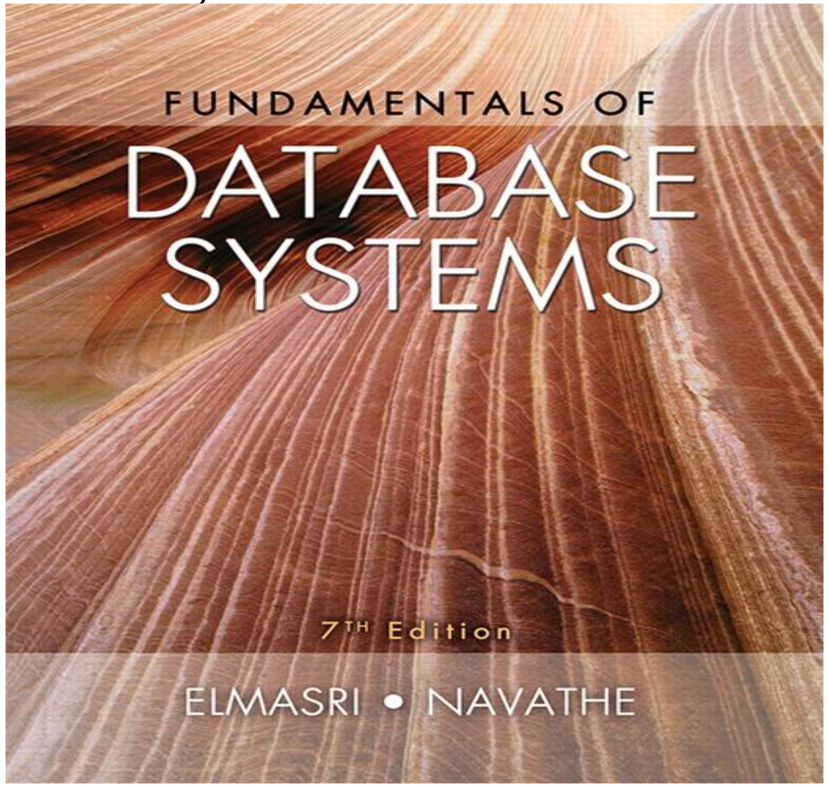
Comp-3150: Database Management Systems

- Ramez Elmasri , Shamkant B. Navathe(2016) Fundamentals of Database Systems (7th Edition), Pearson, isbn 10: 0-13-397077-9;isbn-13:978-0-13-397077-7.

- Chapter 2:
  Database System
  Concepts and Architecture



Chapter 2: Database System Concepts and Architecture Outline

- 1. Data Models, Schemas and Instances
- 2. Three-Schema Architecture
- 3. Data Independence
- 4. DBMS Languages and Interfaces
- 5. Database System Utilities and Tools
- 6. Centralized and Client-Server Architectures
- 7. History of Data Models

1. Data Models, Schemas and Instances

- 1. Data model: defines the data structure, operations and integrity constraints (IC) of the database (consisting of the data elements, their data types, relationships).
- Example data models are the relational data model, entity-relationship (ER) model for defining entities and relationships.
- Example operations on the database are for data retrievals and updates of the database.
- Example IC (integrity constraint) is that to take a course, you need to be a student. This is referential integrity constraint. Other types of IC exist, such as primary key IC for uniquely identifying database entities, domain IC, entity IC, and semantic IC.

1. Data Model, Categories, Schemas and Instances

2. Data Model Categories (Four types)

- **i. Conceptual (high-level, semantic) data models:**
  - Example is ER-model which describes data in concepts close to how users perceive data. Data are defined in terms of entities, attributes and relationships.
  - An entity: represents a real- world object such as an employee or a project from the mini world in the database.
  - An attribute: represents some property of interest that describes an entity (e.g.: employee's name, salary, birthdate.)
  - A relationship: among two or more entities represents an association among the entities. For example, the fact as "each employee works on one or more project" is represented as a works on relationship between an employee and a project. The entity-relationship model is used during database design to define this conceptual model of data.

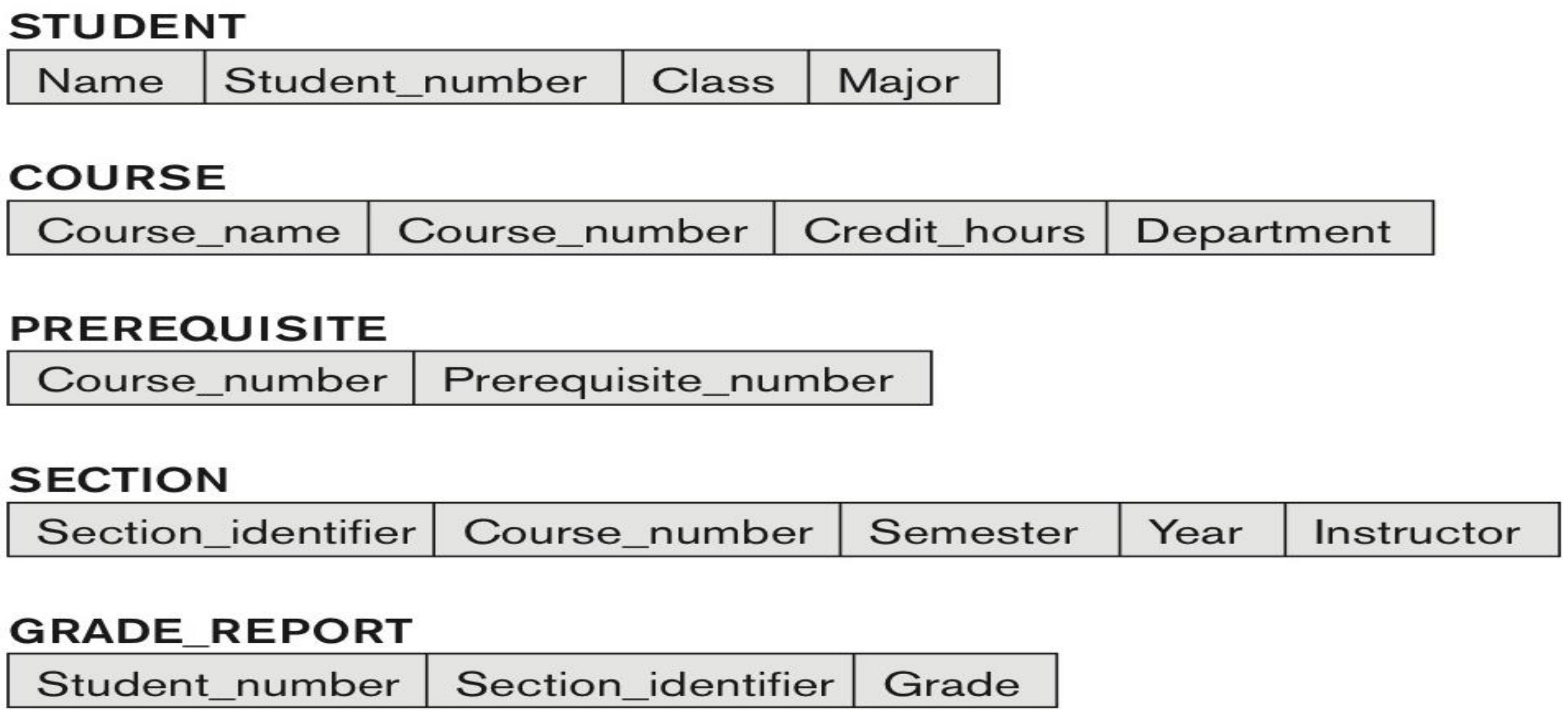
1. Data Model, Categories, Schemas and Instances

- **ii. Physical (low-level, internal) data models:**
  - Provide concepts that describe details of how data is stored in the computer as files by showing record formats, orderings, access paths such as indexing structure for data retrieval.
- **iii. 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).
- **iv. Self-Describing Data Models:**
  - Combine the description of data with the data values. Examples include XML, key-value stores and some NOSQL systems. Used to cater for non-traditional data types like posts, long character blobs, and web data.

1. Data Model, Categories, Schemas and Instances

- 3. Schemas and Instances
- Database Schema:
  - is the description of the database and is made up of all the schemas of the tables or files in the whole database. For example, the schema diagram for the student record database of Fig. 1.2 is given as Fig 2.1 on next slide.
- 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 is called a schema construct.

1. Data Model, Categories, 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.

1. Data Model, Categories, Schemas and Instances

- 3. Schemas and Instances
- 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*

1. Data Model, Categories, Schemas and Instances

- 3. Schemas and Instances (Database State Versus Schema)
- 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 into the system.
- Valid State:
  - A state that satisfies the structure and constraints of the database.

1. Data Model, Categories, Schemas and Instances

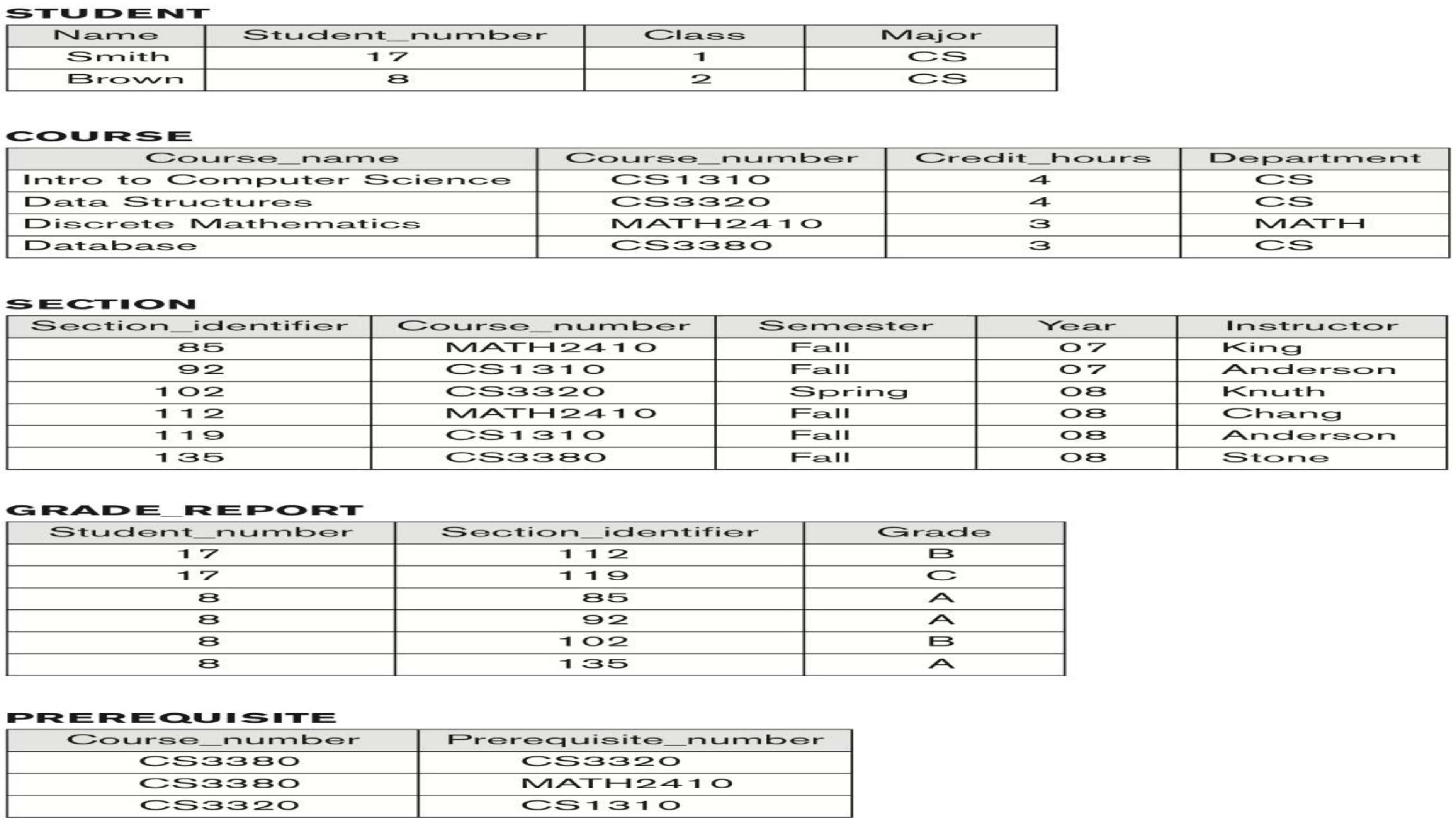
- 3. Schemas and Instances (Database State Versus Schema)

- Distinction
  - The ***database schema*** changes very infrequently. Example is Fig. 2.1
  - The ***database state*** changes every time the database is updated. Example of a database state is Fig. 1.2 shown on next slide.

- **Schema** is also called **intension**.
- **State** is also called **extension**.

1. Data Model, Categories, Schemas and Instances: Example of a Database State

Figure 1.2 A database that stores student and course information.



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. Three-Schema Architecture

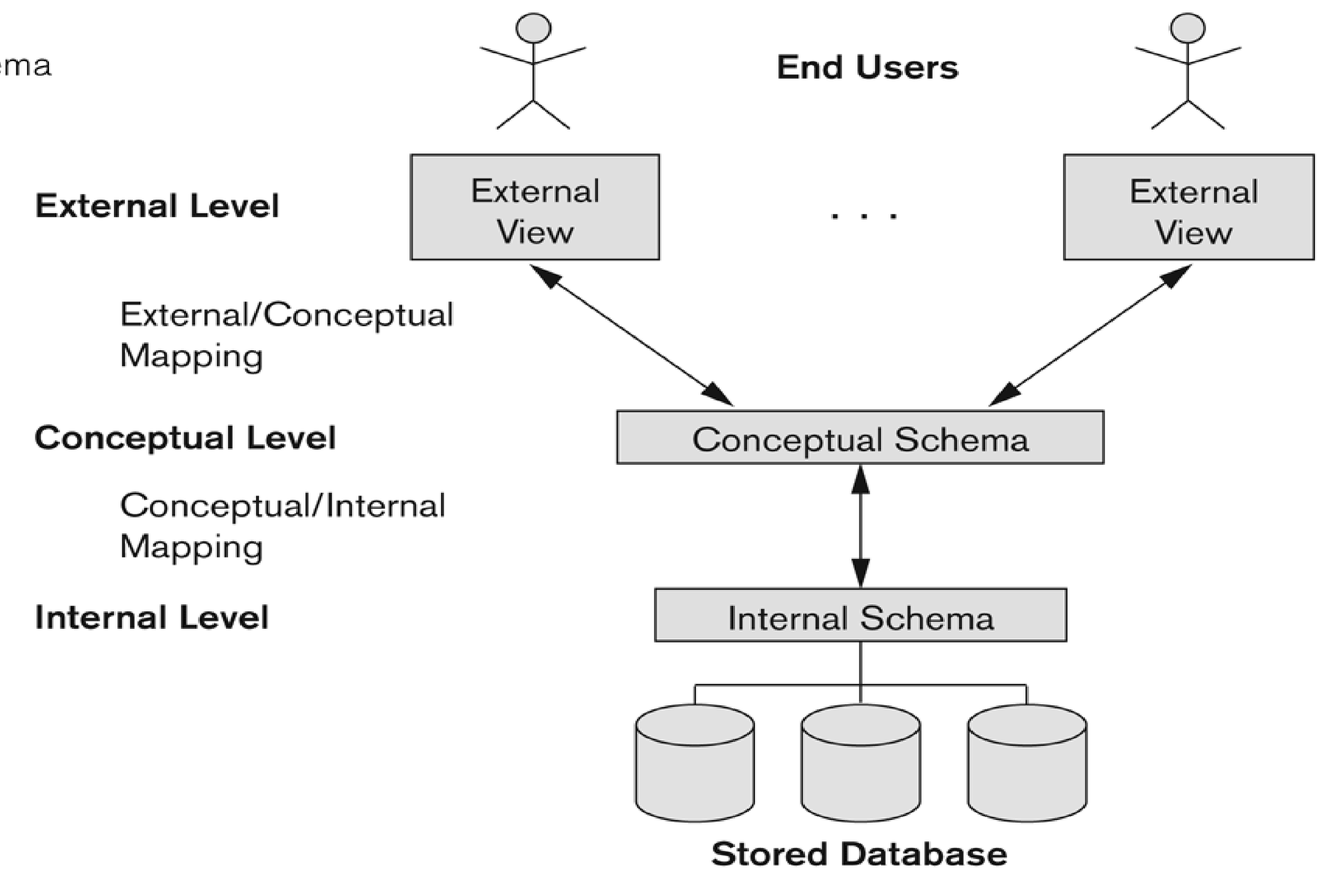
- DBMS uses 3-schema architecture to achieve:
  - data-program separation,
  - data independence,
  - use of catalogue for data description of the schema,
  - and support for multiple user views.

2. Three-Schema Architecture

- DBMS uses 3-schema level architecture to achieve data-program separation, data independence, use of catalogue for data description schema and support for multiple user views. The 3 levels consist of
- a) The external schemas for end users (e.g.: external views and different user views of data through queries) at the external level.
- b) The conceptual schemas at the conceptual level representing the whole database for a community of users using a conceptual or implementation data model such as relational.
- c) The internal schemas at the internal level which describe physical storage structures and access paths of the database (e.g., indexes). See Fig 2.2 next.

2. Three-Schema Architecture

Figure 2.2
The three-schema architecture.



3. Data Independence

- This is the ability of the DBMS to change the schema at a lower level of DB system without having to change the schema at the next higher level.
- Only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence.
- Hence, the application programs need not be changed since they refer to the external schemas
- DBMS provides the following two types of data independence.
- 1. Logical data independence
- 2. Physical data independence

3. Data Independence

- 1. Logical data independence:

- The ability to change the conceptual schema (e.g. get names of students with GPA>85%; and get names and address of students in year 2) without having to change the external schema or application program (when student has an additional attribute “address” for example)

3. Data Independence

- ●2. Physical data independence: It has the ability to change the internal/physical schema without having to change the conceptual schema.
    - ● For example an access path to improve retrieval speed of SECTION records by semester and year should not require a query like “list all sections offered in fall 2008”to be changed.
- ● All 3 levels of abstraction represent data descriptions only.
- ● However, the actual data are stored only at the physical level. The process of transforming requests and results between levels are called mapping and done by the DBMS.

4. DBMS Languages and Interfaces

- 1. Data definition language (DDL):
- The DDL is used to define both the conceptual (e.g., relational) and external (e.g., views) schemas.

- SQL is used in the relational model for the DDL tasks which have three subtasks for each of the three levels:
  - of physical (as storage definition language-SDL done by DBA), conceptual (DDL) and external (as view definition language –VDL).

4. DBMS Languages and Interfaces

- 2. Data manipulation language (DML):
- SQL database language is used for DML operations of data insertion, deletion, modification and retrieval in the relational model.
- a) Standalone DML query language interface: can be non-procedural and used on its own for complex database operations. Example: Oracle SQLPlus. This is called query language.
- b) Programmer interfaces for embedding DML in programming languages:
  - i. **Embedded Approach:** DML can be embedded in programming language like C as with Oracle Pro*C.
  - ii. **Procedure Call Approach:** A library of functions can also be provided to access the DBMS from a programming language like JAVA, C++ (e.g. through JDBC and ODBC connectivities).

4. DBMS Languages and Interfaces

  - **iii. Database Programming Language Approach**: e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components
  - **iv. Scripting Languages:** PHP (client-side scripting) and Python (server-side scripting) are used to write database programs.
- c) Other user-friendly interfaces for interaction with DML built with computer aided software engineering (CASE) tools like Sybase Powerbuilder, Borland Jbuilder, Oracle forms and JDeveloper, e.g.,
  - i. Menu-based, forms-based, graphics-based, etc.
  - ii. Forms-based, designed for naïve users used to filling in entries on a form

4. DBMS Languages and Interfaces

- iii. Graphics-based including Point and Click, Drag and Drop, etc., Specifying a query on a schema diagram
- iv. Natural language: requests in written English
- v. Combinations of the above: including natural language, speech, web browser with keyword search, parametric interfaces with function keys, eg, used by bank tellers.

- d) Mobile Interfaces: allowing users to perform transactions using mobile apps

4. DBMS Languages and Interfaces

- Interfaces for the DBA:
  - Creating user accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access paths

5. Database System Utilities and Tools

- In addition to the DBMS modules shown in Fig 2.3 on next slide, the DB system provides utilities that help the DBA manage the database system. The utilities include:
  - 1. Loading data stored in files into a database. Includes data conversion tools.
  - 2. Backing up the database periodically on tape.
  - 3. Reorganizing database file structures.
  - 4. Performance monitoring utilities.
  - 5. Report generation utilities.
  - 6. Other functions, such as sorting, user monitoring, data compression, etc.

5. Database System Utilities and Tools

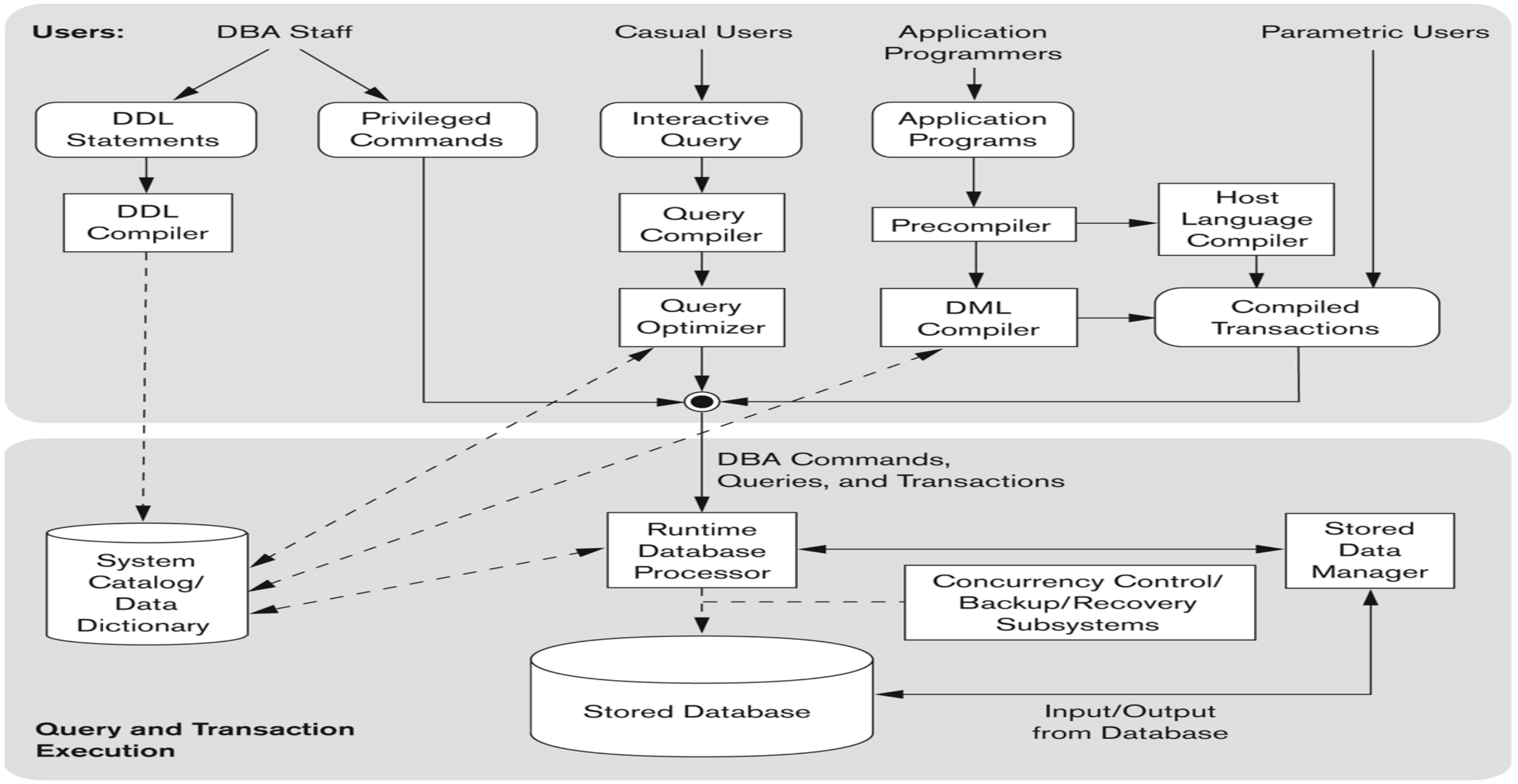


Figure 2.3
Component modules of a DBMS and their interactions.

5. Database System Utilities and Tools

- Other Tools provided by the DB system include:
- 1. Data dictionary / repository:
  - Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.

  2. Application Development Environments and CASE (computer-aided software engineering) tools like Forms, JBuilder, Jdeveloper, etc.

6. Centralized and Client-Server Architectures

- Centralized DBMS:
  - Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.
  - User can still connect through a remote terminal – however, all processing is done at centralized site.

6. Centralized and Client-Server Architectures

- Basic 2-tier Client-Server Architectures
- Specialized Servers with Specialized functions provide database query and transaction services to the clients. Example servers are:
  - Print server
  - File server
  - DBMS server
  - Web server
  - Email server
- Clients (PCs or workstations) can access the specialized servers (Workstations or PCs) as needed through a client software module

7. History of Data Models

- Network Model
- Hierarchical Model
- Relational Model
  - Now in several commercial products (e.g. DB2, ORACLE, MS SQL Server, SYBASE, INFORMIX).
  - Several free open source implementations, e.g. MySQL, PostgreSQL
  - Currently, it is the most dominant model for developing database applications.
- Object-oriented Data Models
- Object-Relational Models