NOTES - PART I

Databases and Database Management Systems

(Based on Chapters 1-2 in

***Fundamentals of Databse Systems* by Elmasri and Navathe, Ed. 3)**

1. 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.

2. Example of a Database (with a Conceptual Data Model)

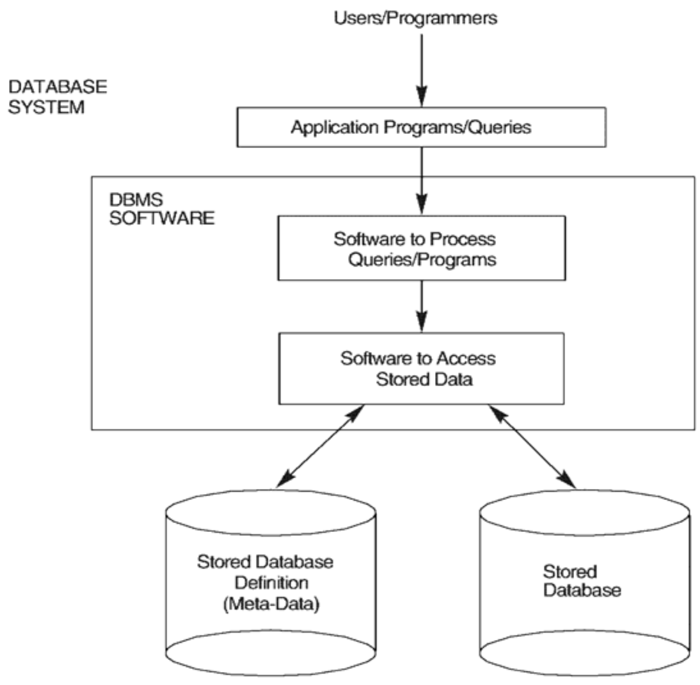
Mini-world for the example: Part of a UNIVERSITY environment.

Some mini-world *entities*:
- STUDENTs
- COURSEs
- SECTIONs (of COURSEs)
- (academic) DEPARTMENTs
- INSTRUCTORs

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

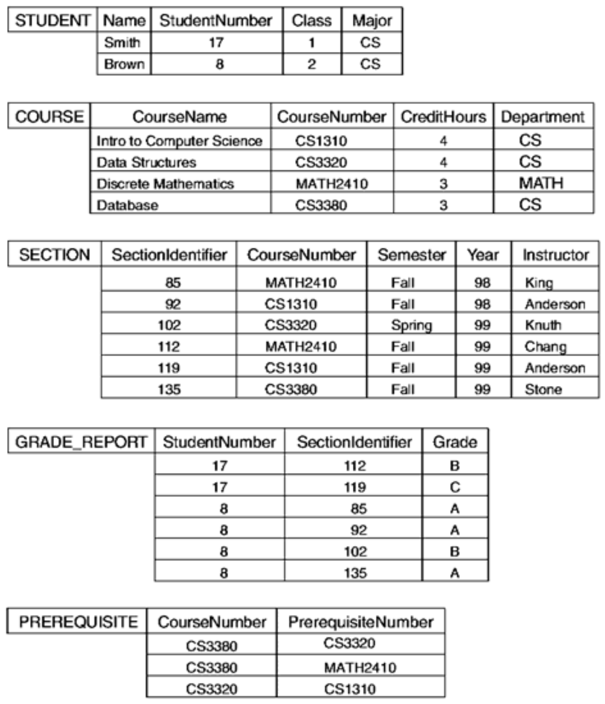
NOTE: The above could be expressed in the *ENTITY-RELATIONSHIP* data model.

Figure 1.1 A simplified database system environment, illustrating the concepts and terminology discussed in Section 1.1.



Users/Programmers
DATABASE SYSTEM
Application Programs/Queries
DBMS SOFTWARE
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

Figure 1.2 An example of a database that stores student records and their grades.



STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

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

SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	92	CS1310	Fall	98	Anderson
	102	CS3320	Spring	99	Knuth
	112	MATH2410	Fall	99	Chang
	119	CS1310	Fall	99	Anderson
	135	CS3380	Fall	99	Stone

GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A
	8	102	B
	8	135	A

PREREQUISITE	CourseNumber	PrerequisiteNumber
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

3. Main Characteristics of Database Technology

- Self-contained nature of a database system: A DBMS **catalog** stores the *description* of the database. The description is called **meta-data**). This allows the DBMS software to work with different databases.

- Insulation between programs and data: Called **program-data independence**. Allows changing data storage structures and operations without having to change the DBMS access programs.

- Data Abstraction: A **data model** is used to hide storage details and present the users with a *conceptual view* of the database.

- 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.

Figure 1.4 Two views derived from the example database shown in Figure 1.2. (a) The student transcript view. (b) The course prerequisite view.

(a)

TRANSCRIPT	StudentName	Student Transcript				
		CourseNumber	Grade	Semester	Year	SectionId
	Smith	CS1310	C	Fall	99	119
		MATH2410	B	Fall	99	112
	Brown	MATH2410	A	Fall	98	85
		CS1310	A	Fall	98	92
		CS3320	B	Spring	99	102
		CS3380	A	Fall	99	135

(b)

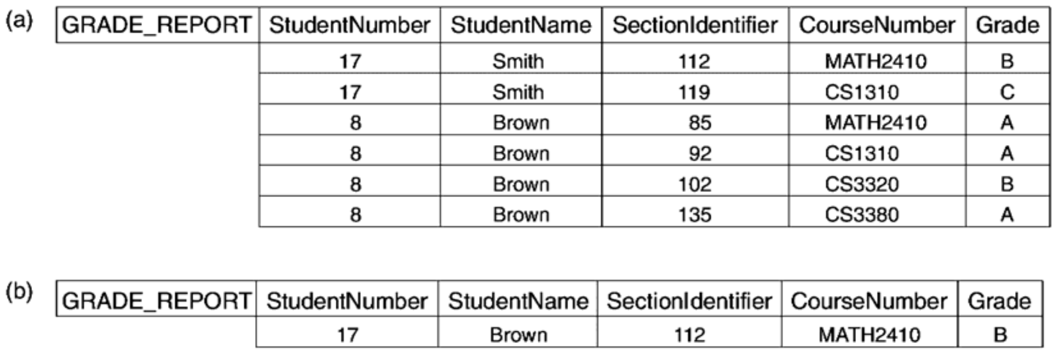
PREREQUISITES	CourseName	CourseNumber	Prerequisites
	Database	CS3380	CS3320
			MATH2410
	Data Structures	CS3320	CS1310

4. Additional Benefits of Database Technology

- Controlling redundancy in data storage and in development and maintenence efforts.
- Sharing of data among multiple users.
- Restricting unauthorized access to data.
- Providing multiple interfaces to different classes of users.
- Representing complex relationships among data.
- Enforcing integrity constraints on the database.
- Providing backup and recovery services.
- Potential for enforcing standards.
- Flexibility to change data structures.
- Reduced application development time.
- Availability of up-to-date information.

! Economies of scale.

Figure 1.5 The redundant storage of data items. (a) *Controlled redundancy:* Including StudentName and CourseNumber in the grade_report file. (b) *Uncontrolled redundancy:* A GRADE_REPORT record that is inconsistent with the STUDENT records in Figure 1.2, because the Name of student number 17 is Smith, not Brown.



(a)

GRADE_REPORT	StudentNumber	StudentName	SectionIdentifier	CourseNumber	Grade
	17	Smith	112	MATH2410	B
	17	Smith	119	CS1310	C
	8	Brown	85	MATH2410	A
	8	Brown	92	CS1310	A
	8	Brown	102	CS3320	B
	8	Brown	135	CS3380	A

(b)

GRADE_REPORT	StudentNumber	StudentName	SectionIdentifier	CourseNumber	Grade
	17	Brown	112	MATH2410	B

5 When <u>not</u> to use a DBMS

Main inhibitors (costs) of using a DBMS:
- High initial investment and possible need for additional hardware.
- Overhead for providing generality, security, recovery, integrity, and concurrency control.

When a DBMS may be unnecessary:
- If the database and applications are simple, well defined, and not expected to change.
- If there are stringent real-time requirements that may not be met because of DBMS overhead.
- If access to data by multiple users is not required.

When no DBMS may suffice:

- If the database system is not able to handle the complexity of data because of modeling limitations

- If the database users need special operations not supported by the DBMS.

6. **Data Models**

Data Model: A set of concepts to describe the *structure* of a database, and certain *constraints* that the database should obey.

Data Model Operations: Operations for specifying database retrievals and updates by referring to the concepts of the data model.

<u>Categories of data models</u>:

- **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.

- **Implementation** (**record-oriented**) data models: Provide concepts that fall between the above two, balancing user views with some computer storage details.

6A. HISTORY OF DATA MODELS

• Relational Model: proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82. Now in several commercial products (ORACLE, SYBASE, INFORMIX, CA-INGRES).

• 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).

• 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.)

• 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. Exemplified in ILLUSTRA and UNiSQL systems.

Figure 2.1 Schema diagram for the database of Figure 1.2.

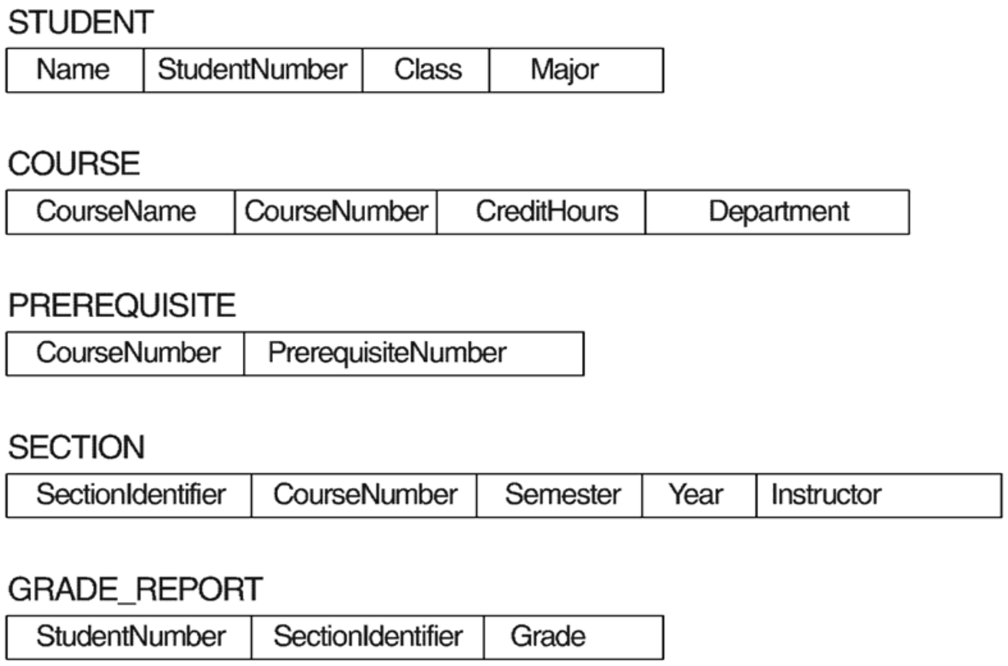
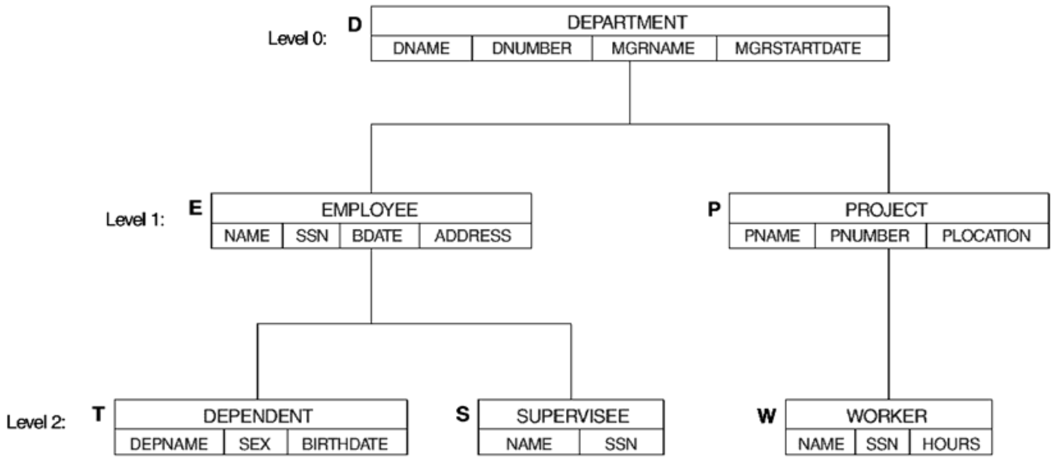


Figure D.4 A hierarchical schema for part of the COMPANY database.



6B. HIERARCHICAL MODEL

ADVANTAGES:

- Hierarchical Model is simple to construct and operate on
- Corresponds to a number of natural hierarchically organized domains - e.g., assemblies in manufacturing, personnel organization in companies
- Language is simple; uses constructs like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT etc.

DISADVANTAGES:

- Navigational and procedural nature of processing
- Database is visualized as a linear arrangement of records
  - Little scope for "query optimization"

Figure 2.4 The schema of Figure 2.1 in the notation of the network data model.

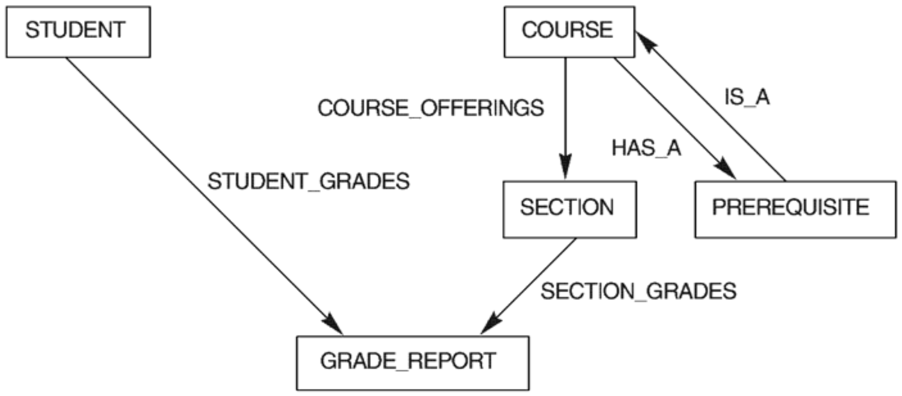
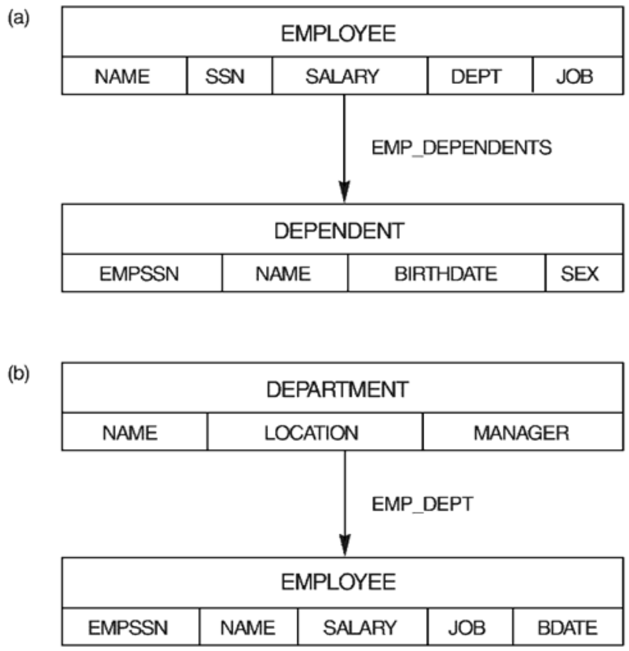


Figure C.7 Different set options. (a) An AUTOMATIC FIXED set. (b) An AUTOMATIC MANDATORY set.



6C. NETWORK MODEL

ADVANTAGES:

• Network Model is able to model complex relationships and represents semantics of add/delete on the relationships.

• Can handle most situations for modeling using record types and relationship types.

• Language is navigational; uses constructs like FIND, FIND member, FIND owner, FIND NEXT within set, GET etc. Programmers can do optimal navigation through the database.

DISADVANTAGES:
• Navigational and procedural nature of processing
• Database contains a complex array of pointers that thread through a set of records.
• Little scope for automated "query optimization"

7. 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.

Schema Diagram: A diagrammatic display of (some aspects of) a database schema.

Database Instance: The actual data stored in a database at a *particular moment in time* . Also called **database state** (or **occurrence**).

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**.

8. Three-Schema Architecture

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

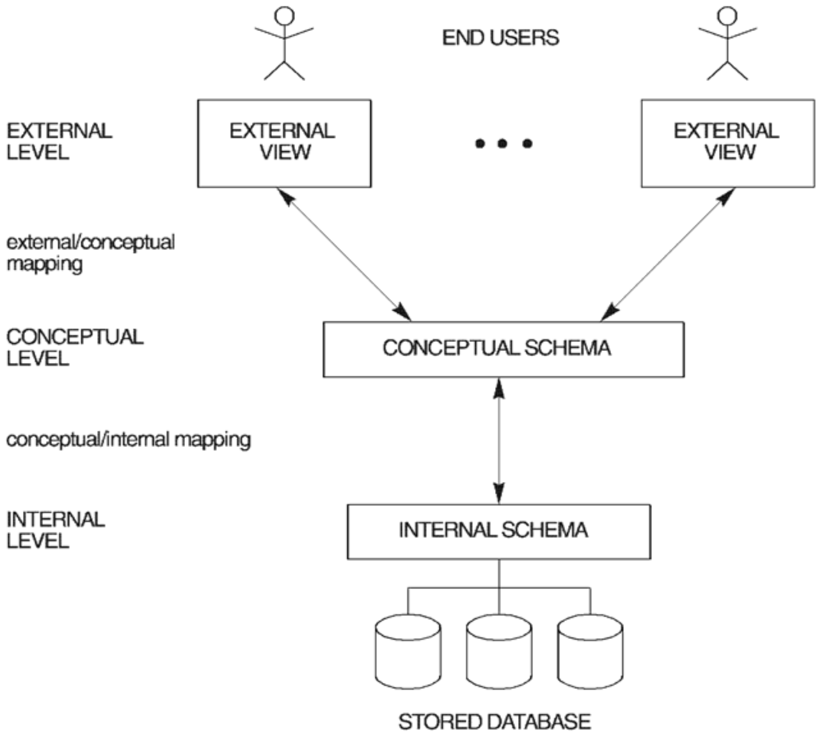
Defines DBMS schemas at *three levels* :
- **Internal schema** at the internal level to describe data storage structures and access paths. Typically uses a *physical* data model.

- **Conceptual schema** at the conceptual level to describe the structure and constraints for the *whole* database. Uses a *conceptual* or an *implementation* data model.

- **External schemas** at the external level to describe the various user views. Usually uses the same data model as the conceptual level.

Mappings among schema levels are also needed. Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.

Figure 2.2 Illustrating the three-schema architecture.



9 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*. Hence, the application programs need not be changed since the refer to the external schemas.

10. 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.

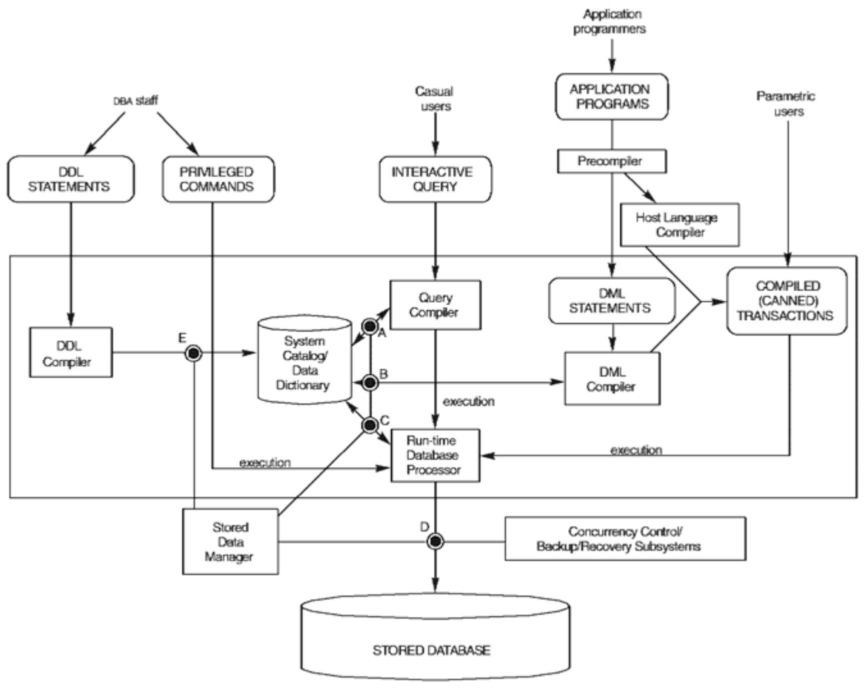
Data Manipulation Language (**DML**): Used to specify database retrievals and updates.

- DML commands (**data sublanguage**) can be *embedded* in a general-purpose programming language (**host language**), such as COBOL, PL/1 or PASCAL.
- Alternatively, *stand-alone* DML commands can be applied directly (**query language**).

11. DBMS Interfaces

- Stand-alone query language interfaces.
- Programmer interfaces for embedding DML in programming languages:
  - Pre-compiler Approach
  - Procedure (Subroutine) Call Approach
- User-friendly interfaces:
  - Menu-based
  - Graphics-based (Point and Click, Drag and Drop etc.)
  - Forms-based
  - Natural language
  - Combinations of the above
  - Speech as Input (?) and Output
  - Web Browser as an interface
- Parametric interfaces using function keys.
- Report generation languages.
- Interfaces for the DBA:
  - Creating accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access path

Figure 2.3 Typical component modules of a DBMS. Dotted lines show accesses that are under the control of the stored data manager.



13. Database System Utilities

To perform certain functions such as:

- *Loading* data stored in files into a database.
- *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.

Data dictionary / repository:

- Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.
- *Active* data dictionary is accessed by DBMS software and users/DBA.
- *Passive* data dictionary is accessed by users/DBA only.

14. 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)

Distributed Database Systems have now come to be known as <u>client server based database systems</u> because they do not support a totally distributed environment, but rather a set of database servers supporting a set of clients.