

Introduction and topics of study

Database Systems

- ❑ **DBMS contains information about a particular enterprise**
 - ❑ **Collection of interrelated data**
 - ❑ **Set of programs to access the data**
 - ❑ **An environment that is both *convenient* and *efficient* to use**
 - ❑ **Accessed by multiple users and applications, often at the same time.**
- ❑ **A modern database system is a complex software system whose task is to manage a large, complex collection of data.**

DBMS - Most Popular Database Management Systems



Real Life Applications

□ Railway Reservation System



shutterstock.com • 1707512422



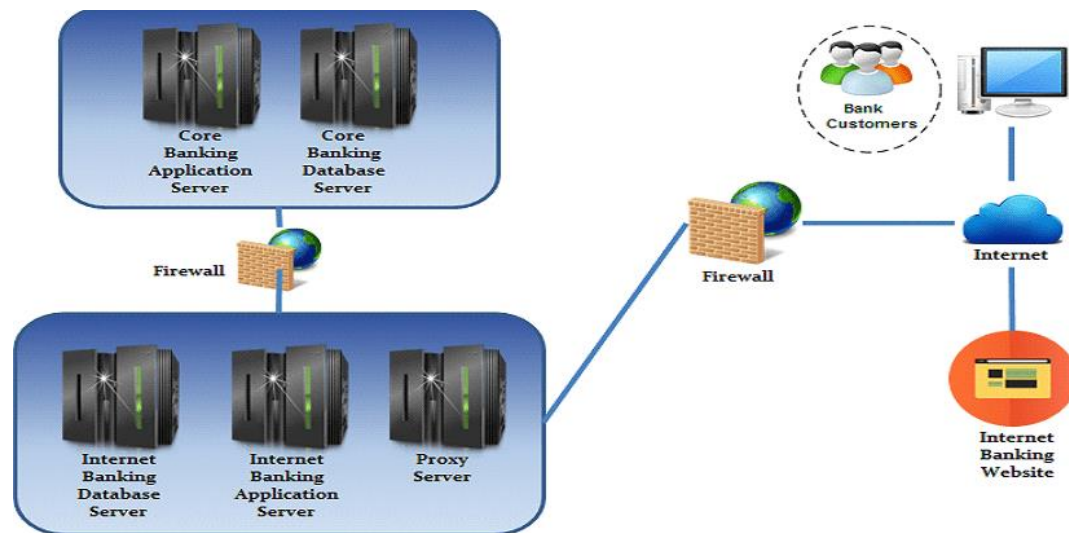
□ Library Management



□ Universities (Education Sector)



□ Banking Sector



Real Life Applications

□ Social Media Sites



Applications of DBMS

www.tutorialsmate.com



Database Applications Examples

- **Enterprise Information**
 - **Sales:** customers, products, purchases
 - **Accounting:** payments, receipts, assets
 - **Human Resources:** Information about employees, salaries, payroll taxes.
- **Manufacturing:** management of production, inventory, orders, supply chain.
- **Banking and finance**
 - **customer information, accounts, loans, and banking transactions.**
 - **Credit card transactions**
 - **Finance:** sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data
- **Universities:** registration, grades

Database Applications Examples (Cont.)

- **Airlines: reservations, schedules**
- **Web-based services**
 - **Online retailers: order tracking, customized recommendations**
 - **Online advertisements**

File Processing Systems Related Issues

In the early days, database applications were built directly on **top of file systems, which leads to:**

- ❑ **Data redundancy and inconsistency:** data is stored in multiple file formats resulting in duplication of information in different files
- ❑ **Data isolation**
 - ❑ Multiple files and different formats
- ❑ **Difficulty in accessing data**
 - ❑ Need to write a new program to carry out each new task
- ❑ **Integrity problems**
 - ❑ Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
 - ❑ Hard to add new constraints or change existing ones

Purpose of Database Systems (Cont.)

❑ Atomicity of updates

- ❑ Failures may leave database in an inconsistent state with partial updates carried out
- ❑ Example: Transfer of funds from one account to another should either complete or not happen at all

❑ Concurrent access by multiple users

- ❑ Concurrent access needed for performance
- ❑ Uncontrolled concurrent accesses can lead to inconsistencies
 - ▶ Ex: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time

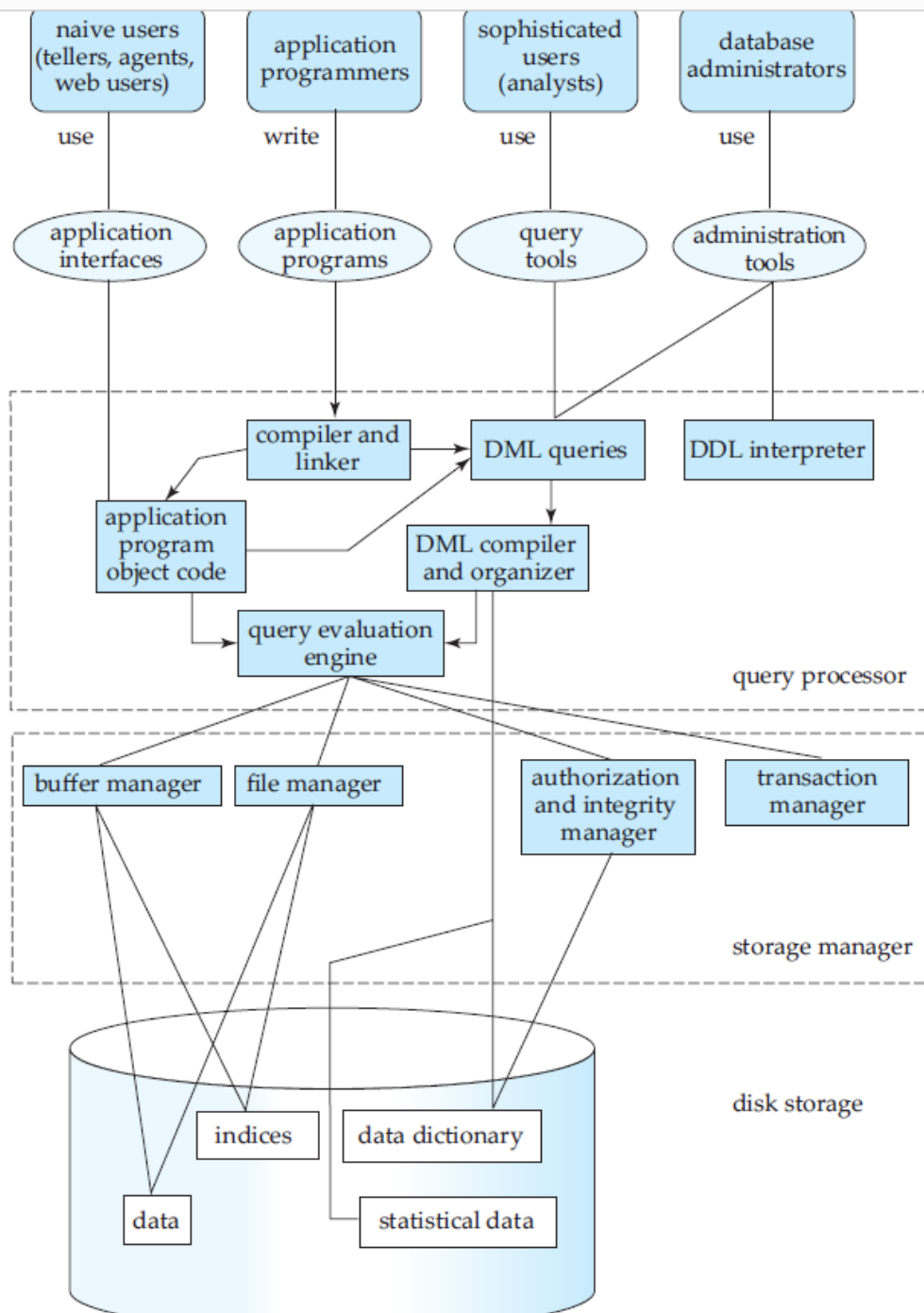
❑ Security problems

- ❑ Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

DBMS offers better data model and language

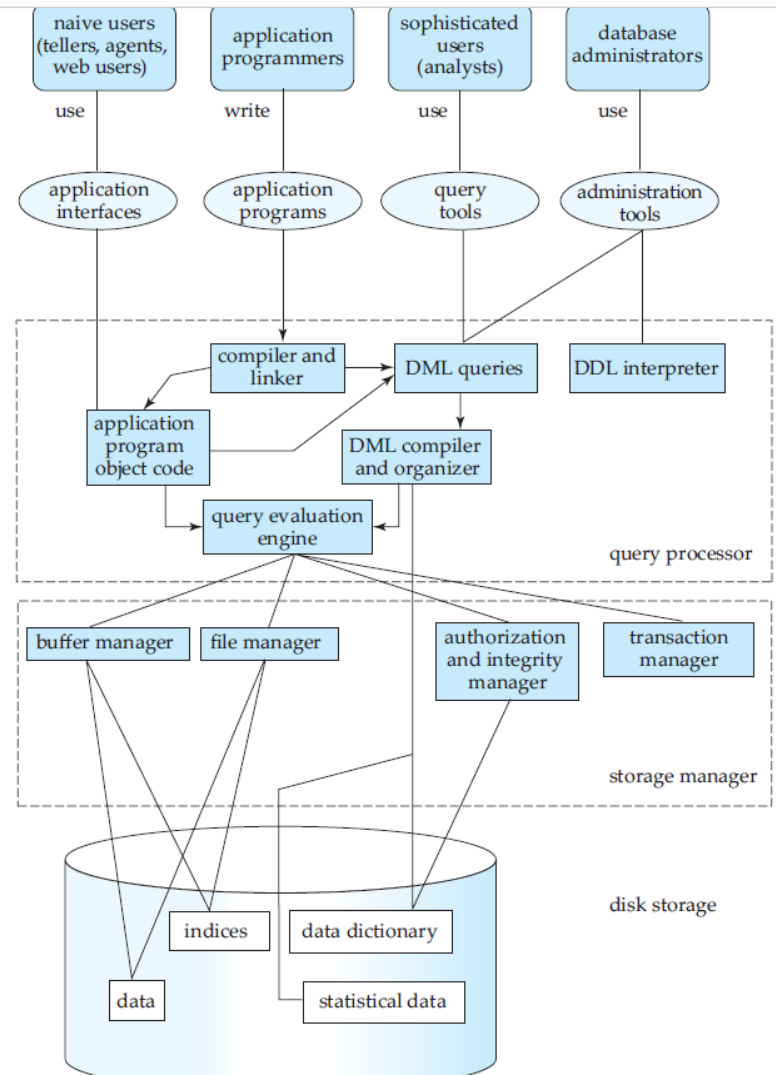
- ❑ **Data models** : capture structure, inter-relationships, validation rules. Describes data, data relationships, data semantics, and consistency constraints.
- ❑ Ex: Relational, ER Data Model
- ❑ **Interactive/Simple Querying**
- ❑ **Performance tuning and storage alternatives**
- ❑ **Complete storage management**
- ❑ **Access control, failure handling, concurrency**

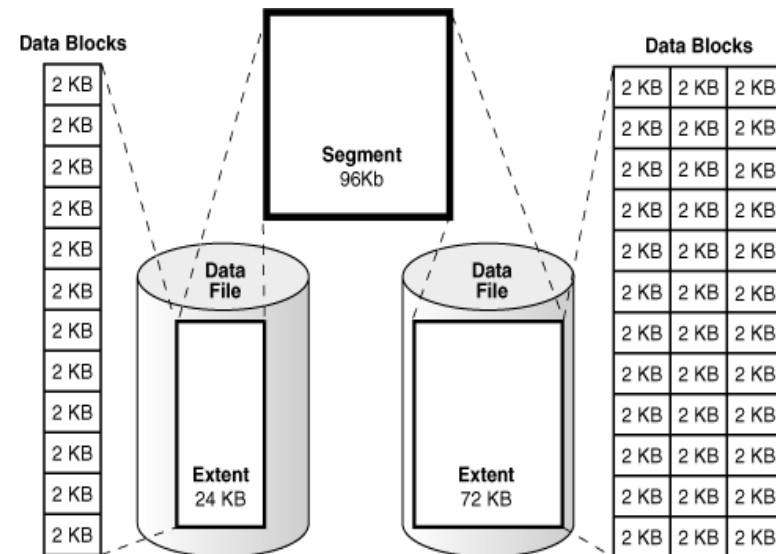
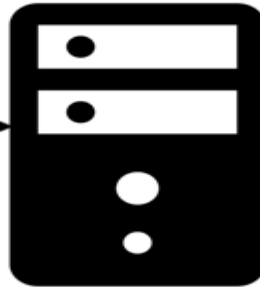


Data Abstraction

- **Data Abstraction** is a process of hiding unwanted or irrelevant details from the end user.

The database systems consist of complicated data structures and relations. For users to access the data easily, these complications are kept hidden, and only the relevant part of the database is made accessible to the users through data abstraction.



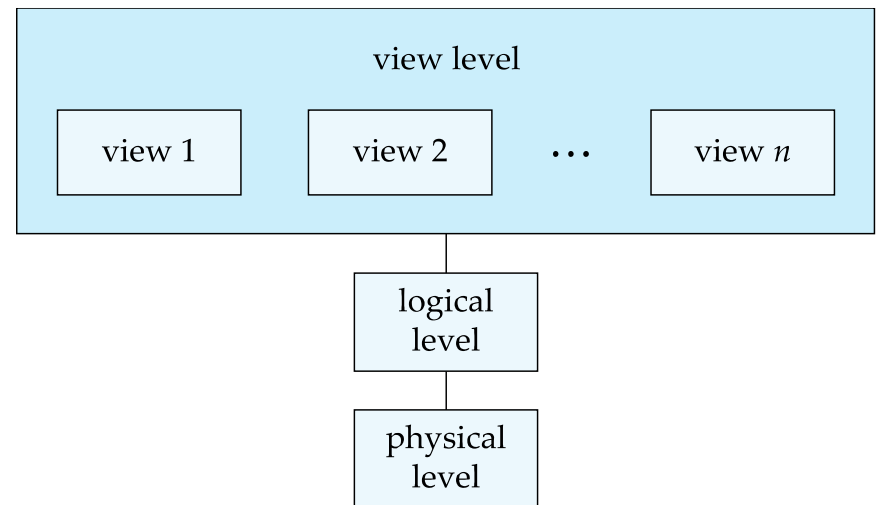


Levels of Abstraction

- ❑ **Physical or Internal Level** - it defines data-structures to store data and access methods used by the database. **(how data is stored)**
- ❑ Example: data is stored in the form of blocks of storage such as bytes, gigabytes etc.
- ❑ **Logical or Conceptual Level** – It defines **what data is stored** and describe the relationship among data.
- ❑ Example: contains tables (fields and attributes) and relationships among table attributes.
- ❑ **View or External Level:** every view only defines a part of the entire data. It also simplifies interaction with the user and it provides many views or multiple views of the same database.

Levels of Abstraction

- **Physical level:** describes how a record (e.g., instructor) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.
- **View level:** Views can hide information (such as an employee's salary) for security purposes.



Instances and Schemas

- The collection of information stored in the database at a particular moment is called an **instance**
- The overall design of the database is called the database **schema**.
- **Logical Schema** – the overall logical structure of the database
 - Example: Tables, views, and constraints.
- **Physical schema** – the overall physical structure of the database

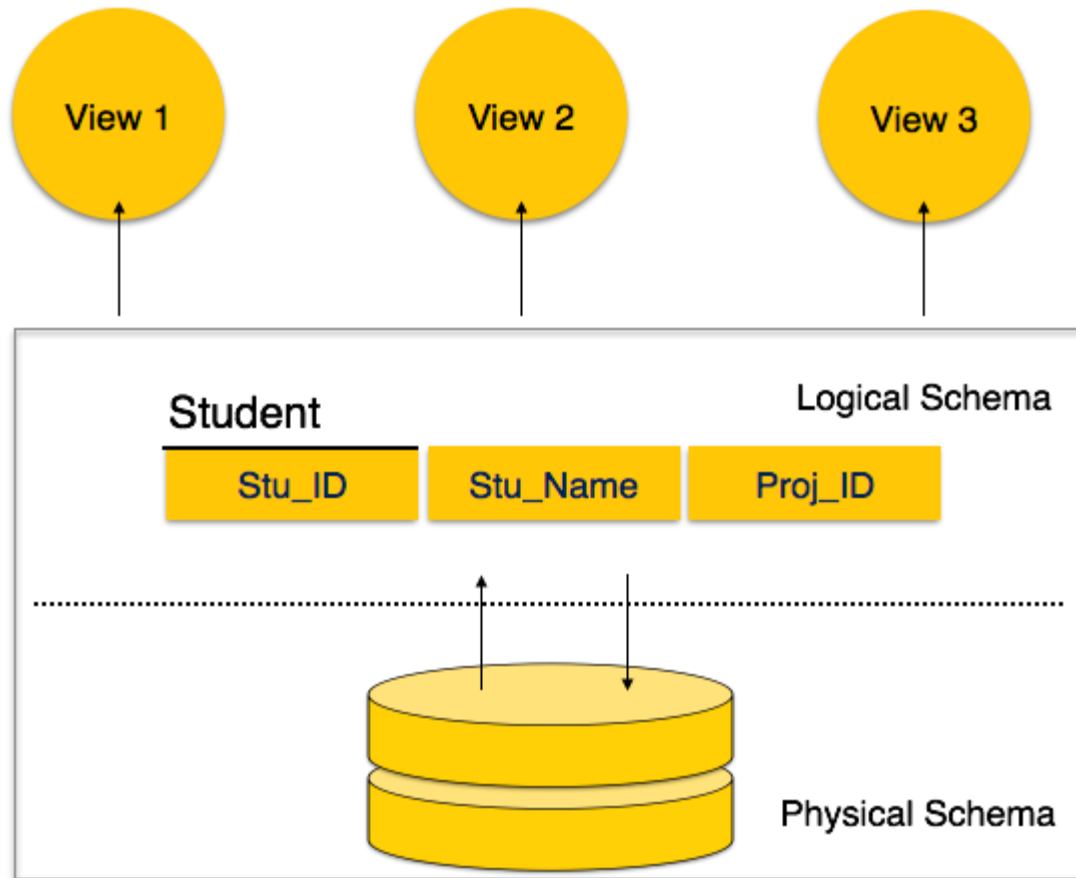
Consider the following database schema and instance.

The flights table shows the airline flights from its headquarter to several destinations.

<u>airline</u>	<u>headquarters</u>
Virgin A.	San Francisco
JetBlue	New York
Alaska	Seattle
Delta	Atlanta

<u>flights</u>	<u>airline</u>	<u>destination</u>	<u>time</u>	<u>no passengers</u>
	Alaska	San Fransisco	15:00	80
	Alaska	Chicago	15:00	220
	Delta	Chicago	12:45	275
	Delta	Chicago	15:00	225

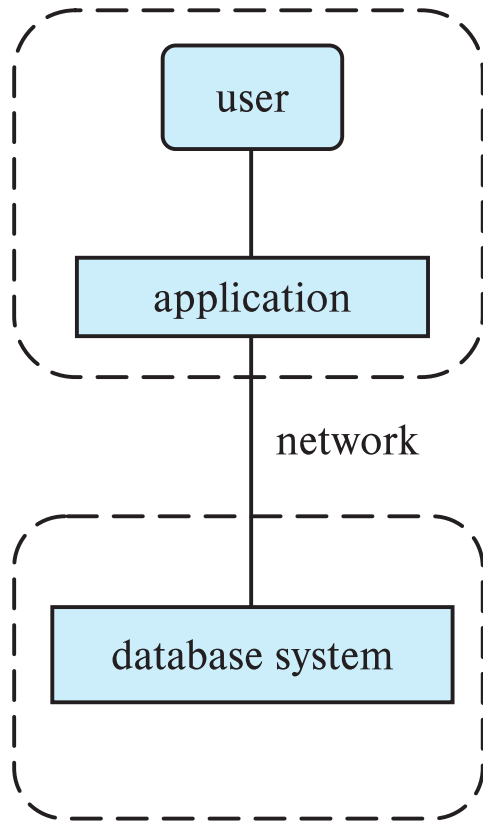
Schemas



Instances and Schemas

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
- Application programs are said to exhibit **physical data independence** if they do not depend on the physical schema, and thus need not be rewritten if the physical schema changes.

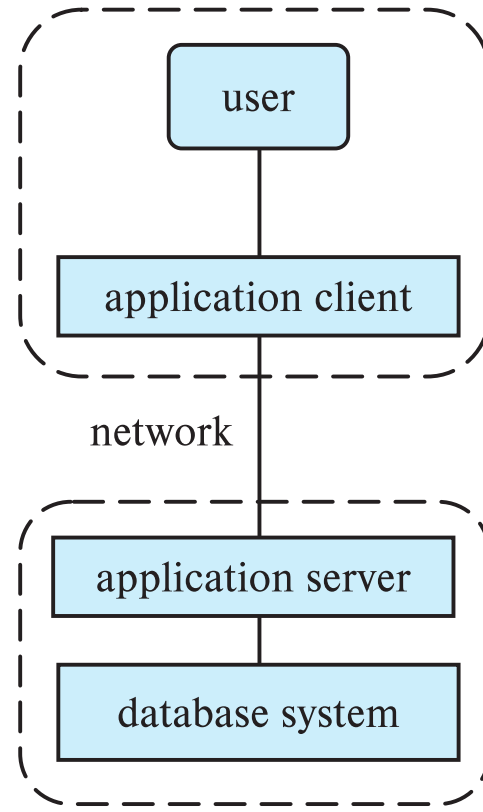
Data base Applications - Two-tier and three-tier architectures



(a) Two-tier architecture

client

server



(b) Three-tier architecture

Database Applications

Database applications are usually partitioned into two or three parts

- **Two-tier architecture -- the application resides at the client machine, where it invokes database system functionality at the server machine**
- **Three-tier architecture -- the client machine acts as a front end and does not contain any direct database calls.**
 - **The client end communicates with an application server, usually through a forms interface.**
 - **The application server in turn communicates with a database system to access data.**

History of Database Systems

□ 1950s and early 1960s:

- Data processing using magnetic tapes for storage
- Punched cards for input




□ Late 1960s and 1970s:

- Hard disks allowed direct access to data
- Network and hierarchical data models in widespread use
- **Ted Codd defines the relational data model**
- High-performance (for the era) transaction processing



History

- **Continuous trends to expand capabilities**
 - SQL becomes industrial standard
 - Parallel and distributed database systems
 - Object-oriented database systems
 - Large decision support and data-mining applications
 - Large multi-terabyte data warehouses
 - Emergence of Web commerce
 - XML and XQuery standards
 - Automated database administration
 - Giant data storage systems
 - ▶ Google BigTable, Hadoop, HBase, ..  NoSQL

**Coming next : Relational Data model
and SQL**