KOLEJ POLY-TECH MARA

CHAPTER 2
AN ARCHITECTURE FOR DATABASE SYSTEM

- Data abstraction levels
- Mapping
- Instances & schema
- Back end vs. Front end

DEGREES OF DATA ABSTRACTION

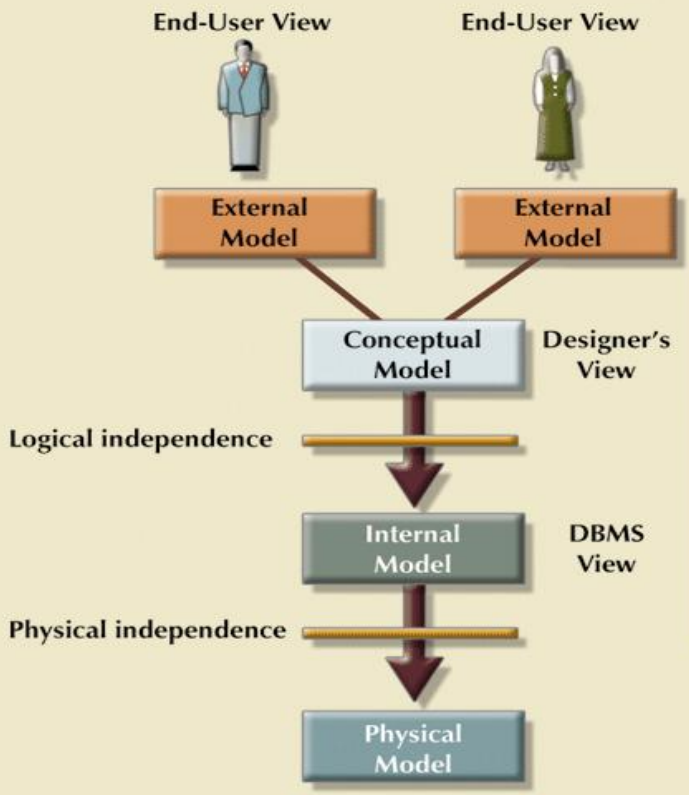
- Way of classifying data models
- Many processes begin at high level of abstraction and proceed to an ever-increasing level of detail
- Designing a usable database follows the same basic process

DEGREES OF DATA ABSTRACTION (CONTINUED)

- American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC)
  - Defined a framework for data modeling based on degrees of data abstraction(1970s):
    - External
    - Conceptual
    - Internal

DEGREES OF DATA ABSTRACTION (CONTINUED)

FIGURE 2.9 Data abstraction levels



Degree of Abstraction		Characteristics
High	ER	Hardware-independent Software-independent
Medium	Relational Object-Oriented	Hardware-independent Software-dependent
Low	Network Hierarchical	Hardware-dependent Software-dependent

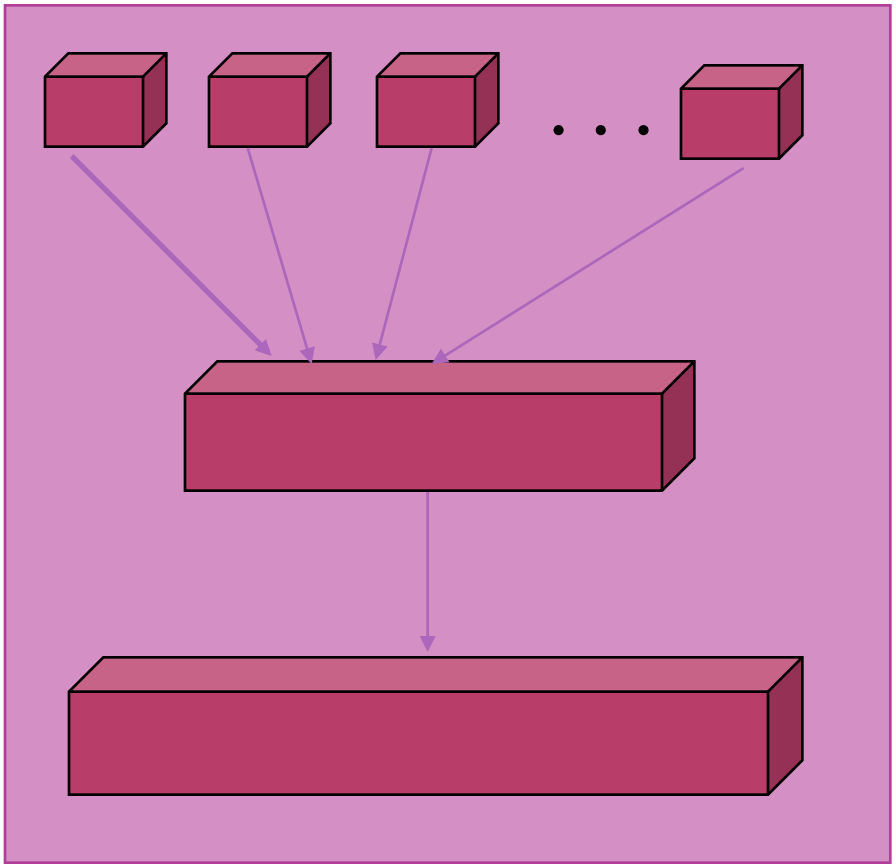
ANSI DATABASE SYSTEM ARCHITECTURE

External Level

External Level
(Individual user view)

Conceptual Level
Community user view

Internal Level
(physical storage view



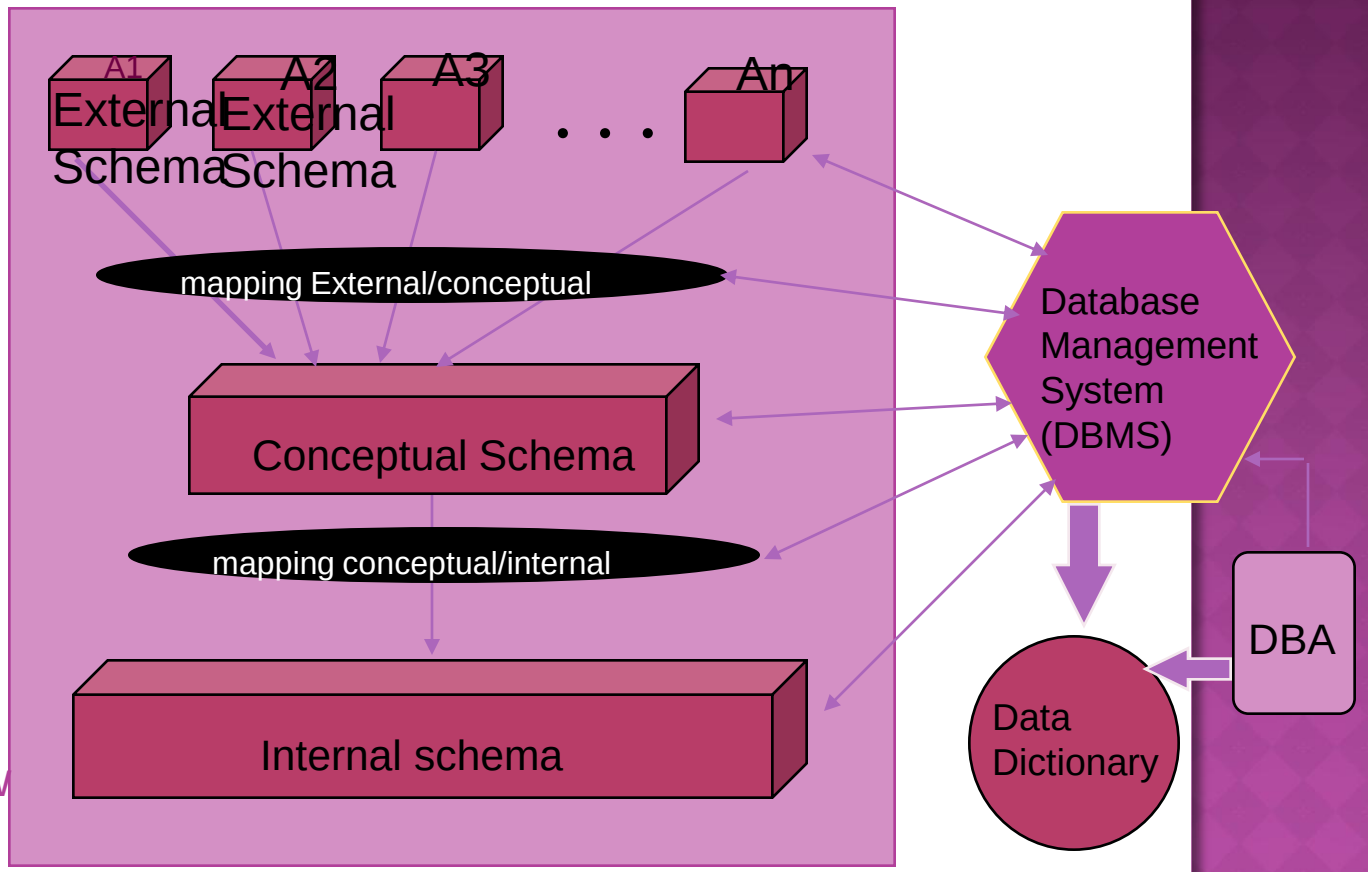
ANSI DATABASE SYSTEM ARCHITECTURE

External Level

External Level
(Individual user view)

Conceptual Level
Community user view

Internal Level
(physical storage view)



LEVELS OF ABSTRACTION

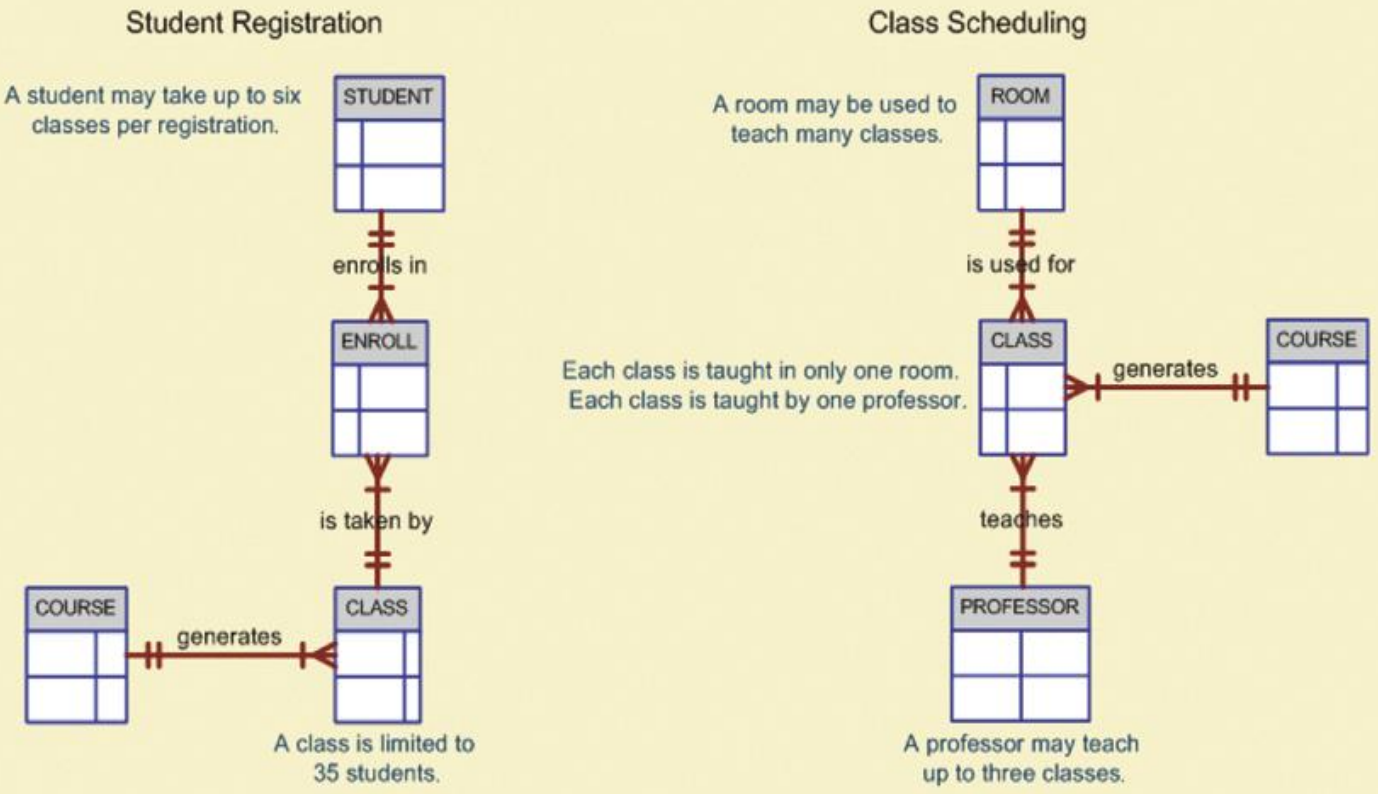
- **Physical level/Internal level:** describes how a record (e.g., customer) is stored.
- **Logical level/Conceptual level:** describes data stored in database, and the relationships among the data.

```
type customer = record
        customer_id : string;
        customer_name : string;
        customer_street : string;
        customer_city : integer;
        end;
```

- **View level/External level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

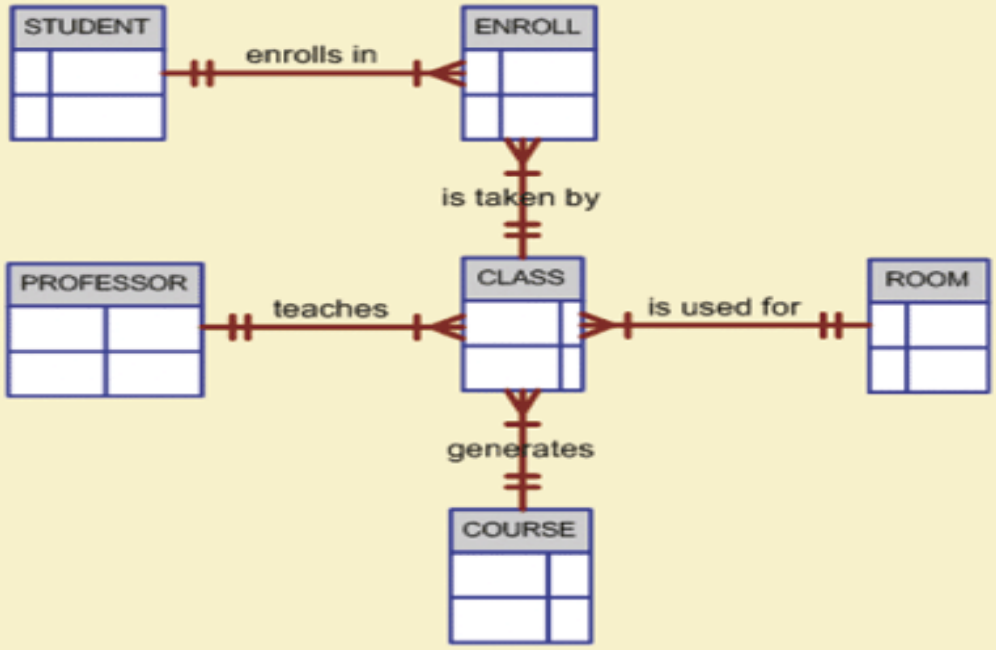
THE EXTERNAL MODEL (CONTINUED)

FIGURE 2.10 External models for Tiny College



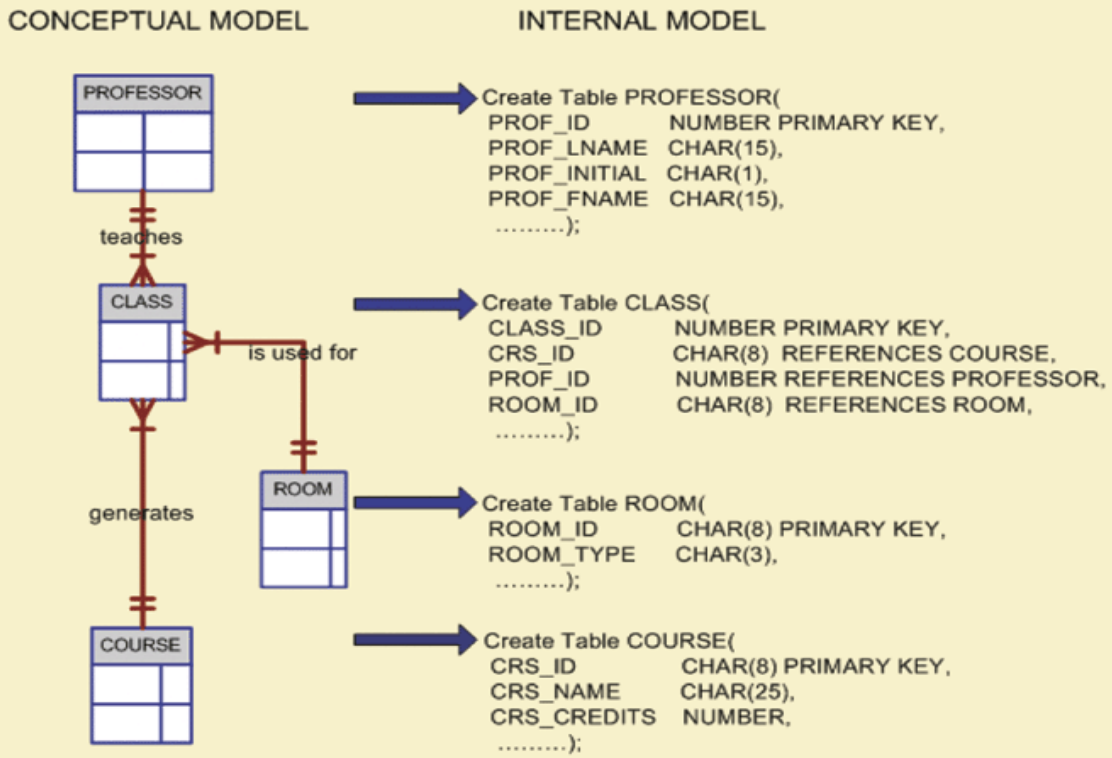
THE CONCEPTUAL MODEL (CONTINUED)

FIGURE 2.11 Conceptual model for Tiny College



THE INTERNAL MODEL (CONTINUED)

FIGURE 2.12 An internal model for Tiny College



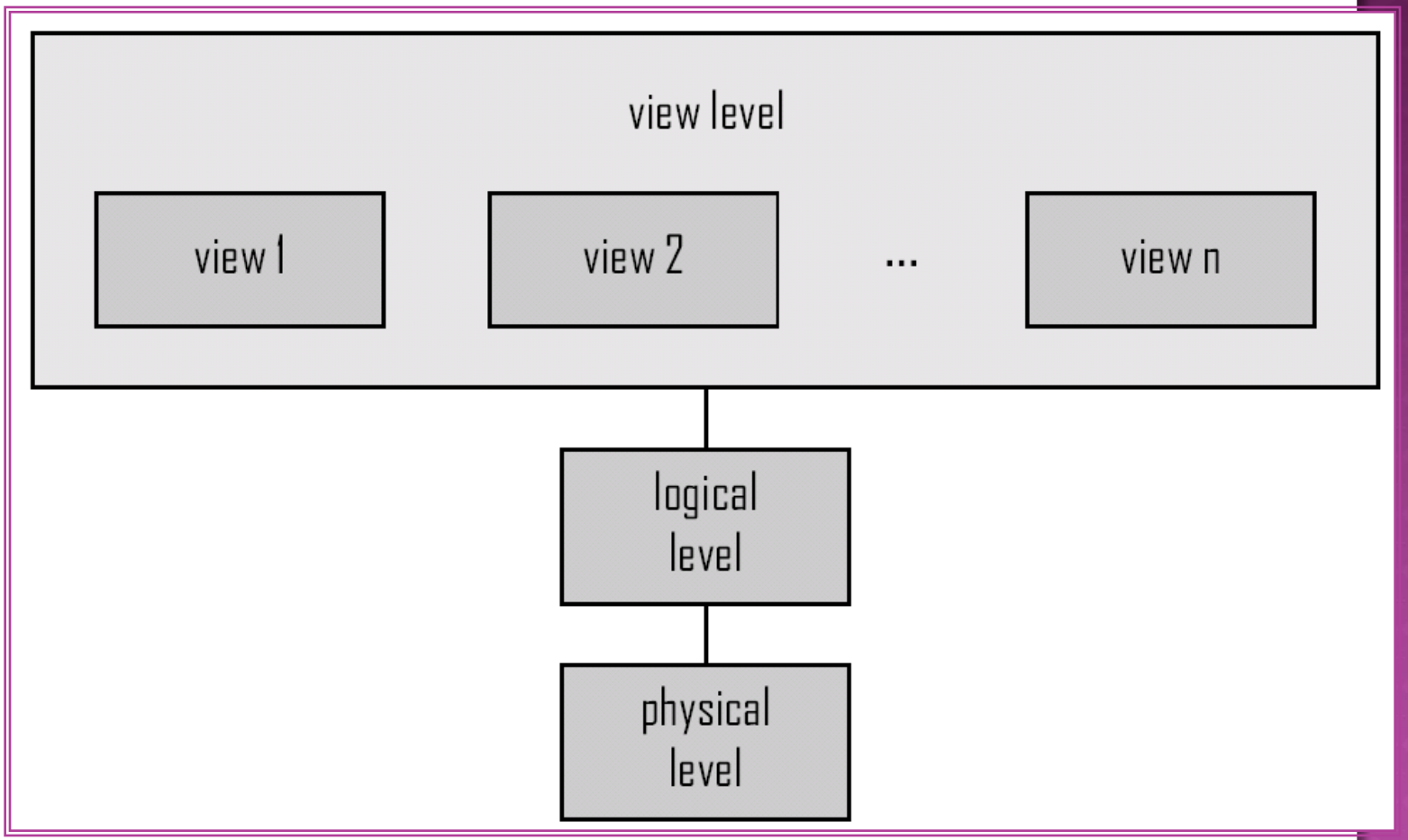
THE PHYSICAL MODEL (CONTINUED)

TABLE 2.3 Levels of Data Abstraction

MODEL	DEGREE OF ABSTRACTION	FOCUS	INDEPENDENT OF
External	High	End-user views	Hardware and software
Conceptual		Global view of data (independent of database model)	Hardware and software
Internal		Specific database model	Hardware
Physical	Low	Storage and access methods	Neither hardware nor software

VIEW OF DATA

An architecture for a database system



Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
    - The description of a database
    - The schema is specified during database design, and is not expected to change frequently
  - Example: The database consists of information about a set of customers and accounts and the relationship between them)
  - Analogous to type information of a variable in a program
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable

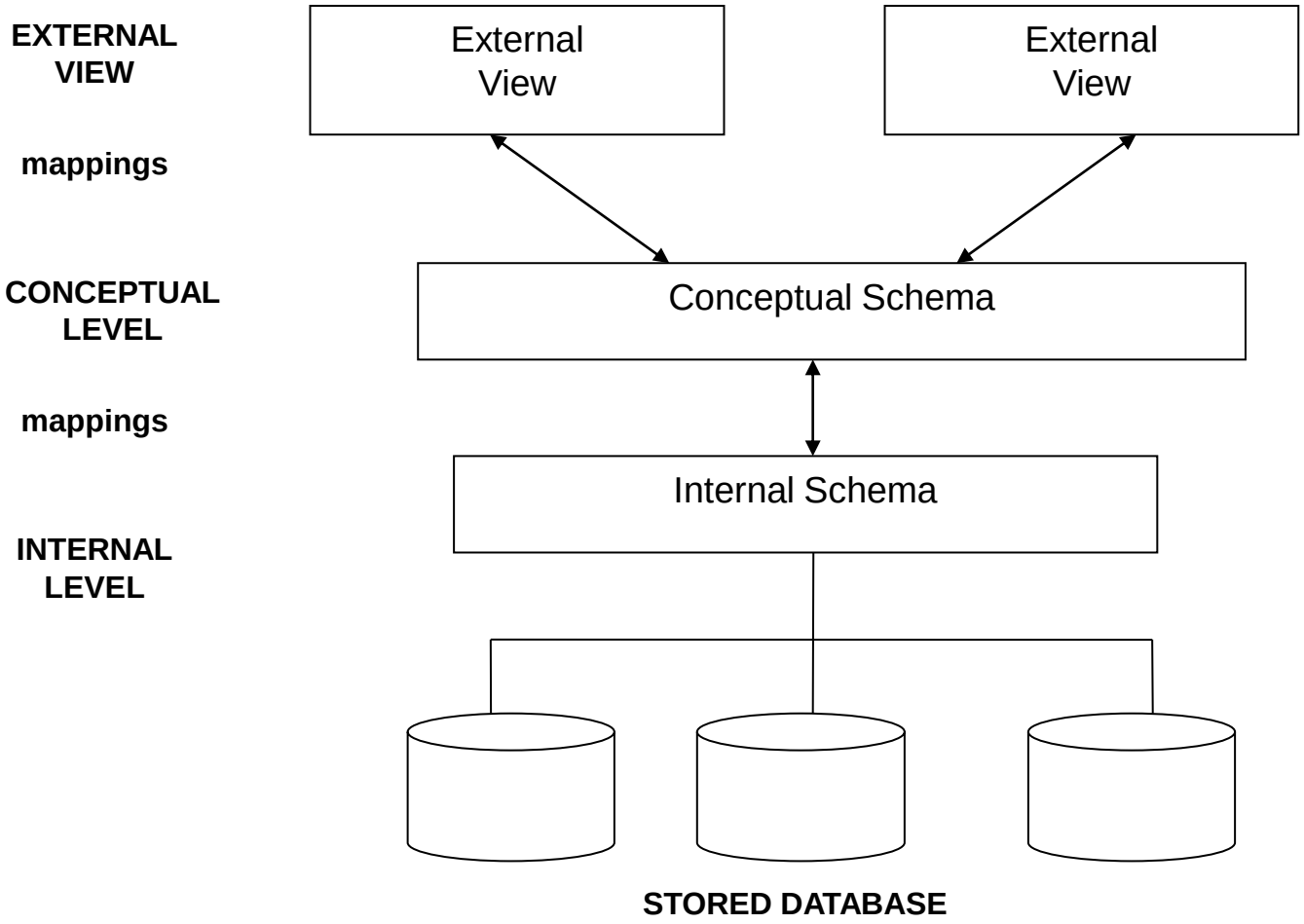
INSTANCES AND SCHEMAS(CONT.)

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.
- **Logical Data Independence**
  - the capacity to change the conceptual schema without having to change external schemas or application programs.
  - EXAMPLE : we may change the conceptual schema to expand the DB by removing a record type or data item.
    - The external schemas that refer only to the remaining data should not be affected.
    - Only the view definitions and the mapping need to be changed in a DBMS that supports logical data independence.

THREE SCHEMA ARCHITECTURE AND DATA INDEPENDENCE

- Remember from the previous chapters, three of the main characteristics of database systems, these are:
  1. Insulation of programs and data
  2. Support of multiple views
  3. Use of a catalogue to store the database description (schema)
- The three schema architecture helps to achieve these characteristics.

THREE SCHEMA ARCHITECTURE



THREE SCHEMA ARCHITECTURE(CONT.)

- The goal of the three schema architecture is to separate the user applications and the physical database. The schemas can be defined at the following levels:
  1. **The internal level** – has an internal schema which describes the physical storage structure of the database. Uses a physical data model and describes the complete details of data storage and access paths for the database.

THREE SCHEMA ARCHITECTURE(CONT.)

2. **The conceptual level** –
   - has a conceptual schema which describes the structure of the database for users.
   - It hides the details of the physical storage structures, and concentrates on describing entities, data types, relationships, user operations and constraints.
   - Usually a representational data model is used to describe the conceptual schema.
   - Independent of both software and hardware .
   - Provides a relatively easily understood macro level view of data environment

THREE SCHEMA ARCHITECTURE(CONT.)

3. **The External or View level** – includes external schemas or user vies. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. Represented using the representational data model.

- The three schema architecture is used to visualize the schema levels in a database. The three schemas are only descriptions of data, the data only actually exists is at the physical level.

THREE SCHEMA ARCHITECTURE(CONT.)

- Each user group refers only to its own external schema.
- The DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the database.
- The process of transforming requests and results between levels is called mapping.

DATABASE ARCHITECTURE-TERMS

- Each user communicates with the database via a language referred as data sub-language (DSL), which is the subset of the language concerned with the retrieval and storage of information in the database.
- Mapping defines the correspondence between data models and external/internal schema.
- DBMS is the software that handles all access to the database and also is responsible for applying the authorization checks and validation procedures.
- DBMS is a bridge between users and data.

BACK END VS. FRONT END

- A **back-end database** is a database that is accessed by users indirectly through an external application rather than by application programming stored within the database itself or by low level manipulation of the data (e.g. through SQL commands).
- A back-end database stores data but does not include end-user application elements such as stored queries, forms, macros or reports.

BACK END VS. FRONT END(CONT.)

- The term **back-end** database is most widely used among developers using small database programming systems which can contain the end-user application programming within the database as a single item.
- The most common of these is Microsoft Access

BACK END VS. FRONT END(CONT.)

- For more advanced database applications it is common to **split the data and the programming parts** in to a *front-end database* and a back-end database where the front-end holds all the application programming.
- The front end is an interface between the user and the back end.
- This has advantages in terms of scalability, performance and concurrency but requires greater effort on the part of the developer.

BACK END VS. FRONT END(CONT.)

- In the long term it may be easier to maintain and upgrade as new versions of the front-end can be deployed independently of the back-end database.
- The front and back-end databases do not always have to be of the same types. For example, it is possible to use a Microsoft Access front-end with a Microsoft SQL Server back-end.