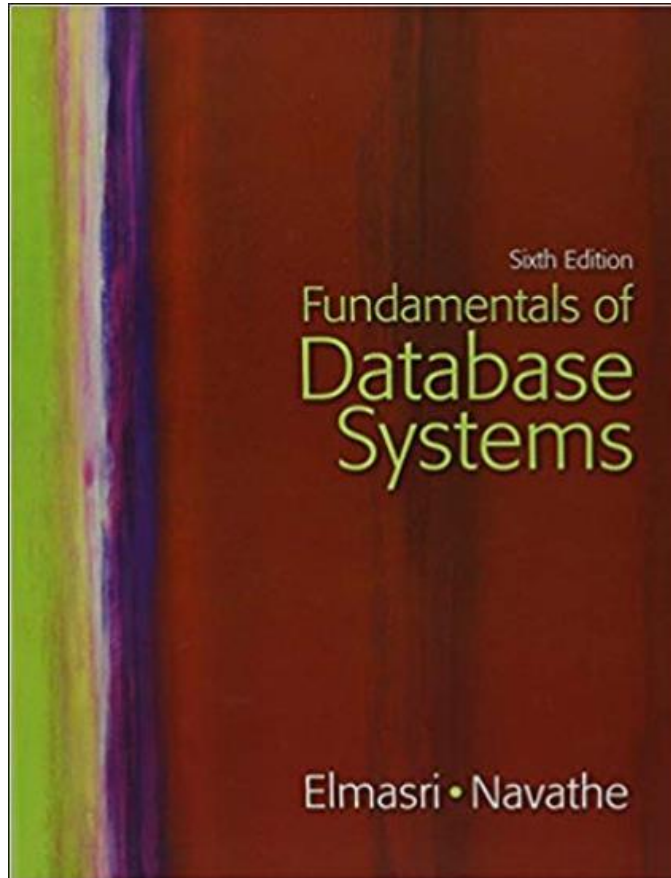
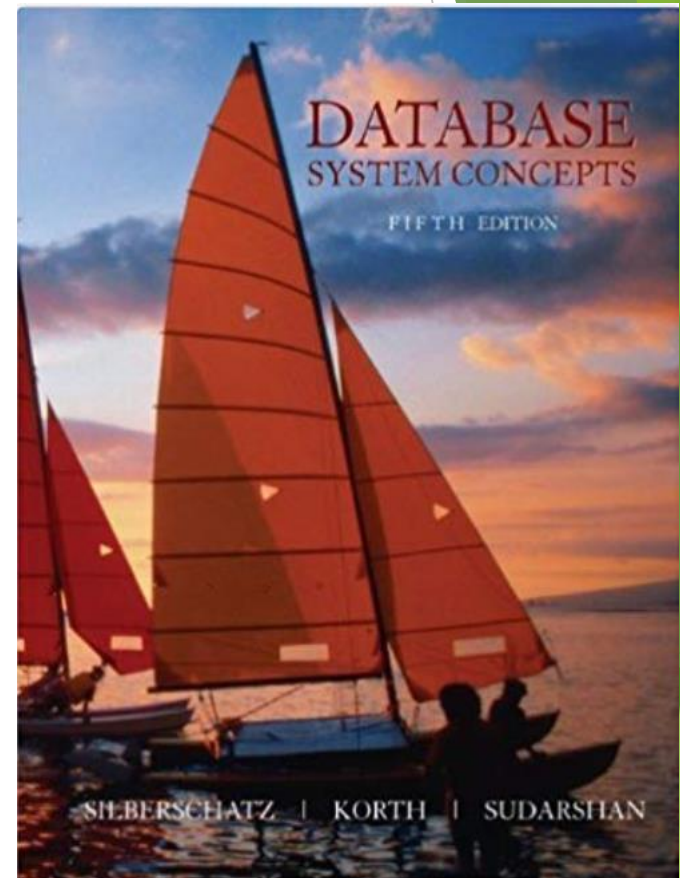


Textbooks



Fundamentals of Database System (6th Edition)
by Elmasri, Navathe



Database System Concepts (5th Edition) by
Berschatz, Korth, Sudharshan

CHAPTER 1

Introduction on Databases and Database Users

Outline

- ▶ Basic Definitions
- ▶ Impact of Databases and Database Technology
- ▶ Types of Databases and Database Applications
- ▶ Typical DBMS Functionality
- ▶ Drawbacks of using file systems to store data
- ▶ Example of a Database (UNIVERSITY)
- ▶ Main Characteristics of the Database Approach
- ▶ Types of Database Users
- ▶ Advantages of Using the Database Approach

Outline

- ▶ Additional Implications of Using the Database Approach
- ▶ Instances and Schemas
- ▶ Data Models
- ▶ Storage, Query processing and transaction management
- ▶ History of Database Technology
- ▶ When Not to Use Databases

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.

Impact of Databases and Database Technology

- ▶ **Businesses:** Banking, Insurance, Retail, Transportation, Healthcare, Manufacturing
- ▶ **Service industries:** Financial, Real-estate, Legal, Electronic Commerce, Small businesses
- ▶ **Education:** Resources for content and Delivery
- ▶ **More recently:** Social Networks, Environmental and Scientific Applications, Medicine and Genetics
- ▶ **Personalized applications:** based on smart mobile devices

Types of Databases and Database Applications

- ▶ Traditional Applications:
 - ▶ Numeric and Textual Databases
- ▶ More Recent Applications:
 - ▶ Multimedia Databases
 - ▶ Geographic Information Systems (GIS)
 - ▶ Biological and Genome Databases
 - ▶ Data Warehouses
 - ▶ Mobile databases
 - ▶ Real-time and Active Databases
- ▶ First part of book focuses on traditional applications

Recent Developments (1)

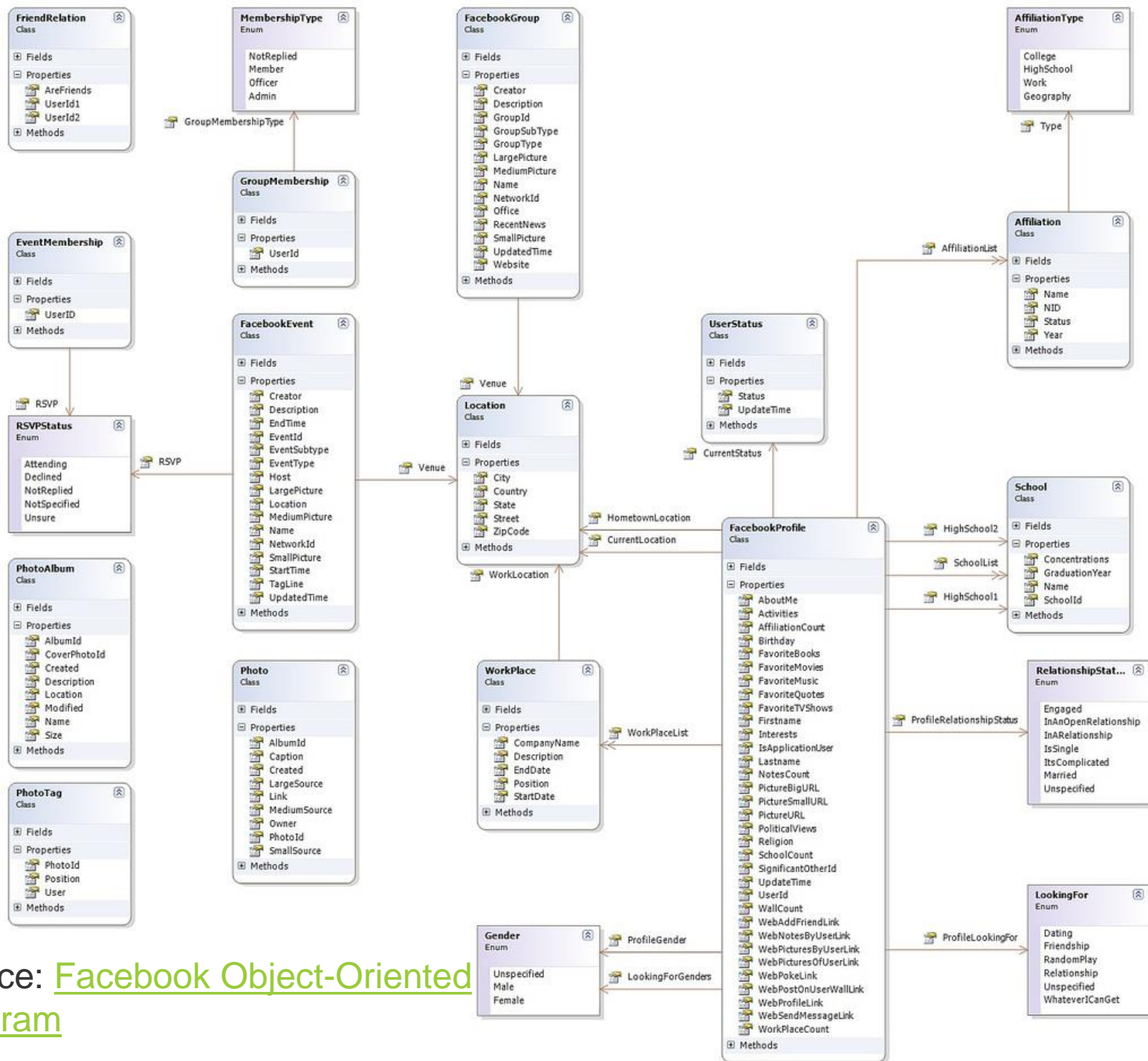
- ▶ Social Networks started capturing a lot of information about people and about communications among people-posts, tweets, photos, videos in systems such as:
 - Facebook
 - Twitter
 - Linked-In
- ▶ All of the above constitutes data
- ▶ Search Engines, Google, Bing, Yahoo: collect their own repository of web pages for searching purposes

Recent Developments (2)

- ▶ New Technologies are emerging from the so-called non-database software vendors to manage vast amounts of data generated on the web:
- ▶ **Big Data** storage systems involving large clusters of distributed computers
- ▶ **NOSQL** (Non-SQL, Not Only SQL) systems
- ▶ A large amount of data now resides on the “cloud” which means it is in huge data centers using thousands of machines.

List of Databases used by Big Companies:

- *The king of scalability, Google, uses **BigTable** (developed by Google).*
- *Facebook uses **Hive** (Data warehouse for Hadoop, supports tables and a variant of SQL called hiveQL) and **Cassandra** (Multi-dimensional, distributed key-value store) for Facebook's private messaging.*
- *Yahoo uses modified **PostgreSQL**.*
- *YouTube uses **MySQL** but they are moving to Google's **BigTable**.*
- *Myspace uses **SQL Server**.*
- *Twitter and Wikipedia uses **MySQL**.*
- *Microsoft uses **SQL Server**, which is very obvious.*
- *Flickr uses **MySQL**.*



source: [Facebook Object-Oriented Diagram](#)

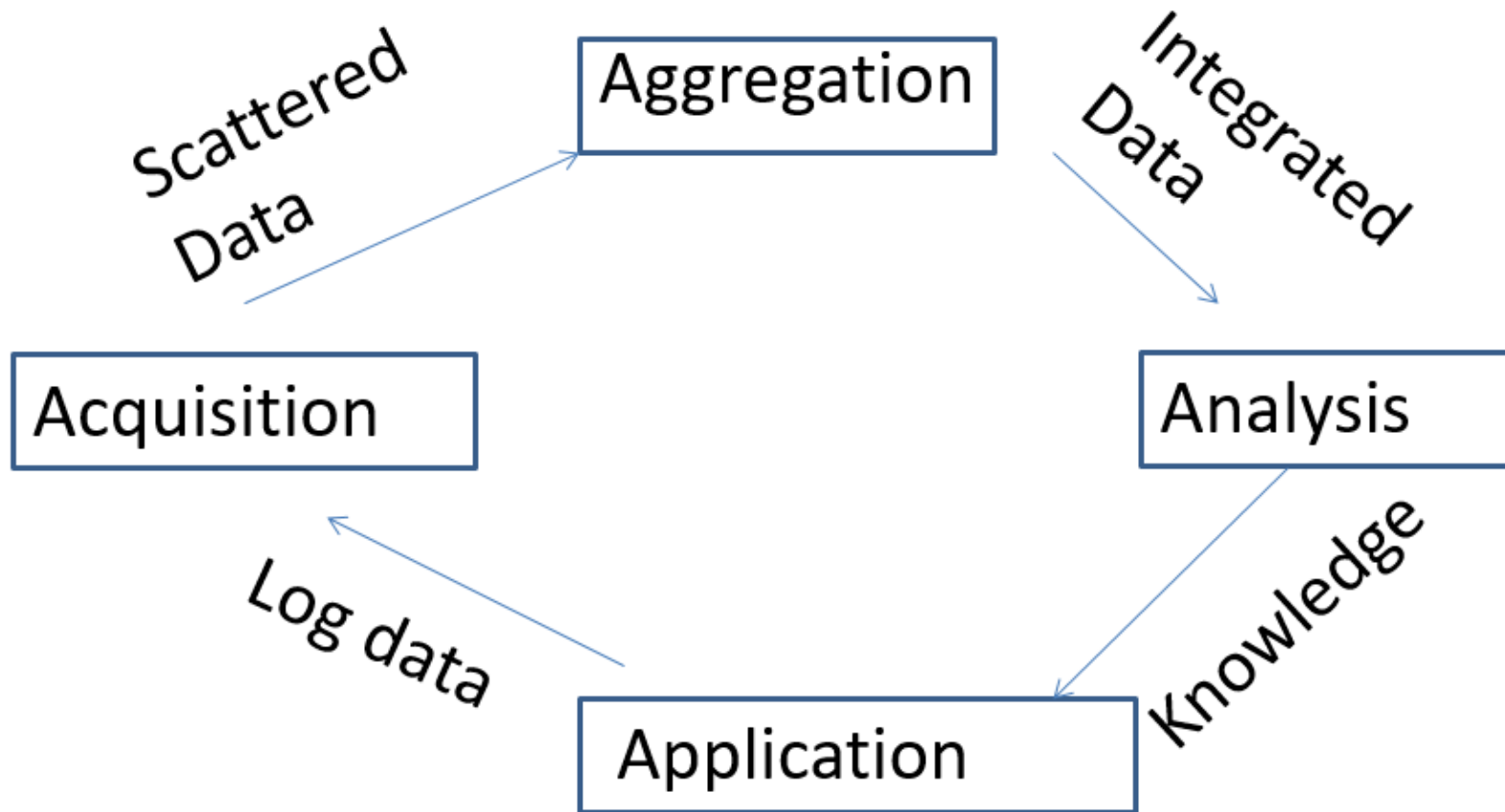
What is “big data”?

- ▶ “Big Data are **high-volume**, **high-velocity**, and/or **high-variety** information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization” (Gartner 2012)
- ▶ Bottom line: Any data that exceeds our current capability of processing can be regarded as “big”
 - ▶ Complicated (intelligent) analysis of data may make a small data “appear” to be “big”

Why is “big data” a “big deal”?

- ▶ **Government**
- ▶ **Private Sector**
 - ▶ Walmart handles more than 1 million customer transactions every hour, which is imported into databases estimated to contain more than 2.5 petabytes of data
 - ▶ Facebook handles 40 billion photos from its user base
 - ▶ Falcon Credit Card Fraud Detection System protects 2.1 billion active accounts world-wide
- ▶ **Science**
 - ▶ Large Synoptic Survey Telescope will generate 140 Terabyte of data every 5 days
 - ▶ Biomedical computation like decoding human Genome and personalized medicine

Lifecycle of Data: 4 “A”s



Simplified database system environment

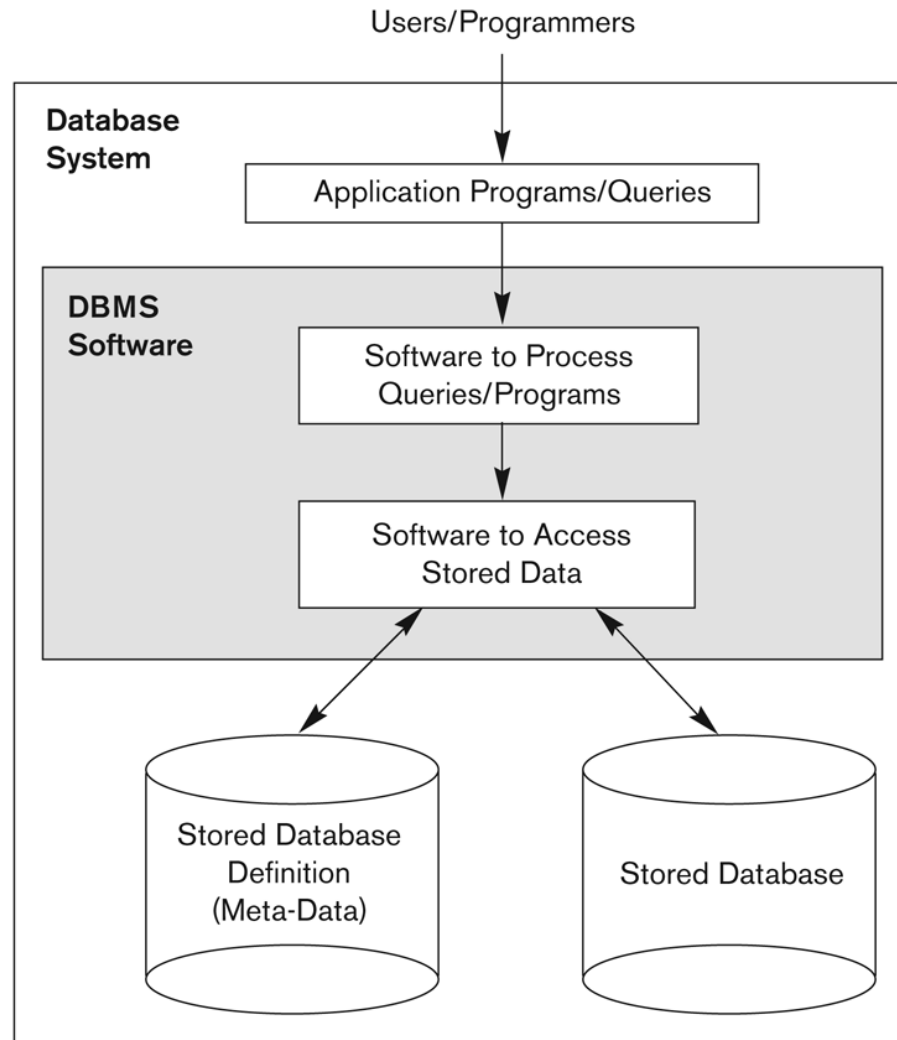


Figure 1.1
A simplified database
system environment.

What a DBMS Facilitates

- ▶ *Define* a particular database in terms of its data types, structures, and constraints
- ▶ *Construct* or load the initial database contents on a secondary storage medium
- ▶ *Manipulating* the database:
 - ▶ Retrieval: Querying, generating reports
 - ▶ Modification: Insertions, deletions and updates to its content
 - ▶ Accessing the database through Web applications
- ▶ *Processing and sharing* by a set of concurrent users and application programs - yet, keeping all data valid and consistent

Other DBMS Functionalities

- ▶ DBMS may additionally provide:
 - ▶ Protection or Security measures to prevent unauthorized access
 - ▶ “Active” processing to take internal actions on data
 - ▶ Presentation and visualization of data
 - ▶ Maintenance of the database and associated programs over the lifetime of the database application

Drawbacks of using file systems to store data

- ▶ Data redundancy and inconsistency
 - ▶ Multiple file formats, duplication of information in different files
- ▶ Difficulty in accessing data
 - ▶ Need to write a new program to carry out each new task
- ▶ Data isolation
 - ▶ Multiple files and formats
- ▶ Integrity problems
 - ▶ Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
 - ▶ Hard to add new constraints or change existing ones

Drawbacks of using file systems to store data (Contd.)

- ▶ Atomicity of updates
 - ▶ Failures may leave database in an inconsistent state with partial updates carried out
 - ▶ Example: Transfer of funds from one account to another should either complete or not happen at all
- ▶ Concurrent access by multiple users
 - ▶ Concurrent access needed for performance
 - ▶ Uncontrolled concurrent accesses can lead to inconsistencies
 - ▶ Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- ▶ Security problems
 - ▶ Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

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

Example of a Database(with a Conceptual Data Model)

- ▶ 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 entities and relationships are typically expressed in a conceptual data model, such as the ENTITY-RELATIONSHIP data model

Example of a Simple Database

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.

Main Characteristics of the Database Approach

- ▶ **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.
- ▶ **Insulation between programs and data:**
 - ▶ Called **program-data independence**.
 - ▶ Allows changing data structures and storage organization without having to change the DBMS access programs
 - ▶ E.g., ADTs

Example of a Simplified Database Catalog

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 alpha characters followed by four digits

Figure 1.3

An example of a database catalog for the database in Figure 1.2.

Main Characteristics of the Database Approach (Contd.)

- ▶ **Data Abstraction:**

- ▶ 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.

Main Characteristics of the Database Approach (Contd.)

- ▶ **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.

Database 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

► Actors on the scene

► 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.

► 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.

Database End Users

- ▶ **Actors on the scene (continued)**
 - ▶ **End-users:** They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
 - ▶ **Casual:** access database occasionally when needed
 - ▶ **Naïve or Parametric:** they make up a large section of the end-user population.
 - ▶ They use previously well-defined functions in the form of “canned transactions” against the database.
 - ▶ Users of Mobile Apps mostly fall in this category
 - ▶ Bank-tellers or reservation clerks are parametric users who do this activity for an entire shift of operations.
 - ▶ Social Media Users post and read information from websites

Database End Users (Contd.)

► **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 the user of a tax program that creates its own internal database.
- Another example is a user that maintains a database of personal photos and videos.

Database Users - Actors on the Scene (Contd.)

► System Analysts and Application Developers

This category currently accounts for a very large proportion of the IT work force.

- **System Analysts:** They understand the user requirements of naïve and sophisticated users and design applications including canned transactions to meet those requirements.
- **Application Programmers:** Implement the specifications developed by analysts and test and debug them before deployment.
- **Business Analysts:** There is an increasing need for such people who can analyze vast amounts of business data and real-time data (“Big Data”) for better decision making related to planning, advertising, marketing etc.

Database Users - Actors behind the Scene

- ▶ **System Designers and Implementors:** Design and implement DBMS packages in the form of modules and interfaces and test and debug them. The DBMS must interface with applications, language compilers, operating system components, etc.
- ▶ **Tool Developers:** Design and implement software systems called tools for modeling and designing databases, performance monitoring, prototyping, test data generation, user interface creation, simulation etc. that facilitate building of applications and allow using database effectively.
- ▶ **Operators and Maintenance Personnel:** They manage the actual running and maintenance of the database system hardware and software environment.

Advantages of Using the Database Approach

- ▶ Controlling redundancy in data storage and in development and maintenance efforts.
 - ▶ Sharing of data among multiple users.
- ▶ Restricting unauthorized access to data. Only the DBA staff uses privileged commands and facilities.
- ▶ Providing persistent storage for program Objects
 - ▶ E.g., Object-oriented DBMSs make program objects persistent
- ▶ Providing storage structures (e.g. indexes) for efficient query processing

Advantages of Using the Database Approach (Contd.)

- ▶ Providing optimization of queries for efficient processing
- ▶ Providing backup and recovery services
- ▶ Providing multiple interfaces to different classes of users
- ▶ Representing complex relationships among data
- ▶ Enforcing integrity constraints on the database
- ▶ Drawing inferences and actions from the stored data using deductive and active rules and triggers

Additional Implications of Using 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.

Levels of Abstraction

- ▶ **Physical level:** describes how a record (e.g., instructor) is stored.
- ▶ **Logical level:** describes data stored in database, and the relationships among the data.

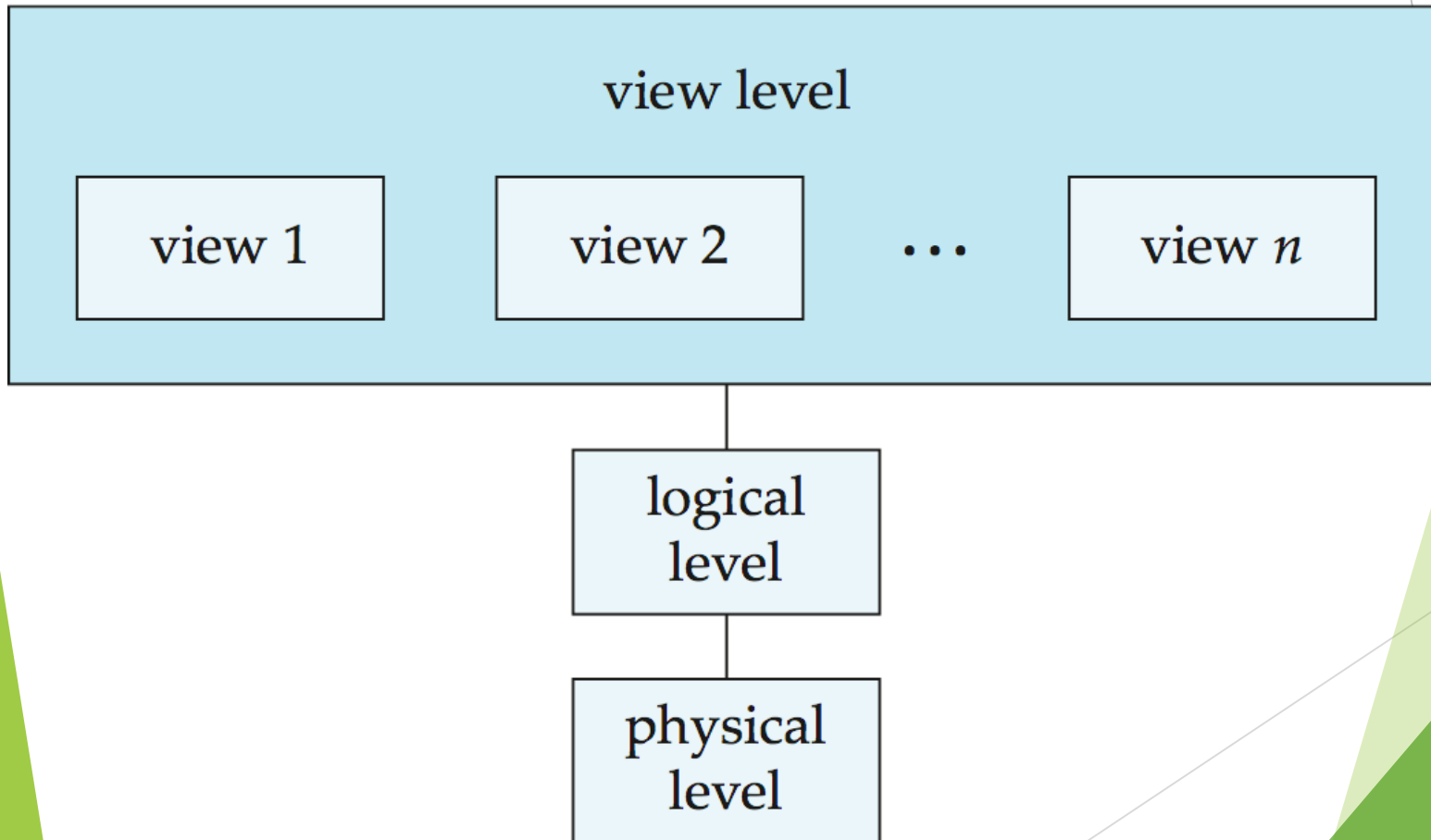
```
type instructor = record
```

```
    ID : string;  
    name : string;  
    dept_name : string;  
    salary : integer;  
end;
```

- ▶ **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



Instances and Schemas

- ▶ Similar to types and variables in programming languages
- ▶ **Logical Schema** - the overall logical structure of the database
 - ▶ Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
 - ▶ Analogous to type information of a variable in a program
- ▶ **Physical schema** - the overall physical structure of the database

Instances and Schemas (Contd.)

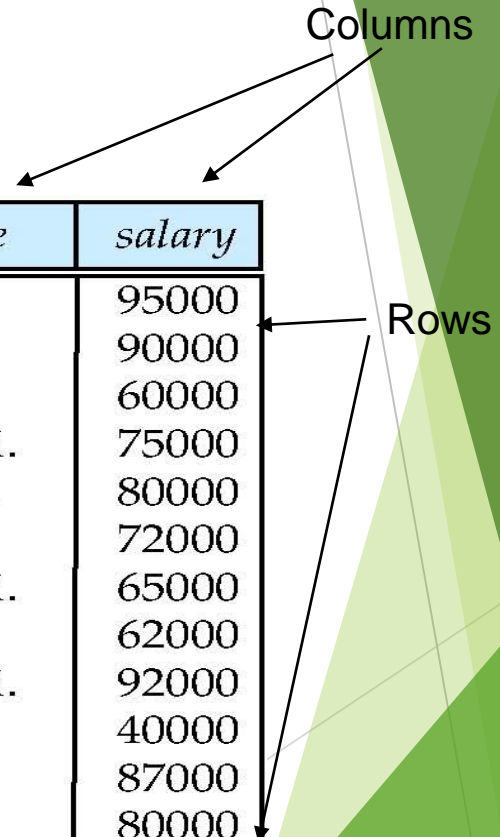
- ▶ **Instance** - the actual content of the database at a particular point in time
 - ▶ Analogous to the value of a variable
- ▶ **Physical Data Independence** - the ability to modify the physical schema without changing the logical schema
 - ▶ Applications depend on the logical schema
 - ▶ In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Data Models

- ▶ A collection of tools for describing
 - ▶ Data
 - ▶ Data relationships
 - ▶ Data semantics
 - ▶ Data constraints
- ▶ Relational model
- ▶ Entity-Relationship data model (mainly for database design)
- ▶ Object-based data models (Object-oriented and Object-relational)
- ▶ Semi-structured data model (XML)
- ▶ Other older models:
 - ▶ Network model
 - ▶ Hierarchical model

Relational Model

- ▶ All the data is stored in various tables.
- ▶ Example of tabular data in the relational model



The diagram illustrates the structure of the table. Two arrows labeled 'Columns' point to the header row, which contains the column names: *ID*, *name*, *dept_name*, and *salary*. Another set of arrows labeled 'Rows' points to the data rows, which contain individual records of instructors.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

A Sample Relational Database

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Data Definition Language (DDL)

- ▶ Specification notation for defining the database schema

Example: **create table** *instructor* (
 ID char(5),
 name varchar(20),
 dept_name varchar(20),
 salary numeric(8,2))

- ▶ DDL compiler generates a set of table templates stored in a **data dictionary**
- ▶ Data dictionary contains metadata (i.e., data about data)
 - ▶ Database schema
 - ▶ Integrity constraints
 - ▶ Primary key (ID uniquely identifies instructors)
 - ▶ Authorization
 - ▶ Who can access what

Data Manipulation Language (DML)

- ▶ Language for accessing and manipulating the data organized by the appropriate data model
 - ▶ DML also known as query language
- ▶ Two classes of languages
 - ▶ **Pure** - used for proving properties about computational power and for optimization
 - ▶ Relational Algebra
 - ▶ Tuple relational calculus
 - ▶ Domain relational calculus
 - ▶ **Commercial** - used in commercial systems
 - ▶ SQL is the most widely used commercial language

SQL

- ▶ The most widely used commercial language
- ▶ SQL is NOT a Turing machine equivalent language
- ▶ SQL is NOT a Turing machine equivalent language
- ▶ To be able to compute complex functions SQL is usually embedded in some higher-level language
- ▶ Application programs generally access databases through one of
 - ▶ Language extensions to allow embedded SQL
 - ▶ Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Database Design

The process of designing the general structure of the database:

- ▶ **Logical Design** - Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
 - ▶ **Business decision** - What attributes should we record in the database?
 - ▶ **Computer Science decision** - What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- ▶ **Physical Design** - Deciding on the physical layout of the database

Database Design (Cont.)

Is there any problem with this relation?

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Design Approaches

- ▶ Need to come up with a methodology to ensure that each of the relations in the database is “good”
- ▶ Two ways of doing so:
 - ▶ Entity Relationship Model
 - ▶ Models an enterprise as a collection of *entities* and *relationships*
 - ▶ Represented diagrammatically by an *entity-relationship diagram*:
 - ▶ Normalization Theory
 - ▶ Formalize what designs are bad, and test for them

Object-Relational Data Models

- ▶ Relational model: flat, “atomic” values
- ▶ Object Relational Data Models
 - ▶ Extend the relational data model by including object orientation and constructs to deal with added data types.
 - ▶ Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
 - ▶ Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
 - ▶ Provide upward compatibility with existing relational languages.

XML: Extensible Markup Language

- ▶ Defined by the WWW Consortium (W3C)
- ▶ Originally intended as a document markup language not a database language
- ▶ The ability to specify new tags, and to create nested tag structures made XML a great way to exchange **data**, not just documents
- ▶ XML has become the basis for all new generation data interchange formats.
- ▶ A wide variety of tools is available for parsing, browsing and querying XML documents/data

Database Engine

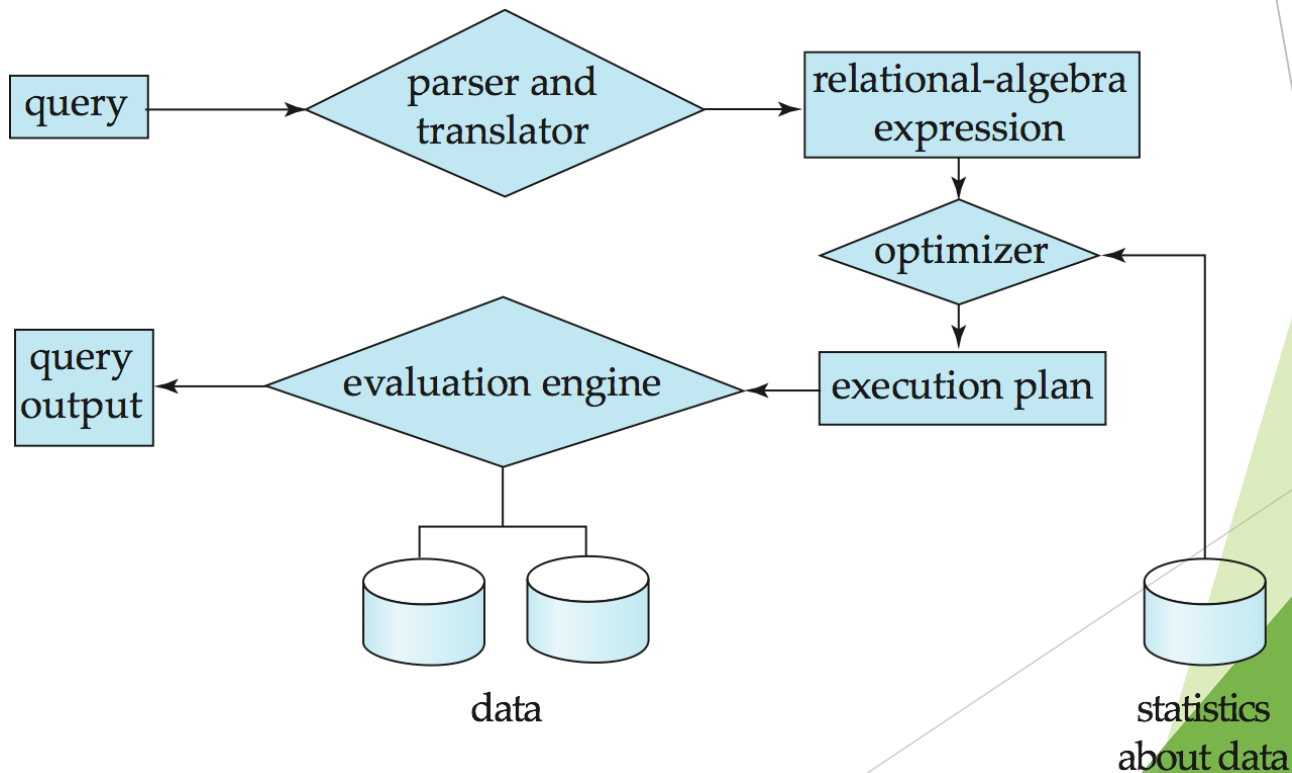
- ▶ Storage manager
- ▶ Query processing
- ▶ Transaction manager

Storage Management

- ▶ **Storage manager** is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- ▶ The storage manager is responsible to the following tasks:
 - ▶ Interaction with the OS file manager
 - ▶ Efficient storing, retrieving and updating of data
- ▶ Issues:
 - ▶ Storage access
 - ▶ File organization
 - ▶ Indexing and hashing

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



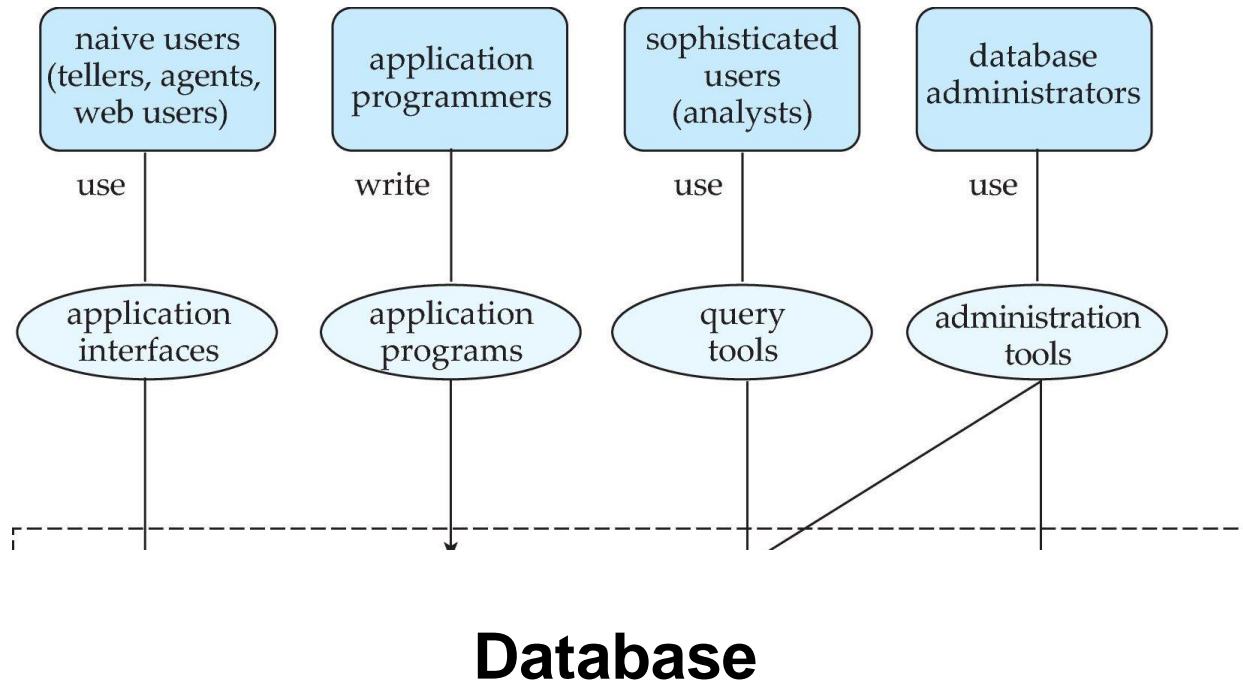
Query Processing (Cont.)

- ▶ Alternative ways of evaluating a given query
 - ▶ Equivalent expressions
 - ▶ Different algorithms for each operation
- ▶ Cost difference between a good and a bad way of evaluating a query can be enormous
- ▶ Need to estimate the cost of operations
 - ▶ Depends critically on statistical information about relations which the database must maintain
 - ▶ Need to estimate statistics for intermediate results to compute cost of complex expressions

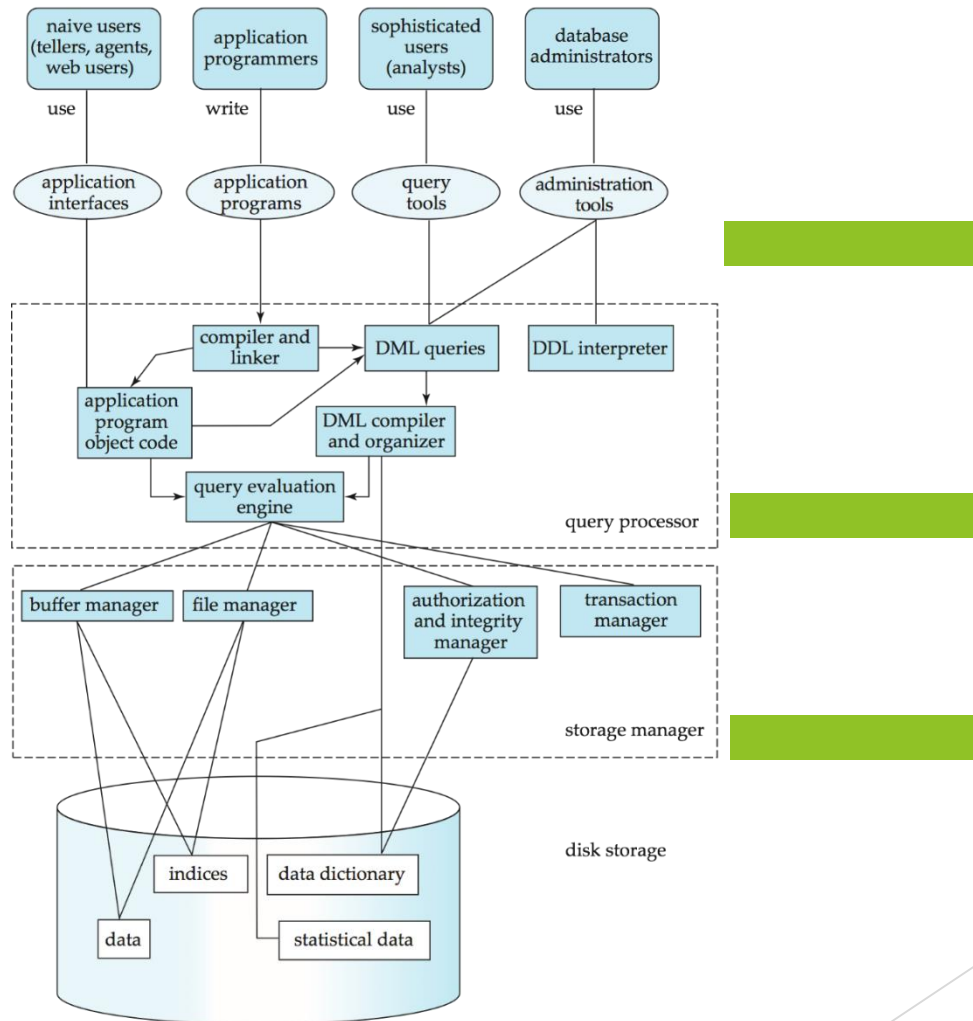
Transaction Management

- ▶ What if the system fails?
- ▶ What if more than one user is concurrently updating the same data?
- ▶ A **transaction** is a collection of operations that performs a single logical function in a database application
- ▶ **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- ▶ **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Database Users and Administrators



Database System Internals



Database Architecture

The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:

- ▶ Centralized
- ▶ Client-server
- ▶ Parallel (multi-processor)
- ▶ Distributed

History of Database Systems

- ▶ 1950s and early 1960s:
 - ▶ Data processing using magnetic tapes for storage
 - ▶ Tapes provided only sequential access
 - ▶ Punched cards for input
- ▶ Late 1960s and 1970s:
 - ▶ Hard disks allowed direct access to data
 - ▶ Network and hierarchical data models in widespread use
 - ▶ Ted Codd defines the relational data model
 - ▶ Would win the ACM Turing Award for this work
 - ▶ IBM Research begins System R prototype
 - ▶ UC Berkeley begins Ingres prototype
 - ▶ High-performance (for the era) transaction processing

History (Cont.)

- ▶ 1980s:
 - ▶ Research relational prototypes evolve into commercial systems
 - ▶ SQL becomes industrial standard
 - ▶ Parallel and distributed database systems
 - ▶ Object-oriented database systems
- ▶ 1990s:
 - ▶ Large decision support and data-mining applications
 - ▶ Large multi-terabyte data warehouses
 - ▶ Emergence of Web commerce
- ▶ Early 2000s:
 - ▶ XML and XQuery standards
 - ▶ Automated database administration
- ▶ Later 2000s:
 - ▶ Giant data storage systems
 - ▶ Google BigTable, Yahoo PNuts, Amazon, ..

When not 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, 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 access to data by multiple users is not required
- ▶ When a DBMS may be infeasible
 - ▶ In embedded systems where a general purpose DBMS may not fit in available storage

When not to use a DBMS

- ▶ When no DBMS may suffice:
 - ▶ If there are stringent real-time requirements that may not be met because of DBMS overhead (e.g., telephone switching systems)
 - ▶ If the database system is not able to handle the complexity of data because of modeling limitations (e.g., in complex genome and protein databases)
 - ▶ If the database users need special operations not supported by the DBMS (e.g., GIS and location based services).

Chapter Summary (1 / 2)

- ▶ Basic Definitions
- ▶ Impact of Databases and Database Technology
- ▶ Types of Databases and Database Applications
- ▶ Typical DBMS Functionality
- ▶ Drawbacks of using file systems to store data
- ▶ Example of a Database (UNIVERSITY)
- ▶ Main Characteristics of the Database Approach
- ▶ Types of Database Users
- ▶ Advantages of Using the Database Approach

Chapter Summary (1 / 2)

- ▶ Additional Implications of Using the Database Approach
- ▶ Instances and Schemas
- ▶ Data Models
- ▶ Storage, Query processing and transaction management
- ▶ History of Database Technology
- ▶ When Not to Use Databases