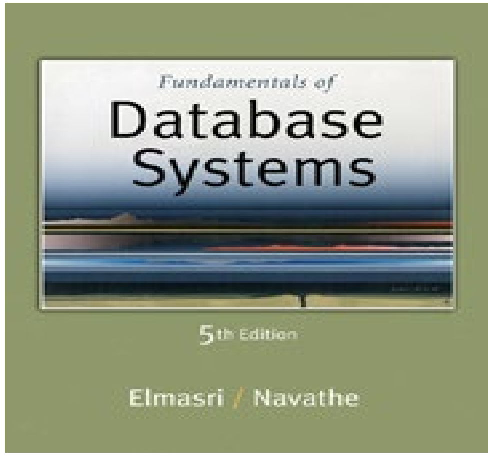
DBMS

Introduction to DBMS

Lecture By

Prof. M. Iqbal Bhat



OUTLINE

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Types of Database Users
- Advantages of Using the Database Approach
- Historical Development of Database Technology
- Extending Database Capabilities
- When Not to Use Databases

What is data, database, DBMS

- Data: Known facts that can be recorded and have an implicit meaning; raw
- Database: a highly organized, interrelated, and structured set of data about a particular enterprise
  - Controlled by a database management system (DBMS)
- DBMS
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database systems are used to manage collections of data that are:
  - Highly valuable
  - Relatively large
  - 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.
- Databases touch all aspects of our lives

Types of Databases and Database Applications

Traditional applications:

- Numeric and textual databases

More recent applications:

- Multimedia databases
- Geographic Information Systems (GIS)
- Biological and genome databases
- Data warehouses
- Mobile databases
- Real-time and active databases

Recent Developments (1)

Social Networks started capturing a lot of information about people and about communications among people-posts, tweets, photos, videos in systems such as:

- Facebook
- Twitter
- Linked-In

All of the above constitutes data

Search Engines, Google, Bing, Yahoo: collect their own repository of web pages for searching purposes

Recent Developments (2)

New technologies are emerging from the so-called non-SQL, non-database software vendors to manage vast amounts of data generated on the web:

- **Big data** storage systems involving large clusters of distributed computers (Chapter 25)
- **NOSQL** (Non-SQL, Not Only SQL) systems (Chapter 24)

A large amount of data now resides on the "cloud" which means it is in huge data centers using thousands of machines.

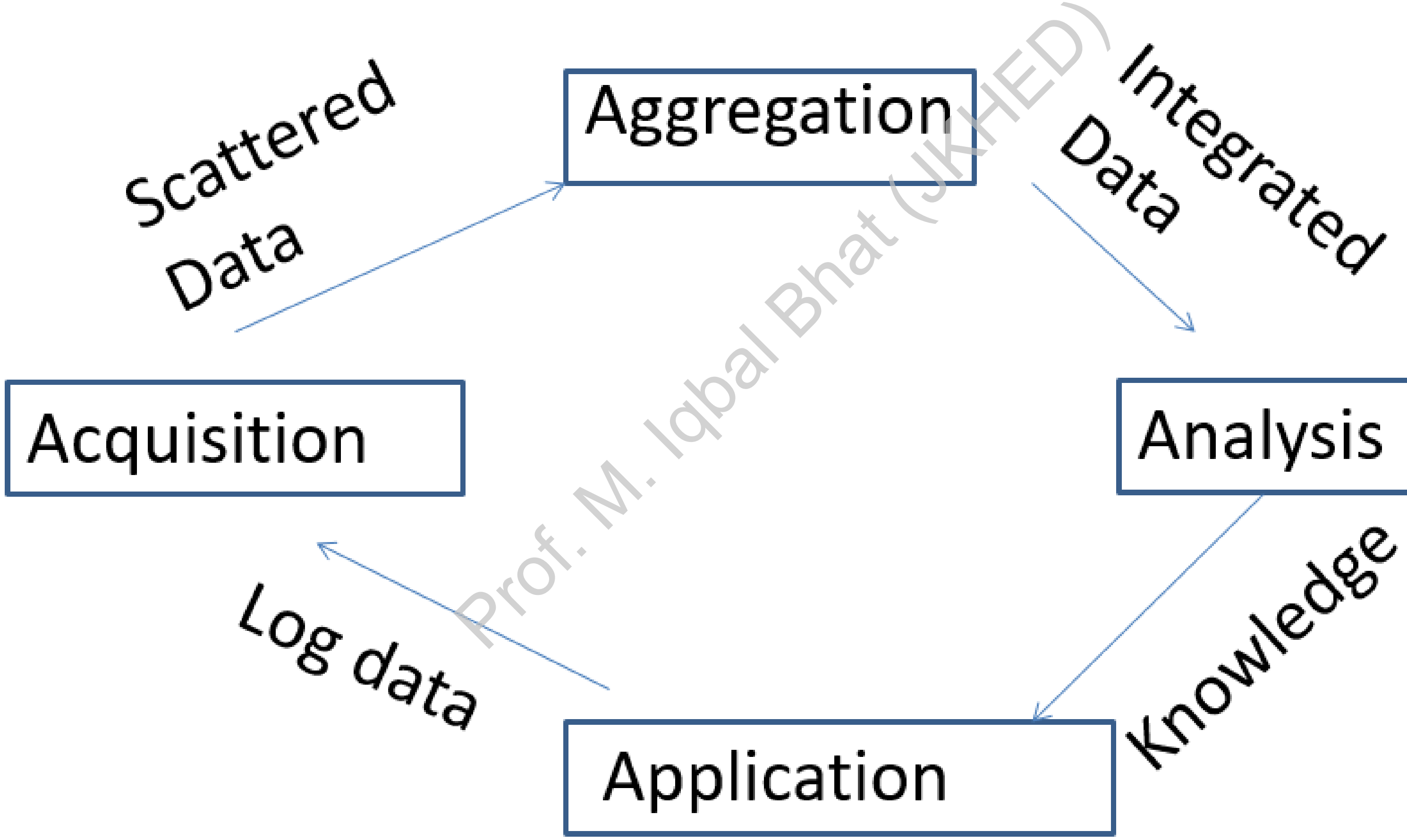
What is “big data”?

- "Big data are high-volume, high-velocity, and/or high-variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization” (Gartner 2012)
- Bottom line: Any data that exceeds our current capability of processing can be regarded as “big”
  - Complicated (intelligent) analysis of data may make a small data “appear” to be “big”

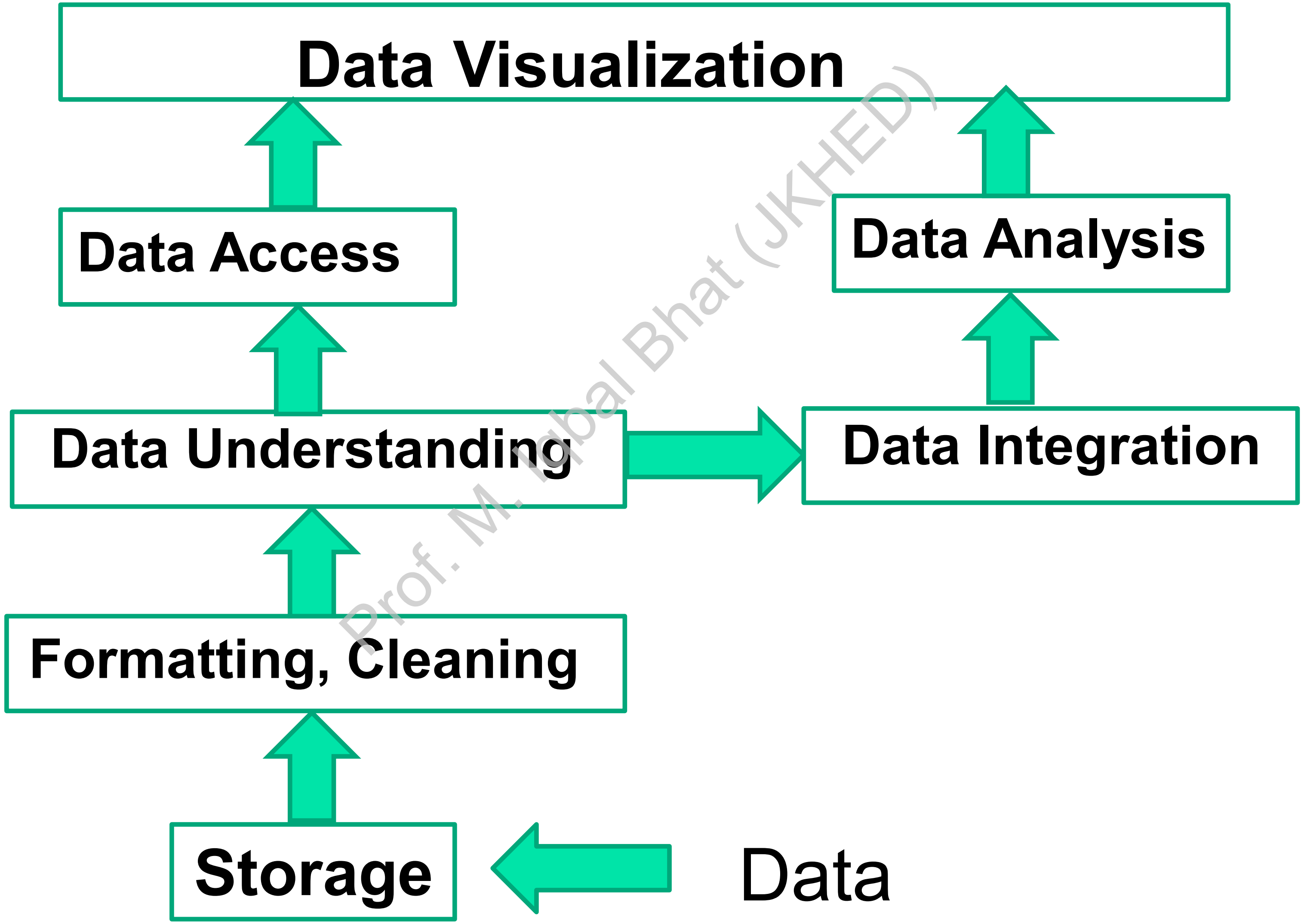
Why is “big data” a “big deal”?

- Government
- Private Sector
  - Walmart handles more than 1 million customer transactions every hour, which is imported into databases estimated to contain more than 2.5 petabytes of data
  - Facebook handles 40 billion photos from its user base
  - Falcon Credit Card Fraud Detection System protects 2.1 billion active accounts world-wide
- Science
  - Large Synoptic Survey Telescope will generate 140 Terabyte of data every 5 days
  - Biomedical computation like decoding human Genome and personalized medicine

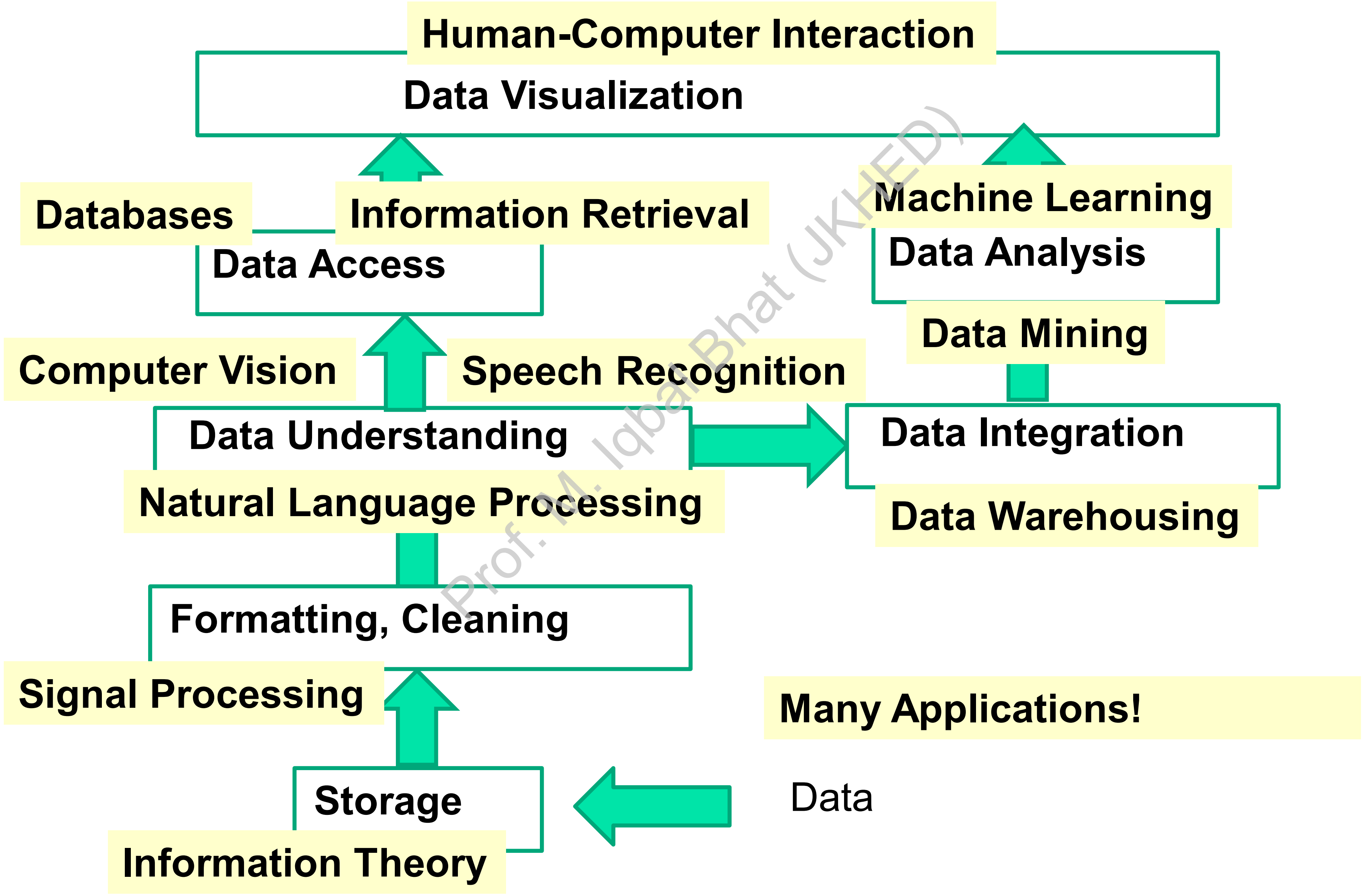
Lifecycle of Data: 4 "A"s



Computational View of Big Data



Big Data & Related Disciplines



Impact of Databases and Database Technology

- Businesses: Banking, Insurance, Retail, Transportation, Healthcare, Manufacturing
- Service industries: Financial, Real-estate, Legal, Electronic Commerce, Small businesses
- Education : Resources for content and Delivery
- More recently: Social Networks, Environmental and Scientific Applications, Medicine and Genetics
- Personalized applications: based on smart mobile devices

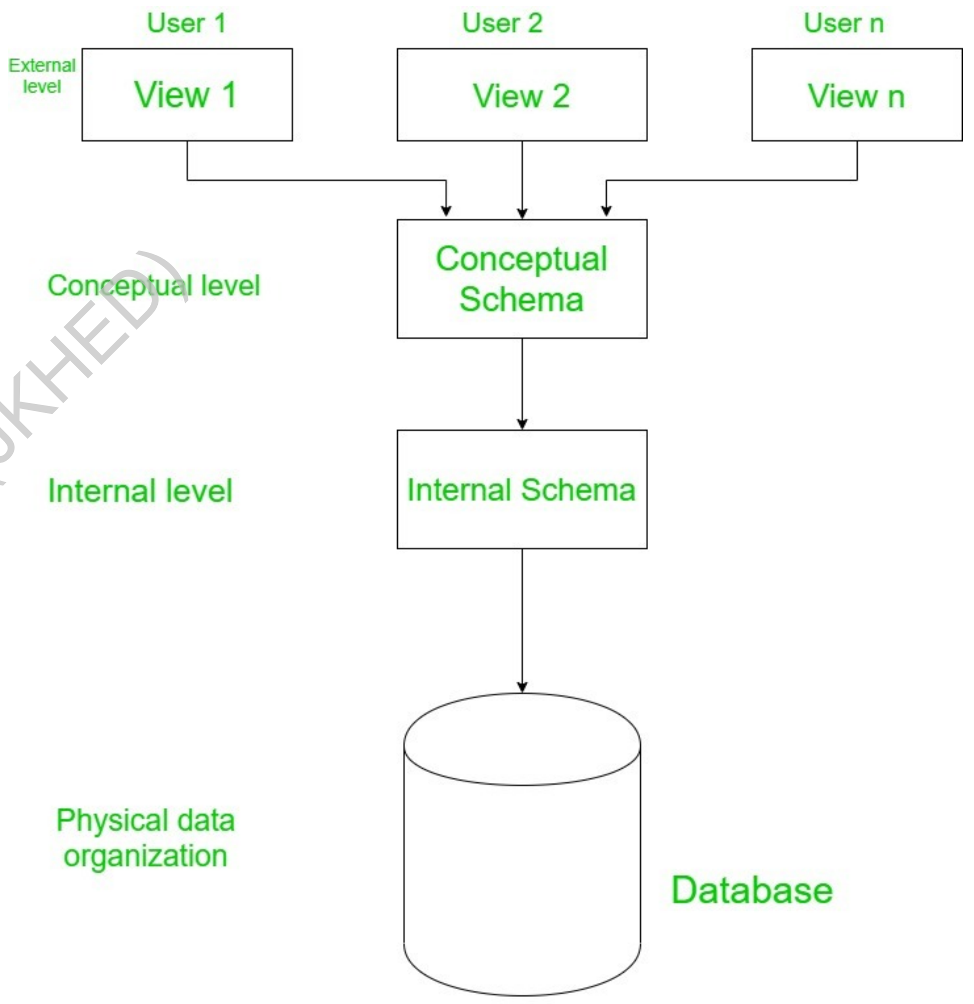
Architecture of

DBMS

The Three-Level ANSI-SPARC Architecture

The Three-Level ANSI-SPARC Architecture

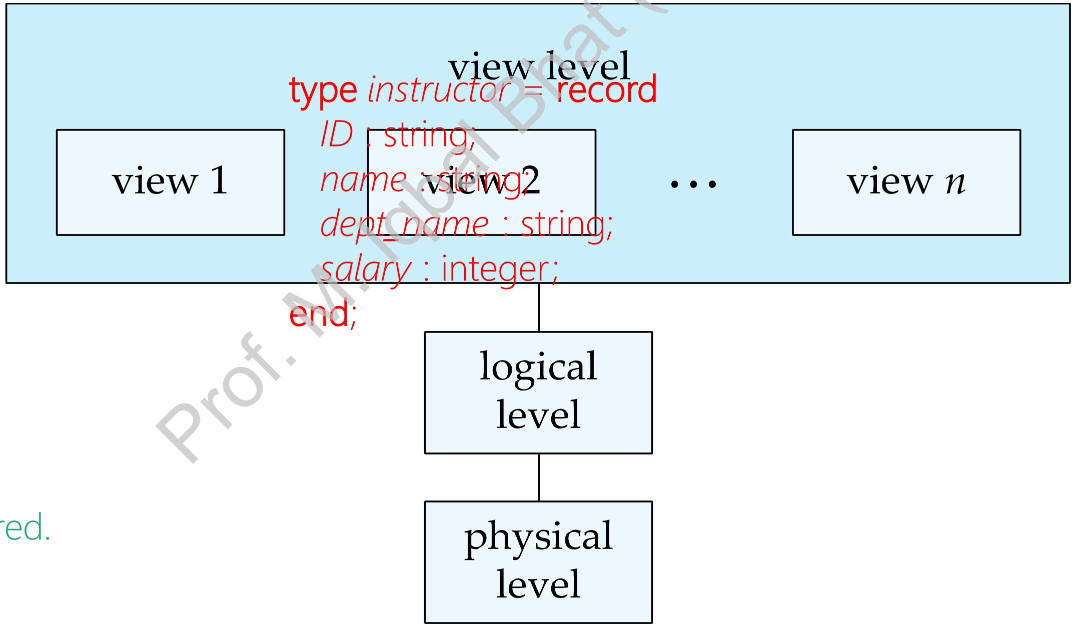
- In 1971, DBTG(DataBase Task Group) realized the requirement for a two-level approach having views and schema and afterward, in 1975, ANSI-SPARC realized the need for a three-level approach with the three levels of abstraction comprises of an external, a conceptual, and an internal level.



The ANSI-SPARC three-level architecture

A simplified architecture for a database system

View level: what application programs see; views can also hide information (such as an instructor's salary) for security purposes.



type *instructor* = record
ID : string;
name : string;
dept_name : string;
salary : integer;
end;

Physical level: describes how a record (e.g., instructor) is stored.

The Three-Level ANSI-SPARC Architecture

The ANSI-SPARC architecture is a three-level database architecture that provides a framework for organizing database management systems.

The three levels of the architecture are external level, conceptual level, and internal level.

The architecture is named after the American National Standards Institute (ANSI) and the Standards Planning And Requirements Committee (SPARC), which developed the framework in the late 1970s.

External Level

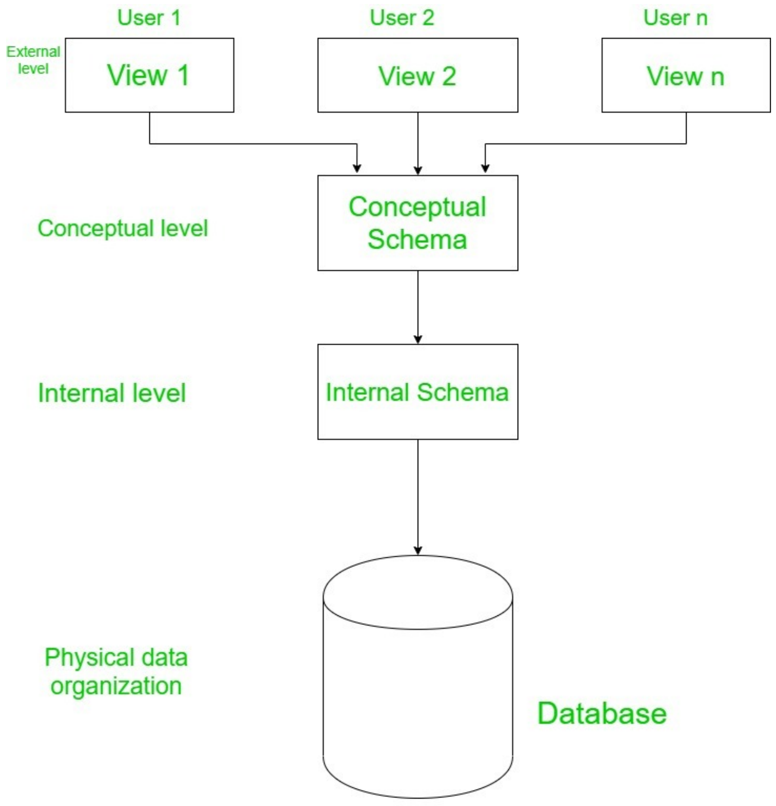
It is the view how the user views the database.

The data of the database that is relevant to that user is described at this level.

The external level consists of several different external views of the database.

In the external view only that entities, attributes, and relationships are included that the user wants. The different views may have different ways of representing the same data.

For example, one user may view name in the form (firstname, lastname), while another may view as (lastname, firstname).



The ANSI-SPARC three-level architecture

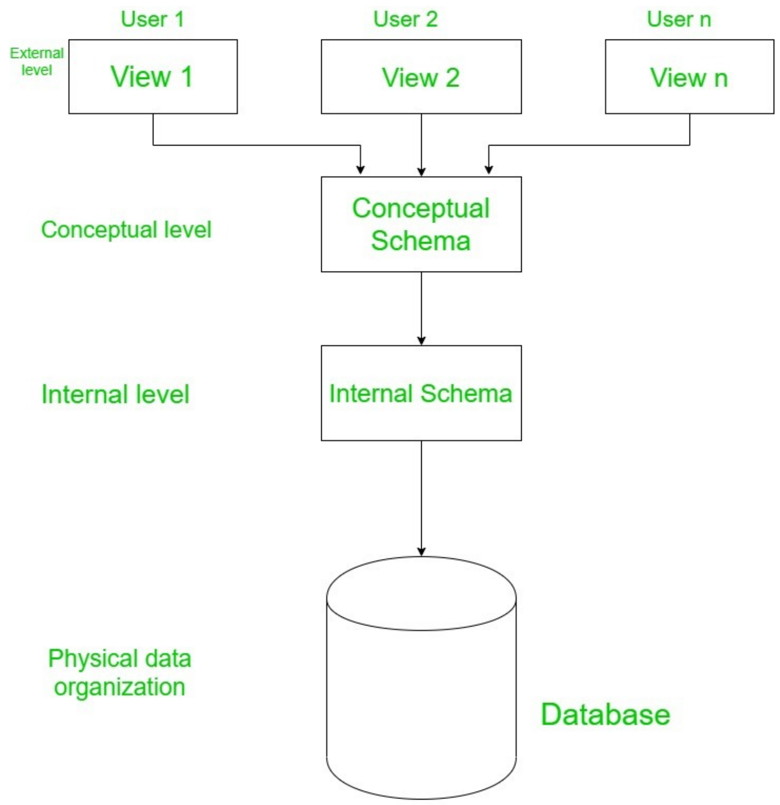
Conceptual level

It is the community view of the database and describes what data is stored in the database and represents the entities, their attributes, and their relationships.

It represents the semantic, security, and integrity information about the data.

The middle-level or the second-level in the three-level architecture is the conceptual level.

This level contains the logical structure of the entire database, it represents the complete view of the database that the organization demands independent of any storage consideration.



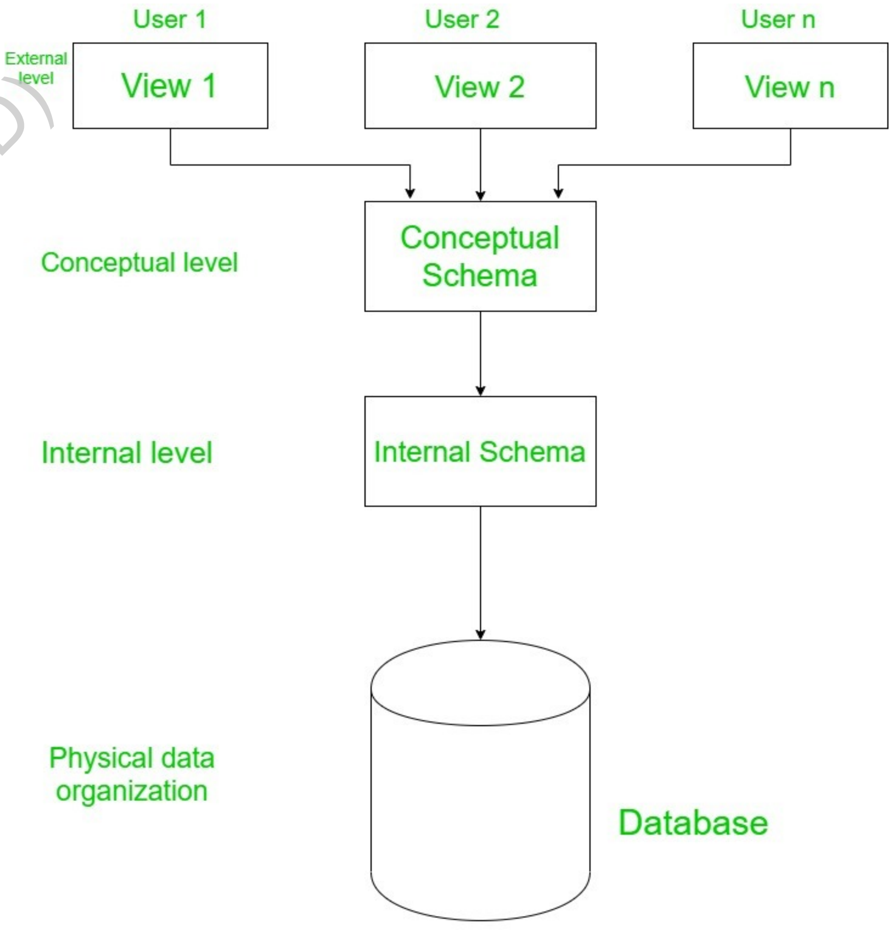
The ANSI-SPARC three-level architecture

Internal level

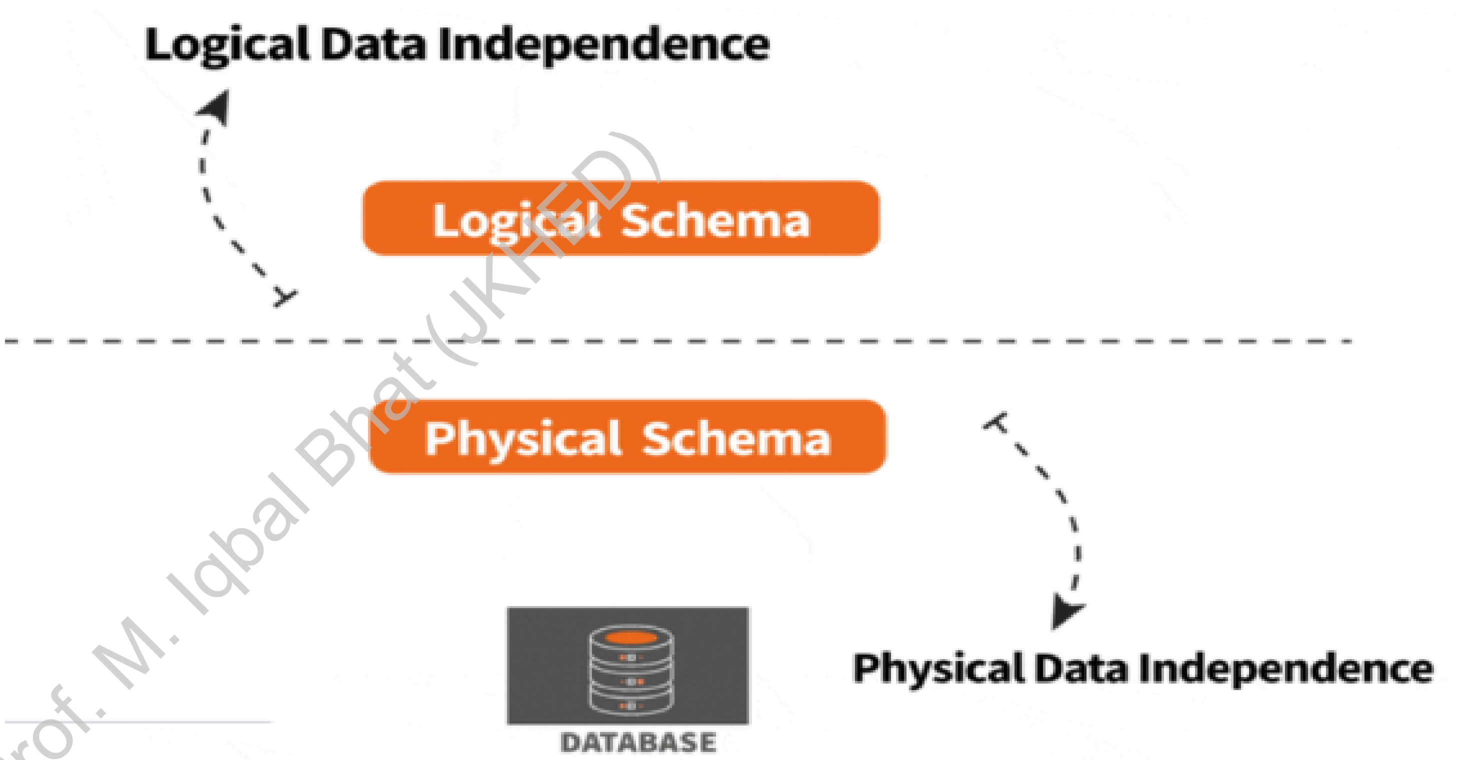
At the internal level, the database is represented physically on the computer.

It emphasizes the physical implementation of the database to do storage space utilization and to achieve the optimal runtime performance, and data encryption techniques.

It interfaces with the operating system to place the data on storage files and build the storage space, retrieve the data, etc.



The ANSI-SPARC three-level architecture



Two Levels of Data Independence

Data Independence

The ANSI-SPARC architecture provides two levels of data independence: logical data independence and physical data independence.

Logical data independence refers to the ability to modify the conceptual schema without affecting the external schema.

Physical data independence refers to the ability to modify the internal schema without affecting the conceptual schema or external schema.

Data independence allows changes to be made to the database without affecting the application programs that use the data.

Logical Data Independence

Logical data independence

Logical Data Independence:

- Logical data independence is the ability to modify the conceptual schema without affecting the external schema.
- This means that changes to the database structure should not require changes to be made to the application programs that access the data.

The Conceptual Schema:

- The conceptual schema is the global view of the entire database and represents the organization's requirements for data storage and retrieval.
- It defines the logical structure of the database and the relationships between data entities.
- The conceptual schema is the middle level in the ANSI-SPARC architecture.

The External Schema

- The external schema is the user's view of the data and how the data is presented to the user.
- It includes user-specific data and operations, such as queries, reports, and forms.
- The external schema is the highest level in the ANSI-SPARC architecture.

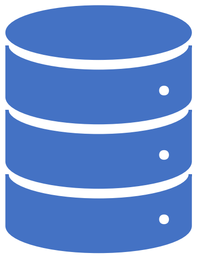
Benefits of Logical Data Independence:

Flexibility: Changes can be made to the database structure without affecting the application programs that use the data.

Reduced maintenance: With logical data independence, the need to modify application programs is reduced, which can save time and resources.

Improved scalability: Logical data independence allows the database to be modified and expanded as the organization's needs change over time.

Physical Data Independence



Physical Data Independence

Physical data independence is the ability to modify the internal schema without affecting the conceptual schema or external schema.

This means that changes to the physical storage of the data should not require changes to be made to the application programs that access the data.

The Internal Schema:

The internal schema defines how data is physically stored and organized on the storage media, such as disks.

It specifies the data storage format, access methods, and data compression techniques.

The internal schema is the lowest level in the ANSI-SPARC architecture.

Benefits of Physical Data Independence

Improved performance: With physical data independence, the internal schema can be modified to take advantage of faster storage media, such as SSDs or cloud storage.

Reduced maintenance: With physical data independence, changes to the physical storage of the data do not require modifications to the application programs that use the data, which can save time and resources.

Improved scalability: Physical data independence allows the database to be modified and expanded as the organization's needs change over time, without requiring changes to the application programs that use the data.

A simplified architecture for a database system

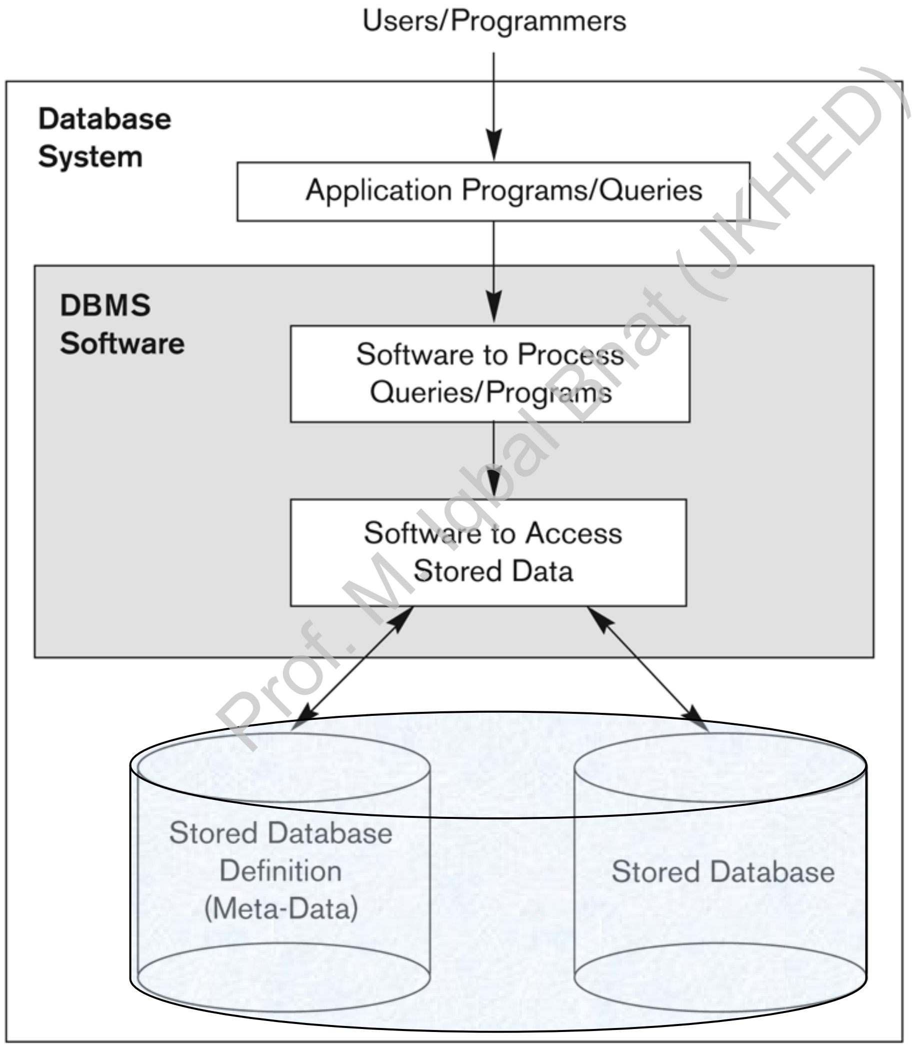
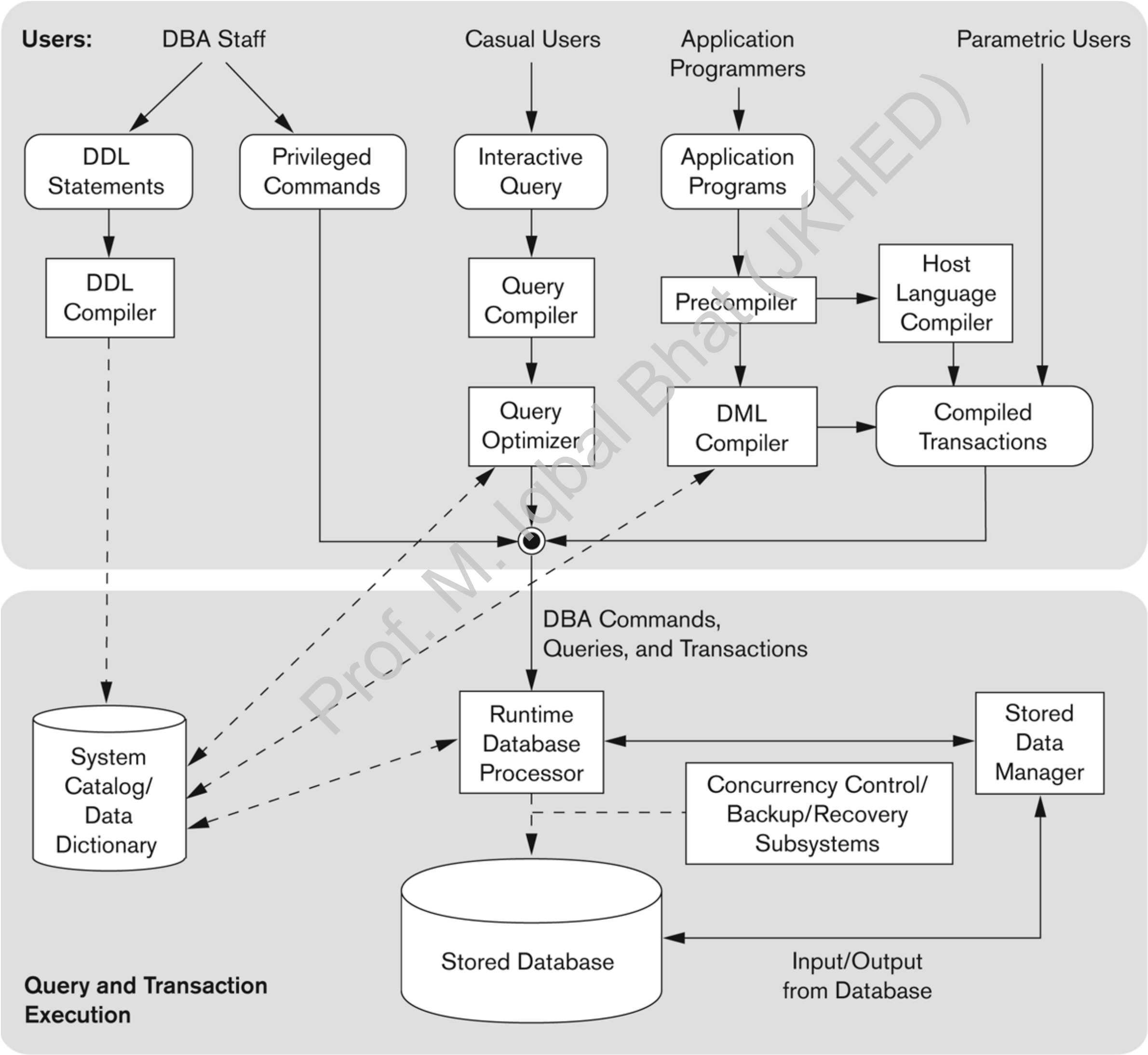
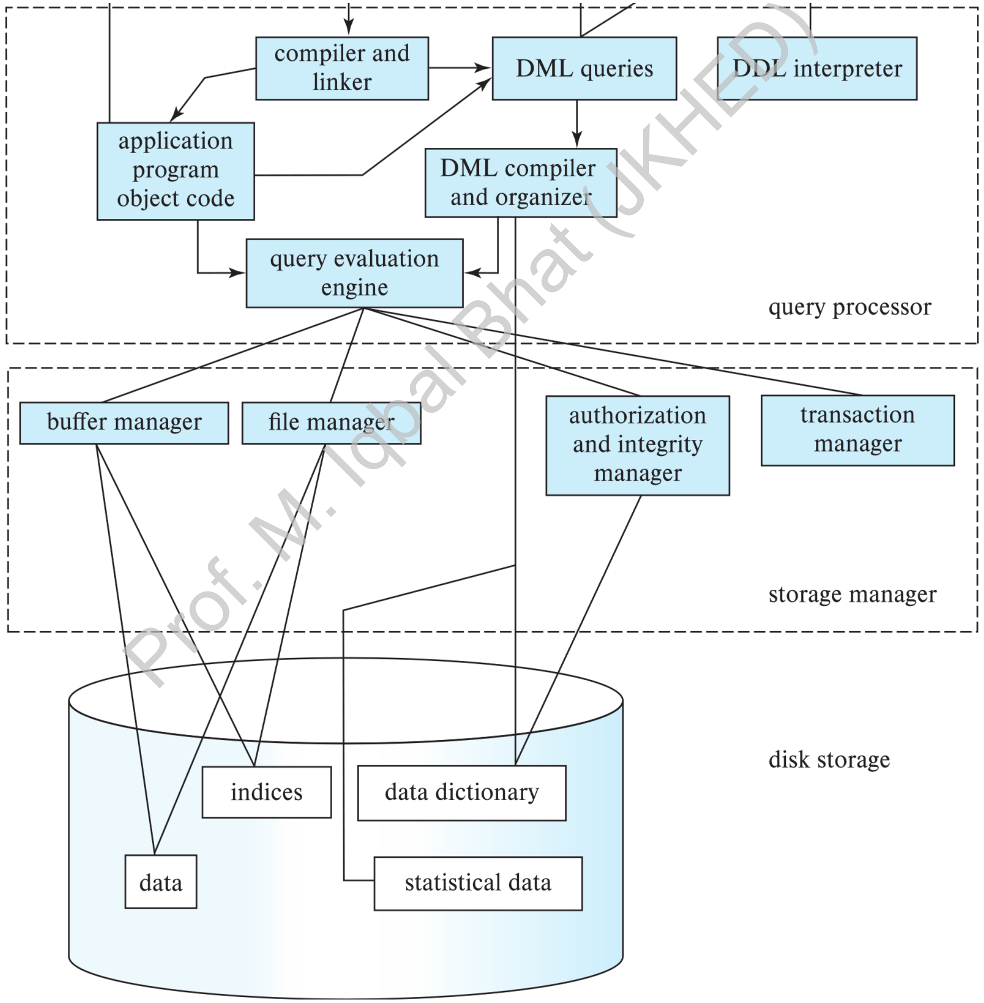


Figure 1.1
A simplified database system environment.

A simplified architecture for a database system



A simplified architecture for a database system



What a DBMS Facilitates

- *Define* a particular database in terms of its data types, structures, and constraints
- *Construct* or load the initial database contents on a secondary storage medium
- *Manipulating* the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- *Processing* and *sharing* by a set of concurrent users and application programs – yet, keeping all data valid and consistent

Other DBMS Functionalities

- DBMS may additionally provide:
  - Protection or security measures to prevent unauthorized access
  - “Active” processing to take internal actions on data
  - Presentation and visualization of data
  - Maintenance of the database and associated programs over the lifetime of the database application

Application Programs and DBMS

- Applications interact with a database by generating
  - Queries: that access different parts of data and formulate the result of a request
  - Transactions: that may read some data and "update" certain values or generate new data and store that in the database

Example of a Database (with a Conceptual Data Model)

- Mini-world for the example:
  - Part of a UNIVERSITY environment
- Some mini-world *entities*:
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (Academic) DEPARTMENTs
  - INSTRUCTORs

Example of a Database (with a Conceptual Data Model)

- Some mini-world *relationships*:
  - SECTIONs *are of specific* COURSEs
  - STUDENTs *take* SECTIONs
  - COURSEs *have prerequisite* COURSEs
  - INSTRUCTORs *teach* SECTIONs
  - COURSEs *are offered by* DEPARTMENTs
  - STUDENTs *major in* DEPARTMENTs
- Note: The above entities and relationships are typically expressed in a conceptual data model, such as the entity-relationship (ER) data or UML class model (see Chapters 3, 4)

Example of a Simple Database

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

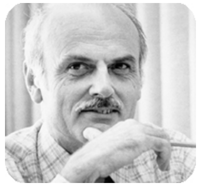
Figure 1.2
A database that stores student and course information.

The relational model

Columns

Rows

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000


E.F. “Ted” Codd

Main Characteristics of the Database Approach

- Self-describing nature of a database system:
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
  - The description is called **meta-data***.
  - This allows the DBMS software to work with different database applications.
- Insulation between programs and data:
  - Called **program-data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs
    - E.g., ADTs

Example of a Simplified Database Catalog

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

Main Characteristics of the Database Approach (continued)

- Data abstraction:
  - A **data model** is used to hide storage details and present the users with a conceptual view of the database.
  - Programs refer to the data model constructs rather than data storage details
- Support of multiple views of the data:
  - Each user may see a different view of the database, which describes **only** the data of interest to that user.

Main Characteristics of the Database Approach (continued)

- Sharing of data and multi-user transaction processing:
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *Concurrency control* within the DBMS guarantees that each transaction is correctly executed or aborted
  - *Recovery* subsystem ensures each completed transaction has its effect permanently recorded in the database
  - OLTP (Online Transaction Processing) is a major part of database applications; allows hundreds of concurrent transactions to execute per second.

Database Users

- Users may be divided into
  - Those who actually use and control the database content, and those who design, develop and maintain database applications (called "*Actors on the Scene*"), and
  - Those who design and develop the DBMS software and related tools, and the computer systems operators (called "*Workers Behind the Scene*").

Database Users – Actors on the Scene

- Actors on the scene
  - Database administrators
    - Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.
  - Database designers
    - Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

Database End Users

- Actors on the scene (continued)
  - End-users: They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
    - Casual: access database occasionally when needed
    - Naïve or parametric: they make up a large section of the end-user population.
      - They use previously well-defined functions in the form of "canned transactions" against the database.
      - Users of mobile apps mostly fall in this category
      - Bank-tellers or reservation clerks are parametric users who do this activity for an entire shift of operations.
      - Social media users post and read information from websites

Database End Users (continued)

- Sophisticated:
  - These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
  - Many use tools in the form of software packages that work closely with the stored database.
- Stand-alone:
  - Mostly maintain personal databases using ready-to-use packaged applications.
  - An example is the user of a tax program that creates its own internal database.
  - Another example is a user that maintains a database of personal photos and videos.

Database Users – Actors on the Scene (continued)

- System analysts and application developers
  - System analysts: They understand the user requirements of naïve and sophisticated users and design applications including canned transactions to meet those requirements.
  - Application programmers: Implement the specifications developed by analysts and test and debug them before deployment.
  - Business analysts: There is an increasing need for such people who can analyze vast amounts of business data and real-time data ("Big Data") for better decision making related to planning, advertising, marketing etc.

Database Users – Actors behind the Scene

- System designers and implementors: Design and implement DBMS packages in the form of modules and interfaces and test and debug them. The DBMS must interface with applications, language compilers, operating system components, etc.
- Tool developers: Design and implement software systems called tools for modeling and designing databases, performance monitoring, prototyping, test data generation, user interface creation, simulation etc. that facilitate building of applications and allow using database effectively.
- Operators and maintenance personnel: They manage the actual running and maintenance of the database system hardware and software environment.

Advantages of Using the Database Approach

- Controlling redundancy in data storage and in development and maintenance efforts.
  - Sharing of data among multiple users.
- Restricting unauthorized access to data. Only the DBA staff uses privileged commands and facilities.
- Providing persistent storage for program Objects
  - E.g., Object-oriented DBMSs make program objects persistent– see Chapter 12.
- Providing storage structures (e.g. indexes) for efficient query processing – see Chapter 17.

Advantages of Using the Database Approach (continued)

- Providing optimization of queries for efficient processing
- Providing backup and recovery services
- Providing multiple interfaces to different classes of users
- Representing complex relationships among data
- Enforcing integrity constraints on the database
- Drawing inferences and actions from the stored data using deductive and active rules and triggers

Additional Implications of Using the Database Approach

- Potential for enforcing standards:
  - Standards refer to data item names, display formats, screens, report structures, meta-data (description of data), Web page layouts, etc.
- Reduced application development time:
  - Incremental time to add each new application is reduced.

Additional Implications of Using the Database Approach (continued)

- Flexibility to change data structures:
  - Database structure may evolve as new requirements are defined.
- Availability of current information:
  - Extremely important for on-line transaction systems such as shopping, airline, hotel, car reservations.
- Economies of scale:
  - Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

Historical Development of Database Technology

- Early database applications:
  - The *Hierarchical* and *Network* models were introduced in mid 1960s and dominated during the seventies.
  - A bulk of the worldwide database processing still occurs using these models, particularly, the hierarchical model using IBM's IMS system.
- Relational model-based systems:
  - Relational model was originally introduced in 1970, was heavily researched and experimented within IBM Research and several universities.
  - Relational DBMS Products emerged in the early 1980s.

Historical Development of Database Technology (continued)

- Object-oriented and emerging applications:
  - Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications.
    - Their use has not taken off much
  - Many relational DBMSs have incorporated object database concepts, leading to a new category called *object-relational* DBMSs (ORDBMSs)
  - *Extended relational* systems add further capabilities (e.g. for multimedia data, text, XML, and other data types)

Historical Development of Database Technology (continued)

- Data on the Web and e-commerce applications:
  - Web contains data in HTML (Hypertext markup language) with links among pages
  - Has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language) (see Ch. 13).
  - Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database (see Ch. 11).
    - Also allow database updates through Web pages

Extending Database Capabilities (1)

- New functionality is being added to DBMSs in the following areas:
  - Scientific applications – physics, chemistry, biology, genetics
  - Spatial: weather, earth and atmospheric sciences and astronomy
  - XML (eXtensible Markup Language)
  - Image storage and management
  - Audio and video data management
  - Time series and historical data management
- The above gives rise to *new research and development* in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.

When not to use a DBMS

- Main inhibitors (costs) of using a DBMS:
  - High initial investment and possible need for additional hardware
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions
- When a DBMS may be unnecessary:
  - If the database and applications are simple, well defined, and not expected to change
  - If access to data by multiple users is not required
- When a DBMS may be infeasible
  - In embedded systems where a general-purpose DBMS may not fit in available storage

When not to use a DBMS

- When no DBMS may suffice:
  - If there are stringent real-time requirements that may not be met because of DBMS overhead (e.g., telephone switching systems)
  - If the database system is not able to handle the complexity of data because of modeling limitations (e.g., in complex genome and protein databases)
  - If the database users need special operations not supported by the DBMS (e.g., GIS and location-based services).

Chapter Summary

- Types of databases and database applications
- Basic definitions
- Typical DBMS functionality
- Example of a database (UNIVERSITY)
- Main characteristics of the database Approach
- Types of database users
- Advantages of using the database approach
- Historical development of database technology
- Extending database capabilities
- When not to use databases