Module - 1

Introduction to Databases

1.1 Basic Definitions:

Databases and database technology are having a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, engineering, medicine, law, education, and library science, to name a few. The word database is in such common use that we must begin by defining a database. Our initial definition is quite general.

- **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.

A database has the following implicit properties:

• A database represents some aspect of the real world, sometimes called the miniworld or the Universe of Discourse (UoD). Changes to the miniworld are reflected in the database.

• A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.

• A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested.

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is hence a general-purpose software system that facilitates the processes of defining, constructing, and manipulating databases for various applications.

Defining a database involves specifying the data types, structures, and constraints for the data to be stored in the database. The database definition or descriptive information is also stored in the database in the form of a database catalog or dictionary; it is called **Meta-data**.

Constructing the database is the process of storing the data itself on some storage medium that is controlled by the DBMS.

Manipulating a database includes such functions 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.

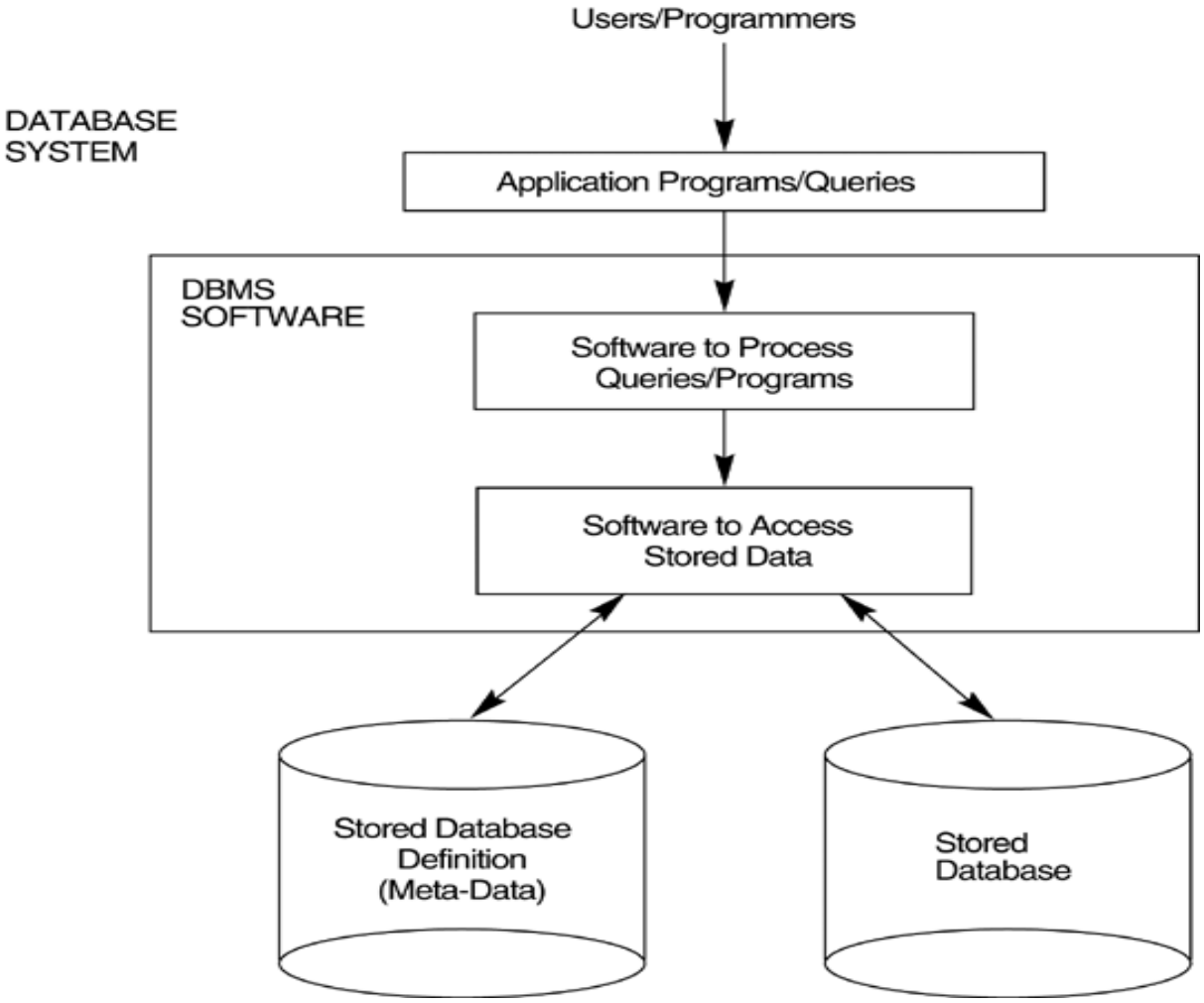
A **query** typically causes some data to be retrieved; a **transaction** may cause some data to be read and some data to be written into the database. Protection includes system protection against hardware or software function and security protection against unauthorized or malicious access. **Maintain** the database system by allowing the system to evolve as requirements change over time.

Properties of Databases

- Represent some aspect of the real world

- Data is logically coherent and has some meaning
- Designed, built and populated for a specific reason
- Built with a target group of users and particular applications in mind

FIGURE1.1
A simplified database system environment.



Data Model:

A set of concepts to describe the structure of a database, the operations for manipulating these structures, and certain constraints that the database should obey.

Data Model Structure and Constraints:

- Constructs are used to define the database structure(datatypes, relationships, constraints)
- Constructs typically include elements (and their data types) as well as groups of elements (e.g. entity, record, table), and relationships among such groups

- Constraints specify some restrictions on valid data; these constraints must be enforced at all times
- **Data Model Operations:**
    - These operations are used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.
    - Operations on the data model may include *basic model operations* (e.g. generic insert, delete, update) and *user-defined operations* (e.g. compute_student_gpa, update_inventory)

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. These are usually specified in an ad-hoc manner through DBMS design and administration manuals
- **Implementation (representational) data models:**
    - which provide concepts that may be easily understood by end users, it hide many details of data storage on disk but can be implemented on a computer system directly.

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

FIGURE1.2a
A database that stores student and course information.

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

FIGURE1.2b
A database that stores student and course information.

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

FIGURE1.2c
A database that stores student and course information.

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

FIGURE1.3
Internal storage format for a STUDENT record.

Data Item Name	Starting Position in Record	Length in Characters (bytes)
Name	1	30
StudentNumber	31	4
Class	35	4
Major	39	4

Characteristics of the Database Approach:

A number of characteristics distinguish the database approach from the traditional approach of programming with files. In traditional file processing, each user defines and implements the files needed for a specific application as part of programming the application.

In the database approach, a single repository of data is maintained that is defined once and then is accessed by various users. In file systems, each application is free to name data elements independently.

The main characteristics of the database approach versus the file-processing approach are the following.

- **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.

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

Figu
An examp
database catalog
database in Figu

- **Insulation between programs and data:**
  - Called **program-data independence:** The structure of data files is stored in the DBMS catalog separately from the access programs.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.
  - In some types of database systems, such as object-oriented and object-relational systems, users can define operations on data as part of the database definitions. **An operation** (also called a function or method) is specified in two parts. The interface (or signature) of an operation includes the operation name and the data types of its arguments (or parameters). The **implementation** (or method) of the operation is specified separately and can be changed without affecting the interface.
  - User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented. This may be termed **program-operation independence.**
- **Data Abstraction:**
  - It is the characteristic that allows program-data independence and program-operation independence.
  - A **data model** is used to hide storage details and present the users with a conceptual view of the database.

- 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.
  - **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 USERS:

- Users may be divided into
  - Those who actually use and control the database content, and those who design, develop and maintain database applications (called "**Actors on the Scene**"), and
  - Those who design and develop the DBMS software and related tools, and the computer systems operators (called "**Workers Behind the Scene**").

Database Users(Actors on the scene)

1. Database administrators:

- Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.

2. Database Designers:

- Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

3. Categories of End-users

- **End-users:** They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
- **Casual end users**: access database occasionally when needed

- **Naïve** or **Parametric end users**: they make up a large section of the end-user population. Their main job function revolves around constantly querying and updating the database. They use previously well-defined functions in the form of "canned transactions" against the database.
- Examples are **bank-tellers**(check account balances and post withdrawals and deposits.) or **reservation clerks**(airlines, hotels, car rental check availability for a given request and make reservations) who do this activity for an entire shift of operations.
  - **Sophisticated:**
    - These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
    - Many use tools in the form of software packages that work closely with the stored database.
  - **Stand-alone:**
    - Mostly maintain personal databases using ready-to-use packaged applications.
    - An example is a tax program user that creates its own internal database.
    - Another example is a user that maintains an address book

4. System Analyst and Application Programmers (Software Engineers) :

- **System Analysts** determine the requirements of end users, especially naïve and parametric end users and develop specifications for canned transactions that meet these requirements.
- **Application programmers** implement these specifications as programs, then they test, debug, document and maintain these canned transactions.

Workers behind the Scene:

These are the users, who design, use, and administer a database, others are associated with the design, development, and operation of the DBMS software and system environment. These persons are typically not interested in the database content itself.

1) DBMS system designers and implementers design and implement the DBMS modules and interfaces as a software package. A DBMS is a very complex software system that consists of many components, or modules, including modules for implementing the catalog, query language processing, interface processing, accessing and buffering data, controlling concurrency, and handling data recovery and security.

2) Tool developers design and implement tools—the software packages that facilitate database modeling and design, database system design, and improved performance. They include packages for database design, performance monitoring, natural language or graphical interfaces, prototyping, simulation, and test data generation.

3) Operators and maintenance personnel (system administration personnel) are responsible for the actual running and maintenance of the hardware and software environment for the database system.

Advantages of Using the Database Approach:

- **Controlling redundancy** in data storage and in development and maintenance efforts. In traditional software development utilizing processing, every user group maintains its own files for handling its data-processing applications.
- **Sharing of data among multiple users.**
- **Restricting unauthorized access to data**. When multiple users share a large database, it is likely that most users will not be authorized to access all information in the database. Foe example, financial data is often considered confidential, and only authorized persons are allowed to access such data.
- **Providing persistent storage for program Objects**: Database can be used to provide persistent storage for program for program object and data structures. This is one of the main reasons for object-oriented database systems. Programming languages typically have complex data structures, such as record types in Pascal or class definitions in C++ or Java.
- **Providing Storage Structures (e.g. indexes) for efficient Query Processing:** Database system must provide capabilities for efficiently executing queries and updates. Because the database is typically stored on disk, the DBMS must provide specialized data structures to speed up disk search for the desired records.
- **Providing backup and recovery services.** A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery.
- **Providing multiple user interfaces to different classes of users:** Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces. Like., Programming language interfaces, menu-driven interfaces, natural language interfaces and so on.,
- **Representing complex relationships among data:** Adatabase may include numerous varieties of data that are interrelated in many ways. A DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficiently.
- **Enforcing integrity constraints on the database:** Most database applications have certain integrity constrains that must hold for the data. A DBMS should provide capabilities for defining and enforcing these constraints. The simplest type of integrity constraint involves specifying a data type for each data.
- **Permitting Inferencing and Actions Using Rules and Triggers:** Drawing inferences and actions from the stored data using deductive and active database systems rules, triggers, stored procedures.

Additional Implications of Using the Database Approach

- **Potential for enforcing standards:**

- This is very crucial for the success of database applications in large organizations. **Standards** refer to data item names, display formats, screens, report structures, meta-data (description of data), Web page layouts, etc.

- **Reduced application development time:** Incremental time to add each new application is reduced.
- **Flexibility to change data structures:** Database structure may evolve as new requirements are defined.
- **Availability of current information:** Extremely important for on-line transaction systems such as airline, hotel, car reservations.
- **Economies of scale:** Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

When not to use a DBMS

- Main inhibitors (costs) of using a DBMS:
  - High initial investment and possible need for additional hardware, software and training.
  - The generality that a DBMS provides for defining and processing data
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions.
- 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.

Example of a database state

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

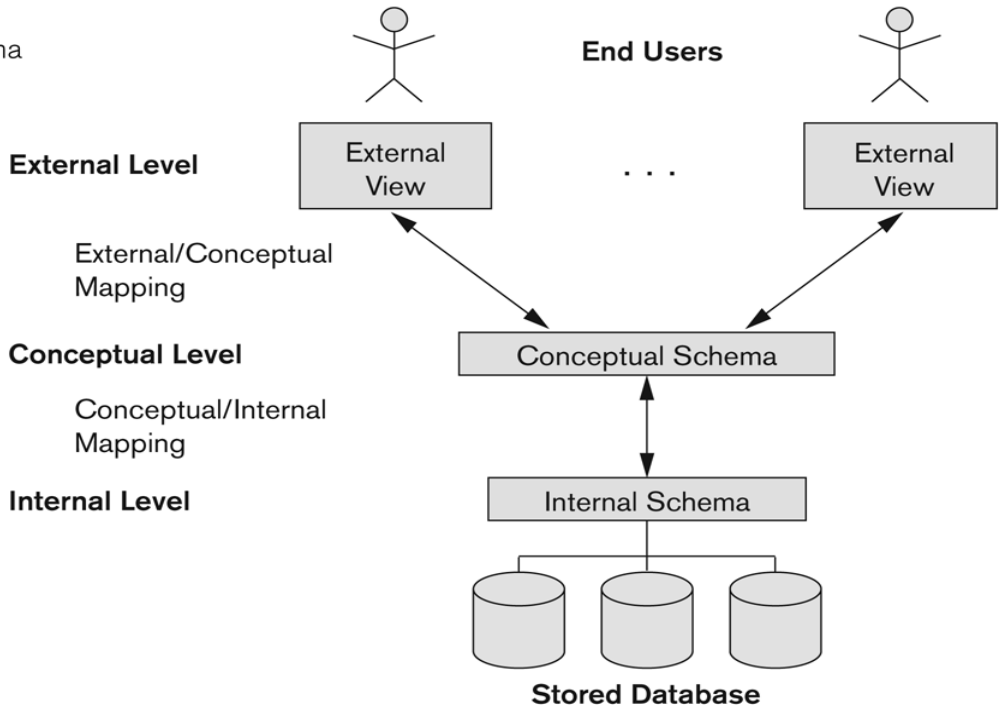
Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

Three-Schema Architecture:

- Proposed to support DBMS characteristics of:
  - Insulation of Program-data(program-data and program-operation independence).
  - Support of multiple views of the data.
  - Use of a catalog to store the database description(schema).
- we specify an architecture for database systems, called the Three-schema architecture, that was proposed to help achieve and visualize these characteristics.
- **Defines DBMS schemas at *three* levels:**
  - ✓ **Internal schema** at the internal level to describe physical storage structures ie describes the complete details of data storage and access paths for the database(e.g indexes).
  - ✓ **Conceptual schema** at the conceptual level to describe the structure of the whole database for a community of users.It hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
  - ✓ **The external or view level** includes a number of external schemas or user views. Each external schema 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.

Figure 2.2
The three-schema architecture.



- **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.
  - Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)

Data Independence

- **Logical Data Independence:**
  - The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.
- **Physical Data Independence:**
  - The capacity to change the internal schema without having to change the conceptual schema.
  - For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance

- 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 they refer to the external schemas

DBMS Languages:

- Data Definition Language (DDL):is used by the DBA and by database designers to define conceptual and internal schemas.
- Data Manipulation Language (DML)
  - ✓ High-Level or Non-procedural Languages: These include the relational language SQL
  - ✓ May be used in a standalone way or may be embedded in a programming language
  - ✓ Low Level or Procedural Languages:
  - ✓ These must be embedded in a programming language
- **Data Definition Language (DDL):**
  - ✓ Used by the DBA and by database designers to define conceptual and internal schemas. The DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog.
  - ✓ In some DBMSs, separate 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. SDL is typically realized via DBMS commands provided to the DBA and database designers
  - ✓ **view definition language (VDL)** are used to define internal and external schemas.
  - ✓ And 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.
  - ✓ In relational DBMSs, SQL is used in the role of VDL to define user or application views as results of predefined queries.
- **Data Manipulation Language (DML):**
  - ✓ Used to specify database retrievals, insertion, deletion and modifications of the data.
  - ✓ DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.
  - ✓ A library of functions can also be provided to access the DBMS from a programming language
  - ✓ Alternatively, stand-alone DML commands can be applied directly (called a *query language*).
- **Types of DML:**
  - High Level or Non-procedural Language:
    - For example, the SQL relational language
    - Are “set”-oriented and specify what data to retrieve rather than how to retrieve it.
    - Also called declarative languages.

- Low Level or Procedural Language:
  - Retrieve data one record-at-a-time;
  - Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

DBMS Interfaces:

- ✓ **Stand-alone query language interfaces**
- ✓ **Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE)**
- ✓ **Programmer interfaces for embedding DML in programming languages**
- ✓ **User-friendly interfaces**
- ✓ **Menu-based, forms-based, graphics-based, etc.**

■ User-friendly interfaces provided by a DBMS may include the following:

1. Menu-based Interfaces for Web Clients or Browsing. These interfaces present the user with lists of options (called menus) that lead the user through the formulation of a request. The query is composed step-by step by picking options from a menu that is displayed by the system. Pull-down menus are a very popular technique in Web-based user interfaces. They are also often used in browsing interfaces, which allow a user to look through the contents of a database in an exploratory and unstructured manner.

2. Apps for Mobile Devices. These interfaces present mobile users with access to their data. For example, banking, reservations, and insurance companies, among many others, provide apps that allow users to access their data through a mobile phone or mobile device.

3. Forms-based Interfaces. It displays a form to each user. Users can fill out all of the form entries to insert new data, or they can fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries. Forms is a form-based language that specifies queries using a form designed in conjunction with the relational database schema.

4. Graphical User Interfaces. A GUI typically displays a schema to the user in diagrammatic form. The user then can specify a query by manipulating the diagram. In many cases, GUIs utilize both menus and forms.

5. Natural Language Interfaces. These interfaces accept requests written in English or some other language and attempt to understand them. It usually has its own schema, which is similar to the database conceptual schema, as well as a dictionary of important words, that are used to interpret the request. If the interpretation is successful, the interface generates a high-level query corresponding to the natural language request and submits it to the DBMS for processing; otherwise, a dialogue is started with the user to clarify the request.

6. Keyword-based Database Search. They use predefined indexes on words and use ranking functions to retrieve and present resulting documents in a decreasing degree of match.

7. Speech Input and Output. The speech input is detected using a library of predefined words and used to set up the parameters that are supplied to the queries. For output, a similar conversion from text or numbers into speech takes place. Applications with limited vocabularies, such as inquiries for telephone directory, flight arrival/departure, and credit card account information, are allowing speech for input and output to enable customers to access this information.

8. Interfaces for Parametric Users. Parametric users, such as bank tellers, often have a small set of operations that they must perform repeatedly. For example, a teller is able to use single function keys to invoke routine and repetitive transactions such as account deposits or withdrawals, or balance inquiries.

9. Interfaces for the DBA. Most database systems contain privileged commands that can be used only by the DBA staff. These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a database.

Classification of DBMSs

- **Based on the data model used**
  - **Traditional: Relational, Network, Hierarchical.**
  - **Emerging: Object-oriented, Object-relational.**

- **Other classifications**
  - Single-user - supports only one user at a time (typically used with personal computers) vs. multi-user - support concurrent multiple users (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)

Variations of Distributed DBMSs (DDBMSs)

- ✓ Homogeneous DDBMS-use the same DBMS software at all the sites
- ✓ Heterogeneous DDBMS-can use different DBMS software at each site
- ✓ Federated or Multidatabase Systems- in which the participating DBMSs are loosely coupled and have a degree of local autonomy.
- ✓ Distributed Database Systems have now come to be known as client-server based database systems because:
- ✓ They do not support a totally distributed environment, but rather a set of database servers supporting a set of clients.

Cost considerations for DBMSs:

- ✓ **Cost Range:** from free open-source systems to configurations costing millions of dollars
- ✓ Examples of free relational DBMSs: MySQL, PostgreSQL, others

- ✓ Commercial DBMS offer additional specialized modules, e.g. time-series module, spatial data module, document module, XML module
- ✓ These offer additional specialized functionality when purchased separately
- ✓ Sometimes called cartridges (e.g., in Oracle) or blades
- ✓ **Different licensing options:** site license, maximum number of concurrent users (seat license), single user, etc.

History of Data Models:

- **Network Model**
- **Hierarchical Model**
- **Relational Model**
- **Object-oriented Data Models**
- **Object-Relational Models**

■ Network Model:

- ✓ The first network DBMS was implemented by Honeywell in 1964-65 (IDS System).
- ✓ Adopted heavily due to the support by CODASYL (Conference on Data Systems Languages) (CODASYL - DBTG report of 1971).
- ✓ Later implemented in a large variety of systems - IDMS (Cullinet - now Computer Associates), DMS 1100 (Unisys), IMAGE (H.P. (Hewlett-Packard)), VAX -DBMS (Digital Equipment Corp., next COMPAQ, now H.P.).

Example of Network Model Schema:

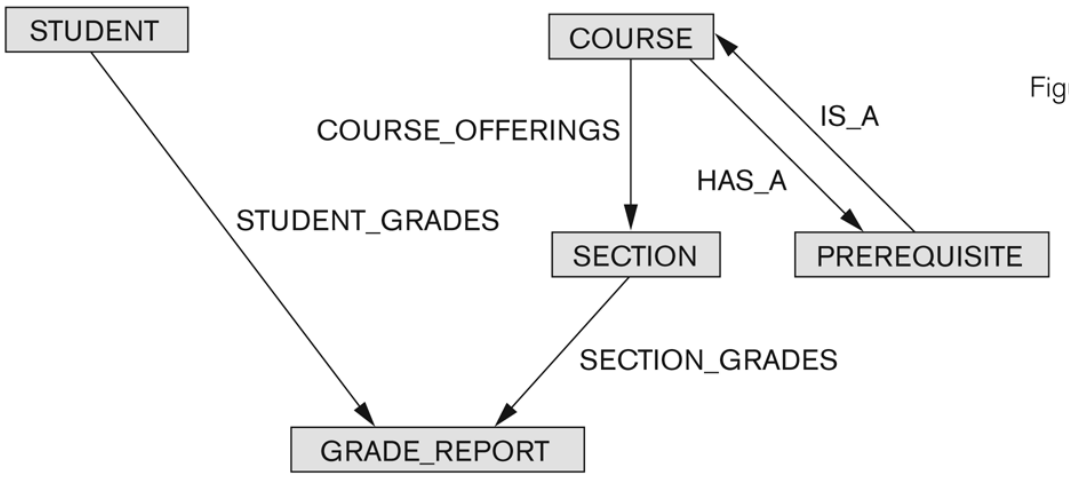


Figure 2.8
The schema of Figure 2.1 in network model notation.

■ Advantages:

- o 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"

- **Hierarchical Data Model:**
  - ✓ Initially implemented in a joint effort by IBM and North American Rockwell around 1965. Resulted in the IMS family of systems.
  - ✓ IBM's IMS product had (and still has) a very large customer base worldwide
  - ✓ Hierarchical model was formalized based on the IMS system

  - **Advantages:**
    - ✓ Simple to construct and operate
    - ✓ Corresponds to a number of natural hierarchically organized domains, e.g., organization ("org") chart
    - ✓ 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"

- **Relational Model:**
  - ✓ Proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82.
  - ✓ Now in several commercial products (e.g. DB2, ORACLE, MS SQL Server, SYBASE, INFORMIX).
  - ✓ Several free open source implementations, e.g. MySQL, PostgreSQL
  - ✓ Currently most dominant for developing database applications.
  - ✓ SQL relational standards: SQL-89 (SQL1), SQL-92 (SQL2), SQL-99, SQL3, …

- **Object-oriented Data Models:**
  - ✓ 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 O2, ORION (at MCC - then ITASCA), IRIS (at H.P.- used in Open OODB).
  - ✓ Object Database Standard: ODMG-93, ODMG-version 2.0, ODMG-version 3.0.
- **Object-Relational Models:**
  - ✓ Most Recent Trend. Started with Informix Universal Server.
  - ✓ Relational systems incorporate concepts from object databases leading to object-relational.
  - ✓ Exemplified in the latest versions of Oracle-10i, DB2, and SQL Server and other DBMSs.
  - ✓ Standards included in SQL-99 and expected to be enhanced in future SQL standards.