Unit I
Chapter 2 :
Database System Concepts and Architecture

Outline

2.1 Data Models, Schemas, and Instances

2.2 Three-Schema Architecture and Data Independence

2.3 Database Languages and Interfaces

2.4 The Database System Environment

2.1 Data Models, Schemas, and Instances

- **Data Abstraction**
- Fundamental characteristics of the DB is to provide some levels of **data abstraction**.

- It refers to the suppression of details of data organization and storage and highlighting the essential features for an improved understanding.

- Different users may perceive data at their preferred level of details.

2.1 Data Models, Schemas, and Instances

- **Data Model:**
  - Collection of concepts that can be used to describe the ***structure*** of a database.
  - Structure of the database typically include data types, relationships and constraints even basic operations.
  - Provides necessary means to achieve this abstraction.

2.1 Data Models, Schemas, and Instances

- Types of concepts they use to describe the database structure

- **Categories of Data Models**
- **A. 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.)

- **B. Physical (low-level, internal) data models:**
  - Provide concepts that describe details of how data is stored in the computer. Eg. Record format, record ordering, access path, index etc.,

2.1 Data Models, Schemas, and Instances

- **Categories of Data Models**
- **C. 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). Concepts are easily understood by end users. Hide many details of data storage but can be implemented directly.
- **D. Self-Describing Data Models**
  - **Data storage combines the description of the data with the data values themselves.**
  - **In traditional DBMS both are separated.**
  - **Ex:** XML, NoSQL Systems (Big Data Applications using Mongo DB)

2.1 Data Models, Schemas, and Instances

- **A. Conceptual data models**
- **What system contains?**
- Provides the concepts of ***entities, relationships and attributes***
  - ***Entity*** is an object which has logical or physical existence
    - Entities: *student, faculty member, course, departments etc.,*
  - ***Attributes*** represents some property of interest that further describes an entity
    - ***Attributes: name, rollNumber, address etc., of student entity, name, empNumber, phoneNumber etc., of faculty entity etc.,***

2.1 Data Models, Schemas, and Instances

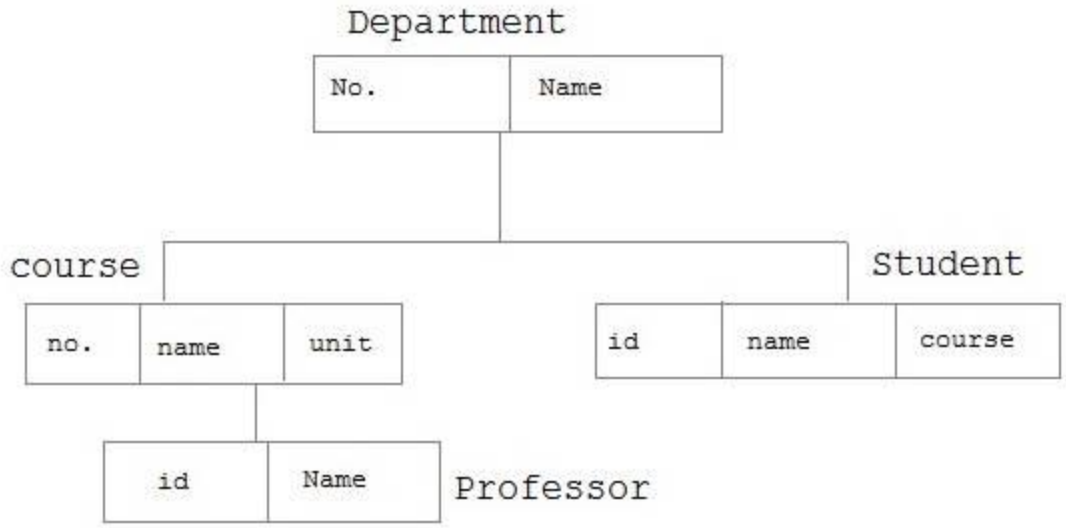
- **A. Conceptual data models**
- Provides the concepts of ***entities, relationships and attributes***
  - *A* ***relationship*** among 2 or more entities represents association among two or more entities
    - Relationships: enrolment relationship between student & course
    - Employment relationship between faculty & department
- ERD and UML (Unified Modelling Language)

2.1 Data Models, Schemas, and Instances

- **B. Representational data models**
- **How it is implemented regardless of the DBMS**
  - **Note: Physical - implemented w.r.t DBMS**
- Represent data by using **record structures**
- **Frequently used** in traditional commercial DBMS
  - i. **Hierarchical model**
  - ii. **Network model**
  - iii. **Relational model (RDBMS)**

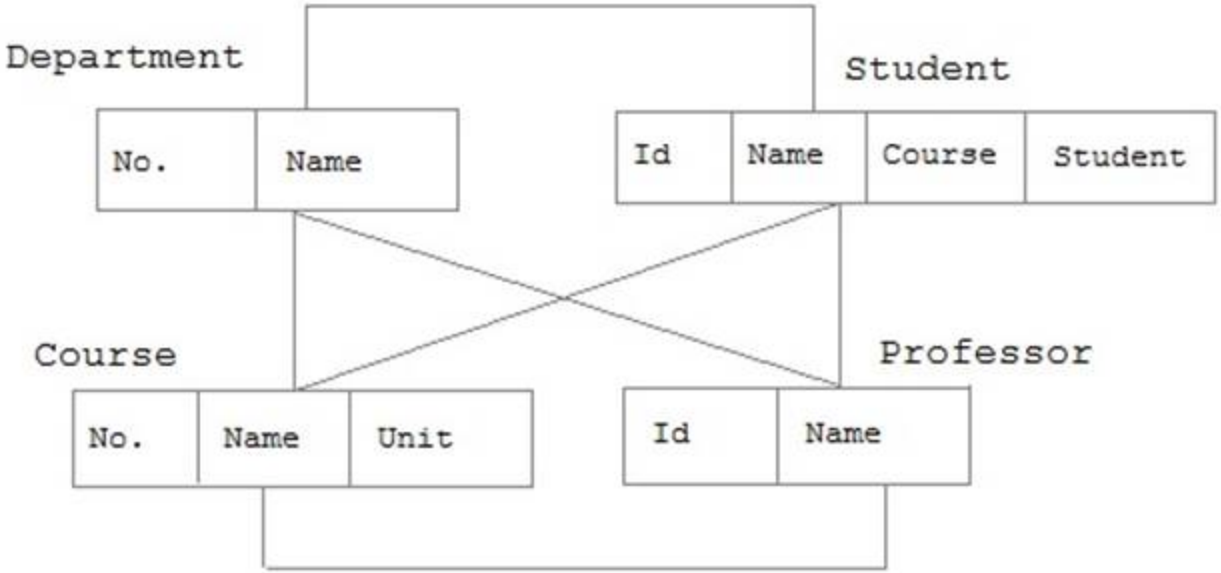
2.1 Data Models, Schemas, and Instances

- **B. Representational data models**
  - **i. Hierarchical model**
    - Data is organized in a **tree like structure**, each **entity** has only one parent but can have several children. At the top hierarchy there is only one entity which is root



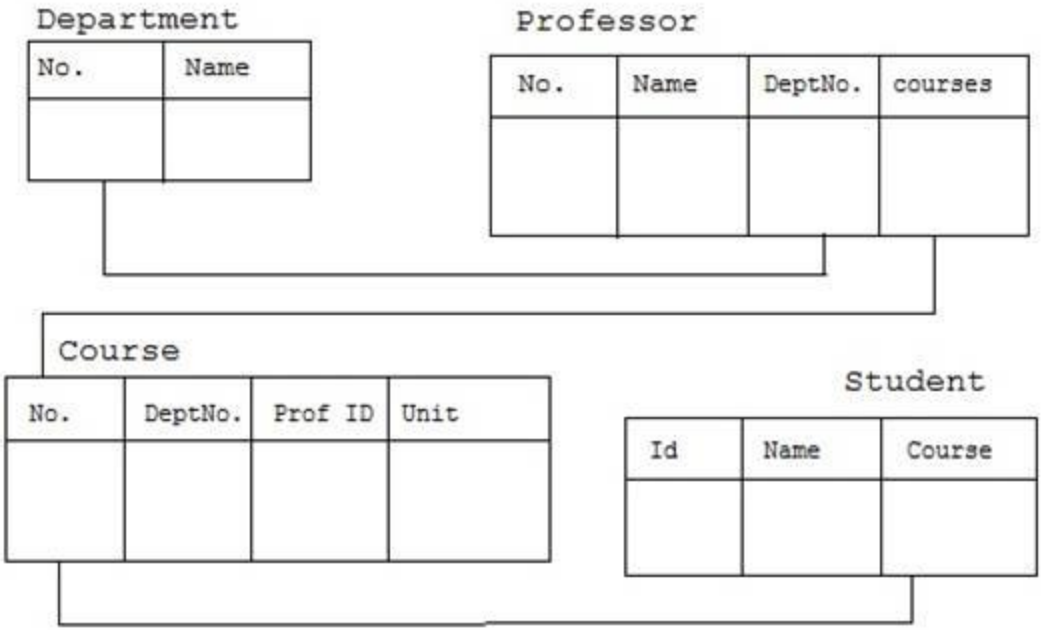
2.1 Data Models, Schemas, and Instances

- **B. Representational data models**
  - **Ii) Network model**
    - Entities are **organized like a graph,** in which some **entities** can be accessed through several path



- **B. Representational data models**
  - **Iii) Relational model**

– Data is organized in two dimensional tables called relation. The **tables or relation are related** to each other



2.1 Data Models, Schemas, and Instances

- Database Schema or Intension
  - **Next level after data model**
  - The ***description*** of a database fields or attributes.
  - Specified during database design, **not expected to change** frequently
  - Most data models have certain **conventions for displaying schema**
- Schema Diagram:
  - An ***illustrative*** display of a database schema.
- Schema Construct:
  - A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	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

2.1 Data Models, Schemas, and Instances

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.

2.1 Data Models, Schemas, and Instances

- **Database State or Extension:**
  - 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*

2.1 Data Models, Schemas, and Instances

- Database State or Extension:
  - **Empty state:** When **new database** is defined we specify its schema only at this point corresponding state is ***empty state***.
  - **Initial Database State**: Refers to the database state when it is **initially loaded**.
  - Every time an update operation is applied to database, we get **another database state.**
  - At any point in time the database has **current state**
  - **Valid State:** A state that satisfies the structure and constraints of the database.

2.1 Data Models, Schemas, and Instances

- DBMS stores **schema constructs and constraints together as meta-data** in DBMS catalog.
- **Schema evolution**: adding new field to the database schema
- Modern **DBMSs support schema evolution when the database is operational**.

2.2 Three-Schema Architecture and Data Independence

- **Three-Schema Architecture**
- **Helps to visualize three characteristics of database**
- Separates user applications from the physical database
- Describes how data in the database is viewed by the users.
- Schemas can be defined at three different levels :
  - **Internal level** : Describes the physical storage structure
  - **Conceptual level** : Describes entire structure of DB (logical)
  - **External level**:Describe part of DB according to users requirements

2.2 Three-Schema Architecture and Data Independence

- Objectives of Three-Schema Architecture
  - All users should be able to access same data.
  - separate each user's view of the database from the way it is physically represented.
  - Users should not need to know physical database storage details.

2.2 Three-Schema Architecture and Data Independence

- **Internal level also known as Internal Schema**
  - The way the DBMS and OS perceive the data.
  - Physical representation of the DB on the computer.
  - Uses physical data model.
  - Physical implementation of the DB to achieve optimal run–time performance and storage space utilization.
    - Storage space allocation for data and indexes
    - Record description for storage
    - Record placement
    - Data compression, encryption

2.2 Three-Schema Architecture and Data Independence

- **Conceptual level has a conceptual schema**
  - Structure of the entire database for a community of users.
  - What data is stored in the database.
  - Usually, representational data model is used to describe the conceptual schema when a database is implemented.
  - Represents : entities, data types, relationships, user operations and constraints.

2.2 Three-Schema Architecture and Data Independence

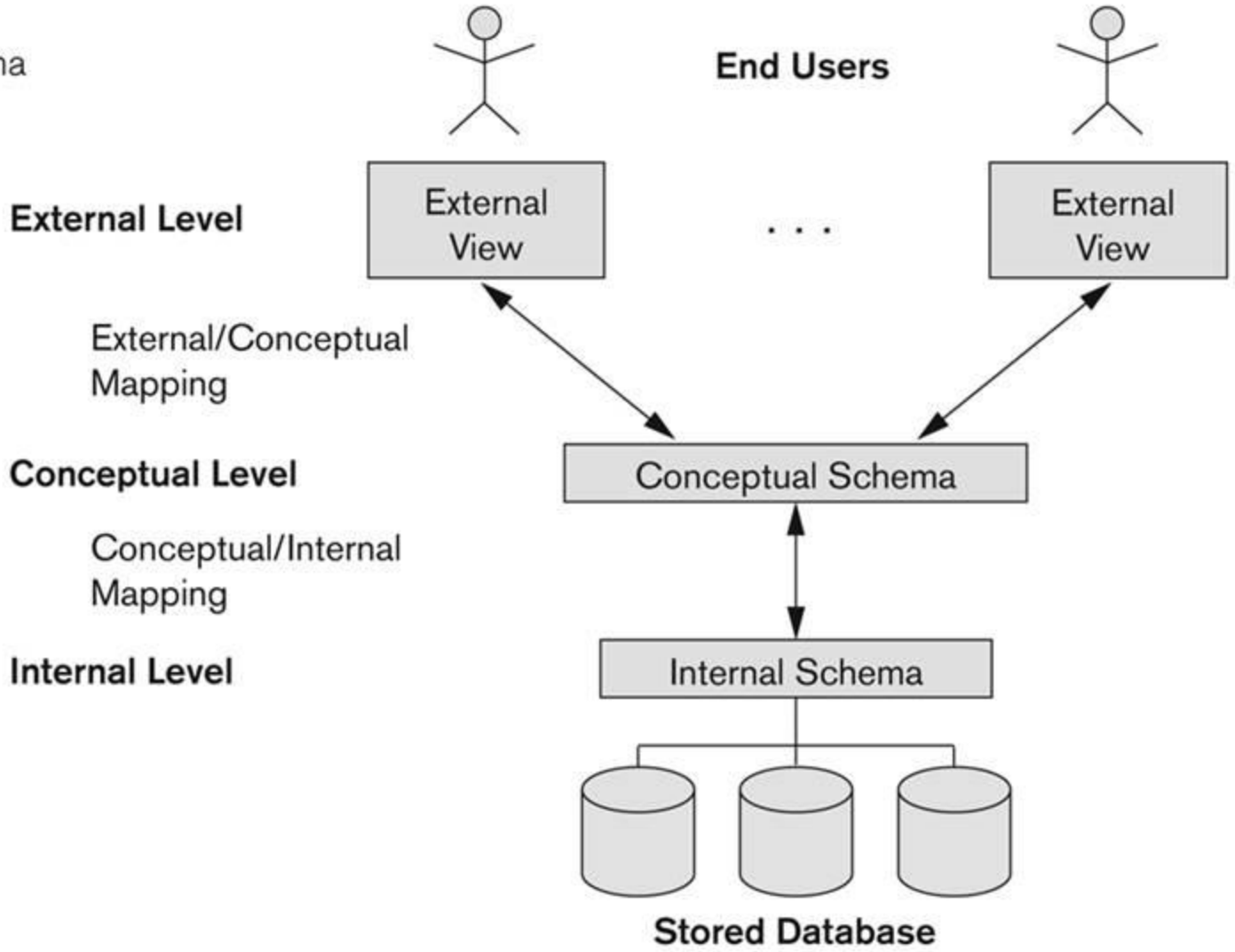
- External or view level includes external schemas or user views

  - Consists of a number of different external views of the DB.
  - Describes part of the DB for particular group of users.
  - Each external schema is implemented using a representational data model.

  - **Exceptions to 3 schema:** Oracle or SQL Server use SQL to realize external and conceptual schemas together.

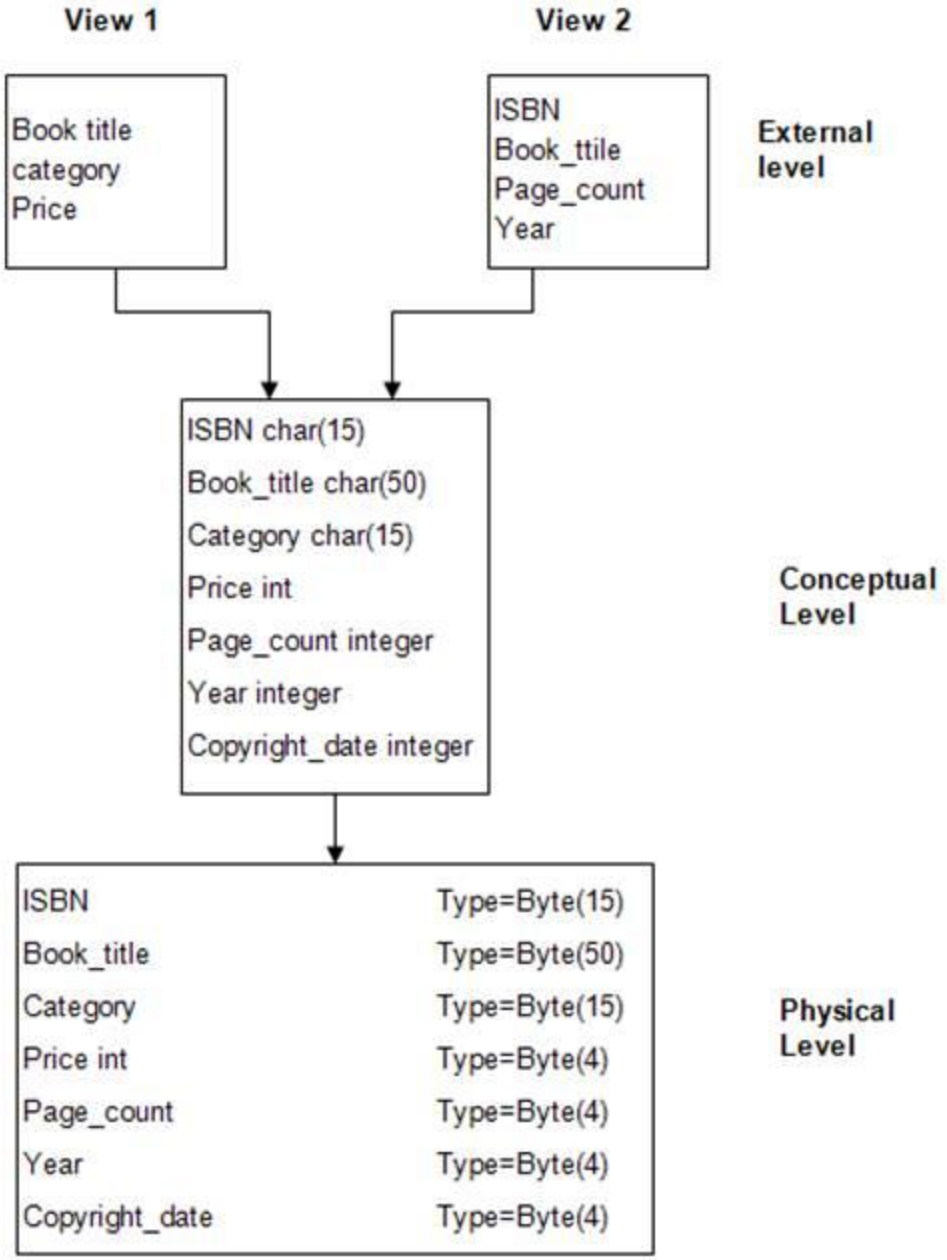
2.2 Three-Schema Architecture and Data Independence

Figure 2.2
The three-schema architecture.

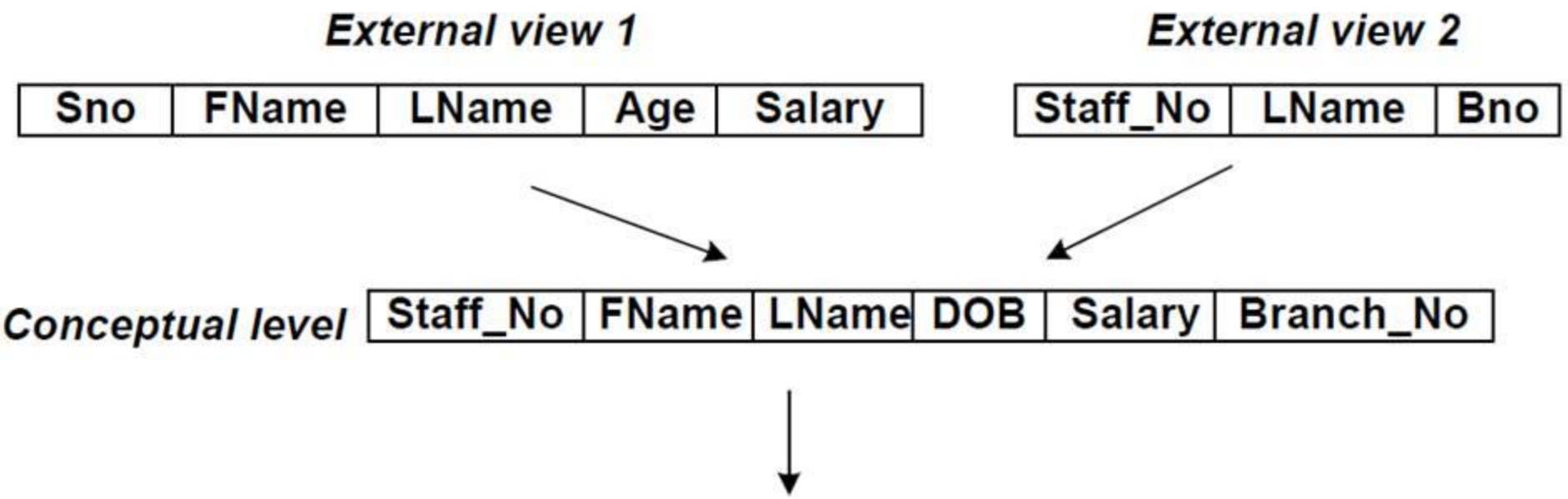


2.2 Three-Schema Architecture and Data Independence

Example 1 of three schema architecture



2.2 Three-Schema Architecture and Data Independence



Internal level

```
struct STAFF {
        int Staff_No;
        int Branch_No;
        char FName[15];
        char LName[15];
        struct date Date_of_Birth;
        float Salary;
        struct STAFF*next;
};
index Staff_No; index Branch_No;
```

Example 2 of three schema architecture

2.2 Three-Schema Architecture and Data Independence

- **Mapping**
- In three schema architecture, each user group refers only to its own external view.
- User specifies a request to generate a new external view;
  - DBMS must transform the request specified at external level into a request at conceptual level, and then into a request at physical level.
  - If requests for data retrieval, data extracted from the database must be presented according to user needs.
- Processes of transforming requests and result between various levels of DBMS architecture: *mapping (formatting).*

2.2 Three-Schema Architecture and Data Independence

- **Data Independence**
- Data independence is defined as the capacity to change the schema at one level of database system without having to change the schema at next higher level.

- Two types
  - Logical data independence
  - Physical data independence

2.2 Three-Schema Architecture and Data Independence

- Data Independence
- **Logical Data Independence:**
  - Capacity to change the conceptual schema without having to change the external schemas and their associated application programs.
  - We may change conceptual schema to expand the database, to change constraints, or to reduce the database.
  - External schemas that refer only to the remaining data should not be affected.

2.2 Three-Schema Architecture and Data Independence

- Data Independence
- **Logical Data Independence:**

Examples

The addition or removal of new entities, attributes, or relationships to the conceptual schema should be possible without having to change existing external schemas or having to rewrite existing application programs.

2.2 Three-Schema Architecture and Data Independence

- Data Independence
- **Physical Data Independence:**
  - The capacity to change the internal schema without having to change the conceptual schema.
  - Changes to the internal schema may be needed due to physical file reorganization
  - For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance or new access path to improve retrieval speed

2.2 Three-Schema Architecture and Data Independence

- Data Independence
- **Physical Data Independence:**

What do you mean by Physical Data Independence

- The ability to change the physical schema without changing the logical schema is called as Physical Data Independence. Changes in the physical schema may include.
- Using new storage devices.
- Using different data structures.
- Switching from one access method to another.
- Using different file organizations or storage structures.
- Modifying indexes.

2.2 Three-Schema Architecture and Data Independence

- **Data Independence**

- Generally, physical data independence exists in most databases in which exact location of data on disk, details of storage, splitting, merging of records and so on are hidden from the user. Application remains unaware of details.

- On the other hand logical data independence is very hard to achieve because it allows structural change without affecting application programs.

- **Only mapping changes schema will not change.**

2.3 Database Languages and Interfaces

- Languages and interfaces for each category of users.
- **Database languages**
- **A) Data definition language (DDL)**
  - Defines conceptual schema and internal schema
  - Used by DBA and database designers

- DBMS DDL compiler processes DDL statements and identifies descriptions of the schema constructs and stores the schema the DBMS catalog.

Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.

2.3 Database Languages and Interfaces

- **Database languages**
- **B) Storage definition language (SDL)**
  - Specifies the internal schema.
  - Mappings between 2 schemas specified in either one of these languages.
  - Recent RDBMSs: there *is no specific language that performs* the role of SDL.
  - Instead, internal schema is specified by a combination of functions, parameters, and specifications related to storage.
  - These permit the DBA staff to control indexing choices and mapping of data to storage.

2.3 Database Languages and Interfaces

- **Database languages**
- **C) View definition language (VDL)**
  - Specifies user views and their mappings to conceptual schema.
  - For a true three-schema architecture, 3rd language is **VDL** to specify user views and their mappings to the conceptual schema
  - Most DBMSs : DDL is used to define both conceptual and external schemas.
  - *In* RDBMSs, SQL is used in the role of VDL to define user or application **views as results** of predefined queries.

2.3 Database Languages and Interfaces

- **Database languages**

- **C) View definition language (VDL)**
  - View is a relation defined in terms of stored tables, except that they are not physically stored.
  - Used in order to simplify complex queries and to define conceptually different views to different classes of users.
  - Ex:

CREATE VIEW <name> **AS** <query>;

```
CREATE VIEW  telephony-purchases AS
    SELECT product, buyer, seller, store
    FROM  Purchase, Product
    WHERE  Purchase.product = Product.name
    AND  Product.category = "telephony"
```

2.3 Database Languages and Interfaces

- **Database languages**
- **D) Data manipulation language (DML)**
  - Allows retrieval, insertion, deletion, modification

- Two types : **high-level or nonprocedural**
- Specify complex database operations concisely without loop
- A **low level** or **procedural DML use loops**.

- DML statements either entered interactively or embedded in a general-purpose programming language (host language).

- DML statements (data sublanguage) are identified within the program and they are extracted by a precompiler and processed by the DBMS.

2.3 Database Languages and Interfaces

- Current DBMSs, preceding types of languages are usually *not considered distinct languages; rather, a comprehensive integrated language is used that includes* constructs for conceptual schema definition, view definition, and data manipulation.
- Structured Query Language (SQL)
  - A typical example of comprehensive database language is the SQL
  - Which represents a combination of DDL, VDL and DML, as well as statements for constraint, specification, schema and other features.

2.3 Database Languages and Interfaces

- DBMS interface is a user interface which allows for the ability to input queries to a database without using the query language itself.
- DBMS interface could be a web client, a local client that runs on a desktop computer, or even a mobile app.

2.3 Database Languages and Interfaces

- **1) Menu-Based Interfaces for Web Clients or Browsing**
  - Present users with list of options (menus)
  - Lead user through formulation of request
  - Query is composed of selection options from menu displayed by system

- **2) Apps for Mobile Devices**
  - Banking, reservations, and insurance companies provide apps
  - Apps have built-in programmed interfaces and they provide a limited menu of options for mobile access to the user data as well as options such as paying bills (for banks) or making reservations (for reservation Web sites).

2.3 Database Languages and Interfaces

- **3) Forms-Based Interfaces**
  - Displays a form to each user.
  - User can fill out form to insert new data or fill out only certain entries.
  - Designed and programmed for naïve users as interfaces to canned transactions.

- **4) Graphical User Interfaces**
  - Displays a schema to the user in diagram form. The user can specify a query by manipulating the diagram. GUIs use both forms and menus.

2.3 Database Languages and Interfaces

- **5) Natural Language Interfaces**
  - Accept requests in written English, or other languages and attempt to understand them.
  - Interface has its own schema, and a dictionary of important words.
- **6) Keyword-based Database Search**
  - Accept strings of natural language (like English) words and match them with documents at specific sites (for local search engines) or Web pages on the Web at large (for engines like Google or Ask).
  - Use predefined indexes on words and use ranking functions to retrieve and present resulting documents.

2.3 Database Languages and Interfaces

- **7) Speech Input and Output**
  - Limited use of speech as an input query and speech as an answer.
  - Application with limited vocabularies such as inquiries for telephone directory flight arrival/departure.
  - Speech input is detected using a library of predefined words and used to set up the parameters that are supplied to the queries.

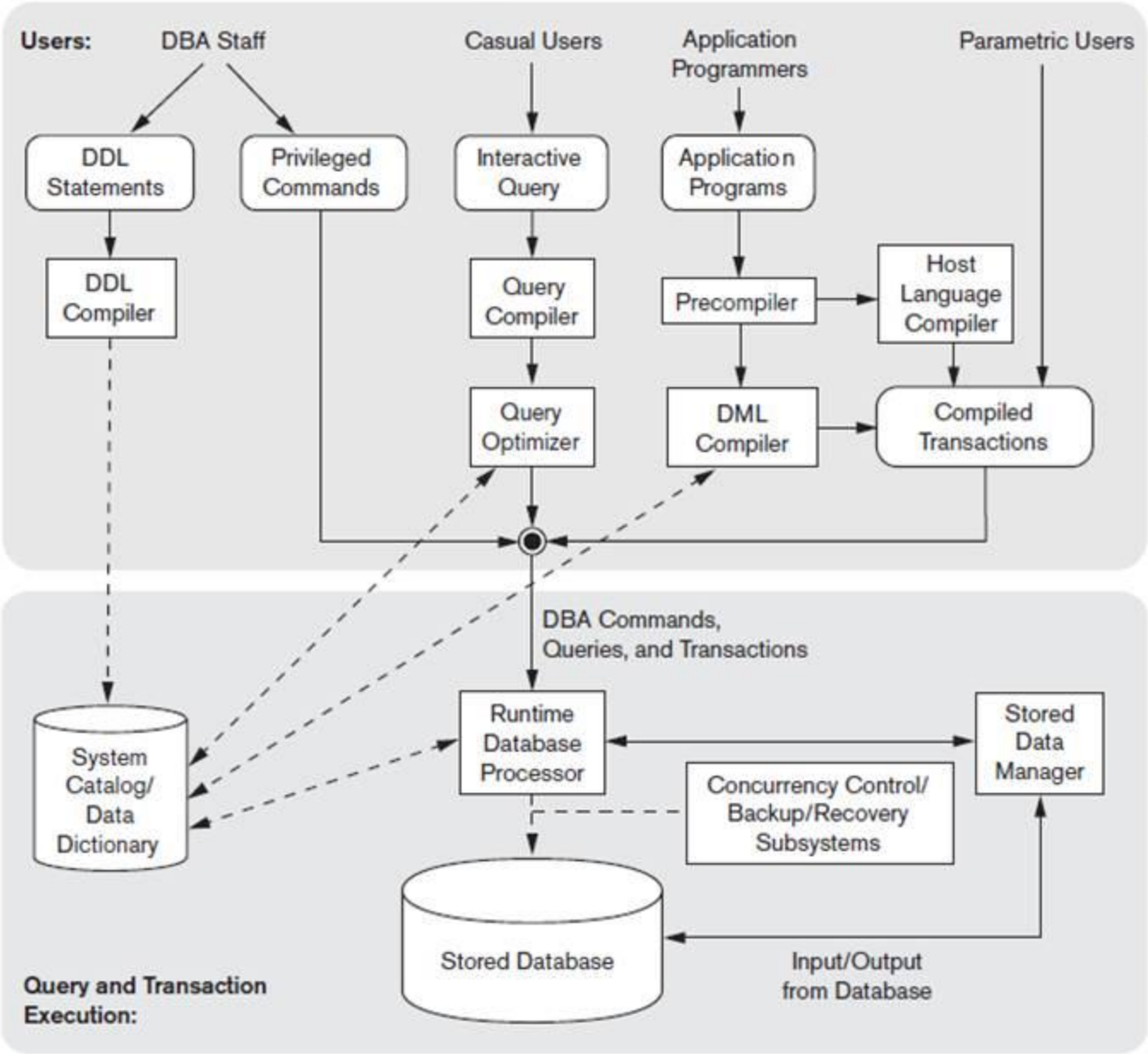
2.3 Database Languages and Interfaces

- **8) Interfaces for Parametric Users**
  - Small set of operations they perform.
  - Analysts and programmers design and implement a special interface for each class of naïve users.
  - Small set of commands included to minimize the number of keystrokes required (I.e. function keys).

- **9) Interfaces for the DBA**
  - Systems contain privileged commands only for DBA staff.
  - Creating accounts, setting parameters, authorizing accounts, changing the schema, reorganizing the storage structures etc.

2.4 Database System Environment

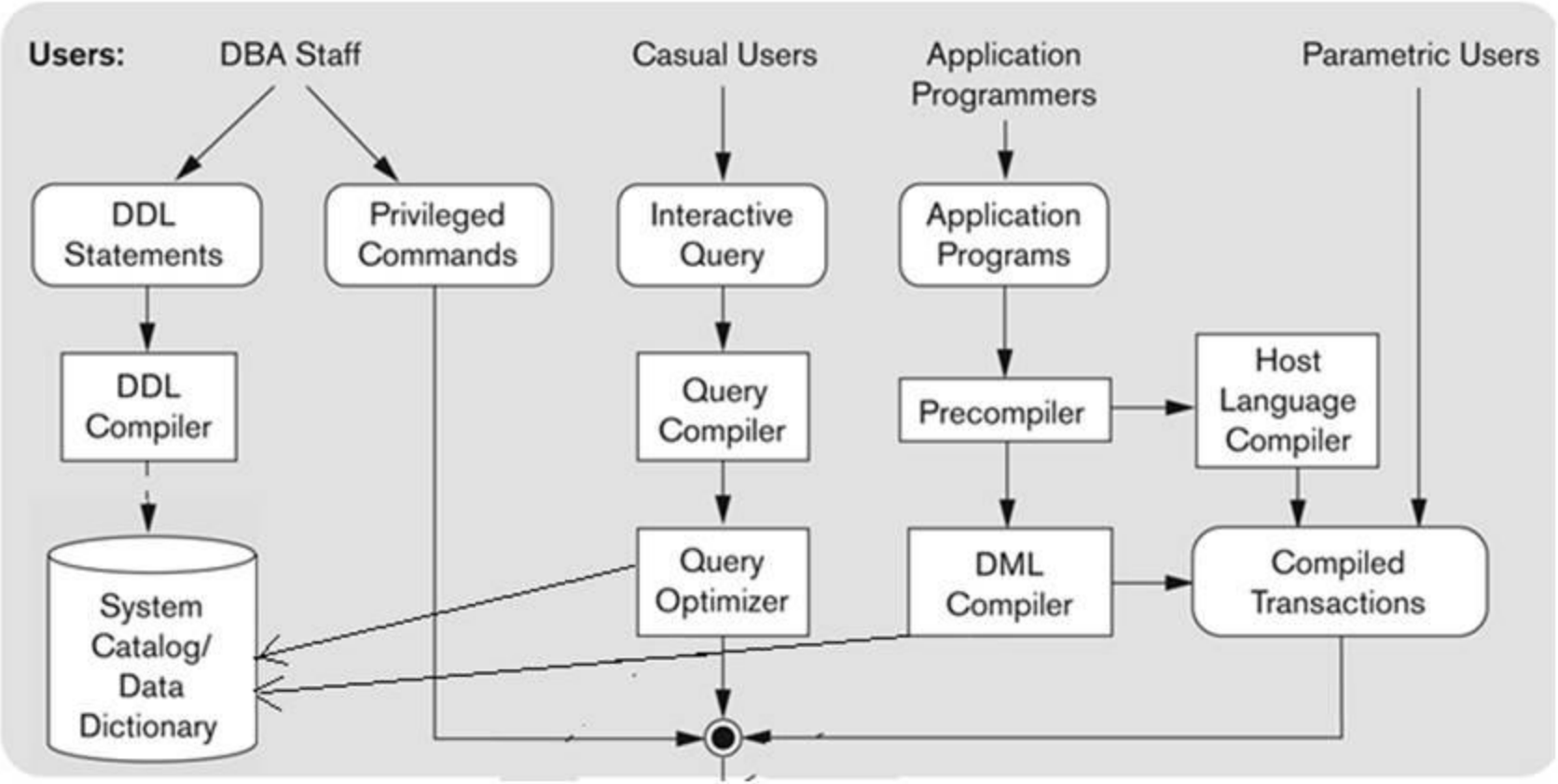
2.4.1 DBMS Component Modules



A) Various users of the DB environment and their interfaces.

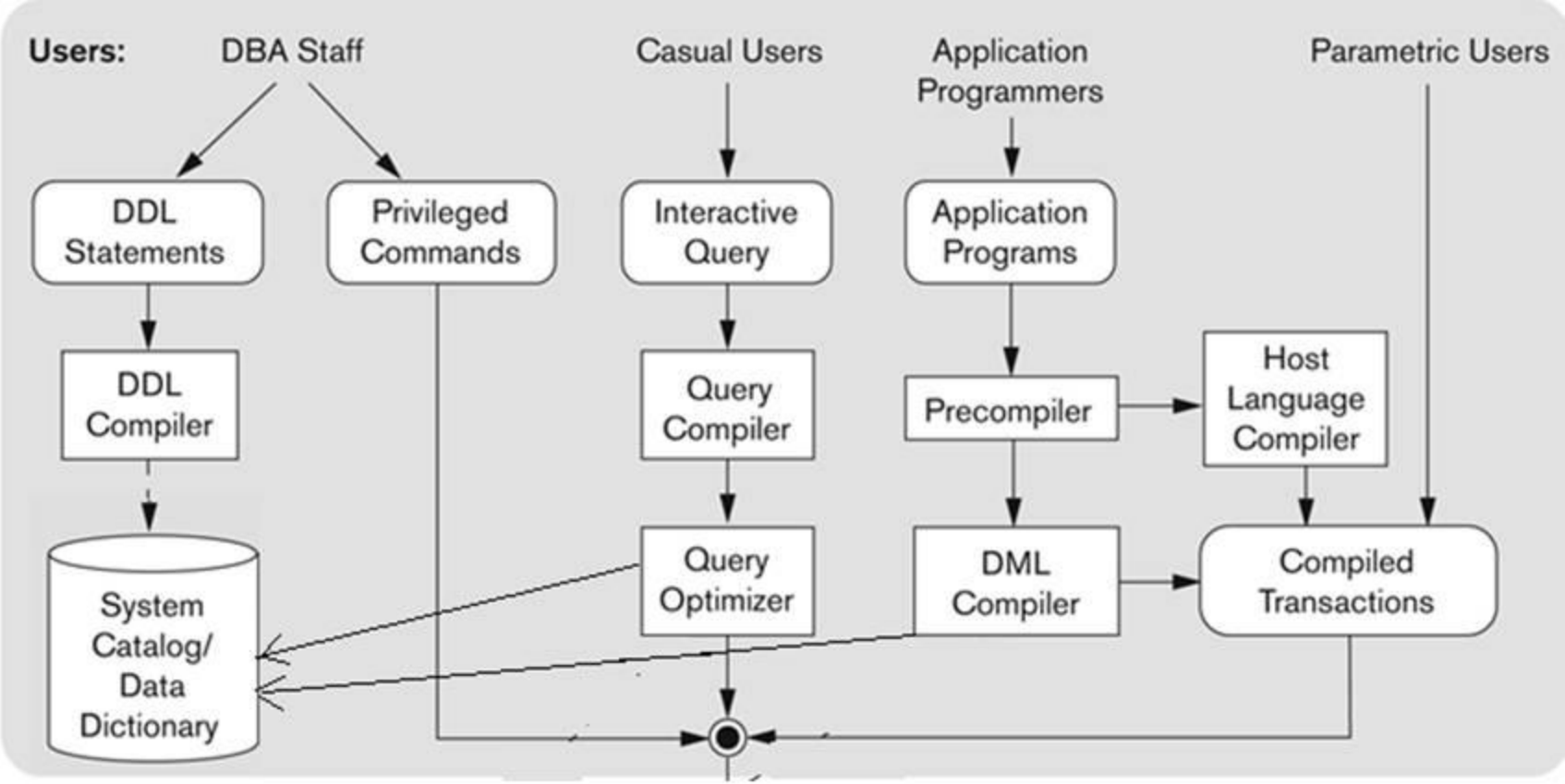
B) Internals (storage of data and processing of transaction).

A) Various users



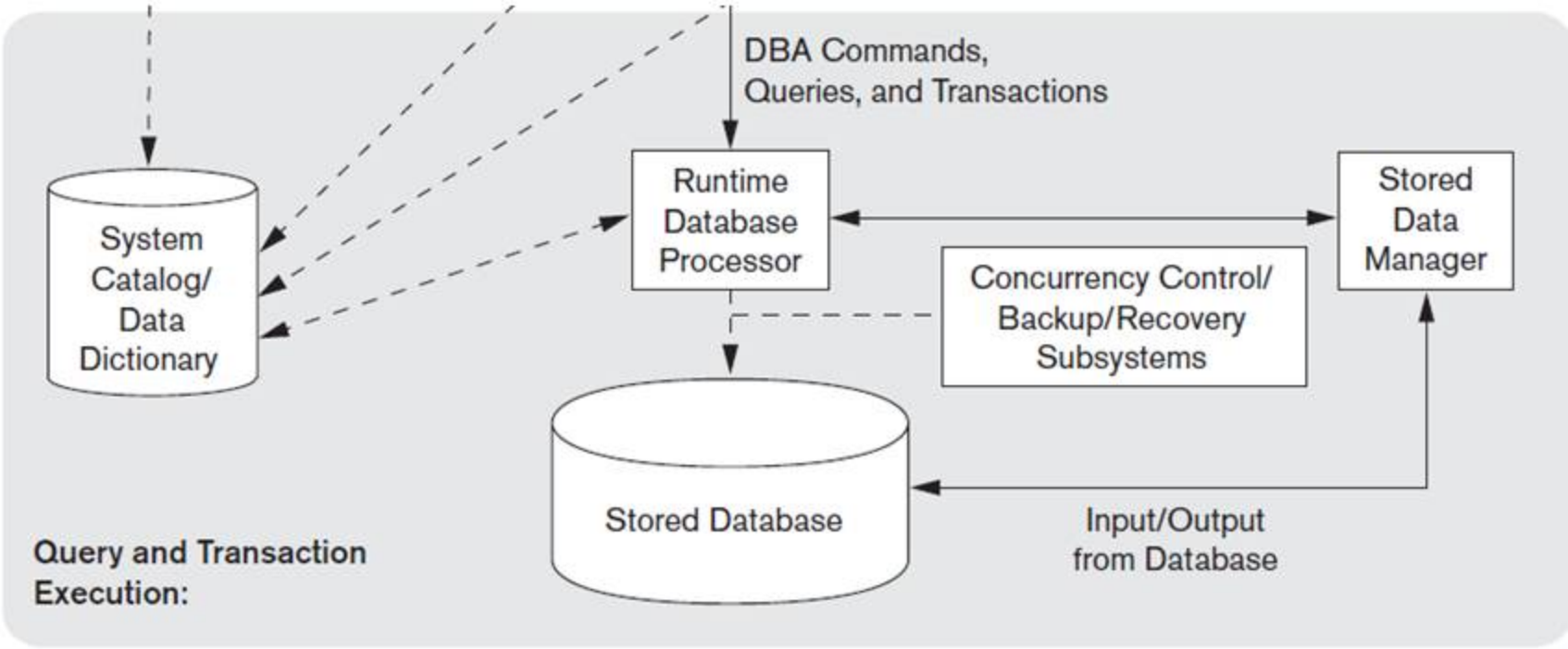
- i) DBA works on defining database and tuning using DDL and privileged commands.
- DDL compiler processes and stores descriptions of schemas in the catalog (name, size, mapping)
- Ii) Casual users uses interface *interactive query (menu, form, mobile)*
- Queries are parsed by *query compiler*.

A) Various users



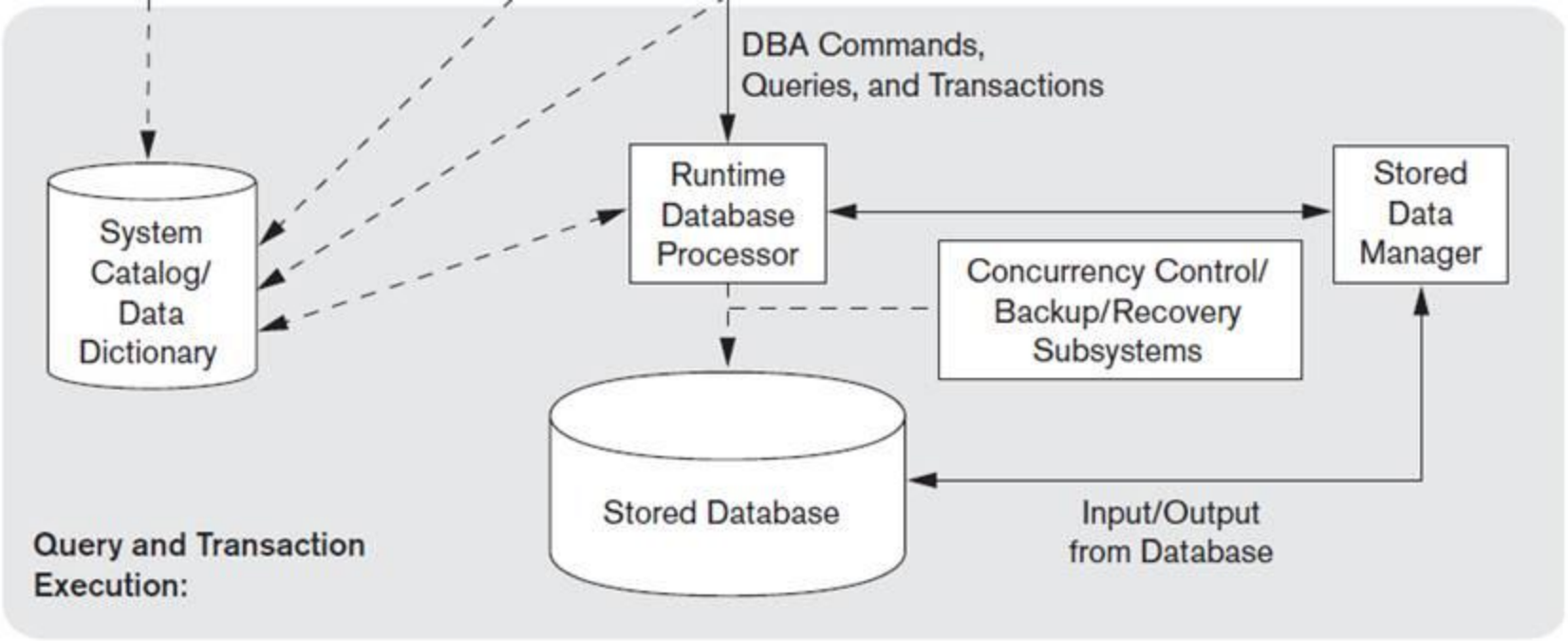
- **Iii) Application programmers** write programs in host languages.
- **Pre-compiler extracts DML commands** and sent to DML compiler **for database access** and program is sent to host language compiler.
- Object code of DML and Host language are linked, forming a **canned transaction.**
- **Iv) Parametric users** do data entry by supplying parameters to predefined transactions.

B) Internals



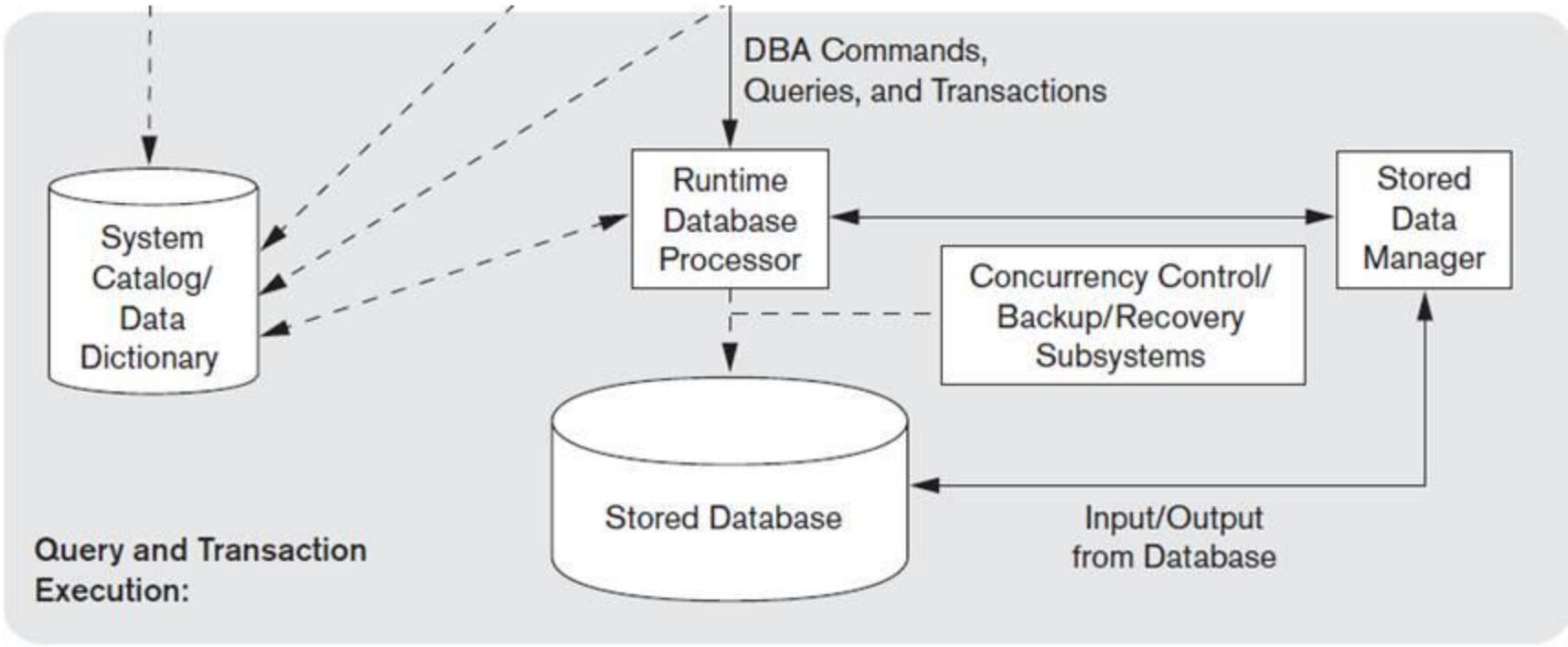
- **Runtime Database Processor**
  - Executes the privileged commands, the executable query plans the canned transactions with runtime parameters.
  - Uses system catalog and stored data manager.
  - Access to the disk goes through the stored data manager.

B) Internals



- **Stored Data Manager**
  - Higher-level stored data manager controls access to DBMS information that is stored on disk (part of the database/catalogue).
  - It may use basic OS services for carrying out low-level data transfer(handling buffers).
  - Once data is in buffers, the other DBMS modules and application programs can process it.

B) Internals



- **Concurrency control/backup/recovery**
  - Integrated into the working of the runtime database processor for transaction management.

- **Help the DBA manage the database system.**

- **Loading: Load existing data files such as text** files or sequential files—into the database. **Current** (source) and **desired** (target) file structure are specified. With this transferring data from one DBMS to another easy.
- Some vendors offer **conversion tools**.

- **Backup. Creates a backup copy of the database, usually by** dumping the entire database onto tape/mass storage medium. **Incremental backups** are often used, where changes since the previous backup are recorded and it is more **complex, but saves storage space.**

2.4.2 Database System Utilities

- **Help the DBA manage the database system.**

- **Database storage reorganization:** **reorganize a** set of database files into **different file** organizations and create **new access paths** to improve performance.

- **Performance monitoring:** **monitors database usage and provides statistics** to the DBA. He uses statistics in making decisions w.r.t reorganize files, add or drop indexes.

- **Other utilities:** sorting files, data compression, monitoring access by users, interfacing with network etc..

- **Other tools (database designers, users, DBMS)**
- **CASE tools:** design phase of database systems.

- **Expanded data dictionary (or data repository) system:**
  - In addition to storing catalog information about schemas and constraints, it stores **design decisions, usage standards, application program descriptions, and user information.**
  - Also called as **information repository accessed *by*** users/DBA rather than by DBMS software.

- **Application development environments, such as PowerBuilder (Sybase) or JBuilder (Borland):**
  - **Provide environment** for developing database applications and include facilities that help in **many facets of database systems,** including database design, GUI development, querying and updating, and application program development.

- DBMS **needs to interface with communications software,** whose function is to allow users from remote locations to access database.
  - **These are connected to the database site through data communications hardware** such as Internet routers, phone lines, long-haul networks, local networks, or satellite communication devices.
  - Many commercial database systems have **communication packages and they are called DB/DC system.**
  - **In addition, some distributed DBMSs are** physically distributed over multiple machines. In this case, communications networks are needed to connect the machines. These are often **local area networks (LANs), but they can also be other types of networks.**