RV College of Engineering®

Go, change the world

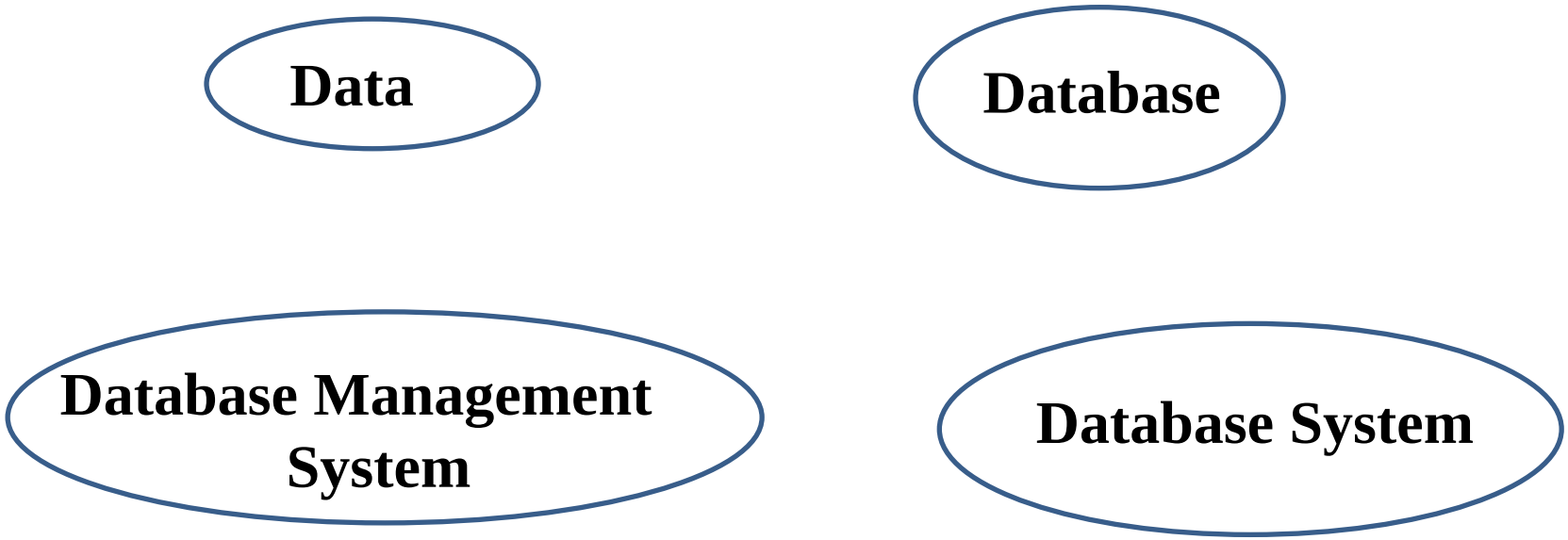
Unit - I

Introduction to Databases, Database Languages and Architecture

Outline

- Introduction
- Basic Definitions
- Characteristics of the Database Approach
- Data models, Schema and instances
- Three schema architecture and Data Independence
- Database Languages and Interfaces
- Database System Environment
- Centralized and Client/ Server Architectures of DBMSs

Database Management System



Basic Definitions

- **Data:**
  - Known facts that can be recorded and have an implicit meaning.
- **Database:**
  - A collection of related data.
- **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):**
  - is a collection of programs that enables users to create and maintain a database.
  - The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.
- **Database System:**
  - The DBMS software together with the data itself. Sometimes, the applications are also included.

- **Defining a database involves specifying**
- the data types, structures, and constraints of the data to be stored in the database.
- The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called **meta-data.**
- **Constructing-** the database is the process of storing the data on some storage medium that is controlled by the DBMS.
- **Manipulating a database includes functions such as querying-** the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.

- Sharing a database allows multiple- users and programs to access the database simultaneously.
- An application program accesses the database by sending queries or requests for data to the DBMS.
- A query typically causes some data to be retrieved; a transaction may cause some data to be read and some data to be written into the database

- Other important functions provided by the DBMS include *protecting the database* and *maintaining it over a long period of time*.
- ***Protection includes system protection-*** against hardware or software malfunction (or crashes) and *security protection* against unauthorized or malicious access.
- A typical large database may have a life cycle of many years, so the DBMS must be able to **maintain the database system by** allowing the system to evolve as requirements change over time.

Simplified database system environment

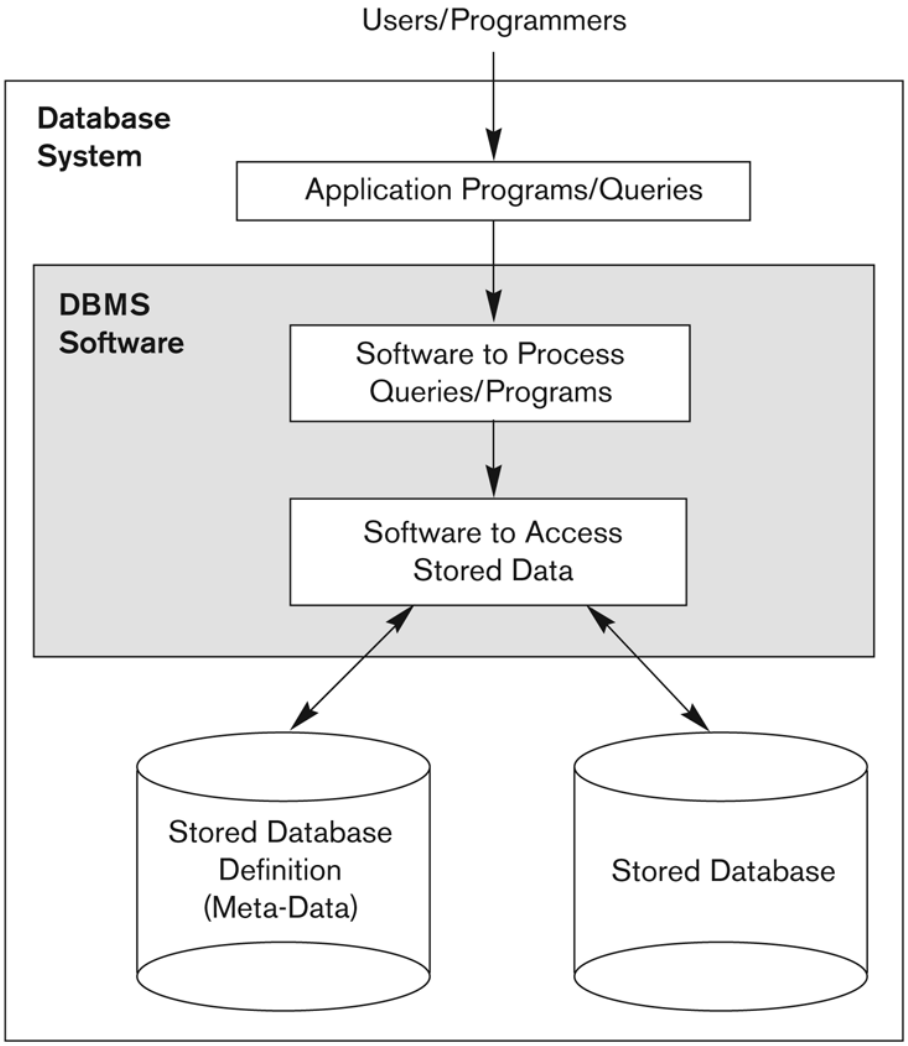


Figure 1.1
A simplified database system environment.

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
  - "Active" processing to take internal actions on data
  - Presentation and Visualization of data
  - Maintaining the database and associated programs over the lifetime of the database application
    - Called database, software, and system maintenance

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

- Note: The above entities and relationships are typically expressed in a conceptual data model, such as the ENTITY-RELATIONSHIP data model (see Chapters 3, 4)

Example of a simple database

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.

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.
- **Insulation between programs and data:**
  - traditional file processing - structure of a file may require *changing all programs that access this file*
  - Called **program-data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.

Example of a simplified database catalog

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

Main Characteristics of the Database Approach (continued)

- **Data Abstraction:**
  - User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented - **program-operation independence.**
  - The characteristic that allows program-data independence and program-operation independence is called **data abstraction**
  - A DBMS provides users with a **conceptual representation of data that does not include many of the details of how the data is stored or how the operations are implemented**

- A **data model-** is a type of data abstraction that is used to provide this conceptual representation. The data model uses logical concepts, such as objects, their properties, and their interrelationships, that may be easier for most users to understand than computer storage concepts.
- Hence, the data model *hides storage and implementation details that are not of interest to most database 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.
  - A view may be a subset of the database or it may contain **virtual data that is derived from the database files but is not explicitly stored**
  - Some users may not need to be aware of whether the data they refer to is stored or derived.
  - A multiuser DBMS whose users have a variety of applications must provide facilities for defining multiple views.

Main Characteristics of the Database Approach (continued)

- **Sharing of data and multi-user transaction processing:**
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *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.

Data Models

- The architecture of DBMS packages has evolved from the early monolithic systems, where the whole DBMS software package is one tightly integrated system, to the modern DBMS packages that are modular in design, with a client-server system architecture.
- A **client module is typically designed so that it will run on a user workstation or personal computer.**
- The other kind of module, called a **server module, typically handles data storage, access, search, and other functions**

Data Models

- **Data Abstraction:**
  - The suppression of details of data organization and storage and the highlighting of essential features for better understanding
- **Data Model:**
  - a collection of concepts that can be used to describe the structure of a database—provides the necessary means to achieve this abstraction.
  - By *structure of a database - mean the data types, relationships, and constraints that should hold on the data. Most data models also include a set of* ***basic operations for specifying retrievals and updates on the database.***

Data Models

- In addition to the basic operations provided by the data model, it is becoming more common to include concepts in the data model to specify the **dynamic aspect or behavior of a database application.**
- This allows the database designer to specify a set of valid user-defined operations that are allowed on the database objects

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.
  - Concepts provided by low-level data models are generally meant for computer specialists, not for typical end users
- **Representational (Implementation) data models:**
  - Provide concepts that may be understood by end users but that are not too different from the way data is organized within the computer
  - Hide some details of data storage but can be implemented on a computer system in a direct way
  - Used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

- **Conceptual (high-level, semantic) data models:**
  - use concepts such as entities, attributes, and relationships.
  - An **entity** represents a real-world object or concept, such as an employee or a project, that is described in the database.
  - An **attribute** represents some property of interest that further describes an entity, such as the employee's name or salary.
  - A **relationship** among two or more entities represents an interaction among the entities
- **Representational (Implementation) data models:**
  - **relational data model, as well as the so-called** legacy data models—the **network and hierarchical models**—that have been widely used in the past.
  - Representational data models represent data by using record structures and hence are sometimes called **record-based data models.**
  - object data models as a new family of higher-level implementation data models that are closer to conceptual data models.
- Physical data models describe how data is stored in the computer by representing information such as record formats, record orderings, and access paths.
  - An **access path is a structure that makes the search for particular database records efficient.**

Schemas versus Instances

- Database Schema:
  - The ***description*** of a database.
  - Includes descriptions of the database structure, data types, and the constraints on the database.
  - specified during database design and is not expected to change frequently
- Schema Diagram:
  - An ***illustrative*** display of a database schema.
  - displays the structure of each record type but not the actual instances of records
- Schema Construct:
  - A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.
- 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).

- Database State:
  - 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 or populated into the system.
- Valid State:
  - A state that satisfies the structure and constraints of the database.
- Schema and State Distinction
  - The ***database schema*** changes very infrequently. The ***database state*** changes every time the database is updated.
  - **Schema** is also called **intension** whereas **State** is also called **extension**.
  - the schema is not supposed to change frequently, it is not uncommon that changes need to be applied to the schema once in a while as the application requirements change - **schema evolution.**

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.

Example of a database state

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

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

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

PREREQUISITE

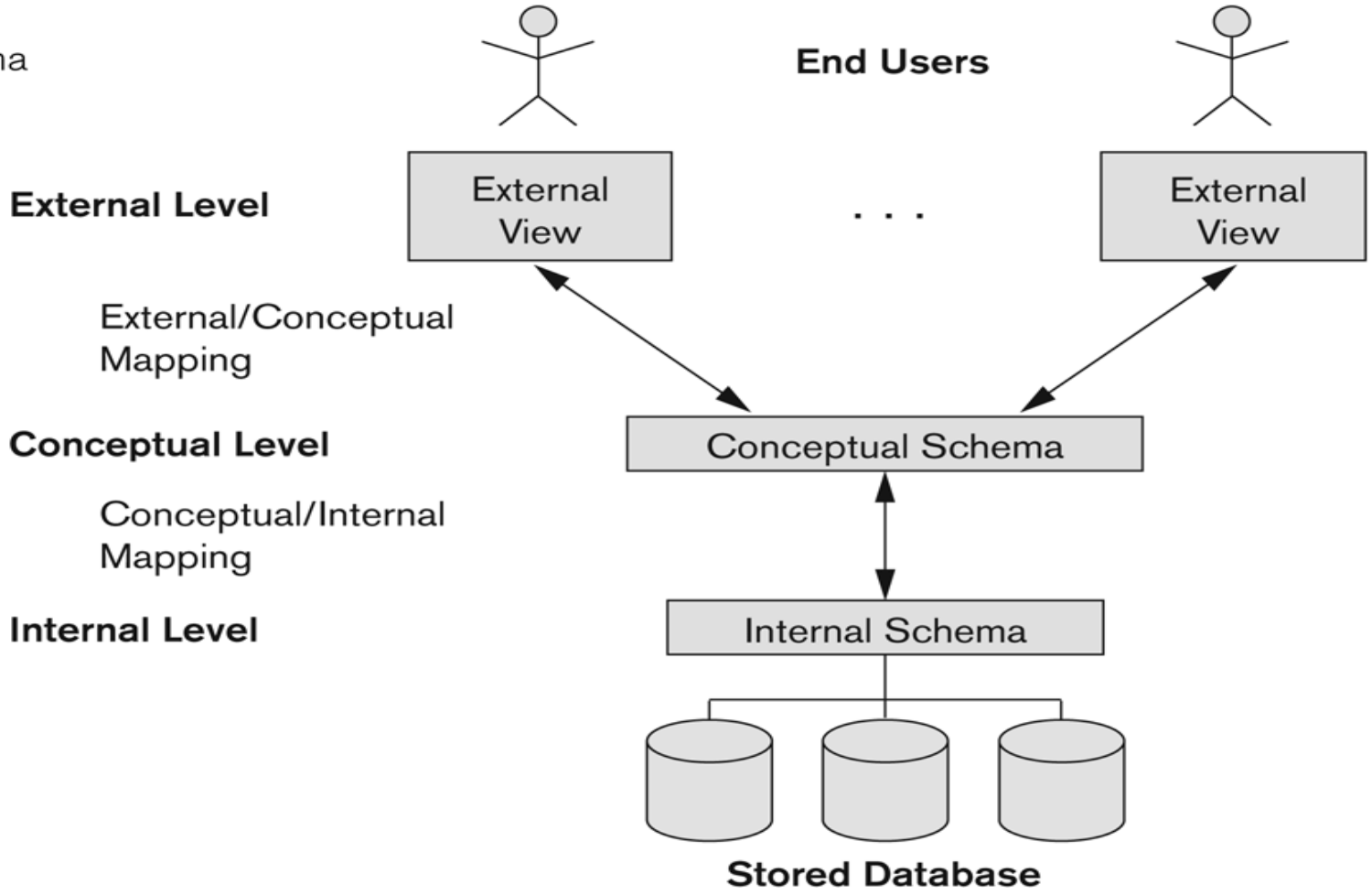
Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Three-Schema Architecture

- Proposed to help achieve the DB characteristics
  - Specially to separate user applications and the physical database
- It defines DBMS schemas at ***three*** levels:
  - **Internal schema or Internal level** at the internal level to describe physical storage structures and access paths (e.g indexes).
    - Typically uses a **physical** data model and describes the complete details of data storage and access paths for the data base
  - **Conceptual Level or Conceptual schema** at the conceptual level to describe the structure for the whole database for a community of users. It hides the details of the physical storage structures and concentrates on describing entities, data types, relationships, and constraints.
    - Uses a **conceptual** or an **implementation** data model.
  - **External schemas** at the external level to 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.
    - Usually uses the high level data model or an implementation data model

The three-schema architecture

Figure 2.2
The three-schema architecture.



•The three schemas are only descriptions of data; the only data that actually exists is at the physical level.

In a DBMS based on the three-schema architecture, each user group refers only to its own external schema.

Hence, 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 stored database.

If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view.

The processes of transforming requests and results between levels are called **mappings.**

Data Independence

- **Data Independence-** The ability to change the schema at one level of the database system without having to change the schema at the next higher level
  - **Logical Data Independence:**
    - The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.
    - We may change the conceptual schema to expand the database(by adding a record type or data item), or to reduce the database(by removing record type or data item)
  - **Physical Data Independence:**
    - The capacity to change the internal schema without having to change the conceptual schema.
    - Changes to internal schema may be needed because some physical files has to be reorganized

- Whenever we have a multiple-level DBMS, its catalog must be expanded to include information on how to map requests and data among the various levels
- The DBMS uses additional software to accomplish these mappings by referring to the mapping information in the catalog.
- Data independence is accomplished because, when the schema is changed at some level, the schema at the next higher level remains unchanged; only the *mapping between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed.*
- The three-schema architecture can make it easier to achieve true data independence, both physical and logical.
- However, the two levels of mappings create an overhead during compilation or execution of a query or program, leading to inefficiencies in the DBMS

DBMS Languages

- **Data Definition Language (DDL):** a language that is used to define database schemas. The DDL statement is used to identify description of the schema construct and store the schema description in the DBMS catalog.
- DDL is used to specify the conceptual schema only
- the **storage definition language (SDL),** is used to specify the internal schema. The mappings between the two schemas may be specified in either one of these languages.
- **For a true three-schema architecture, we would need a third language, the view definition language (VDL),** to specify user views and their mappings to the conceptual schema, but in most DBMSs the DDL is used to define both conceptual and external schemas.
- **Data Manipulation Language (DML):** a language that is used to manipulate (retrieve, insert, delete, and modify) data.
  - A **high-level or non-procedural DML** can be used on its own to specify complex database operations in a concise manner, such as SQL.
  - A **low-level or procedural DML** typically retrieves individual records or objects from the database and processes each separately, such as PL/SQL.
  - Embedded Languages**:** DML with Java, VB, C++, etc.

DBMS Interfaces

- Stand-alone query language interfaces
  - Example: Entering SQL queries at the DBMS interactive SQL interface (such as, SQL*Plus in ORACLE)
- Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces
  - Menu-based, popular for browsing on the web
  - Forms-based, designed for naïve users
  - Graphics-based
  - Natural language: requests in written English
  - Parametric interfaces, e.g., bank tellers using function keys.
  - Interfaces for the DBA

DBMS Component Modules

- A typical DBMS consists of the following components:
  - DDL compiler: process schema definitions, specified in the DDL statements, and stores descriptions of the schemas in the system catalog.
  - The catalog includes information such as the names of files, data items, storage details of each file, mapping information among schemas, and constraints, in addition to many other types of information that are needed by the DBMS modules.
  - DBMS software modules then look up the catalog information as needed.

- Run-time database processor: handles database access at run time. It receives retrieval and update operations and carries them out on the database.
- Access to disk goes through the stored data manager.
- The query compiler handles high-level queries that are entered interactively.
- It parses, analyzes, and compiles or interprets a query by creating database access code, and then generates calls to the run-time processor for executing the code.
- The pre-compiler extracts DML commands from an application program written in a host programming language
- These commands are sent to the DML compiler for compilation into object code for database access.
- The rest of the program is sent to the host language compiler.
- The object codes for the DML commands and the rest of the program are linked, forming a canned transaction whose executable code includes calls to the runtime database processor.

DBMS Component modules

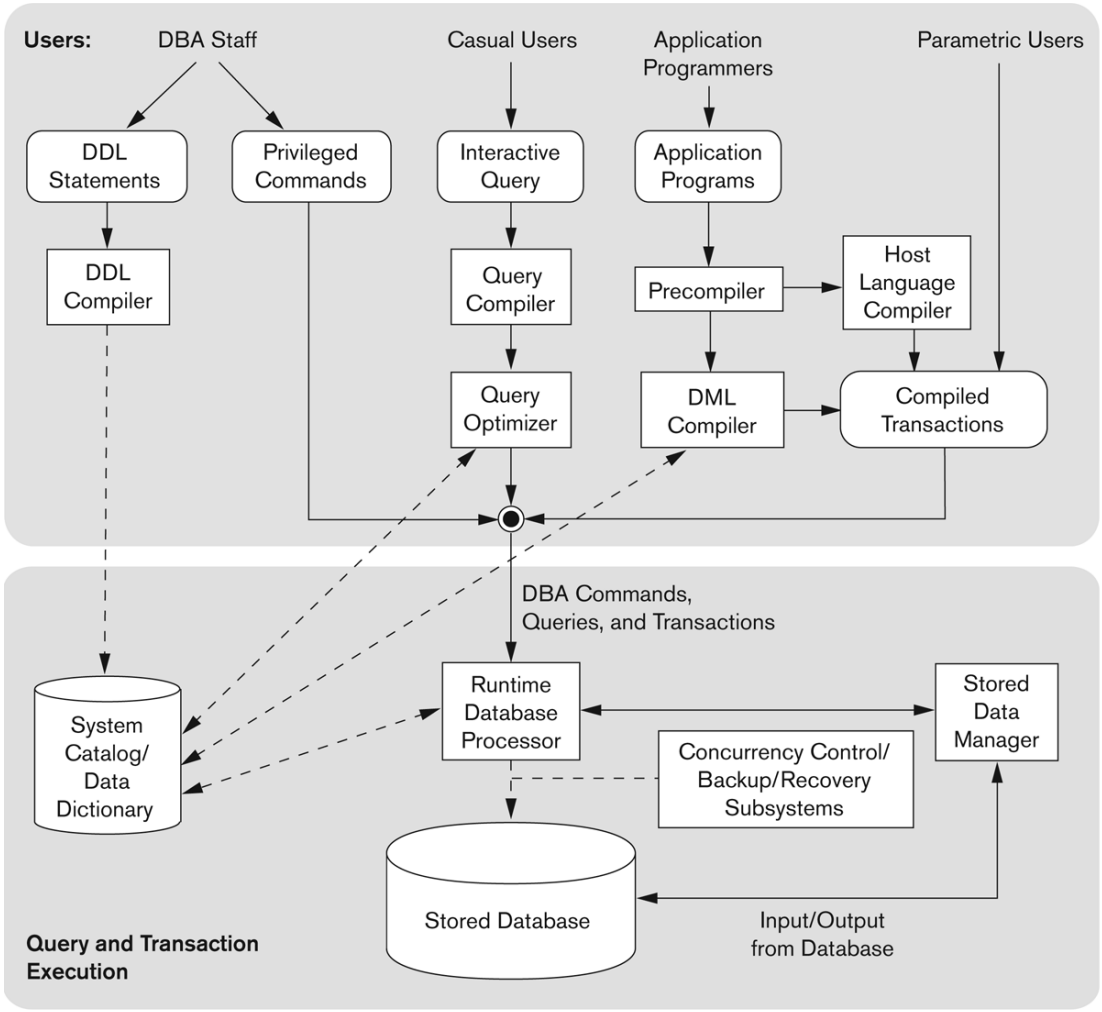


Figure 2.3
Component modules of a DBMS and their interactions.

Database System Utilities

- There are some functions that are not provided through the normal DBMS components rather they are provided through additional programs called utilities. Some of these are:
  - Loading or import utility: used to load or import existing data files into the database.
  - Usually, the current (source) format of the data file and the desired (target) database file structure are specified to the utility, which then automatically reformats the data and stores it in the database.
  - Backup utility: used to create backup copies of the database, usually by dumping the entire database onto tape.
  - File reorganization utility: is used to reorganize a database file into a different file organization to improve performance.
  - Performance monitoring utility: is used to monitor database usage and provides statistics to the DBA.
  - Other utilities may be available for sorting files, handling data compression, monitoring access by users

Database System Utilities

- There are some functions that are not provided through the normal DBMS components rather they are provided through additional programs called utilities. Some of these are:
  - Loading or import utility: used to load or import existing data files into the database.
  - Backup utility: used to create backup copies of the database, usually by dumping the entire database onto tape.
  - File reorganization utility: is used to reorganize a database file into a different file organization to improve performance.
  - Performance monitoring utility: is used to monitor database usage and provides statistics to the DBA.

Other Tools

- Data dictionary / repository:
  - Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, etc.

- CASE tools:
  - Used in the design phase of database
    - Examples: Rational Rose, TOAD

- Application Development Environments:
  - Provides facilities for developing database applications including database design, graphical user interface development, querying and updating, and application program development.
    - PowerBuilder (Sybase), JBuilder (Borland), JDeveloper 10G (Oracle)

Client Server Architecture for DBMS

- The client-server architecture is increasingly being incorporated into commercial DBMS packages.
- In relational DBMSs, many of which started as centralized systems, the system components that were first moved to the client side were the user interface and application programs.
- Because SQL provided a standard language for RDBMSs, it created a logical dividing point between client and server.
- Hence, the query and transaction functionality remained at the server side.
- In such an architecture, the server is often called a **query server or transaction server, because it provided these two functionalities.**
- In RDBMSs, the server is also often called an SQL server, since most RDBMS servers are based on the SQL language and standard

- In such a client-server architecture, the user interface programs and application programs can run at the client side.
- When DBMS access is required, the program establishes a connection to the DBMS—which is on the server side—and once the connection is created, the client program can communicate with the DBMS.
- A standard called Open Database Connectivity (ODBC) provides an Application Programming Interface (API), which allows client-side programs to call the DBMS, as long as both client and server machines have the necessary software installed.
- Most DBMS vendors provide ODBC drivers for their systems.
- Hence, a client program can actually connect to several RDBMSs and send query and transaction requests using the ODBC API, which are processed at the server sites.
- Any query results are sent back to the client program, which can process or display the results as needed.
- Another related standard for the JAVA programming language, called JDBC, has also been defined.
- This allows JAVA client programs to access the DBMS through a standard interface.

- The second approach to client-server was taken by some object-oriented DBMSs.
- Because many of those systems were developed in the era of client-server architecture, the approach taken was to divide the software modules of the DBMS between client and server in a more integrated way.
- For example, the server level may include the part of the DBMS software responsible for handling data storage on disk pages, local concurrency control and recovery, buffering and caching of disk pages, and other such functions. Meanwhile, the client level may handle the user interface, data dictionary functions, DBMS interaction with programming language compilers, global query optimization/concurrency control/recovery, structuring of complex objects from the data in the buffers, and other such functions.
- In this approach, the client-server interaction is more tightly coupled and is done internally by the DBMS modules—some of which reside in the client—rather than by the users.
- The exact division of functionality varies from system to system. In such a client-server architecture, the server has been called a data server, because it provides data in disk pages to the client, which can then be structured into objects for the client programs by the client-side DBMS software itself.