  - a recovery control system, which restores the database to a previous consistent state following a hardware or software failure;
  - a user-accessible catalog, which contains descriptions of the data in the database.

Functions of a DBMS:

- **Data Definition:** Defining data types, structures, and constraints.
- **Data Manipulation:** Retrieval, insertion, updating, and deletion of data.
- **Controlled Access:** Ensuring security, integrity, concurrency, and recovery.
- **Catalog Management:** Maintaining metadata for efficient data handling.

1.3.3 (Database) Application Programs

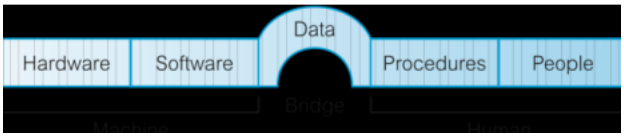
Application programs: A computer program that interacts with the database by issuing an appropriate request (typically an SQL statement) to the DBMS

A database view is a subset of a database and is based on a query that runs on one or more database tables.

Benefits of views in the database:

- ***Views provide a level of security.*** Views can be set up to exclude data that some users should not see.
- ***Views provide a mechanism to customize the appearance of the database***. For example, the Contracts Department may wish to call the monthly rent field (rent) by the more obvious name, Monthly Rent
- ***A view can present a consistent, unchanging picture of the structure of the database,*** even if the underlying database is changed (for example, fields added or removed, relationships changed, files split, restructured, or renamed).

13.4 Components of the DBMS Environment



1. **Hardware:** Computing infrastructure (servers, storage devices). The hardware can range from a single personal computer to a single mainframe or a network of computers.

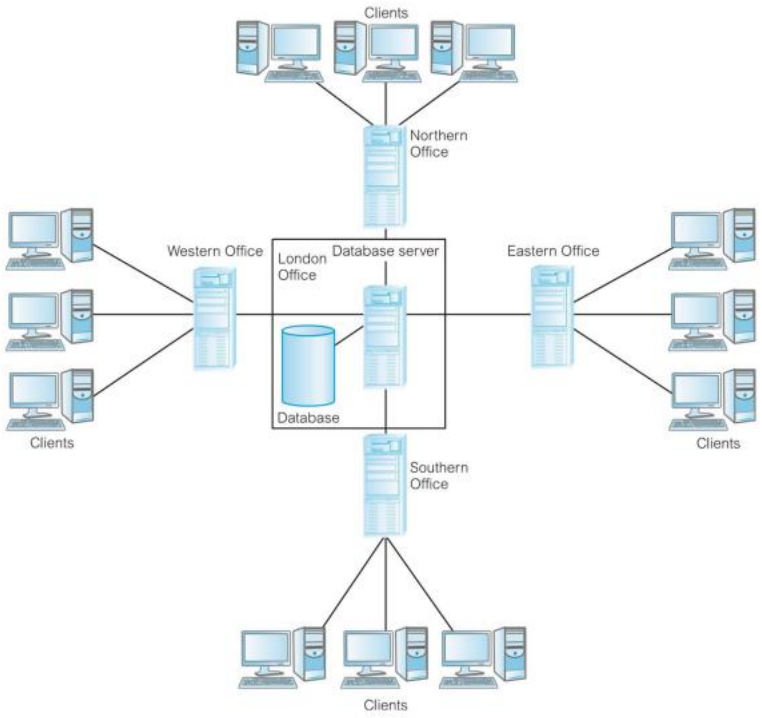


Figure 1.9 *DreamHome* hardware configuration.

2. **Software:** DBMS, applications, and OS. The software component comprises the DBMS software itself and the application programs, together with the operating system, including network software if the DBMS is being used over a network. Typically, application programs are written in a third-generation programming language (3GL), such as C, C++, C#, Java, Visual Basic, COBOL, Fortran, Ada, or Pascal, or a fourth-generation language (4GL), such as SQL, embedded in a third- generation language.

•

1. **Data:** Operational data and metadata.
2. **Procedures:** Rules for managing and using databases. It also refers to Procedures refer to the instructions and rules that govern the design and use of the database.

These may consist of instructions on how to:

• Log on to the DBMS.

• Use a particular DBMS facility or application program.

• Start and stop the DBMS.

• Make backup copies of the database.

• Handle hardware or software failures. This may include procedures on how to identify the failed component, how to fix the failed component (for example, telephone the appropriate hardware engineer), and, following the repair of the fault, how to recover the database.

• Change the structure of a table, reorganize the database across multiple disks, improve performance, or archive data to secondary storage.

3. **People:** Data administrators, DBAs, designers, developers, and end-users. The final component is the people involved with the system

1.3.5 Roles in the DBMS Environment

1.Database administrator

- **Data Administrator (DA):** Manages organizational data policies.

Roles of a database administrator

 Physical Realization of the Database*:* The DBA is responsible for the physical design and implementation of the database, ensuring it functions effectively within the organizational framework.

 Security and Integrity Control: The DBA manages security measures to protect data integrity and prevent unauthorized access, which is critical for maintaining trust and compliance.

 Operational Maintenance: The DBA oversees the ongoing maintenance of the operational database system, ensuring it remains functional and efficient for users.

 Performance Optimization: The DBA ensures satisfactory performance of applications, which is vital for user satisfaction and operational efficiency.

 Technical Expertise: The DBA possesses detailed knowledge of the target DBMS and the system environment, enabling them to troubleshoot issues and implement best practices effectively.

2. Database Designers

1. Logical Database Designers: identify data entities, attributes, relationships, and constraints within a database, focusing on a comprehensive understanding of the organization's data.

Role:

- Involve prospective database users early in developing the data model.

Conduct two stages:

- **Conceptual Database Design**: High-level data organization, independent of implementation details.
- **Logical Database Design**: Specific data model targeting (e.g., relational, hierarchical).

2. Physical Database Designers: Focus on efficient data storage and access strategies. Responsible for implementing the logical database design into a physical structure.

ROLES OF Physical Database Designers

- Map logical designs into tables and integrity constraints.
- Select storage structures and access methods to ensure optimal performance.
- Design security measures for data protection.

3. Application Developers

Application developers are responsible for implementing application programs that provide the required functionality for end-users once the database is established.
 Role:

- Work from specifications created by systems analysts to develop application software.
- Write program statements that interact with the Database Management System (DBMS) to perform operations such as retrieving, inserting, updating, and deleting data.
- Utilize third-generation or fourth-generation programming languages for development.

4. End-Users

- **Definition**: Clients of the database designed and implemented to meet their information needs.

1. Naïve Users

- **Definition**: Users who are typically unaware of the underlying Database Management System (DBMS).
- **Role**:
  - Access the database through specially designed application programs that simplify operations.
  - Perform tasks by entering simple commands or selecting options from menus, without needing knowledge of the database structure.
  - Example: A checkout assistant using a barcode reader to retrieve item prices from the database without understanding the DBMS.

2. Sophisticated Users

- **Definition**: Users who are familiar with the database structure and the features of the DBMS.
- **Role**:
    - Utilize high-level query languages, such as SQL, to perform operations on the database.
    - May write their own application programs for specific tasks, leveraging their understanding of the system.

- **History of DBMS Development:**

TIMeFRaMe	DeVelOPMeNT	COMMeNTS
1960s (onwards)	File-based systems	Precursor to the database system. Decentralized approach: each department stored and controlled its own data.
Mid-1960s	hierarchical and network data models	Represents first-generation DBMSs. Main hierarchical system is IMS from IBM and the main network system is IDMS/R from Computer Associates. Lacked data independence and required complex programs to be developed to process the data.
1970	Relational model proposed	Publication of e. F. Codd's seminal paper "A relational model of data for large shared data banks," which addresses the weaknesses of first-generation systems.
1970s	Prototype RDBMSs developed	During this period, two main prototypes emerged: the Ingres project at the University of California at Berkeley (started in 1970) and the System R project at IBM's San José Research Laboratory in California (started in 1974), which led to the development of SQL.
1976	eR model proposed	Publication of Chen's paper "The entity-Relationship model—Toward a unified view of data." eR modeling becomes a significant component in methodologies for database design.
1979	Commercial RDBMSs appear	Commercial RDBMSs like Oracle, Ingres, and DB2 appear. These represent the second generation of DBMSs.
1987	ISO SQL standard	SQL is standardized by the ISO (International Standards Organization). There are subsequent releases of the standard in 1989, 1992 (SQL2), 1999 (SQL:1999), 2003 (SQL:2003), 2008 (SQL:2008), and 2011 (SQL:2011).
1990s	OODBMS and ORDBMSs appear	This period initially sees the emergence of OODBMSs and later ORDBMSs (Oracle 8, with object features released in 1997).
1990s	Data warehousing systems appear	This period also see releases from the major DBMS vendors of data warehousing systems and thereafter data mining products.
Mid-1990s	Web–database integration	The first Internet database applications appear. DBMS vendors and third-party vendors recognize the significance of the Internet and support web–database integration.
1998	XML	XML 1.0 ratified by the W3C. XML becomes integrated with DBMS products and native XML databases are developed.

Figure 1.10 historical development of database systems.

- **Advantages of DBMS:**
    1. Control of data redundancy.
    2. Enhanced data consistency.
    3. Improved data sharing and accessibility.
    4. Better security and integrity.
    5. Easier maintenance and independence from physical data changes.
    6. Backup and recovery support.

TaBle 1.2 Advantages of DBMSs.

Control of data redundancy	economy of scale
Data consistency	Balance of conflicting requirements
More information from the same amount of data	Improved data accessibility and responsiveness
Sharing of data	Increased productivity
Improved data integrity	Improved maintenance through data independence
Improved security	Increased concurrency
enforcement of standards	Improved backup and recovery services

- **Disadvantages of DBMS:**
    1. Complexity and higher costs.
    2. Performance issues for small applications.
    3. Greater vulnerability due to centralization.

Chapter 2: Database Environment

A major aim of a database system is to provide users with an abstract view of data, hiding certain details of how data is stored and manipulated.

2.1 The Three-Level ANSI-SPARC Architecture

Database architecture refers to the design, structure, and organization of a database system. It defines how data is stored, accessed, managed, and processed within a system to meet various requirements for efficiency, reliability, and scalability.

The levels form a three-level architecture comprising an **external,** a **conceptual**, and an **internal level.**

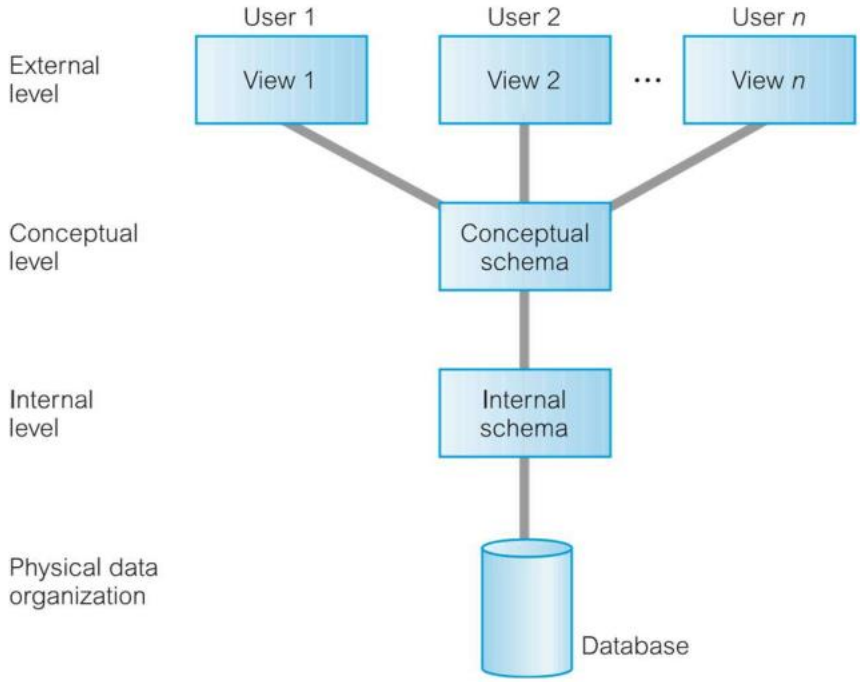
The way the DBMS and the operating system perceive the data is at the **internal level.**

2.1. The Three-Level ANSI-SPARC Architecture

• The DBA should be able to change the database storage structures without affecting the users' views.

• The internal structure of the database should be unaffected by changes to the physical aspects of storage, such as the changeover to a new storage device.

• The DBA should be able to change the conceptual structure of the database without affecting all users



2.1.1 External Level

External level	The users' view of the database. This level describes that part of the database that is relevant to each user.

The external view includes only those entities, attributes, and relationships in the "real world" that the user is interested in. Other entities, attributes, or relationships that are not of interest may be represented in the database, but the user will be unaware of them.

2.1.2 Conceptual Level

Conceptual level	The community view of the database. This level describes *what* data is stored in the database and the relationships among the data.

The middle level in the three-level architecture is the conceptual level. This level contains the logical structure of the entire database as seen by the DBA. It is a complete view of the data requirements of the organization that is independent of any storage considerations. The conceptual level represents:

• All entities, their attributes, and their relationships;

• The constraints on the data;

• Semantic information about the data;

2.1.3 Internal Level

Internal level	The physical representation of the database on the computer. This level describes *how* the data is stored in the database.

The internal level covers the physical implementation of the database to achieve optimal runtime performance and storage space utilization. It covers the data structures and file organizations used to store data on storage devices. It interfaces with the operating system access methods (file management techniques for storing and retrieving data records) to place the data on the storage devices, build the indexes, retrieve the data, and so on. The internal level is concerned with such things as:

• Storage space allocation for data and indexes;

• Record descriptions for storage (with stored sizes for data items);

• Record placement;

• Data compression and data encryption techniques

2.1.4 Schemas, Mappings, and Instances

The overall description of the database is called the database schema.

1. **Schema Levels:**
    - **External Schema:** Represents different views of data tailored to user needs.
        - Also called subschemas.
        - Multiple external schemas exist per database.
    - **Conceptual Schema:** Describes the entire database structure, including entities, attributes, relationships, and integrity constraints.
        - Only one conceptual schema per database.
    - **Internal Schema:** Provides a complete description of stored records, representation methods, data fields, indexes, and storage structures.
        - Only one internal schema per database.
2. **Mapping Between Schemas:**
    - The **DBMS** is responsible for mapping between external, conceptual, and internal schemas.
    - The mappings include:
        - **Conceptual/Internal Mapping:**
            - Maps logical records to physical storage.
            - Resolves differences such as entity names, attribute names, data types, and constraints.
        - **External/Conceptual Mapping:**

- Maps names in the user's external view to the conceptual schema.

3. **Schema Consistency:**
   - The DBMS checks schema consistency to ensure each external schema is derivable from the conceptual schema.

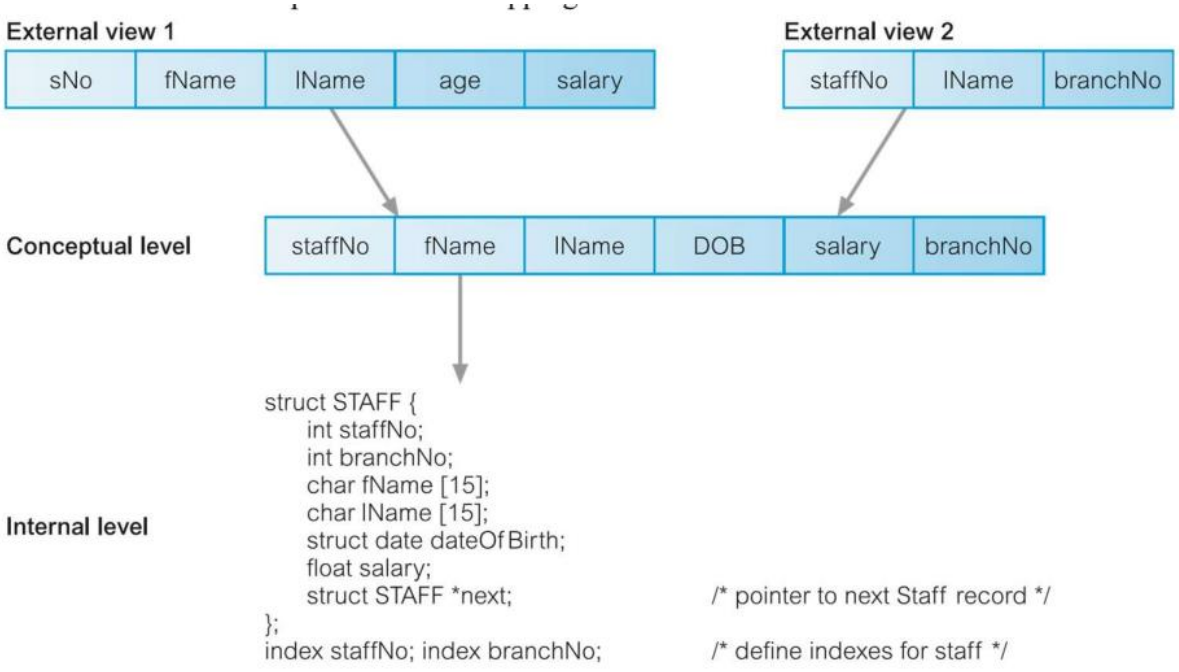


Figure 2.2 Differences between the three levels.

The data in the database at any particular point in time is called a **database instance**. Therefore, many **database instances** can correspond to the same database schema**. The schema** is sometimes called the **intension** of the database; an instance is called **an extension (or state)** of the database

2.1.5 Data Independence

Definition:
Data independence is the ability to modify a database schema at one level without affecting the schema at the next higher level. It ensures that changes in the storage or logical structure of data do not impact the applications or user views accessing the data.

Types of Data Independence:

1. **Logical Data Independence:**
   - Refers to the ability to change the **conceptual schema** without affecting the external schema or application programs.
   - Example Changes:
     - Adding new attributes or entities
     - Modifying relationships between entities
     - Removing fields

- Importance: Keeps user views and applications stable when changes occur in the logical structure.

2. **Physical Data Independence:**
   - Refers to the ability to change the **internal schema** (physical storage structure) without affecting the conceptual schema.
   - Example Changes:
     - Changing data storage format
     - Modifying access methods (e.g., adding indexes)
     - Partitioning tables
   - Importance: Allows optimization of storage and performance without disturbing the logical design.

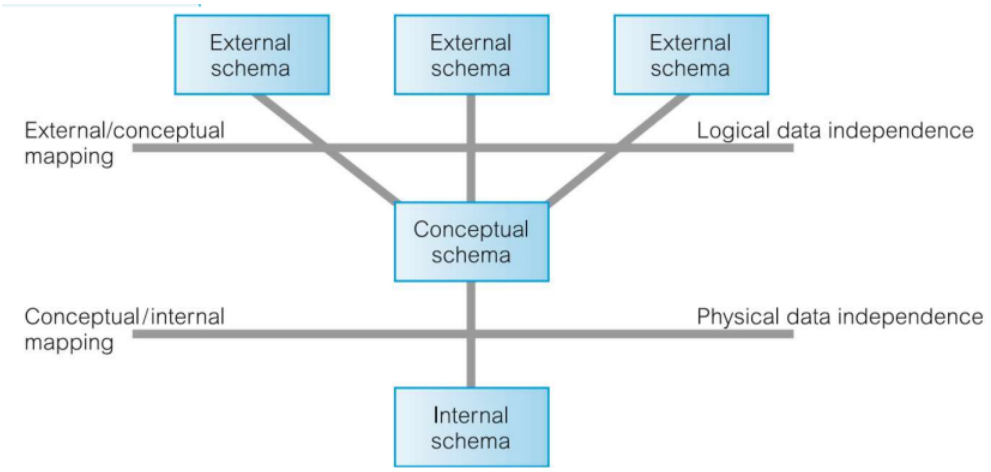


Figure 2.3
Data independence and the ANSI-SPARC three-level architecture.

Why Data Independence Matters:

- **Application Stability:** Reduces the need to rewrite applications when the database structure changes.
- **Maintenance Efficiency:** Simplifies maintenance and evolution of databases.
- **System Flexibility:** Enhances adaptability to new storage technologies or performance optimizations.

2.2 Database Languages

Database Languages with Definitions and Examples

1. **Data Definition Language (DDL)**
   - **Definition:** DDL is used to define, create, and manage the structure of database objects such as tables, indexes, and views.
   - **Common Commands:**
     - CREATE: Defines new database objects.
     - ALTER: Modifies existing structures.
     - DROP: Deletes database objects.
   - **Example:**

```
CREATE TABLE Students (
  ID INT PRIMARY KEY,
  Name VARCHAR(50)
);
```

2. **Data Manipulation Language (DML)**
   - **Definition:** DML is used to query, insert, update, and delete data stored in the database.
   - **Common Commands:**
     - SELECT: Retrieves data from tables.
     - INSERT: Adds new records.
     - UPDATE: Modifies existing data.
     - DELETE: Removes data.
   - **Example:**
   - INSERT INTO Students (ID, Name) VALUES (1, 'Olivier');

3. **Data Control Language (DCL)**
   - **Definition:** DCL is used to control access to the database by granting or revoking permissions for users.
   - **Common Commands:**
     - GRANT: Provides specific privileges.
     - REVOKE: Removes granted privileges.
   - **Example:**
   - GRANT SELECT ON Students TO user1;

4. **Transaction Control Language (TCL)**
   - **Definition:** TCL is used to manage and control transactions within a database, ensuring data consistency.
   - **Common Commands:**
     - COMMIT: Saves all changes made during the transaction.
     - ROLLBACK: Reverts changes to the previous state.
     - SAVEPOINT: Sets points within a transaction to partially roll back changes.
   - **Example:**
   - BEGIN;
   - UPDATE Students SET Name = 'Blessking' WHERE ID = 1;
   - COMMIT;

Fourth-Generation Languages (4GLs)

Definition:

4GLs are high-level, non-procedural languages that focus on what needs to be done rather than how to do it, requiring fewer lines of code compared to 3GLs.

Components:

- **Presentation Languages:** Query languages (e.g., SQL), report generators.

- **Specialty Languages:** Spreadsheets, database languages.
- **Application Generators:** Tools for building database applications.
- **Very High-Level Languages:** Generate application code.

Types of Generators:

- **Forms Generators:** Create input/display screens.
- **Report Generators:** Produce custom database reports.
- **Graphics Generators:** Visualize data as charts.
- **Application Generators:** Automate application creation using prebuilt modules.

Advantages:

- Faster development and improved productivity.
- Easier programming for non-technical users.

Disadvantages:

- Limited flexibility and problem-solving capabilities.

1. Data Models

A data model refers to An integrated collection of concepts for describing and manipulating data, relationships between data, and constraints on the data in an organization.

- **Purpose of a Data Model:**
  - Provides a higher-level description of an organization's data requirements.
  - Acts as an integrated collection of concepts for:
    - Describing data.
    - Manipulating data.
    - Defining relationships between data.
    - Enforcing constraints on the data.
  - Serves as a bridge between the low-level Data Definition Language (DDL) used by a DBMS and the business/user perspective.
- **Core Components of a Data Model:**
  - **Structural Component:**
    A set of rules for constructing databases.
  - **Manipulative Component:**
    Defines the operations that can be performed on the data (e.g., retrieval, updating, modifying the structure).
  - **Integrity Constraints:**
    Rules to ensure the accuracy and validity of the data.

2. Levels of Data Models (Reflecting the ANSI-SPARC Architecture)

- **External Data Model:**

Represents each user's view of the organization (also known as the Universe of Discourse or UoD).

- **Conceptual Data Model:**
  - Represents the logical (or community) view of the entire organization.
  - Is DBMS-independent.
- **Internal Data Model:**
  - Represents the conceptual schema in a form that the DBMS can understand.
  - Deals with physical storage details.

---

3. Categories of Data Models

A. Object-Based Data Models

- **Key Concepts:**
  - **Entities:** Distinct objects such as people, places, or events.
  - **Attributes:** Properties that describe entities.
  - **Relationships:** Associations between entities.
- **Common Types:**
  - **Entity-Relationship (ER) Model:** Widely used for database design.
  - **Semantic Model**
  - **Functional Model**
  - **Object-Oriented Model:**
    - Extends the entity concept to include both state (attributes) and behavior (actions/methods).

B. Record-Based Data Models

- **Structure:**
  - Composed of fixed-format records (each record type defines a fixed number of fields, often with fixed lengths).
- **Principal Types:**
  1. **Relational Data Model**
     - Data is represented in tables.
     - Relationships between tables may be implicit (e.g., matching key attributes).
     - Supports a declarative approach to querying (specifies *what* data is needed, not *how* to retrieve it).
  2. **Network Data Model:**
     1. Data is represented as collections of records with explicit relationships via sets.
     2. Uses a generalized graph structure where records are nodes and sets (pointers) are edges.
  3. **Hierarchical Data Model:**
     1. A special case of the network model.
     2. Represents data in a tree structure (each record/node has only one parent).
     3. More restrictive than the network model.

C. Physical Data Models

- **Focus:**
  - Describe how data is stored and accessed physically on the computer.
  - Include details such as record structures, record orderings, and access paths.
- **Examples:**
  - The unifying model.
  - The frame memory.

4. Conceptual Modeling

- **Definition:**
  - The process of constructing a model of the enterprise's information needs without getting into implementation details.
  - This model is known as the **conceptual data model**.
- **Importance of the Conceptual Schema:**
  - It is considered the heart of the database.
  - Must accurately represent all the data requirements of the enterprise.
  - Any omissions or inaccuracies can lead to missing or incorrectly represented information, affecting external views and overall database integrity.
- **Distinction Between Models:**
  - **Conceptual Data Model:**
    - Independent of implementation details (e.g., DBMS, programming languages).
  - **Logical Data Model:**
    - Refines the conceptual model with knowledge of the target DBMS's underlying data model (e.g., relational model).
- **Design Methodology:**
  - Start by producing a conceptual data model.
  - Refine this into a logical model that aligns with the target DBMS (commonly the relational data model).

2.4 Functions of a DBMS

Below is a concise summary of the key functions of a DBMS:

1. **Data Storage, Retrieval, and Update:**
   - Provides the basic capability to store, retrieve, and update data.
   - Hides the physical details of data storage from the user.
2. **User-Accessible Catalog (Data Dictionary/System Catalog):**
   - Maintains metadata about data items, relationships, schemas, and user access rights.
   - Supports centralized control, consistency, security, and impact analysis for changes.
3. **Transaction Support:**
   - Ensures that all changes within a transaction are either fully completed or fully undone (atomicity).
   - Prevents inconsistencies in case of failures or system crashes.

4. **Concurrency Control:**
    - Manages simultaneous access by multiple users.
    - Prevents interference and maintains consistency when users update the database concurrently.
5. **Recovery Services:**
    - Provides mechanisms to restore the database to a consistent state after a failure (e.g., crash, hardware errors).
6. **Authorization Services:**
    - Controls access so that only authorized users can view or modify the data.
    - Enforces security by restricting access to sensitive information.
7. **Support for Data Communication:**
    - Integrates with communication software to handle data requests and responses, especially in distributed environments.
    - Ensures remote and networked users can access a centralized database.
8. **Integrity Services:**
    - Enforces data quality by ensuring that all data and modifications comply with predefined rules (constraints).
    - Helps maintain the overall correctness and consistency of the database.
9. **Services to Promote Data Independence:**
    - Provides features (such as views) to shield applications from changes in the physical or logical structure of the database.
    - Aims to minimize the impact of changes to the database schema on application programs.
10. **Utility Services:**
    - Offers additional tools and utilities for database administration, such as import/export tools, performance monitoring, index reorganization, and garbage collection.
    - Enhances the overall manageability and performance of the DBMS.

Chapter 3 Database Architectures and the Web

3.1 Multi-user DBMS Architectures

3.1.1 Teleprocessing:

- Centralized architecture with a single CPU and dumb terminals.
- High burden on the central computer.

3.1.2 File-Server Architecture:

File server	A computer attached to a network with the primary purpose of providing shared storage for computer files such as documents, spreadsheets, images, and databases.

- Files are stored on a file server, but processing occurs on individual workstations.

- Disadvantages: High network traffic, full DBMS required on each workstation, complex concurrency control.

3.1.3 Traditional Two-Tier Client–Server Architecture

- **Two-tier**: Client handles presentation and business logic; server manages data.

TABLE 3.1 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

3.1.4 **Three-tier**: Adds an **application server** between the client and the database server.

This new architecture proposed three layers, each potentially running on a different platform:

(1) The user interface layer, which runs on the end-user's computer (the client).

(2) The business logic and data processing layer. This middle tier runs on a server and is often called the application server.

(3) A DBMS, which stores the data required by the middle tier. This tier may run on a separate server called the database server.

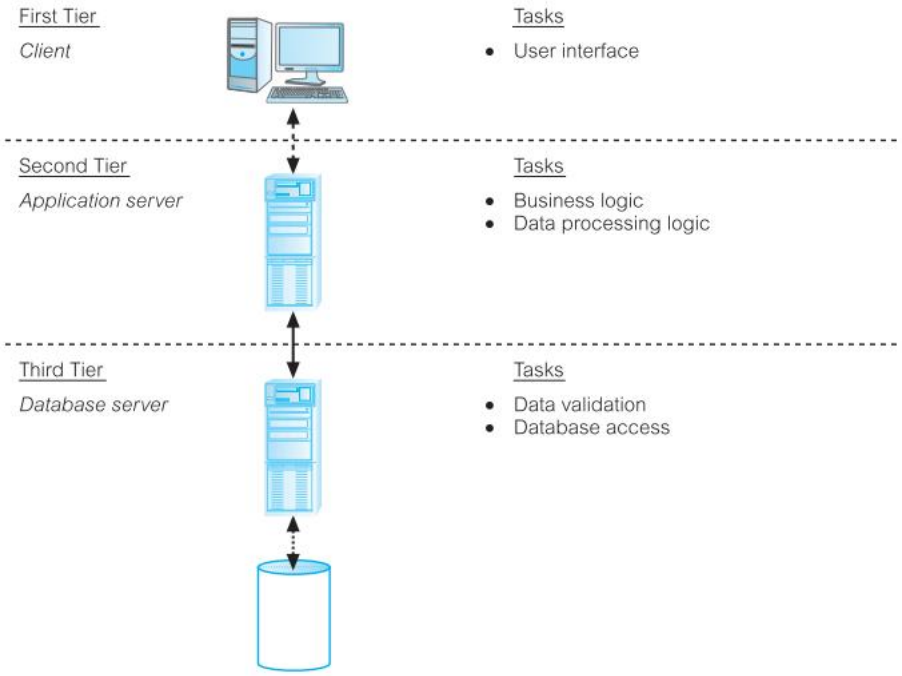
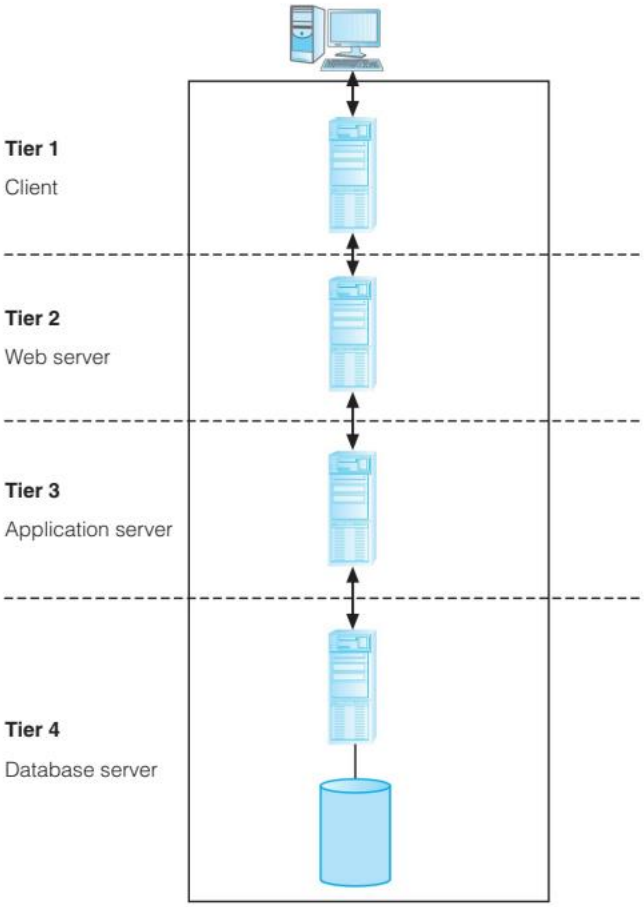


Fig: **3 tier architecture**

3.1.5 **N-tier**: Further divides the middle tier for scalability (e.g., web server and application server).

The three-tier architecture can be expanded to n tiers, with additional tiers providing more flexibility and scalability.



Application servers

Application server	Hosts an application programming interface (API) to expose business logic and business processes for use by other applications.

An application server must handle several complex issues:

- concurrency;
- network connection management;
- providing access to all the database servers;
- database connection pooling;
- legacy database support;
- clustering support;
- load balancing;
- Failover.

3.1.6 **Middleware**:

Middleware	Computer software that connects software components or applications.

Types of Middleware (Hurwitz, 1998)

1. **Asynchronous Remote Procedure Call (RPC)**:
    - Allows a client to request a service from a remote server without waiting for a response.
    - **Advantages**: Highly scalable.
    - **Disadvantages**: Low recoverability (client must restart if the connection breaks).
    - **Use Case**: Suitable when transaction integrity is not required.
2. **Synchronous RPC**:
    - Similar to asynchronous RPC, but the client is blocked until the server completes the request.
    - **Advantages**: High recoverability.
    - **Disadvantages**: Less scalable.
    - **Examples**: Java RMI, XML-RPC, Microsoft .NET Remoting, CORBA, Thrift.
3. **Publish/Subscribe**:
    - Asynchronous messaging protocol where subscribers receive messages from publishers based on their interests.
    - **Advantages**: Decouples publishers and subscribers, enabling scalability and dynamic network topologies.
    - **Examples**: TIBCO Rendezvous, Ice (Internet Communications Engine).
4. **Message-Oriented Middleware (MOM)**:
    - Supports asynchronous communication between client and server applications using message queues.
    - **Advantages**: Ensures message delivery even if the destination application is busy or disconnected.
    - **Examples**: IBM WebSphere MQ, Microsoft MSMQ, Java Messaging Service (JMS), Oracle MessageQ.
5. **Object-Request Broker (ORB)**:
    - Manages communication and data exchange between objects in distributed systems.
    - **Advantages**: Promotes interoperability between objects from different vendors.
    - **Examples**: CORBA (Common Object Requesting Broker Architecture), Orbix.
6. **SQL-Oriented Data Access**:
    - Connects applications to databases and translates SQL requests into the database's native language.
    - **Advantages**: Eliminates the need for database-specific coding.
    - **Examples**: ODBC (Open Database Connectivity), JDBC (Java Database Connectivity), database gateways.

3.1.7 Transaction Processing Monitors

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

Monitors provide significant advantages, including:

- **Transaction routing**: The TP Monitor can increase scalability by directing transactions to specific DBMSs.

• **Managing distributed transactions**: The TP Monitor can manage transactions that require access to data held in multiple, possibly heterogeneous, DBMSs.

• **Load balancing:** The TP Monitor can balance client requests across multiple DBMSs on one or more computers by directing client service calls to the least loaded server.

• **Funnelling**: In environments with a large number of users, it may sometimes be difficult for all users to be logged on simultaneously to the DBMS.

• **Increased reliability:** The TP Monitor acts as a transaction manager, performing the necessary actions to maintain the consistency of the database, with the DBMS acting as a resource manager.

3.2 Web Services and Service-Oriented Architectures (SOA)

3.21 Web Services:

Web service	A software system designed to support interoperable machine-to-machine interaction over a network.

Web services enable interoperable machine-to-machine interactions over a network. They allow applications to integrate across the Internet, supporting both B2C (Business to Consumer) and B2B (Business to Business) interactions.

Key Features of Web Services:

- **No User Interface**: Web services share business logic, data, and processes through a programmatic interface, not via web browsers.
- **Examples**:
  - Microsoft Bing Maps & Google Maps: Access to location-based services like maps, driving directions, geocoding, and reverse geocoding.
  - Amazon S3: Simple Web service interface for storing and retrieving data.
  - Geonames: Location-related services such as returning Wikipedia entries and time zones.

- DOTS: Services like company information, reverse telephone number lookup, email address validation, weather info, and IP address-to-location determination.
- Xignite: Financial information services, including US equities data and financial news.

Technologies and Standards:

- **XML (Extensible Markup Language)**: Fundamental for data representation.
- **SOAP (Simple Object Access Protocol)**: Platform- and language-independent protocol for exchanging structured information.
- **WSDL (Web Services Description Language)**: XML-based protocol for describing and locating web services.
- **UDDI (Universal Discovery, Description, and Integration)**: XML-based registry for listing web services on the Internet, accessible via SOAP.

RESTful Web Services:

- Focuses on Representational State Transfer (REST) rather than SOAP-based communications.
- REST services use simple operations (PUT, GET, POST, DELETE) to manage resources identified by Uniform Resource Identifiers (URIs).
- REST adopts a client-server architecture, typically using HTTP, for stateless communication.

3.2.2 Service-Oriented Architectures (SOA)

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.

The following are a set of common SOA principles that provide a unique design approach for building Web services for SOA:

• **Loose coupling:** Services must be designed to interact on a loosely coupled basis;

• **Reusability:** Logic that can potentially be reused is designed as a separate service;

• **Contract:** Services adhere to a communications contract that defines the information exchange and any additional service description information, specified by one or more service description documents;

• **Abstraction**: Beyond what is described in the service contract, services hide logic from the outside world;

Composability: Services may compose other services, so that logic can be repre- sented at different levels of granularity thereby promoting reusability and the creation of abstraction layers;

• **Autonomy:** Services have control over the logic they encapsulate and are not dependent upon other services to execute this governance;

• **Stateless:** Services should not be required to manage state information, as this can affect their ability to remain loosely-coupled;

- **Discoverability**: Services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms.

3.3 Distributed DBMSs

Distributed database	A logically interrelated collection of shared data (and a description of this data), physically distributed over a computer network.
Distributed DBMS	The software system that permits the management of the distributed database and makes the distribution transparent to users.

A distributed database management system (DDBMS) consists of a single logical database that is 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 communications network.
A DDBMS therefore has the following characteristics:

- a collection of logically related shared data;
- data split into several fragments;
- fragments may be replicated;
- fragments/replicas are allocated to sites;
- sites are linked by a communications network;
- data at each site is under the control of a DBMS;
- DBMS at each site can handle local applications, autonomously;
- each DBMS participates in at least one global application.

3.3.1 Distributed processing

Distributed processing	A centralized database that can be accessed over a computer network.

3.4 Data Warehousing

Data warehouse	A consolidated/integrated view of corporate data drawn from disparate operational data sources and a range of end-user access tools capable of supporting simple to highly complex queries to support decision making.

- Architecture:
  - **Operational Data Store (ODS)**: Staging area for data.
  - **Load Manager**: Extracts and loads data.
  - **Warehouse Manager**: Manages data (e.g., transformations, indexing).
  - **Query Manager**: Handles user queries.
  - **Metadata**: Describes data and processes.

3.4 Cloud Computing

Cloud computing

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 (NIST, 2011)[1].

Virtualization is the creation of a virtual version of something, such as a server, operating system, storage device, or network resource. The aim is to provide these resources with minimal management or service provider interaction.

The essential characteristics are as follows:

• **On-demand self-service**. Consumers can obtain, configure, and deploy cloud services themselves using cloud service catalogues, without requiring the assistance of anyone from the cloud provider.

• **Broad network access**. The most vital characteristic of cloud computing, namely that it is network based, and accessible from anywhere, from any standardized platform (e.g., desktop computers, laptops, mobile devices).

• **Resource pooling.** The cloud 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.

• **Rapid elasticity.** Resource pooling avoids the capital expenditure required for the establishment of network and computing infrastructure.

1. **Service Models**:
    - **SaaS (Software as a Service)**: Software and associated data are centrally hosted on the cloud.
    - **PaaS (Platform as a Service)**: Platform for application development (e.g., Google App Engine, Microsoft Azure). PaaS a computing platform that allows the creation of web applications quickly and easily and without the complexity of buying and maintaining the software and infrastructure underneath it.
    - **IaaS (Infrastructure as a Service)**: Virtualized infrastructure (e.g., Amazon EC2). Iaas delivers servers, storage, network and oper- ating systems—typically a platform virtualization environment—to consumers as an on-demand service, in a single bundle and billed according to usage
2. **Deployment Models**:
    - **Private Cloud**: For a single organization.
    - **Community Cloud**: Shared by a specific community.
    - **Public Cloud**: Available to the general public.
    - **Hybrid Cloud**: Combination of private, community, and public clouds.

3.5.1 Benefits and Risks of Cloud Computing

Cloud computing presents numerous benefits and some potential risks for organizations:

Benefits:

1. **Cost Reduction**: Organizations avoid up-front capital expenses on servers, software licenses, and IT personnel by shifting these responsibilities to the cloud provider.
2. **Scalability/Agility**: Resources can be provisioned and relinquished as needed, supporting business growth and agility.
3. **Improved Security**: Cloud providers can offer better security than in-house systems, and address compliance standards more efficiently.
4. **Improved Reliability**: Cloud providers can focus on increasing system reliability, offering 24/7 support and high-level computing.
5. **Access to New Technologies**: Organizations can access the latest hardware, software, and IT functionality.
6. **Faster Development**: Cloud platforms provide core services, templates, and tools that accelerate development cycles.
7. **Large-Scale Prototyping/Load Testing**: Cloud computing facilitates large-scale prototyping and load testing, which would be difficult and expensive with in-house servers.
8. **Flexible Working Practices**: Cloud computing allows staff to work flexibly, accessing files from web-enabled devices and supporting global collaboration.
9. **Increased Competitiveness**: Organizations can focus on core competencies and further develop capabilities, increasing competitiveness.

Risks:

1. **Network Dependency**: Dependency on network availability means that power outages and service interruptions can prevent access to cloud facilities.
2. **System Dependency**: Organizations may rely heavily on the cloud provider's availability and reliability.
3. **Cloud Provider Dependency**: The risk of a cloud provider going out of business or being acquired, which could disrupt services.
4. **Lack of Control**: Organizations may not have full control over their data and may face issues with data portability, interoperability, integrity, confidentiality, intervenability, and isolation.
5. **Lack of Information on Processing (Transparency)**: Insufficient information about cloud service processing operations can pose risks to data controllers and data subjects.

3.5.2 Cloud-Based Database Solutions

Cloud-Based Database Solutions: DaaS and DBaaS

Data as a Service (DaaS):

- **Definition**: Offers the ability to define and query data in the cloud via APIs, without using typical DBMS interfaces like SQL.

- **Examples**:
    - Urban Mapping: Provides geographic data for embedding into websites and applications.
    - Xignite: Makes financial data available to customers.
    - Hoovers: Provides business data on various organizations.

Database as a Service (DBaaS):

- **Definition**: Offers full database functionality to developers. Managed by a management layer responsible for monitoring, scaling, and resource allocation.
- **Capabilities**:
    - On-demand, self-service provisioning and management of database instances.
    - Automated monitoring and compliance with service definitions and quality levels.
    - Fine-grained metering of usage for charge-back functionality.

DBaaS Architectural Options:

1. **Separate Servers**: Each tenant has a hosted server machine dedicated to their database. High isolation, but higher maintenance costs.
2. **Shared Server, Separate Database Server Process**: Multiple tenants share a single server machine, each with their own database and server process. Common in virtualization environments.
3. **Shared Database Server, Separate Databases**: Tenants share a single database server but have separate databases. Allows for better resource utilization.
4. **Shared Database, Separate Schema**: Single database server with data grouped into schemas for each tenant. Requires permission structures to ensure data access is limited to authorized users.
5. **Shared Database, Shared Schema**: Tenants share the same database tables with columns identifying the owner of each row. Lowest hardware and backup costs but requires additional security measures.

3.6 Components of a DBMS

A DBMS is a sophisticated software system composed of several components or modules, each with a specific function. Here is a summarized architecture of a DBMS:

1. **Query Processor**:
    - Transforms user queries into a series of low-level instructions directed to the database manager.
2. **Database Manager (DM)**:
    - Interfaces with application programs and queries.

- Determines required conceptual records and calls the file manager to perform the request.

Components of the Database Manager:

- **Authorization Control**: Verifies user permissions for operations.
- **Command Processor**: Executes operations once user authority is confirmed.
- **Integrity Checker**: Ensures operations meet integrity constraints.
- **Query Optimizer**: Determines the optimal strategy for query execution.
- **Transaction Manager**: Handles operations received from transactions.
- **Scheduler**: Manages concurrent operations to prevent conflicts.
- **Recovery Manager**: Ensures the database remains consistent during failures.
- **Buffer Manager (Cache Manager)**: Manages data transfer between main memory and secondary storage.

3. **File Manager**:
   - Manipulates underlying storage files and manages storage space allocation.
   - Maintains lists of structures and indexes in the internal schema.
   - Coordinates with access methods to read/write data into the system buffer.
4. **DML Preprocessor**:
   - Converts Data Manipulation Language (DML) statements in an application program into standard function calls.
5. **DDL Compiler**:
   - Converts Data Definition Language (DDL) statements into metadata tables stored in the system catalog.
6. **Catalog Manager**:
   - Manages and maintains access to the system catalog, which is accessed by most DBMS components.

Data Structures:

- **Data and Index Files**: Store data and indexes.
- **System Catalog**: Contains metadata and control information.

3.7 Oracle Architecture

1. **Physical Database Structures:**
   - **Data files:**
     These files store the actual data of the database (such as tables and indexes). Data files are grouped into table spaces, and a database can range from having a single table space with one data file to multiple table spaces with several data files.
   - **Redo Log Files:**
     A set of at least two redo log files records every change made to the database. They are critical for recovery, ensuring that even if data isn't yet written permanently to data files, the changes can be recovered.

- **Control Files:**
  These files contain metadata about the database's physical structure, including a list of all data files and redo log files. For robustness, control files (and often redo log files) are multiplexed across multiple devices.

2. **Oracle Instance Components:**
   - **Processes and Memory Areas:**
     The Oracle instance includes both background processes and user/server processes. It uses two major memory areas:
     - **System Global Area (SGA):**
       A shared memory area allocated at start up that holds:
       - **Database Buffer Cache:** Recently accessed data blocks.
       - **Redo Log Buffer:** Holds entries that will be written to redo log files.
       - **Shared Pool:** Caches shared SQL areas, parse trees, execution plans, and data dictionary information.
       - Other specific pools (Large Pool, Java Pool, Streams Pool) and the Fixed SGA for internal housekeeping.
     - **Program Global Area (PGA):**
       A memory region for each process that stores session-specific data and control information.
3. **Oracle Processes:**
   - **Background Processes:**
     These handle essential system functions:
     - **Database Writer (DBWR):** Writes modified (dirty) blocks from the buffer cache to data files.
     - **Log Writer (LGWR):** Writes the redo log buffer entries to the redo log files.
     - **Checkpoint (CKPT):** Coordinates checkpoints by updating data files and control files.
     - **System Monitor (SMON):** Handles crash recovery and defragmentation.
     - **Process Monitor (PMON):** Cleans up after failed user processes.
     - **Archiver (ARCH):** Archives filled redo log files.
     - **Recoverer (RECO):** Manages failed or suspended distributed transactions.
     - **Flashback/Recovery Writer (RVWR):** Manages flashback logs for data recovery.
     - **Manageability Monitors (MMON and MMNL):** Gather and write performance statistics.
   - **User and Server Processes:**
     User processes (from client applications) connect to the database server, which creates corresponding dedicated server processes to handle the requests. Communication between client and server processes is managed by Oracle Net Services, which supports various network protocols.
4. **Interaction Example:**
   - A client application connects via Oracle Net Services.
   - The server creates a dedicated process to handle the connection.

- SQL statements executed by the client are processed by checking the shared pool, accessing data from the SGA or data files, and updating buffers.
- Modified data is written to disk by DBWR and changes are recorded immediately in redo log files by LGWR upon commit.
- Background processes continuously monitor and manage various aspects of system performance and recovery.