


Database Management System

Lecture 1 Fundamental Database Concepts

CONTENTS

- **Basics of DBMS**
 - **Purpose of DBMS**
 - **Applications of DBMS**
 - **Views of Data**
 - **Instances and Schema**
 - **Data Models**
 - **Database Languages**
 - **Responsibility of Database Administrator**
- 

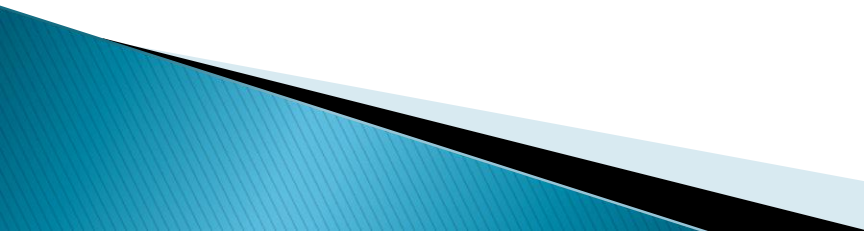
Database Management System (DBMS)

- ▶ 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

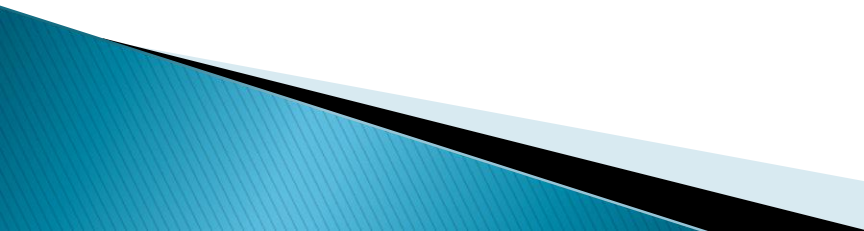
DBMS is a software (i.e. programs along with environment) which manages interrelated data about a particular enterprise.

Database Management System (DBMS)

- ▶ **Data** is facts/ information.
 - ▶ A **database** is any collection of data.
 - ▶ A **DBMS** is a software system designed to maintain a database.
 - ▶ A **Database Management System (DBMS)** is a software package designed to store and manage databases.

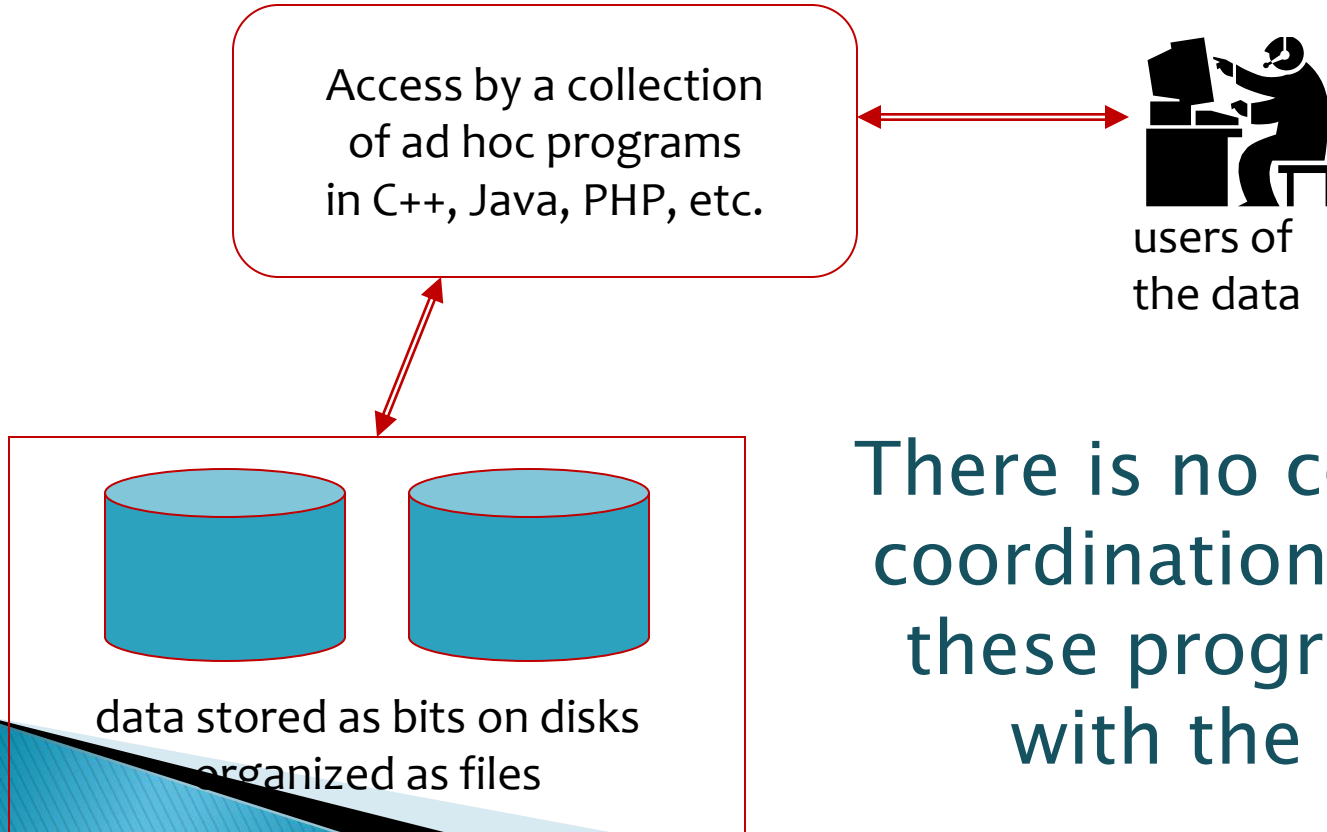
 - ▶ We use a DBMS when
 - there is a large amount of data
 - security and integrity of the data are important
 - many users access the data concurrently
- 

Basic Definitions

- ▶ **Database:** A collection of related data.
 - ▶ **Data:** Known facts that can be recorded and have an implicit meaning.
 - ▶ **Mini-world:** Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
 - ▶ **Database Management System (DBMS):** A software package/system to facilitate the creation and maintenance of a computerized database.
 - ▶ **Database System:** The DBMS software together with the data itself. Sometimes, the applications are also included.
- 

Why Use a DBMS?

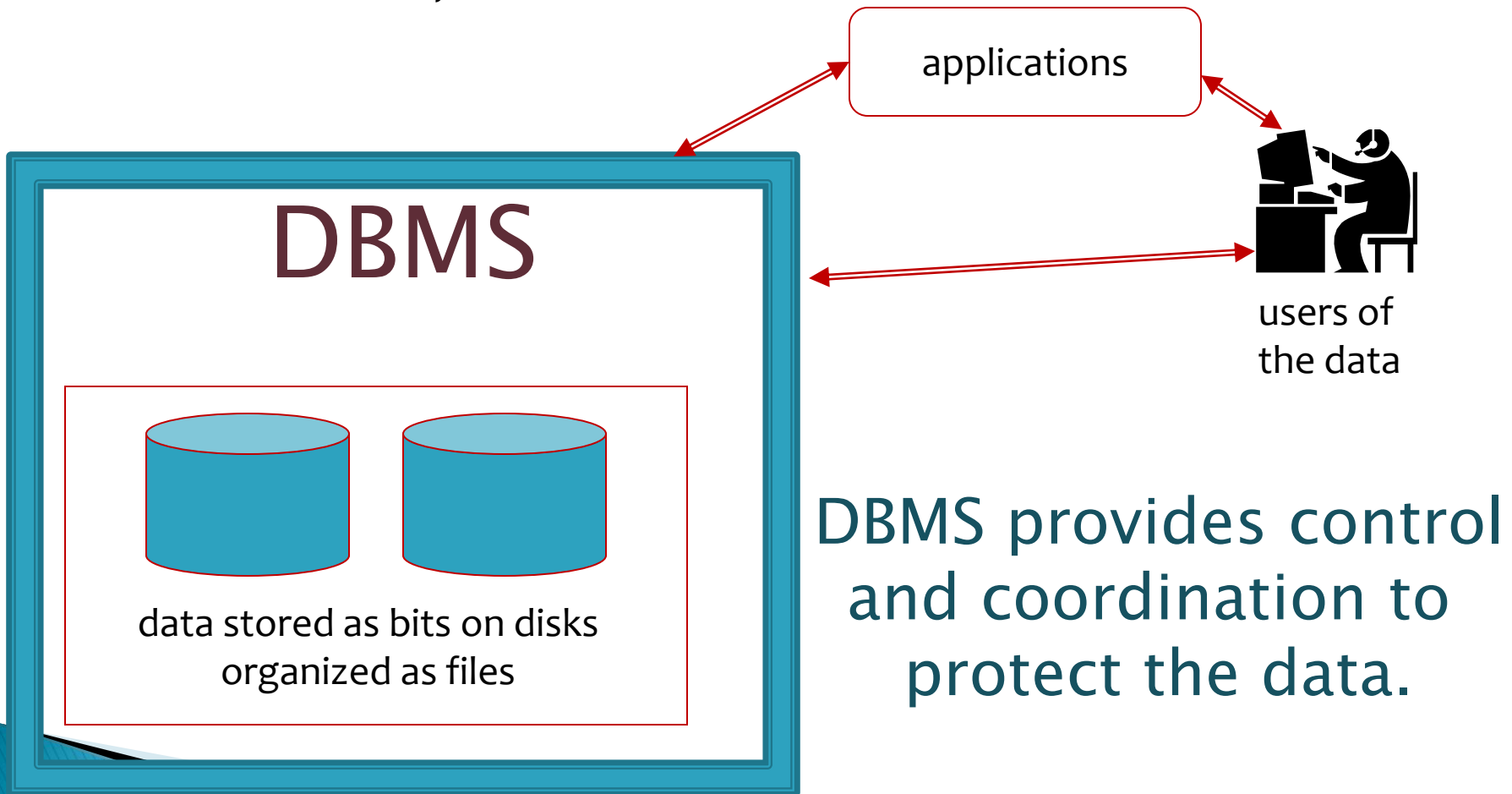
- ▶ Without a DBMS, we'd have:



There is no control or coordination of what these programs do with the data

Why Use a DBMS?

- ▶ With a DBMS, we have:



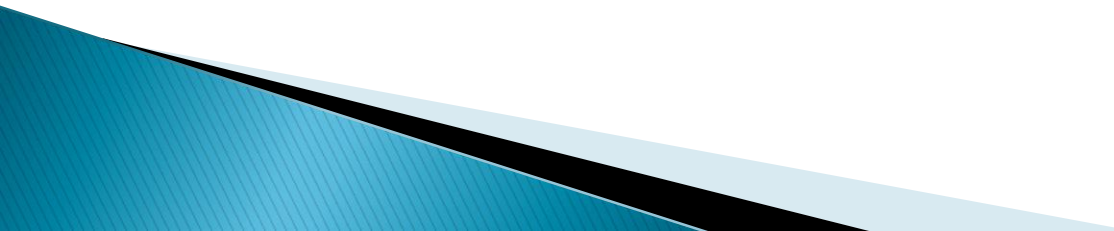
Purpose of Database Systems

- ▶ In the early days, database applications were built directly on top of file systems
- ▶ Drawbacks of using file systems to store data:
 - Data redundancy and inconsistency
 - Multiple file formats, duplication of information in different files
 - Difficulty in accessing data
 - Need to write a new program to carry out each new task
 - Data isolation — multiple files and formats
 - 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

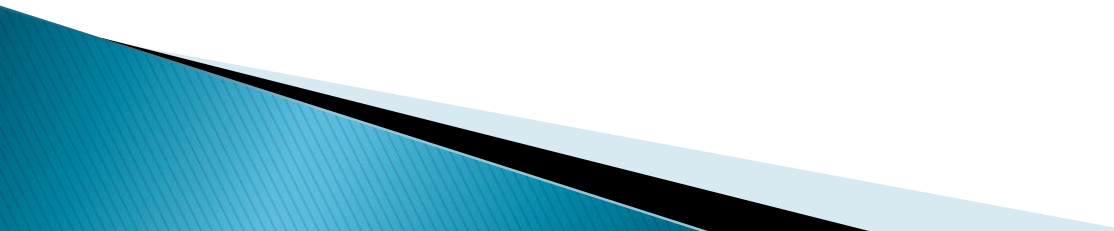
- ▶ Drawbacks of using file systems (cont.)
 - Atomicity of updates
 - Failures may leave database in an inconsistent state with partial updates carried out
 - Example: Transfer of funds from one account to another should either complete or not happen at all
 - Concurrent access by multiple users
 - Concurrent accessed needed for performance
 - Uncontrolled concurrent accesses can lead to inconsistencies
 - Example: Two people reading a balance and updating it 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

Typical DBMS Functionality

- ▶ **Define a database** : in terms of data types, structures and constraints
 - ▶ Construct or Load the Database on a secondary storage medium
 - ▶ **Manipulating the database** : querying, generating reports, insertions, deletions and modifications to its content
 - ▶ Concurrent Processing and Sharing by a set of users and programs – yet, keeping all data valid and consistent
- 

Typical DBMS Functionality

Other features:

- Protection or Security measures to prevent unauthorized access
 - “Active” processing to take internal actions on data
 - Presentation and Visualization of data
- 


Main Characteristics of the Database Approach

- ▶ Self-describing nature of a database system: A DBMS **catalog** stores the *description* of the database. The description is called **meta-data**). This allows the DBMS software to work with different databases.
- ▶ Insulation between programs and data: Called **program-data independence**. Allows changing data storage structures and operations without having to change the DBMS access programs.
- ▶ Data Abstraction: A **data model** is used to hide storage details and present the users with a *conceptual view* of the database.

Main Characteristics of the Database Approach

- ▶ 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.
- ▶ Sharing of data and multiuser transaction processing : allowing a set of concurrent users to retrieve and to update the database. Concurrency control within the DBMS guarantees that each **transaction** is correctly executed or completely aborted. OLTP (Online Transaction Processing) is a major part of database applications.

Applications of DBMS

- **Banking:** all transactions
 - **Airlines:** reservations, schedules
 - **Universities:** registration, grades
 - **Sales:** customers, products, purchases
 - **Online retailers:** order tracking, customized recommendations
 - **Manufacturing:** production, inventory, orders, supply chain
 - **Human resources:** employee records, salaries, tax deductions
- 

Levels of Abstraction

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

type *customer* = **record**

customer_id : string;

customer_name : string;

customer_street : string;

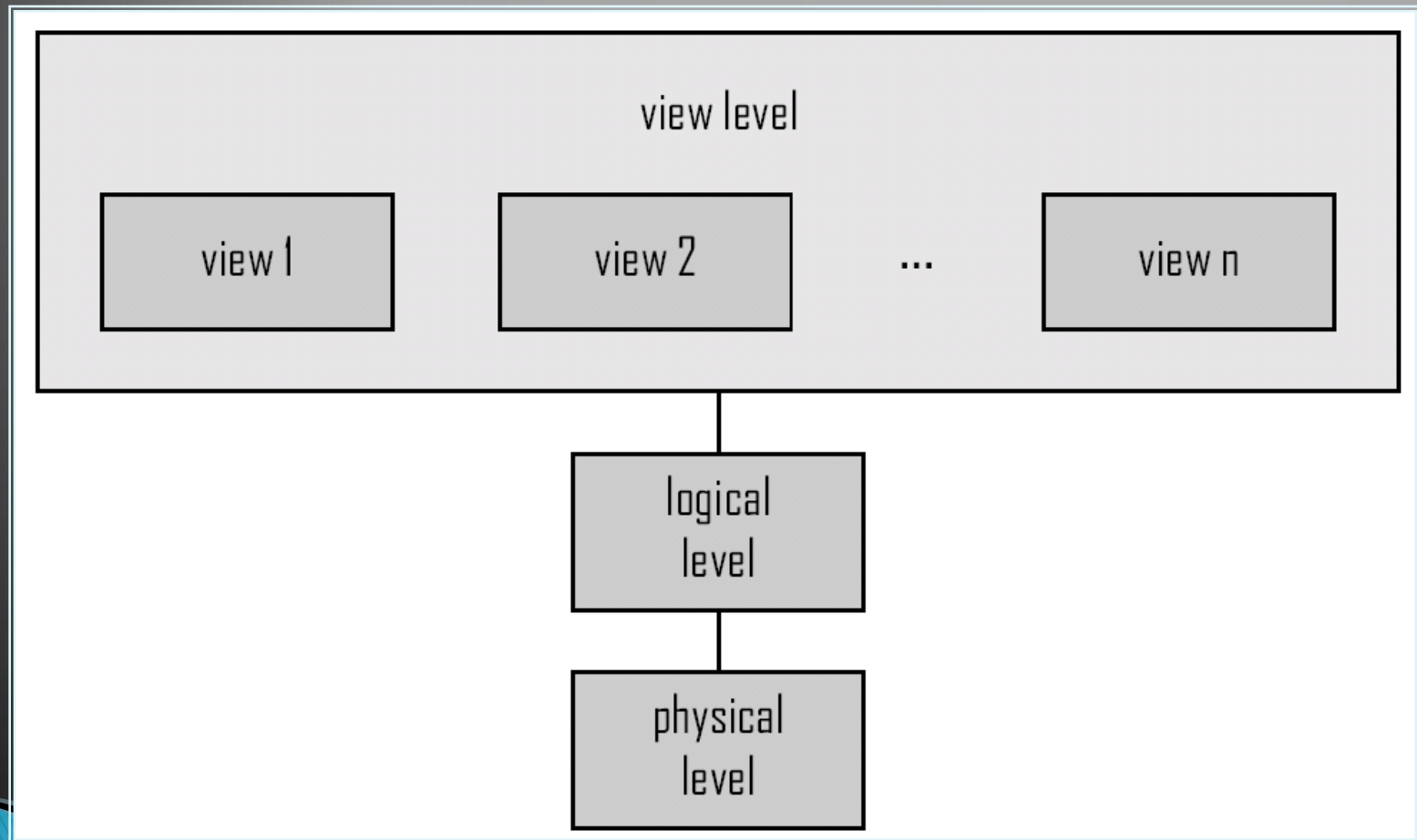
customer_city : string;

end;

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

View of Data

An architecture for a database system



Instances and Schemas

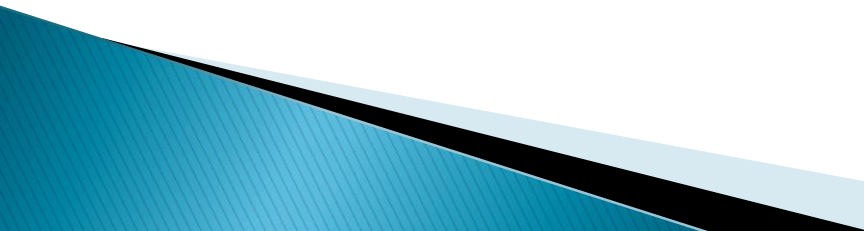
- ▶ Similar to types and variables in programming languages
- ▶ **Schema** – the logical structure of the database
 - Structural Description of the type of facts held in a database.
 - Example: The database consists of information about a set of customers and accounts and the relationship between them)
 - 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
- ▶ **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.

Data Models

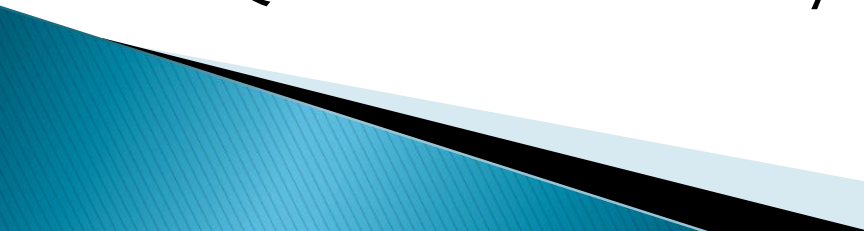
- ▶ A collection of tools for describing
 - Data
 - Data relationships
 - Data semantics
 - Data constraints

There are a number of different ways of organizing a schema, i.e. of modeling a database structure, these ways are known as **Data Models**.

Types of Data Models

- ▶ Relational model
 - ▶ Entity–Relationship data model (mainly for database design)
 - ▶ Object–based data models (Object–oriented and Object–relational)
 - ▶ Semistructured data model (XML)
 - ▶ Other older models:
 - Network model
 - Hierarchical model
- 

Data Manipulation Language (DML)

- ▶ Language for accessing and manipulating the data organized by the appropriate data model
 - DML also known as query language
 - ▶ Two classes of languages
 - **Procedural** – user specifies what data is required and how to get those data (PL/SQL)
 - **Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data (SQL)
 - ▶ SQL is the most widely used query language
- 

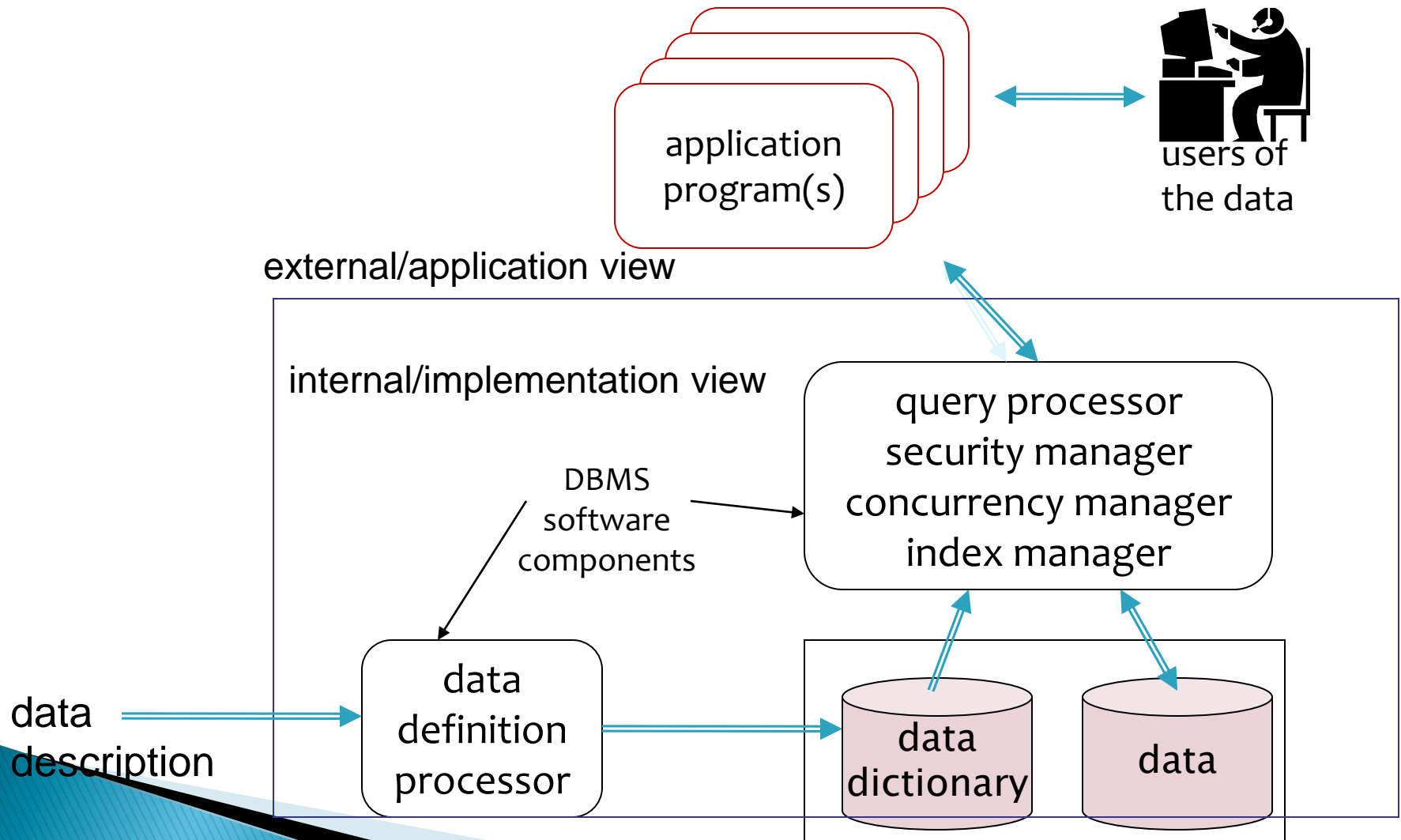
Data Definition Language (DDL)

- ▶ Specification notation for defining the database schema

Example: **create table** *account* (
 account_number **char**(10),
 branch_name **char**(10),
 balance **integer**)

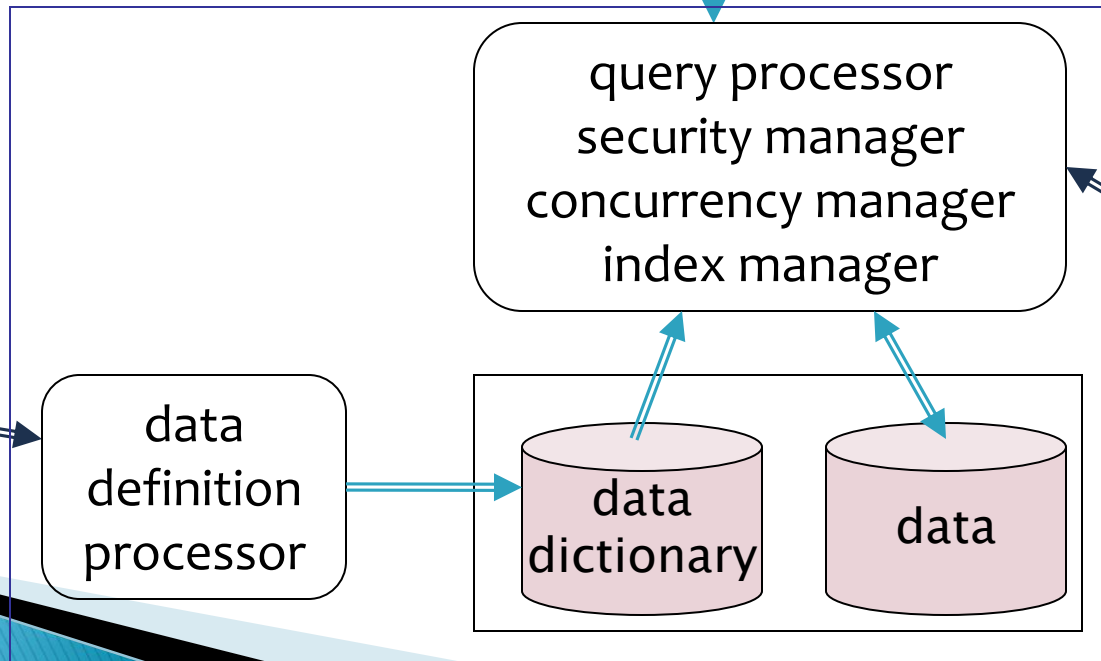
- ▶ DDL compiler generates a set of tables stored in a *data dictionary*
- ▶ Data dictionary contains metadata (i.e., data about data)
 - Database schema
 - Data *storage and definition* language
 - Specifies the storage structure and access methods used
 - Integrity constraints
 - Domain constraints
 - Referential integrity (e.g. *branch_name* must correspond to a valid branch in the *branch* table)
 - Authorization

DBMS Structure



DBMS Languages

DML: data manipulation language
QL: query language
GPL: general purpose languages



system configuration languages

DDL:
data definition language

Database Users

Users are differentiated by the way they expect to interact with the system

- ▶ **Application programmers** – interact with system through DML calls
- ▶ **Sophisticated users** – form requests in a database query language
- ▶ **Specialized users** – write specialized database applications that do not fit into the traditional data processing framework
- ▶ **Naïve users** – invoke one of the permanent application programs that have been written previously
 - Examples, people accessing database over the web, bank tellers, clerical staff

Database Administrator

- ▶ Coordinates all the activities of the database system
 - has a good understanding of the enterprise's information resources and needs.
- ▶ Database administrator's duties include:
 - Storage structure and access method definition
 - Schema and physical organization modification
 - Granting users authority to access the database
 - Backing up data
 - Monitoring performance and responding to changes
 - Database tuning

Database Architecture

The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:

- ▶ Centralized
 - ▶ Client-server
 - ▶ Parallel (multiple processors and disks)
 - ▶ Distributed
- 