CHAPTER 1

Databases and Database Users

CA500202--Database Management System and SQL

- Module I Database, need for DBMS, users, DBMS architecture, data models, views of data, data independence, database languages, Relational Model-Basic concepts, keys, integrity constraints, ER model-basic concepts, ER diagram, weak entity set, ER to Relational, relationships, generalization, aggregation, specialization
- Module II Codd's rules, Relational model concepts , Relational algebra- Select, Project, Join, Relational calculus-tuple relational calculus and domain relational calculus, Specifying constraints management systems, Anomalies in a database, Functional dependencies, NormalizationFirst, Second, Third, Boyce Codd normal forms, multi-valued dependency and Fourth normal form, Join dependency and Fifth normal form. Relational database query languages-Basics of SQL, Data definition in SQL- Data types, Creation, Insertion, Viewing, Updation, Deletion of tables, Modifying the structure of the tables, Renaming, Dropping of tables, Data constraints-I/O constraints, ALTER TABLE command.
- Module III Database manipulation in SQL- Computations done on the table- Select command, Logical operators, Range searching, Pattern matching, Grouping data from tables in SQL, GROUP BY, HAVING clauses, Joins-Joining multiple tables, Joining tables to itself, DELETE, UPDATE, Views-Creation, Renaming the column of a view, Destroys view- Program with SQL, Security-locks, Types of locks, Levels of locks, Cursors - working with cursors, error handling, Developing stored procedures,- Creation, Statement blocks, Conditional execution, Repeated execution, Cursor-based repetition, Handling Error conditions, Implementing triggers, Creating triggers, Multiple trigger interaction.

CA500202--Database Management System and SQL

- Module IV Concept of transaction, ACID properties, serializability, states of transaction, Concurrency control, Locking techniques, Time stamp based protocols, Granularity of data items, Deadlock, Failure classifications, storage structure, Recovery & atomicity, Log base recovery, Recovery with concurrent transactions, Database backup & recovery, Remote Backup System, Database security issues
- Module V Object Oriented Database Management Systems (OODBMS) - concepts, need for OODBMS, composite objects, issues in OODBMSs, advantages and disadvantages of OODBMS. Distributed databases - motivation - distributed database concepts, types of distribution, architecture of distributed databases, the design of distributed databases, distributed transactions, commit protocols for distributed databases
- Reference
- Text 1. Elmasri and Navathe, Fundamentals of Database Systems, 5th Edition, Pearson
- 2. Abraham Silbersehatz, Henry F. Korth and S.Sudarshan, Database System Concepts, 6 th Edition, Tata McGraw-Hill.
- 3.JamesR.Groff and Paul N. Weinberg The complte reference SQL Second edition,Tata McGraw Hil

What is data, database?

- Database –a collection of related data: a highly organized, interrelated, and structured set of data about a particular enterprise
  - Controlled by a database management system (DBMS)
- Data: Raw and Known facts that can be recorded and have an implicit meaning

Impact of Databases and Database Technology

- Databases and database technology are having a major impact on the growing use of computers.
- 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

Example of a database

- Consider the names, telephone numbers, and addresses of the people.
- You may have recorded this data in an indexed address book, or you may have stored it on a diskette, using a personal computer and software such as EXCEL. This is a collection of related data with an implicit meaning and hence is a database.

Implicit properties of a database

- 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.

Manual or Computerized database

- A database may be generated and maintained manually or it may be computerized.
- The library card catalog is an example of a database that may be created and maintained manually.
- A computerized database may be created and maintained either by a group of application programs written specifically for that task or by a database management system.

What a DBMS Facilitates

- A database management system (DBMS) is a collection of programs that enables users to create and maintain a database.
- *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

What is DBMS

- DBMS
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database systems are used to manage collections of data that are:
  - Highly valuable
  - Relatively large
  - Accessed by multiple users and applications, often at the same time.
- A modern database system is a complex software system whose task is to manage a large, complex collection of data.
- Databases touch all aspects of our lives

Data , Database and DBMS

- A database system is basically a computer based record keeping system. •
- Data Anything can be a data (raw facts and figures) • Information
- Processed Data (meaningful data) •
- Database : An organized, persistent collection of logically related data •
- Database System : A computer based record keeping system: whose overall purpose is to record and maintain information. It manages the database of an enterprise

Example-University database

- UNIVERSITY database for maintaining information concerning students, courses, and grades in a university environment.

The database is organized as five files,

- STUDENT file stores data on each student;
- COURSE file stores data on each course;
- SECTION file stores data on each section of a course;
- GRADE_REPORT file stores the grades that students receive in the various sections they have completed;
- PREREQUISITE file stores the prerequisites of each course.

To define University database

- Specify the structure of the records of each file by specifying the different types of data elements to be stored in each record.
- STUDENT record includes data to represent the student's Name, StudentNumber, Class and Major (MATH, computer science or CS)
- COURSE record includes data to represent the CourseName, CourseNumber, CreditHours, and Department (the department that offers the course);
- Specify a data type for each data element within a record.
- For example, Name of STUDENT is a string of alphabetic characters, StudentNumber of STUDENT is an integer, and Grade of GRADE_REPORT is a single character from the set {A, B, C, D, F, I}.

To construct the UNIVERSITY database

- To construct the UNIVERSITY database, we store data to represent each student, course, section, grade report, and prerequisite as a record in the appropriate file.
- Records in the various files may be related. For example, the record for "Smith" in the STUDENT file is related to two records in the GRADE_REPORT file that specify Smith's grades in two sections

Database manipulation

- Database manipulation involves querying and updating.
- Examples of queries are
- "retrieve the list of all courses and grades—of Smith";
- "list the names of students who took the section of the Database course offered in 1999 and their grades in that section";
- "what are the prerequisites of the Database course?“
- Examples of updates are
- "change the class of Smith to Sophomore"; "

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.

The relational model

Columns

ID	*name*	*dept_name*	*salary*
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

Rows

E.F. “Ted” Codd

Traditional File based system

- The File Based System File based systems are an early attempt to computerise the manual filing system.
- This works well when the number of items to be stored is small or the process is only needed to store and retrieve information.

Limitations of File Based System

- Separation and Isolation of Data : When the data is stored in separate files it becomes difficult to access.
- Duplication of Data :Decentralised approach leads to uncontrolled duplication of data. .=> wastage of storage space, time and money.
- Inconsistent Data: Decentralised approach leads to uncontrolled duplication of data. .=> wastage of storage space, time and mone

Example of Redundancy

- Programs to print a student's transcript and to enter new grades into the file are implemented.
- A second user, the accounting office, may keep track of students' fees and their payments.
- Although both users are interested in data about students, each user maintains separate files—and programs to manipulate these files—because each requires some data not available from the other user's files.
- This redundancy in defining and storing data results in wasted storage space and in redundant efforts to maintain common data up-to-date.
- In the database approach, a single repository of data is maintained that is defined once and then is accessed by various users.

Limitations of File Based System

- Data Dependence : The physical structure and storage of data files and records are defined in the application code. => program data dependence.
- Incompatible File Formats : The structure of the file embedded in application program totally dependent on the programming language. This incompatibility makes them difficult to process jointly.
- Fixed Queries : Query or reports needed by the organization has to be developed by the application programmer.

Main Characteristics of the Database Approach

- Self-Describing Nature of a Database System
- Insulation between Programs and Data, and Data Abstraction
- Support of Multiple Views of the Data
- Sharing of Data and Multiuser Transaction Processing

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 stored in the catalog is called **meta-data***.
  - The catalog is used by the DBMS software and also by database users who need information about the database structure
  - This allows the DBMS software to work with different database applications.-for example, **a university database, a banking database, or a company database**

Self-describing nature of a database system: -comparison with traditional file processing

- In traditional file processing, data definition is typically part of the application programs themselves.
- Hence, these programs are constrained to work with only one specific database, whose structure is declared in the application programs.
- For example, a PASCAL program may have record structures declared in it;
- a C++ program may have "struct" or "class" declarations;
- a COBOL program has Data Division statements to define its files.
- Whereas file-processing software can access only specific databases, DBMS software can access diverse databases by extracting the database definitions from the catalog and then using these definitions

Main Characteristics of the Database Approach

- **Insulation between programs and data :**
  - Traditional file processing stores the structure of data files in the access programs→ any change in structure of files result in change of programs
  - DBMS seperates data from programs --Called **program-data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.

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 enumerared 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.

Example of a traditional file vs DBMS

- For example, a file access program may be written in such a way that it can access only STUDENT records of the structure
- If we want to add another piece of data to each STUDENT record, say the Birthdate, such a program will no longer work and must be changed.
- In a DBMS environment, we just need to change the description of STUDENT records in the catalog to reflect the inclusion of the new data item Birthdate; no programs are changed.
- The next time a DBMS program refers to the catalog, the new structure of STUDENT records will be accessed and used.

Main Characteristics of the Database Approach (continued)

- **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 how the data is stored or how the operations are implemented.
  - Data model is a type of data abstraction that is used to provide this conceptual representation.
  - Eg:Object oriented model

Support of multiple views of the data:

- A database typically has many users, 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.
- Some users may not need to be aware of whether the data they refer to is stored or derived.
- A multiuser DBMS whose users have a variety of applications must provide facilities for defining multiple views.
- For example, one user of the database may be interested only in the transcript of each student;
- A second user, who is interested only in checking that students have taken all the prerequisites of each course they register.

Main Characteristics of the Database Approach (continued)

- **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
  - Example :**when several reservation clerks try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one clerk at a time for assignment to a passenger**

Main Characteristics of the Database Approach (continued)

- **Sharing of data and multi-user transaction processing:**
  - *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; allows hundreds of concurrent transactions to execute per second.

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

Recent Developments

- 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

- New technologies are emerging from the so-called non-SQL, non-database software vendors to manage vast amounts of data generated on the web:
  - **Big data** storage systems involving large clusters of distributed computers (Chapter 25)
  - **NOSQL** (Non-SQL, Not Only SQL) systems (Chapter 24)
- A large amount of data now resides on the “cloud” which means it is in huge data centers using thousands of machines.

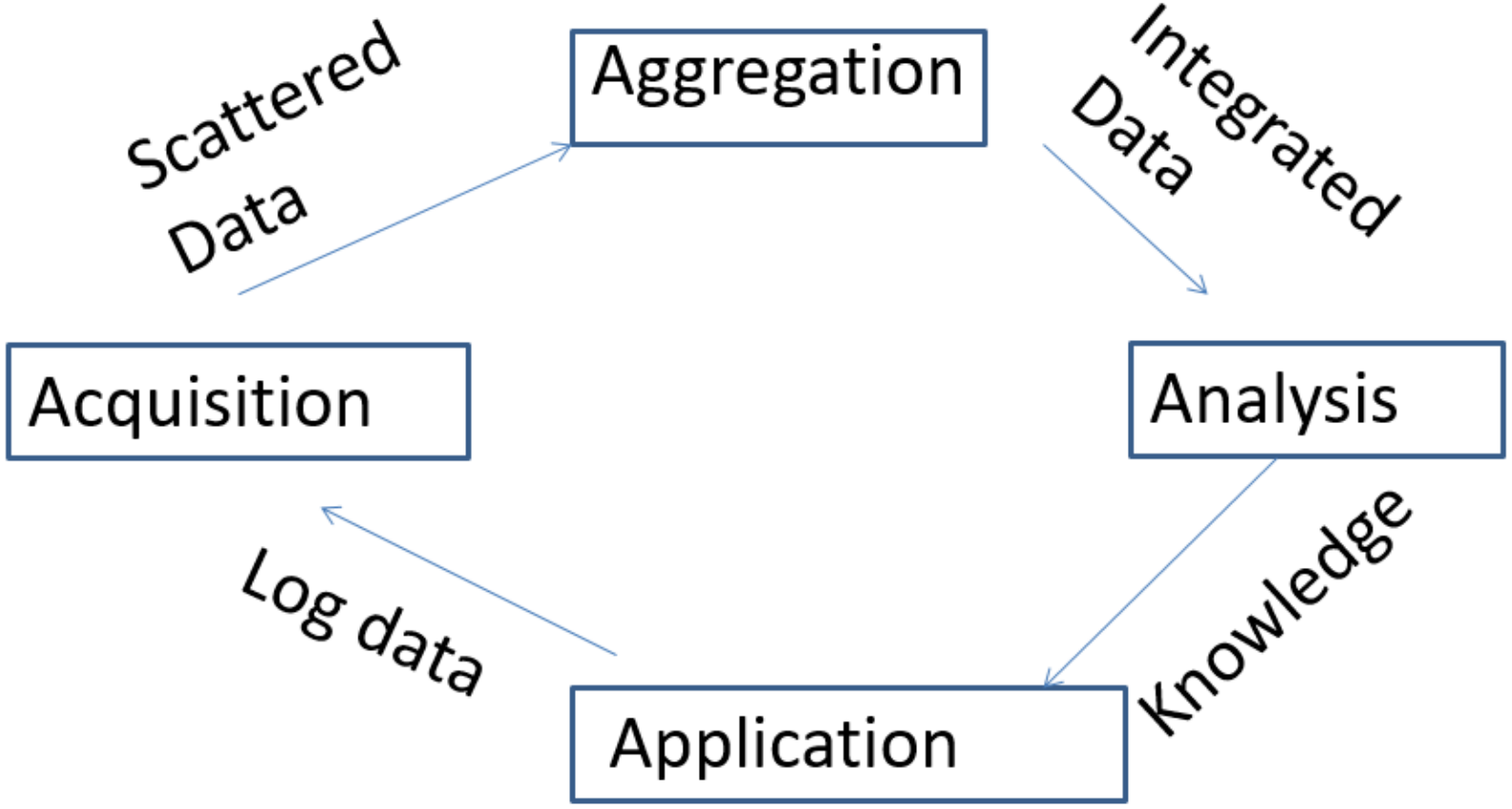
What is “big data”?

- "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)
  - Three Vs? Other Vs?
    - Veracity: refers to the trustworthiness of the data
    - Value: will data lead to the discovery of a critical causal effect?
- 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”

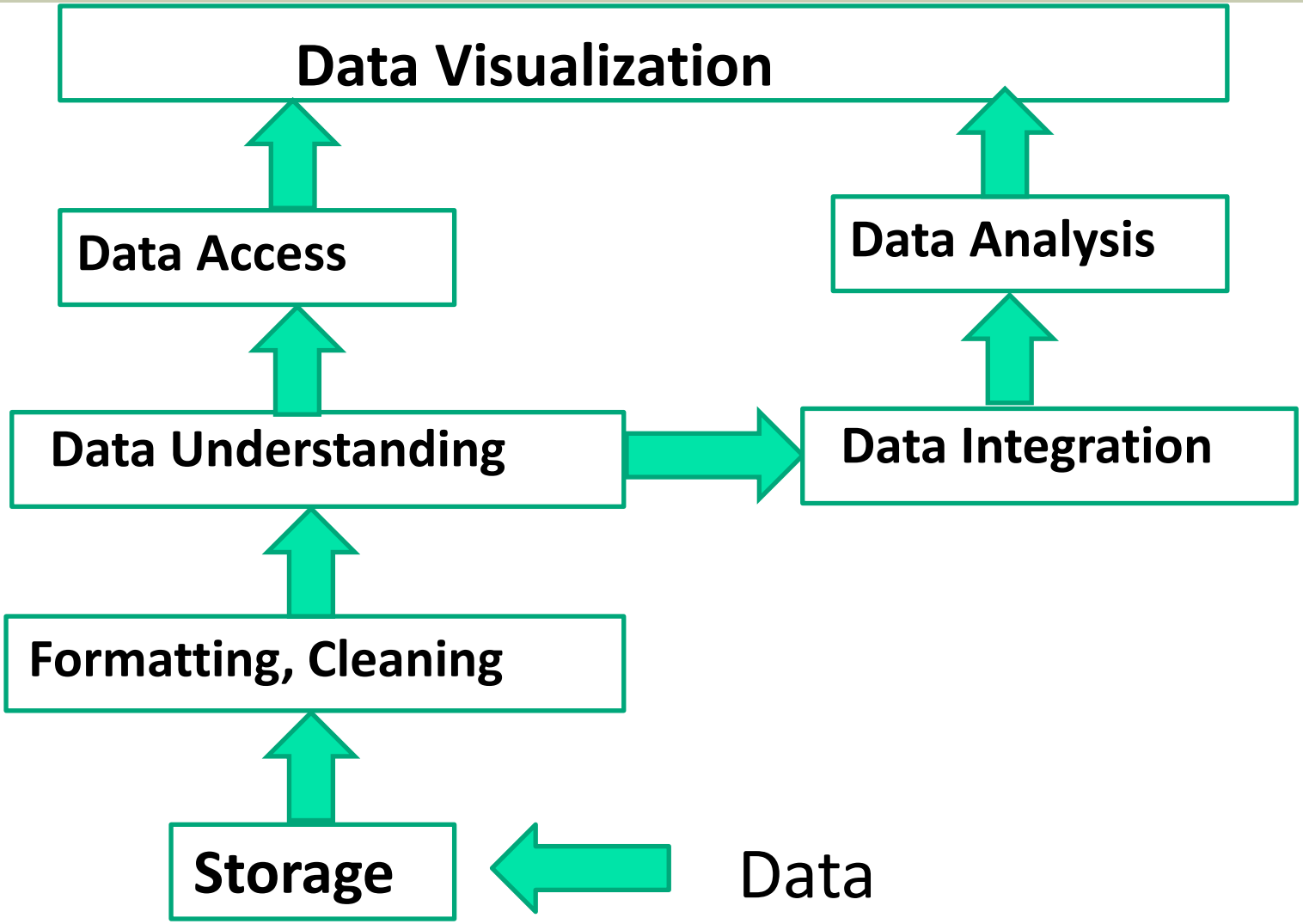
Why is “big data” a “big deal”?

- 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

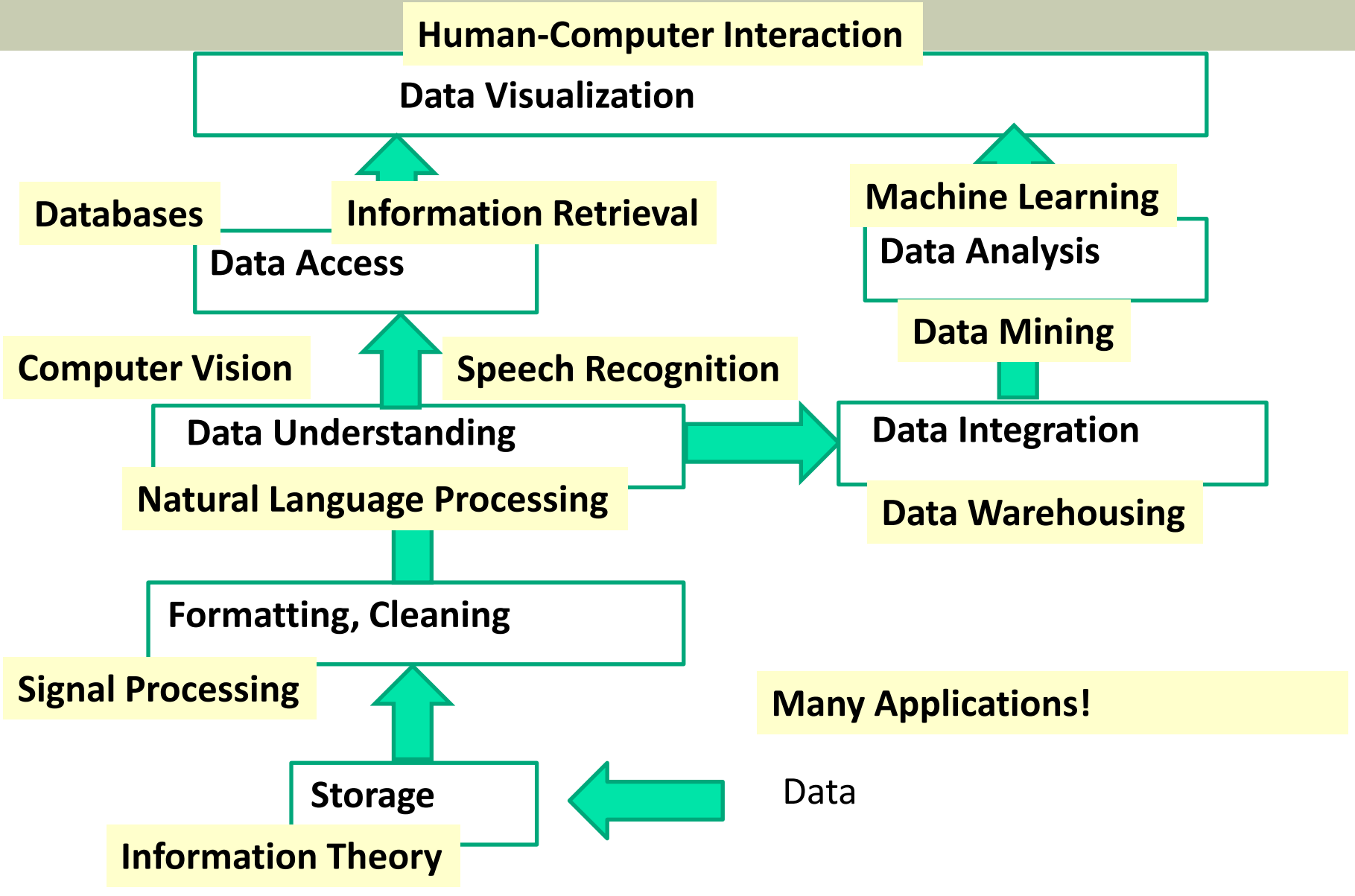
Lifecycle of Data: 4 “A”s



Computational View of Big Data



Big Data & Related Disciplines



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 (continued)

- **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 (continued)

- System analysts and application developers
  - 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 DBMS

- Controlling Redundancy
- Restricting Unauthorized Access
- Providing Persistent Storage for Program Objects and Data Structures
- Permitting Inferencing and Actions Using Rules
- Providing Multiple User Interfaces
- Representing Complex Relationships Among Data
- Enforcing Integrity Constraints
- Providing Backup and Recovery

Controlling Redundancy

- In traditional software development utilizing file processing, every user group maintains its own files for handling its data-processing applications.
- For example, consider the UNIVERSITY database example, two groups of users might be the course registration personnel and the accounting office. In the traditional approach, each group independently keeps files on students. The accounting office also keeps data on registration and related billing information, whereas the registration office keeps track of student courses and grades.
- Much of the data is stored twice: once in the files of each user group. Additional user groups may further duplicate some or all of the same data in their own files.

Controlling Redundancy

- This redundancy in storing the same data multiple times leads to several problems.
- First, there is the need to perform a single logical update—such as entering data on a new student—multiple times: once for each file where student data is recorded. This leads to duplication of effort.
- Second, storage space is wasted when the same data is stored repeatedly, and this problem may be serious for large databases.
- Third, files that represent the same data may become inconsistent- because an update is applied to some of the files but not to others.

Controlling Redundancy

- In the database approach, the views of different user groups are integrated during database design.
- For consistency, we should have a database design that stores each logical data item—such as a student's name or birth date—in only one place in the database. This does not permit inconsistency, and it saves storage space.
- In some cases, controlled redundancy may be useful for improving the performance of queries. For example, we may store StudentName and CourseNumber redundantly in a GRADE_REPORT file

Restricting Unauthorized Access

- When multiple users share a database, it is likely that some users will not be authorized to access all information in the database.
- For example, financial data is often considered confidential, and hence only authorized persons are allowed to access such data.
- Some users may be permitted only to retrieve data, whereas others are allowed both to retrieve and to update. Hence, the type of access operation—retrieval or update—must also be controlled.
- Typically, users or user groups are given account numbers protected by passwords, which they can use to gain access to the database.
- A DBMS should provide a security and authorization subsystem, which the DBA uses to create accounts and to specify account restrictions. The DBMS should then enforce these restrictions automatically.

Providing Persistent Storage for Program Objects and Data Structures

- Databases can be used to provide persistent storage for program objects and data structures. This is one of the main reasons for the emergence of the object-oriented database systems.
- Programming languages typically have complex data structures, such as class definitions in C++. The values of program variables are discarded once a program terminates, unless the programmer explicitly stores them in permanent files, which often involves converting these complex structures into a format suitable for file storage.
- When the need arises to read this data once more, the programmer must convert from the file format to the program variable structure.

Providing Persistent Storage for Program Objects and Data Structures

- Object-oriented database systems are compatible with programming languages such as C++ and JAVA, and the DBMS software automatically performs any necessary conversions. Hence, a complex object in C++ can be stored permanently in an object-oriented DBMS.
- Such an object is said to be persistent, since it survives the termination of program execution and can later be directly retrieved by another C++ program.
- The persistent storage of program objects and data structures is an important function of database systems.
- Traditional database systems often suffered from the so-called impedance mismatch problem, since the data structures provided by the DBMS were incompatible with the programming language's data structures.

Permitting Inferencing and Actions Using Rules

- Some database systems provide capabilities for defining deduction rules for inferencing new information from the stored database facts. Such systems are called **deductive database systems.**
- For example, there may be complex rules in the miniworld application for determining when a student is on probation. These can be specified declaratively as rules, which when compiled and maintained by the DBMS can determine all students on probation.
- In a traditional DBMS, an explicit procedural program code would have to be written to support such applications. But if the miniworld rules change, it is generally more convenient to change the declared deduction rules than to recode procedural programs.

Providing Multiple User Interfaces

- Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces.
- These include query languages for casual users;
- programming language interfaces for application programmers;
- forms and command codes for parametric users;
- menu-driven interfaces and natural language interfaces for stand-alone users.

Providing Multiple User Interfaces

- Both forms-style interfaces and menu-driven interfaces are commonly known as graphical user interfaces (GUIs).
- Many specialized languages and environments exist for specifying GUIs.
- Capabilities for providing World Wide Web access to a database—or web-enabling a database—are also becoming increasingly common.

Representing Complex Relationships Among Data

- A database may include numerous varieties of data that are interrelated in many ways.
- Consider the example. The record for Brown in the student file is related to four records in the GRADE_REPORT file. Similarly, each section record is related to one course record as well as to a number of GRADE_REPORT records—one for each student who completed that section.
- A DBMS must have the capability to represent a variety of complex relationships among the data as well as to retrieve and update related data easily and efficiently.

Enforcing Integrity Constraints

- Most database applications have certain integrity constraints 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 item.
- For example, we may specify that the value of the Class data item within each student record must be an integer between 1 and 5 and that the value of Name must be a string of no more than 30 alphabetic characters.
- A more complex type of constraint that occurs frequently involves specifying that a record in one file must be related to records in other files.

Enforcing Integrity Constraints

- For example, we can specify that "every section record must be related to a course record."
- Another type of constraint specifies uniqueness on data item values, such as "every course record must have a unique value for CourseNumber." These constraints are derived from the meaning or semantics of the data and of the miniworld it represents.
- It is the database designers' responsibility to identify integrity constraints during database design. Some constraints can be specified to the DBMS and automatically enforced. Other constraints may have to be checked by update programs or at the time of data entry.

Providing Backup and Recovery

- A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery.
- For example, if the computer system fails in the middle of a complex update program, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the program started executing.
- Alternatively, the recovery subsystem could ensure that the program is resumed from the point at which it was interrupted so that its full effect is recorded in the database.

Additional Implications of Using the Database Approach

- Potential for enforcing standards:
  - **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.

Additional Implications of Using the Database Approach (continued)

- 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 shopping, airline, hotel, car reservations.
- Economies of scale:
  - Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

Historical Development of Database Technology

- Early database applications:
  - The *Hierarchical* and *Network* models were introduced in mid 1960s and dominated during the seventies.
  - A bulk of the worldwide database processing still occurs using these models, particularly, the hierarchical model using IBM's IMS system.
- Relational model-based systems:
  - Relational model was originally introduced in 1970, was heavily researched and experimented within IBM Research and several universities.
  - Relational DBMS Products emerged in the early 1980s.

Historical Development of Database Technology (continued)

- Object-oriented and emerging applications:
  - Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications.
    - Their use has not taken off much
  - Many relational DBMSs have incorporated object database concepts, leading to a new category called *object-relationa*l DBMSs (ORDBMSs)
  - *Extended relational* systems add further capabilities (e.g. for multimedia data, text, XML, and other data types)

Historical Development of Database Technology (continued)

- Data on the Web and e-commerce applications:
  - Web contains data in HTML (Hypertext markup language) with links among pages
  - Has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language) (see Ch. 13).
  - Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database (see Ch. 11).
    - Also allow database updates through Web pages

Extending Database Capabilities (1)

- New functionality is being added to DBMSs in the following areas:
  - Scientific applications – physics, chemistry, biology, genetics
  - Spatial: weather, earth and atmospheric sciences and astronomy
  - XML (eXtensible Markup Language)
  - Image storage and management
  - Audio and video data management
  - Time series and historical data management
- The above gives rise to *new research and development* in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.

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).