

CS470: Introduction to Database Management Systems

Introduction (Chapters 1 and 2)

**V Kumar
School of Computing and Engineering
University of Missouri-Kansas City**

Introduction

A *database* is a *fully connected* repository of *information*. "*Fully Connected*" is its one of the most important property, which indicates that the pieces of *information* stored in the database are inter-connected either semantically or through some link. This connectivity allows the user to go from one piece of *information* to another piece in the same session. Thus, a manager (database user) after accessing the job history of an employee can access company history or the academic history of another employee in the same session. No need to log out and log back in to access the desired information.

In this course we will study in detail how *information* (data + data semantics) is managed (structured and stored) and how *data* is processed. To fully understand all these and to be able to use them require that a number of basic concepts must be clearly understood. We define below some of the essential concepts:

- Data:** A known fact, which is usually has some value, numerical or character or bits, for example, value of SSN, value of phone number, and so on. Note that just these values are meaningless. For example, a number 111 55 3333 is not very useful unless we know its meaning (semantics). Thus, if we say SSN is 111-55-3333, then only it becomes useful.
- Information:** *Data + relevant semantics* creates information. For example, SSN = 111-55-3333 or Account No. = 111-55-3333. "SSN =" or "Account No." is semantics which is added to the data to build information.
- Object:** Anything that can be talked about or can be described. Example: House, Car, etc. In other words an object has a value and associated semantics.
- Attributes:** Differentiate two objects. Example: Color, Weight, Time, Age, etc.
- Entity:** An entity is an object, which can be identified with a set of attributes, for example entity $E = \{Name, Attributes, Data\}$.
- Database:** A repository of a set of semantically related entities. The repository as a whole has some meaning that can be understood by people.

A part of the real world, e.g., a university, or a bank, or Window NT, etc., can be fully described in a database using data and relevant semantics. Thus, in the database the University of Missouri-Kansas City can be structured and stored using a set of entities, correct semantics and values. For example, Name = University of Missouri-Kansas City, Location = Rockhill Road, Address = 5100 Rockhill, and so on. The data value (e.g., University of Missouri-Kansas City) is stored in the database and the semantics (Name =) is stored in the "catalogue". When data value is changed its semantics does not change but when the information changes, then both data and semantics must change.

There are two types of real world objects (a) *dynamic*, i.e., they changes continuously and (b) *static*, i.e., they never change. Changes to a dynamic object must reflect in the database. For example, weather of a geographical location is a dynamic entity because it continuously changes and biological parents of a person is a static entity because this relationship never changes.

The attribute values of an entity may change. For example, the age of an employee of a company changes, the medical history may change, employee salary may change, etc. These changes must be incorporated in the database to preserve the facts. This state of the database is referred to as "*consistent*". These changes are incorporated in the database by a set of software

modules. When we combine the database with the set of software module, then the complete system is referred to as a Database Management System (DBMS) or Database System (DBS).

Note that a database reflects the structure of an organization (part of the real world) in terms of *entities and their attributes*. These *entities* must be stored in the computer using some *data structure* for manipulation. The *data structures* used to store entities in a database is referred to as *files*. A *file* has the following components.

Database: A set of related files available to the user and accessed via system software. We will see how these files are grouped together later.

File: It is a set of identical *records* which can be accessed via system software.

Record: It is a set of logically related *fields*. Records are accessed and manipulated via system software.

Field: It is a set of bytes which stores attribute value. A field is the lowest level of logical structure..

At the physical level everything is seen as bytes or bits. However, in dealing or studying database technology, we do not talk about bits or bytes.

Data Model

Data is stored in the database for processing. When the data volume is large, then for efficient processing a model is required. For example, if one has to manage thousands of SSN, phone numbers, and addresses, then it is helpful to organize them in some model. You may put SSN first, then phone numbers, and finally all addresses. This means you have a model, which also defines searching methods, new data inserting methods, and so on. So a model is a data representation template that defines rules for processing these data. We will study in great detail different kinds of model and their properties.

Database Processing

A database must always store facts so that a user always receives the correct information. It is possible that the same data might be modified by more than one user. For example, the husband may deposit some money and at the same time his wife may also deposit or withdraw some money from the same joint account. If these operations are not processed carefully, then the final value may not represent the fact. The database system uses complex mechanism, referred to as “Concurrency Control Mechanism (CCM)” to guarantee correctness (*consistency*) of database data. We will study these mechanisms in brief later.

Database Recovery

Database system may fail for a variety of reasons. If it does, then it must be recovered so that most recent processing is not lost. Database recovery is a complex process and we will study in brief database system recovery protocols.

A Simple Database

We present a very simple database.

Part of the real world: Undergraduate Students.

Name of the database: Undergraduate Student Database

Entities: Students, Prerequisite, Grade report, Courses, and Section.

Student

Name	Student No.	Class	Major
Smith	17	1	CS
Brown	8	2	Phy

Prerequisite

Course No.	Prerequisite
CS431	CS352
CS470	CS352

Grade Report

Sid	SecID	Grade
1	A	A
2	A	B
3	B	A
4	C	C

Course

Course	Course No	Credit Hrs.	Dept.
Intro. To C	CS101	3	CS
Intro to OS	CS431	3	CS
Intro to DBS	CS470	3	CS
Intro to DS	CS352	3	CS

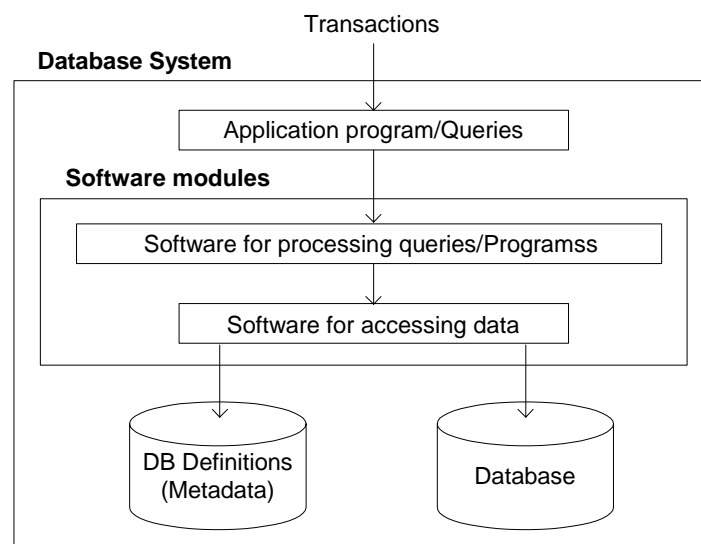
Section

Sec.	Course No.	Sem.	Yr	Instructor
B	CS101	F	98	Hines
A	CS431	W	99	Appie
A	CS470	W	99	Kumar
A	CS352	S	97	Mullins

Information about students is routinely processed so data must be stored in a suitable data model. The model, which is used here, is called *Relation Data Model*, which will be discussed in detail later. Data values are stored in a tabular structure and the related semantics (i.e., meaning of each entity, attribute, referred to as “metadata”.) is stored in database catalogue.

These entities are modified by a set of system software modules. A partial list of operation a user can perform using the set of software modules are as follows. Note that all these activities can be performed in a single session.

- Reads a row (group of fields: Course, Course No. Credit Hrs, Dept., etc.).
- Modifies a field of a row of a table.
- Create a row with desired fields.
- Create a file (set of rows).
- Delete a file/record/field.
- Print the desired information.
- Store new information in a file, etc.

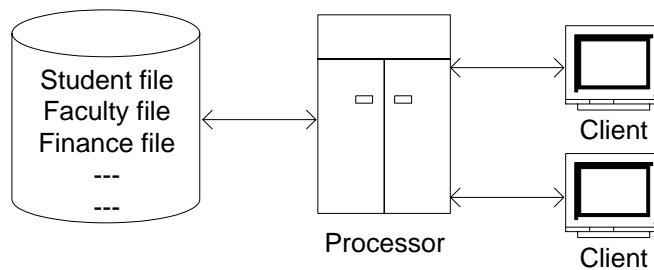


A simplified Database Management System architecture

A user issues a query to the database and the database management system makes sure that the request is processed. The entire processing requires a number of components (hardware and software) for completion. The above diagram illustrates some of the essential components of a database management system.

File Processing System

It is a simple *disconnected* system. Data such as student record, university finance, faculty record, etc. are stored in individual file. For example student record is stored in a file, faculty record is stored in a totally different file, and so on. There may be some common data, for example student names, between the student and the faculty files. The system is disconnected in the sense that each file exists independently and there is no way one can navigate, for example, from student file to faculty file. They are processed in a disconnected manner. The following diagram illustrates a typical file system architecture.



A File Processing System

A typical file processing session processes only one file at a time. Thus a client can open student file process it and closes it before it can process faculty file. Note that opening both files on the desktop monitor for editing is not the same as navigating and processing multiple files in the same session.

Difference between a DBMS and a Conventional File Processing System

A conventional file processing system can also process data stored in these files, however, it has a number of limitations, which makes it unsuitable for processing and managing data efficiently. Suppose we have the following set of files in a university:

- Faculty file containing salary information. Processing software Payroll system
- Class file containing class scheduling data and class timetable. Processing software Scheduling system
- Student data files containing grade report information. Processing software Grade posting system.

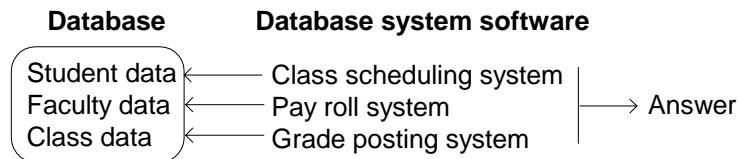
Suppose a user wants to get the answer of the following query:

Get the salary paid to each instructor who teaches a class scheduled by the class scheduling system.

The conventional file processing system treats each file separately. The payroll system processes only the faculty file, the class scheduling system processes only the class data, and the grade posting system processes only student data. To obtain the answer to the query three programs must be executed separately in three independent sessions. In this scheme, unfortunately,

there is no guarantee that data files would be compatible and consistent. The faculty data file might be written in COBOL binary format, whereas an incompatible PL/1 record format might be used for the class data file. In this situation one file must be converted to the format of the other before it can be processed. Furthermore, same data may exist in student files and faculty files, which could be inconsistent.

These problems and limitations are easily eliminated by database management system, where the faculty, class, and student data can be processed as an integrated whole. Since the files have been created by the DBMS all of the data are compatible. Such processing is called integrated processing and so the name of such system is Integrated Database Management System. The following diagram illustrates the way a DBMS handles the files.



Advantages of DBMS

- **Data Integration:** Files are integrated so no separate sessions are required to process data stored in different files.
- **Minimizing Data Redundancy:** In file processing the same information may be stored in more than one file. Such duplication (redundancy) is not required in DBMS.
- **Data Independency:** In file processing system the program such as payroll package interacts directly with the data file. This requires that the program must contain the description of the format (record type, field type, file organization etc.) of the file it uses. This creates problem when a file is changed. For example if the ZIP field is expanded to nine digit, all programs that access a file containing ZIP code will need to be modified, even if those programs do not manipulate ZIP code. This situation does not occur in a DBMS.
 - **Physical Data Independency:** It ensures that the physical organization of data files is transparent to the application programs. For example changes in the physical location such as disk to magnetic tape or to drum does not affect any application program.
 - **Logical Data Independency:** It ensures that the logical organization of data files is transparent to the application program. For example changes in a file type from index sequential to random, serial to inverted organization etc., does not affect the application programs.
- **Multiple Views of the Data:** A database has a variety of users. Some may view the stored information differently than others. The registrar may view students as customers, a faculty may view students as nice person to have intellectual discussion. So the registrar may be interested to see a student's financial situation, a faculty may like to see student's academic record. These two users (registrar and faculty) will have a different set of information from the database regarding a student.
- **Better Data Management (Shareability and Access Flexibility):** One department is responsible for maintaining the data structure, i.e. centralized management. The standard also makes sure that desired record can be accessed via many different routes. For example an

employee record can be accessed via his/her social security number or via his/her name or via telephone number etc.

- **Reliability:** Database system does not fail very often.
- **Integrity:** Since a database represents a picture of a part of the real world at any time its contents must be consistent, i.e., tells the correct story. In a system controlled centrally it is easy to maintain data integrity. Therefore, at any time the database gives a correct picture.

Disadvantages

- **Expensive:** More hardware (CPU, memory), Higher operating cost.
- **Complex:** Difficult to develop and implement.
- **Recovery:** More difficult.
- **Vulnerability to failure:** High.

Some Important Terms

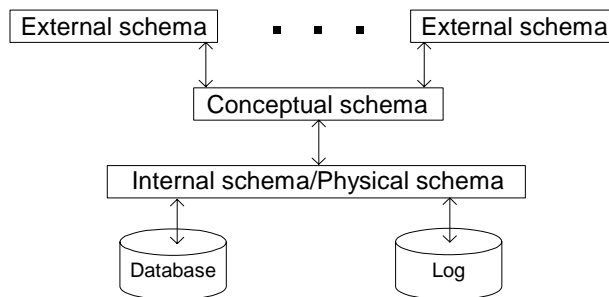
- **Structure:** Shows how related components are put together and a set of rules to manage them.
- **Instances (Occurrence):** When correct values are assigned to the structure, then an instance of the structure is created.

The following diagrams illustrate the structure and its multiple occurrences.

Structure		Instance 1		Instance 2		Instance 3	
Name	SSN	Name	SSN	Name	SSN	Name	SSN
		Stack	112233445	Mullins	100223344	Ram	100100100
		Appie	111222333	Hines	188990077	Hari	222333898
				Peter	599599599	Prasad	123412345
						Tony	119988777

- **Schema:** Structure of the database. So in database terminology the structure is a schema and the instance of a schema is a database. Generally there are three types:
 - **Internal schema:** Describes how the data is going to be stored on the disk.
 - **Conceptual schema:** Describes the structure of the database to the database designer.
 - **External schema:** Describes the structure of the database to end users. An end user gets the view of the database via external schema.

The three schemas relationship is illustrated in the following diagram:



Data Definition Language (DDL): A high level language to define data types. Every database system has its own DDL.

Data Manipulation Language (DML): A language to manipulate database.

Storage Definition Language (SDL): Describes how data will be stored on the database disks.

View Definition Language (VDL): Describes how view will be presented to an end user.