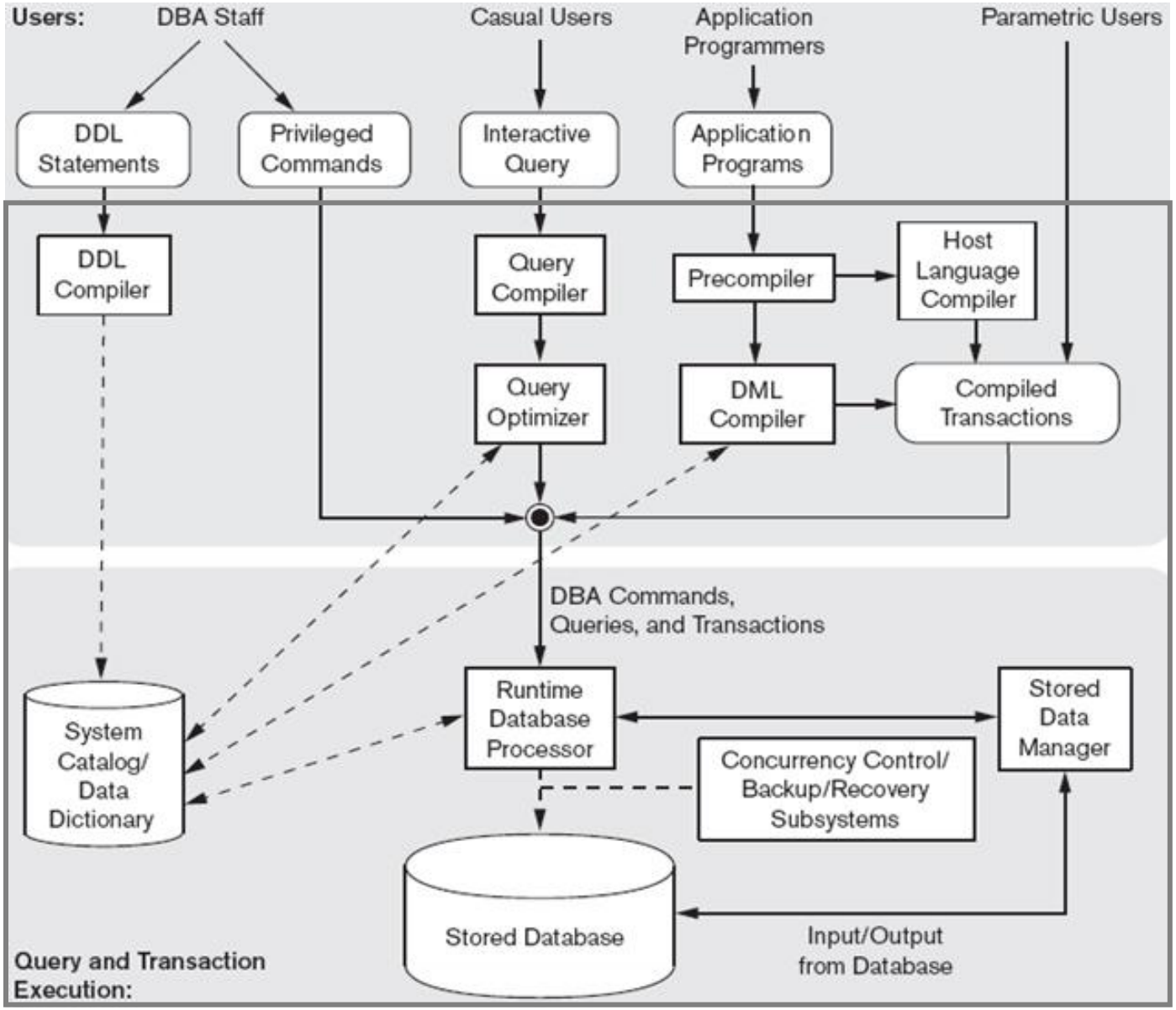
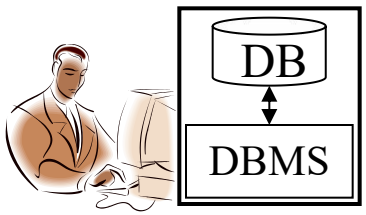
Chapter 2 Database System Concepts and Architecture



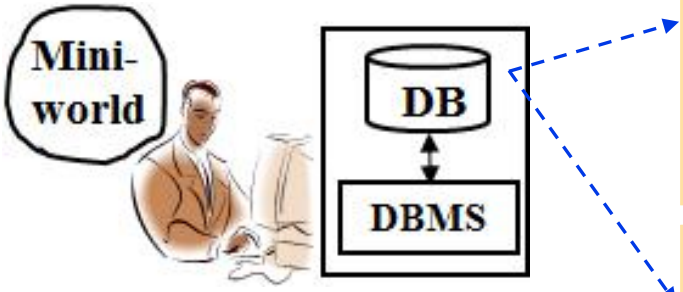


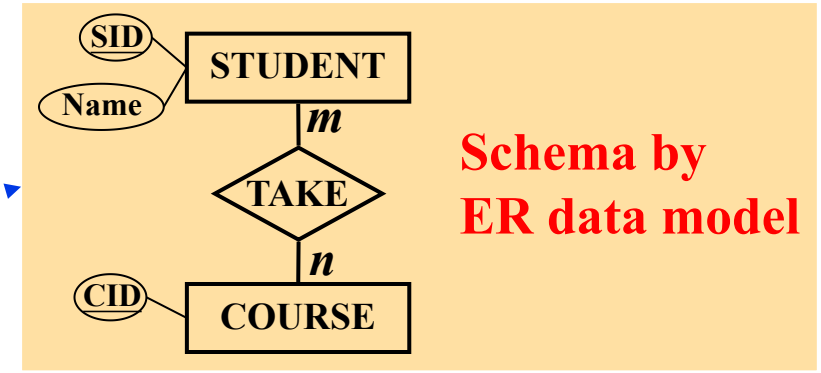
Database System Concepts and Architecture

- Data models, schemas, and instances
- Three-schema architecture and data independence
- Database languages and interfaces
- The database system environment
- Centralized and client/server architectures for DBMSs
- Classification of database management systems

Data Models

- **Data Model**
  - A set of concepts to describe the *structure* of a database, and certain *constraints* that the database should obey.
    - ER data model: {entity, relationship, attribute, key, …}
    - Relational data model: {relation, tuple, attribute, primary key, …}





Schema by Relational data model

STUDENT

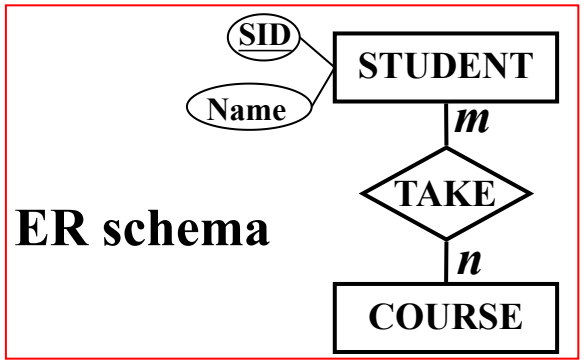
Name	StudentNumber	Class	Major

COURSE

CourseName	CourseNumber	CreditHours	Department

Categories of data models

- **Conceptual (high-level, semantic)** data models:
  - Provide concepts that are close to the way many users *perceive* data, e.g. ER model.
  - 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.
- **Implementation (representational)** data models:
  - Provide concepts that fall between the above two, balancing user views with some computer storage details, e.g. relational model.



Relational DB schema

STUDENT

Name	StudentNumber	Class	Major

COURSE

CourseName	CourseNumber	CreditHours	Department

History of Data Models

- Network Model:
  - the first one to be implemented by Honeywell in 1964-65 (IDS System). Adopted heavily due to the support by CODASYL (CODASYL - DBTG report of 1971).
  - Later implemented in a large variety of systems - IDMS (Cullinet - now CA), DMS 1100 (Unisys), IMAGE (H.P.), VAX -DBMS (Digital Equipment Corp.).
- Hierarchical Data Model:
  - implemented in a joint effort by IBM and North American Rockwell around 1965. Resulted in the IMS family of systems. The most popular model. Other system based on this model: System 2k (SAS inc.)
- Relational Model:
  - proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82. Now in several commercial products (SQL Server, ORACLE, DB2, INFORMIX, SYBASE).

History of Data Models

- Object-oriented Data Model(s):
  - several models have been proposed for implementing in a database system.
  - One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE).
  - Additionally, systems like O_2, ORION (at MCC - then ITASCA), IRIS (at H.P.- used in Open OODB).
- Object-Relational Models:
  - Most Recent Trend. Started with Informix Universal Server. Exemplified in the latest versions of SQL Server, Oracle-10i, and DB2 etc. systems.

Schemas versus Instances

- **Database Schema**:
  - ✓ The *description* of a database. Includes descriptions of the **database structure** and the **constraints** that should hold on the database.
- **Database Instance:**
  - ✓ The actual data stored in a database at a *particular moment in time*. Also called **database state** (or **occurrence**).

STUDENT

Name	StudentNumber	Class	Major

COURSE

CourseName	CourseNumber	CreditHours	Department

PREREQUISITE

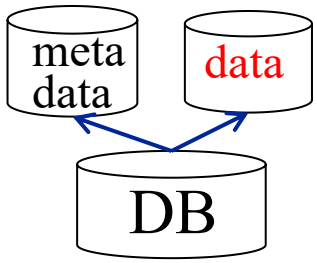
CourseNumber	PrerequisiteNumber

SECTION

SectionIdentifier	CourseNumber	Semester	Year	Instructor

GRADE_REPORT

StudentNumber	SectionIdentifier	Grade



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

STUDENT

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

PREREQUISITE

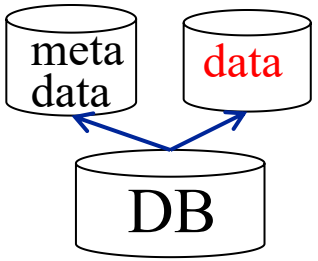
Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

. . .

. . .

Database Schema Vs. Database State

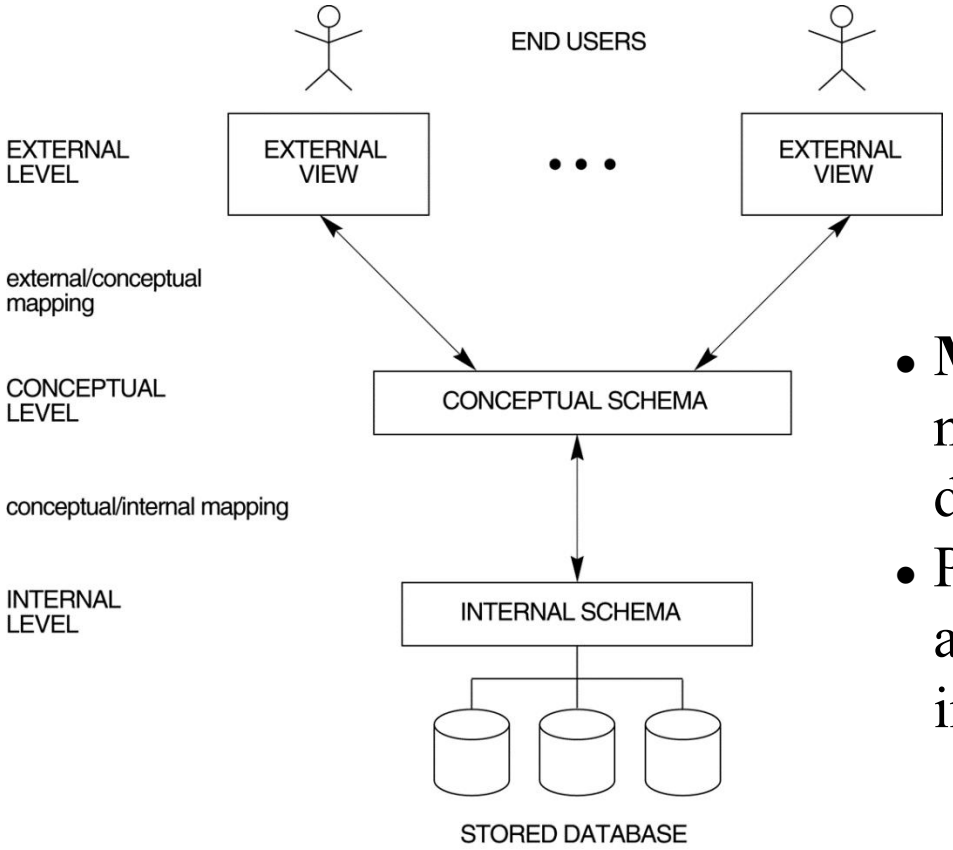
- **Database State:**
  - ✓ Refers to the content of a database at a moment in time.
- **Initial Database State:**
  - ✓ Refers to the database when it is loaded
- **Valid State:**
  - ✓ A state that satisfies the structure and constraints of the database; e.g., constraint: age≦ 120, CreditHours ≦ 4
- **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 called **extension**.



COURSE	CourseName	CourseNumber	CreditHours	Department
	Intro to Computer Science	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Mathematics	MATH2410	3	MATH
	Database	CS3380	3	CS

Three-Schema Architecture

- Proposed to support DBMS characteristics of:
  - ✓ Support of **multiple views** of the data.
  - ✓ **Program-data independence**.



- **Mappings** among schema levels are needed to transform requests and data.
- Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.

Three-Schema Architecture

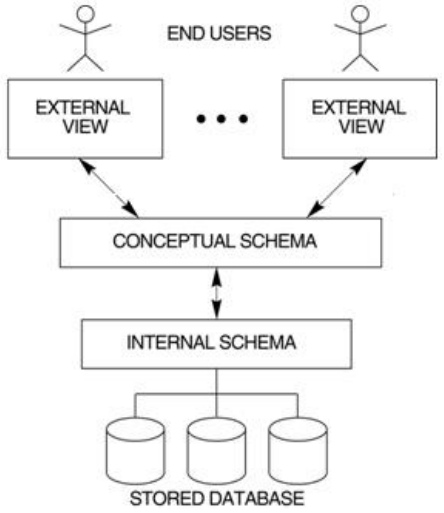
- Defines DBMS schemas at *three levels*:
  - ✓ **External schemas**

    at the external level to describe the various user views. Usually uses the same data model as the conceptual level.
  - ✓ **Conceptual schema**

    at the conceptual level to describe the structure and constraints for the *whole* database for a community of users. Uses a *conceptual* or an *implementation* data model.
  - ✓ **Internal schema**

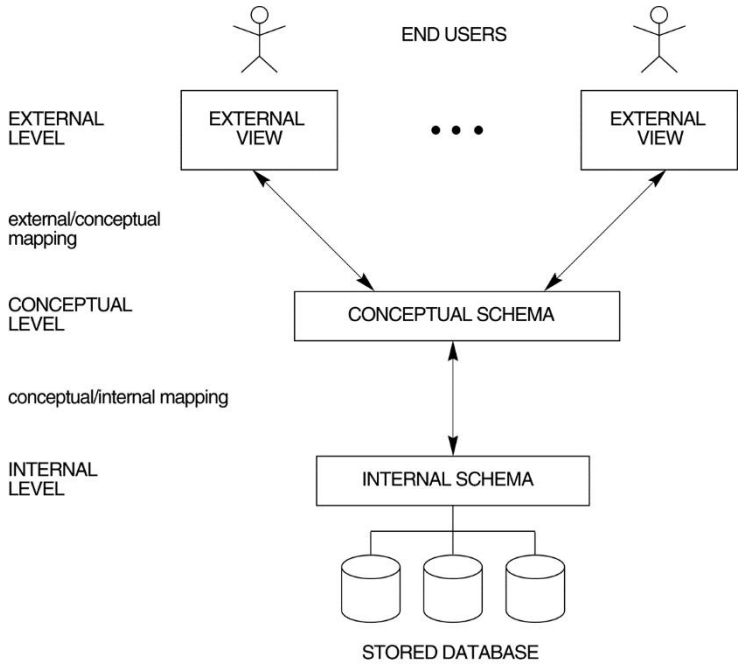
    at the internal level to describe physical storage structures and access paths. Typically uses a *physical* data model.

DBA defines DB schema



Data Independence

- **Logical Data Independence**:
  - ✓ The capacity to change the conceptual schema without having to change the external schemas and their application programs.
- **Physical Data Independence**:
  - ✓ The capacity to change the internal schema without having to change the conceptual schema.

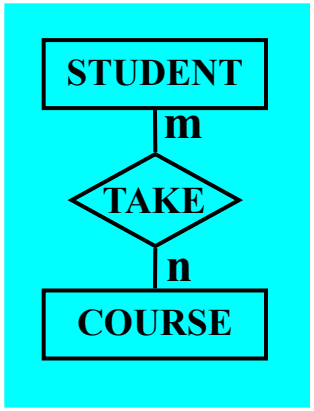


- When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence. The higher-level schemas themselves are *unchanged*.

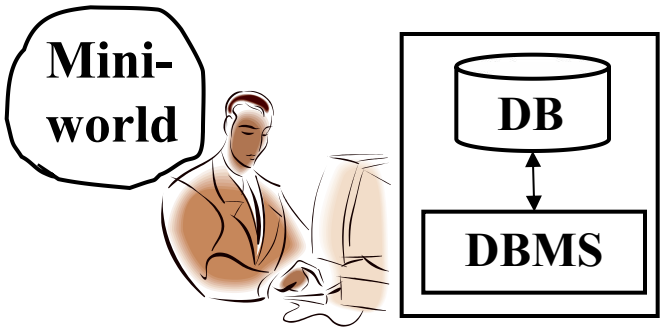
DBMS Languages

- **DDL and DML**
- **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).

Design DB by designer
(using conceptual data model)



Define DB

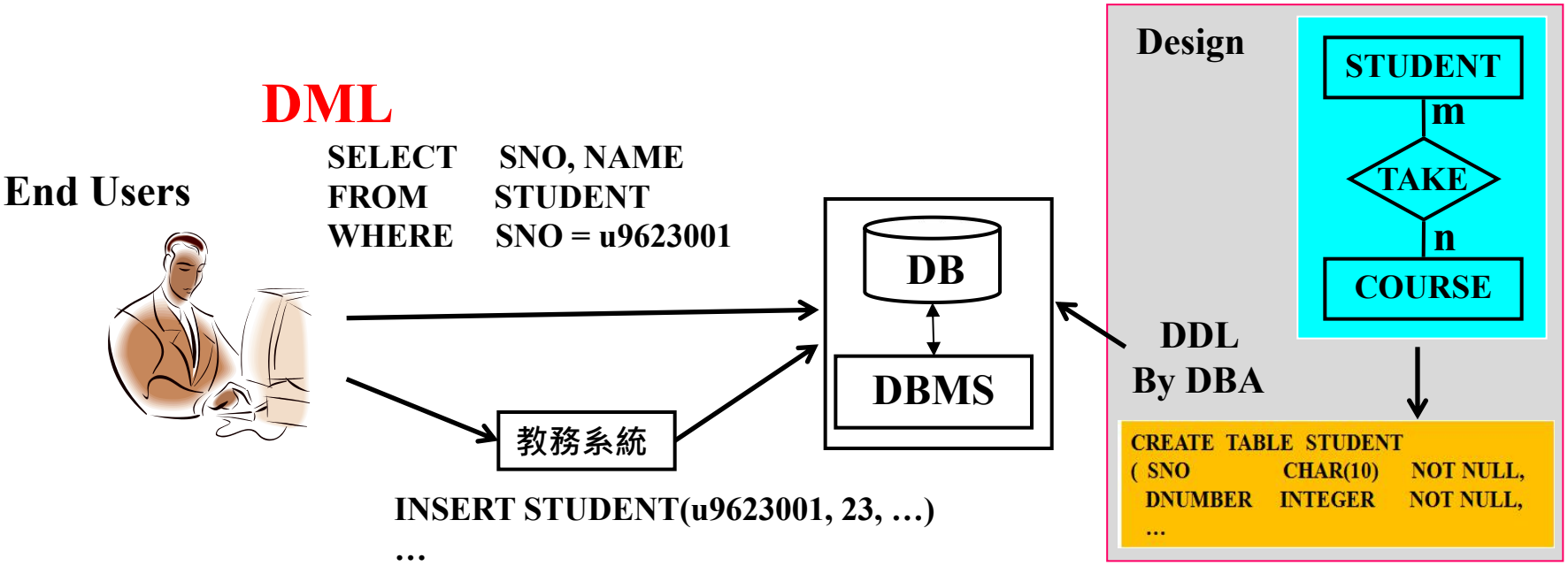


DDL

```
CREATE  TABLE  STUDENT
( SNO            CHAR(10)    NOT NULL,
  DNUMBER        INTEGER     NOT NULL,
  …
```

DBMS Languages

- **Data Manipulation Language (DML):**
  - Used to specify database retrievals and updates.
  - ***Stand-alone*** DML commands can be applied directly (**query language**).
  - DML commands (**data sublanguage**) can be ***embedded*** in a general-purpose programming language (**host language**), such as Java, C or Delphi.



DBMS Languages

- **High Level** or **Non-procedural Languages:**
  - Also called *declarative* languages.
  - e.g., SQL, are *set-oriented* and specify *what* data to retrieve than how to retrieve.
- **Low Level** or **Procedural Languages:**
  - *record-at-a-time*; they specify *how* to retrieve data and include constructs such as looping.

```
SELECT    SNO, NAME
FROM      STUDENT
WHERE     SEX = M
```

```
for (i = 0, TotalStuNum, i++)
    if (Student[i].Sex == M)
        print(Student[i].Sno, Student[i].Name)
```

FIGURE 9.14 Program segment JDBC2, a JAVA program segment that uses JDBC for a query with a collection of tuples in its result.

```
//Program Segment JDBC2:
0)    import java.io.* ;
1)    import java.sql.*
      …
2)    class printDepartmentEmps {
3)       public static void main (String args []) throws SQLException, IOException {
4)       try { Class.forName("oracle.jdbc.driver.OracleDriver")
5)       } catch (ClassNotFoundException x) {
6)          System.out.println ("Driver could not be loaded") ;
7)          }
8)       String dbacct, passwrd, lname ;
9)       Double salary ;
10)      Integer dno ;
11)      dbacct = readentry("Enter database account:") ;
12)      passwrd = readentry("Enter pasword:") ;
13)      Connection conn = DriverManager.getConnection
14)         ("jdbc:oracle:oci8:" + dbacct + "/" + passwrd) ;
15)      dno = readentry("Enter a Department Number: ") ;
16)      String q = "select LNAME, SALARY from EMPLOYEE where DNO = " +
         dno.tostring() ;
17)      Statement s = conn.createStatement() ;
18)      ResultSet r = s.executeQuery(q) ;
19)      while (r.next()) {
20)         lname = r.getString(1) ;
21)         salary = r.getDouble(2) ;
22)         system.out.printline(lname + salary) ;
23)      } }
24)   }
```

// Procedural language: JAVA

// embedded SQL

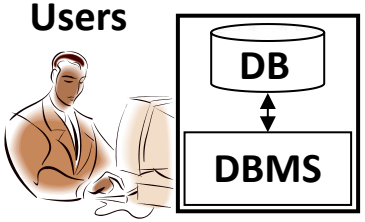
set-at-a-time

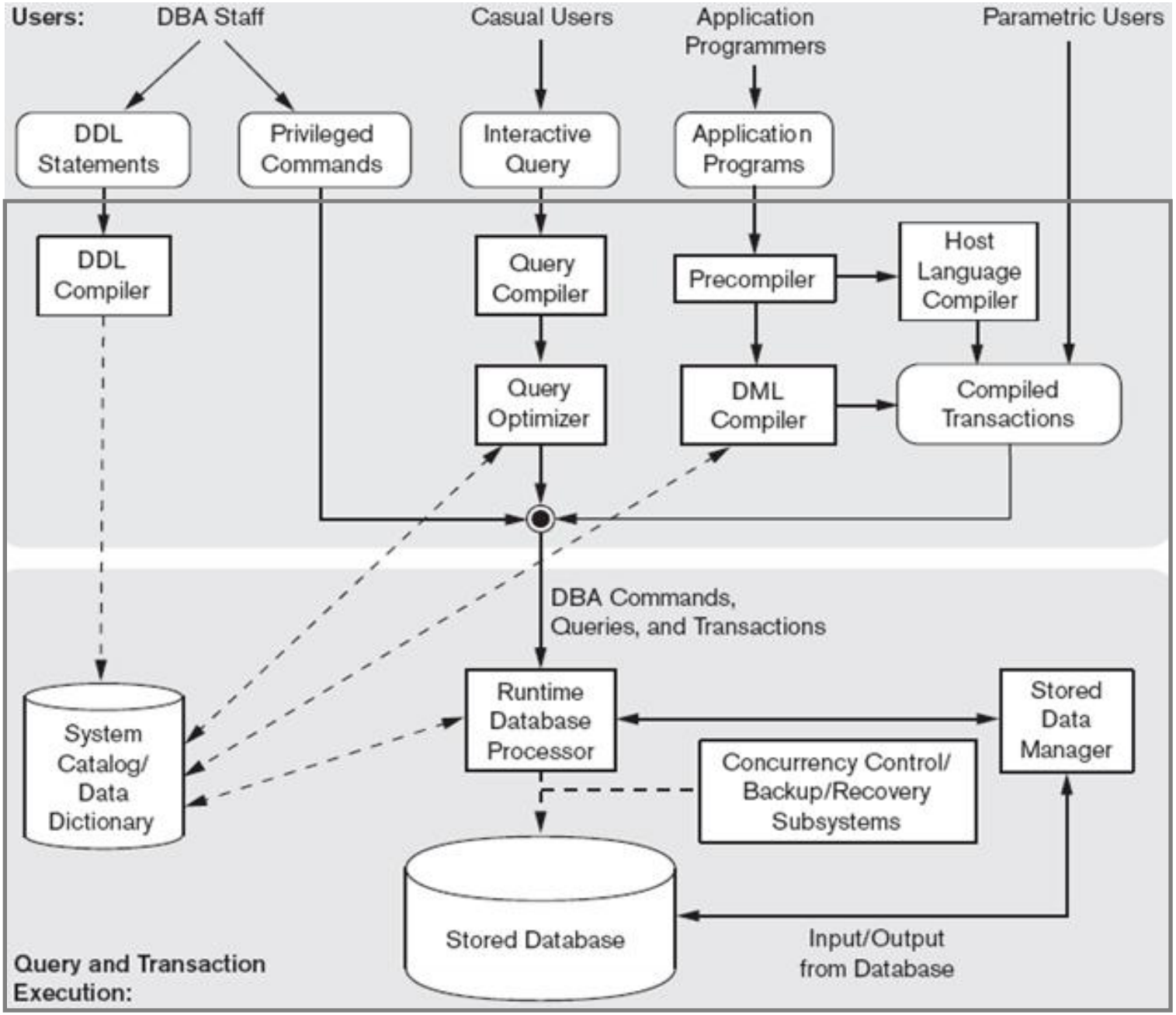
record-at-a-time

DBMS Interfaces and Other Utilities

Users

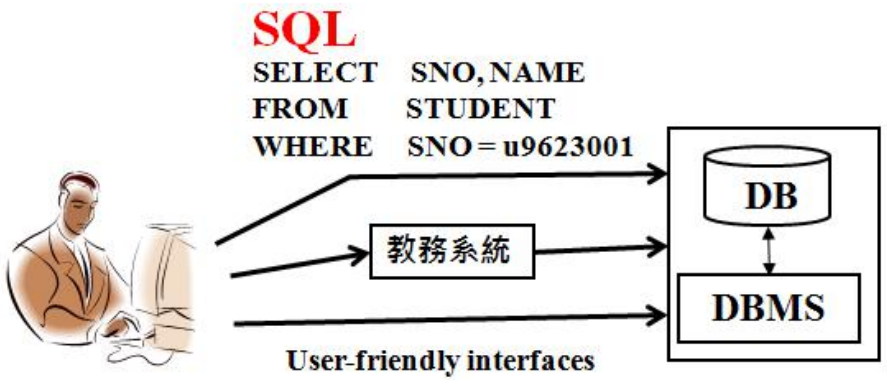
DBMS
+
DB





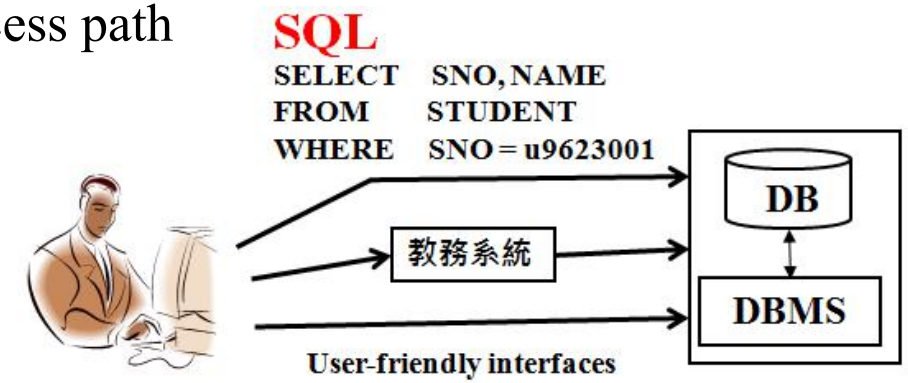
DBMS Interfaces

- Stand-alone query language interfaces (e.g., SQL).
- Programmer interfaces for embedding DML in programming languages:
  - Pre-compiler Approach
  - Procedure (Subroutine) Call Approach
- User-friendly interfaces:
  - Menu-based, popular for browsing on the web
  - Forms-based, designed for naïve users
  - Graphics-based (Point and Click, Drag and Drop etc.)
  - Natural language: requests in written English
  - Combinations of the above



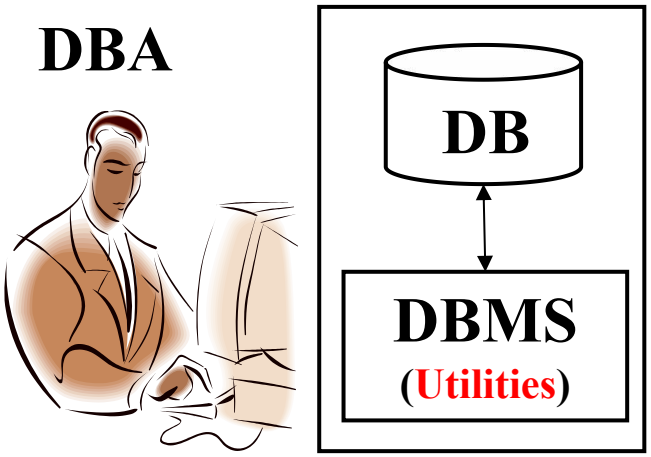
Other DBMS Interfaces

- Speech as Input and Output
- Web Browser as an interface
- Parametric interfaces (e.g., bank tellers) using function keys.
- Interfaces for the DBA:
  - ✓ Creating accounts, granting authorizations
  - ✓ Setting system parameters
  - ✓ Changing schemas or access path

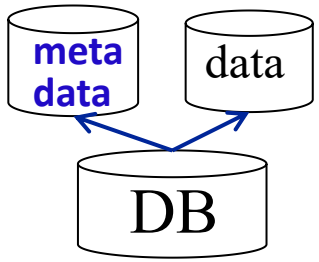


Database System Utilities

- To perform certain functions such as:
  - ✓ ***Loading*** data stored in files into a database. Includes data conversion tools.
  - ✓ ***Backing up*** the database periodically on tape.
  - ✓ ***Reorganizing*** database file structures.
  - ✓ ***Report generation*** utilities.
  - ✓ ***Performance monitoring*** utilities.
  - ✓ Other functions, such as ***sorting***, ***user monitoring***, ***data compression***, etc.



Other Tools

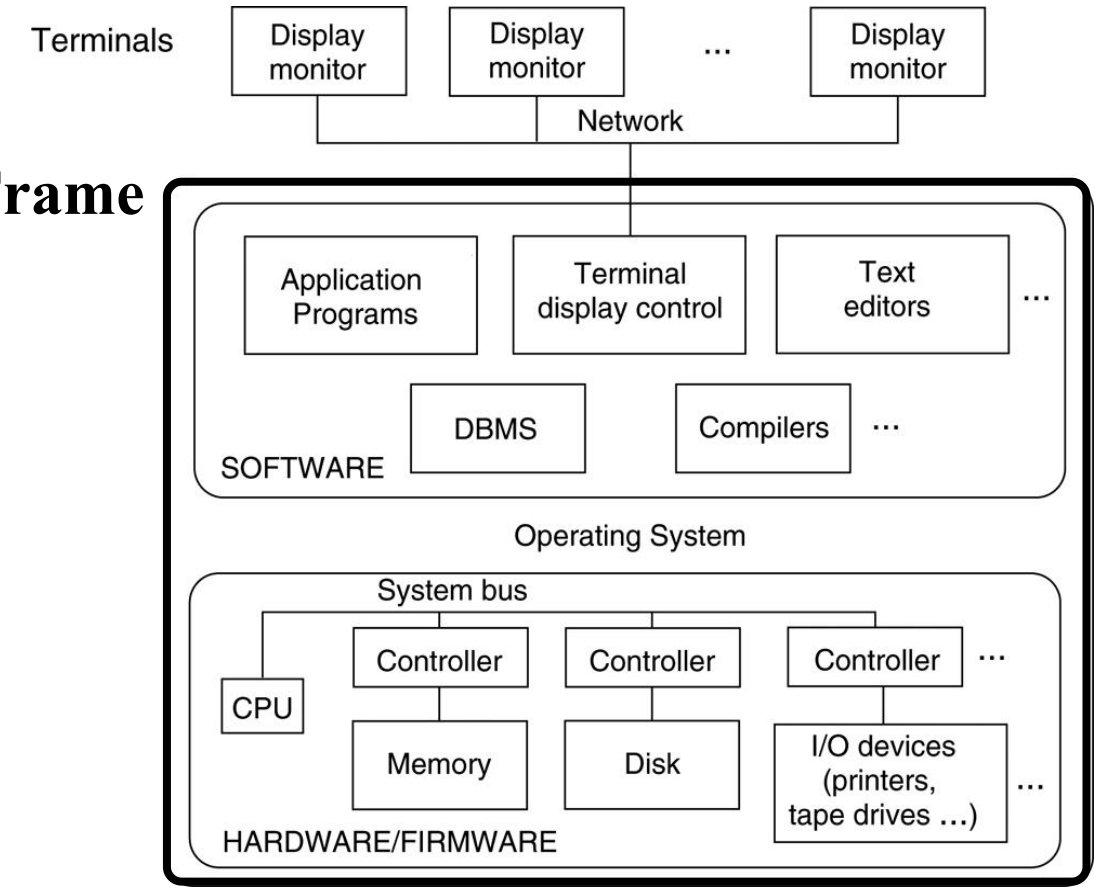


- **Data dictionary / repository**:
  - ✓ Used to store schema descriptions and constraints
- **Information repository**
  - ✓ Extended data dictionary
  - ✓ Including other information such as *design decisions*, *application program descriptions*, *user information*, *usage standards*, etc.
- **Application Development Environments and CASE (computer-aided software engineering) tools:**
  - ✓ Examples – Power builder (Sybase), Builder (Borland)

Centralized and Client-Server Architectures

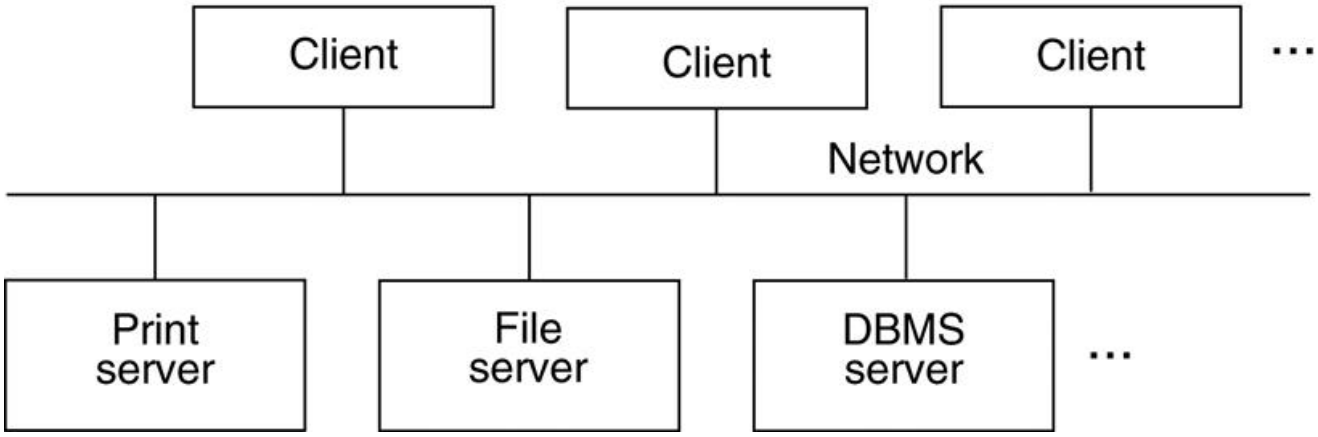
- **Centralized DBMS:**
  - combines everything into single system including- DBMS software, hardware, application programs and user interface processing software.

Main Frame



Basic Client-Server Architectures

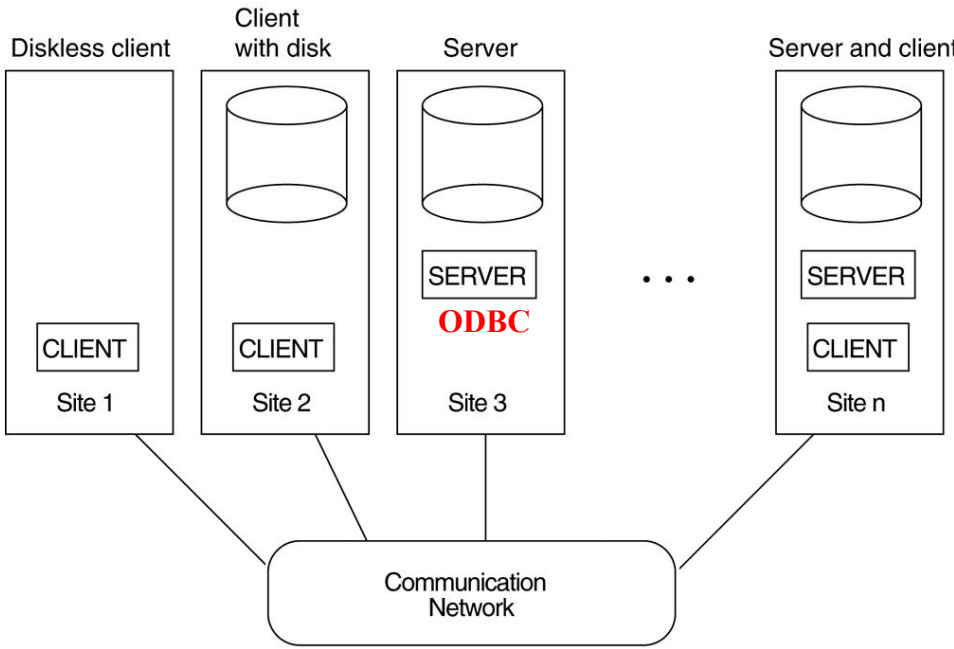
- Specialized Servers with Specialized functions
  - Printer, file, email, Web, DBMS servers
- **DBMS Servers**
  - Provides **database query** and **transaction services** to the clients
  - Sometimes called query and transaction servers



Two Tier Client-Server Architecture

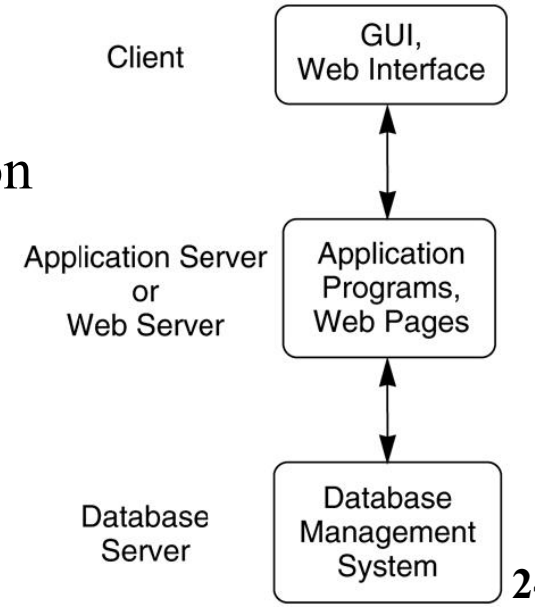
- **User Interface Programs and Application Programs** run on the client side
- Interface called **ODBC (Open Database Connectivity)**
  - ✓ provides an **Application Program Interface (API)** allow client side programs to call the DBMS. Most DBMS vendors provide ODBC drivers.
- A client program may connect to several DBMSs.
- Other variations of clients are possible:

  e.g., in some DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc. In such situations the server may be called the **Data Server**.



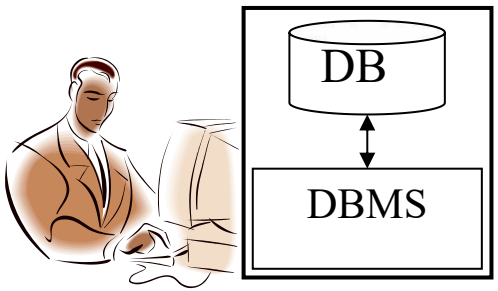
Three Tier Client-Server Architecture

- Common for Web applications
- Intermediate Layer called Application Server or Web Server:
  - stores the web connectivity software and the rules and business logic (constraints) part of the application used to access the right amount of data from the database server
  - acts like a conduit for sending partially processed data between the database server and the client.
- Additional Features- Security:
  - encrypt the data at the server before transmission
  - decrypt data at the client



Classification of DBMSs

- Based on the data model used:
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented, Object-relational.
- Other classifications:
  - Single-user (typically used with micro- computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)



Variations of Distributed Environments

- Homogeneous DDBMS
- Heterogeneous DDBMS
- Federated (or Multidatabase) Systems
  - Loosely coupled and have a degree of local autonomy