

Data and Applications

Shubhankar Kamthankar

Lec 02 Notess

23-08-21

Course Instructor - Prof. Kamal Karlapalem

1 Simplified Database System

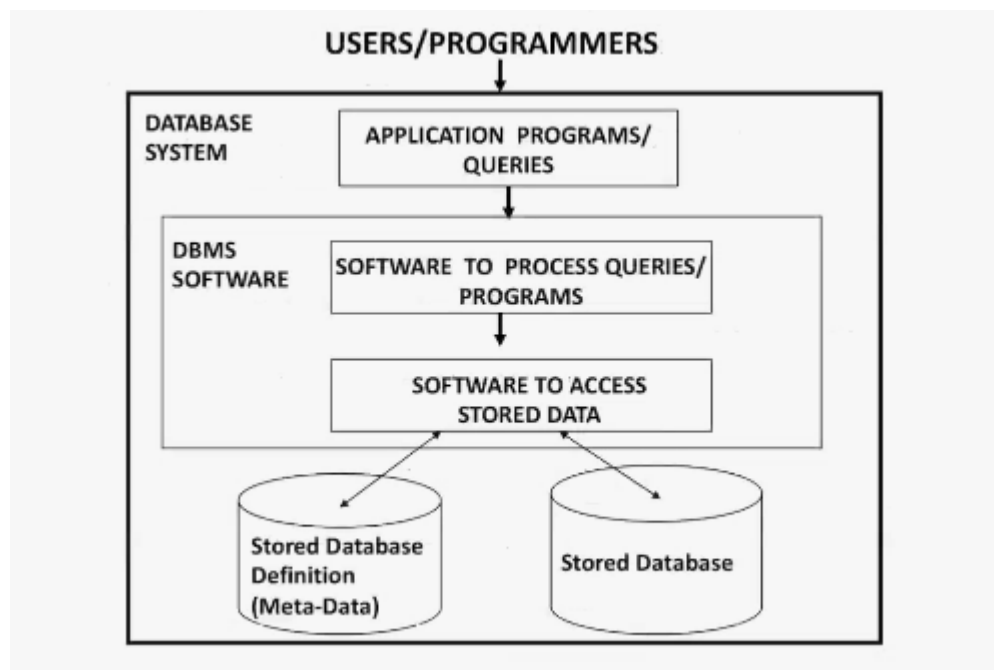


Figure 1:

Let's look at this figure from bottom up:

- The stored database exists(obviously). The stored database doesn't make any sense unless there's a definition of that stored database (else it'd just be random data that makes no sense)
- This definition of stored database is also called as Meta-Data
- The course does not deal with the DBMS Software (It'll be covered in the Data Systems course)

- Consider an environment (Universe of Discourse), come up with data requirements and design a database for the application.

Questions to ask self while designing Database :

- Have I captured all the data requirements?
- Have I captured all the aspects of the database in order to support that particular application of the user?

It is possible that you might miss out on a few requirements, or might have captured it wrongly. Those are the aspects we also concern ourselves with.

2 A Historical Take

In 1970s, computers were new. They were just magnetic disks based storage system and the concept of files was popular at the time(paper-based). This was later mimicked by the computer file system. This later on came to be reflected in the database systems as well. The main features to be highlighted are :

- The database aspects are stored in files. This became a solution. eg: Consider course registration, you'd have one file for the list of students, another for course details and so on.(or you could have a single file containing all the info)
- You open these files using a certain program to access and modify the details of this information as necessary.
- Consider another example, i.e College Library Database. You have library books' information being stored in files. The replication of the effort of this read and write utility was there for similar applications.
- It was later realised that read/write was a part of the generic functionality. So, the traditional file systems had to closely integrate the data structures in the program to the file definitions.
- The cost of maintaining these programs and their correctness became an issue. (As the tight integration between the program and the file system wasn't implemented at that time.
- The idea brought by the Database System: The data already exists in the files. **So, can I describe this data?** So, the groundbreaking idea was to store the Meta-Data and the actual Data in a single system. If this was realised then potentially one would be liberated from specific application requirements. (i.e you no longer had to create a new software for every database)

Post adopting this model, the following procedure became the norm: Suppose you wish to edit a field. Firstly the DB will check whether the entry for the field exists in the Meta-Data. If the entry exists, the DB actually proceeds to with the user request, else it throws an error. Hence the data and the MetaData correspond to each other. This is also known as **SELF DESCRIBING NATURE OF THE DB**

3 Example of a Database

Example Database

STUDENT			
Name	Student Number	Class	Major
Smith	17	1	COSC
Brown	8	2	COSC

GRADE REPORT		
Student Number	Section-Identifier	Grade
17	85	A
18	102	B+

PREREQUISITE	
Course Number	Prerequisite Number
COSC3380	COSC3320
COSC3320	COSC1310

COURSE			
Course Name	Course Number	Credit Hours	Department
Intro to CS	COSC1310	4	COSC
Data Structures	COSC3320	4	COSC
Discrete Mathematics	MATH2410	3	MATH
Data Base	COSC3380	3	COSC

SECTION				
Section-Identifier	Course Number	Semester	Year	Instructor
85	MATH2410	Fall	91	King
92	COSC1310	Fall	91	Anderson
102	COSC3320	Spring	92	Knuth
135	COSC3380	Fall	92	Stone

Figure 2:

Observe the above figure carefully. In the above data, a copy of table headers are stored in the meta-data. (i.e for the STUDENT data table, the metadata would contain the fields Name, Student Number, Class, Major, etc.)

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.

In the database approach, a single repository of data is maintained that is defined once and then is accessed by various users.

Self- Describing nature of the database System: The database system contains not only the database itself but also a complete definition (meta-data) of the database which is stored in the **ssystem catalog**

Insulation between programs and data, and Data Abstraction: The structure of data files is stored separately from the access programs (program-data independence).

The characteristic that allows program data independence is called **Data Abstraction**. A DBMS provides user with a conceptual representation of data that does not include many of the details of how the data is stored.

- Support for multiple views of the data.
- Sharing of Data and Multi-user Transaction Processing.

A typical database management system has all the above characteristics, whereas a typical file processing system does not.

5 Actors on Scene (unimportant):

The people whose jobs involve day-to-day use of a large database

- Database Administrators
- Database Designers
- End Users: Casual, Naive, Sophisticated and Stand-alone users
- System Analysts and Application Programmers

Workers Behind the scenes:

- DBMS Designers and Implementers
- Tool Developers
- Operators and Maintenance Personnel

6 Intended Uses of a DBMS

- Controlling redundancy - Multiple data tables calling (or needing) a common field are stored just once in the metadata in order to control the redundancy and wastage of data, and to avoid inconsistency (different versions of the data in a single database is not desirable)
- Restricting Unauthorized Access - There is a enough security restrictions put in place.
- Representing complex relationships among data - Self-explanatory
- Enforcing integrity constraints - For example, in an employee's salary field, only numbers as in input is accepted.
- Providing Backup and Recovery
- Providing multiple User Interfaces - There might exist multiple interfaces through which you can access your data. eg. Consider Banking. You can access your details through ATM card, mobile, net-banking, etc. All of which are distinct user interfaces
- Persistent Storage for Program Objects and Data Structures
- Database Interfacing Using Deduction Rules

Implications of the Database Approach:

- Reduced application development time
- flexibility
- Available up-to-date information
- Economics of Scale

7 When NOT to use a DBMS

Main costs of using of 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

8 Data Model

The fundamental question that arises is : How do you specify the metadata? The answer to this question is the data model. Data model has **concepts** to describe **structure** of a database. Also, the operations on these structures along with the constraints. Hence, there is a notion of finiteness about the data, which gives (rather necessitates) the structure. It is thus implicitly states 'what is related to what?'

The CONCEPT will give rise to the terminology to denote the meta-data about the data.

Data model

- A set of concepts to describe the **structure** of a database, the **operations** for maintaining these structures, and certain **constraints** that the database should avoid.

Data Model Structure and Constraints

- Constructs are used to define the database structure
- 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.

9 Operations on a Data Model

- 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_sstudent_{gpa}, update_inventory*)

10 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:** Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

11 Schemas versus Instances

Schema -

- Database Schema
 - The **description** of a database
 - Includes descriptions of the database structure, data types, and the constraints on the database.
- 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
- 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, etc.

12 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 state when it is initially loaded into the system
- Valid State:
 - A state that satisfies the structure and constraints of the database.
- Distinction
 - The **database schema** changes very infrequently
 - The **database state** changes every time the database is updated.
- **Schema** is also called **intension**
- **State** is also called **extension**

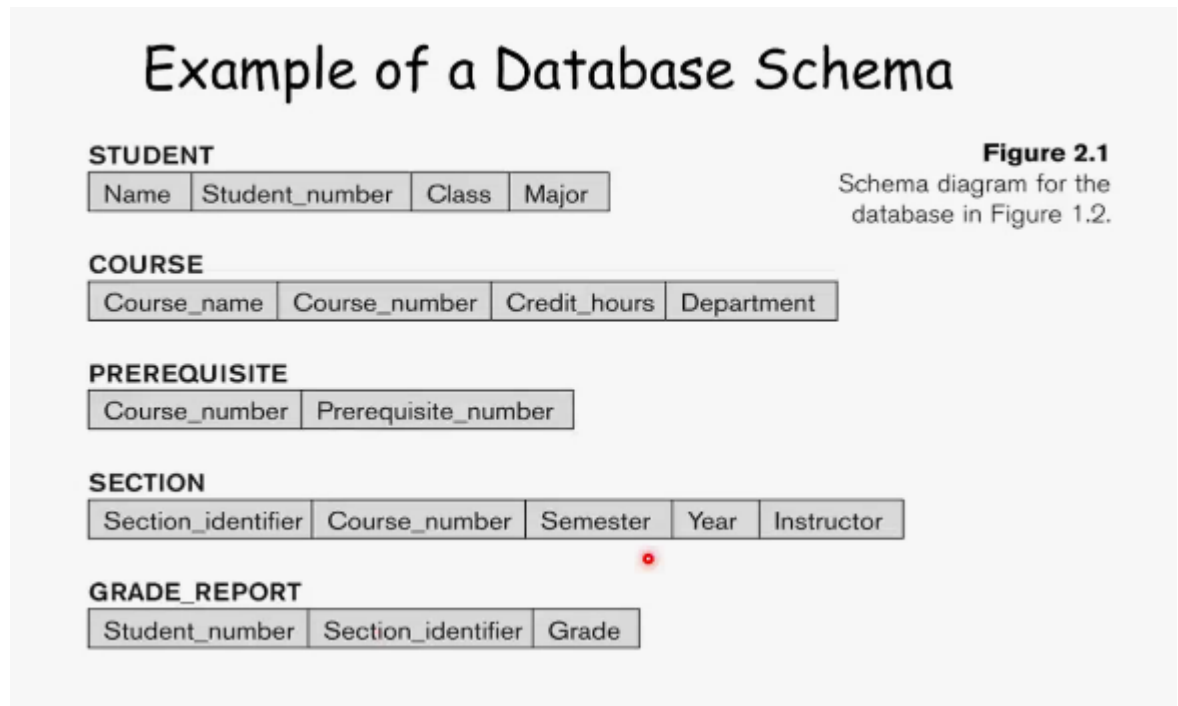


Figure 3:

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	Kruth
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.

Figure 4:

13 Three Schema Architecture

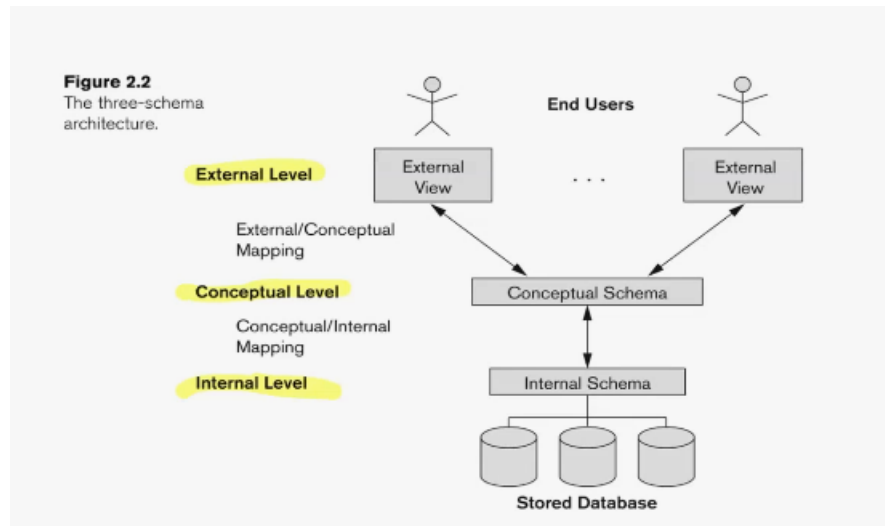


Figure 5:

To be continued the next lecture.