International Institute of Information Technology Bangalore

Introduction to Database Systems

CS 301

Course overview

- Ability to deal with data plays a critical role virtually in all disciplines of Information Technology.
- The core course titled "Database Systems" is the first level course that builds the foundations needed for dealing with persistent data.
- This course covers all essential topics in database management in a fast-track mode.
- The foundations laid in this course will serve as required pre-requisite to several elective courses in the areas of Data Science and Software Engineering (e.g., Data Modeling, Data Analytics, GIS, Spatial Computing, OOAD, and so on).

Goal of the course

- To introduce the fundamental concepts for designing, using and implementing database “systems” and database “applications”.
- To explore the fundamentals of database design.
- To learn database system implementation techniques.

Course objectives

At the end of the course, you should have knowledge and competencies in the following areas:

- Understand the principles of conceptual modelling.
- Design databases.
- Principles of database programming.
- Knowledge of DBMS components.
- Other data management technologies (e.g., data exchange, in-memory, etc.).

Course contents

- Information systems: Basic concepts (models, schema, data, information, knowledge), elements of information systems, overview of database systems.
- Conceptual modeling: Introduction to conceptual modeling, entity relationship models, UML class diagrams.
- Relational databases: Relational data model, database design concepts, DB design via OR mapping, SQL tutorial, functional dependencies, overview of normal forms.
- DBMS: Components of a DBMS, storage structures – primary, clustering, secondary, multi-level, query processing – overview, query transformation, query evaluation, transaction processing – overview, ACID properties, concurrency control – schedules, serializability, deadlocks.
- Other topics: Spatial Databases, etc.

Learning outcomes

1	Understand the introductory concepts of database models, systems, architectures, terminology and languages.
2	Understand the entity–relationship modelling and database design.
3	Explain conceptual modeling, entity relationship models, UML class diagrams.
4	Explain the principles of relational database and SQL.
5	State the normalisation and relational design theory, define the functional dependencies and normal forms.
6	Describe file organisation on disk, file structure, indexing of database files and physical database design.
7	Explain strategies for query processing and query optimization
8	Summarise transaction processing concepts, concurrency control and database recovery from failures.

Reference books

1. Fundamentals of Database Systems; R. Elmasri and S. Navathe; *Addison-Wesley, 2000.*
2. An Introduction to Database Systems; Bipin Desai; *Galgotia Publications (West Publishing), 1991.*
3. Modern Database Management (Fourth Edition); F. McFadden, J. Hoffer; *Benjamin/Cummings (Narosa), 1994.*
4. An Introduction to Database Systems (Seventh Edition); C. J. Date; *Addison-Wesley, 2000.*
5. Principles of Database Systems (Second Edition); J. D. Ullman; *Galgotia Publishing, 1994.*
6. Database Processing: Fundamentals, Design, Implementation (Fifth Edition); D. M. Kroenke; *Prentice-Hall, 1994.*
7. Database Systems Concepts, Abraham Silberschatz, Henry F. Korth and S. Sudarshan, 7th Edition, *McGrawHill, 2019.*
8. A First Course in Database System, Jeffrey D. Ullman and Jennifer Widom, *Pearson Education, 2002.*

Assessments

- Assignment – 1 (10 marks)
- Mid term (30 marks)
- Assignment -2 (10 marks)
- Assignment -3 (20 marks)
- Final exam (30 marks)

Consider the following examples:

- Banking system
- Making online reservation (such as hotel, airline, etc.)
- Library catalogue – searching for a specific book title
- Online shopping (amazon) or from a large supermarket
- Social media web sites – Facebook, Twitter – containing posts, tweets, images, video clips in big data format (NOSQL systems) stored in cloud storage.
- Multimedia databases (images, audio clips, video streams, etc.).
- Geographic Information Systems (GIS such as map, weather data, satellite images, attribute information).

all involve accessing some kind of database.

- Data warehouses* and OLAP (online analytical processing) systems are widely used in industries to extract and analyse useful business information from very large databases for decision making.

- Database search techniques are used in the WWW to improve the search for information by internet users.

- So, databases have a pivotal role in business, electronic commerce, social media, engineering, medicine, genetics, law, education, etc.

Data warehouse - "A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process". It focuses on the **modeling and analysis of data for decision makers.*

Defining a Database (DB)

Definition of database (DB)

- A "database" is a collection of **related data** (i.e. facts that can be recorded and have meaning) that is managed by a *database management system (DBMS)* or just *database system*.

- A "DBMS" is a computerised **system** that enables users to create and maintain a DB.

- DBMS is a *general purpose / special purpose (such as banking) software system* that facilitates the processes of *defining*, *constructing*, *manipulating*, and *sharing* DB among users and applications.

- The database and DBMS software are together called a ***database system***.

- Example, consider the names, mobile numbers and addresses of your friends.

- It is also possible to store them in an indexed address book or on a hard drive using MS Access/Excel.

- So, a collection of *related data* with an implicit meaning is a DB.

- A DB has the following implicit properties:

1. A DB represents some aspects of the "real world" and any changes in real world are reflected in the DB.

2. A DB is a "logically coherent collection of data" with some inherent meaning. Therefore, a random assortment of data cannot be a DB.

3. A DB is designed, built and populated with data for a specific purpose. It has users and serve some application.

So, a DB has some source from which the data are derived, some degree of interaction with events in the real world, and interested audience in the DB contents.

Understanding the *terminologies* in DB definition

- *Definition of a DB* – Defining a DB involves specifying the data types, structures, and constraints of the data to be stored in the DB.

- *Meta-data* – DB descriptive information is stored by the DBMS in a DB catalogue or dictionary (meta-data).

- *DB construction* – Constructing the DB is the process of storing the data on some storage medium that is controlled by the DBMS.

- *Manipulating a DB* – It means querying the DB to retrieve specific data, updating the DB to reflect changes, and generate reports.

- *Sharing* – It means allowing multiple users and programs to access the DB simultaneously.

- *Query* – An application program accesses the DB by sending queries or requests for data to the DBMS. A *query* typically causes some data to be retrieved and a *transaction* may cause some data to be read/written into the DB.

- *Protection* – Protecting the DB (system protection against hardware and software failure and security protection against unauthorised or malicious access).

- *Maintenance* – DBMS should be able to maintain the database system over a long period of time by allowing the system to evolve as requirements change over time.

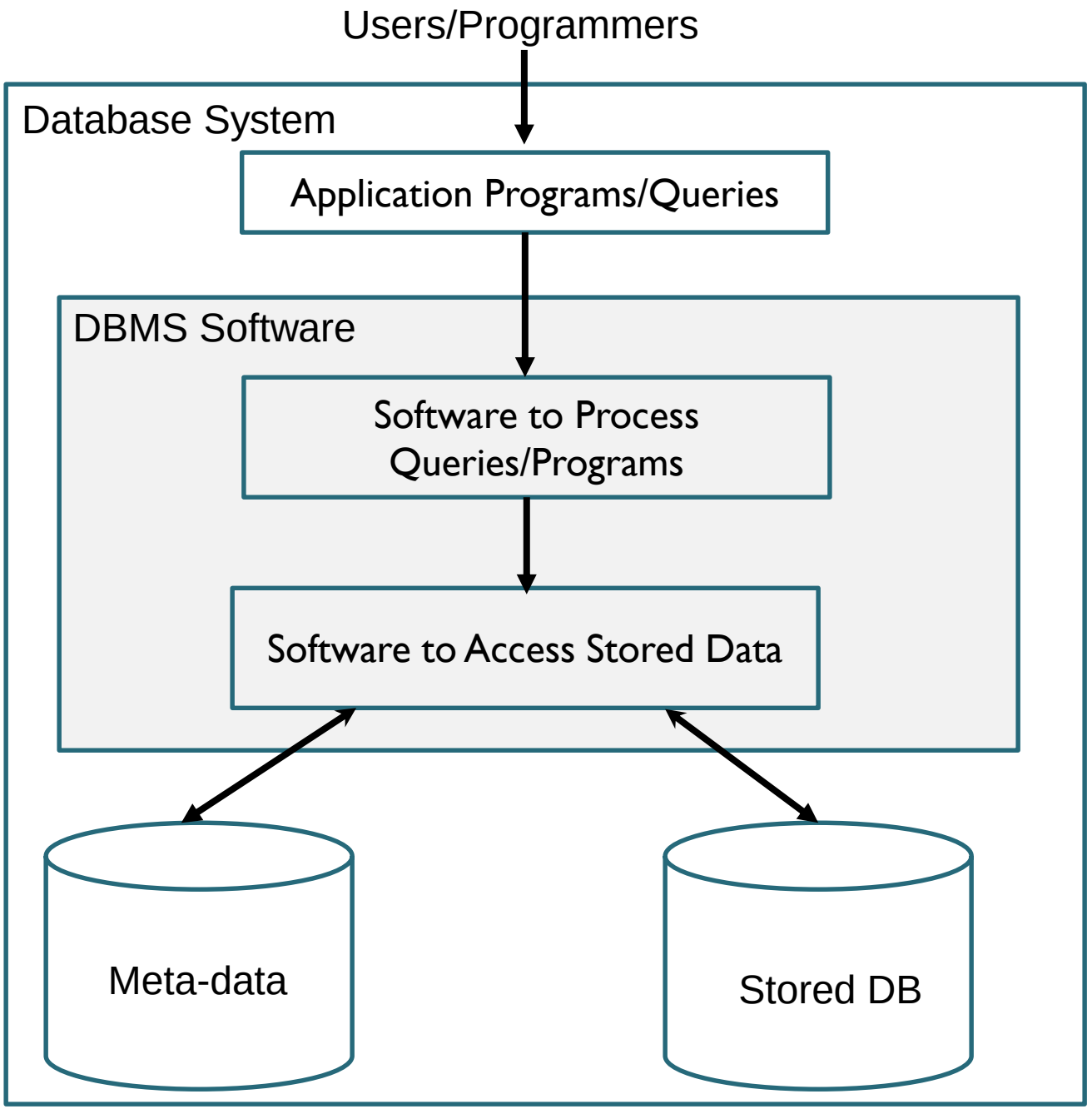

Figure: DB system environment.

What do you expect a database to do?

- A DBMS is expected to:

1. Allow users to create new DBs and specify their schema (logical structure of the data), using a specialised language called a ***data-definition language***.
2. Give users the ability to query the data (a "query" is a DB lingo for a question about the data) and modify the data, using an appropriate language – *query language* or ***data manipulation language***.
3. Support the storage of very large amounts of data – GB, TB, PB over a long period of time, keeping it secure and safe from unauthorised access and use, and allowing efficient access to the data for queries and DB modifications.
4. Control access to data from many users at once, without allowing the actions of one user to affect other users and without allowing "simultaneous accesses" to corrupt the data accidently.

Example of a large commercial DB – Amazon.com

- Amazon contains over 310 million users, millions of books, CDs, videos, DVDs, games, electronics, apparel, house hold items, etc.
- The DB occupies 160 exabytes (160 billion gigabytes/160 trillion bytes) stored in hundreds of servers.
- Millions of visitors access the website each day to make purchases.
- The DB is continually updated as new items are added to the inventory and updated as purchases are transacted.

Understanding a Database with an Example

- Consider a university/institute DB used for maintaining information of students, courses and grades.

STUDENT

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

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	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

- To define a DB, we must specify the "structure of the records" of each file by specifying different "data elements" for each record.

- Example, STUDENT record includes data to represent the student's *Name*, *Student_number*, *Class* (such as 1, 2, 3, 4, 5) and *Major* (such as Mathematics, CS), etc. Similarly, for COURSE, SECTION, GRADE_REPORT and PREREQUISITE.

- We also specify "data type" for each data element within a record. Example, Name is string, Student_number is integer, or use coding scheme so as to represent value of data item such as Class (1 - freshman, 2 - sophomore, 3 - junior, 4 - senior, 5 - graduate).

- Records in the various files may be related. Example, the record for Smith in the STUDENT file is related to two records in the GRAD_REPORT file that specify Smith's grades in two sections (see STUDENT and GRADE_REPORT table in previous slide).

- What types of queries can be made from the University DB?
  - Retrieve the transcript – a list of all courses and grades of Smith.
  - Names of the students who took Database course in fall of 2008 and their grades in that section.
  - List the prerequisites of the "Database" course.

What types of update can be made?

1. Change the class of Smith from 1 to 2.
2. Create a new section for the DB course for this semester.
3. Enter a grade ‘A’ for ‘Smith’ in the DB course.

STUDENT

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

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	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

- Design of a new DB application starts with a "*requirements specification and analysis*".

1. The requirements are documented in detail and transformed into a "*conceptual design*" that is transformed into a DB implementation.

2. It is then translated to a "*logical design*" usually expressed in a data model in a DBMS.

3. The final stage is "*physical design*" during which detailed specifications are provided for storing and accessing the DB.

- Finally, the DB design is implemented, populated with data, updated and maintained.

Requirements specification and analysis

↓

Conceptual design *(example*: using E-R model*)*

↓

Logical design (data model, *example:* RDBMS)

↓

Physical design

↓

DB design is implemented, populated with data, updated and maintained.

Characteristics of the DB approach

- In traditional file processing, "each user" defines and implements the files needed for a specific application.

- For example, one user, the grade reporting office will keep files on students and their grades – this is used for student's transcripts and to enter new grades. Another user, the accounts office will keep track of payments.

- Although both users are interested in data about students, each user maintains a separate file and program *to manipulate these files* because each requires some data not available from other user's files.

- This leads to "redundancy" and redundant efforts in maintaining common data.

- "In DB approach, a "single repository" maintains data that is defined once and then "accessed by various users" through queries."

- The main characteristics of the DB approach versus file-processing are:

1. Self-describing nature of DBMS.
2. Insulation between programs and data, and data abstraction.
3. Support of multiple views of the data.
4. Sharing of data and multiuser transaction processing.

1. Self-Describing Nature of DB System

- A DB contains a complete definition or description of the DB structure and constraints stored in ***DBMS catalog*** – with structure of each file, type and storage format and constraints – called ***meta-data***.

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 enumerated type with all known majors.
XXXXNNNN is used to define a type with four alphabetic characters followed by four numeric digits.

- This "*catalog*" is used by the DBMS and users about information on the DB structure.

- In traditional file processing, data definition is typically part of the application program themselves – these programs work with only one specific DB whose structure is declared in the application programs.

- Example, in a file processing application, file structure and location of Name within a STUDENT record might be coded, such as application program written in C++.

- On the other hand, DBMS can access diverse DBs by extracting the DB definition from the catalog.

2. Insulation between programs and data

- The structure of data files is stored in the DBMS catalog separately from the access programs. This is called "*program-data independence*".

- In o-o (object-oriented) and o-r (object-relational) systems, users can define "operations / functions / methods" on data as part of DB definitions. Operation includes *operation name* and *arguments* (*or parameters*). Example, CALCULATE_GPA can be applied to STUDENT to calculate the grade point average.

- So, the *implementation* (or *method*) of the operation is specified separately and can be changed without affecting the interface. This is called "*program-operation independence*".

- The characteristics that allow program-data independence and program-operation independence is called "***data abstraction***".

3. Support of multiple views of the data

- A DB may have many types of users and may require a different perspective or "**view**" of the DB.

- A view is a subset of the DB or may contain virtual data that is derived from the DB files but is not explicitly stored.

- Example, **one user** may be interested in the transcript of each student and a **second user** might be interested only in checking that students have taken all the prerequisites of each course (see next slide).

TRANSCRIPT

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

COURSE_PREREQUISITES

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

4. Sharing of data and multiuser transaction processing

- A multiuser DBMS should allow multiple users to access the DB at the same time.
- The DBMS must include "concurrency control" to ensure that several users trying to update the same data can do in a "controlled manner" to have a correct result.
- Example, several reservation agents try to assign a seat in an airline flight, DBMS must ensure that each seat can be accessed by only one agent at a time – called OLTP (online *transaction* processing).

- *A transaction is an executing program that includes one or more DB access – such as reading or updating of DB records.*

- Transaction properties:

1. Isolation property – ensures that each transaction appears to execute in isolation from other transactions, even though hundreds of transactions may be executing simultaneously.

2. The atomicity property – ensures that either all the DB operations in a transaction are executed or none are.

Users of DBMS

Users of DBMS

Actors on the scene

- DB administrators
- DB designers
- End users
  - Casual end users
  - Naïve or parametric end users
  - Sophisticated end users
  - Standalone users
- System analyst / Application programmers (Software Developers / Engineers)

Actors behind the scene

- DBMS system designers and implementers
- Tool developers
- Operators and maintenance personnel

Users of DBMS

Actors on the scene

- **DB administrators (DBA)** – responsible for *authorizing access* to the database, *coordinating and monitoring its use*, *acquiring software and hardware resources*, accountable for *security breaches*, *poor system response time*, etc.

- **DB designers** – responsible for identifying the data to be stored in the DB and for choosing appropriate structure to represent and store the data (usually done before the DB is actually implemented and populated with data).

  - They ***communicate*** with the prospective DB users ***to understand and meet their requirements***.

  - They ***develop "views" of the DB*** that meet the data and processing requirements.

Users of DBMS

- **End users** – are the people whose jobs require access to the DB for querying, updating and generating reports.
  - **Casual end users** – They occasionally access the DB and may need different information each time. They use sophisticated DB query interface for their requests (middle or high level managers).
  - **Naïve or parametric end users** – They constantly query and update the DB using standard "**canned transactions**" – that have been programmed and tested. Example of such tasks are **mobile apps** for Bank customers, Reservation for airlines, hotels, car rentals, courier companies using apps for entering package details, social media user post, etc.
  - **Sophisticated end users** – They include engineers, scientists, business analysts who use applications for complex requirements.

Users of DBMS

- End users –
  - **Standalone users** – They maintain personal DB by using ready-made program packages that provide easy to use menu-based GUIs. Example, financial software to maintain personal financial data.

- System analyst / Application programmers (also called Software Developers / Engineers)

  - **System analyst** determine the requirements of end users and develop "**specifications**" for canned transactions to meet these requirements.

  - **Application programmers -** implement these "**specifications**" as programs, test, debug, document and maintain the canned transactions.
  - They are familiar with the full range of capabilities provided by the DBMS.

Users of DBMS

Actors behind the scene

- **DBMS system designers and implementers** – They **design and implement the complex systems of DBMS modules** (for *implementing the catalog, query language processing, interface processing, accessing and buffering of data, controlling concurrency and handling data recovery and security, etc.*) and interface as a software package.

- **Tool developers** – Tools are optional packages that are purchased separately. Example, *packages for DB design, performance monitoring, natural language and graphical interfaces, prototyping, simulation and test data generation*.

- **Operators and maintenance personnel** – They are responsible for *the actual running and maintenance of the hardware and software environment for the database system*. They do not use the DB for their own purpose.

Advantages of using DBMS

- Control redundancy
- Restrict unauthorised access
- Provide persistent storage to program objects
- Provide storage structures and search techniques for efficient query processing
- Provide backup and recovery
- Provide multiple user interface
- Represent complex relationship among data
- Enforce integrity constraints
- Permit inferencing and actions using rules and triggers
- Potential for enforcing standards
- Reduced application development time
- Flexibility
- Availability of up-to-date information
- Economies of scale

Extending DB capabilities for new applications

DB systems offer extensions to support the specialised requirements of

- Scientific applications – that store large amounts of data resulting from experiments such as high energy physics, human genome mapping, discovery of protein structures, etc.
- Storage and retrieval of images – scanned photographs, scientific images, medical images, etc.
- Storage and retrieval of videos - movies, video clips, etc.
- Data mining – applications that analyse large amounts of data to search for the occurrences of specific patterns or relationships / unusual patterns.
- Spatial data analysis and GIS – store and analyse spatial locations of data such as satellite data, weather information, maps, autonomous vehicles, etc.
- Time series – applications that store information at regular points in time. Example, daily sales, temperature, etc.

Why was basic relational systems not suitable for many real time applications?

- More complex data structures were needed for modelling the applications than the simple relational representations.
- New data types were needed in addition to the basic data types.
- New operations and query languages constructs were necessary to manipulate the new data types.
- New storage and indexing structures were needed for efficient searching on the new data types.

Summary

- Database – a collection of related data, where data means recorded facts.
- A DBMS is a generalised software package for implementing and manipulating a computerised DB.
- The database and software together form a database system.
- There are several identified characteristics that distinguish DB approach from traditional file processing applications.
- There are many categories of DB users.
- We saw a list of capabilities that should be provided by the DBMS software to the DB administrators, designers, and end users.
- Finally, we looked into emergence of new system for handling "Big Data" applications.