DBMS Architecture

Dr. Geeta Kasana
Assistant Professor
Thapar Institute of Engineering and Technology
Patiala

DBMS Architecture

The design of a DBMS depends on its a**rchitecture**.

DBMS architecture helps in design, development, implementation and maintenance of a database.

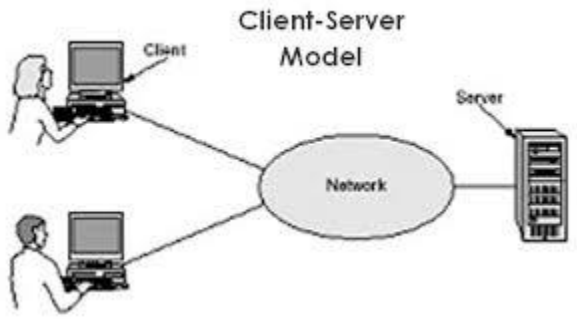
A database stores critical information for a business. Selecting the correct Database Architecture helps in quick and secure access to this data.

DBMS architecture depends upon

- The underlying computer system on which database system runs
- How users are connected to the database to get their request done

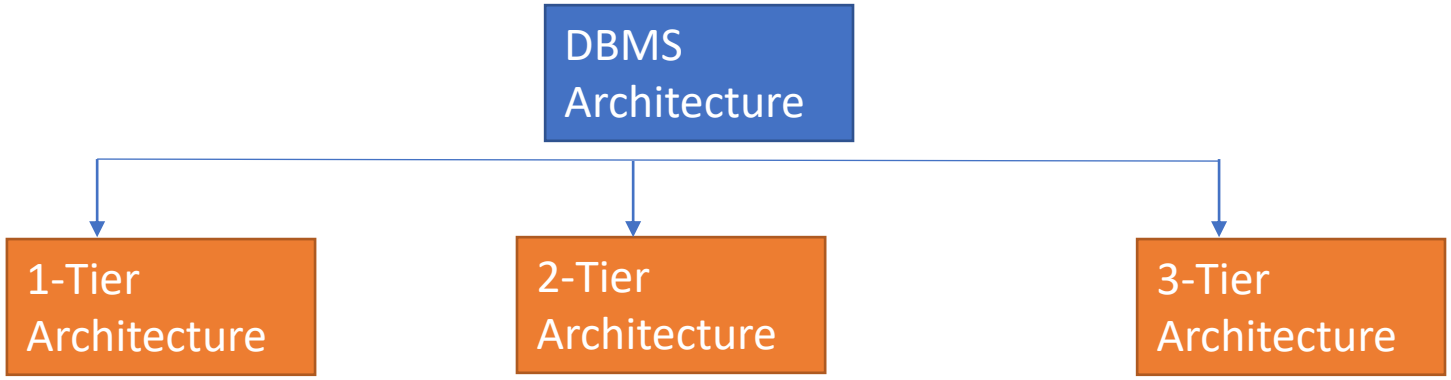
Client /Server Architecture

- The basic client server architecture is used to deal with a large number of PCs , web servers and other components that are connected with networks.
- The Client/Server architecture consists of many PCs and a workstation which are connected via the networks. Client request the server for a service. Server provides the requested service to the client.



Types of DBMS Architecture

Database architecture can be seen as single tier or multi-tier



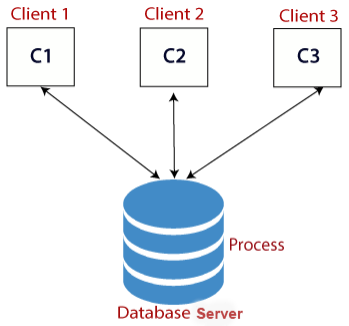
Single or 1-tier Architecture

Single Tier Architecture

- In Single or 1-tier architecture , the database is directly available to the user. The Client Server and Database all reside on the same machine. i.e user can directly sit on the DBMS and uses it.
- Any changes done here will be directly be done on the database itself. It doesn't use or provide handy tool for end users.
- No Network connection is required to perform the action on the database.
- 1-tier architecture is used
  - Where data does not change frequently and where no multiple user is accessing the system.
  - For development of the local application, where programmers can directly communicate with the database for the quick responses
  - Such architecture is rarely used in production.

2-Tier Architecture

- It is same as client-server
- In 2-tier architecture, applications on the client end can directly communicate with database at the server side.
- An Application Programming Interfaces(APIs) like ODBC and JDBC are used by client side program to call the DBMS.
- The user interfaces and application programs are run on the client side.
- To communicate with the DBMS, client side application establish a connection with the server side.
- The server side is responsible to provide the functionalities like query processing and transaction management.



2-tier Architecture

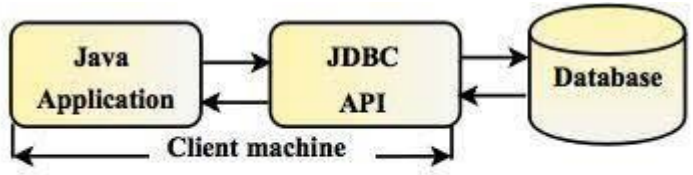
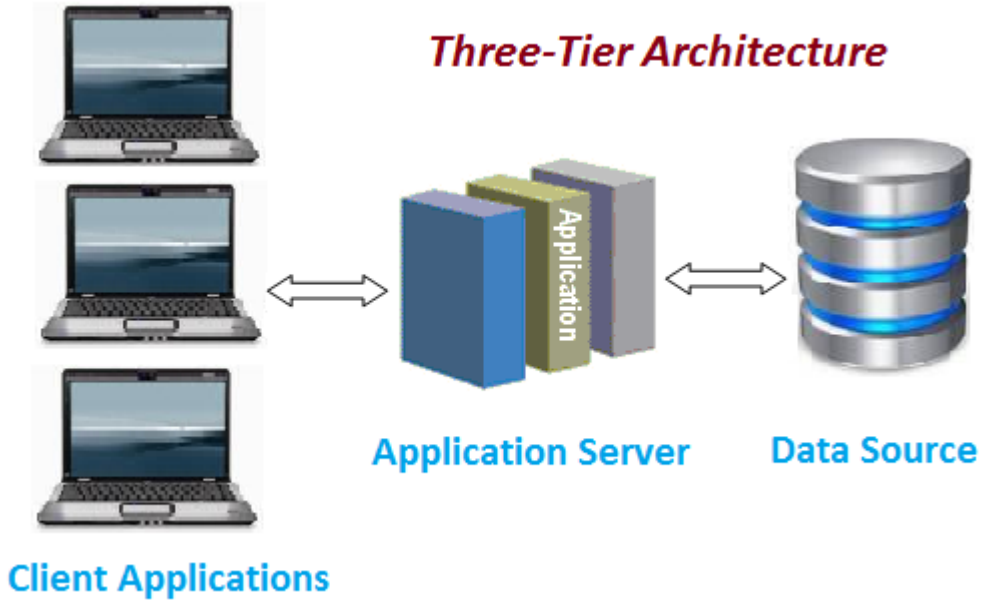


Fig: Two-tier Architecture of JDBC

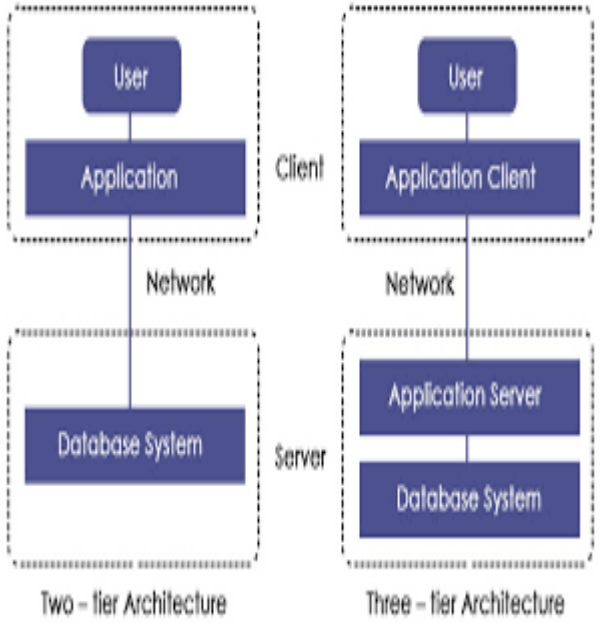
2-Tier Architecture

- The 2- Tier architecture is used inside any organization, where multiple clients accessing the database server directly.
- Eg: Railway reservation from counter, where clerk as a client accesses the railway server directly.
- Advantages:
  - Direct and Faster Communication
  - Maintenance and understanding is easier
  - Compatible with existing system
- Disadvantages:
  - Scalability: i.e It gives poor performance when there is large number of users
  - Less secure as client can access the server directly.

3-Tier Architecture



3-Tier Architecture



- The 3- Tier architecture contains another layer of Application Server between the client & server. In this architecture, client can't directly communicate with the server.
- The application on the client –end interacts with application server which further communicates with database system and then the query processing and transaction management takes place.
- The intermediate layer of Application Server acts as a medium for exchange of partially processed data between server and client.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application

3-Tier Architecture

- The 3- Tier architecture is used in large web applications
- It is the most popular DBMS architecture
- Advantages:
  - Enhanced scalability: due to distributed deployment of application servers. Now , individual connections need not ne maintained between client and server.
  - Data Integrity is maintained : Since there is middle layer between the client and server, data corruption can be avoided/removed.
  - Security is improved: This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.
  - Disadvantage:
  - Increased Complexity:

Schemas versus Instances

- Database Schema:
  - The ***description*** of a database.
  - Includes descriptions of the database structure, data types, and the constraints on the database.
- Schema Diagram:
  - An ***illustrative*** display of (most aspects of) a database schema.
- Schema Construct:
  - A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.

Schemas versus Instances

- Database State:
  - The actual data stored in a database at a ***particular moment in time***. This includes the collection of all the data in the database.
  - Also called database instance (or occurrence or snapshot).
    - The term *instance* is also applied to individual database components, e.g. *record instance, table instance, entity instance*

Database Schema vs. Database State

- Distinction
  - The ***database schema*** changes very infrequently.
  - The ***database state*** changes every time the database is updated.

- **Schema** is also called **intension**.
- **State** is also called **extension**

Example of a Database Schema

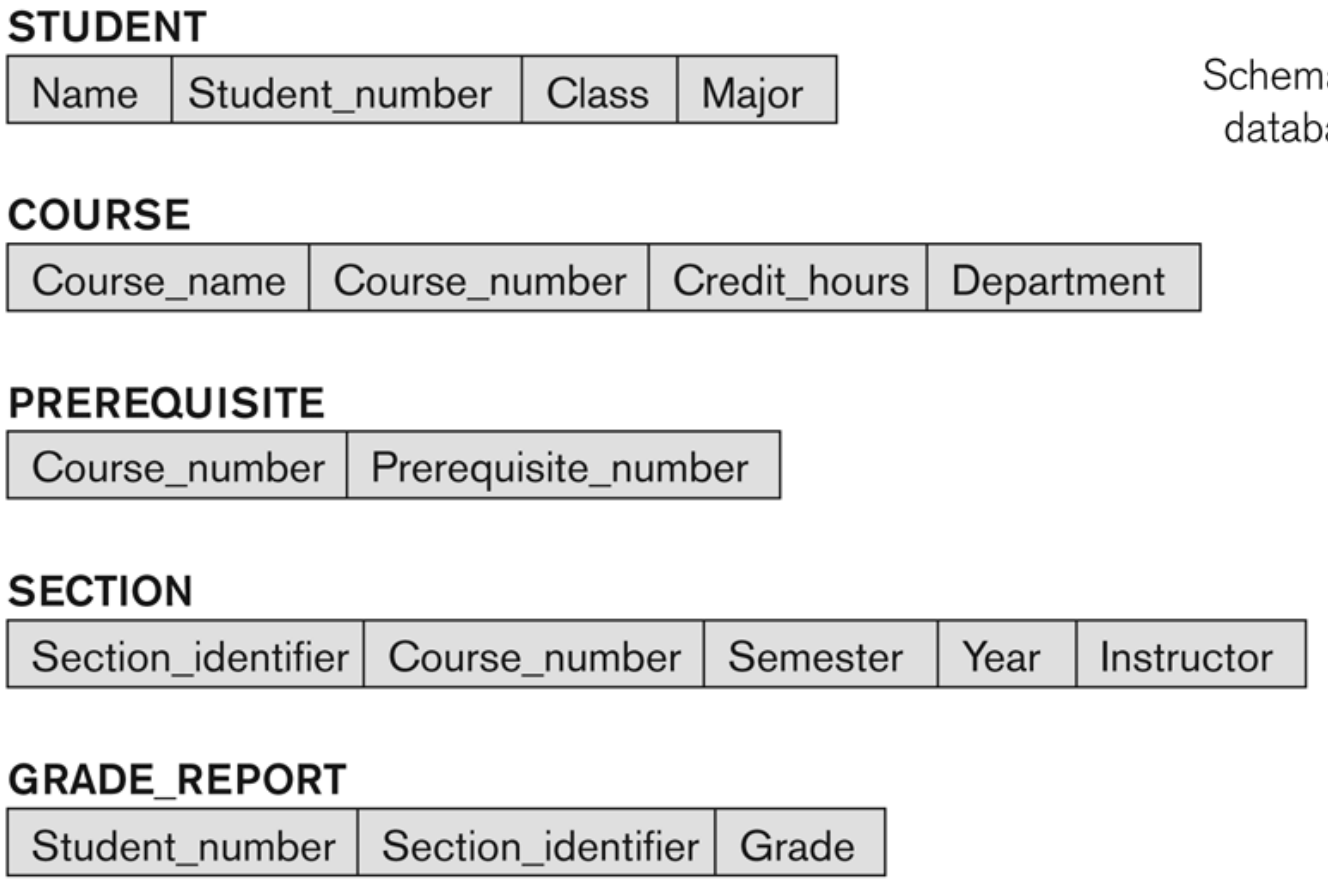


Figure 2.1
Schema diagram for the database in Figure 1.2.

Example of a database state

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

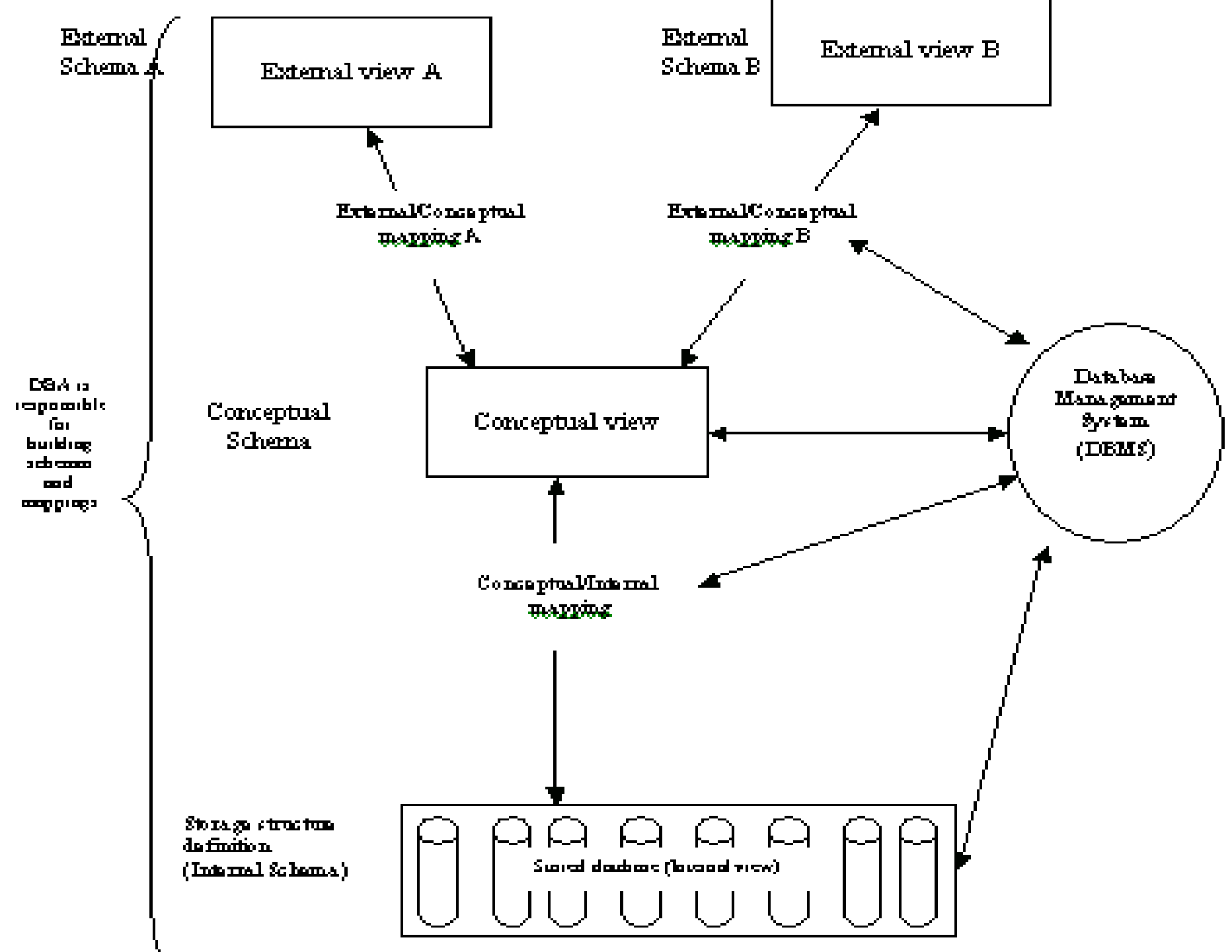
PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

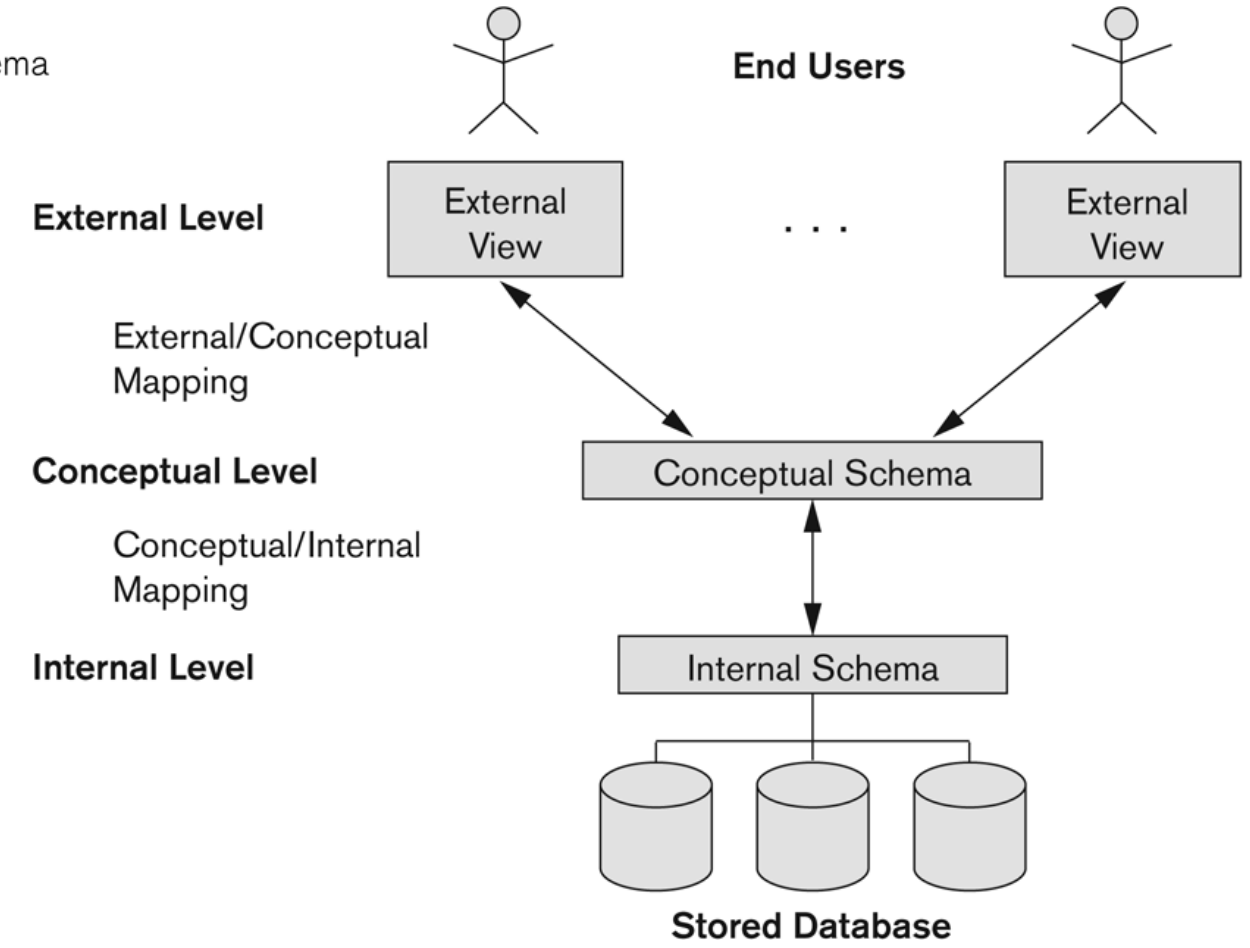
Three-Schema Architecture

- Proposed to support DBMS characteristics of:
  - **Program-data independence.**
  - Support of **multiple views** of the data.
- Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization


External Schema A
External view A
External Schema B
External view B
External/Conceptual mapping A
External/Conceptual mapping B
DBA is responsible for building schemas and mappings
Conceptual Schema
Conceptual view
Database Management System (DBMS)
Conceptual/Internal mapping
Storage structure definition (Internal Schema)
Stored database (Internal view)

Three-Schema Architecture

Figure 2.2
The three-schema architecture.



Three-Schema Architecture

External Level or View level

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

• For example, one user may view dates in the form (day, month, year), while another may view dates as (year, month, day).

Three-Schema Architecture

Conceptual Level or Logical level

•It is the community view of the database.

•This level describes what data is stored in the database and the relationships among the data.

•It represents:

❑ All entities, their attributes, and their relationships;

❑ The constraints on the data;

❑Security and integrity information.

Three-Schema Architecture

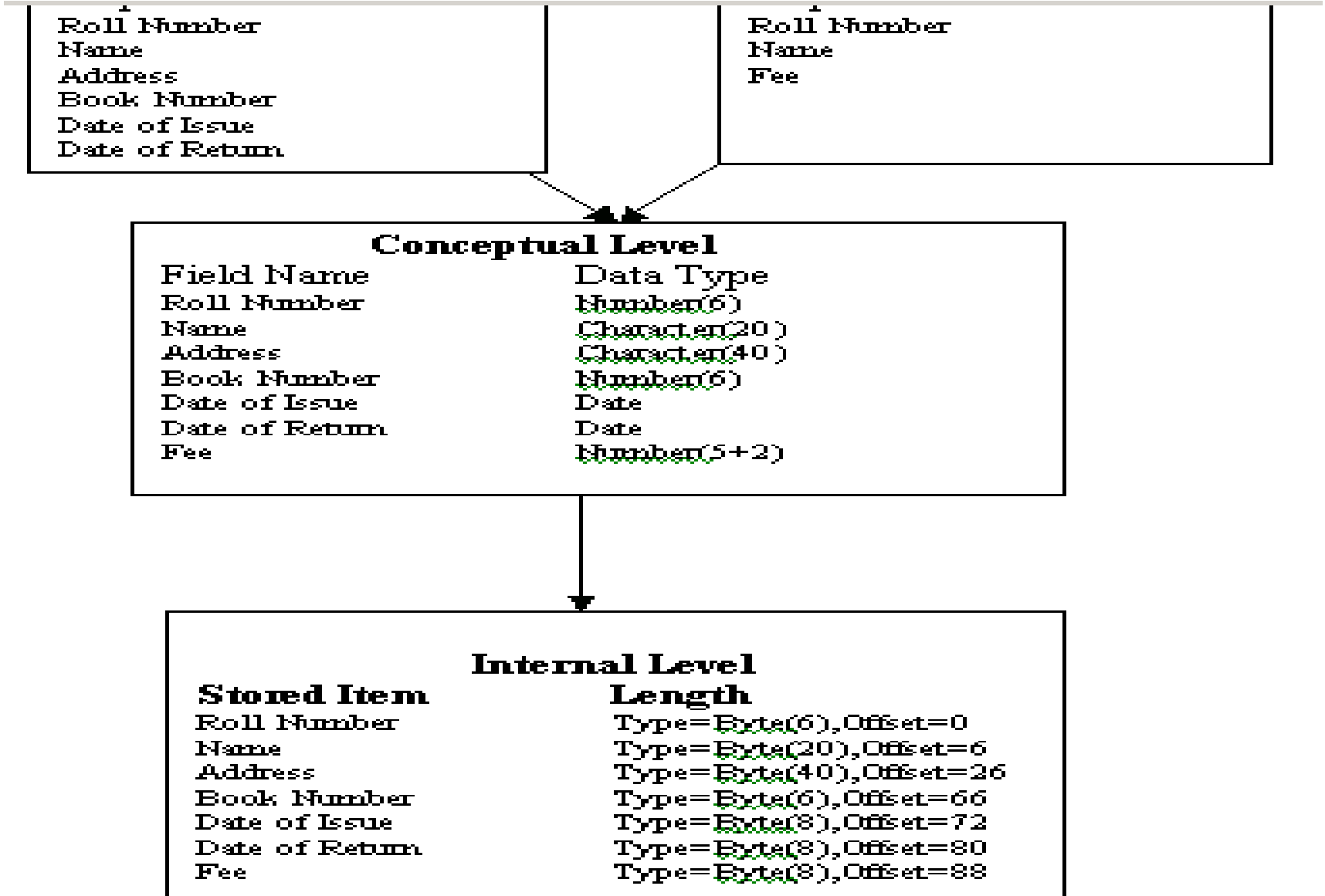
Internal Level or Storage level

- It is the physical representation of the database on the computer.
- This level describes how the data is stored in the database.

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

Three-Schema Architecture



Three-Schema Architecture

Mappings among schema levels are needed to transform requests and data.

- Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
- Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)

Conceptual/Internal Mapping

- Conceptual schema is related to the internal schema by the conceptual/internal mapping. This enables the DBMS to find the actual record or combination of records in physical storage that constitute a logical record in conceptual schema.

Data Independence

- **Logical Data Independence:**
  - The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.
- **Physical Data Independence:**
  - The capacity to change the internal schema without having to change the conceptual schema.
  - For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance

Metadata or Data Dictionary

A metadata (also called the data dictionary) is the data about the data. Data dictionary may be either active or passive.

An active data dictionary (also called integrated data dictionary) is managed automatically by the database management software.

The passive data dictionary (also called non-integrated data dictionary) is the one used only for documentation purposes.

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
  - **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.

Categories of 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.
      - Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.

Categories of End-users

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 a tax program user that creates its own internal database.

Another example is a user that maintains an address book

Simplified database system environment

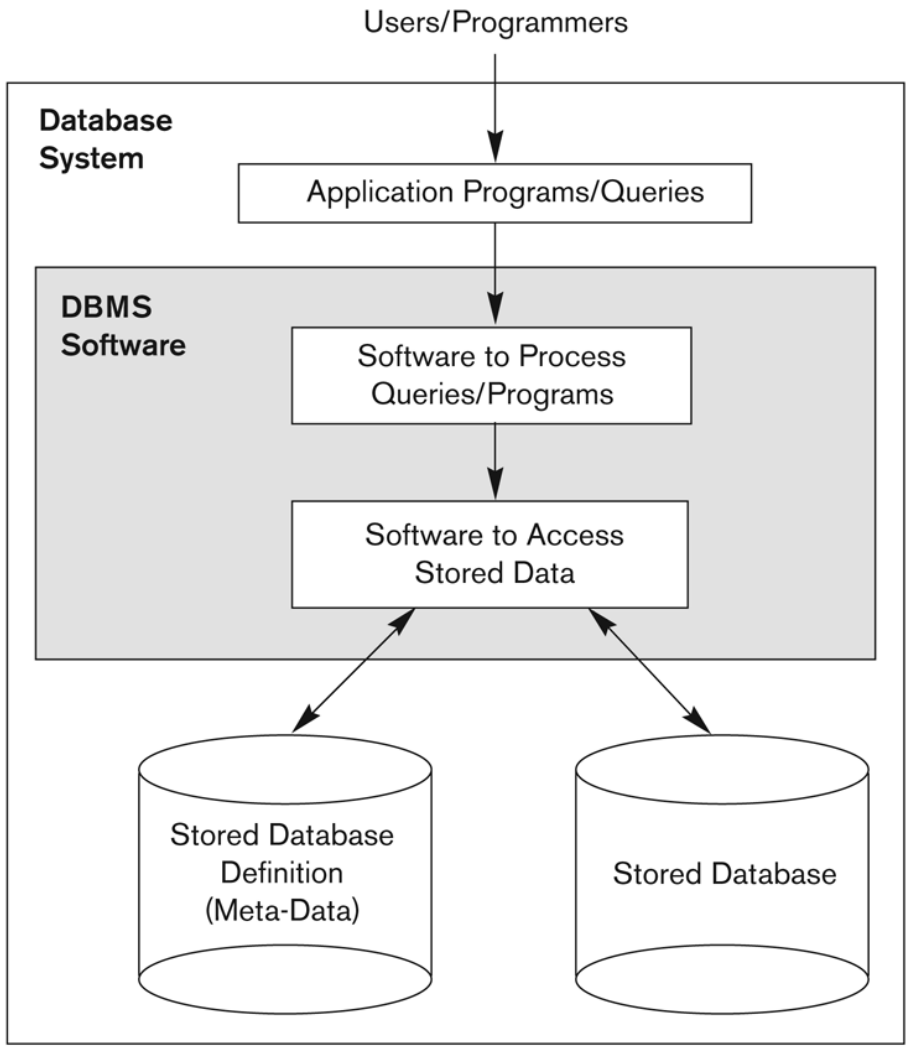
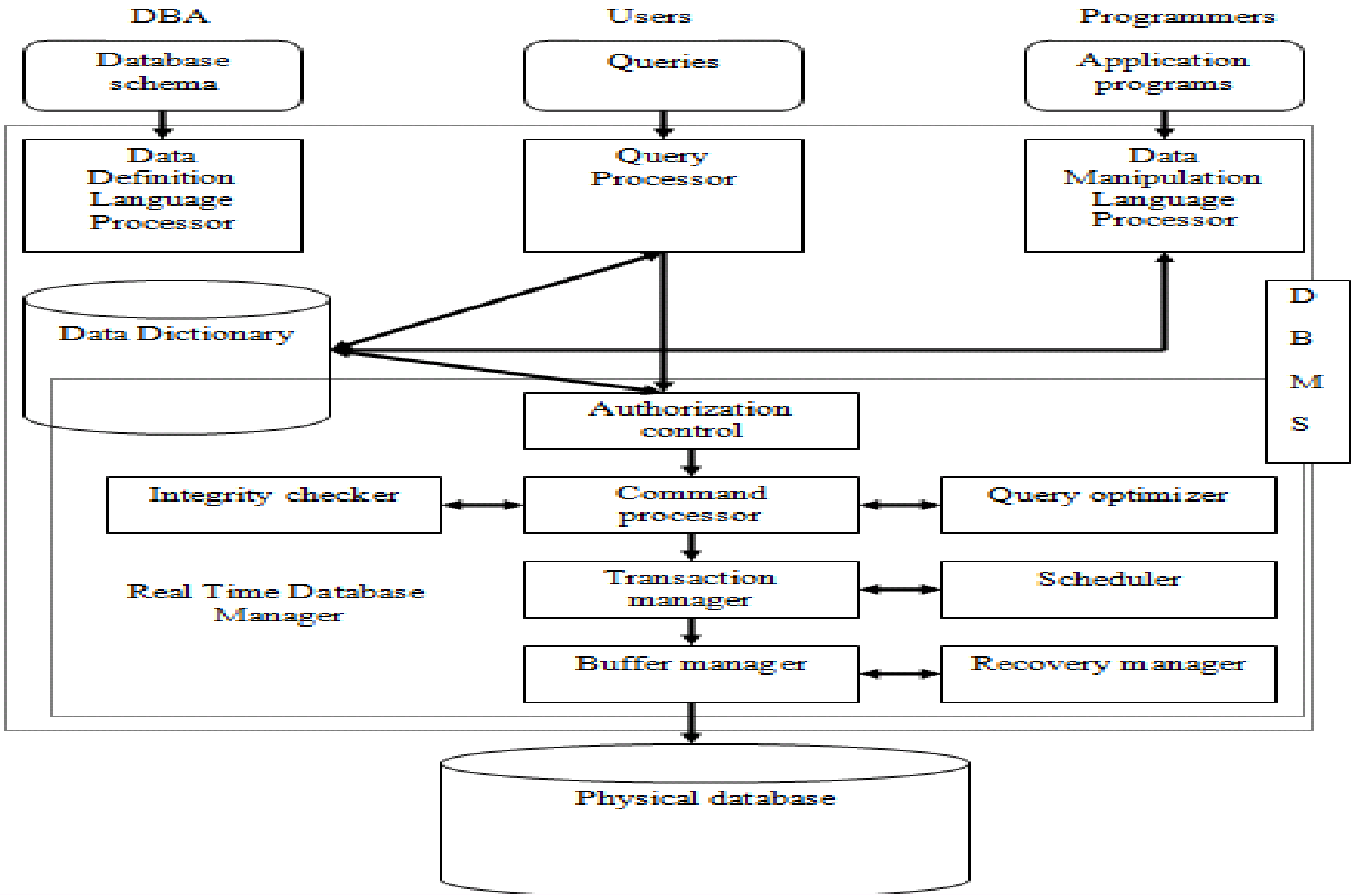


Figure 1.1
A simplified database system environment.

Components of a DBMS




Thank
you