DBMS

Week One

Content we will cover in this semester

Basics	Relational Languages	Database Design	Application Design & Development
Big Data Analytics	Storage Management & Indexing	Query processing and Optimization	Transaction Management

Part One

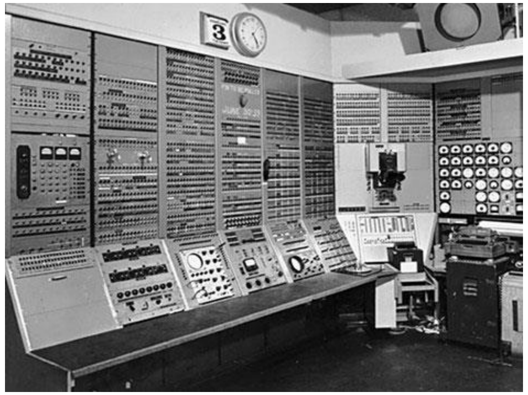
BASICS

Chapter 1 Introduction

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

Database-System Applications

Evolution to Database Systems



History of Databases (1960-2020)

1960s
Databases were coined, two popular data models: Network (Eg. CODASYL) & Hierarchical (Eg. IMS)

1970s
Relational databases (RDBMS) first appear.

1980s
SQL (Standard Query Language) came into limelight. Commercial success to RDBMS.

1990s
Emergence of Oracle, PowerBuilders, VB and others. Excel/Access became defacto for personal productivity. Non-relational databases emerge.

2000s
Object oriented programming came into picture, new applications were developed and databases continued to grow as Internet became mainstream.

2010s
APIs became mainstream.
New age database applications came about, mimicking the ease of use of a spreadsheet. New players emerge in the non-relational database space.

2020...
Rise of low/no-code applications, APIs and easy to use, non-technical databases continue to take on the world and power next generation of application development

stackby.com

Importance of Data

- IMPROVE PEOPLE'S LIVES
- MAKE INFORMED DECISIONS
- STOP MOLEHILLS FROM TURNING INTO MOUNTAINS
- GET THE RESULTS YOU WANT
- FIND SOLUTIONS TO PROBLEMS
- BACK UP YOUR ARGUMENTS
- STOP THE GUESSING GAME
- BE STRATEGIC IN YOUR APPROACHES
- KNOW WHAT YOU ARE DOING WELL
- KEEP TRACK OF IT ALL
- MAKE THE MOST OF YOUR MONEY
- ACCESS THE RESOURCES AROUND YOU

Abstraction for Managing Complexity

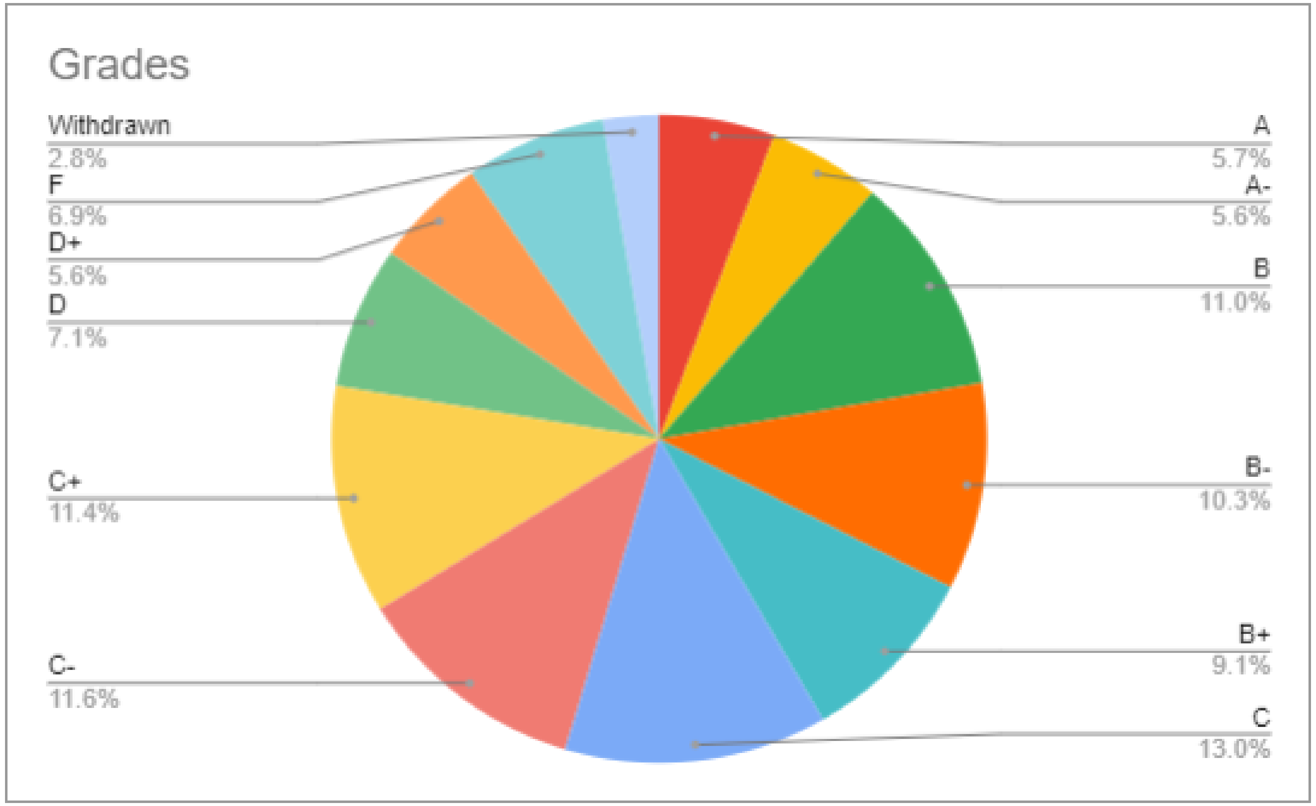
- Abstraction allows users to interact with complex systems without understanding their internal details.
- Database systems provide a high-level abstraction of data storage and organization.
- Abstraction simplifies interaction for users and programmers.

Representative Applications

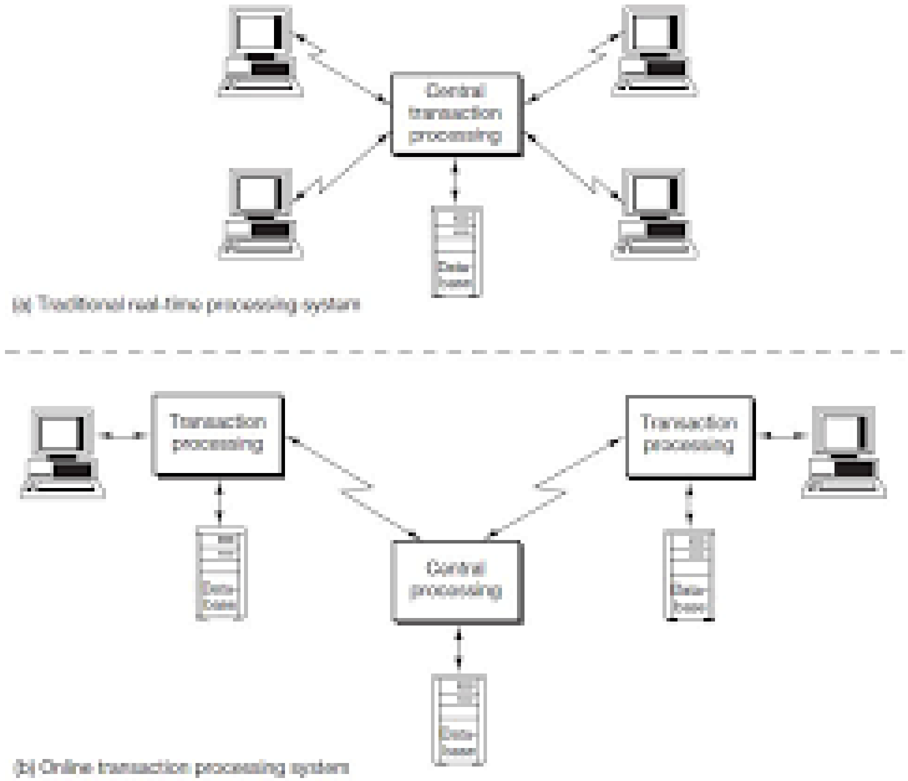
- Enterprise information (sales, accounting, human resources)
- Manufacturing (supply chain management, production tracking)
- Banking and finance (customer information, transactions, holdings)
- Universities (student records, course registrations)
- Airlines (reservations, schedules)
- Web-based services (social media, online retail, advertisements)
- Document databases, navigation systems, etc.

Changing User Interaction with Databases

Session	Subject	Student Intake Year	Program	Registration No	StudentName	Grade	Grade Point	Total Marks	Final Examination	Midterm Exam
FA21	Design and Analysis of Algorithms	FA18	BSCS	FA18-BSCS-0077	SYEDTAHAHASAN	A	4.00	93.00	40.00	
FA21	Design and Analysis of Algorithms	FA19	BSCS	FA19-BSCS-0019	SYEDSHAHZAIBRAFIQ	A	4.00	89.00	36.00	
FA21	Design and Analysis of Algorithms	SP20	MCS	SP20-MCS-0013	FARHANAHMED	A	4.00	88.00	32.00	
FA21	Design and Analysis of Algorithms	SP20	MCS	SP20-MCS-0018	AMNAAHMED	A	4.00	86.00	30.67	
FA21	Design and Analysis of Algorithms	SP20	MCS	SP20-MCS-0029	ASADZAKIR	A	4.00	87.00	34.67	
FA21	Design and Analysis of Algorithms	SP20	MCS	SP20-MCS-0032	FARZAMYOUSUF	A	4.00	86.00	34.67	
FA21	Design and Analysis of Algorithms	SP21	MCS	SP21-MCS-0018	ARBAZKHAN	A	4.00	90.00	37.33	
FA21	Design and Analysis of Algorithms	FA21	MSCS	FA21-MSCS-0007	RAMEEZQURESHI	A-	3.67	85.00	30.67	
FA21	Design and Analysis of Algorithms	SP20	MCS	SP20-MCS-0035	SEEMAALIAFRIDI	B	3.00	74.00	25.33	
FA21	Design and Analysis of Algorithms	FA21	MSCS	FA21-MSCS-0049	MUHAMMADHAMZAARAB	B-	2.67	72.00	26.67	
FA21	Design and Analysis of Algorithms	SP20	MCS	SP20-MCS-0039	AMIRQADEER	B-	2.67	73.00	26.67	
FA21	Design and Analysis of Algorithms	SP21	MCS	SP21-MCS-0005	MUHAMMADTALHABINTAHIR	B-	2.67	70.00	24.00	
FA21	Design and Analysis of Algorithms	SP20	BSCS	SP20-BSCS-0071	MARIUMZAFARREHMAN	B+	3.33	79.00	33.33	
FA21	Design and Analysis of Algorithms	FA19	MSCS	FA19-MSCS-0028	AYAZAHMED	C	2.00	64.00	14.67	
FA21	Design and Analysis of Algorithms	SP20	MCS	SP20-MCS-0033	SYEDADNANSHAH	C	2.00	65.00	20.00	
FA21	Design and Analysis of Algorithms	SP21	MCS	SP21-MCS-0007	MUHAMMADSALMANSAJID	C	2.00	64.00	13.33	
FA21	Design and Analysis of Algorithms	FA18	MCSW	FA18-MCSW-0022	FAHADIFTIKHARKHAN	C-	1.67	58.00	14.67	
FA21	Design and Analysis of Algorithms	SP18	BSCS	SP18-BSCS-0028	SYEDKHAWARHUSSAINBURNI	C-	1.67	59.00	18.67	
FA21	Design and Analysis of Algorithms	SP20	BSCS	SP20-BSCS-1015	SHAHOON	C-	1.67	60.00	20.00	
FA21	Design and Analysis of Algorithms	SP20	MCS	SP20-MCS-0005	SYEDTAQIRIZVI	C-	1.67	58.00	18.67	
FA21	Design and Analysis of Algorithms	SP20	MCS	SP20-MCS-0037	MAJIDALI	C-	1.67	60.00	13.33	
FA21	Design and Analysis of Algorithms	SP19	MCSW	SP19-MCSW-0007	TALHAHUSSAIN	D	1.00	50.00	10.67	



Modes of Database Usage



Purpose of Database Systems

Disadvantages of File-Processing Systems

- Data redundancy and inconsistency arise due to different structures and formats of files and application programs.
- Difficulty in accessing data arises when application programs are not available for specific queries.
- Data isolation occurs due to the scattered nature of data in various files.
- Integrity problems result from enforcing consistency constraints across various files and programs.
- Atomicity problems involve ensuring that operations either fully complete or leave the system unchanged in the event of a failure.
- Concurrent-access anomalies lead to inconsistencies when multiple users update data simultaneously.
- Security issues arise when different users require different levels of data access.

Transition to Database Systems

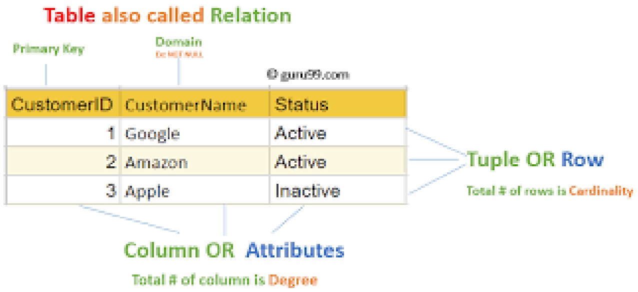
- Database systems were developed in response to the challenges faced by file-processing systems.
- Database systems provide solutions to the limitations of file-based systems.

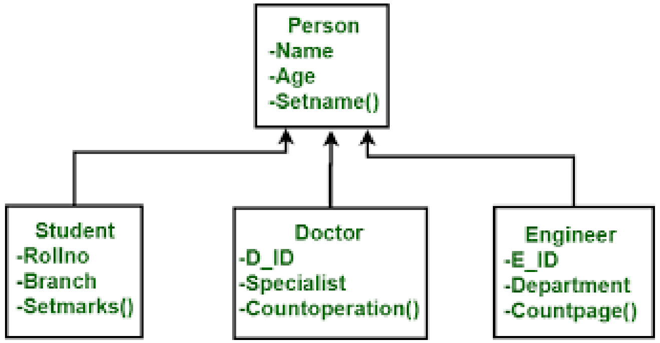
View of Data

Data Models

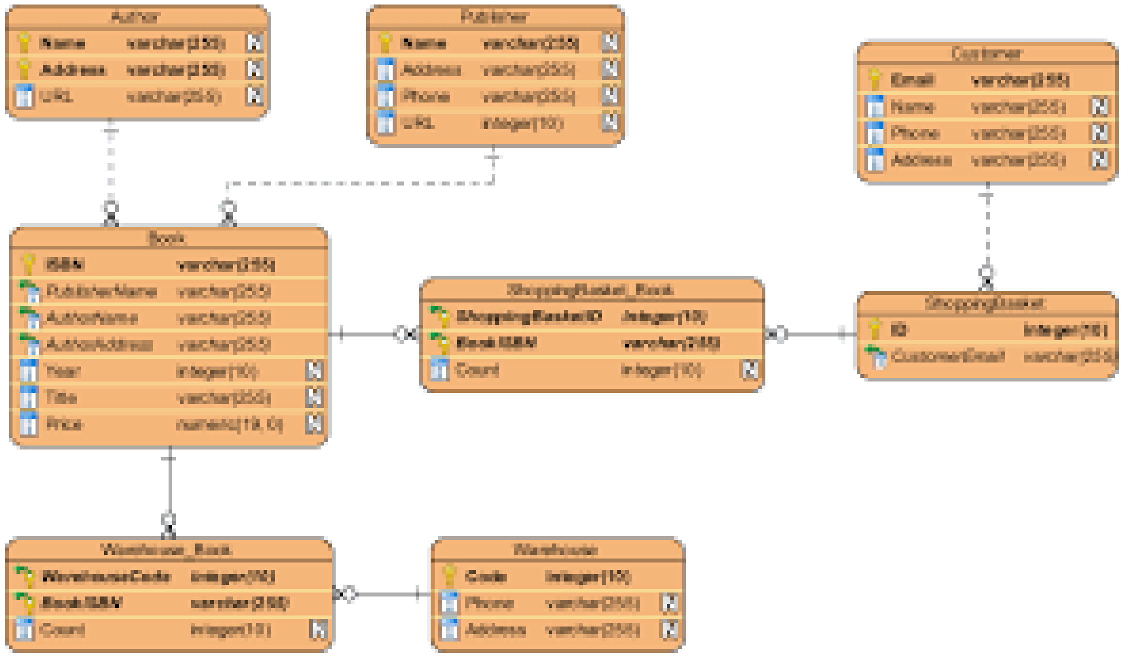
- A data model provides conceptual tools to describe data, relationships, semantics (meaning), and constraints.
- Different data models exist, including the relational, entity-relationship, semi-structured, and object-based models.
- The relational model is the most widely used and serves as the foundation for many database applications.

Data Models



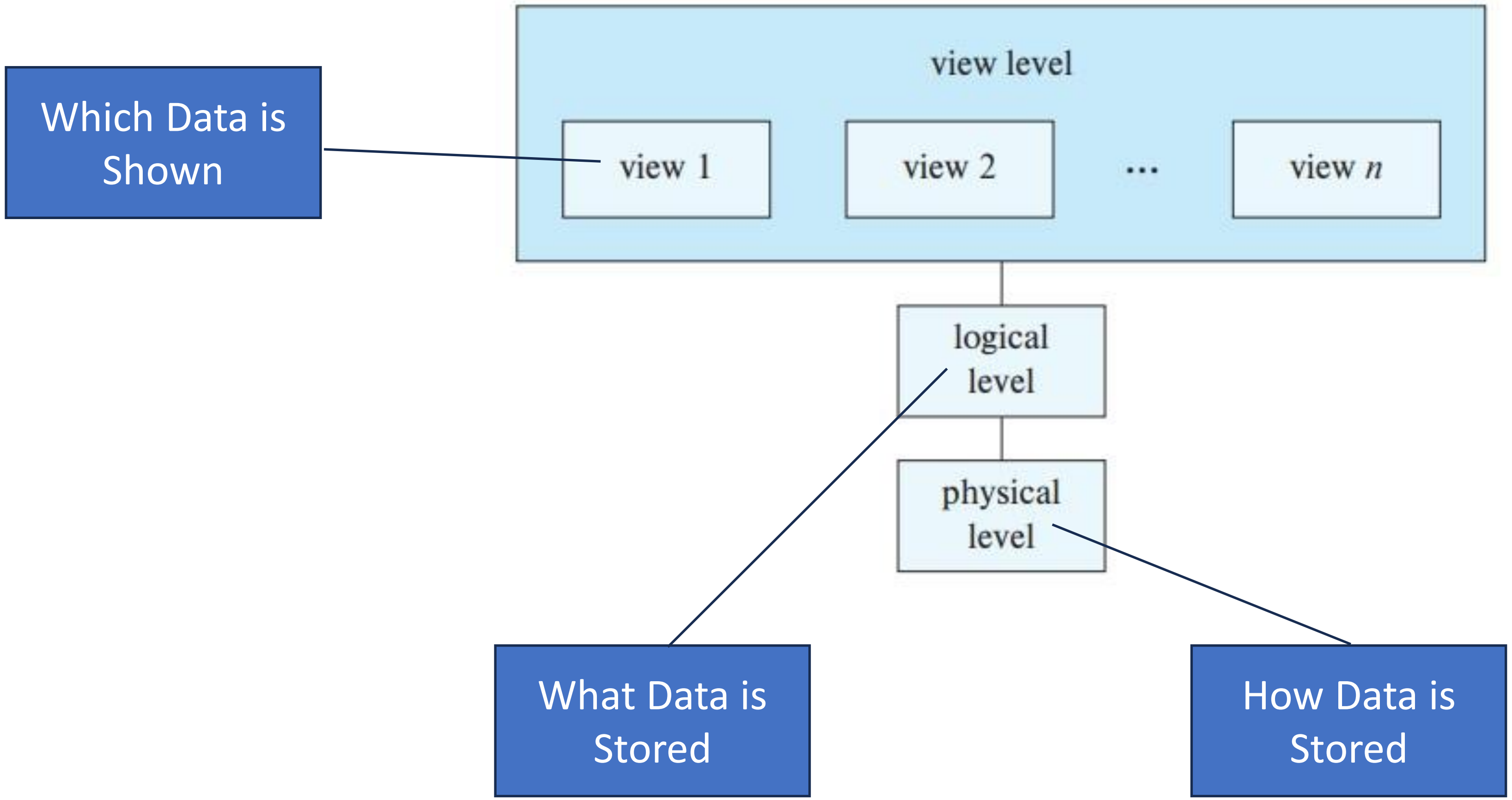



```
## Document 1 ##
{
  "customerID": "103248",
  "name":
  {
    "first": "AAA",
    "last": "BBB"
  },
  "address":
  {
    "street": "Main Street",
    "number": "101",
    "city": "Acity",
    "state": "NY"
  },
  "ccOnFile": "yes",
  "firstOrder": "02/28/2003"
}
```

Data Abstraction

- Database systems use different levels of data abstraction to hide implementation complexities from users.
- The levels of abstraction are physical, logical, and view.
- The physical level describes how data are stored,
- the logical level describes what data are stored and relationships,
- and the view level presents a subset of the data to specific users.


Which Data is Shown
view level
view 1
view 2
...
view n
logical level
physical level
What Data is Stored
How Data is Stored

Instances and Schemas

- A database instance refers to the collection of data stored in the database at a specific time.
- The database schema refers to the overall design of the database.
- Schemas are partitioned into physical, logical, and view levels.
- Logical schema is crucial for application programs, and changes to the physical schema should not affect application programs if physical data independence is achieved.
- Flexibility in logical schema design is essential for modern database applications.

Database Languages

Data-Definition Language (DDL)

- DDL is used to specify the database schema and properties of the data.
- It includes data storage and definition language to define storage structure and access methods.
- DDL is used to define integrity constraints like domain constraints, referential integrity, and authorization.

SQL Data-Definition Language

- SQL provides a rich DDL for defining tables, data types, integrity constraints, and authorizations.
- Examples of DDL statements include creating tables with specific columns and data types, specifying primary keys, and defining relationships.

Data-Manipulation Language (DML)

- DML enables users to retrieve, insert, delete, and modify data in the database.
- Two types of DMLs:
  - procedural DML (user specifies how to get data) and
  - declarative DML (user specifies what data is needed without specifying how to get it).

SQL Data-Manipulation Language

- SQL is a nonprocedural query language for retrieving and manipulating data.
- SQL queries take tables as input and return a single table as output.
- Examples of SQL queries include retrieving specific attributes from a table and combining information from multiple tables using joins.

Database Access from Application Programs

- Application programs interact with the database using DML statements.
- Application programs are written in a host language (e.g., C/C++, Java) and include embedded SQL queries.
- Application-program interfaces like ODBC and JDBC are used to send DML and DDL statements to the database for execution.

Database Design

Context of Database Design

- Databases are part of an enterprise's operations and may serve as a supporting role or provide information as the product.
- Database design involves designing the database schema, but a complete application environment requires consideration of broader issues.

High-Level Data Model

- A high-level data model offers a conceptual framework for specifying data requirements and structuring the database.
- Initial database design involves understanding user data needs through interactions with domain experts and users.
- The outcome of this phase is a specification of user requirements, indicating what data is needed.

Translating User Requirements

- The designer selects a data model and uses it to translate user requirements into a conceptual schema of the database.
- The schema provides an overview of the enterprise, focusing on data and relationships without specifying physical storage details.

Choosing a Design Approach

- In the context of the relational model, the "what" part involves business decisions about what attributes to capture.
- The "how" part involves two approaches: using the entity-relationship model to capture relationships or employing normalization algorithms to generate tables.

Functional Requirements

- A fully developed conceptual schema indicates the functional requirements of the enterprise.
- Functional requirements involve specifying operations or transactions that will be performed on the data, such as modification, retrieval, and deletion.

Logical-Design Phase

- In the logical-design phase, the designer maps the high-level conceptual schema to the implementation data model used by the database system.
- This mapping results in a system-specific database schema, which forms the basis for the subsequent physical-design phase.

Physical-Design Phase

- The physical-design phase focuses on specifying physical features of the database, including file organization and internal storage structures.
- These features are crucial for optimizing the performance of the database system.

Database Engine

Storage Manager

- The storage manager bridges the gap between the low-level data stored in the database and the applications and queries interacting with the system.
- It interacts with the file manager to handle the storage of raw data on disk using the file system provided by the operating system.
- The storage manager is responsible for tasks such as data retrieval, storage, and updates.
- Its components include:
  - Authorization and integrity manager: Checks integrity constraints and user authorization for data access.
  - Transaction manager: Ensures database consistency despite failures and conflicts between concurrent transactions.
  - File manager: Manages disk storage space allocation and data structures for representing information stored on disk.
  - Buffer manager: Fetches data from disk to main memory and decides which data to cache in main memory.

Query Processor

- The query processor simplifies data access and facilitates efficient query execution.
- It includes:
  - DDL interpreter: Interprets Data Definition Language (DDL) statements and records them in the data dictionary.
  - DML compiler: Translates Data Manipulation Language (DML) queries into evaluation plans of low-level instructions.
  - Query evaluation engine: Executes the low-level instructions generated by the DML compiler.

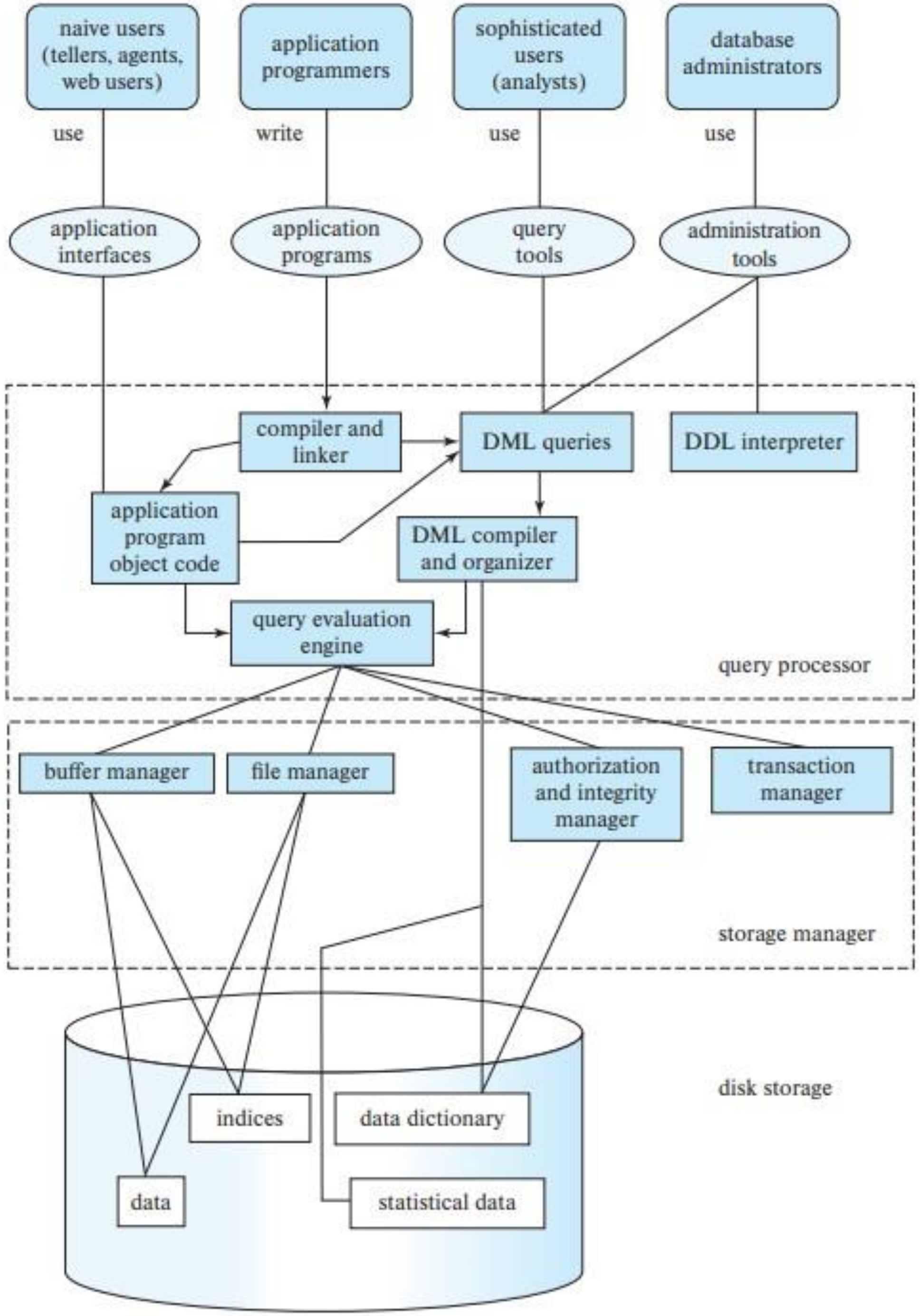
Transaction Management

- Transaction management ensures the atomicity, consistency, isolation, and durability (ACID) properties of database transactions.
- It allows a sequence of operations to be treated as a single unit of work that either completes fully or has no effect on the database.
- Components of the transaction manager include:
  - Concurrency-control manager: Manages concurrent execution of transactions to maintain database consistency.
  - Recovery manager: Detects system failures and restores the database to a consistent state before the failure occurred.

Database and Application Architecture

Centralized Architecture and System Structure

- This architecture is suitable for shared-memory server architectures where multiple CPUs exploit parallel processing.
- Users interact with the system through query tools, application interfaces, and application programs.
- Components like the query processor, storage manager, buffer manager, file manager, transaction manager, and authorization manager work together to manage the database.

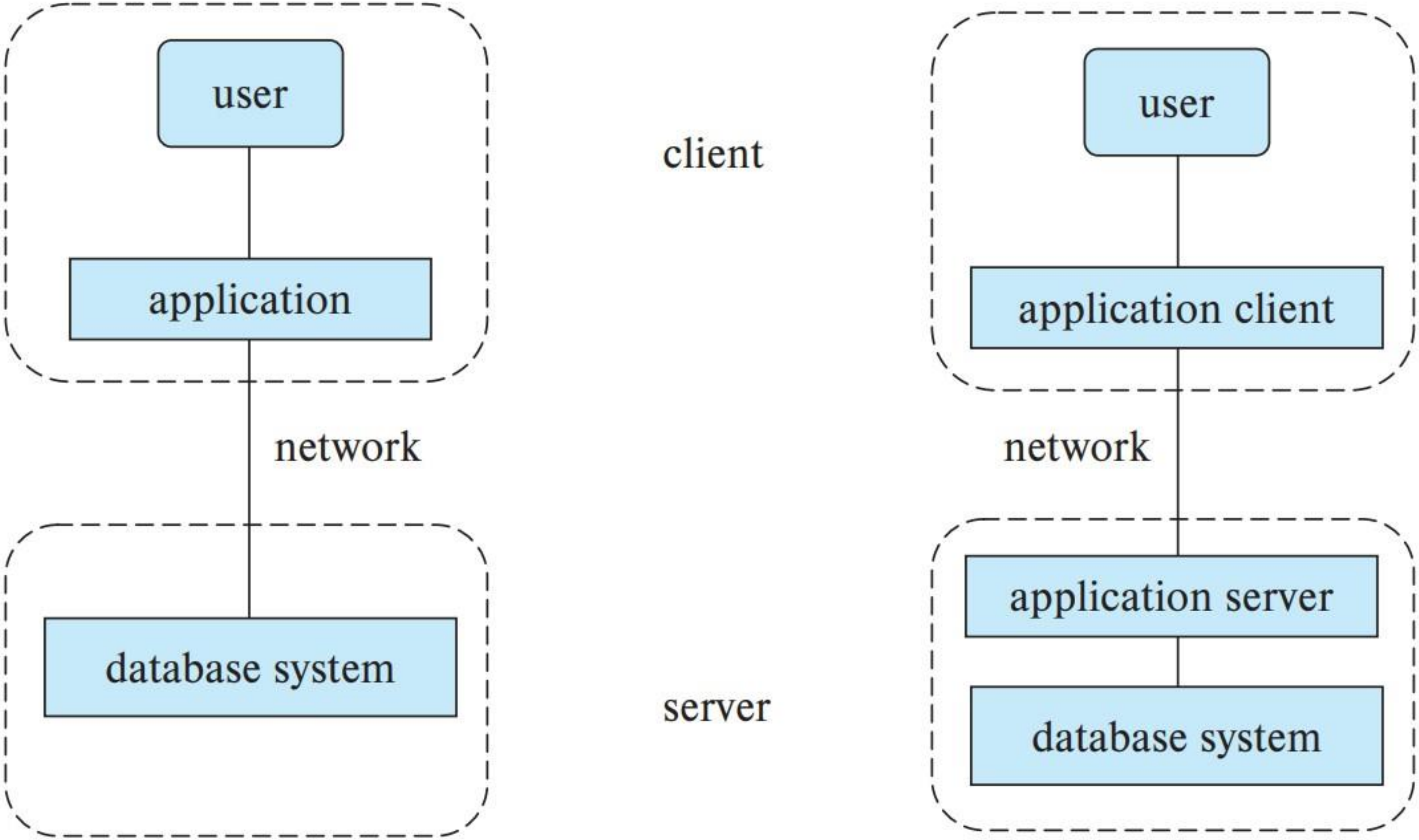

naive users
(tellers, agents, web users)
application programmers
sophisticated users
(analysts)
database administrators
use
write
use
use
application interfaces
application programs
query tools
administration tools
compiler and linker
DML queries
DDL interpreter
application program object code
DML compiler and organizer
query evaluation engine
query processor
buffer manager
file manager
authorization and integrity manager
transaction manager
storage manager
disk storage
indices
data dictionary
data
statistical data

Scalability and Distributed Databases

- To handle larger data volumes and higher processing speeds, parallel databases run on clusters of multiple machines.
- Distributed databases allow data storage and query processing across geographically separated machines.

Database Application Architecture

- Database applications can be classified into two-tier and three-tier architectures.
- Two-tier architecture involves the application residing on the client machine, which directly invokes database functionality on the server machine.
- Modern applications use a three-tier architecture, where the client is a front-end (web browser, mobile app) that communicates with an application server.
- The application server communicates with the database system, housing the application's business logic.
- Three-tier architecture enhances security and performance compared to two-tier architecture.



(a) Two-tier architecture

(b) Three-tier architecture