CHAPTER 1

Databases and Database Users

OUTLINE

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Types of Database Users
- Advantages of Using the Database Approach
- Historical Development of Database Technology
- Extending Database Capabilities
- When Not to Use Databases

What is data, database, DBMS

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

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
- *A number of recent applications are described later in the book (for example, Chapters 24,25,26,27,28,29)*

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-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.

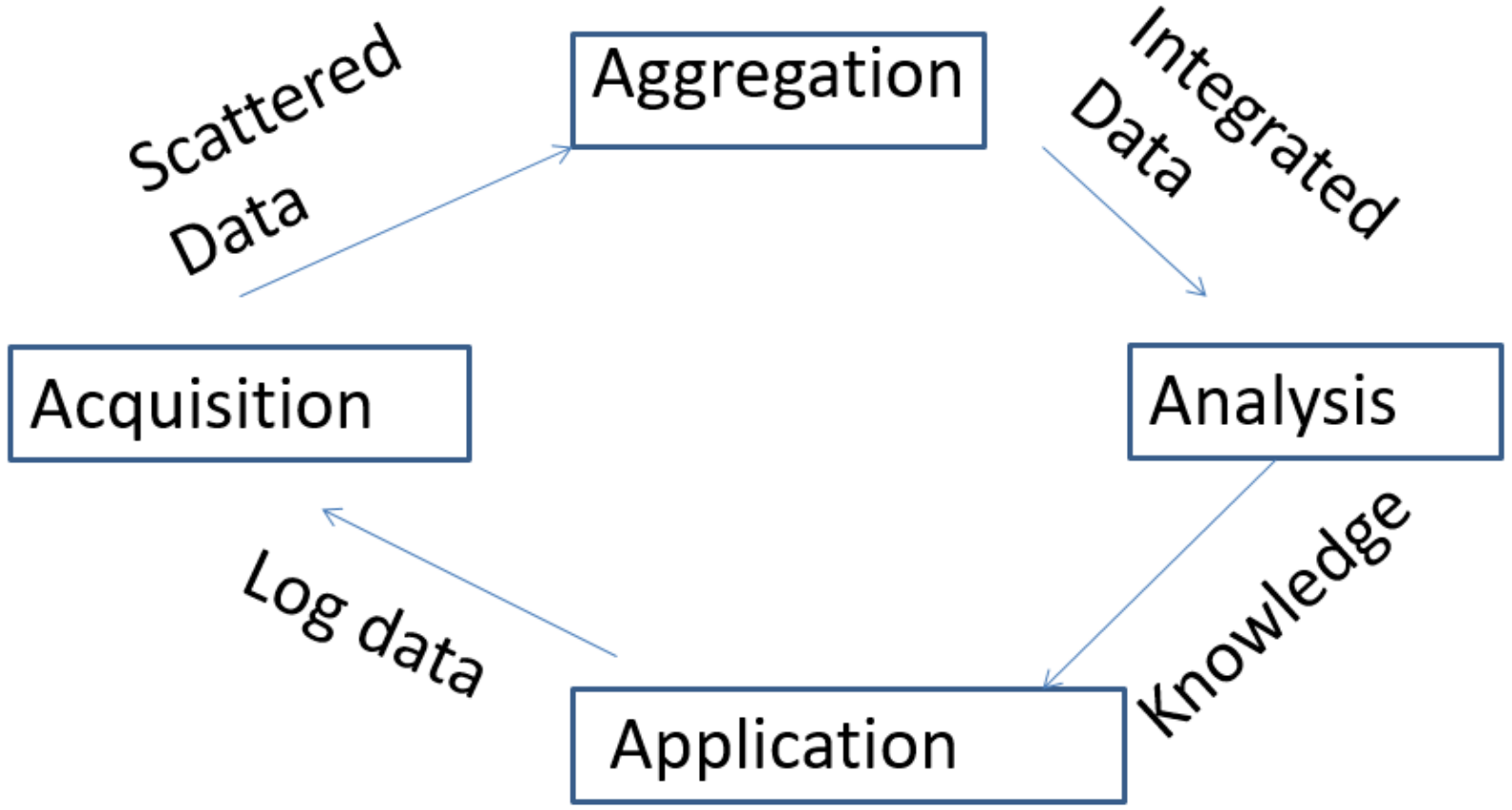
What is “big data”?

- "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”

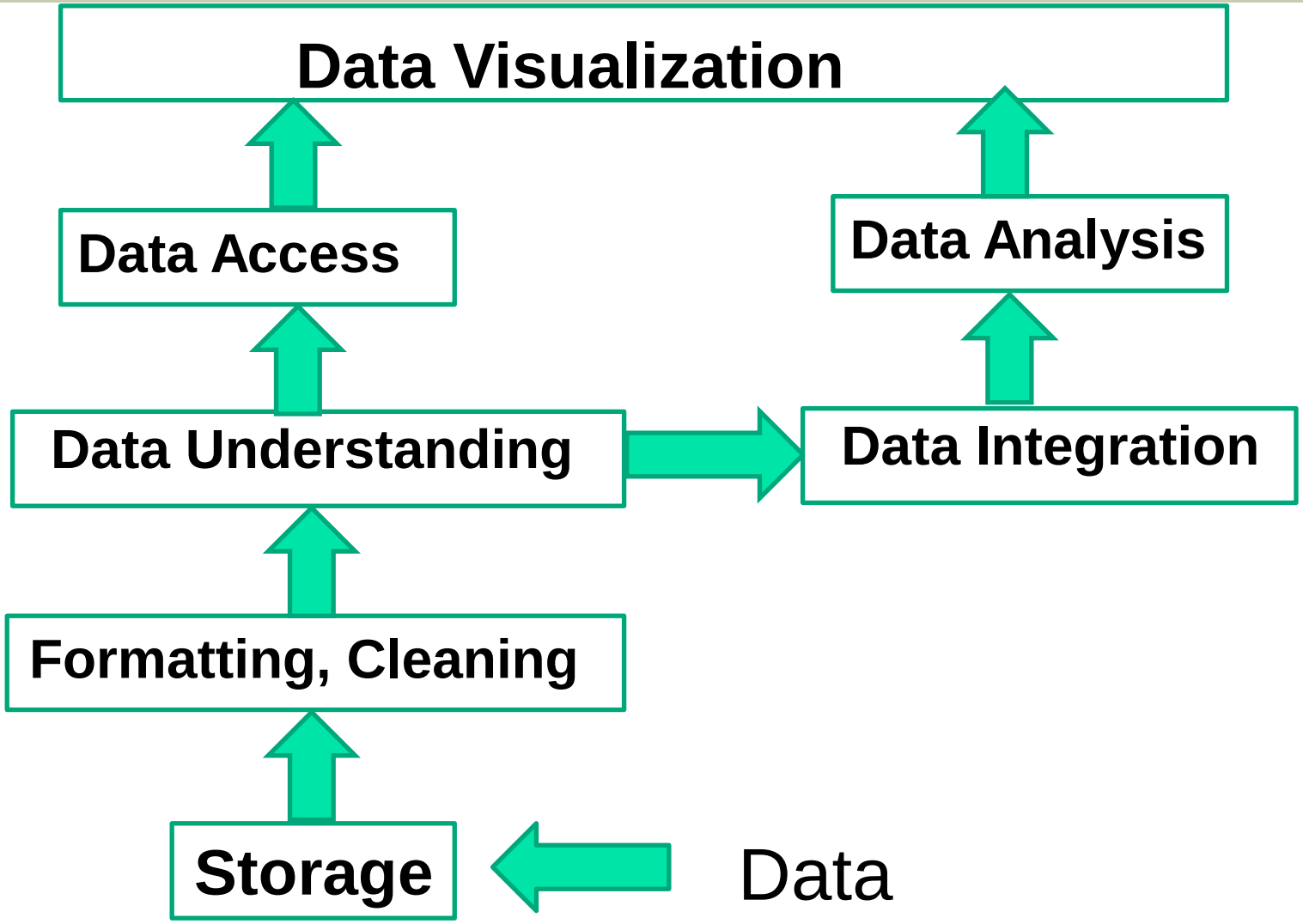
Why is “big data” a “big deal”?

- 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

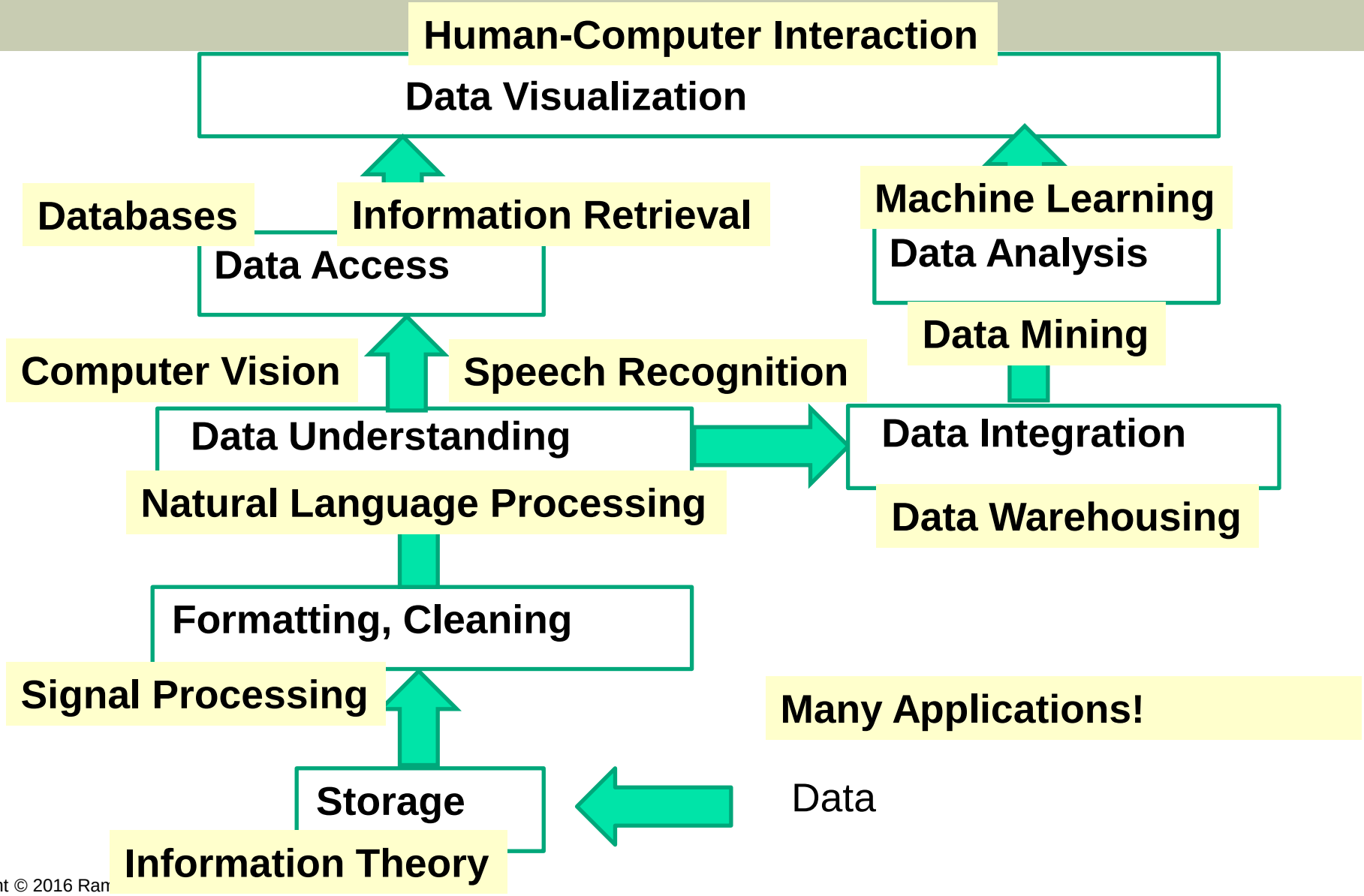
Lifecycle of Data: 4 “A”s



Computational View of Big Data



Big Data & Related Disciplines



Basic Definitions

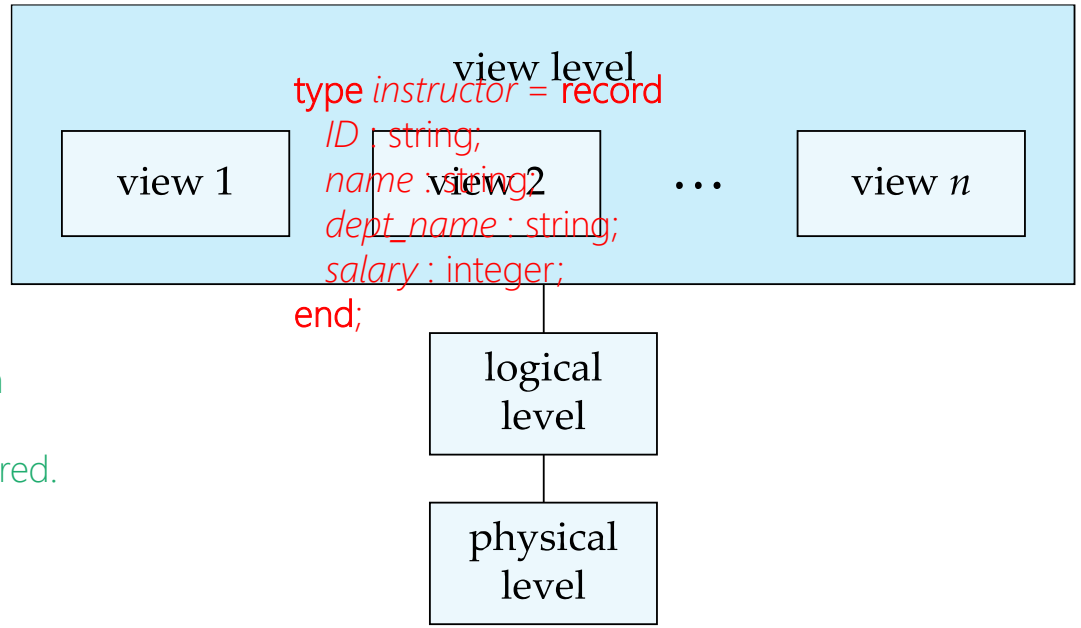
- Database:
  - A collection of related data.
- Data:
  - Known facts that can be recorded and have an implicit meaning.
- Mini-world:
  - Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- Database Management System (DBMS):
  - 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

A simplified architecture for a database system

View level: what application programs see; views can also hide information (such as an instructor's salary) for security purposes.



Physical level: describes how a record (e.g., instructor) is stored.

A simplified architecture for a database system

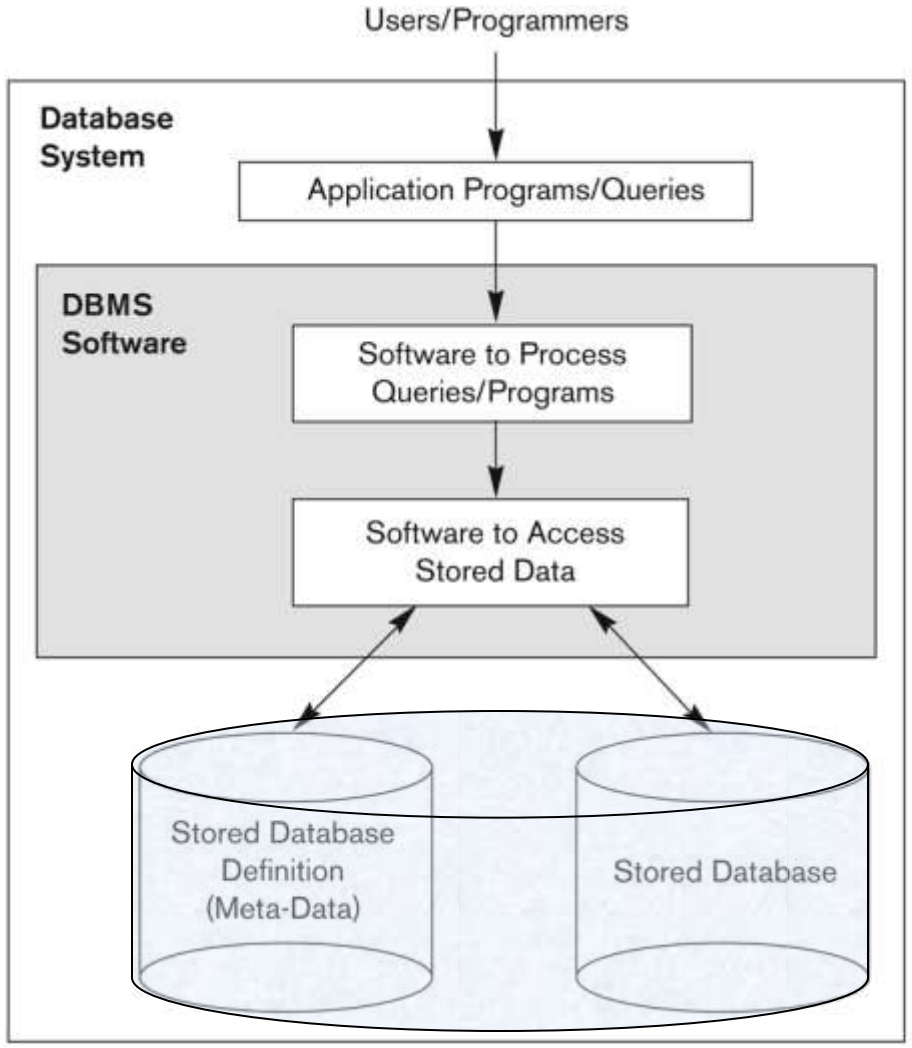
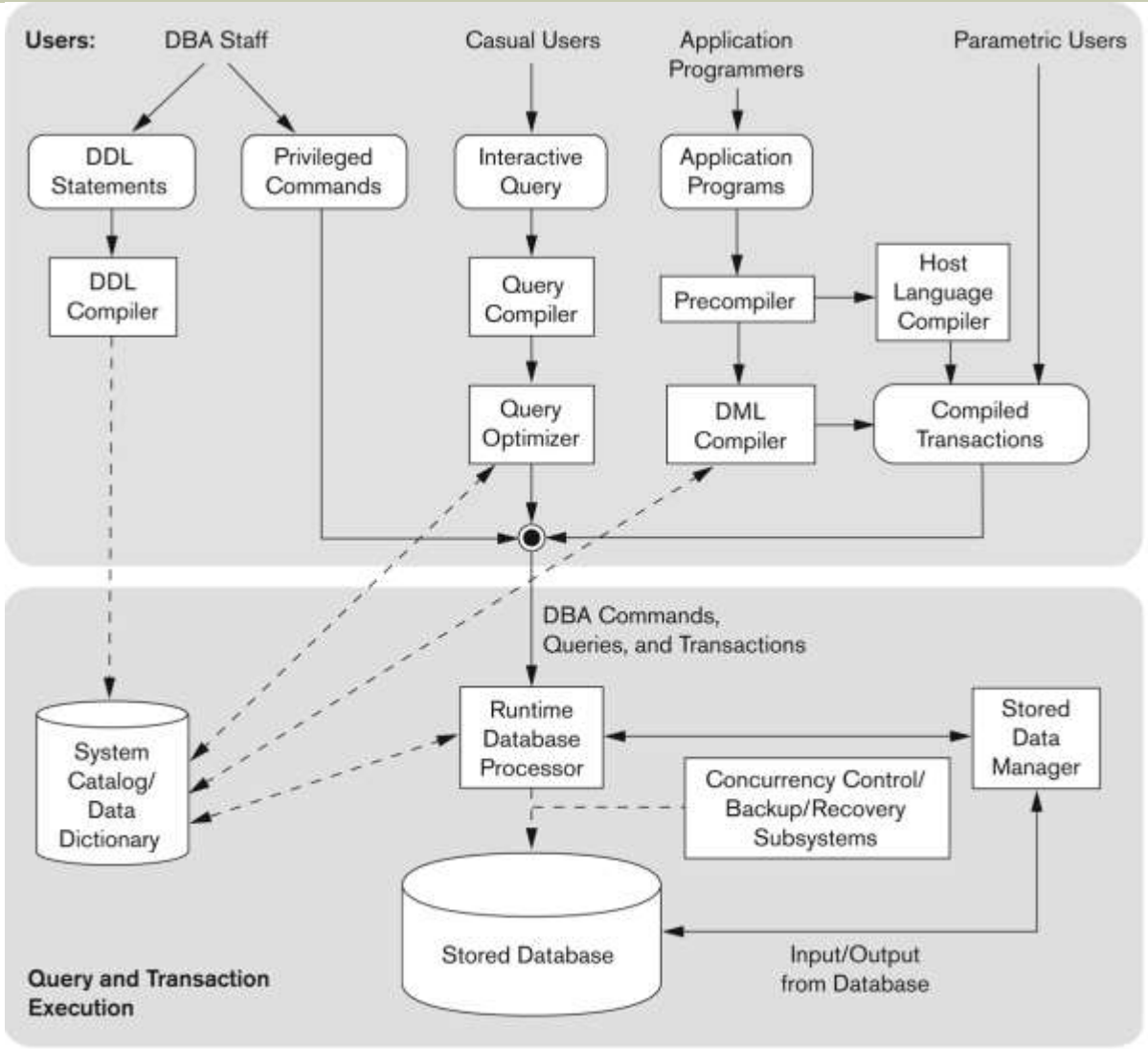
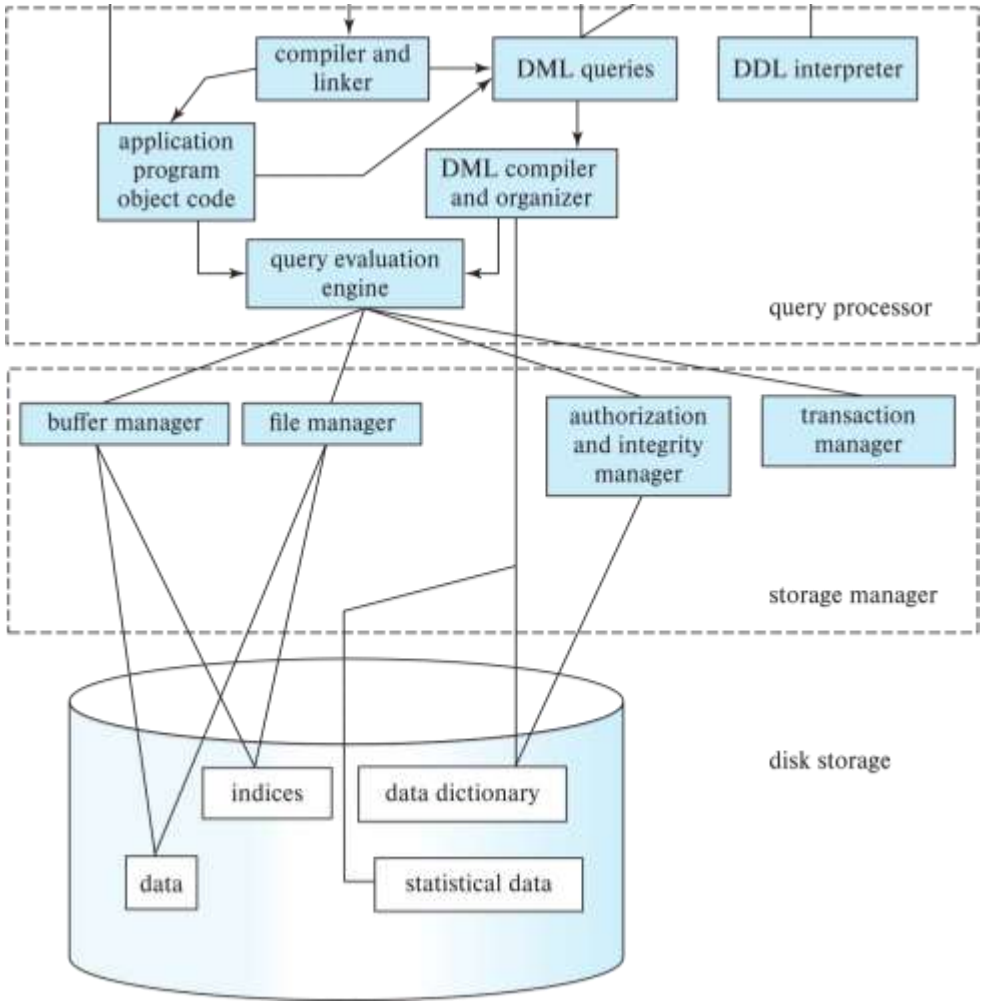


Figure 1.1
A simplified database system environment.

A simplified architecture for a database system



A simplified architecture for a database system



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

Application Programs and DBMS

- 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

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 (ER) data or UML class model (see Chapters 3, 4)

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

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 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.

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
- 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 (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
  - *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.

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 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– see Chapter 12.
- Providing storage structures (e.g. indexes) for efficient query processing – see Chapter 17.

Advantages of Using the Database Approach (continued)

- 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 Models (continued)

- **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)

Categories of Data Models

- **Conceptual (high-level, semantic) data models:**
  - Provide concepts that are close to the way many users perceive data.
    - (Also called ***entity-based*** or ***object-based*** data models.)
- **Physical (low-level, internal) data models:**
  - Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals
- **Implementation (representational) data models:**
  - Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).
- **Self-Describing Data Models:**
  - Combine the description of data with the data values. Examples include XML, key-value stores and some NOSQL systems.

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.

Example of a Database Schema

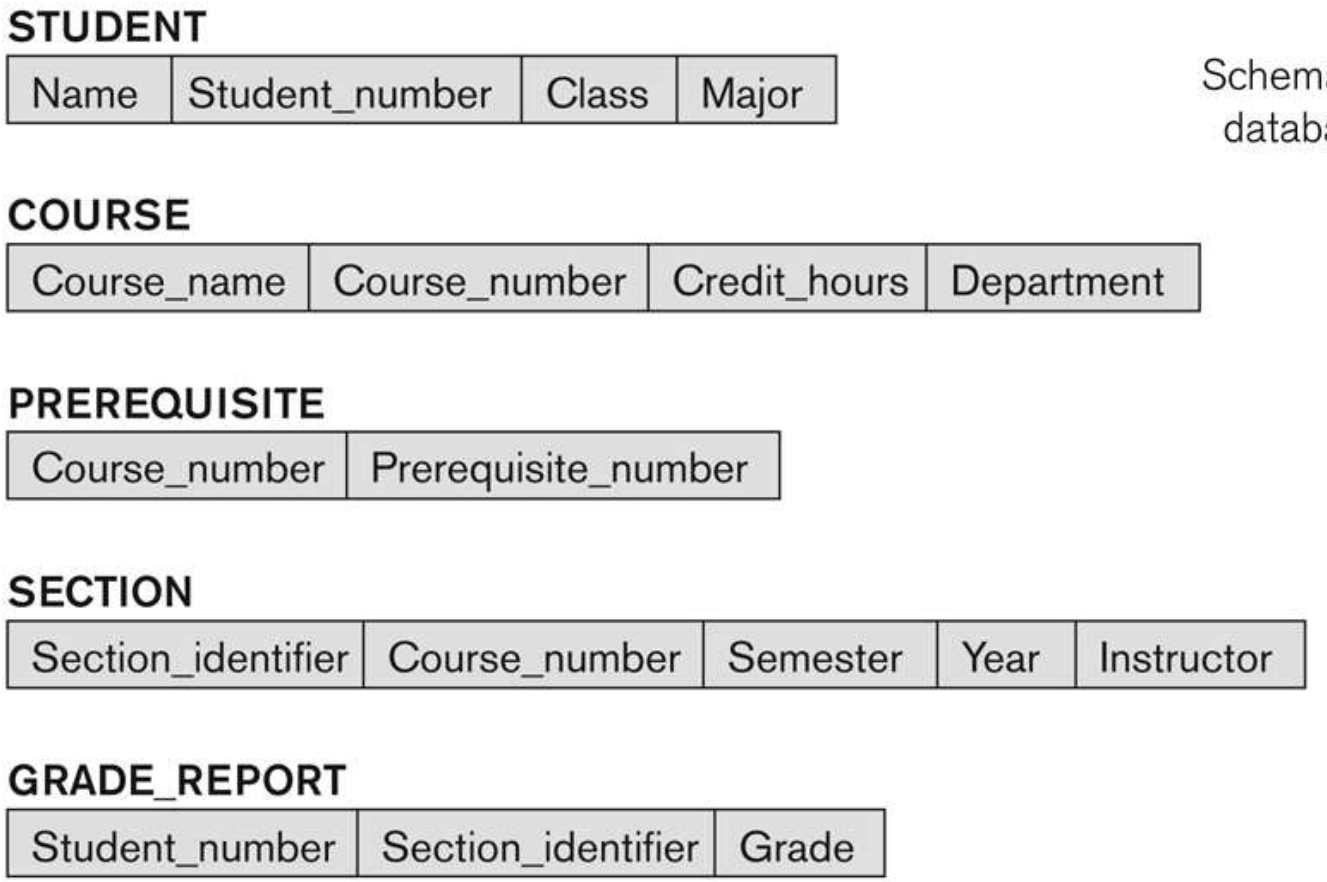


Figure 2.1
Schema diagram for the database in Figure 1.2.

Schemas versus Instances

- 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.

Database Schema vs. Database State (continued)

- 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

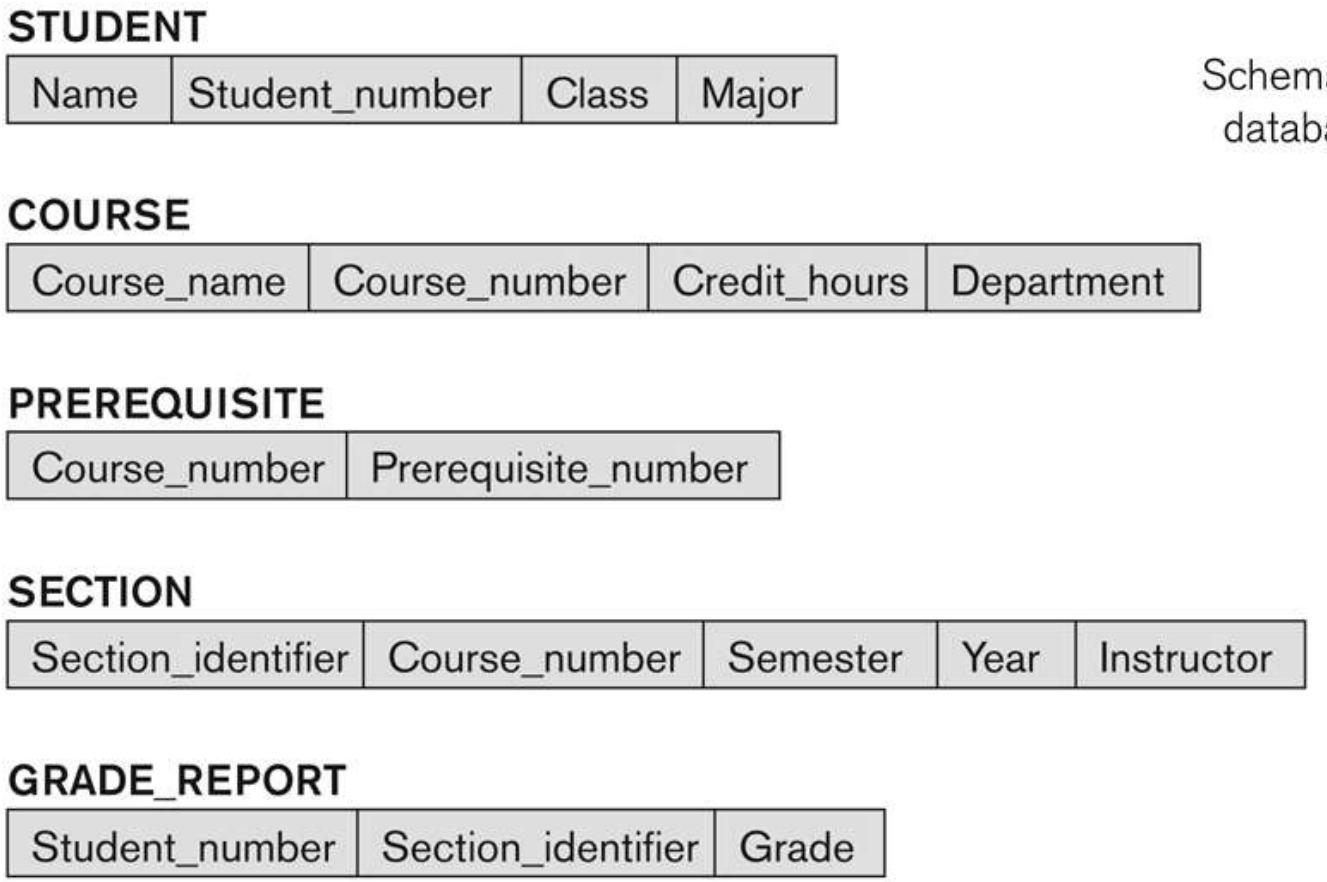


Figure 2.1
Schema diagram for the database in Figure 1.2.

Example of a database state

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	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.

Three-Schema Architecture

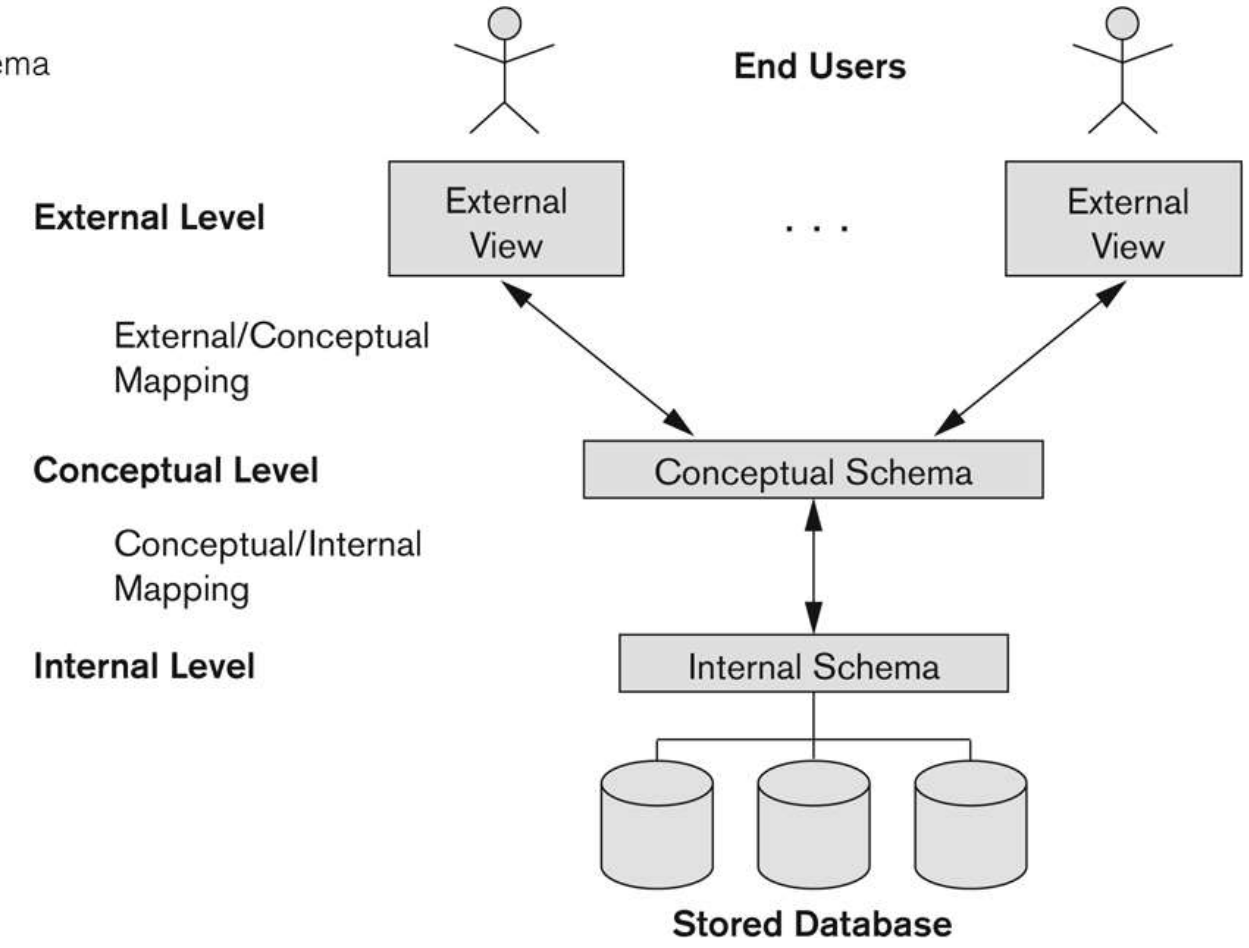
- Proposed to support DBMS characteristics of:
  - **Program-data independence.**
  - Support of **multiple views** of the data.
- Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization

Three-Schema Architecture

- Defines DBMS schemas at ***three*** levels:
  - **Internal schema** at the internal level to describe physical storage structures and access paths (e.g indexes).
    - Typically uses a **physical** data model.
  - **Conceptual schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users.
    - Uses a **conceptual** or an **implementation** data model.
  - **External schemas** at the external level to describe the various user views.
    - Usually uses the same data model as the conceptual schema.

The three-schema architecture

Figure 2.2
The three-schema architecture.



Three-Schema Architecture

- Mappings among schema levels are needed to transform requests and data.
  - Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
  - Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)

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

Data Independence (continued)

- 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)
- Data Manipulation Language (DML)
  - High-Level or Non-procedural Languages: These include the relational language SQL
    - May be used in a standalone way or may be embedded in a programming language
  - Low Level or Procedural Languages:
    - These must be embedded in a programming language

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).
  - In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define internal and external schemas.
    - SDL is typically realized via DBMS commands provided to the DBA and database designers

DBMS Languages

- **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*).

Types of DML

- **High Level or Non-procedural Language:**
  - For example, the SQL relational language
  - Are “set”-oriented and specify what data to retrieve rather than how to retrieve it.
  - Also called **declarative** languages.
- **Low Level or Procedural Language:**
  - Retrieve data one record-at-a-time;
  - Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

DBMS Interfaces

- 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
- User-friendly interfaces
  - Menu-based, forms-based, graphics-based, etc.
- Mobile Interfaces:interfaces allowing users to perform transactions using mobile apps

DBMS Programming Language Interfaces

- Programmer interfaces for embedding DML in a programming languages:
  - **Embedded Approach**: e.g embedded SQL (for C, C++, etc.), SQLJ (for Java)
  - **Procedure Call Approach**: e.g. JDBC for Java, ODBC (Open Databse Connectivity) for other programming languages as API's (application programming interfaces)
  - **Database Programming Language Approach**: e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components
  - **Scripting Languages:** PHP (client-side scripting) and Python (server-side scripting) are used to write database programs.

User-Friendly DBMS Interfaces

- Menu-based (Web-based), popular for browsing on the web
- Forms-based, designed for naïve users used to filling in entries on a form
- Graphics-based
  - Point and Click, Drag and Drop, etc.
  - Specifying a query on a schema diagram
- Natural language: requests in written English
- Combinations of the above:
  - For example, both menus and forms used extensively in Web database interfaces

Other DBMS Interfaces

- Natural language: free text as a query
- Speech : Input query and Output response
- Web Browser with keyword search
- Parametric interfaces, e.g., bank tellers using function keys.
- Interfaces for the DBA:
  - Creating user accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access paths

Database System Utilities

- To perform certain functions such as:
  - Loading data stored in files into a database. Includes data conversion tools.
  - Backing up the database periodically on tape.
  - Reorganizing database file structures.
  - Performance monitoring utilities.
  - Report generation utilities.
  - Other functions, such as sorting, user monitoring, data compression, etc.

Other Tools

- Data dictionary / repository:
  - Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.
- Application Development Environments and CASE (computer-aided software engineering) tools:
  - PowerBuilder (Sybase), JBuilder (Borland), JDeveloper 10G (Oracle)

Typical DBMS Component Modules

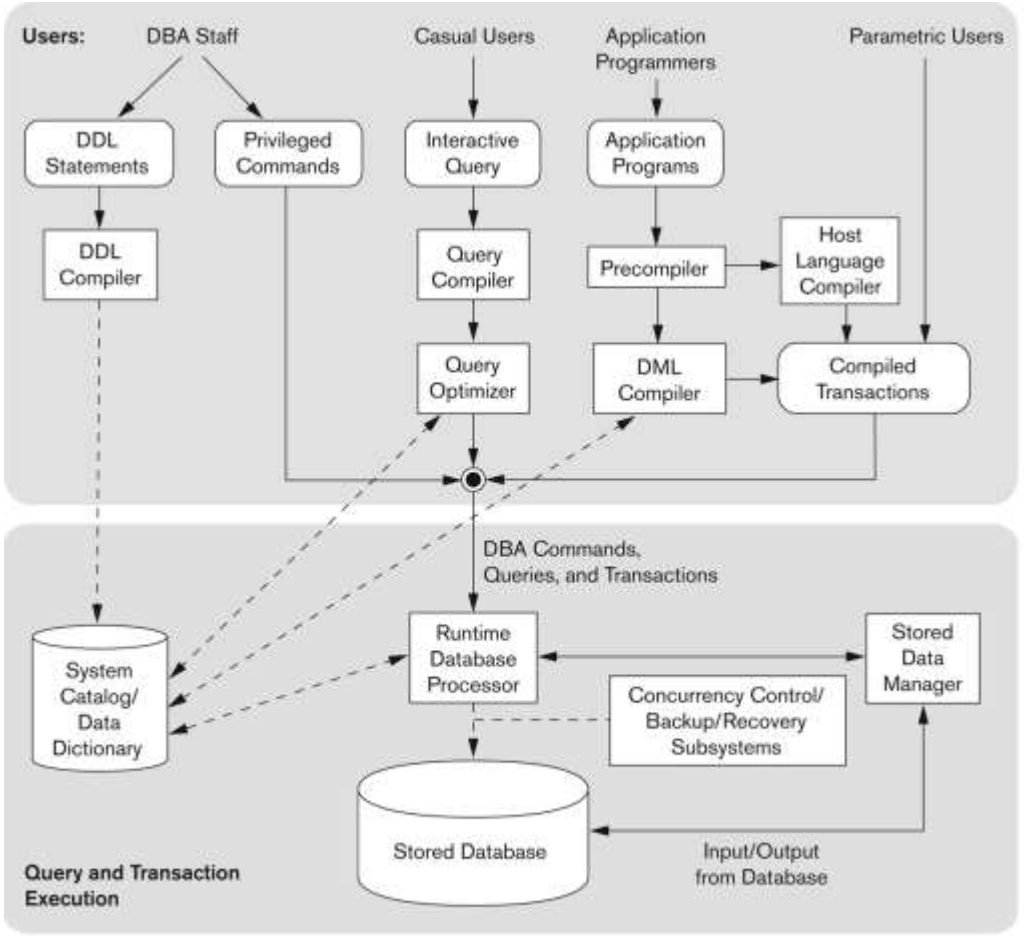


Figure 2.3
Component modules of a DBMS and their interactions.