BBM371- Data Management

Lecture 1:

Course policies,

Introduction to Database Management Systems

Today

- Intoduction
  - About the class
  - Organization of the course

- Introduction to Database Management Systems (DBMS)

About the class

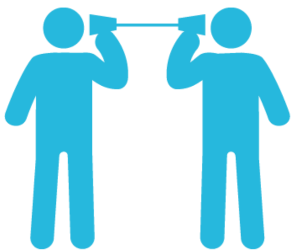
Resources

The course web page is

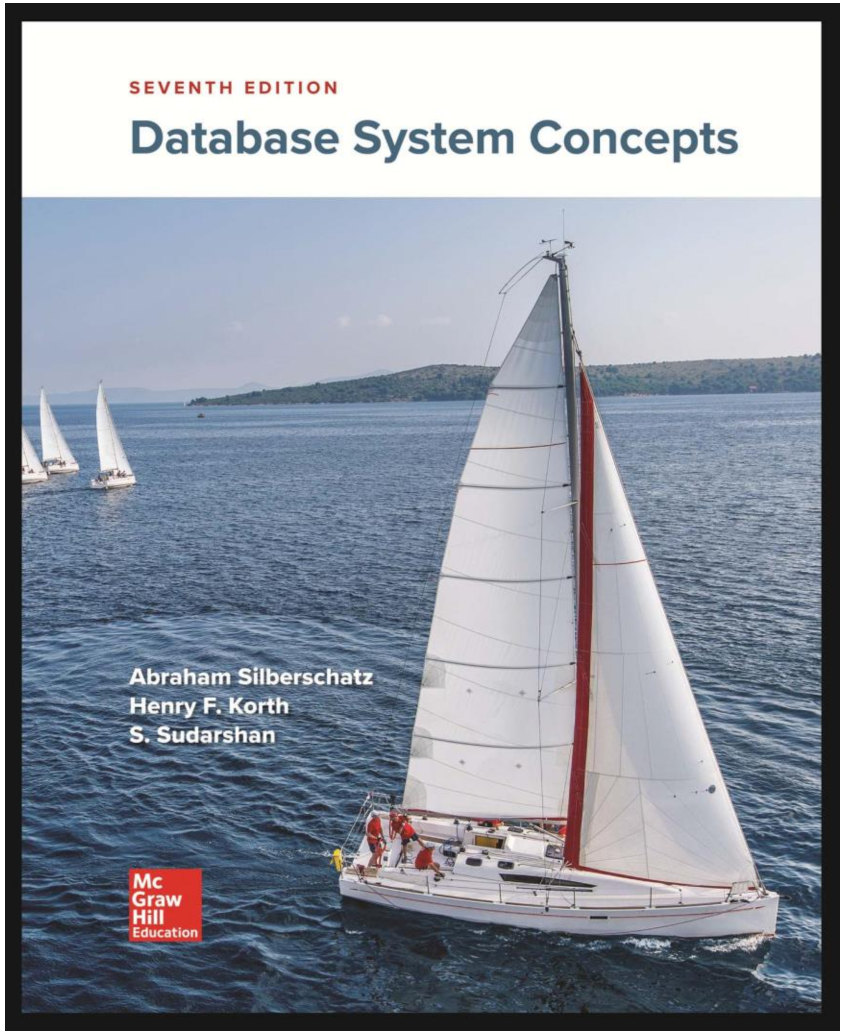
http://web.cs.hacettepe.edu.tr/~bbm371

Announcements will be posted on Piazza

http://piazza.com/hacettepe.edu.tr/fall2020/bbm371

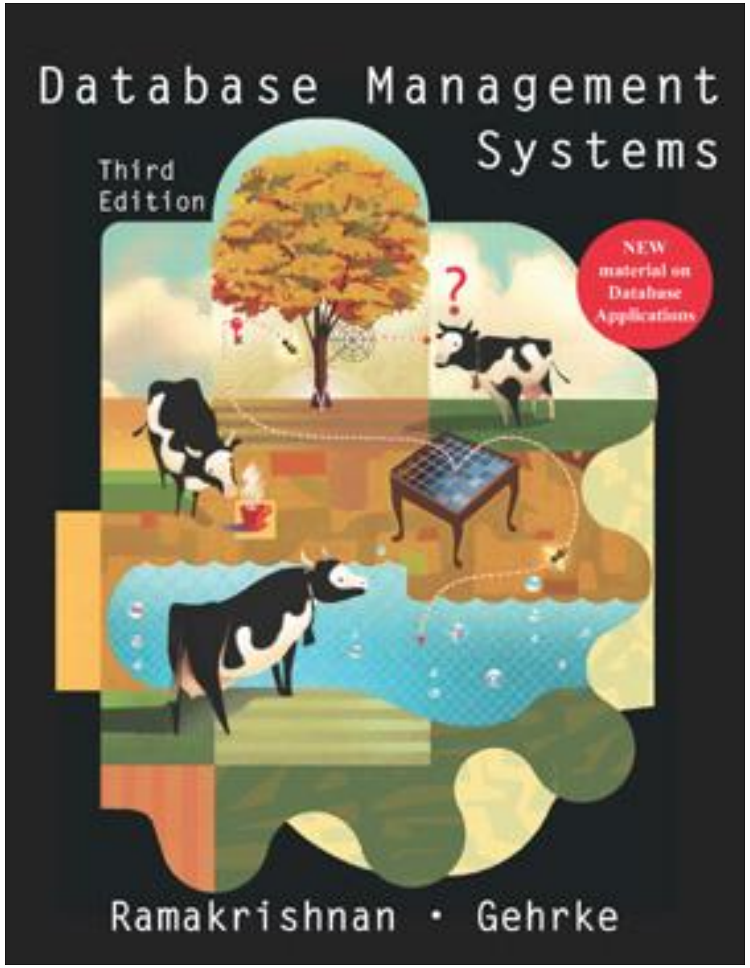


Textbook



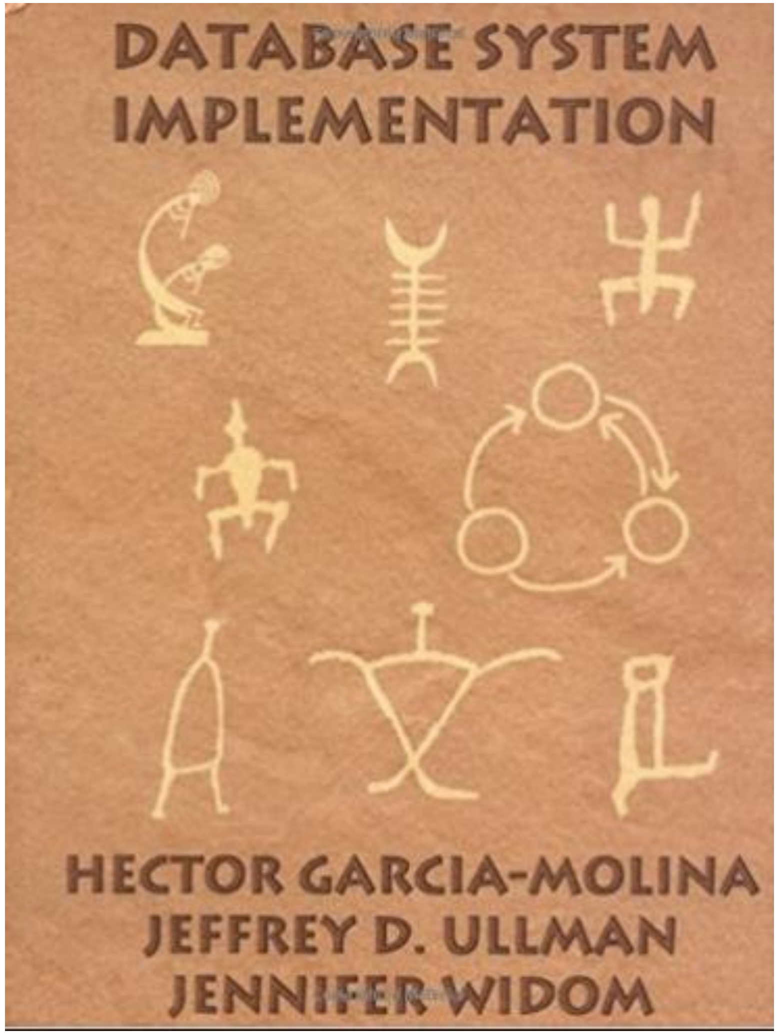
Avi Silberschatz, Henry F. Korth, S. Sudarshan: Database System Concepts, Seventh Edition. McGraw-Hill Book Company 2020, ISBN 9780078022159 (db-book.com)

Reference Book - 1



Database Management Systems, Raghu Ramakrishnan, McGraw-Hill Education

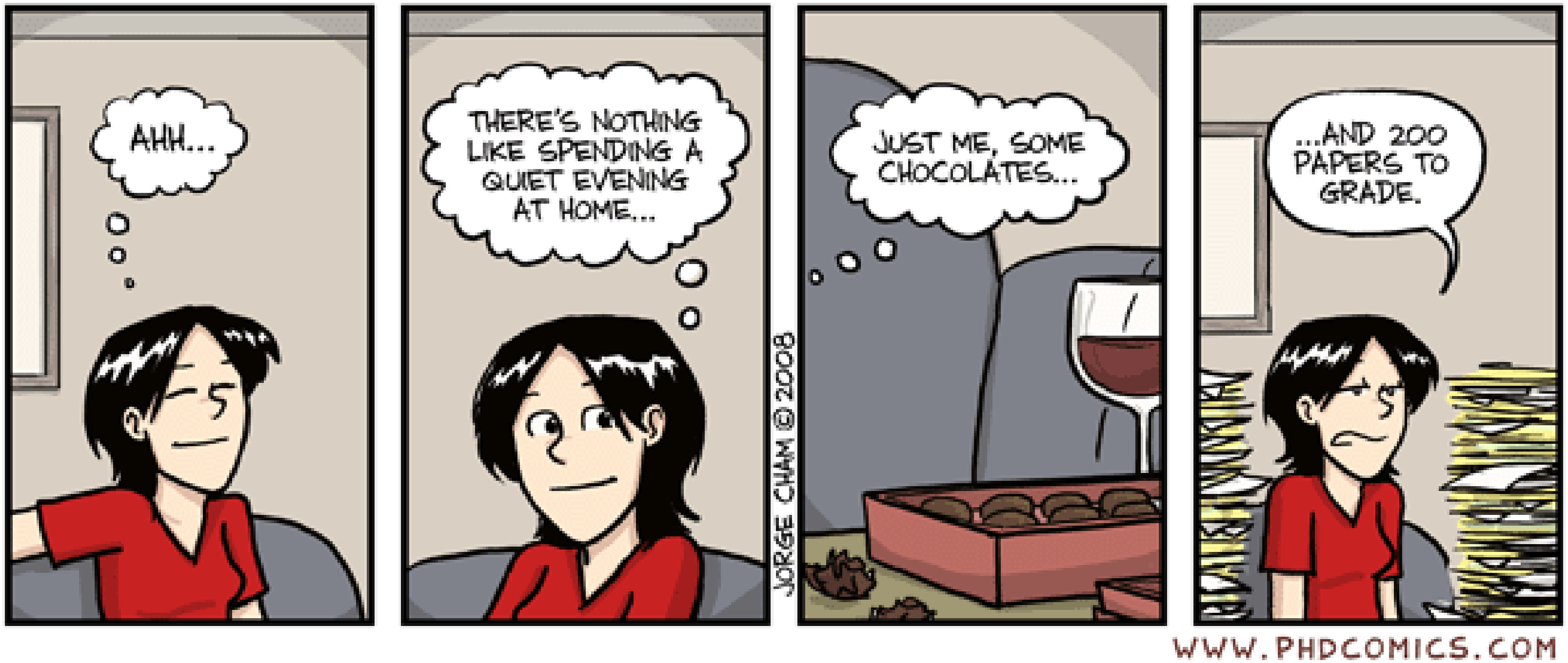
Reference Book



Database System Implementation, Hector Garcia-Molina, Jeffrey D. Ullman, Jenniver Widom

Course Work and Grading

- **Quizes (25 points)**
  - 5 out of 6
- **Midterm exams (25 points)**
  - Closed book and notes
- **Final exam (50 points)**
  - Closed book and notes



Course Overview (Tentative)

Week	Date	Topic	Assessments
1	8.10.2020	Introduction to Data Management and Databases, Architecture	
2	15.10.2020	Entity Relationship Model	
3	22.10.2020	Relational Data Model	
4	29.10.2020	No Lecture (Republic Day)	
5	5.11.2020	SQL	Q1
6	12.11.2020	Intermediate SQL	
7	19.11.2020	Advanced SQL	Q2
8	26.11.2020	Query Processing (join algorithms, external sorting)	Q3
9	3.12.2020	Midterm Review	MIDTERM EXAM
10	10.12.2020	Physical Storage Systems	
11	17.12.2020	Data Storage Structures	Q4
12	24.12.2020	Tree Based Indexing	
13	31.12.2020	Hash Based Indexing	Q5
14	7.1.2021	Spatial Data Management	Q6
15-16		FINAL EXAM	

Introduction to Database Management Systems

What is Data?

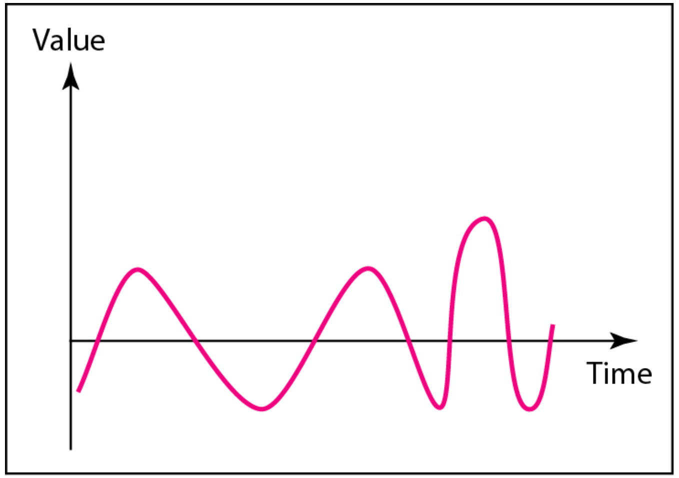
- **Data:** Almost any kind of unorganized fact(s).

- Examples:
  - You throw a dice for a million times. Results are your data.
  - Anything you see in this classroom.
  - Music on a CD.
  - A computer file.

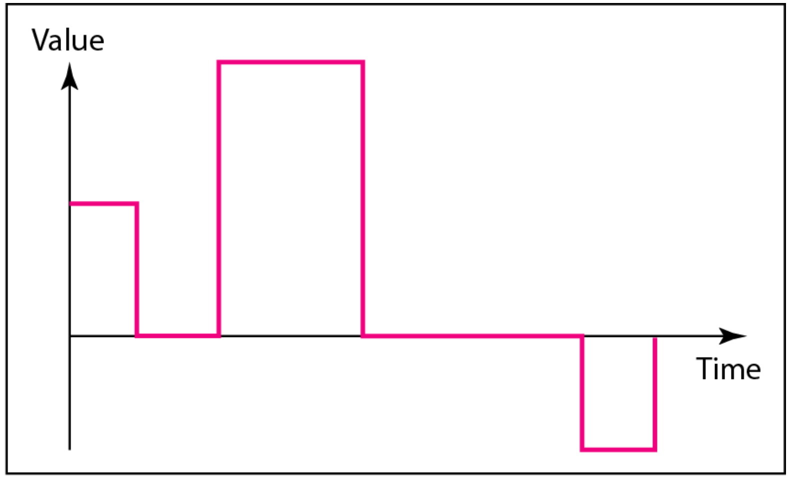
What is Signal?

- Signal is the encoding of the data that is needed for transmission.

- Analog
- Digital



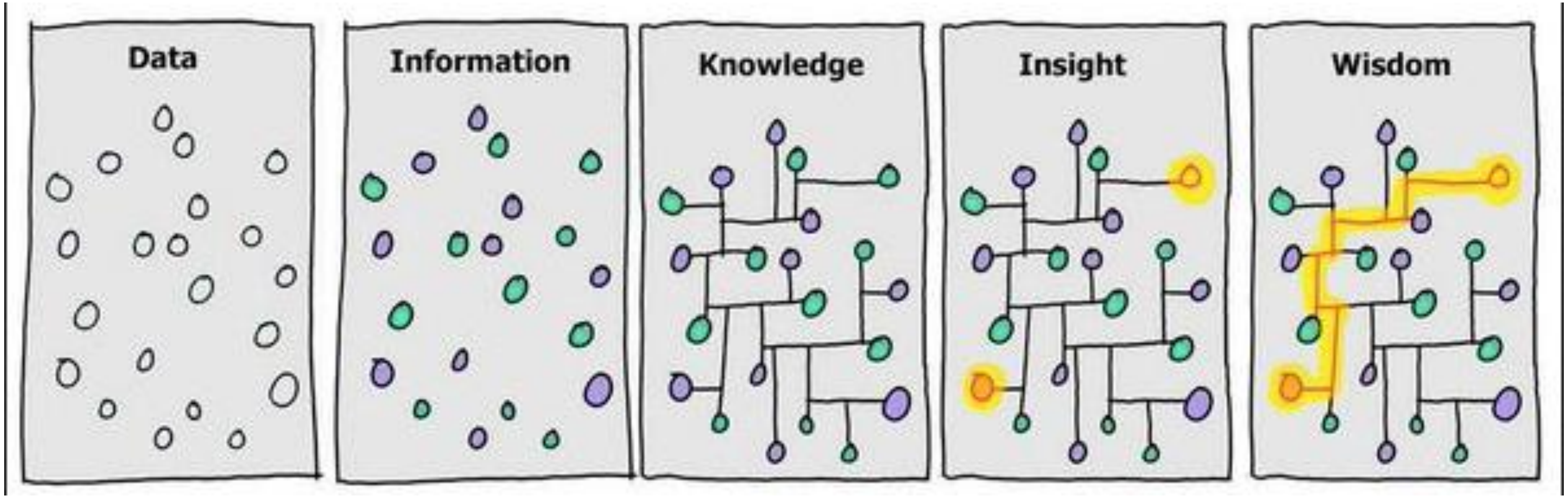
a. Analog signal



b. Digital signal

What is Information?

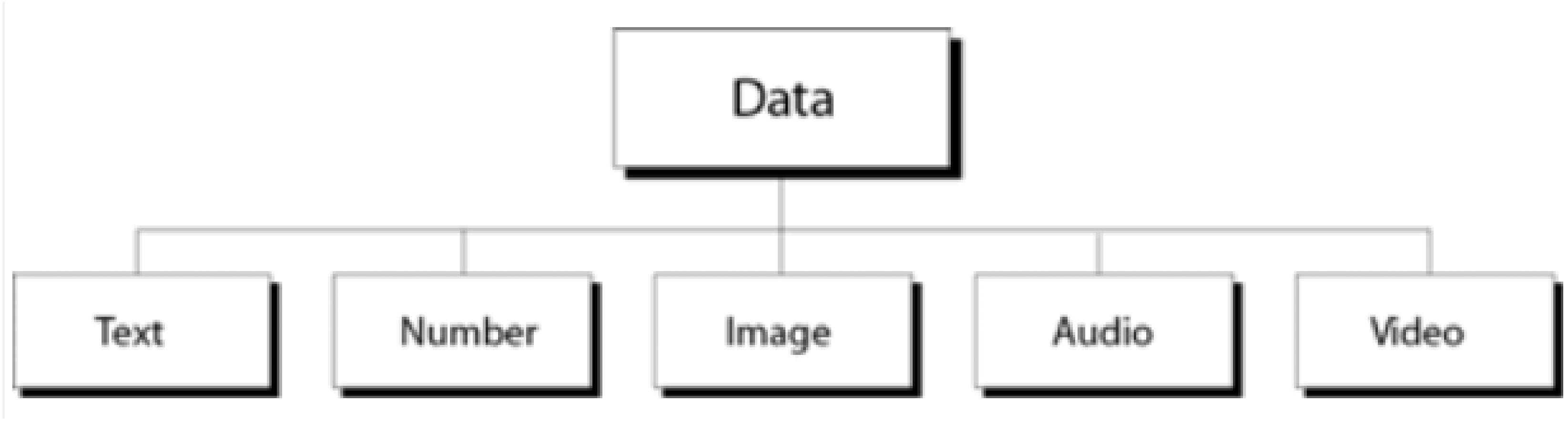
- Data becomes information when it is processed and organized and thereby it becomes useful.



Data-Centric Applications

- Applications in which data plays an important role
  - Airline reservation systems
    - Data: aircrafts, flights, flight attendants, passengers, etc.
- Banking applications
  - Data: clients, deposits, withdraws, etc.
- Hospital systems
  - Data: patients, physicians, diagnosis, prescriptions, etc.
- University systems
  - Data: students, teaching staff, courses, enrollments, etc.

How to represent Data?



Purpose of Database Systems

- In the early days, database applications were built directly on top of file systems, which leads to:
- Data redundancy and inconsistency: data is stored in multiple file formats resulting induplication 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

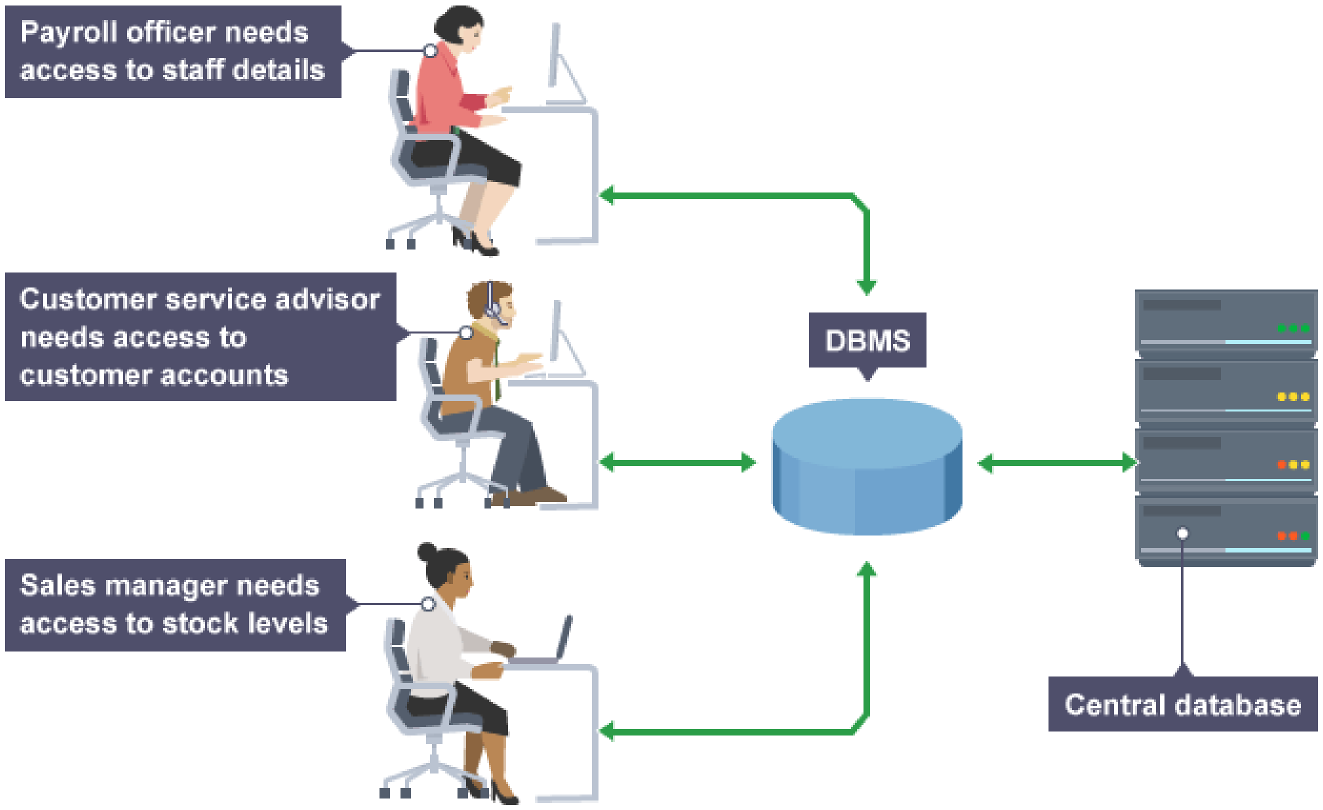
Purpose of Database Systems (cont.)

- 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
    - Ex: 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

Why Use a Database System?

- Data independence and efficient access
- Reduced application and development time
- Data integrity and security
- Uniform data administration
- Concurrent access
- Recovery from crashes



What is Management?

The process of dealing with things (or people)!

- Initiation/Setting Objectives
- Planning
- Design and Implementation
- Execution
- Monitoring and Control

What is a DBMS?

- A very large, integrated collection of data.
- Models real-world enterprise
- A Database Management System (DBMS) is a software package designed to store and manage databases
- Information about:
  - Entities: such as students, faculty, courses
  - Relationships: between entities for example a student is enrolled to a course

History of DBMS

- 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 (Michael Stonebraker) begins Ingres prototype
    - Oracle releases first commercial relational database
  - High-performance (for the era) transaction processing

History of DBMS (cont.)

- 2000s
  - Big data storage systems
    - Google BigTable, Yahoo PNuts, Amazon,
    - "NoSQL" systems.
  - Big data analysis: beyond SQL
    - Map reduce and friends
- 2010s
  - SQL reloaded
    - SQL front end to Map Reduce systems
    - Massively parallel database systems
    - Multi-core main-memory databases

Example of a Traditional Database Application

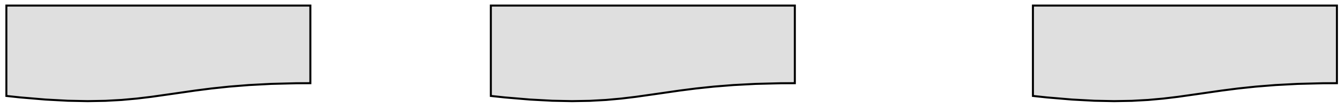
Suppose we are building a system to store the information about:

- students
- courses
- professors
- who takes what, who teaches what

Can we do it without a DBMS ?

Sure we can! Start by storing the data in files:

students.txt   courses.txt   professors.txt



Now write C/C++, Java or Python programs to implement specific tasks

Doing it without a DBMS...

- Enroll "Mary Johnson" in "CSE444":

Write a program to do the following:

Read 'students.txt'
Read 'courses.txt'
Find&update the record "Mary Johnson"
Find&update the record "CSE444"
Write "students.txt"
Write "courses.txt"

Why Study Databases?

- Shift from computation to information
  - Low-end users: Web Applications needs to organize information (a mess will not be effective)
  - High-end users: Scientific applications now have data management problems!

- Datasets increasing in diversity and volume
  - Digital libraries, interactive video, Human Genome project etc.

- DBMS encompasses most of CS
  - OS, languages, AI, multimedia etc.

Data Models

- A data model is a collection of concepts for describing data. (high-level). A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- A schema is a description of a particular collection of data, using the given data model

- 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 Data Model

- The relational model of data is the most widely used model today.
  - Main concept: relation, basically a table with rows and columns
  - Every relation has a schema, which describes the columns, or fields.
  - Schema is defined by: name of schema, the name of each field (or attribute or column) and type of each field

Students(**sid**: *string*, **name**: *string*, **login**: *string*, **age**: *integer*, **gpa**:*real*)

Entity: Student

- **Students**(sid: string, name: string, login: string, age: integer, gpa: real)

Sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8

Using age as a field is not a good idea, why?

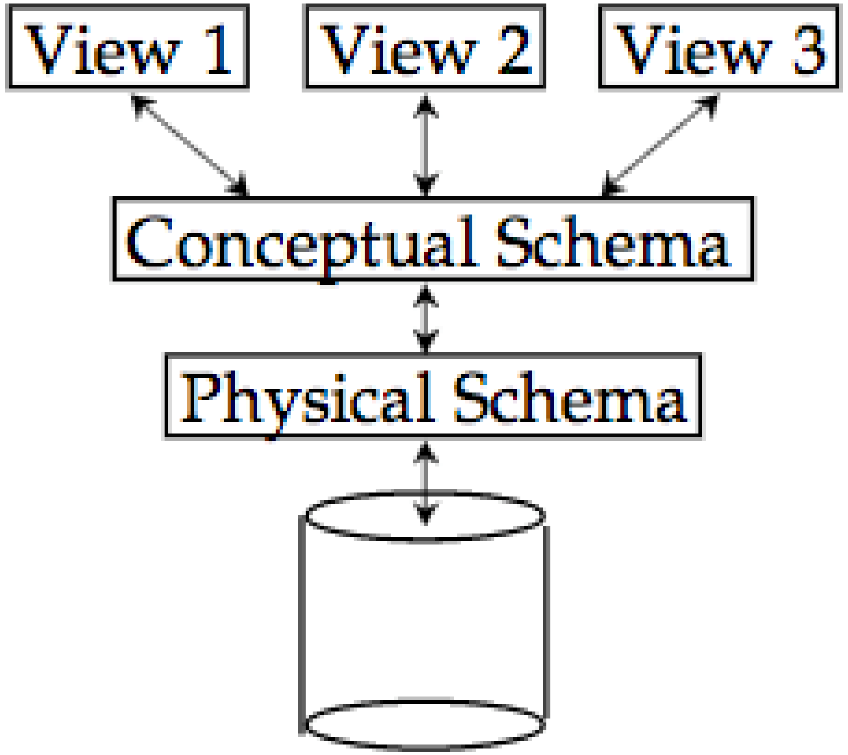
Record

Attribute
(field or column)

Integrity Constraints: We can define the field sid to be unique or age to be larger than 0. Rules for records to satisfy

Levels of Abstraction

- Unlike programmers of early systems, programmer of relational system does not need to implement lower level details
- Many views, single conceptual (logical) schema and physical schema.
  - Views (external level) describe how users see the data.
  - Conceptual schema (logical level) defines logical structure
  - Physical schema (physical level) describes the files and indexes used



External Level

deptN	dChai	sNam	sPos
356	2	HS	Prof
....	....	...	....

Conceptual Layer

Dept-Table

deptno	dNam	dAdr	dChair
356	BilM	mmm	2
357	EleM	ggg	4

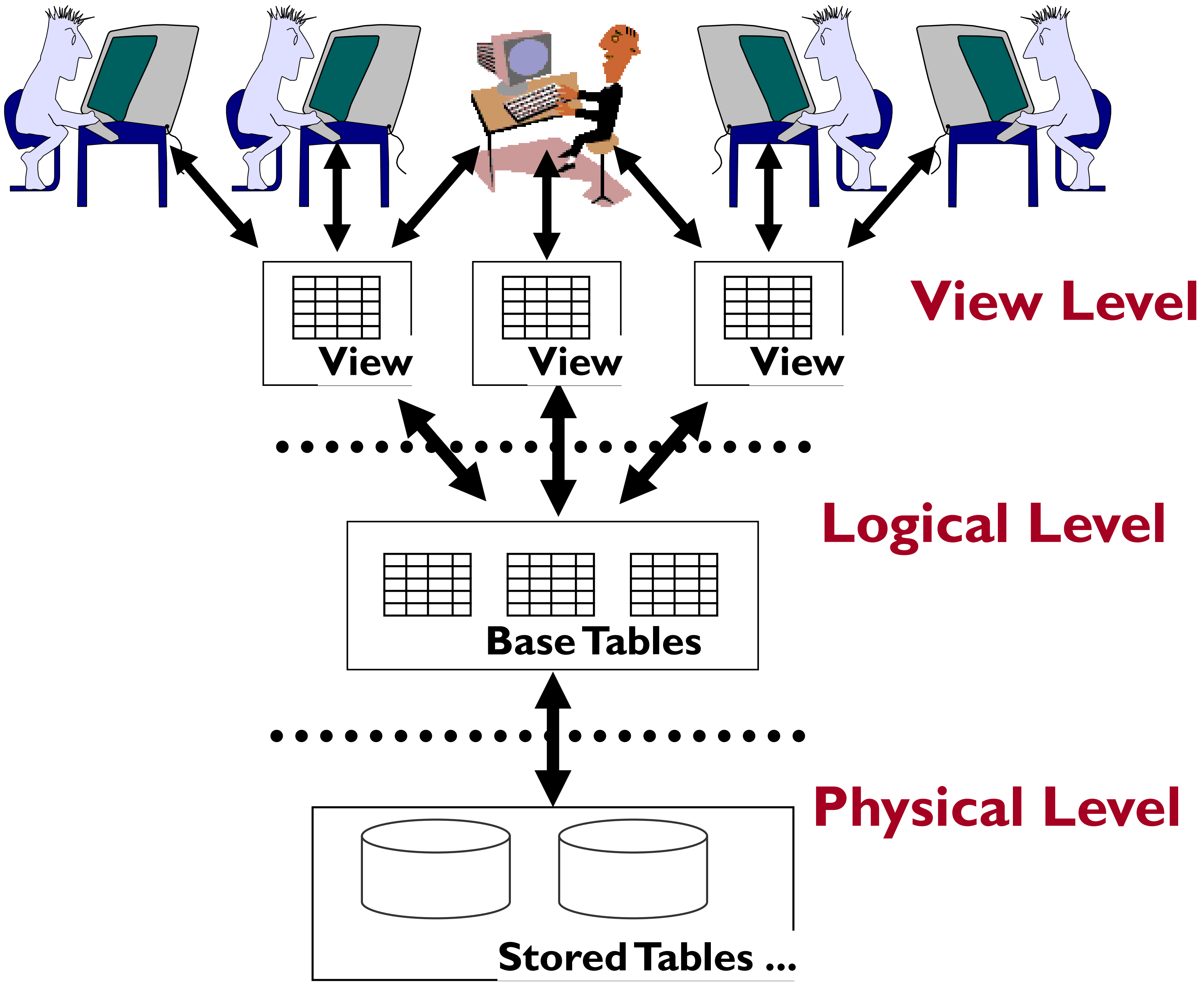
relation

Staff-Table

sId	sNam	sBranc	sPos
1	Ebru	Inf.Ret	Assoc
2	HS	Inf.Ret	Prof

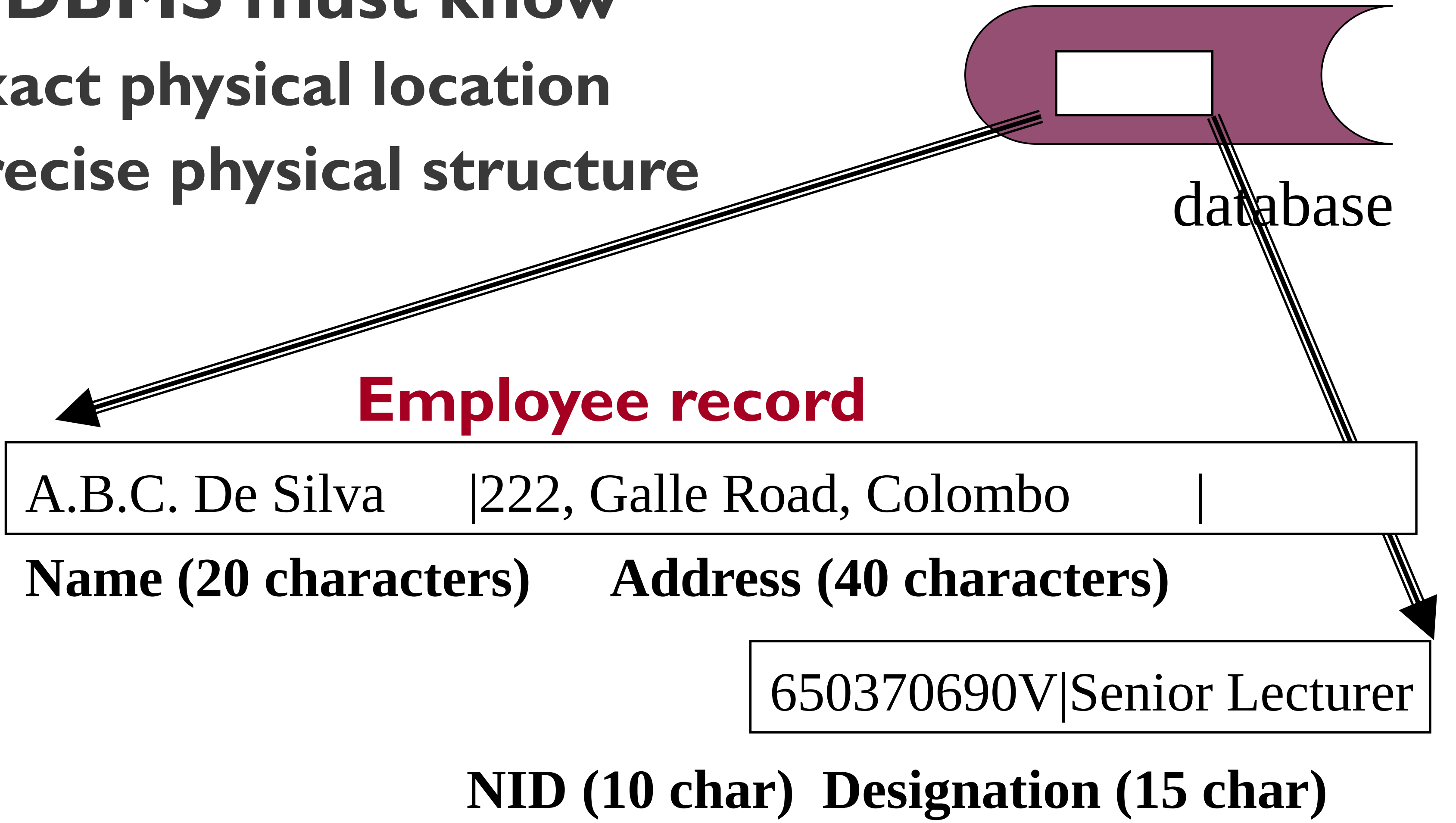
Physical Layer

Dept-File	356BilMmmm2357EleMggg4
Staff-File	1EbruInf.Ret.Assoc2HSInf.Ret.Prof

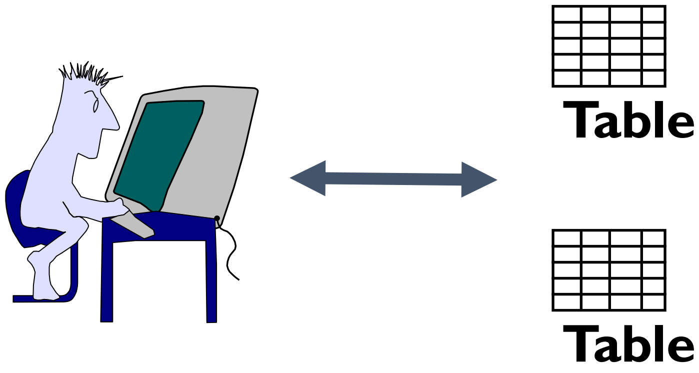

View
View
View
View Level
Logical Level
Base Tables
Physical Level
Stored Tables ...

Physical Layer

- **The DBMS must know**
  - **exact physical location**
  - **precise physical structure**



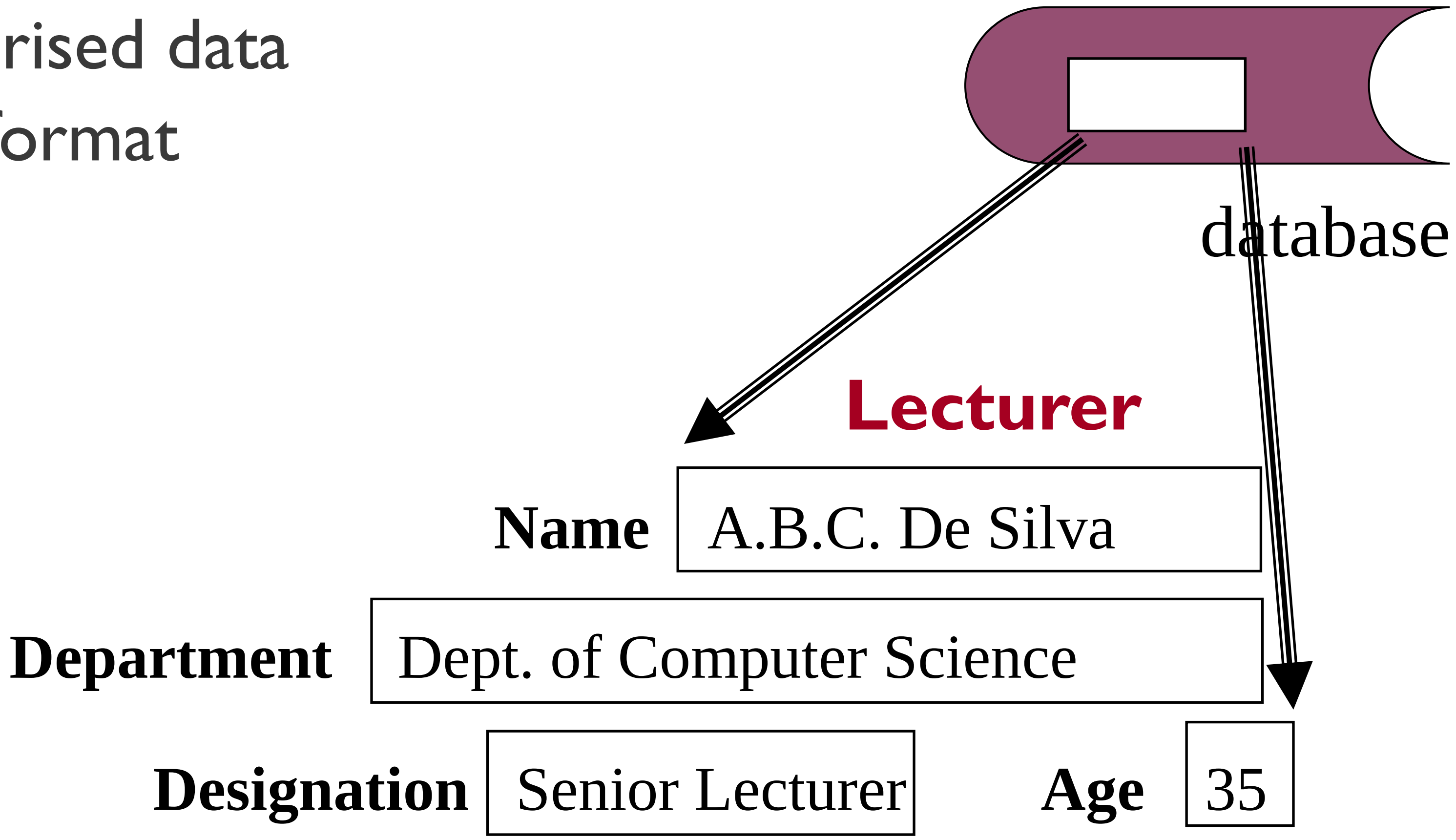
Logical (Conceptual) Layer



- The conceptual model is a logical representation of the entire contents of the database.
- The conceptual model is made up of base tables.
- Base tables are "real" in that they contain physical records.

External View

- The user/application see
  - authorised data
  - own format



External View (cont.)

- External views allow to
  - hide unauthorised data
    - e.g. salary, dob
  - provide user view
    - e.g. view employee name, designation, department data taken from employee and department files
  - derive new attributes
    - e.g. age derived from dob

Example: University Database

- Conceptual schema:
  - Students(sid:string, name:string, login:string, age:integer, gpa:real)
  - Courses(cid:string, cname:string, credits:integer)
  - Enrolled(sid:string, cid:string, grade:string)
- Physical schema:
  - Relations stored as unordered files
  - Index on first column of Students
- External Schema (View):
  - Course_info(cid:string,enrollment:integer)

Data Independence

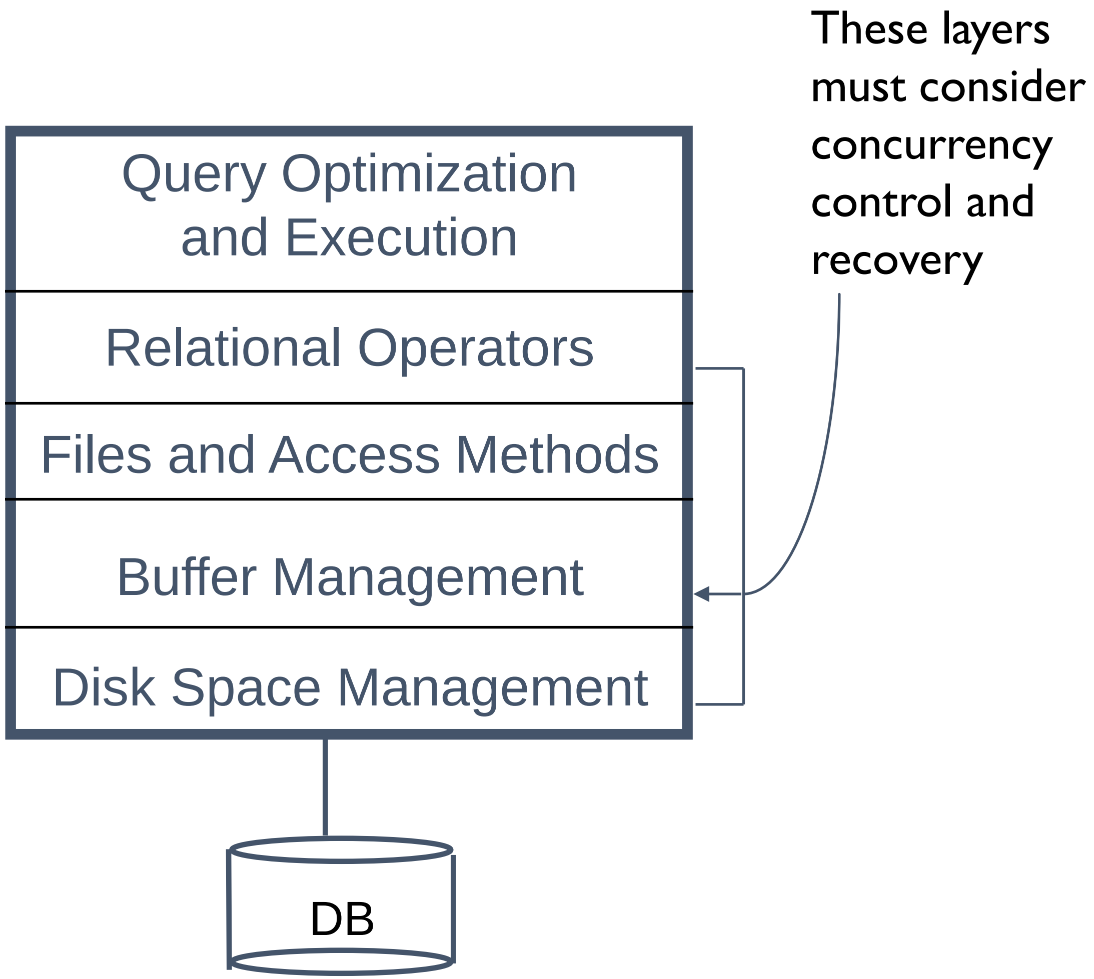
- Applications insulated from how data is structured and stored

- **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.

One of the most important benefits of using a DBMS!

Structure of a Database System

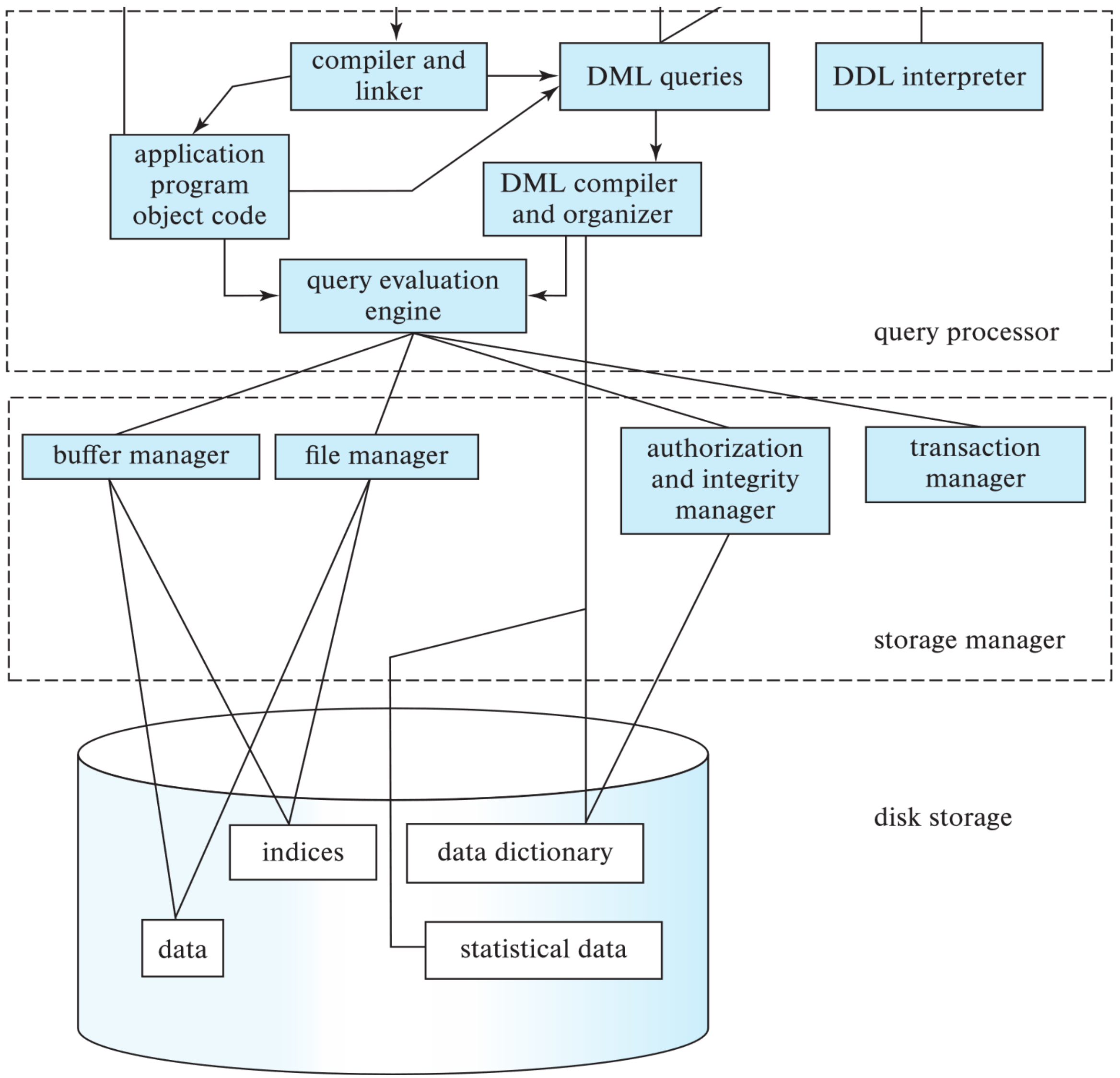
- A typical database system has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- This is one of several possible architectures; each system has its own variations.



Database Architecture

- Centralized databases
  - One to a few cores, shared memory
- Client-server,
  - One server machine executes work on behalf of multiple client machines.
- Parallel databases (will be discussed in BBM471)
  - Many core shared memory
  - Shared disk
  - Shared nothing
- Distributed databases (will be discussed in BBM471)
  - Geographical distribution
  - Schema/data heterogeneity

Database Architecture (Centralized/Shared-Memory)



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 updating the data organized by the appropriate data model
  - DML also known as query language
- There are basically two types of data-manipulation language
  - **Procedural DML** -- require a user to specify what data are needed and how to get those data.
  - **Declarative DML** -- require a user to specify what data are needed without specifying how to get those data.
- Declarative DMLs are usually easier to learn and use than are procedural DMLs.
- Declarative DMLs are also referred to as non-procedural DMLs
- The portion of a DML that involves information retrieval is called a **query** language.

SQL Query Language

- SQL query language is nonprocedural. A query takes as input several tables (possibly only one) and always returns a single table.
- Example to find all instructors in Comp. Sci. dept

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.'

- 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

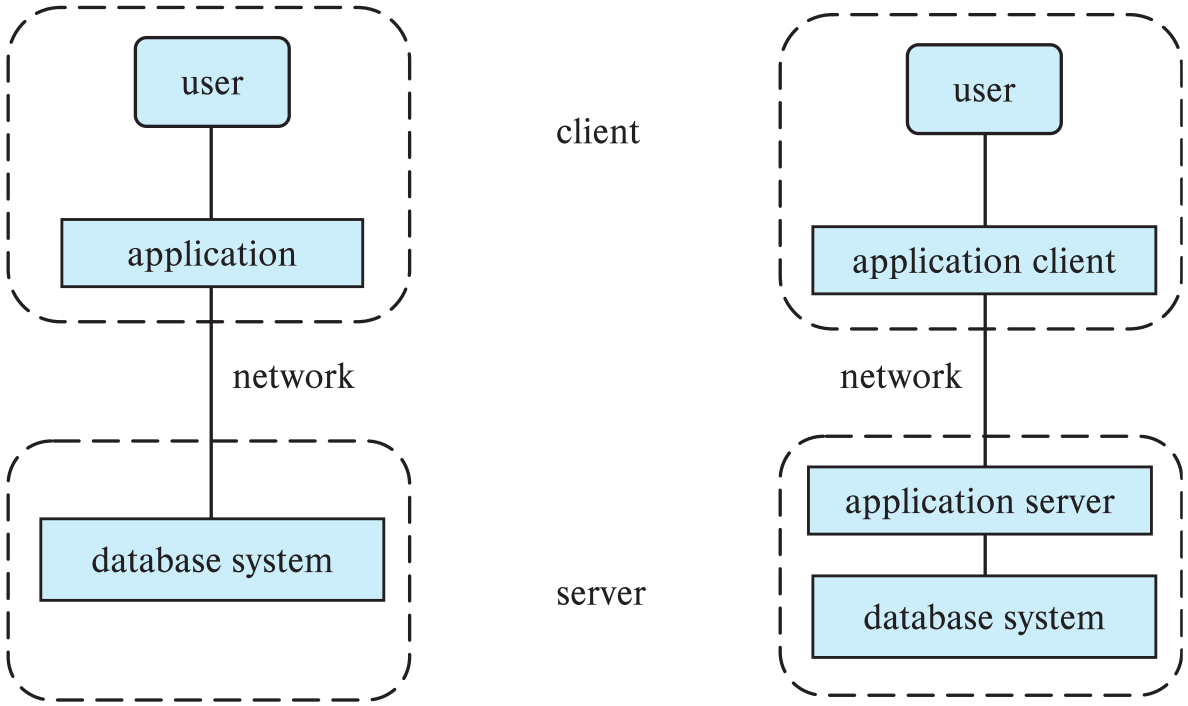
Database Access from Application Program

- Non-procedural query languages such as SQL are not as powerful as a universal Turing machine.
- SQL does not support actions such as input from users, output to displays, or communication over the network.
- Such computations and actions must be written in a **host language**, such as C/C++, Java or Python, with embedded SQL queries that access the data in the database.
- **Application programs** -- are programs that are used to interact with the database in this fashion.

Database Applications

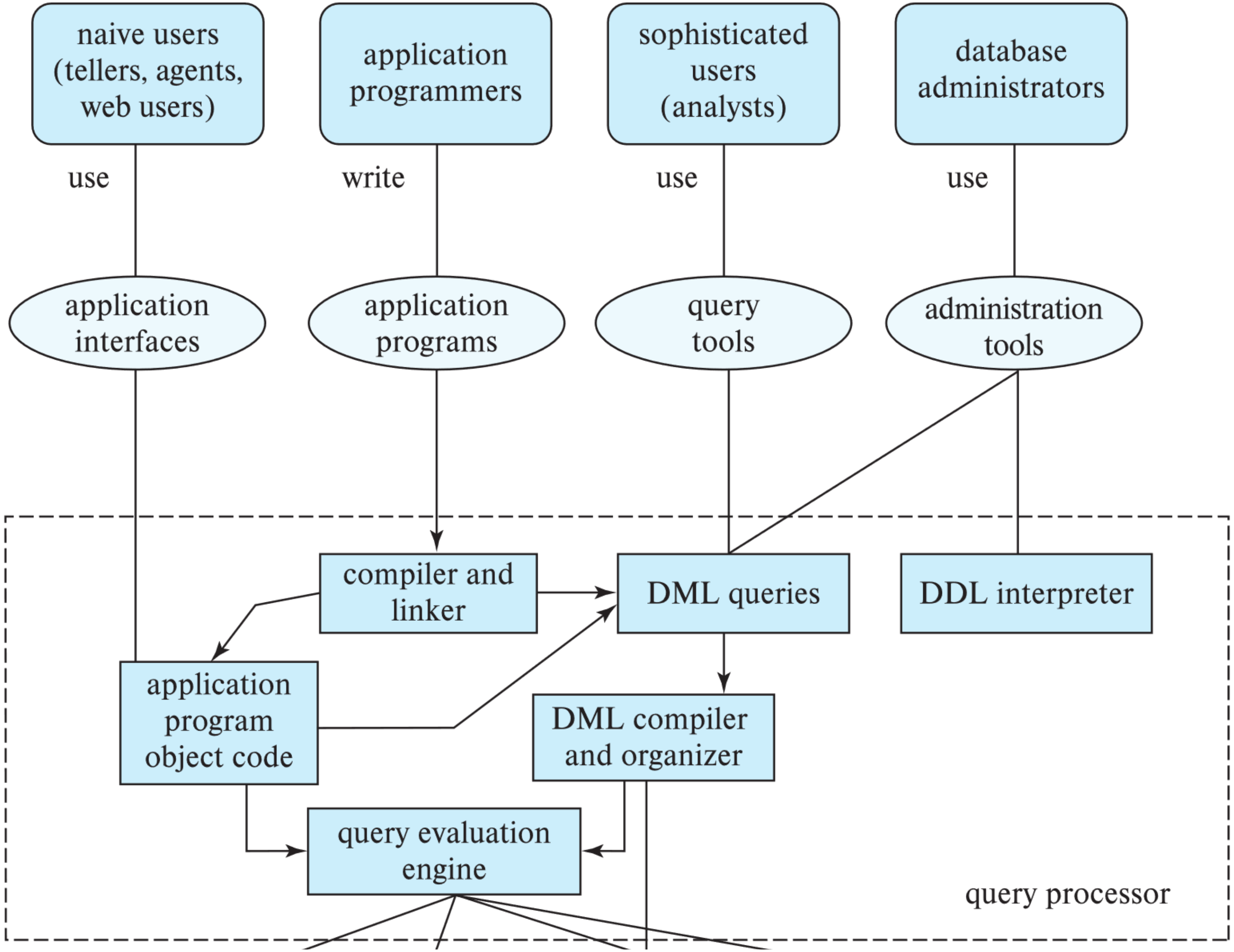
Database applications are usually partitioned into two or three parts

- (a) Two-tier architecture -- the application resides at the client machine, where it invokes database system functionality at the server machine
- (b) Three-tier architecture -- the client machine acts as a front end and does not contain any direct database calls.
  - The client end communicates with an application server, usually through a forms interface.
  - The application server in turn communicates with a database system to access data.



(a) Two-tier architecture (b) Three-tier architecture

Database Users



An Overview of Database Concepts

- **Database Design**
  - Entity Relationship (ER) Diagrams ★
  - Data Definition Language (DDL) ★
  - Normalization / Functional Dependencies ❯
  - DBMS Models ❯
- **Querrying**
  - Structured Query Language (SQL) ★
  - Relational Query Languages (RA, TRC, DRC) ❯
- **Physical Storage**
  - Utilizing the Memory Hierarchy (Buffering) ★
  - How to Store Data in Files ★
  - Finding data fast; Indexing Structures ★
  - External Sorting ★
- **Transaction Management**
  - Query Optimization ❯
  - Concurrency Control ❯
  - Crash Recovery ❯

★ Covered in this course

❯ Covered in BBM 471