An Overview and Introduction to Databases

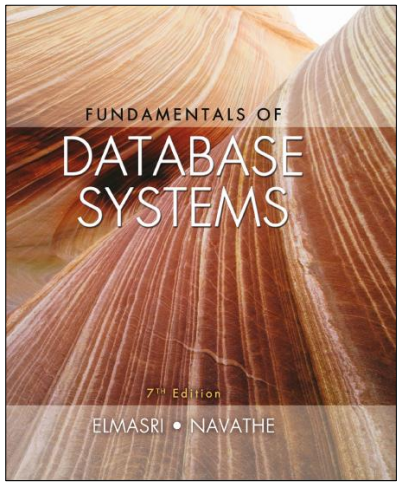
Week 1

Learning Objectives

- Big Data and Focus of this Unit
- Basic Definitions of Databases
- Typical DBMS Functionality
- Example of a Database (University)
- Main Characteristics of the Database Approach
- Types of Database Users
- Advantages of Using the Database Approach
- Data Models and Their Categories
- Schemas, Instances, and States
- Three-Schema Architecture and Data Independence
- DBMS Languages and Interfaces
- Database Architectures
- Historical Development of Database Technology

Fundamentals of Database Systems

Seventh Edition



Introduction to BigData

Chapter 1

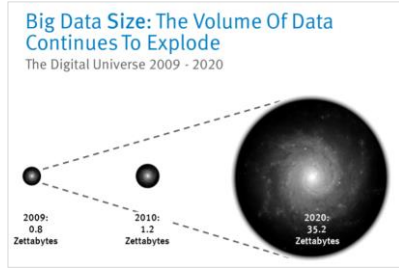
Databases and Database Users

Chapter 2

Database System Concepts and Architecture

Pearson

Big Data



- Big data represents the information assets characterized by such a high **volume**, **velocity** and **variety** (**3Vs**) to require specific technology and analytical methods for its transformation into value (De Mauro et al., A Formal Definition of Big Data based on its Essential Features, Library Review. 65 (3), 2016)
- High Volume – It is the quantity of data gathered, generated and stored.
  - 1990's measured in Terabytes (RDBMS & Data Warehouse)
  - 2000's measured in Petabytes (Content & Digital Asset Management)
  - 2010's measured in Exabytes (NoSQL & KEY/VALUE)
  - 44x increase from 2009 to 2020 (0.8 to 35.2 Zettabytes)
- High Velocity – high speed at which all this data is received (e.g., data streams) and also acted upon (real-time response).
- High Variety – different types of data: ***structured*** (tables etc.), ***semi-structured*** (XML/JSON, graph data), unstructured (text data).
- Challenges: how to capture, integrate, ***store, manage,*** and analyze big data

Pearson

Focus of this Unit

- Learn the principles and techniques of database systems – the powerful technology for ***storing*** and ***managing*** data
- Learn traditional techniques (RDBMS & SQL) for handling ***structured*** data
- Learn new techniques (NoSQL & Key/Value stores) for handling ***semi-structured*** data
- Briefly introduce techniques for handling *unstructured* data.

Pearson

5

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.
  - Some popular Relational DBMSs: Oracle, DB2, SQL Server, SYBASE, Informix, MySQL, PostgreSQL
- **Database System:**
  - The DBMS software together with the data itself. Sometimes, the applications are also included.

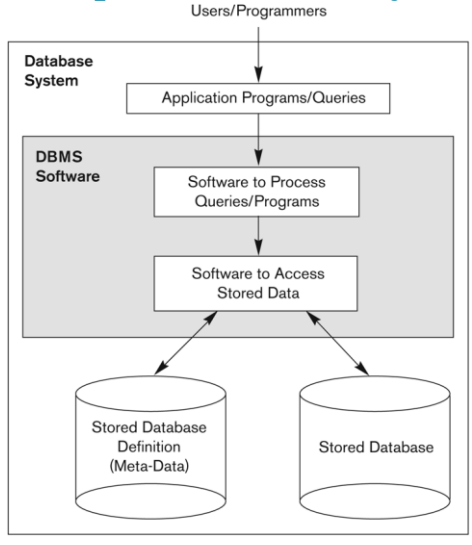
Pearson

6

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

Figure 1.1 Simplified database system environment



Typical DBMS Functionality

- **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

Application Activities Against a Database

- Applications interact with a database by generating
  - **Queries:** that access different parts of data and formulate the result of a request
  - **Transactions:** that may read some data and "update" certain values or generate new data and store that in the database
- Applications must not allow unauthorized users to access data
- Applications must keep up with changing user requirements against the database

Additional DBMS Functionality

- **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
    - Called database, software, and system maintenance

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
- **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 ER model (see Chapters 3, 4)

Figure 1.2 Example of a simple database

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

Figure 1.2 Example of a simple database (Cont'd)

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

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.

* Some newer systems such as a few NoSQL systems need no meta-data: they store the data definition within its structure making it self describing

Figure 1.3 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

Main Characteristics of the Database Approach (Cont'd)

- **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.
- **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

- **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.
- **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 on the Scene (Cont'd)

- **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, using previously well-defined functions in the form of "canned transactions" against the database, e.g., users of Mobile Apps, bank-tellers or reservation clerks, social media users (post and read information from websites)
  - **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 – Workers 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.
- Providing Storage Structures (e.g. indexes) for efficient Query Processing.
- 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.

Data Models

- **Data Model:**
  - A set of concepts to describe the **structure** of a database, the **operations** for manipulating these structures, and certain **constraints** that the database should obey.
- **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
- **Data Model Operations:**
  - 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_student_gpa, update_inventory)

Schemas Versus Instances

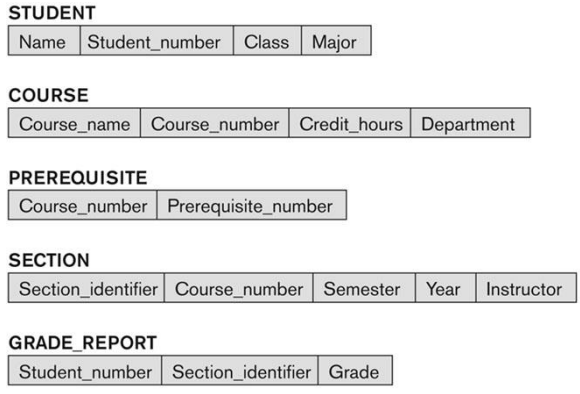
- Database Schema:
  - The **description** of a database.
  - Includes descriptions of the database structure, data types, and the constraints on the database.
- Schema Diagram:
  - An **illustrative** display of (most aspects of) a database schema.
- Schema Construct:
  - A **component** of the schema 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**

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**.

Example of a Database Schema and State

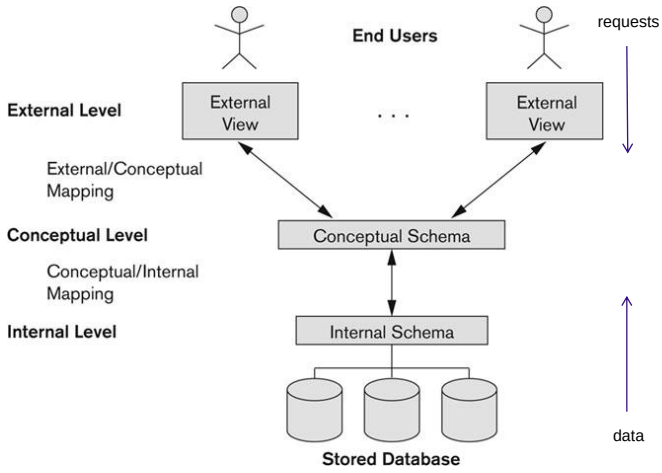
Figure 2.1 Schema diagram for the database in Figure 1.2.



Database state:

refer **Figure 1.2** - a database that stores student and course information.

Figure 2.2 The Three-Schema Architecture



Defines DBMS schemas at **three** levels:

External schemas at the external level to describe the various user views.

Conceptual schema at the conceptual level to describe the structure and constraints for the whole database for a community of users.

Internal schema at the internal level to describe physical storage structures and access paths (e.g indexes).

Data Independence

- **Logical Data Independence:**
  - The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.
- **Physical Data Independence:**
  - The capacity to change the internal schema without having to change the conceptual schema.
  - For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance
- When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence.
- The higher-level schemas themselves are **unchanged**.
  - Hence, the application programs need not be changed since they refer to the external schemas.

DBMS Languages

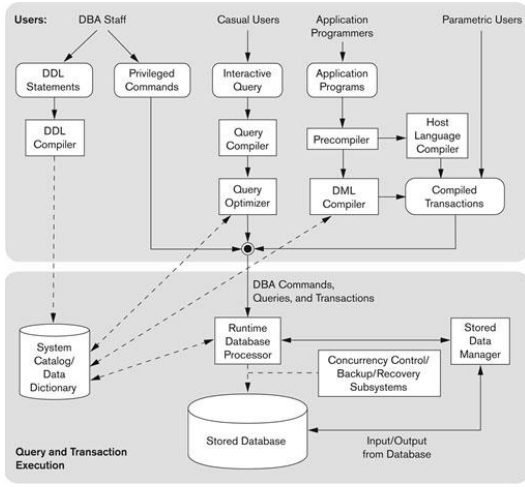
- Data Definition Language (DDL)
  - Used by the DBA and database designers to specify the conceptual schema of a database.
  - In many DBMSs, the DDL is also used to define internal and external schemas (views).
- Data Manipulation Language (DML)
  - Used to specify database retrievals and updates
  - DML commands (data sublanguage) can be **embedded** in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.
    - A library of functions can also be provided to access the DBMS from a programming language
  - Alternatively, stand-alone DML commands can be applied directly (called a **query language**).

DBMS Interfaces and Utilities

- Stand-alone query language interfaces
  - Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE)
- Programmer interfaces for embedding DML in programming languages (e.g. embedded SQL, SQLJ, JDBC, ODBC, PL/SQL)
- User-friendly interfaces
  - Menu-based, forms-based, graphics-based, etc.
- Mobile Interfaces: interfaces allowing users to perform transactions using mobile apps
- Utilities to perform certain functions such as loading data from files into a database, backing up the database periodically on tape, reorganizing database file structures, performance monitoring utilities, report generation utilities, etc.

Typical DBMS Component Modules

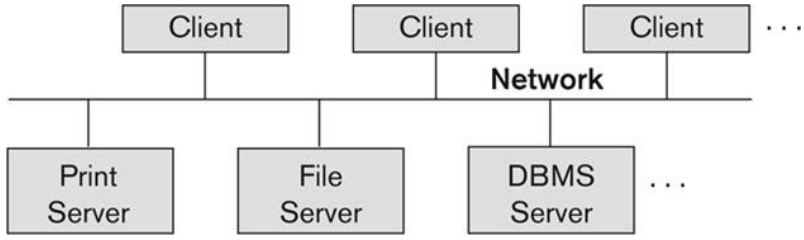
Figure 2.3 Component modules of a DBMS and their interactions.



DBMS Architectures - Centralized and Client-Server

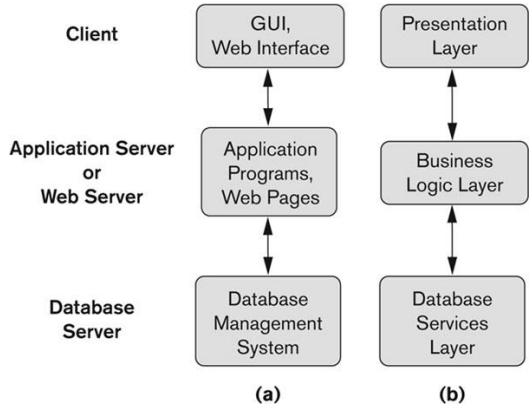
- Centralized DBMS:
  - Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.
  - User can still connect through a remote terminal – however, all processing is done at centralized site.
- Client-Server Architectures
  - Clients can access the specialized servers as needed
  - DBMS server provides database query and transaction services to the clients
  - Applications running on clients utilize an Application Program Interface (**API**) to access server databases via standard interface such as ODBC (Open Database Connectivity standard) or JDBC for Java programming access

Figure 2.5 Logical Two-Tier Client Server Architecture



Three-Tier Client-Server Architecture

Figure 2.7 Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.



Historical Development of Database Technology

- **Early Database Applications:** The Hierarchical and Network Models (formatted models) were introduced in mid 1960s and dominated in 1970s (now called legacy systems e.g., IBM's IMS system).
- **Relational Model based Systems:** Relational model was originally introduced in 1970 by E.F. Codd, was heavily researched and experimented within IBM Research and several universities. Relational DBMS Products emerged in the early 1980s.
- **Object-oriented and emerging applications:**
  - Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980s/early 1990s to cater to the need of complex data processing. Their use has not taken off much.
  - Many relational DBMSs have incorporated object database concepts, leading to a new category called **object-relational** DBMSs (ORDBMSs)
  - **Extended relational** systems add further capabilities (e.g. for multimedia data, spatial data management, data warehousing and data mining, time series)

Historical Development of Database Technology (Cont'd)

- **Data on the Web and E-commerce Applications:**
  - Web contains data in HTML (Hypertext markup language) with links among pages.
  - This has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language). (see Ch13).
  - Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database (see Ch11).
- **Background since the advent of the 21st Century:**
  - First decade of the 21st century has seen tremendous growth in user generated data and automatically collected data from applications and search engines.
  - Social Media platforms such as Facebook and Twitter are generating millions of transactions a day and businesses are interested to tap into this data to "understand" the users
  - Cloud Storage and Backup is making unlimited amount of storage available to users and applications

Historical Development of Database Technology (Cont'd)

- **Emergence of Big Data Technologies and NoSQL databases**
  - New data storage, management and analysis technology was necessary to deal with the onslaught of data in petabytes a day (10**15 bytes or 1000 terabytes) in some applications – this started being commonly called as "Big Data".
  - Hadoop (which originated from Yahoo) and Mapreduce Programming approach to distributed data processing (which originated from Google) as well as the Google file system have given rise to Big Data technologies (Chapter 25). Further enhancements are taking place in the form of Spark based technology.
  - NoSQL (Not Only SQL- where SQL is the de facto standard language for relational DBMSs) systems have been designed for rapid search and retrieval from documents, processing of huge graphs occurring on social networks, and other forms of unstructured data with flexible models of transaction processing (Chapter 24).

Copyright