



**RV College of
Engineering**

Go, change the world

Unit 1 (Introduction to Database Systems)

Original Content:

Ramez Elmasri and Shamkant B. Navathe

Dr. Poonam Ghuli

*Associate Professor, Department of CSE
RV College of Engineering, Bengaluru - 59*

Contents

- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Database Users
- Data Models
- Schemas and Instances
- Three-schema Architecture and Data Independence
- Database Languages and Interfaces
- The Database System Environment.

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.

Problems with traditional approach

- Inconsistency
- Redundancy
- Loss of flexibility
- Multiple files
- Update Problem

Types of Databases and Database Applications

- Traditional Applications:
 - Numeric and Textual Databases
- More Recent Applications:
 - Multimedia Databases
 - Geographic Information Systems (GIS)
 - Data Warehouses
 - Real-time and Active Databases
 - Many other applications

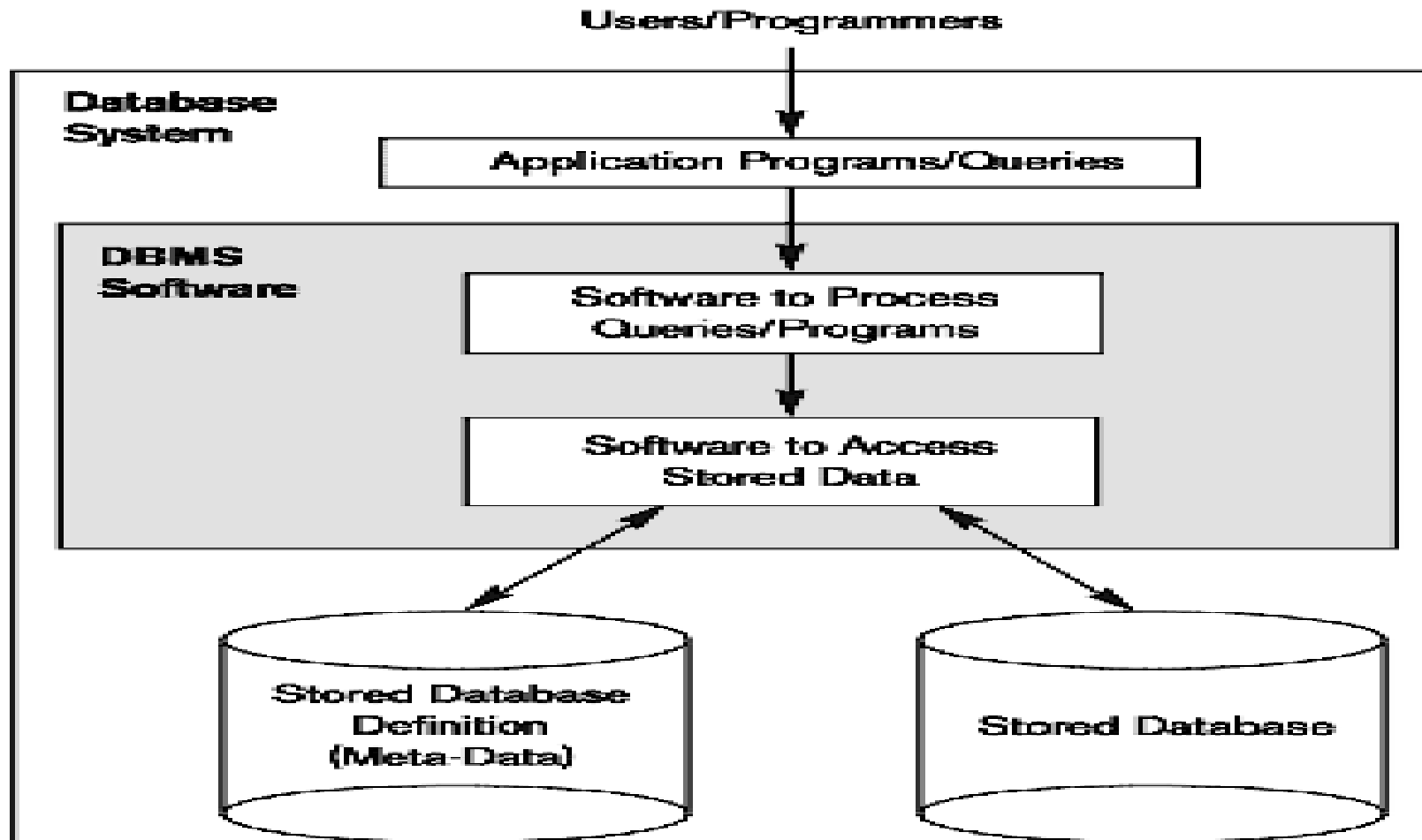
Typical DBMS Functionality

- **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

Typical DBMS Functionality

- Other features:
 - **Protection or Security** measures to prevent unauthorized access
 - **Presentation and Visualization of data**
 - Maintaining the database and associated programs over the lifetime of the database application
 - Called database, software, and system maintenance

Simplified database system environment



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 belong to SECTIONs
 - COURSEs *have prerequisite* COURSEs
 - INSTRUCTORs *teach* SECTIONs
 - COURSEs *are offered by* DEPARTMENTs
 - STUDENTs *major in* DEPARTMENTs

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 such as University DB, Banking DB or Company DB.
- **Insulation between programs and data:**
 - Called **program-data independence**.
 - Allows changing data structures and storage organization without having to change the DBMS access programs.

Main Characteristics of the Database Approach

- **Data Abstraction:**

- The characteristic that allows program-data independence and program-operation independence
 - Program-operation independence
 - User application programs can operate on the data by invoking operations through their names and arguments, regardless of how the operations are implemented
- A **data model** is used to hide storage details and **present** the users with **a conceptual view** of the database.
- **Relational model** hides how the data is stored and how the operations are implemented. DB is represented in terms of entities, attributes and relationships among entities that is understood by most users.
- **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

- A DBMS must enforce following **transaction properties**
 - **Isolation**
 - Each transaction appears to execute in isolation from other transactions.
 - **Atomicity**
 - Either all the database operations in a transaction are executed or none are.
- **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. This 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**
 - **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.
 - **System Analysts and Application Programmers (Software Engineers)**
 - Familiar with the full range of capabilities provided by the DBMS to accomplish their tasks

Categories of End-users

- **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.
 - **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

Workers behind the Scene

- **DBMS System Designers and Implementers**

- Design and implement the DBMS modules and interfaces as a software package

- **Tool Developers**

- Design and implement tools – the software packages that facilitate database modeling and design, database system design, and improved performance

- **Operators and Maintenance Personnel**

- Responsible for the actual running and maintenance of the hardware and software environment for the database system

Data Models

- **Data Abstraction**

- Suppression of details of data organization and storage and the highlighting of the essential features for an improved understanding of data

- **Data Model:**

- A collection of concepts to describe the **structure** of a database, the **operations** for manipulating these structures, and certain **constraints** that the database should obey.
- It provides necessary means to achieve data abstraction

- **Data Model Structure and Constraints:**

- Constructs are used to define the database structure
- Constructs typically include **elements** (and their **data types**) as well as groups of elements (e.g. **entity, record, table**), and **relationships** among such groups
- Constraints specify some restrictions on valid data; these constraints must be enforced at all times

Data Models

- **Data Model Operations:**

- These operations are used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.
- Operations on the data model may include ***basic model operations*** (e.g. generic insert, delete, update) and ***user-defined operations*** (e.g. compute_student_gpa, update_inventory)

Categories of Data Models

- **Conceptual (high-level, semantic) data models:**
 - Provide concepts that are close to the way many users perceive data.
 - (Also called *entity-based* or *object-based* data models.)
- **Physical (low-level, internal) data models:**
 - Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals
- **Implementation (representational) data models:**
 - Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

Database Schema

- 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.

Example of a Database Schema

STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

Figure 2.1

Schema diagram for the
database in Figure 1.2.

Database State

- 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*
 - Refers to the ***content*** of a database at a moment in time.
- Initial Database State
 - Refers to the database state when it is initially loaded into the system.
- Valid State
 - A state that satisfies the structure and constraints of the database.

DB Schema v/s DB 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**.

Three-Schema Architecture

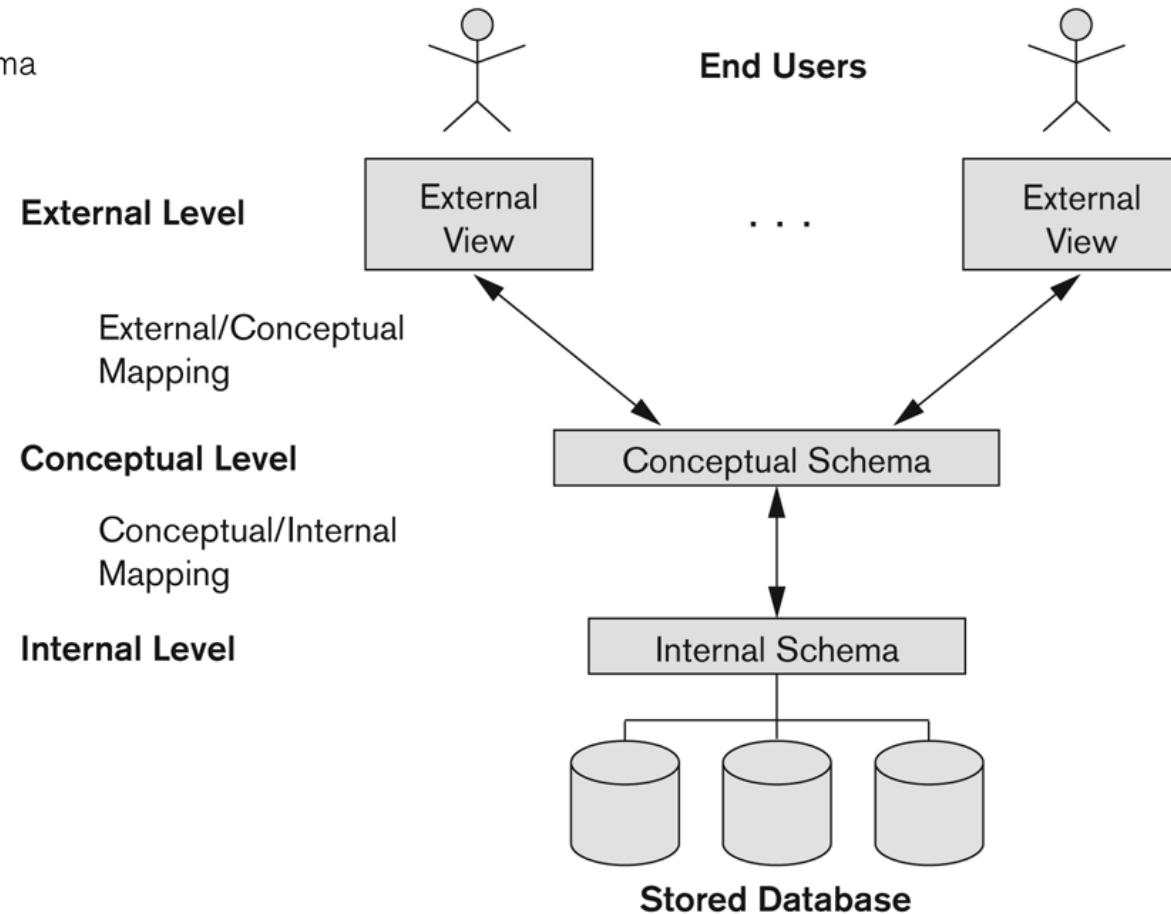
- Proposed to support DBMS characteristics of:
 - **Program-data independence.**
 - Support of **multiple views** of the data.
- Its goal is to separate the user applications and the physical database.
- Not explicitly used in commercial DBMS products, but has been useful in explaining database system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

The Three-Schema Architecture

- Defines DBMS schemas at **three** levels:
 - **Internal schema** at the internal level to describe physical storage structures and access paths (e.g indexes).
 - Typically uses a **physical** data model.
 - **Conceptual schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users. It describes entities, data types, relationships, user operations and constraints.
 - Uses a **conceptual** or an **implementation** data model.
 - **External schemas** at the external level (or view level) to describe the various user views.
 - Usually uses the same data model as the conceptual schema.

The Three-Schema Architecture

Figure 2.2
The three-schema
architecture.



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

Data Independence

- When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence.
- The higher-level schemas themselves are **unchanged**.
 - Hence, the application programs need not be changed since they refer to the external schemas.

DBMS Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
 - High-Level or Non-procedural Languages: These include the relational language SQL
 - May be used in a standalone way or may be embedded in a programming language
 - Low Level or Procedural Languages:
 - These must be embedded in a programming language

DBMS Languages

- **Data Definition Language (DDL):**

- Used by the DBA and database designers to specify the conceptual schema of a database.
- In many DBMSs, the DDL is also used to define internal and external schemas (views).
- In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define internal and external schemas.
 - SDL is typically realized via DBMS commands provided to the DBA and database designers

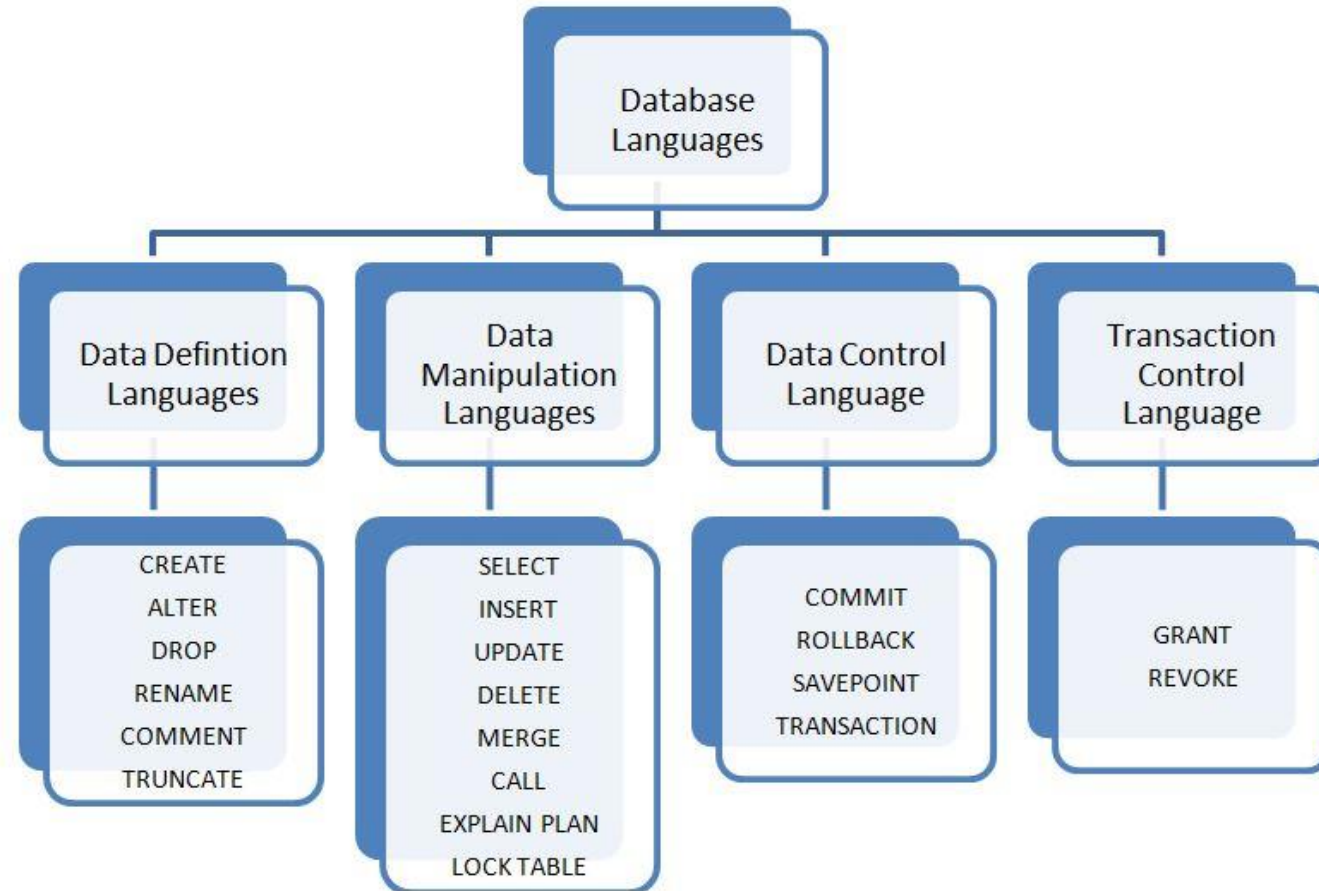
DBMS Languages

- **Data Manipulation Language (DML):**
 - Used to specify database retrievals and updates
 - DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.
 - A library of functions can also be provided to access the DBMS from a programming language
 - Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

Types of DML

- **High Level or Non-procedural Language:**
 - For example, the SQL relational language
 - Are “set”-oriented and specify what data to retrieve rather than how to retrieve it. Many records can be retrieved in one DML statement
 - Also called **declarative** languages.
- **Low Level or Procedural Language:**
 - Retrieve data one record-at-a-time;
 - Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

DBMS Languages



Typical DBMS Component Modules

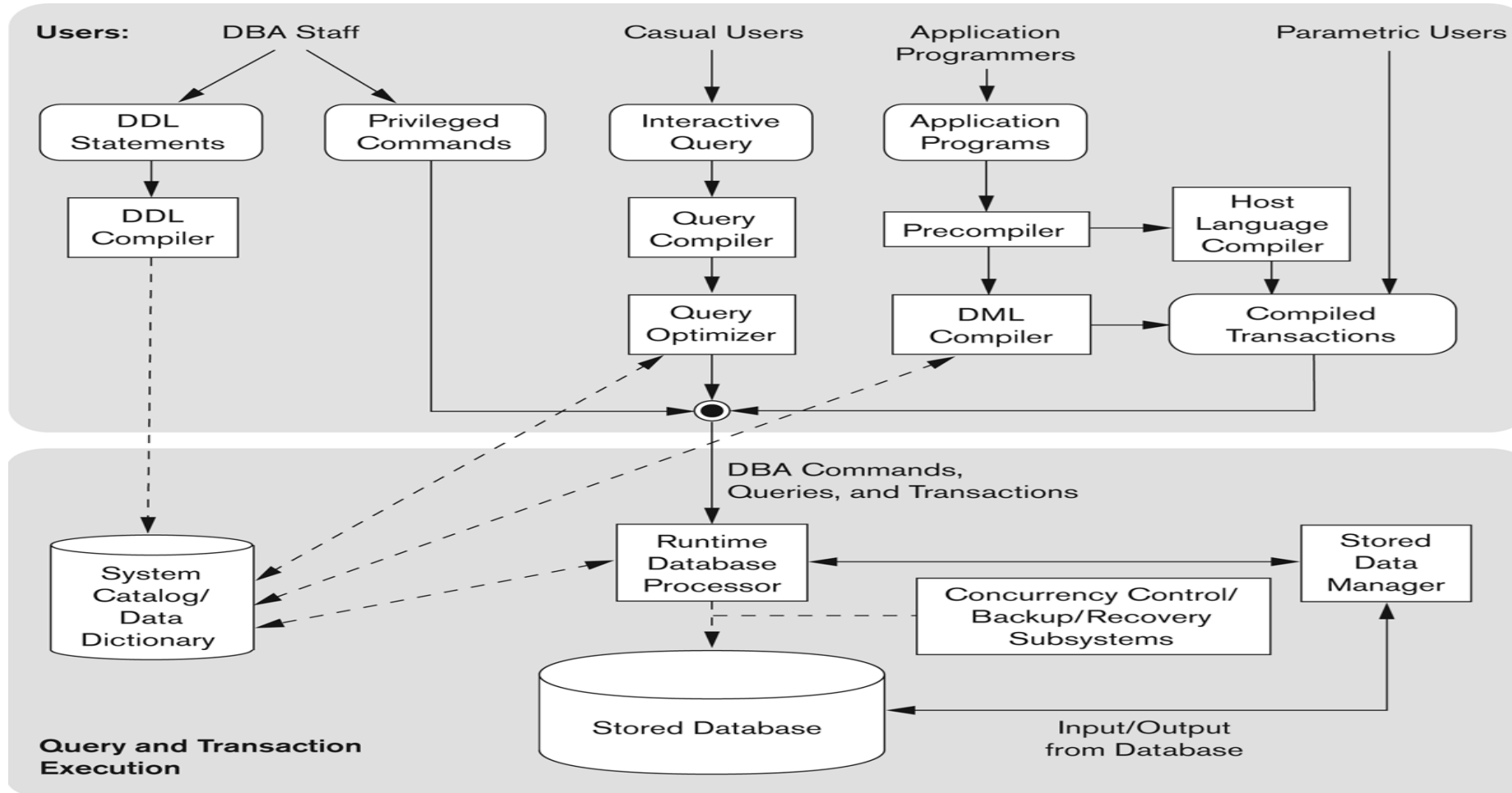


Figure 2.3
Component modules of a DBMS and their interactions.

Thank YOU