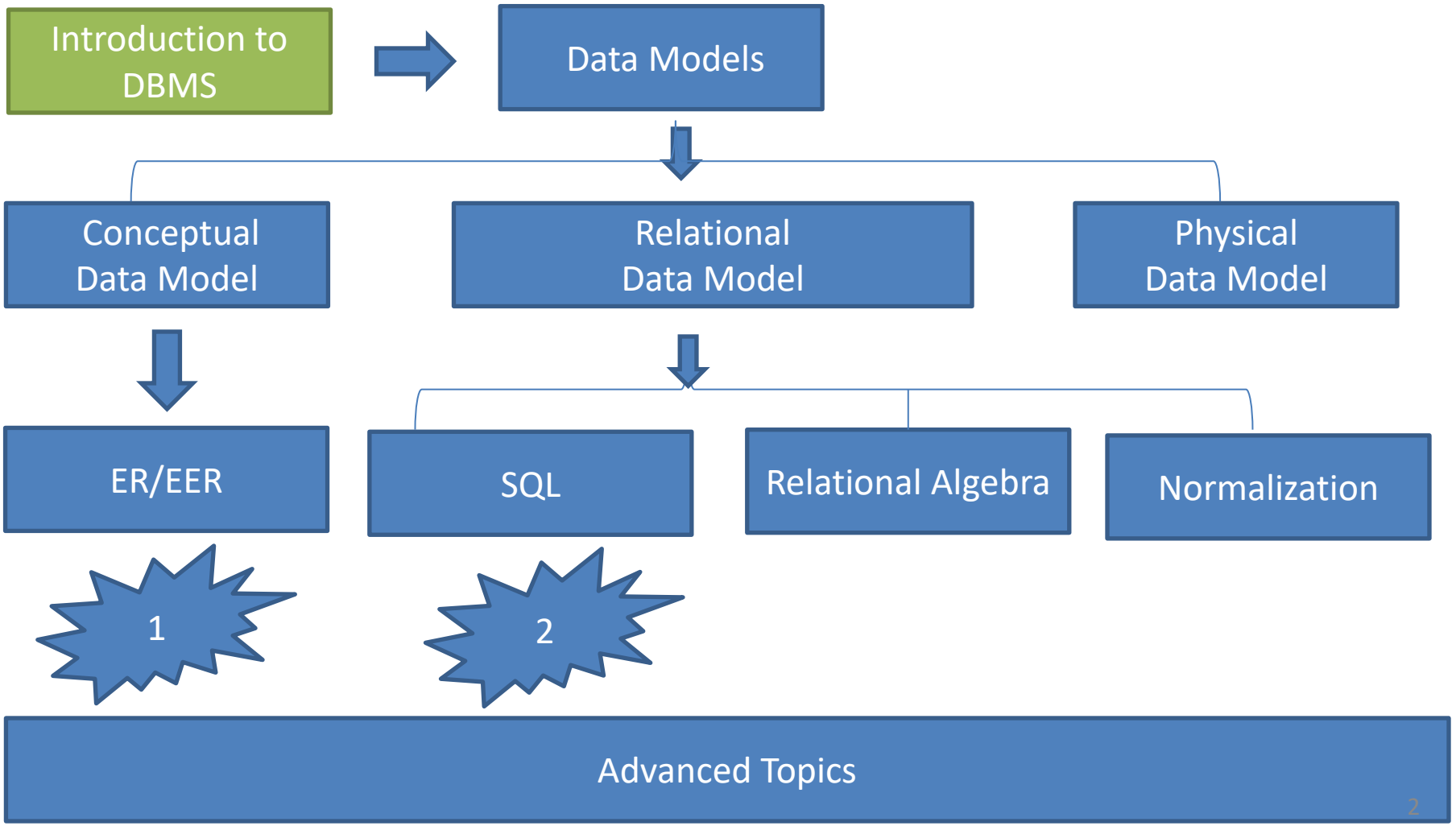
An Introduction to the Database Management Systems

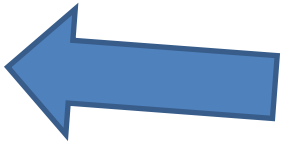
By
Hossein Rahmani

Slides originally by Book(s) Resources

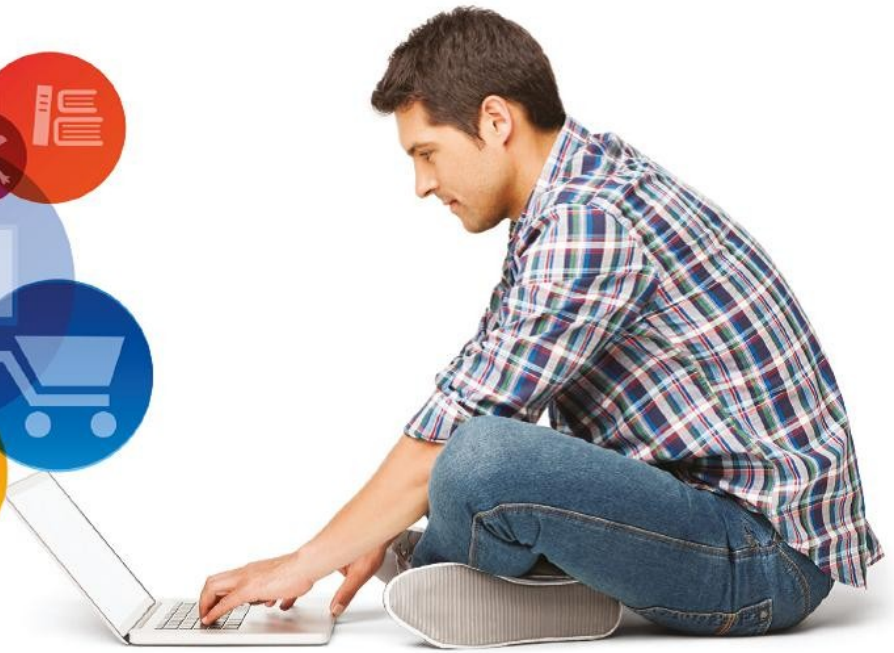
Road Map


Introduction to DBMS
Data Models
Conceptual Data Model
Relational Data Model
Physical Data Model
ER/EER
SQL
Relational Algebra
Normalization
1
2
Advanced Topics

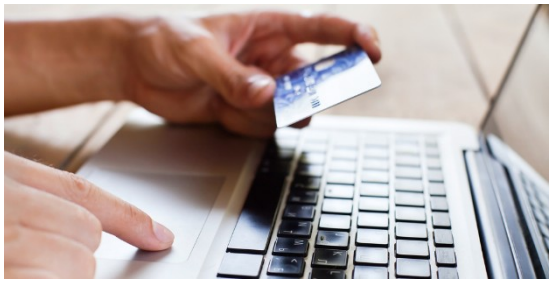
Introduction to Databases

- Introduction 
- An Example
- Characteristics of the Database Approach
- Actors on the Scene
- Workers behind the Scene
- Advantages of Using the DBMS Approach
- A Brief History of Database Applications
- When Not to Use a DBMS

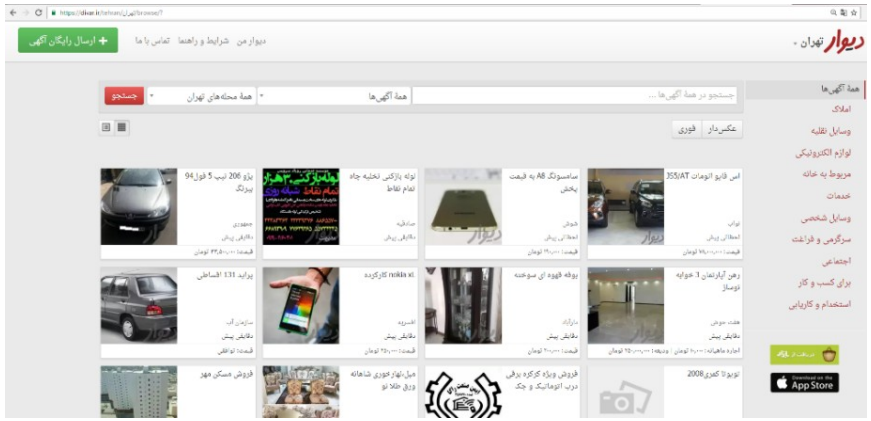
Welcome to Information Technology Era

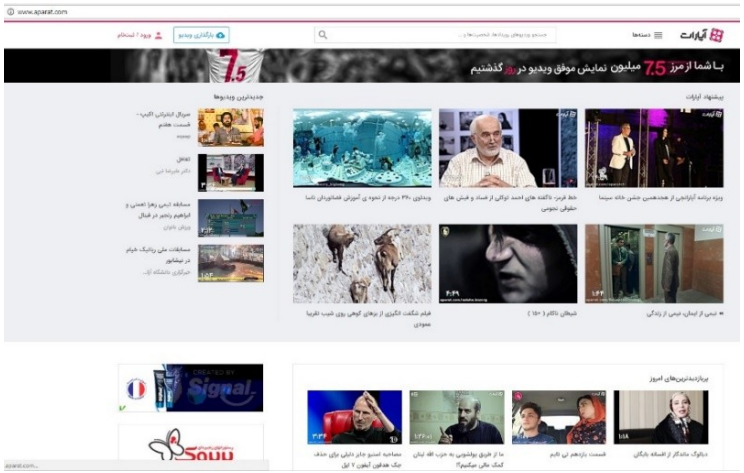


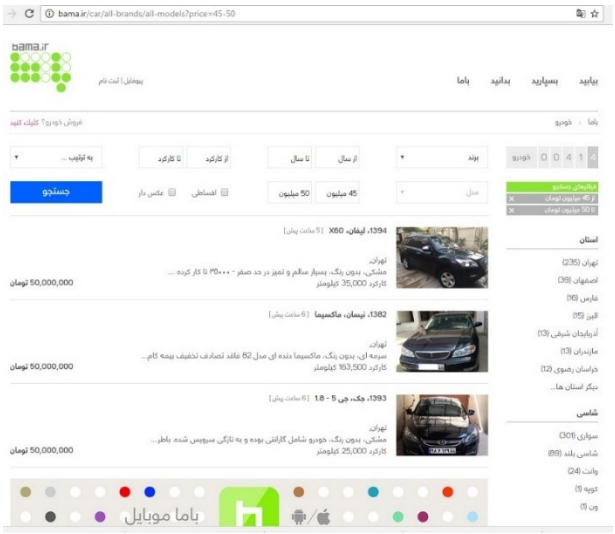
Welcome to Information Technology Era!



Online applications
what you do not see in the behind!





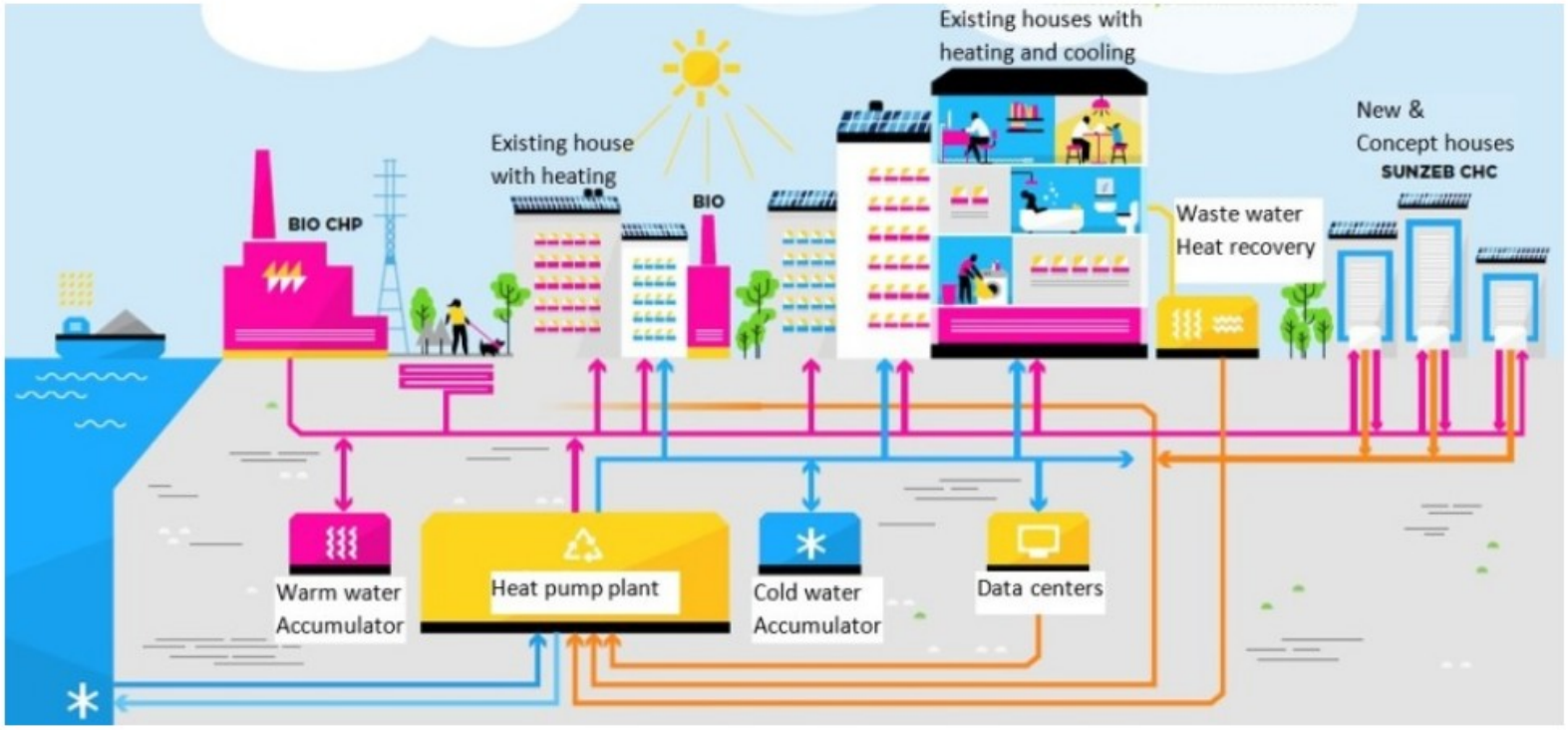


what you do not see in the behind!

What you do not see in the behind!

Helsinki, city with award winning district heating/cooling system

Excess heat of data centers in Helsinki is **circulated** to households via heat pumps and



Data and Information

DATA: *Facts concerning people, objects, vents or other entities. Databases store data.*

INFORMATION: *Data presented in a form suitable for interpretation.*

KNOWLEDGE: *Insights into appropriate actions based on interpreted data.*

Data

- Data **are** raw facts and figures that on their own have no meaning

- These can be any alphanumeric characters i.e. text, numbers, symbols

Data Examples

- Yes, Yes, No, Yes, No, Yes, No, Yes
- 42, 63, 96, 74, 56, 86
- 111192, 111234

- None of the above data sets have any meaning until they are given a **CONTEXT** and **PROCESSED** into a useable form

Data Into Information

- To achieve its aims the organisation will need to **process** data into information.
- Data needs to be turned into meaningful information and presented in its most useful format
- Data must be processed in a **context** in order to give it meaning

Information

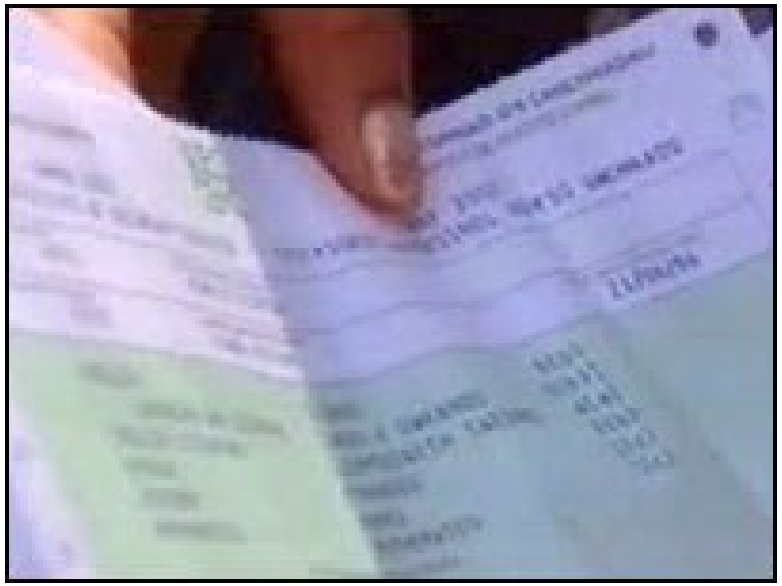
- Data that has been processed within a context to give it meaning

OR

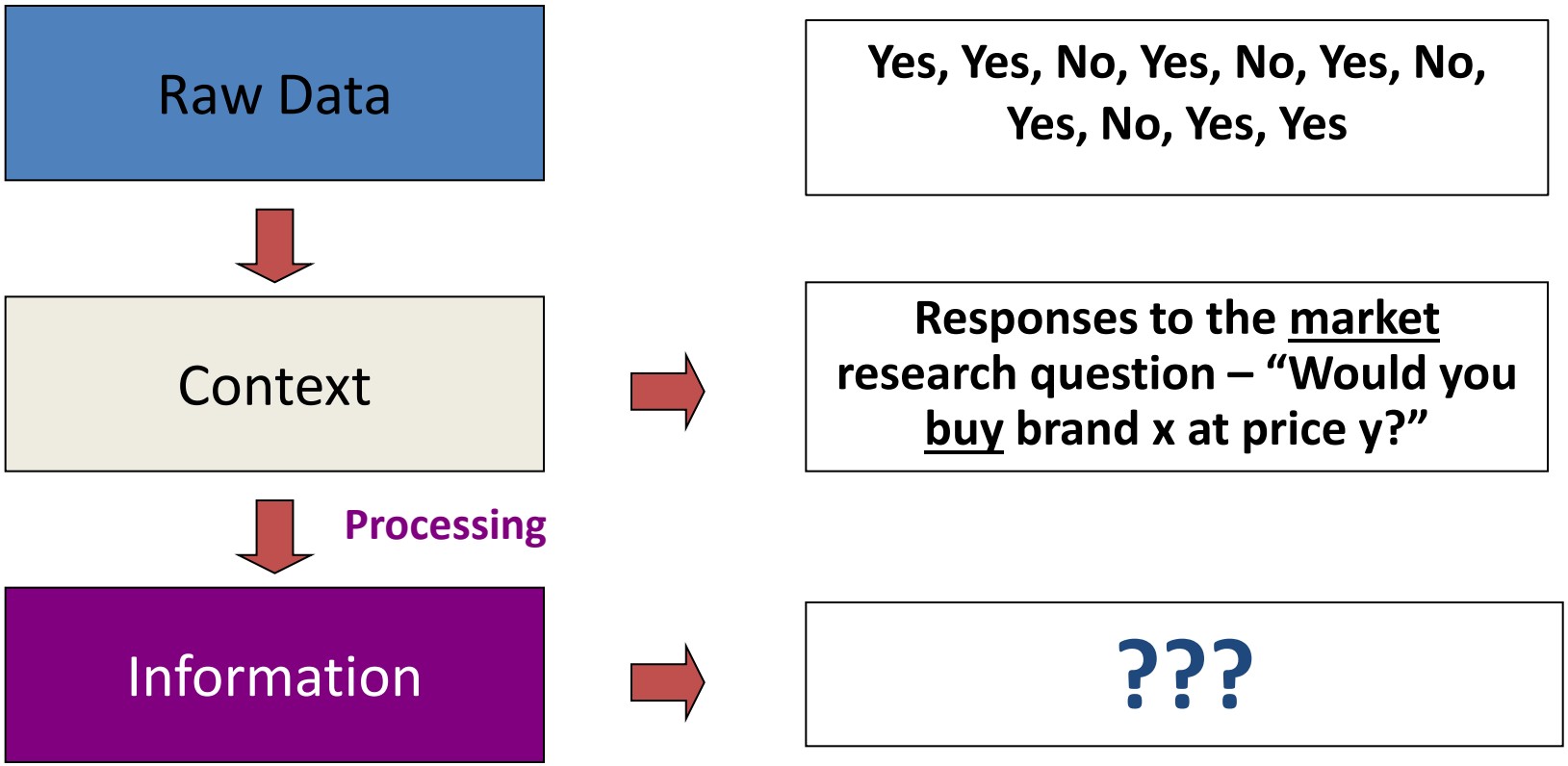
- Data that has been processed into a form that gives it meaning

Examples

- In the next 2 examples explain how the data could be processed to give it meaning
- What information can then be derived from the data?

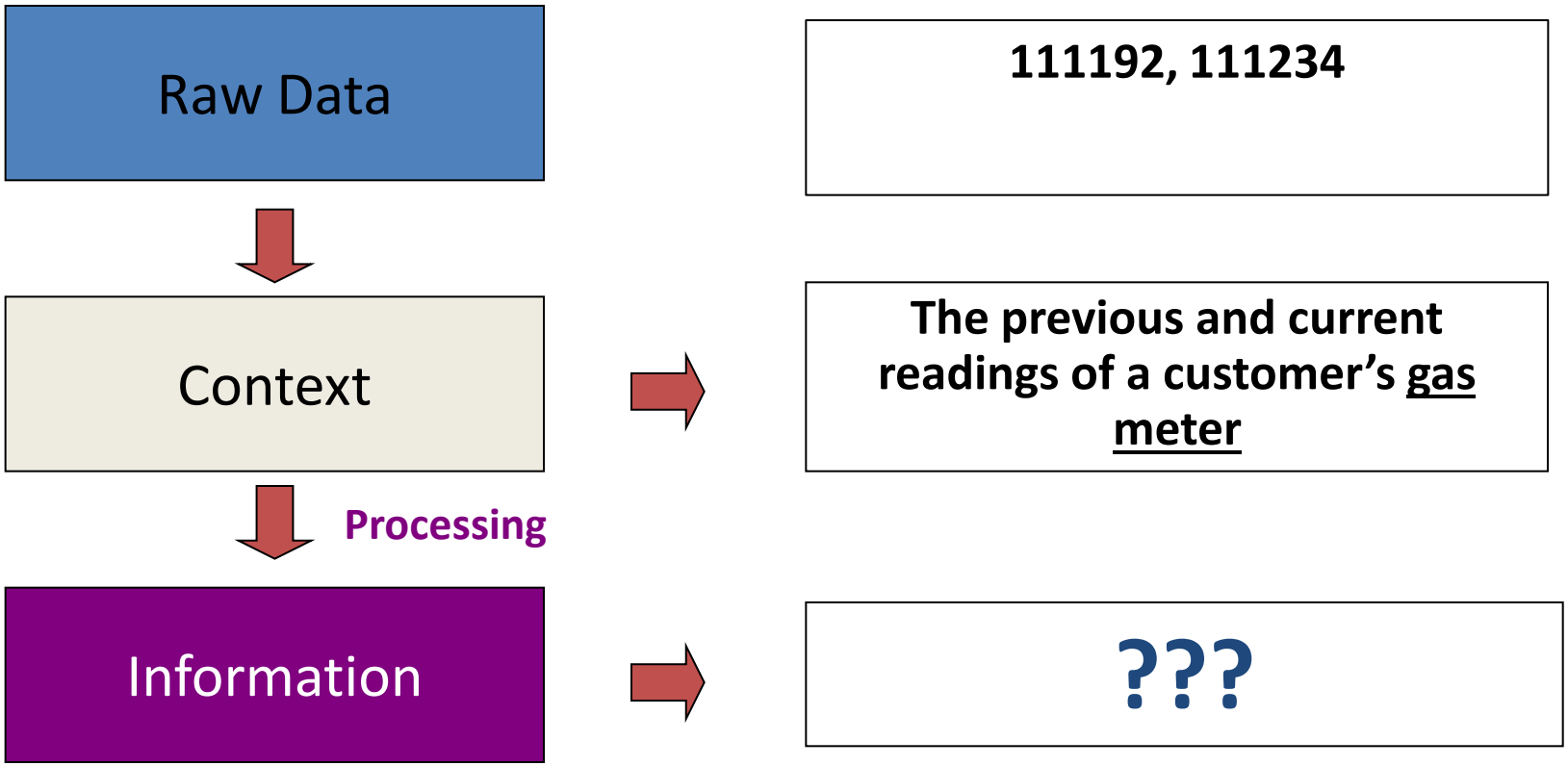


Example 1


Raw Data
Context
Processing
Information
Yes, Yes, No, Yes, No, Yes, No, Yes, No, Yes, Yes
Responses to the market research question – "Would you buy brand x at price y?"
???

- We could add up the yes and no responses and calculate the percentage of customers who would buy product X at price Y. The information could be presented as a chart to make it easier to understand.

Example 2


Raw Data
Context
Processing
Information
111192, 111234
The previous and current readings of a customer's gas meter
???

By subtracting the second value from the first we can work out how many units of gas the consumer has used.
This can then be multiplied by the price per unit to determine the customer's gas bill.

Knowledge

- Knowledge is the understanding of rules needed to interpret information

"…the capability of understanding the relationship between pieces of information and what to actually do with the information"

Debbie Jones – www.teach-ict.com

Knowledge Examples

- Using the 2 previous examples:
  - A Marketing Manager could use this information to decide whether or not to raise or lower price y
  - Looking at the pattern of the customer's previous gas bills may identify that the figure is abnormally low and they are fiddling the gas meter!!!

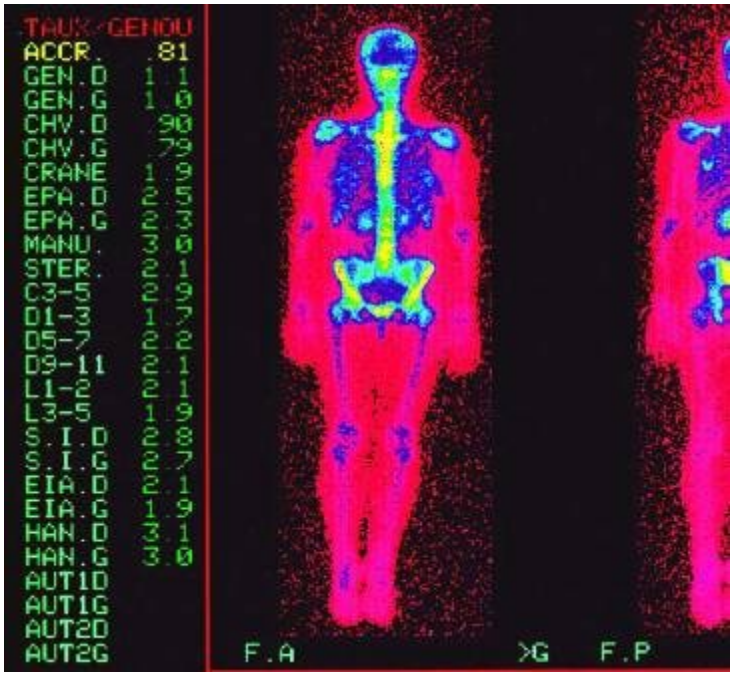
Not working factory

Knowledge Workers

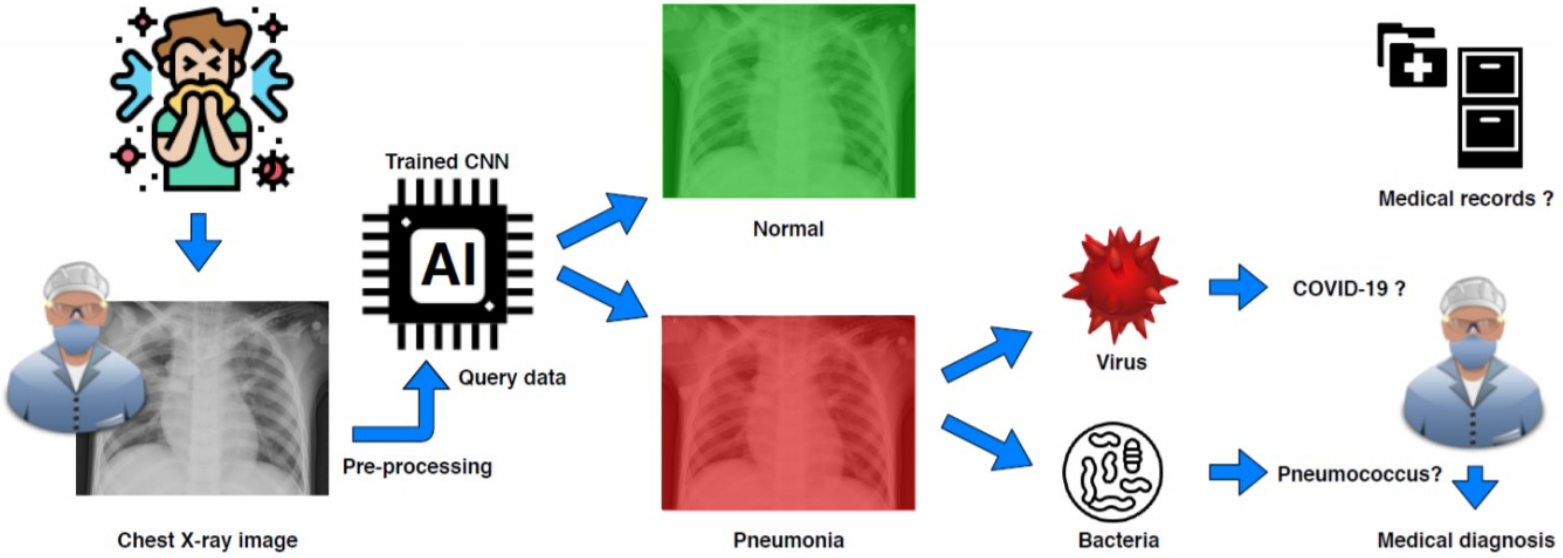
- Knowledge workers have specialist knowledge that makes them "experts"
  - Based on formal and informal rules they have learned through training and experience
- Examples include doctors, managers, librarians, scientists…

Expert Systems

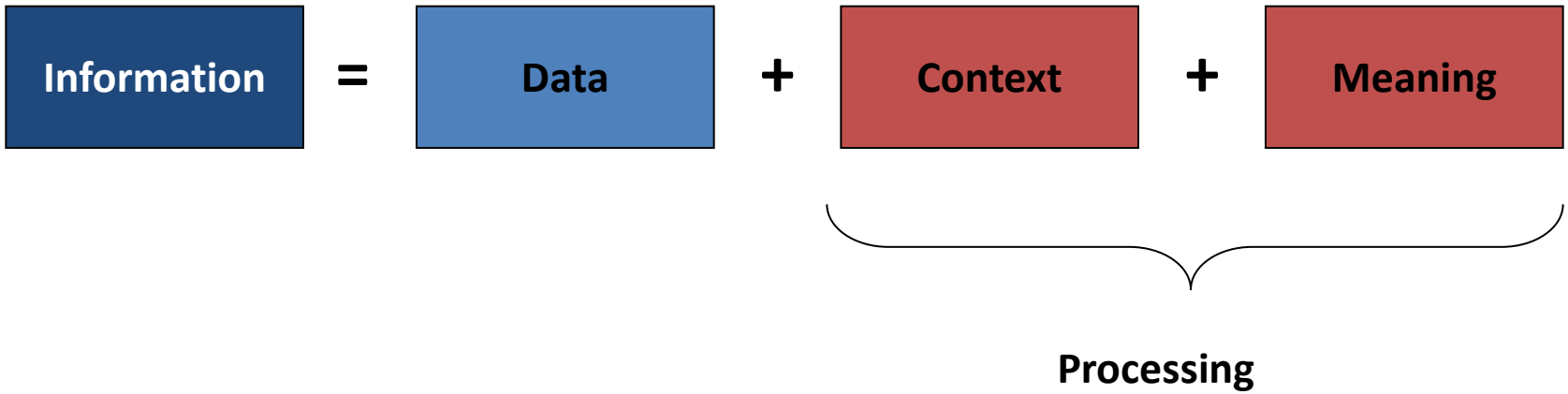
- Because many rules are based on probabilities computers can be programmed with “subject knowledge” to mimic the role of experts
- One of the most common uses of expert systems is in medicine
  - The ONCOLOG system shown here analyses patient data to provide a reference for doctors, and help for the choice, prescription and follow-up of chemotherapy



Expert System
covid 19!



Summary



Data – raw facts and figures

Information – data that has been processed (in a context) to give it meaning

Ever Growing Databases

- Twitter: Generate approximately 12 TB of data per day.
- Facebook: Facebook data grows by over 500 TB daily.
- New York Stock: Exchange 1TB of data everyday

Online social network

- Over 900 million Facebook users worldwide
  - Over 150 million in U.S.
  - Over 450 million access via mobile
  - 300 million pictures uploaded to Facebook daily
- Over 140 million Twitter users; over 340 million Tweets sent daily
- Over 175 million LinkedIn members in over 200 countries

Overview

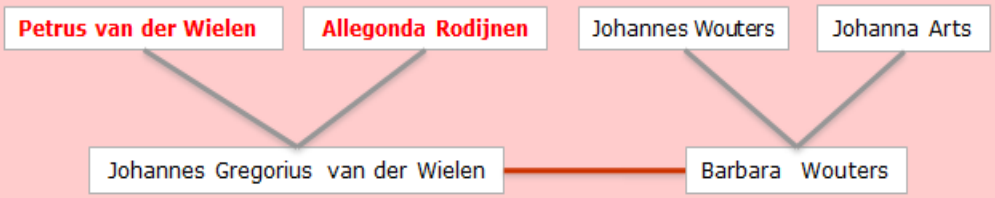
- **Traditional database applications**
  - Store textual or numeric information
- **Multimedia databases**
  - Store images, audio clips, and video streams digitally
- **Geographic information systems (GIS)**
  - Store and analyze maps, weather data, and satellite images

Traditional textual Databases (Dutch Civil Registers)

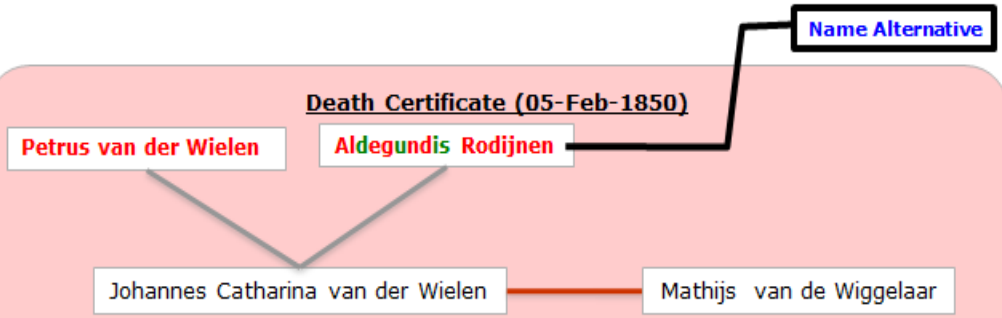
Property Transfer (14-Jan-1805)

Petrus van der Wielen en zijn vrouw **Allegonda Rodijnen**, woonachtig in Grave verklaren voor notaris W. Smits aldaar dat zij 12.023 gulden en 3 stuivers geleend hebben van Mr. Edm. Ruijs en zijn vrouw Clara M.P. van Aefferden. Getuigen: Mr. A.J.J.H. Verheijen, amptman van Grave, en Peter le Muller. Geprotocolleerde toevoegingen: - Op verzoek van Mr. Edm. Ruijs, wonend te Grave, wordt op vertoon van bovenstaande akte, deze geregistreerd te Velp op 15.01.1805 door Benj.A. de Jong, secretaris. In de marge is nog vermeld dat de akte na betaling van de 40e penning op 14.01.1805 te Grave is geregistreerd in het Protocol van Veste en Verbanden. Het geheel geregistreerd op 24.04.1805 door E. van Rooij, beëdigd klerk.

Marriage Certificate (16-Feb-1817)



Death Certificate (05-Feb-1850)



Property Transfer (12-Feb-1805)

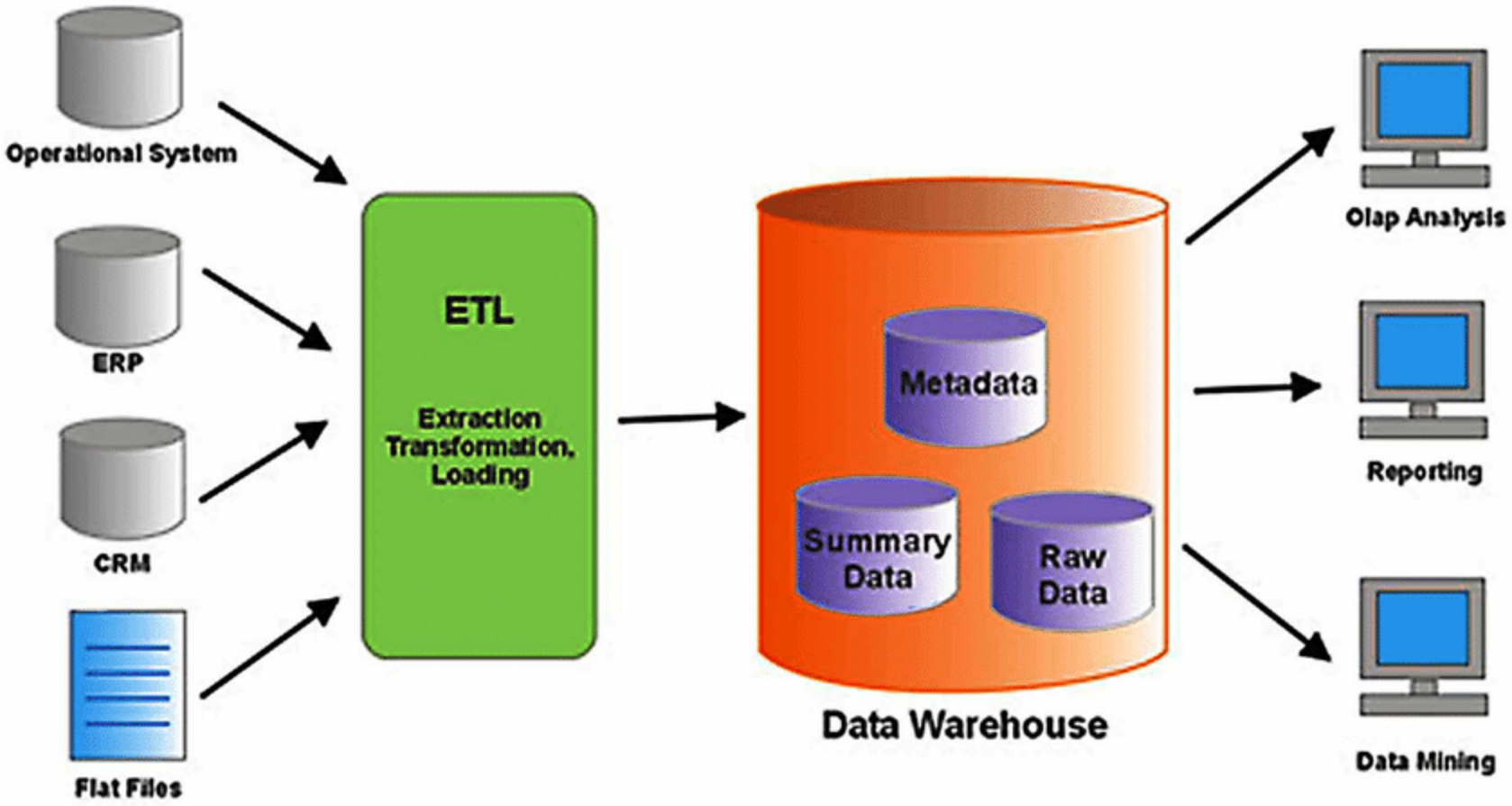
Petrus van der Wielen en zijn vrouw **Allegonda Rodijnen**, woonachtig in Grave verklaren voor notaris Wijnand Smits aldaar dat zij van Olimpia Knoop, wonend te 's-Hertogenbosch. 1120 gulden geleend hebben. Getuigen: P.H. Heimach en J.T. Esmans. Geprotocolleerde toevoegingen: - Bovenstaande akte na bewijs van betaling van de 40e penning geregistreerd in het Prothocol van Vesten en Verbandenż ter secretarie van Grave op 14.02.1805 door W. Smits, secr. Embertus J. van Rooij verzoekt de schepenen van Reek op vertoon van een kopie van de obligatie deze te registreren. Aldus geschiedt op 09.11.1805. Het geheel geregistreerd op 14.11.1805 te Velp door Benj. A. de Jong, secr. In de kantlijn: Arreté le présent Registre le 12. Juillet 1811. De Muller.

(Dutch) Name Alternatives

Hendrik	Ghendrik, Haendrik, Handrik, Handrikus, Hanri, Hanricus, Hdendrik, Hdndrikus, Hebdricus, Hebdrik, Hebdrikus, Hebndrik, Hederikus, Hednerikus, Hednriekus, Hednrik, Hednrikus, Hedricus, Hedrik, Hedrikkus, Hedrikus, Heemerik, Heenderikus, Heendrik, Heeninck, Heenrich, Heijn, Heijnderik, Heijndrik, Heijnk, Heimich, Heinderijcus, Heinderik, Heindich, Heindrich, Heindrick, Heindricus, Heinich, Heinirch, Heink, Heinrch, Heinreich, ...
Gerrit	Gaarit, Gaarts, Gadus, Gaerrit, Ganes, Garard, Garardus, Garet, Garhard, Garhardus, Garijt, Garit, Garretjan, Garrie, Garrijt, Garrir, Garris, Garrit-jan, Garritjan, Garrits, Garrti, Garrut, Garryt, Gearardus, Geard, Geardus, Gearrdus, Gebhart, Gedrit, Geeerts, Geer, Geeraardse, Geeraardt, Geeraart, Geeraerdus, Geerard, Geerardt, Geerardus, Geerart, Geerdis, Geerds, Geerdt, Geerdus, Geeret, Geerhardus, Geerid, Geerie, Geerijt, Geerrad, ...
Maria	Amria, Ma;ria, Maaaike, Maaartje, Maaatje, Maageje, Maagie, Maagja, Maagjen, Maagke, Maagtje, Maahje, Maaia, Maaiake, Maaie, Maaiek, Maaige, Maaigie, Maaigje, Maaiike, Maaij, Maaije, Maaijeke, Maaijge, Maaijgje, Maaijke, Maaijken, Maaijkje, Maaijtje, Maaik, Maaika, Maaiken, Maaikje, Maaikke, Maaile, Maailke, Maaitke, Maajke, Maaken, Maaltje, Maamk, Maamke?, Maanke, Maantje, Maardje, Maare, Maaregi, Maaretje, Maargje, ...

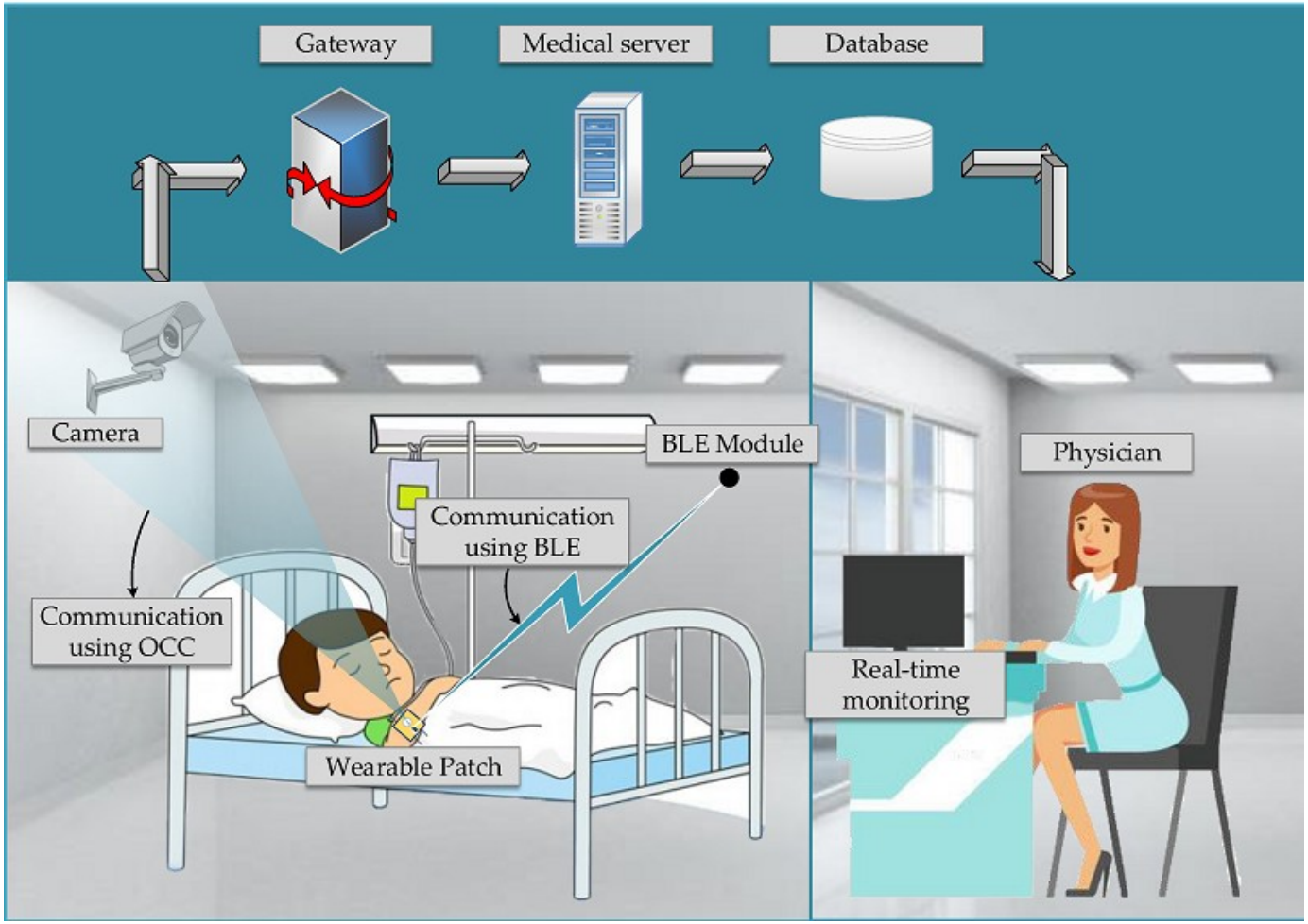
Overview (cont'd.)

- **Data warehouses and online analytical processing (OLAP) systems**
  - Extract and analyze useful business information from very large databases
  - Support decision making
- **Real-time and active database technology**
  - Control industrial and manufacturing processes

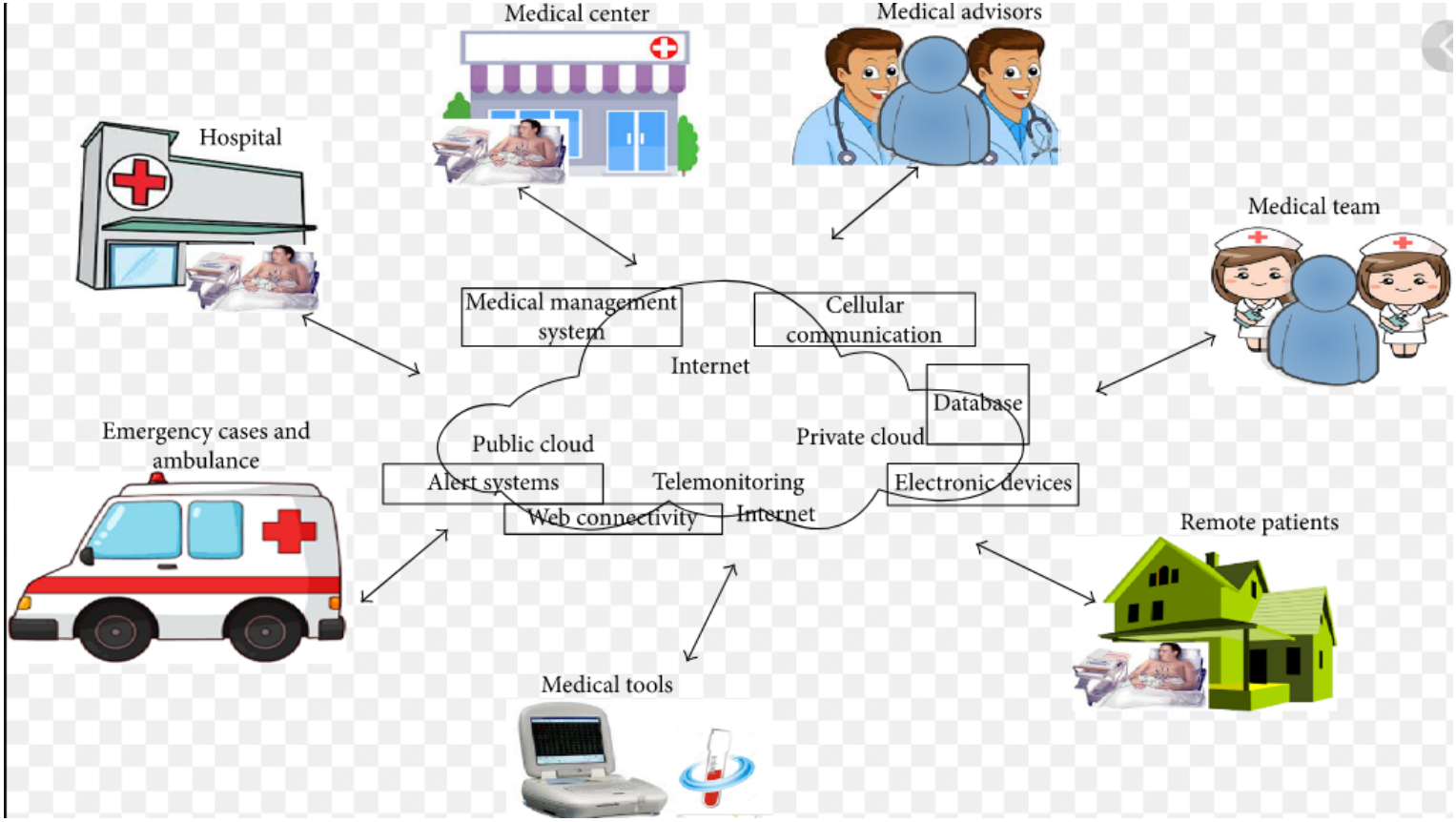

Operational System
ERP
CRM
Flat Files
ETL
Extraction
Transformation,
Loading
Metadata
Summary
Data
Raw
Data
Data Warehouse
Olap Analysis
Reporting
Data Mining

Real Time Databases

Real-time monitoring



Real-time monitoring



Introduction

- **Database**
  - Collection of related data
  - Known facts that can be recorded and that have implicit meaning
  - **Miniworld** or **universe of discourse (UoD)**
  - Represents some aspect of the real world
  - Logically coherent collection of data with inherent meaning
  - Built for a specific purpose

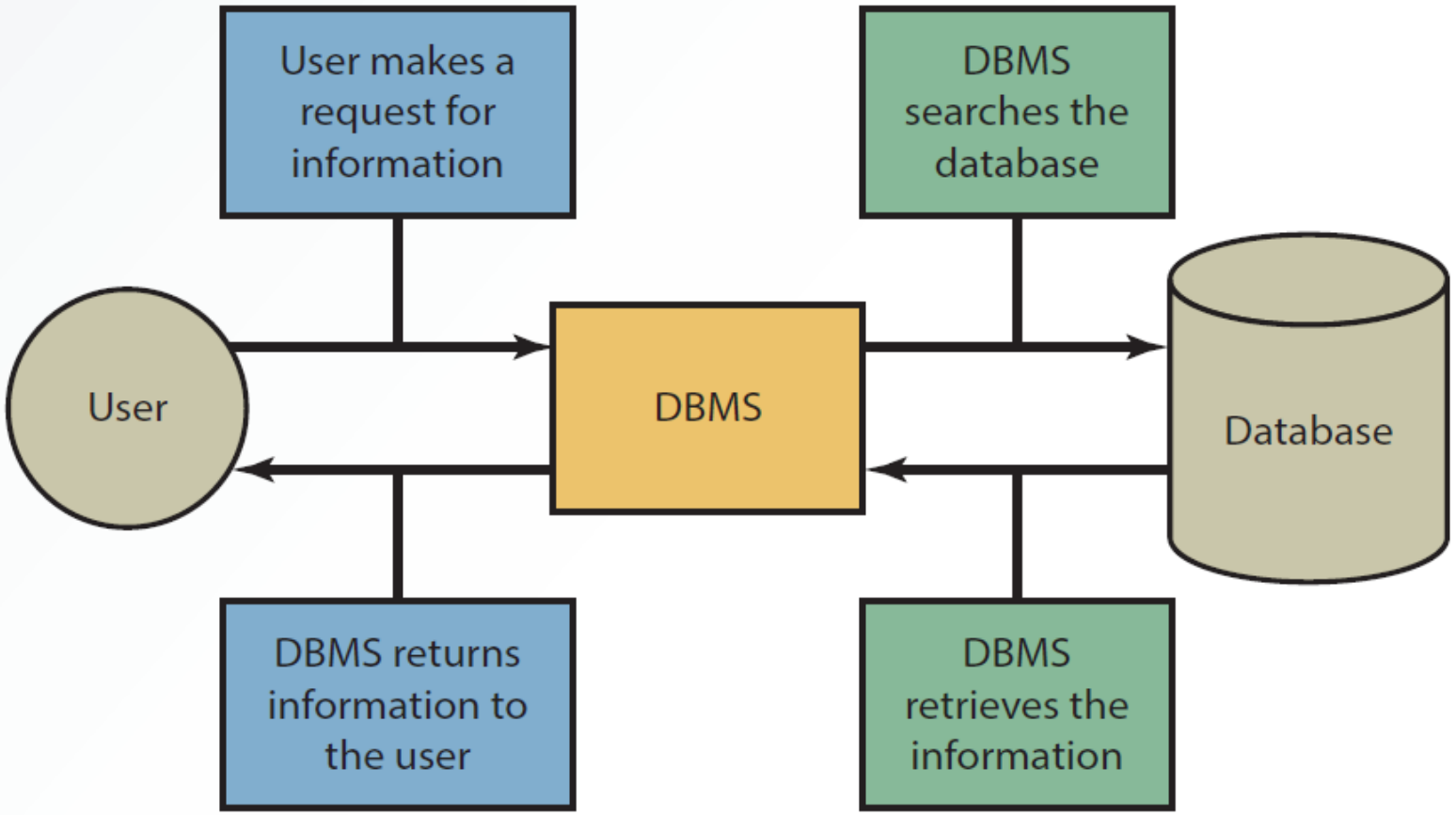
Introduction (cont'd.)

- Example of a large commercial database
  - Amazon.com, University, Company
- **Database management system (DBMS)**
  - Collection of programs
  - Enables users to create and maintain a database
- **Defining** a database
  - Specify the data types, structures, and constraints of the data to be stored

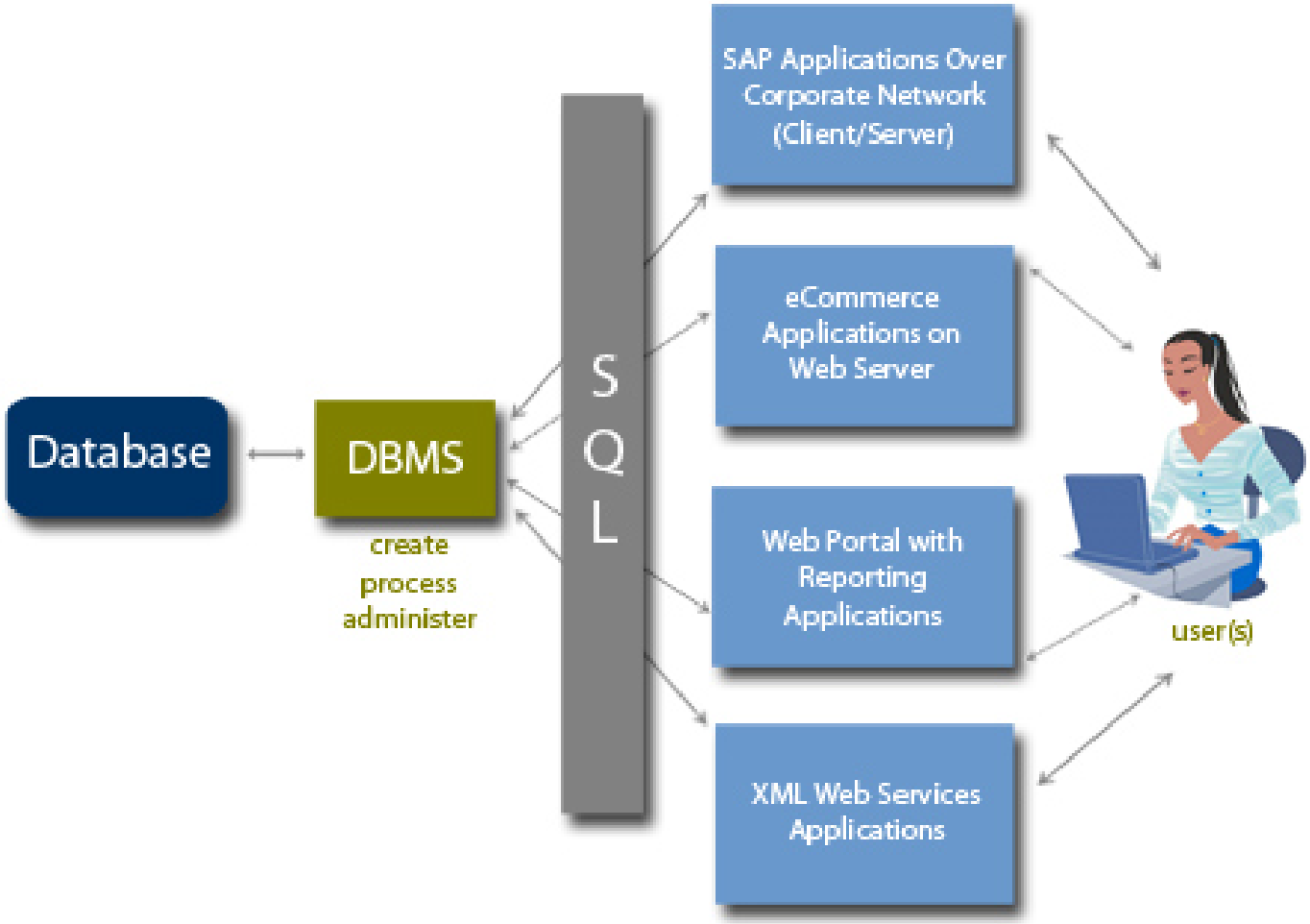
DBMS


Users
Server Applications
Client Applications
Access Data Via
DBMS

Exhibit 3.2 Interaction Between the User, DBMS and Database



DBMS


Database
DBMS
create
process
administer
S
Q
L
SAP Applications Over Corporate Network (Client/Server)
eCommerce Applications on Web Server
Web Portal with Reporting Applications
XML Web Services Applications
user(s)

Introduction (cont'd.)

- **Meta-data**
  - Database definition or descriptive information
  - Stored by the DBMS in the form of a database catalog or dictionary
- **Manipulating** a database
  - Query and update the database miniworld
  - Generate reports

Metadata

- The term "meta" comes from a Greek word that denotes something of a higher or more fundamental nature. Metadata, then,

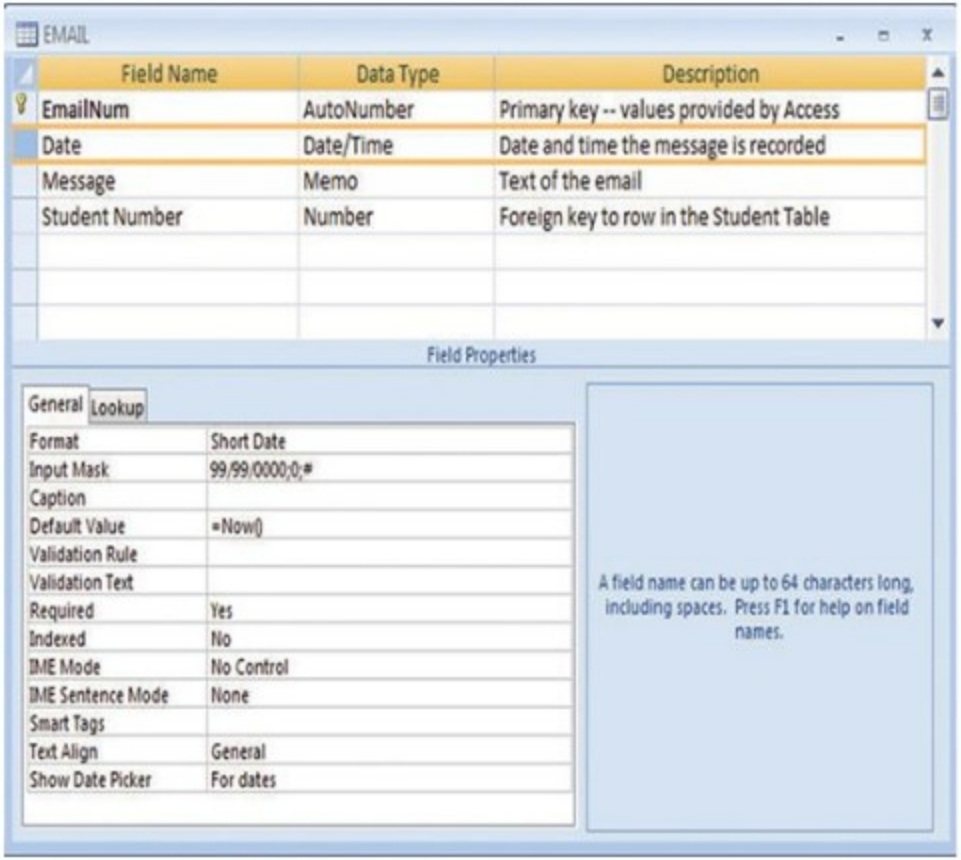
  is “data about other data”

Defining Metadata

- Does data about data mean anything?
  - Librarians equate it with a complete bibliographic record
  - Information technologists equate it to database schema or definitions of the data elements

Metadata

- **Database**:
  - A database is a self-describing collection of integrated records.
- **Metadata**
  - Data that describe data



Database Catalog

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerated type with all known majors.
XXXXNNNN is used to define a type with four alpha characters followed by four digits.

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

Introduction (cont'd.)

- **Sharing** a database
  - Allow multiple users and programs to access the database simultaneously
- **Application program**
  - Accesses database by sending queries to DBMS
- **Query**
  - Causes some data to be retrieved

Introduction (cont'd.)

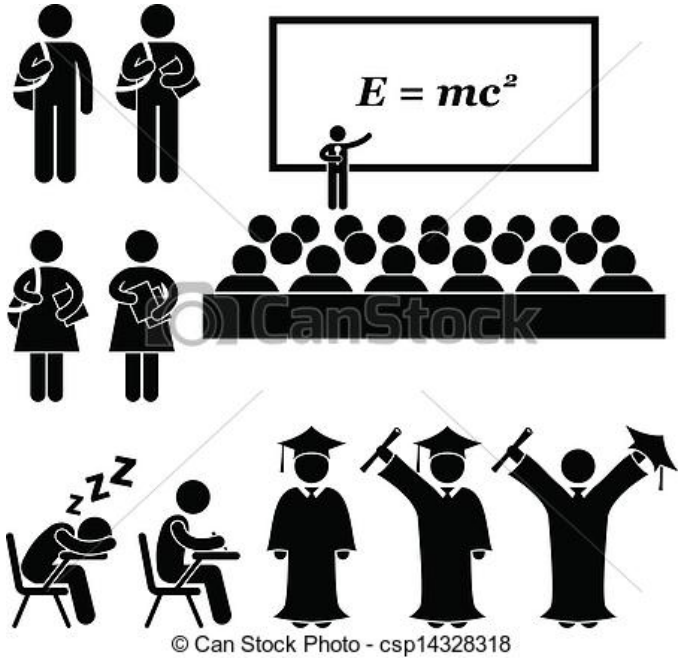
- **Transaction**
  - May cause some data to be read and some data to be written into the database
- **Protection** includes:
  - System protection
  - Security protection
- **Maintain** the database system
  - Allow the system to evolve as requirements change over time

Introduction to Databases

- Introduction
- An Example
- Characteristics of the Database Approach
- Actors on the Scene
- Workers behind the Scene
- Advantages of Using the DBMS Approach
- A Brief History of Database Applications
- When Not to Use a DBMS

An Example

- UNIVERSITY database
  - Information concerning students, courses, and grades in a university environment
- **Data records**
  - STUDENT
  - COURSE
  - SECTION
  - GRADE_REPORT
  - PREREQUISITE



An Example (cont'd.)

- Specify structure of records of each file by specifying **data type** for each **data element**
  - String of alphabetic characters
  - Integer
  - Etc.

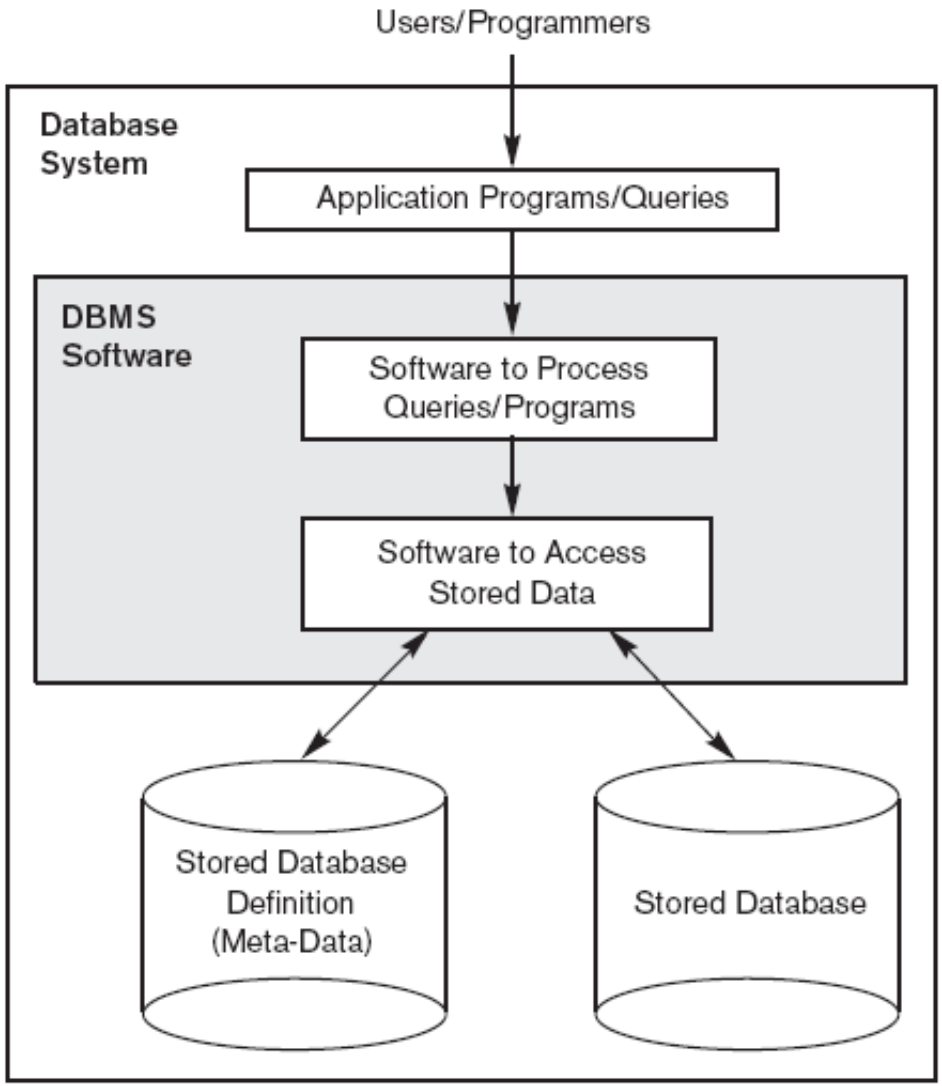


Figure 1.1
A simplified database system environment.

An Example (cont'd.)

- Construct UNIVERSITY database
  - Store data to represent each student, course, section, grade report, and prerequisite as a record in appropriate file
- Relationships among the records
- Manipulation involves querying and updating

An Example (cont'd.)

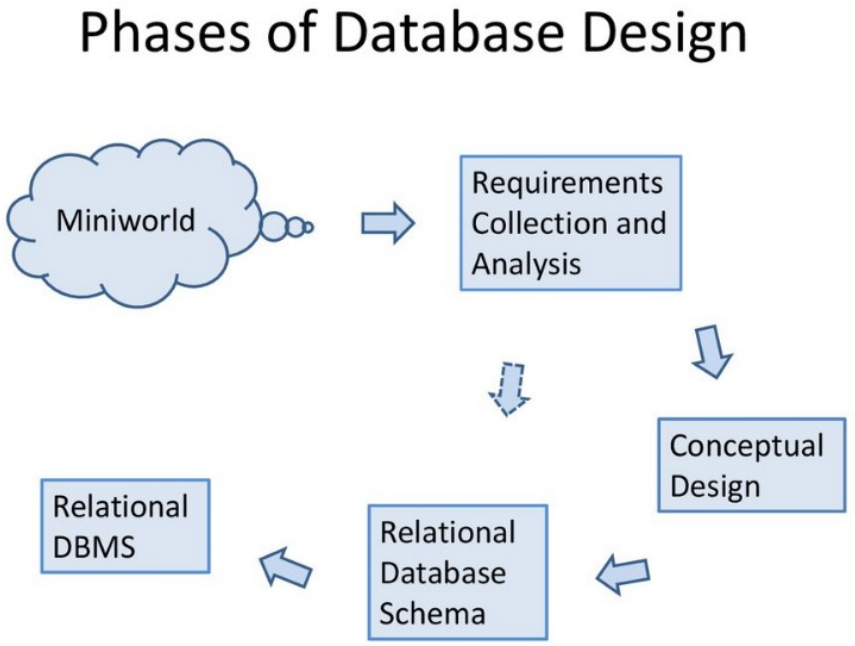
- Examples of queries:
  - Retrieve the transcript
  - List the names of students who took the section of the ‘Database’ course offered in fall 2008 and their grades in that section
  - List the prerequisites of the ‘Database’ course

An Example (cont'd.)

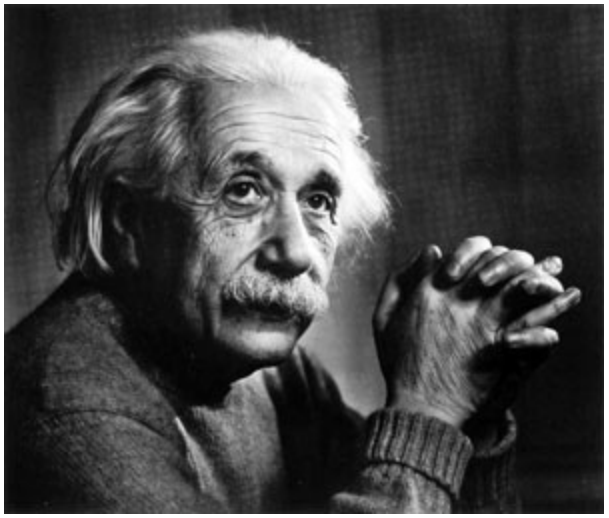
- Examples of updates:
  - Create a new section for the ‘Database’ course for this semester
  - Enter a grade of ‘A’ for ‘Smith’ in the ‘Database’ section of last semester

An Example (cont'd.)

- Phases for designing a database:
  - **Requirements specification and analysis**
  - **Conceptual design**
  - **Logical design**
  - **Physical design**



Requirement Engineering



if I have one hour to save the world I would spend *fifty-five minutes defining the problem and only five minutes finding the solution*.

Requirement Engineering

- Before jumping right into solving a problem, we should step back and invest time and effort to improve our understanding of it.

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

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.

Quiz 1

- Considering the database state shown in the previous slide, What is the <u>complete</u> meaning of the following tuples?

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
112	MATH2410	Fall	08	Chang

GRADE_REPORT

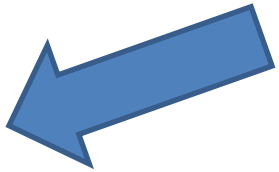
Student_number	Section_identifier	Grade
8	102	B

PREREQUISITE

Course_number	Prerequisite_number
CS3380	MATH2410

Introduction to Databases

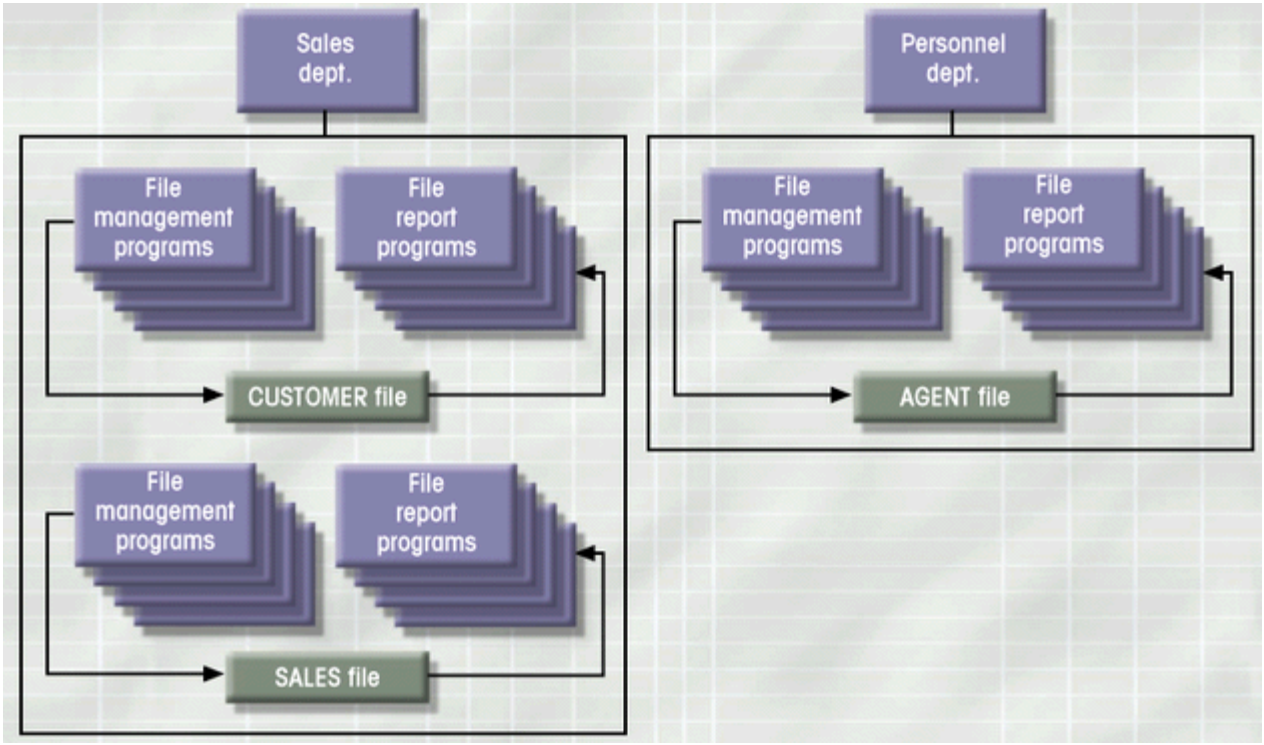
- Introduction
- An Example
- Characteristics of the Database Approach
- Actors on the Scene
- Workers behind the Scene
- Advantages of Using the DBMS Approach
- A Brief History of Database Applications
- When Not to Use a DBMS



Characteristics of the Database Approach

- Traditional **file processing**
  - Each user defines and implements the files needed for a specific software application
- Database approach
  - Single repository maintains data that is defined once and then accessed by various users

File System: Example

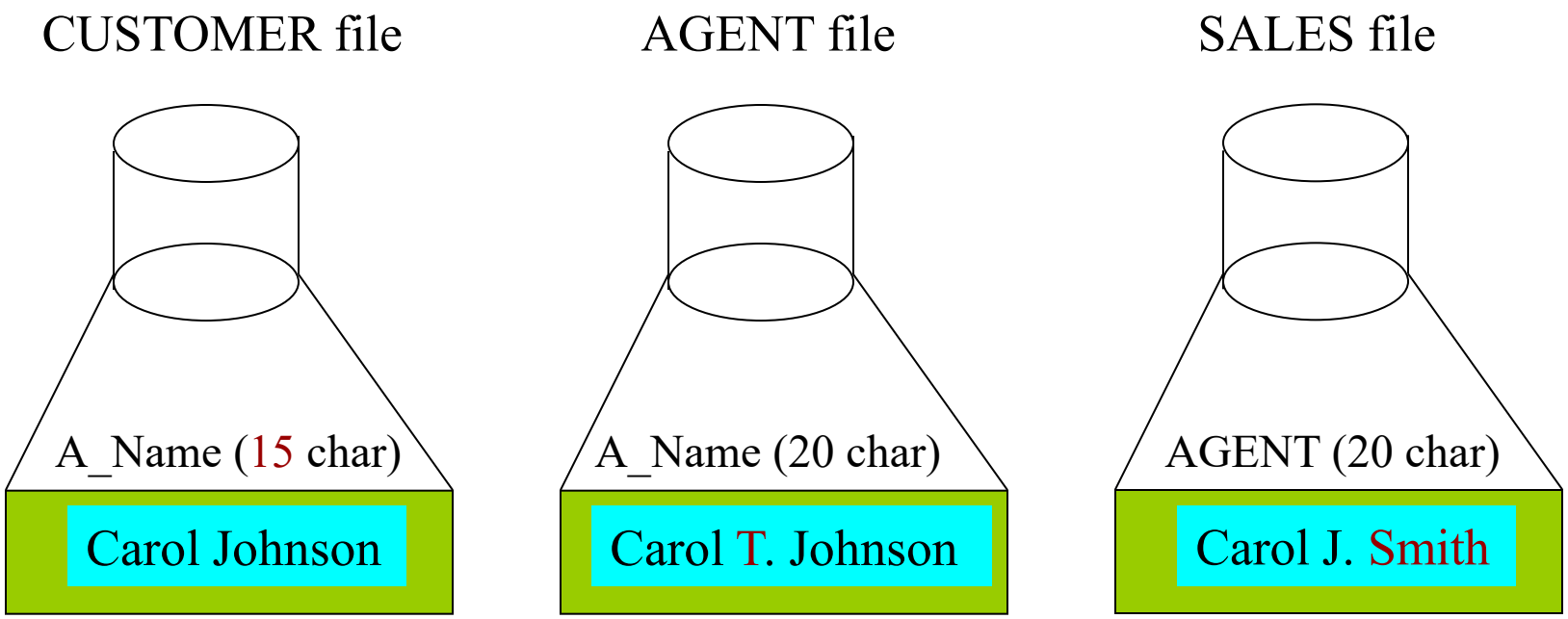


Database Systems: Design, Implementation, & Management: Rob & Coronel

File System: Weakness

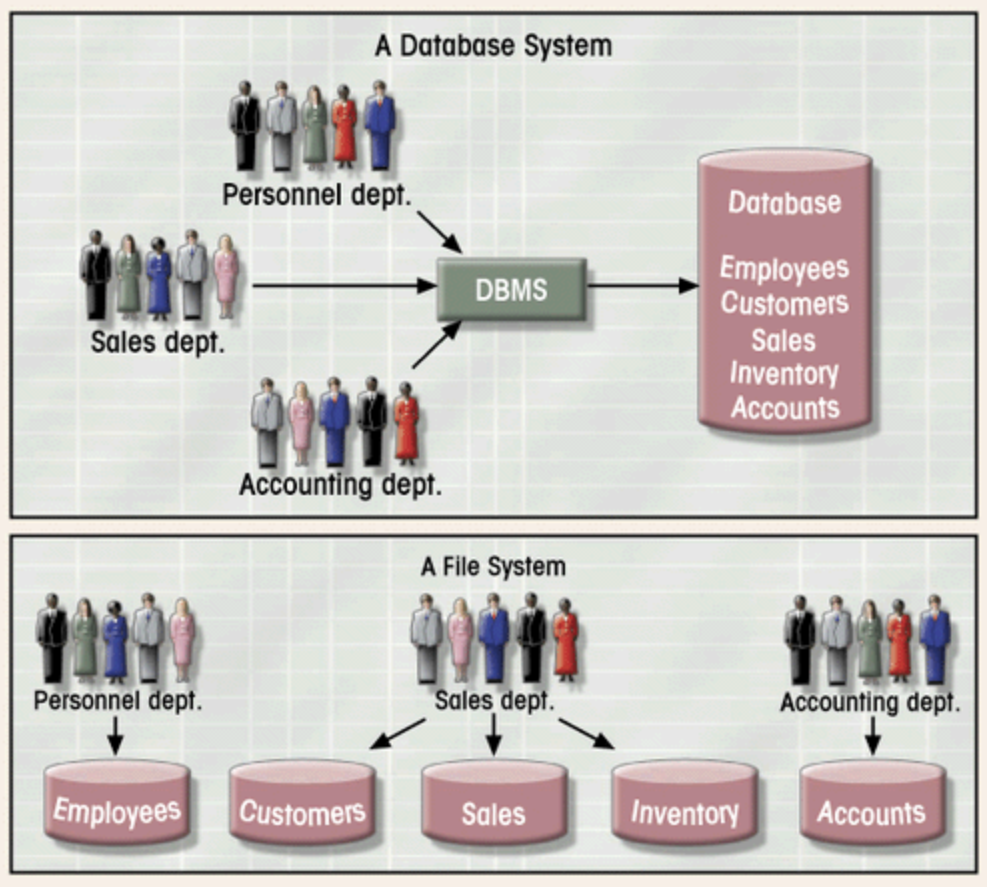
- Weakness
  - "Islands of data" in scattered file systems.
- Problems
  - Duplication
    - same data may be stored in multiple files
  - Inconsistency
    - same data may be stored by different names in different format
  - Rigidity
    - requires customized programming to implement any changes
    - cannot do ad-hoc queries
- Implications
  - Waste of space
  - Data inaccuracies
  - High overhead of data manipulation and maintenance

File System: Problem Case



- inconsistent field name, field size
- inconsistent data values
- data duplication

Database System vs. File System



Database Systems: Design, Implementation, & Management: Rob & Coronel

Disadvantages of File Processing

- ✖ **Program-Data Dependence**
  - + All programs maintain metadata for each file they use
- ✖ **Duplication of Data**
  - + Different systems/programs have separate copies of the same data
- ✖ **Limited Data Sharing**
  - + No centralized control of data
- ✖ **Lengthy Development Times**
  - + Programmers must design their own file formats
- ✖ **Excessive Program Maintenance**
  - + 80% of information systems budget

Problems with Data Dependency

- ✖ Each application programmer must maintain his/her own data
- ✖ Each application program needs to include code for the metadata of each file
- ✖ Each application program must have its own processing routines for reading, inserting, updating, and deleting data
- ✖ Lack of coordination and central control
- ✖ Non-standard file formats

Duplicate Data

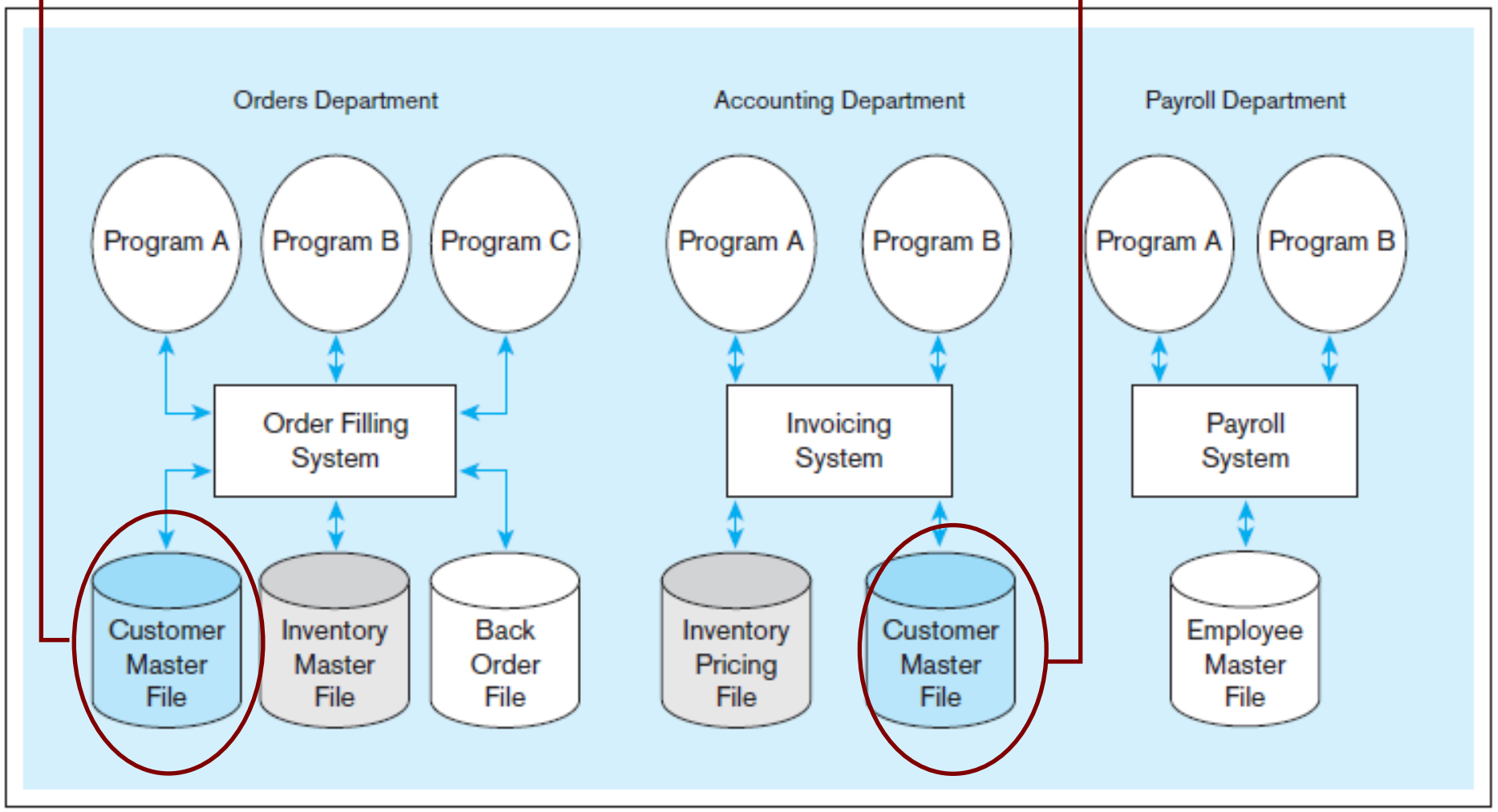


FIGURE 1-2 **Old file processing systems at Pine Valley Furniture Company**

Problems with Data Redundancy

- ✖ Waste of space to have duplicate data
- ✖ Causes more maintenance headaches
- ✖ The biggest problem:
  - ✚ **Data changes in one file could cause inconsistencies**
  - ✚ Compromises in ***data integrity***

SOLUTION: The DATABASE Approach

- ✖ Central repository of shared data
- ✖ Data is managed by a controlling agent
- ✖ Stored in a standardized, convenient form

Requires a Database Management System (DBMS)

Characteristics of the Database Approach (cont'd.)

- Main characteristics of 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

Self-Describing Nature of a Database System

- Database system contains complete definition of structure and constraints
- **Meta-data**
  - Describes structure of the database
- Database **catalog** used by:
  - DBMS software
  - Database users who need information about database structure

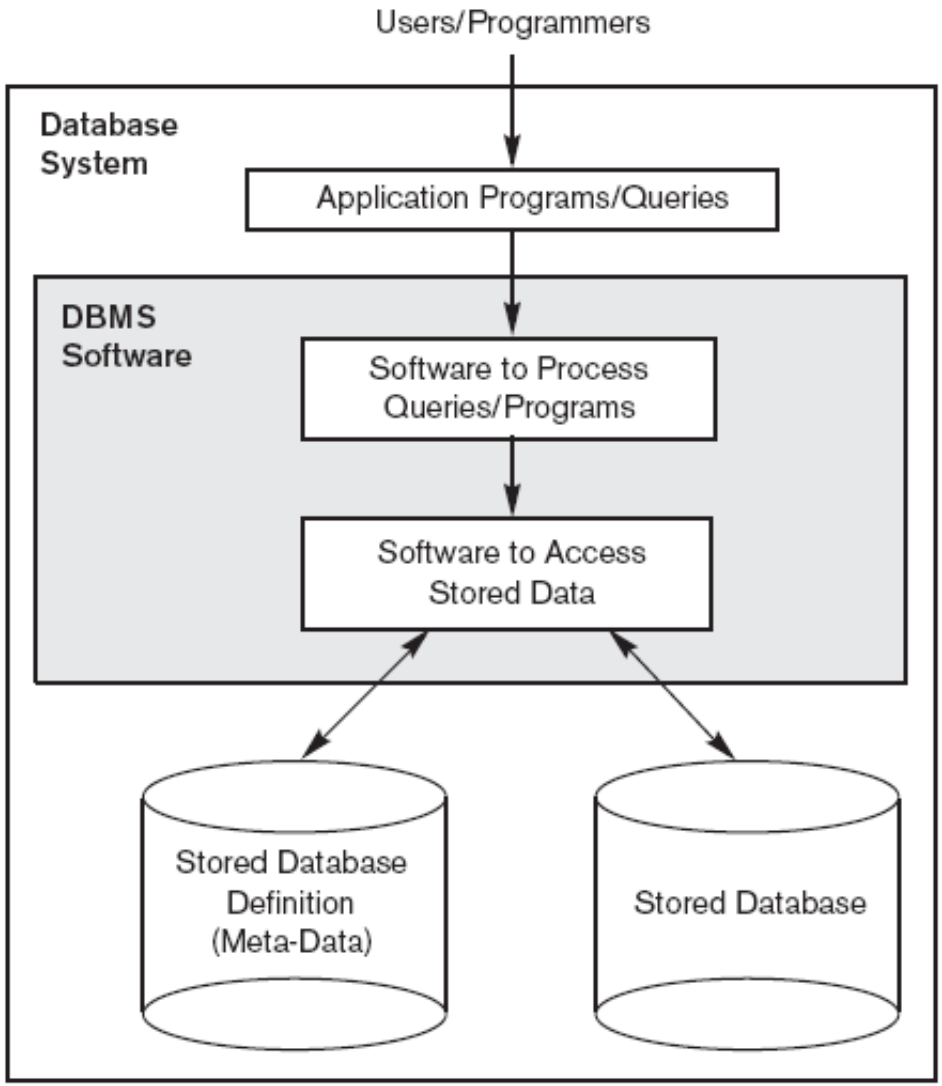


Figure 1.1
A simplified database system environment.

Insulation Between Programs and Data

- **Program-data independence**
  - Structure of data files is stored in DBMS catalog separately from access programs
- Program-operation independence
  - **Operations** specified in two parts:
    - Interface includes operation name and data types of its arguments
    - Implementation can be changed without affecting the interface

Data Abstraction

- **Data abstraction**
  - Allows program-data independence and program-operation independence
- **Conceptual representation** of data
  - Does not include details of how data is stored or how operations are implemented
- **Data model**
  - Type of data abstraction used to provide conceptual representation

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerated type with all known majors.
XXXXNNNN is used to define a type with four alpha characters followed by four digits.

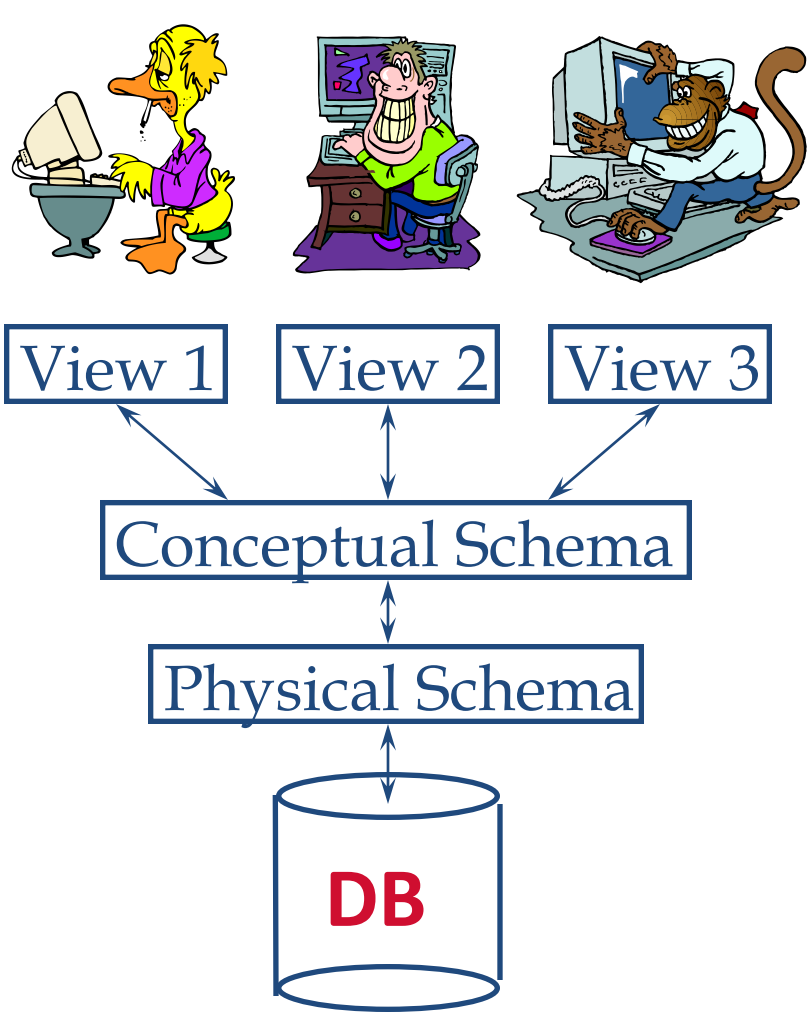
Figure 1.3
An example of a database catalog for the database in Figure 1.2.

Support of Multiple Views of the Data

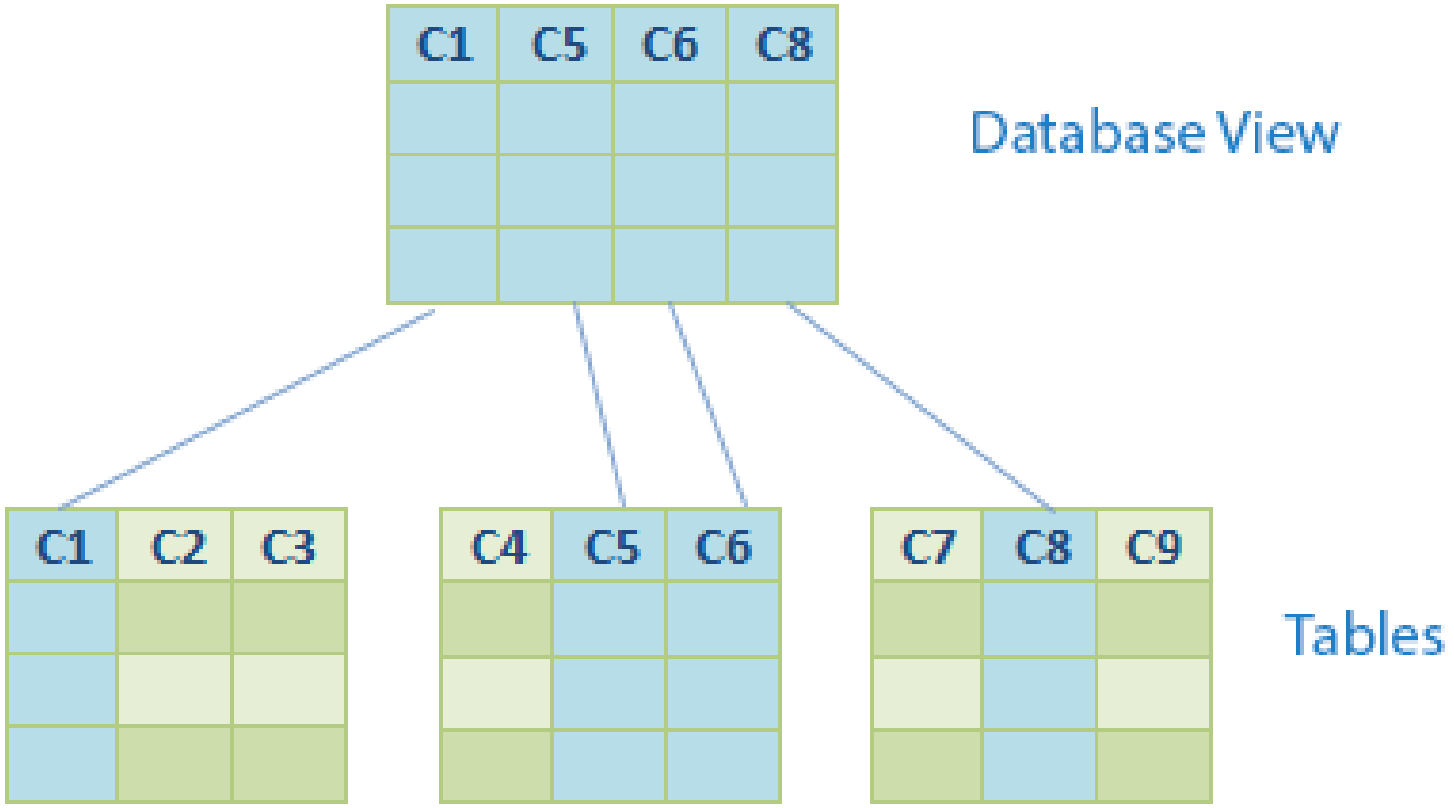
- **View**
  - Subset of the database
  - Contains **virtual data** derived from the database files but is not explicitly stored
    - Birth-Date <-> age
- Multiuser DBMS
  - Users have a variety of distinct applications
  - Must provide facilities for defining multiple views

Levels of Abstraction

- Views describe how users see the data.
- Conceptual schema defines logical structure
- Physical schema describes the files and indexes used.



Database Views



Two Views

(a)

TRANSCRIPT	StudentName	Student Transcript				
		CourseNumber	Grade	Semester	Year	SectionId
	Smith	CS1310	C	Fall	99	119
		MATH2410	B	Fall	99	112
	Brown	MATH2410	A	Fall	98	85
		CS1310	A	Fall	98	92
		CS3320	B	Spring	99	102
		CS3380	A	Fall	99	135

(b)

PREREQUISITES	CourseName	CourseNumber	Prerequisites
	Database	CS3380	CS3320
			MATH2410
	Data Structures	CS3320	CS1310

Sharing of Data and Multiuser Transaction Processing

- Allow multiple users to access the database at the same time
- **Concurrency control software**
  - Ensure that several users trying to update the same data do so in a controlled manner
    - Result of the updates is correct
- **Online transaction processing (OLTP) application**

Sharing of Data and Multiuser Transaction Processing (cont'd.)

- **Transaction**
  - Central to many database applications
  - Executing program or process that includes one or more database
  - **Isolation** property
    - Each transaction appears to execute in isolation from other transactions
  - **Atomicity** property
    - Either all the database operations in a transaction are executed or none are
    - Very Important: Bank Transactions

Concurrent execution of user programs

- Why?
  - Utilize CPU while waiting for disk I/O
    - (database programs make heavy use of disk)
  - Avoid short programs waiting behind long ones
    - e.g. ATM withdrawal while bank manager sums balance across all accounts

Concurrent execution

- Interleaving actions of different programs: trouble!

  Example:

  - Bill transfers $100 from savings to checking
    Savings –= 100; Checking += 100
  - Meanwhile, Bill's wife requests account info.

  Bad interleaving:

    - Savings –= 100
    - Print balances
    - Checking += 100

  – Printout is missing $100 !

Concurrency Control

- DBMS ensures such problems don't arise
- Users can pretend they are using a single-user system. (called "Isolation")
  - Thank goodness!

Key concept: **Transaction**

- an atomic sequence of database actions (reads/writes)
- takes DB from one consistent state to another

Example

- Here, *consistency* is based on our knowledge of banking "semantics"
- In general, up to writer of transaction to ensure transaction preserves consistency
- DBMS provides (limited) automatic enforcement, via integrity constraints
  - e.g., balances must be >= 0

Concurrent transactions

- Goal: **execute xacts {T1, T2, … Tn}, and ensure a consistent outcome**
- *One option:* **"serial" schedule (one after another)**
- *Better:* **allow interleaving of xact actions, as long as outcome is equivalent to some serial schedule**

Possible Enforcement Methods

- Optimistic: **permit arbitrary interleaving, then check equivalence to serial sched.**

- Pessimistic: **xacts set** ***locks*** **on data objects, such that illegal interleaving is impossible**

Locking example

- T1 (Bill): *Savings –= 100; Checking += 100*
- T2 (Bill's wife): *Print(Checking); Print(Savings)*

– T1 and T2 both lock Savings and Checking objects

– If T1 locks Savings & Checking first, T2 must wait

Ensuring Transaction Properties

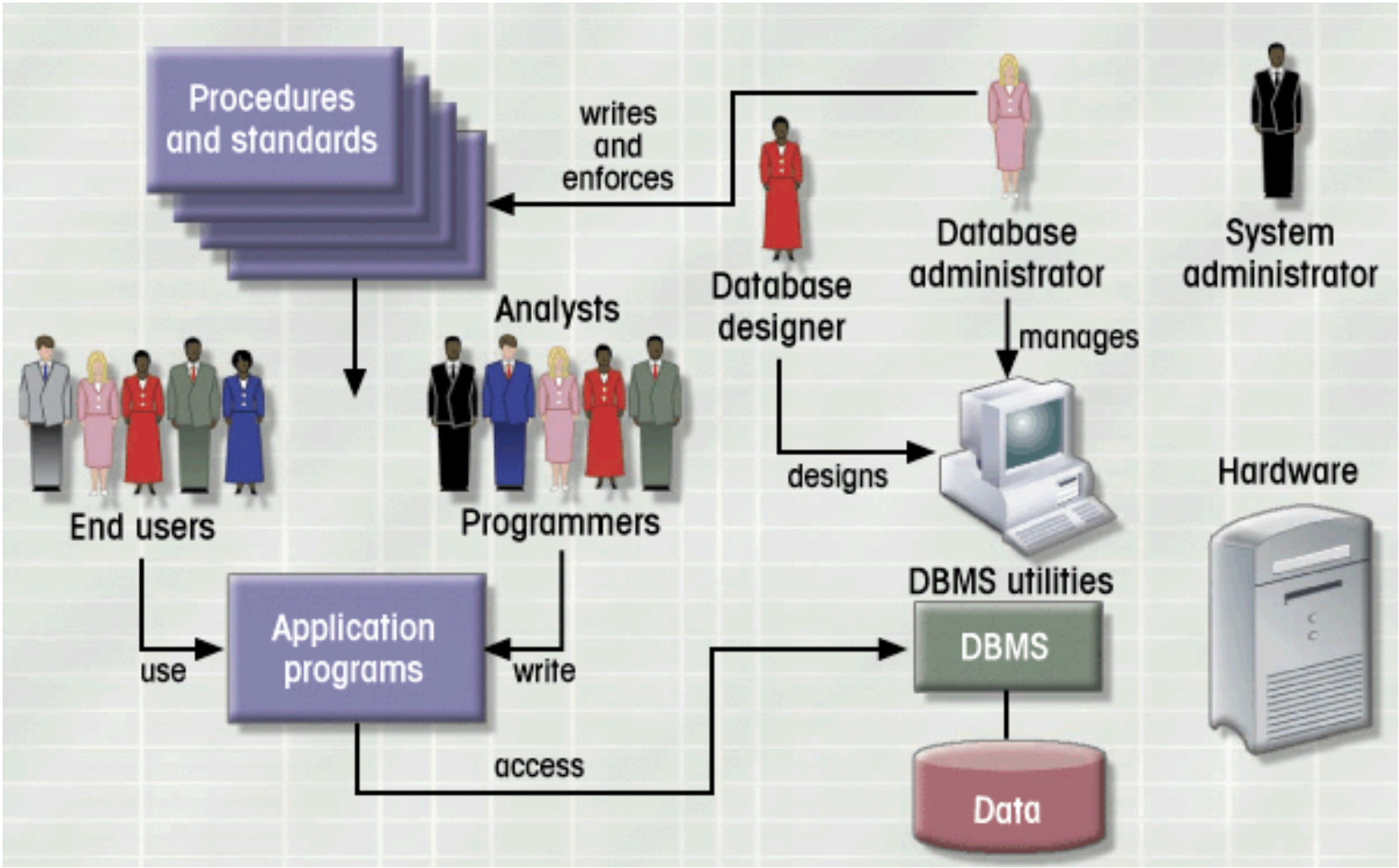
- DBMS ensures:
  - ***atomicity*** **even if xact aborted (due to deadlock, system crash, …)**

- Idea: Keep a *log* of all actions carried out by the DBMS:
  - Record all DB modifications in log, *before* they are executed
  - To abort a xact, undo logged actions in reverse order
  - If system crashes, must:
    1) **undo** partially executed xacts (ensures **atomicity**)

  - *trickier than it sounds!*

Introduction to Databases

- Introduction
- An Example
- Characteristics of the Database Approach
- Actors on the Scene
- Workers behind the Scene
- Advantages of Using the DBMS Approach
- A Brief History of Database Applications
- When Not to Use a DBMS


Procedures and standards
writes and enforces
Database administrator
System administrator
Database designer
Analysts
manages
End users
Programmers
designs
Hardware
DBMS utilities
Application programs
use
write
DBMS
access
Data

Actors on the Scene

- **Database administrators (DBA)** are responsible for:
  - Authorizing access to the database
  - Coordinating and monitoring its use
  - Acquiring software and hardware resources
- **Database designers** are responsible for:
  - Identifying the data to be stored
  - Choosing appropriate structures to represent and store this data

Actors on the Scene (cont'd.)

- **End users**
  - People whose jobs require access to the database
  - Types
    - Casual end users
    - Naive or parametric end users
    - Sophisticated end users
    - Standalone users

Actors on the Scene (cont'd.)

- **System analysts**
  - Determine requirements of end users
- **Application programmers**
  - Implement these specifications as programs

Introduction to Databases

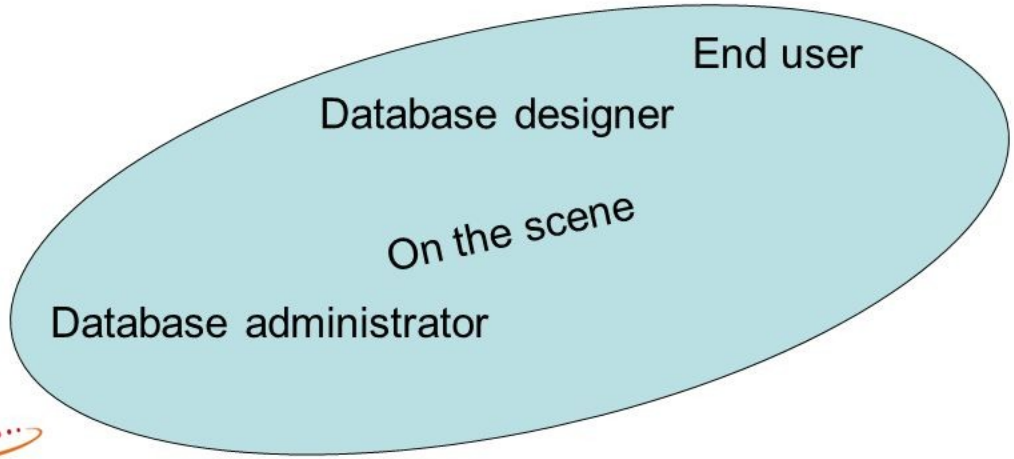
- Introduction
- An Example
- Characteristics of the Database Approach
- Actors on the Scene
- Workers behind the Scene
- Advantages of Using the DBMS Approach
- A Brief History of Database Applications
- When Not to Use a DBMS

Workers behind the Scene

- **DBMS system designers and implementers**
  - Design and implement the DBMS modules and interfaces as a software package
- **Tool developers**
  - Design and implement **tools**
- **Operators and maintenance personnel**
  - Responsible for running and maintenance of hardware and software environment for database system

Actors

Behind the scene
Tool developers
DBMS designers and implementers

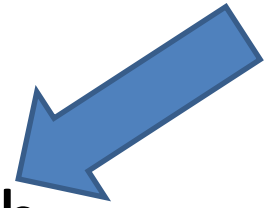


PROFESSIONSHØJSKOLEN
University College Nordjylland

18

Introduction to Databases

- Introduction
- An Example
- Characteristics of the Database Approach
- Actors on the Scene
- Workers behind the Scene
- Advantages of Using the DBMS Approach
- A Brief History of Database Applications
- When Not to Use a DBMS



Advantages of Using the DBMS Approach

- Controlling redundancy
  - **Data normalization**
  - **Denormalization**
    - Sometimes necessary to use **controlled redundancy** to improve the performance of queries

Controlling Redundancy

- This **redundancy in storing the same data multiple**
- times leads to several problems:
  - Logic update – we need to update several times
  - Storage space is wasted
  - The file that represent the same data may become *inconsistent*

Advantages of Using the DBMS Approach

- Restricting unauthorized access
  - Security and authorization subsystem
  - Privileged software

Database Security and the DBA

- The database administrator (**DBA**) is the central authority for managing a database system.
  - The DBA's responsibilities include
    - granting privileges to users who need to use the system
    - classifying users and data in accordance with the policy of the organization
- The DBA is responsible for the overall security of the database system.

Database Security and the DBA (2)

- The DBA has a DBA account in the DBMS
  - Sometimes these are called a system or superuser account
  - These accounts provide powerful capabilities such as:
    - 1. Account creation
    - 2. Privilege granting
    - 3. Privilege revocation
    - 4. Security level assignment

Access Protection, User Accounts, and Database Audits

- Whenever a person or group of persons need to access a database system, the individual or group must first apply for a user account.
  - The DBA will then create a new **account id** and **password** for the user if he/she deems there is a legitimate need to access the database
- The user must log in to the DBMS by entering account id and password whenever database access is needed.

1.3 Access Protection, User Accounts, and Database Audits(2)

- The database system must also keep **track of all operations** on the database that are applied by a certain user throughout **each login session**.
  - To keep a record of all updates applied to the database and of the particular user who applied each update, we can modify **system log**, which includes an entry for each operation applied to the database that may be required for recovery from a transaction failure or system crash.

1.3 Access Protection, User Accounts, and Database Audits(3)

- If any tampering with the database is suspected, a **database audit** is performed
  - A database audit consists of reviewing the log to examine all accesses and operations applied to the database during a certain time period.
- A database log that is used mainly for security purposes is sometimes called an **audit trail**.

Types of Privileges

- The **account level**:
  - At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.
- The **relation level** (or **table level**):
  - At this level, the DBA can control the privilege to access each individual relation or view in the database.

Types of Privileges

- The privileges at the **account level** apply to the capabilities provided to the account itself and can include
  - the **CREATE SCHEMA** or **CREATE TABLE** privilege, to create a schema or base relation;
  - the **CREATE VIEW** privilege;
  - the **ALTER** privilege, to apply schema changes such adding or removing attributes from relations;
  - the **DROP** privilege, to delete relations or views;
  - the **MODIFY** privilege, to insert, delete, or update tuples;
  - and the **SELECT** privilege, to retrieve information from the database by using a **SELECT** query.

Types of Privileges

- In SQL the following types of privileges can be granted on each individual relation R:
  - **SELECT** (retrieval or read) privilege on R:
    - Gives the account retrieval privilege.
    - In SQL this gives the account the privilege to use the **SELECT** statement to retrieve tuples from R.
  - **MODIFY** privileges on R:
    - This gives the account the capability to modify tuples of R.
    - In SQL this privilege is further divided into **UPDATE**, **DELETE**, and **INSERT** privileges to apply the corresponding SQL command to R.
    - In addition, both the **INSERT** and **UPDATE** privileges can specify that only certain attributes can be updated by the account.

Revoking Privileges

- In some cases it is desirable to grant a privilege to a user temporarily. For example,
  - The owner of a relation may want to grant the **SELECT** privilege to a user for a specific task and then revoke that privilege once the task is completed.
  - Hence, a mechanism for **revoking** privileges is needed. In SQL, a **REVOKE** command is included for the purpose of **canceling privileges**.

An Example

- Suppose that the DBA creates four accounts
  - A1, A2, A3, A4
- and wants only A1 to be able to create base relations. Then the DBA must issue the following GRANT command in SQL

```
GRANT CREATETAB TO A1;
```

- In SQL2 the same effect can be accomplished by having the DBA issue a **CREATE SCHEMA** command as follows:

```
CREATE SCHAMA EXAMPLE AUTHORIZATION
A1;
```

An Example

- User account A1 can create tables under the schema called **EXAMPLE**.
- Suppose that A1 **creates** the two base relations **EMPLOYEE** and **DEPARTMENT**
  - A1 is then **owner** of these two relations and hence all the relation privileges on each of them.
- Suppose that A1 wants to grant A2 the privilege to insert and delete tuples in both of these relations, but A1 does not want A2 to be able to propagate these privileges to additional accounts:

```
GRANT INSERT, DELETE ON
    EMPLOYEE, DEPARTMENT TO A2;
```

An Example

EMPLOYEE

Name	Ssn	Bdate	Address	Sex	Salary	Dno

DEPARTMENT

Dnumber	Dname	Mgr_ssn

Figure 23.1
Schemas for the two relations EMPLOYEE and DEPARTMENT.

An Example

- Suppose that A1 wants to allow A3 to retrieve information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts.
- A1 can issue the command:

```
GRANT SELECT ON EMPLOYEE, DEPARTMENT
    TO A3 WITH GRANT OPTION;
```

- A3 can grant the **SELECT** privilege on the **EMPLOYEE** relation to A4 by issuing:

```
GRANT SELECT ON EMPLOYEE TO A4;
```

  - Notice that A4 can't propagate the SELECT privilege because GRANT OPTION was not given to A4

An Example

- Suppose that A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3; A1 can issue:

```
REVOKE SELECT ON EMPLOYEE FROM A3;
```

- The DBMS must now automatically revoke the SELECT privilege on EMPLOYEE from A4, too, because A3 granted that privilege to A4 and A3 does not have the privilege any more.

An Example

- Finally, suppose that A1 wants to allow A4 to update only the SALARY attribute of EMPLOYEE;
- A1 can issue:

```
GRANT UPDATE ON EMPLOYEE (SALARY) TO
A4;
```

  - The **UPDATE** or **INSERT** privilege can specify particular attributes that may be updated or inserted in a relation.
  - Other privileges (**SELECT**, **DELETE**) are not attribute specific.

Advantages of Using the DBMS Approach (cont'd.)

- Providing **persistent** storage for program objects
  - Complex object in C++ can be stored permanently in an object-oriented DBMS

Advantages of Using the DBMS Approach (cont'd.)

- Providing storage structures and search techniques for efficient query processing
  - **Indexes and hashing**
  - **Buffering** and **caching**
  - **Query processing and optimization**

Indexes

Indexes: Basic Concepts

- Indexing mechanisms used to speed up access to desired data.
  - E.g., author catalog in library
- **Search Key** - attribute to set of attributes used to look up records in a file.
- An **index file** consists of records (called **index entries**) of the form

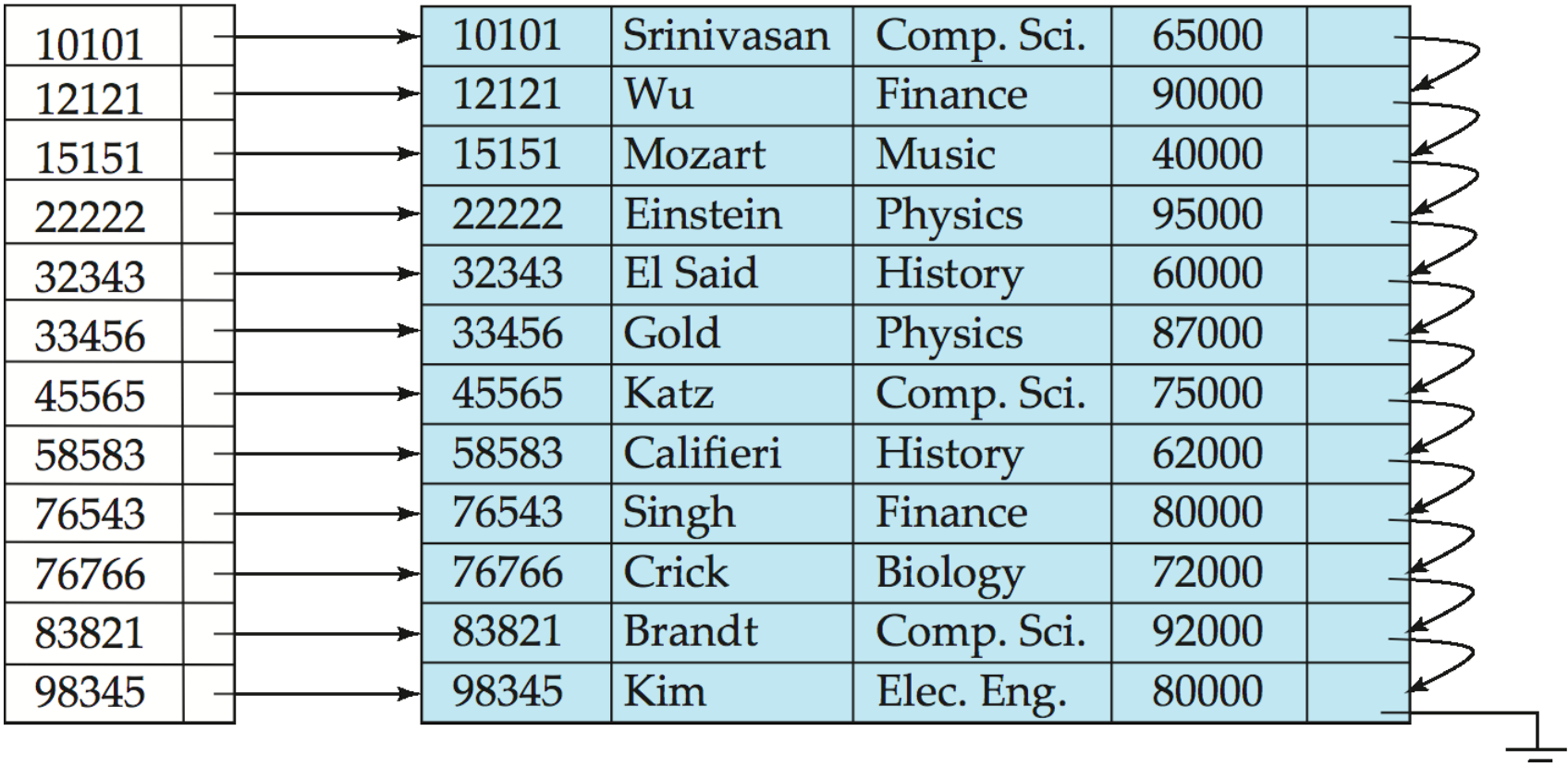
- Index files are typically much smaller than the original file
- Two basic kinds of indices:
  - **Ordered indices:** search keys are stored in sorted order
  - **Hash indices:** search keys are distributed uniformly across "buckets" using a "hash function".

Ordered Indices

- In an **ordered index,** index entries are stored sorted on the search key value. E.g., author catalog in library.
- **Primary index:** in a sequentially ordered file, the index whose search key specifies the sequential order of the file.
  - Also called **clustering index**
  - The search key of a primary index is usually but not necessarily the primary key.
- **Secondary index**: an index whose search key specifies an order different from the sequential order of the file. Also called non-clustering index**.**
- Index-sequential file**:** ordered sequential file with a primary index.

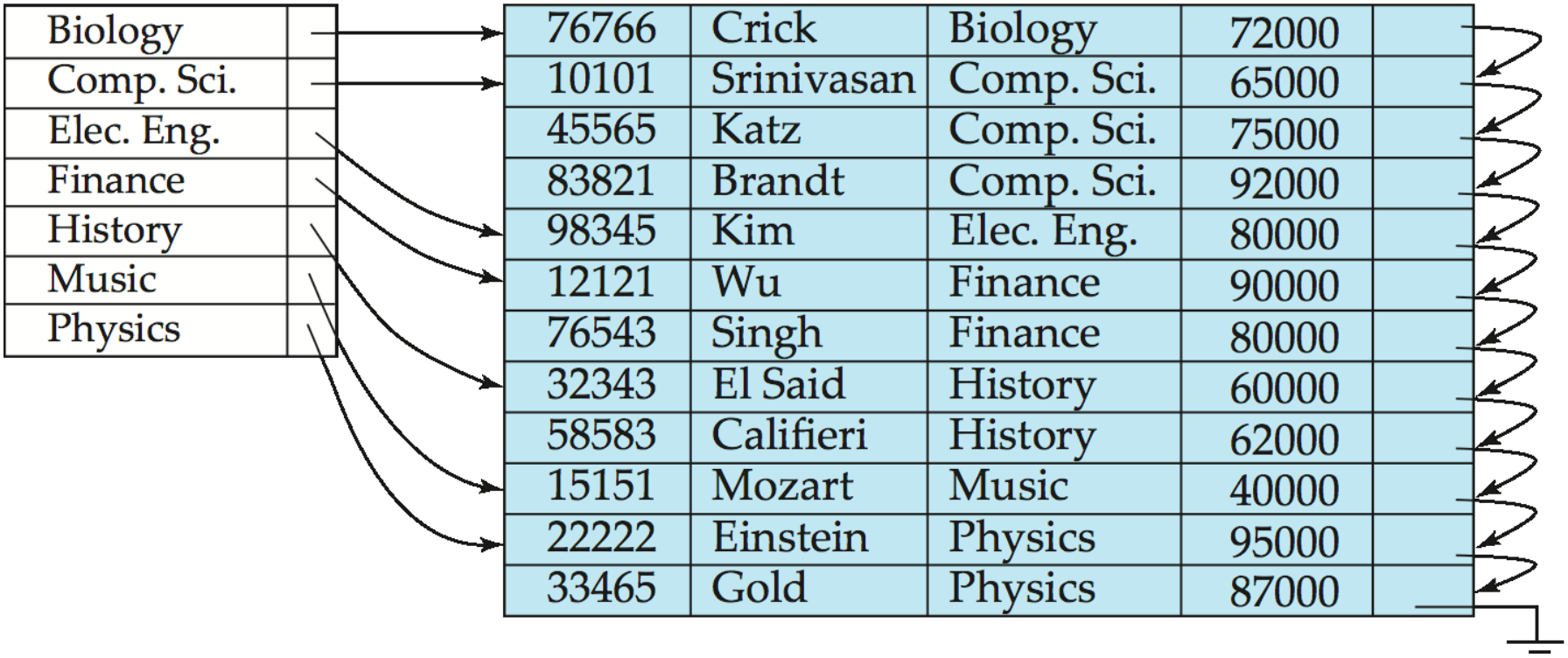
Dense Index Files

- **Dense index** — Index record appears for every search-key value in the file.
- E.g. index on *ID* attribute of *instructor* relation



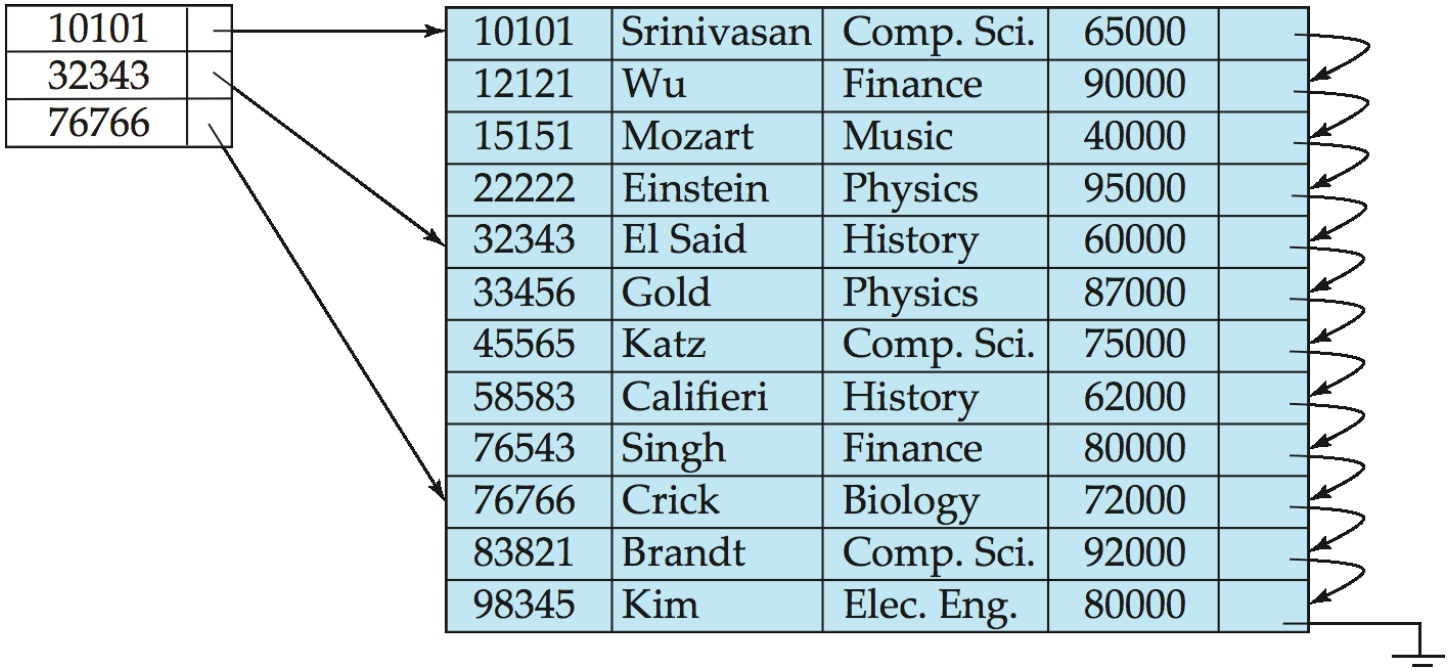
Dense Index Files (Cont.)

- Dense index on *dept_name*, with *instructor* file sorted on *dept_name*



Sparse Index Files

- **Sparse Index**: contains index records for only some search-key values.
  - Applicable when records are sequentially ordered on search-key
- To locate a record with search-key value *K* we:
  - Find index record with largest search-key value < *K*
  - Search file sequentially starting at the record to which the index record points



Static Hashing

- A **bucket** is a unit of storage containing one or more records (a bucket is typically a disk block).
- In a **hash file organization** we obtain the bucket of a record directly from its search-key value using a **hash function.**
- Hash function *h* is a function from the set of all search-key values *K* to the set of all bucket addresses *B.*
- Hash function is used to locate records for access, insertion as well as deletion.
- Records with different search-key values may be mapped to the same bucket; thus entire bucket has to be searched sequentially to locate a record.

Example of Hash File Organization

Hash file organization of *instructor* file, using *dept_name* as key
(See figure in next slide.)

- There are 10 buckets,
- The binary representation of the *i*th character is assumed to be the integer *i.*
- The hash function returns the sum of the binary representations of the characters modulo 10
  - E.g. h(Music) = 1 h(History) = 2
    h(Physics) = 3 h(Elec. Eng.) = 3

Example of Hash File Organization

bucket 0

bucket 1

15151	Mozart	Music	40000

bucket 2

32343	El Said	History	80000
58583	Califieri	History	60000

bucket 3

22222	Einstein	Physics	95000
33456	Gold	Physics	87000
98345	Kim	Elec. Eng.	80000

bucket 4

12121	Wu	Finance	90000
76543	Singh	Finance	80000

bucket 5

76766	Crick	Biology	72000

bucket 6

10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

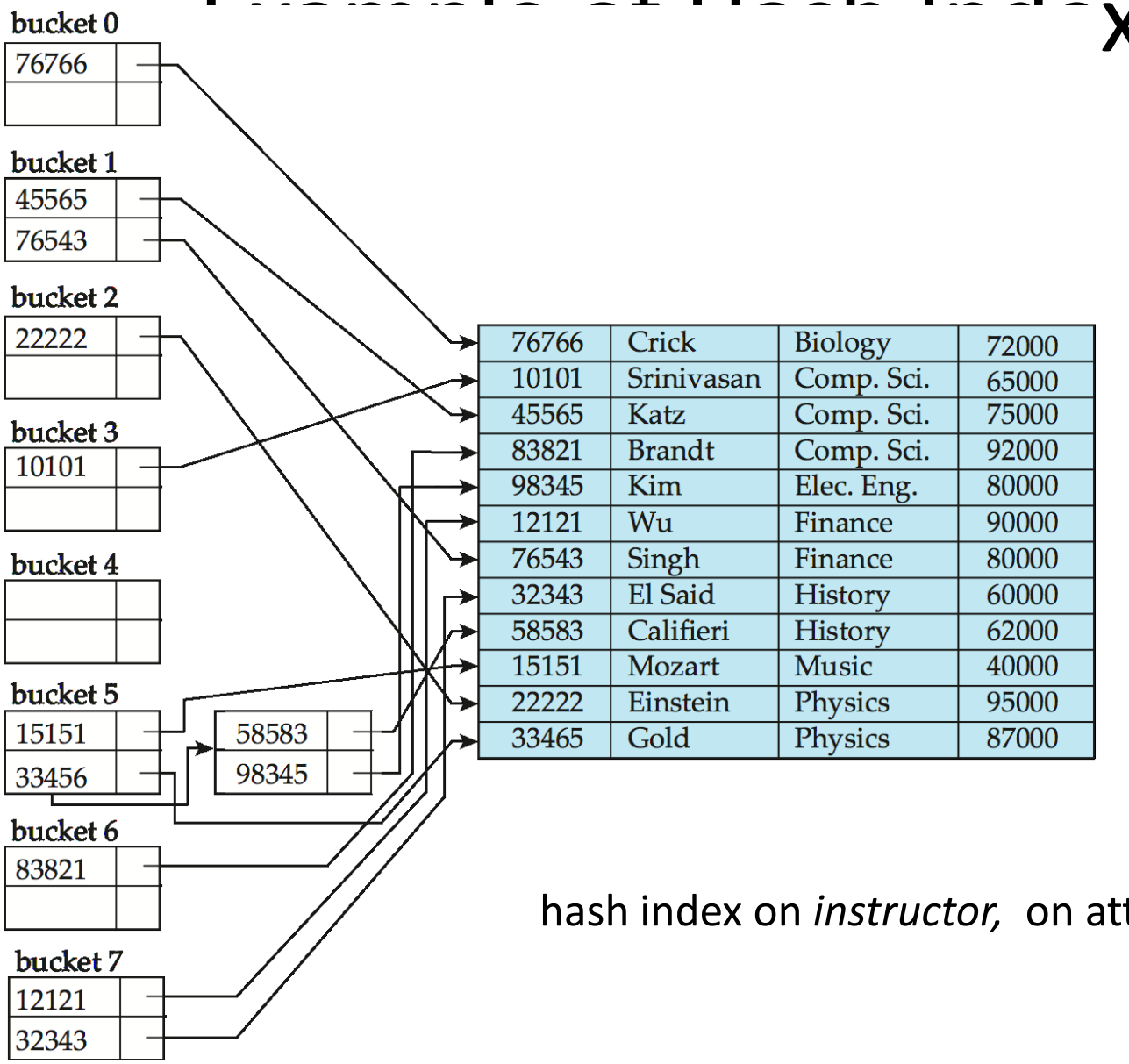
bucket 7

Hash file organization of *instructor* file, using *dept_name* as key (see previous slide for details).

Hash Functions

- Worst hash function maps all search-key values to the same bucket; this makes access time proportional to the number of search-key values in the file.
- An ideal hash function is **uniform**, i.e., each bucket is assigned the same number of search-key values from the set of *all* possible values.
- Ideal hash function is **random**, so each bucket will have the same number of records assigned to it irrespective of the *actual distribution* of search-key values in the file.

Example of Hash Index

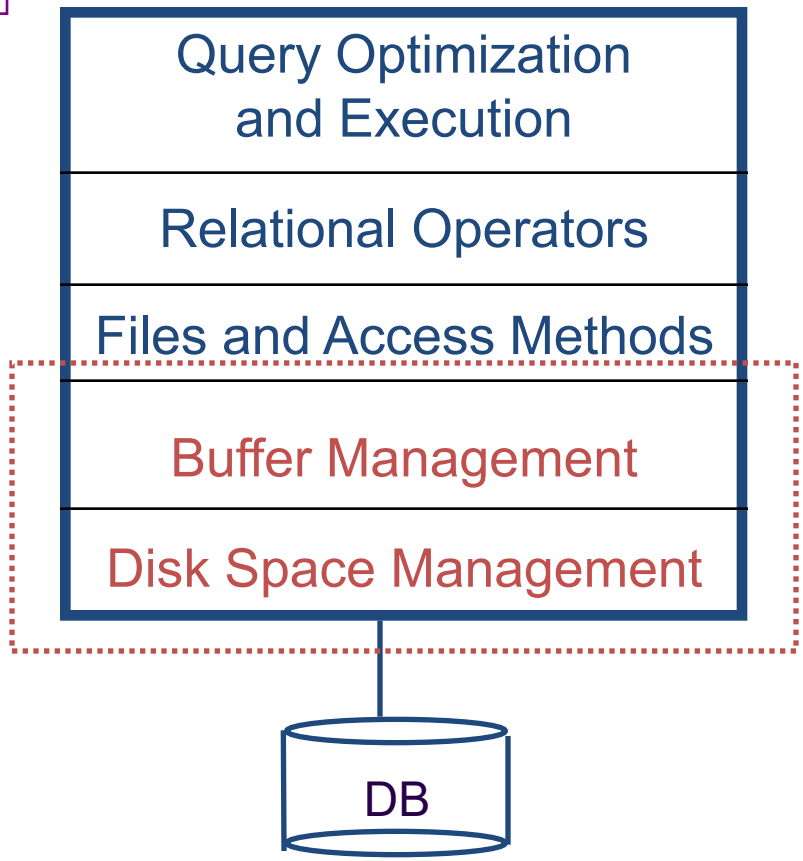


hash index on *instructor,* on attribute *ID*

Buffering and Caching

Disks, Memory, and Files

The BIG picture…



Disks and Files

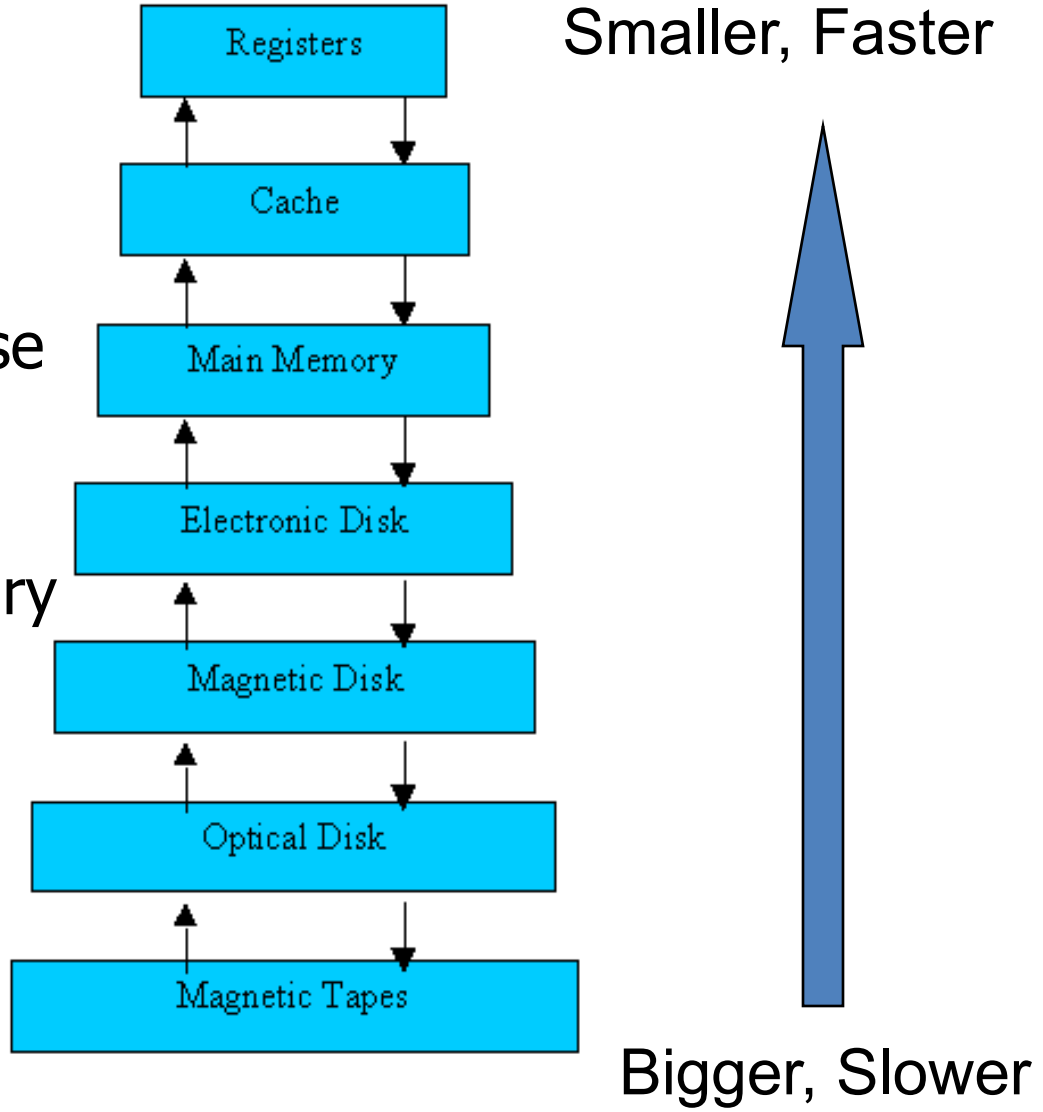
- DBMS stores information on disks.
- This has major implications for DBMS design!
  - READ: transfer data from disk to main memory (RAM).
  - WRITE: transfer data from RAM to disk.
  - Both are high-cost operations, relative to in-memory operations, so must be planned carefully!

The Storage Hierarchy

–Main memory (RAM) for currently used data.

–Disk for the main database (secondary storage).

–Tapes for archiving older versions of the data (tertiary storage).



Source: Operating Systems Concepts 5th Edition

Disks

- Secondary storage device of choice.
- Main advantage over tapes: *random access* vs. *sequential*.
- Data is stored and retrieved in units called *disk blocks* or *pages*.
- Unlike RAM, time to retrieve a disk block varies depending upon location on disk.
  - Therefore, relative placement of blocks on disk has major impact on DBMS performance!

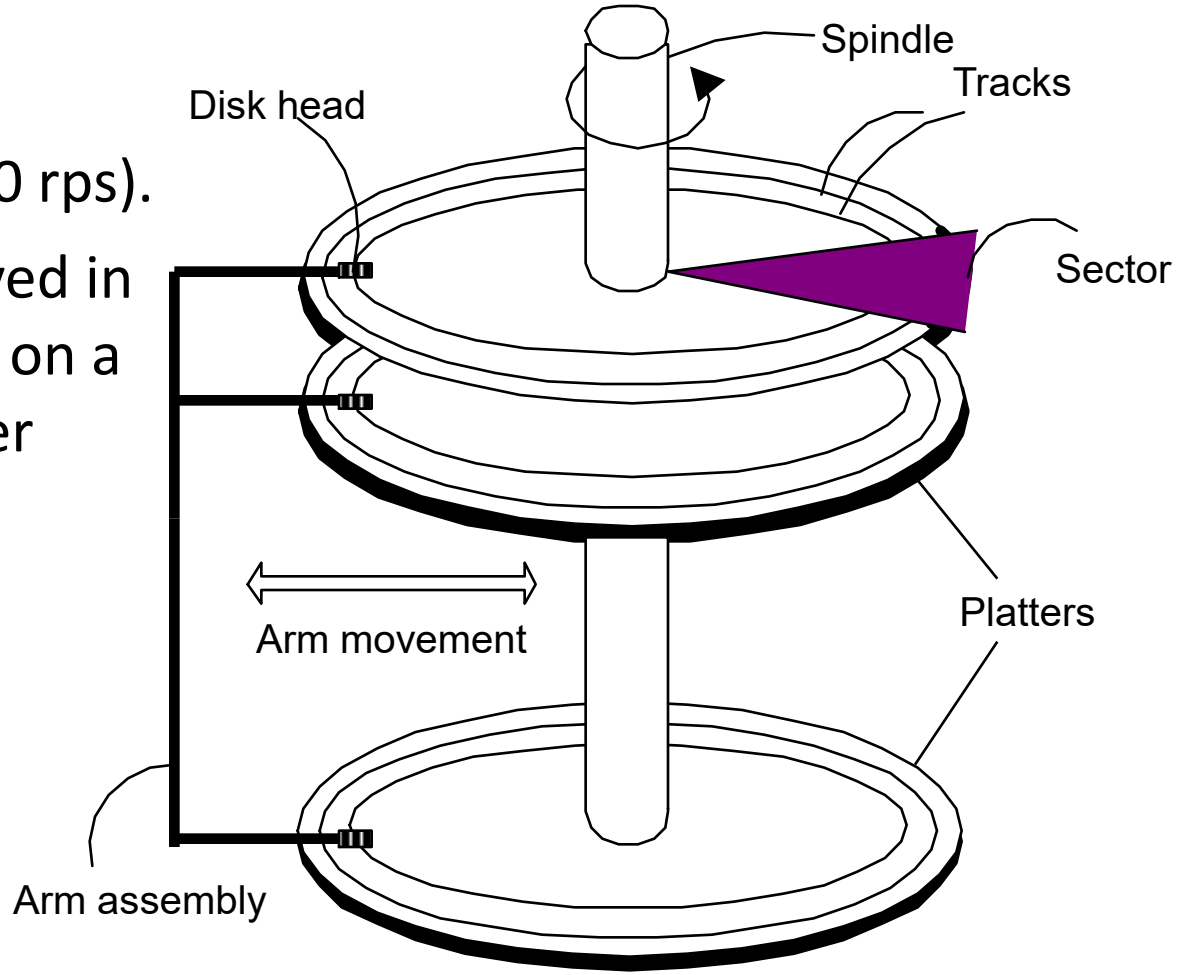
Components of a Disk

The platters spin (say, 120 rps).

The arm assembly is moved in or out to position a head on a desired track. Tracks under heads make a *cylinder* (imaginary!).

Only one head reads/writes at any one time.

❖ *Block size* is a multiple of *sector size* (which is fixed).



Accessing a Disk Page

- Time to access (read/write) a disk block:
  - *seek time* (moving arms to position disk head on track)
  - *rotational delay* (waiting for block to rotate under head)
  - *transfer time* (actually moving data to/from disk surface)
- Seek time and rotational delay dominate.
  - Seek time varies between about 0.3 and 10msec
  - Rotational delay varies from 0 to 4msec
  - Transfer rate around .08msec per 8K block
- Key to lower I/O cost: reduce seek/rotation delays! Hardware vs. software solutions?

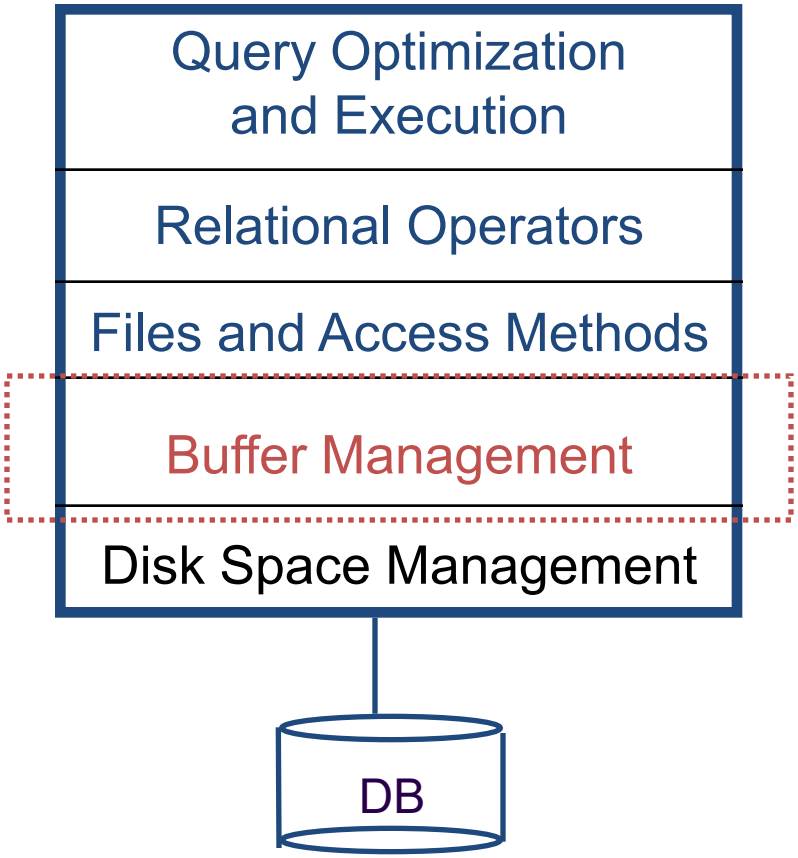
Arranging Pages on Disk

- *`Next'* block concept:
  - blocks on same track, followed by
  - blocks on same cylinder, followed by
  - blocks on adjacent cylinder
- Blocks in a file should be arranged sequentially on disk (by `next'), to minimize seek and rotational delay.
- For a sequential scan, *pre-fetching* several pages at a time is a big win!

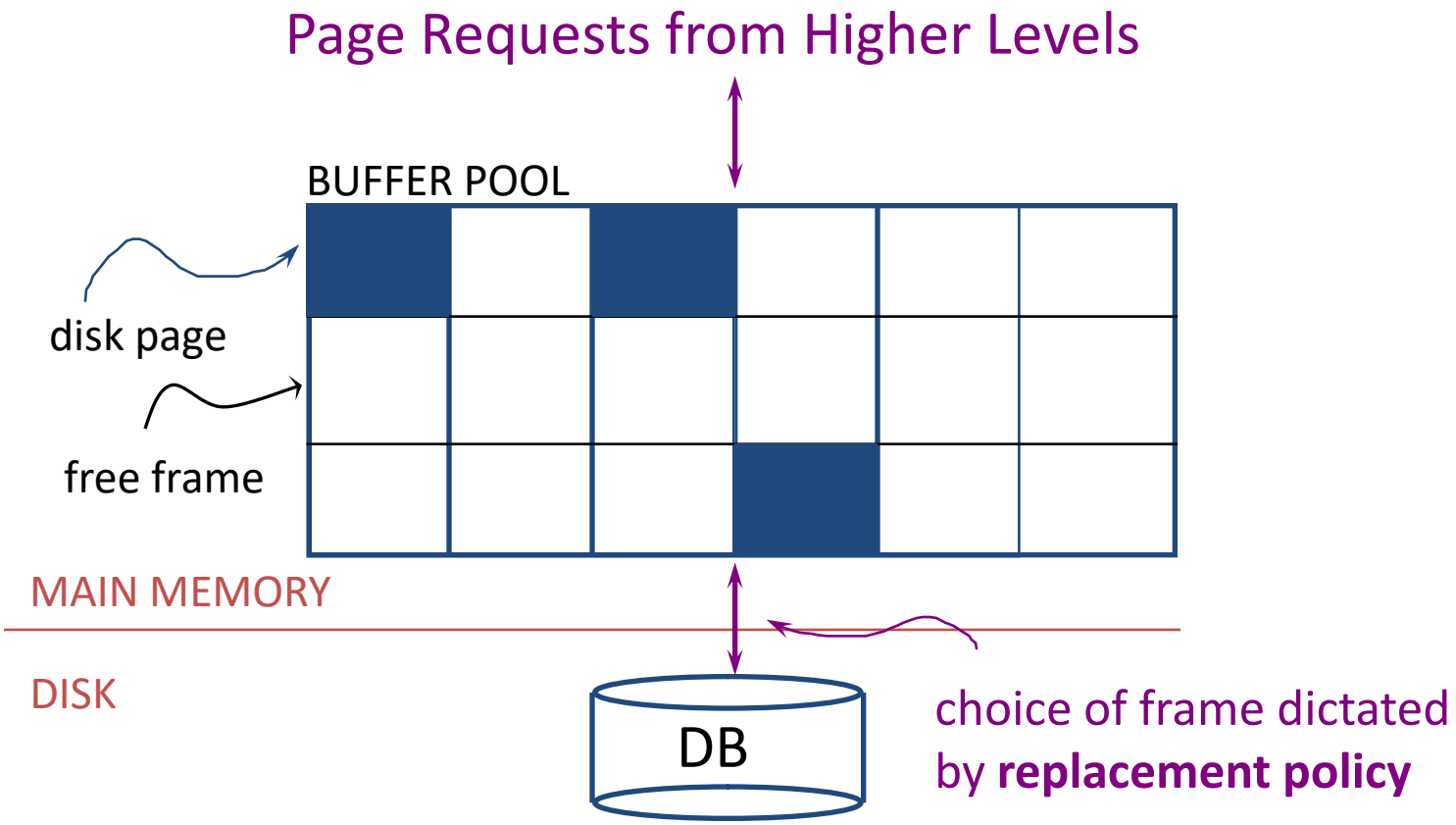
Disk Space Management

- Lowest layer of DBMS software manages space on disk (using OS file system or not?).
- Higher levels call upon this layer to:
  - allocate/de-allocate a page
  - read/write a page
- Best if a request for a *sequence* of pages is satisfied by pages stored sequentially on disk!
  - Responsibility of disk space manager.
  - Higher levels don't know how this is done, or how free space is managed.
  - Though they may make performance assumptions!
    - Hence disk space manager should do a decent job.

Context


Query Optimization
and Execution
Relational Operators
Files and Access Methods
Buffer Management
Disk Space Management
DB

Buffer Management in a DBMS



- *Data must be in RAM for DBMS to operate on it!*
- *Buffer Mgr hides the fact that not all data is in RAM*

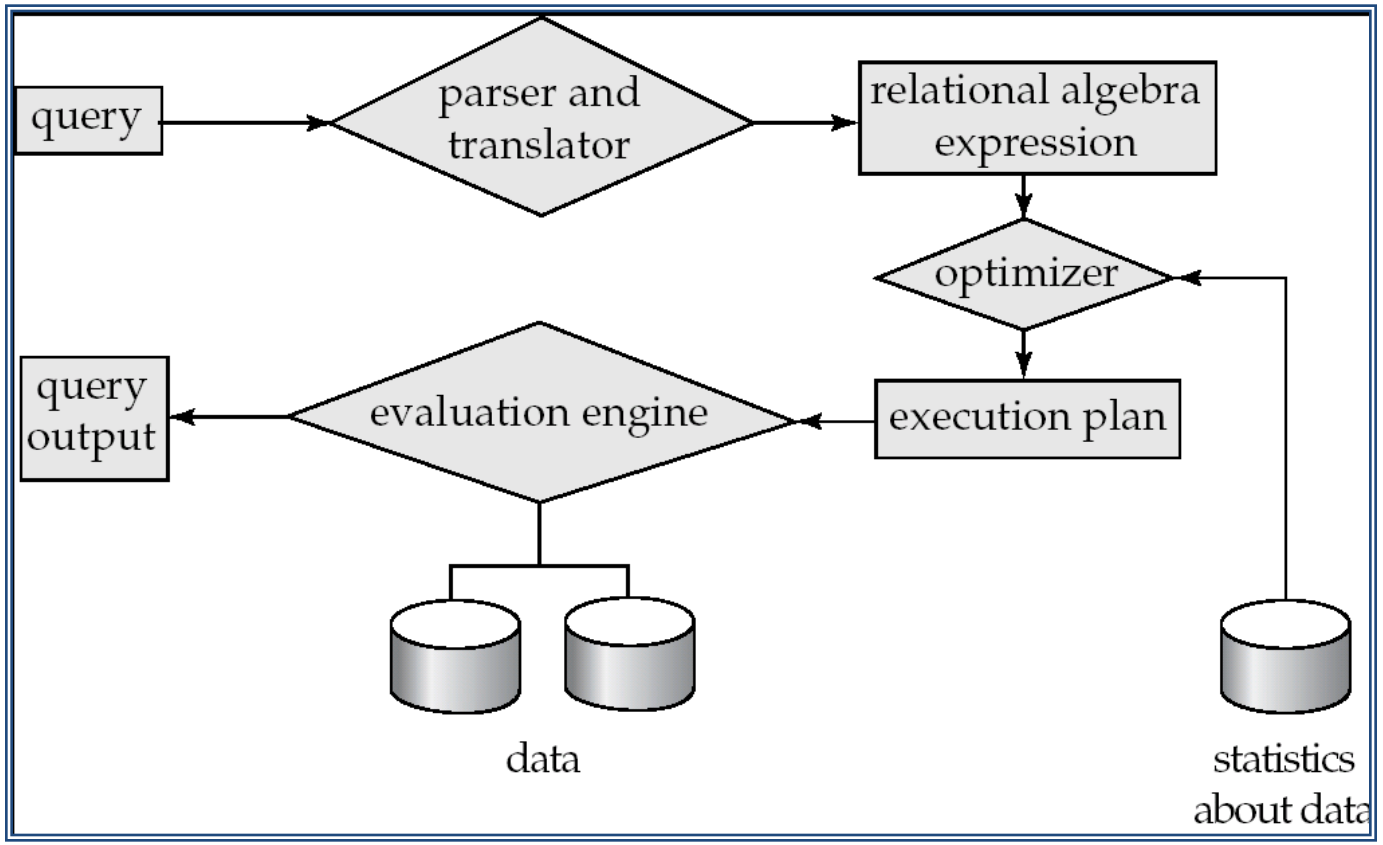
Query processing and optimization

What is Query Processing?

- Query processing: Activities involved in extracting data from a database.
  - Translation of queries in high-level DB languages into expressions that can be used at physical level of file system.
  - Includes query optimization and query evaluation.
-

Three Basic Steps in Query Processing

1. **Parsing and translation**
2. **Optimization**
3. **Evaluation**



Parsing and translation

- Translate the query into its internal form.
  - This is then translated into relational algebra.
- Parser checks syntax, verifies relations.
- A relational algebra expression may have many equivalent expressions
  - E.g., $\sigma_{balance<2500}(\prod_{balance}(account))$ is equivalent to

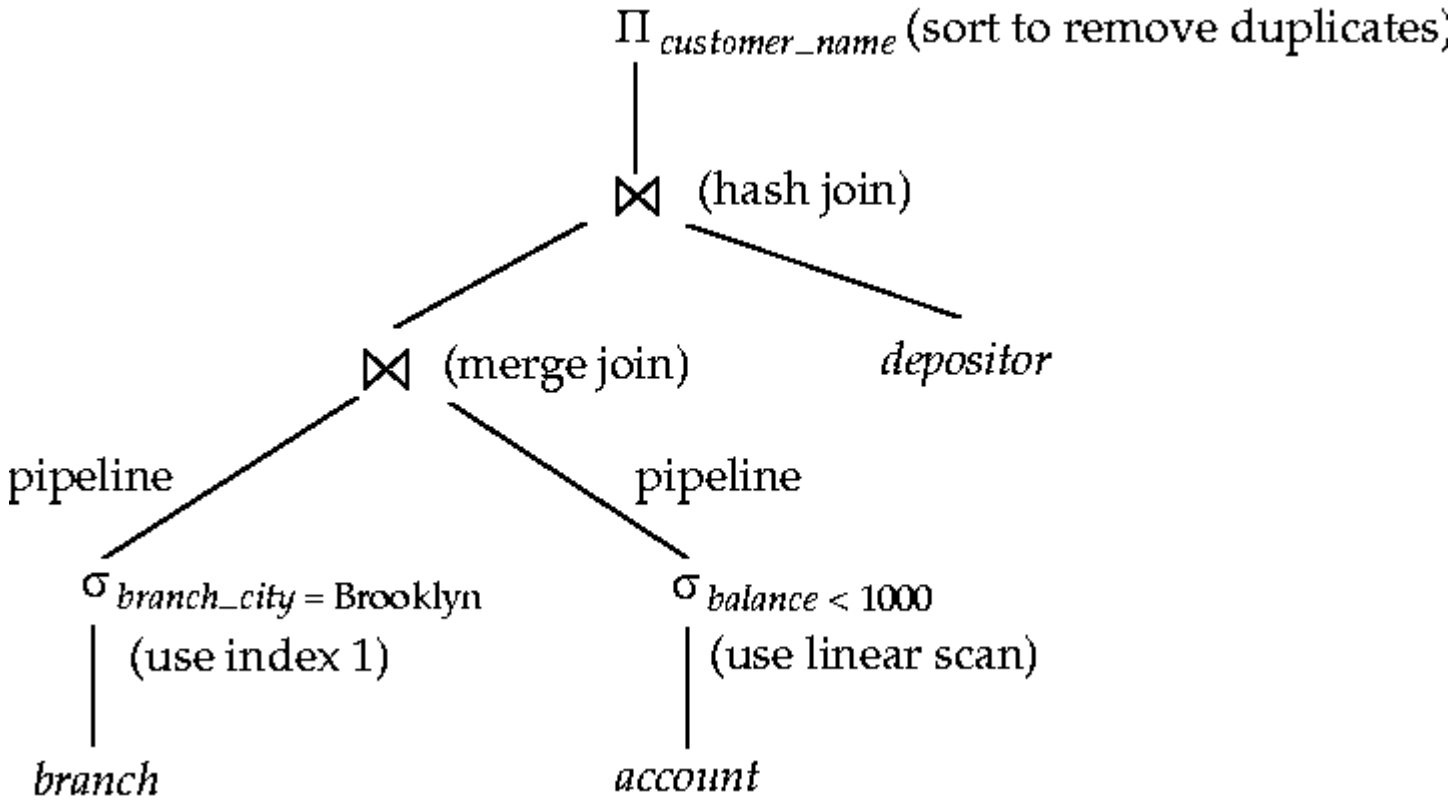
$$\prod_{balance}(\sigma_{balance<2500}(account))$$

Parsing and translation (cont.)

- Evaluation-plan: Annotated expression specifying detailed evaluation strategy.
  - e.g., can use an index on *balance* to find accounts with balance < 2500,
  - or can perform complete relation scan and discard accounts with balance $\geq$ 2500

Query Optimization

- An **evaluation plan** defines exactly what algorithm is used for each operation, and how the execution of the operations is coordinated.



Query Optimization

- Amongst all equivalent evaluation plans choose the one with lowest cost.
  - Cost is estimated using statistical information from the database catalog
    - e.g. number of tuples in each relation, size of tuples, etc.
    - How to measure query costs
    - How to optimize queries, that is, how to find an evaluation plan with lowest estimated cost

Query Optimization

- Estimation of plan cost based on:
  - Statistical information about relations. Examples:
    - number of tuples, number of distinct values for an attribute
  - Statistics estimation for intermediate results
    - to compute cost of complex expressions
  - Cost formulae for algorithms, computed using statistics

Evaluation

- The query-execution engine takes a query-evaluation plan, executes that plan, and returns the answers to the query.
- Parsed execution plan for previously executed SQL statements is stored in Shared pool (a portion of memory or buffer).
  - If a new SQL statement (query) is exactly the same string as the one in the shared pool, no need to call optimizer and recalculate the execution plan for the SQL statement.

Measures of Query Cost

- Cost is generally measured as total elapsed time for answering query
- Factors contribute to time cost
  - *Disk accesses*
    - *How does the index/hashing approach impact?*
  - *CPU*
  - Network *communication*

Measures of Query Cost

- Typically disk access is the predominant cost, and is also relatively easy to estimate.
- Measured by taking into account
  - Number of seeks * average-seek-cost
  - Number of blocks read* average-block-read-cost
  - Number of blocks written*average-block-write-cost
    - Cost to write a block is greater than cost to read a block
      - data is read back after being written to ensure that the write was successful

Advantages of Using the DBMS Approach (cont'd.)

- Providing backup and recovery
  - **Backup and recovery subsystem** of the DBMS is responsible for recovery

Backup

A ***procedure*** for making extra copies of data for the purpose of restoration in case of loss or damage.

Backup Modes

- Hot backup
  - allows backup of the database while the database is running and available to users.
  - performance degrades during the backup period
  - takes longer than a cold backup
- Cold backup
  - requires database shutdown before backup begins
  - physical files are backed up while shutdown
  - database is unavailable to users during backup period
  - faster than a hot backup

Backup Types

- Complete (Full)
  - copy all database and related files
  - delete the archive log files
- Cumulative (Differential)
  - copy blocks that have changed since last full backup
    or
  - copy all archive log files generated since last full backup
- Incremental
  - copy blocks that have change since the last partial backup
    or
  - copy all log files generated since last partial backup

Let's Talk Strategy

- **Ideally** we'd like to backup everything all the time and keep it around forever.
- **Realistically**, we cannot do that.
- You need a combination of **short-term** and **long-term** strategies. For example:
  - 1 Full back up per month
  - Differential back up per day

Database Recovery

- Process of restoring database from either cold or hot backup files
- Necessary after database failure
- Cold backup recovery:
  - Shut down database
  - Restore backed up control file and datafiles to database server
  - Restart database

Advantages of Using the DBMS Approach (cont'd.)

- Providing multiple user interfaces
  - **Graphical user interfaces (GUIs)**
- Representing complex relationships among data
  - May include numerous varieties of data that are interrelated in many ways

Complex Relations

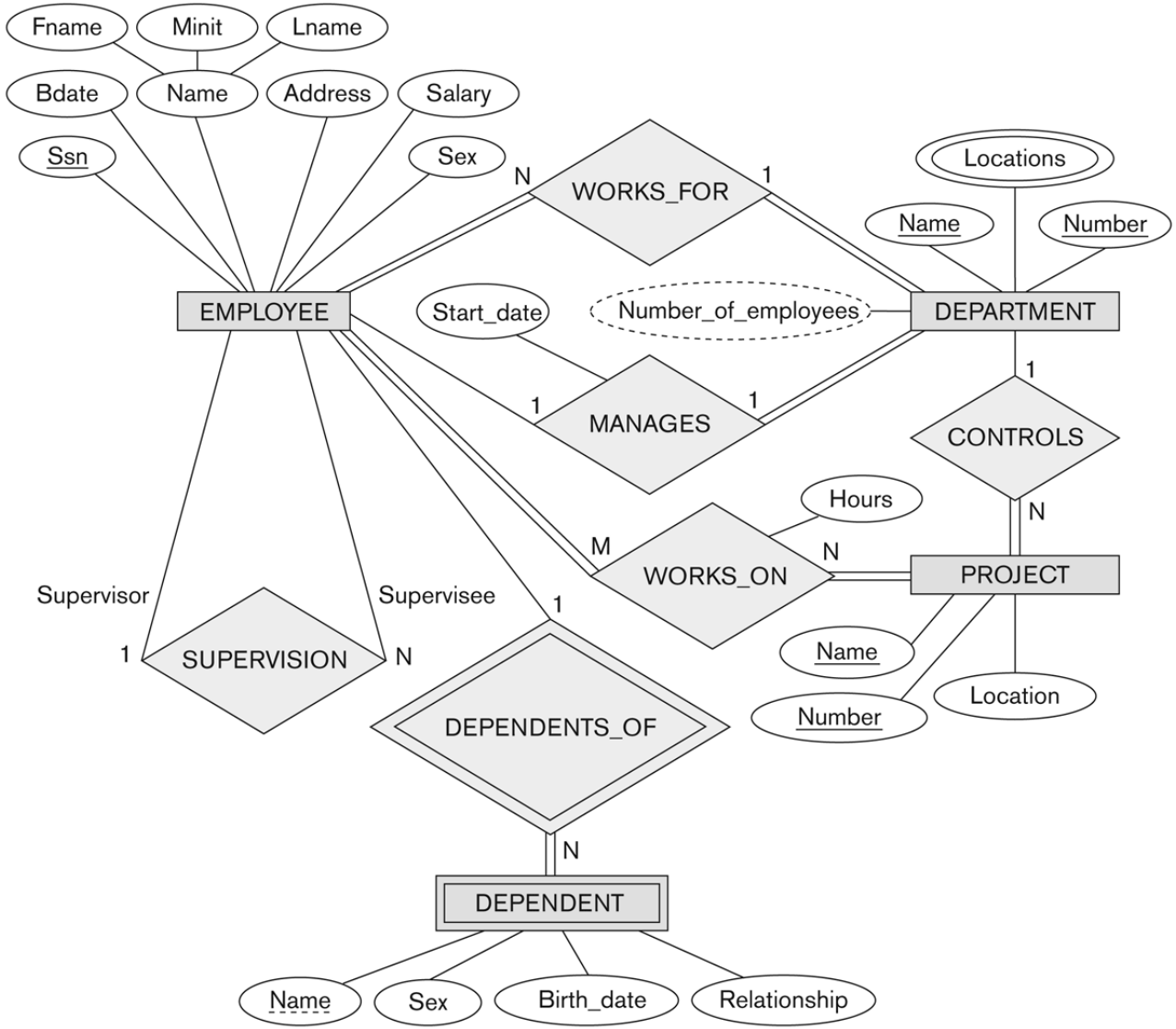
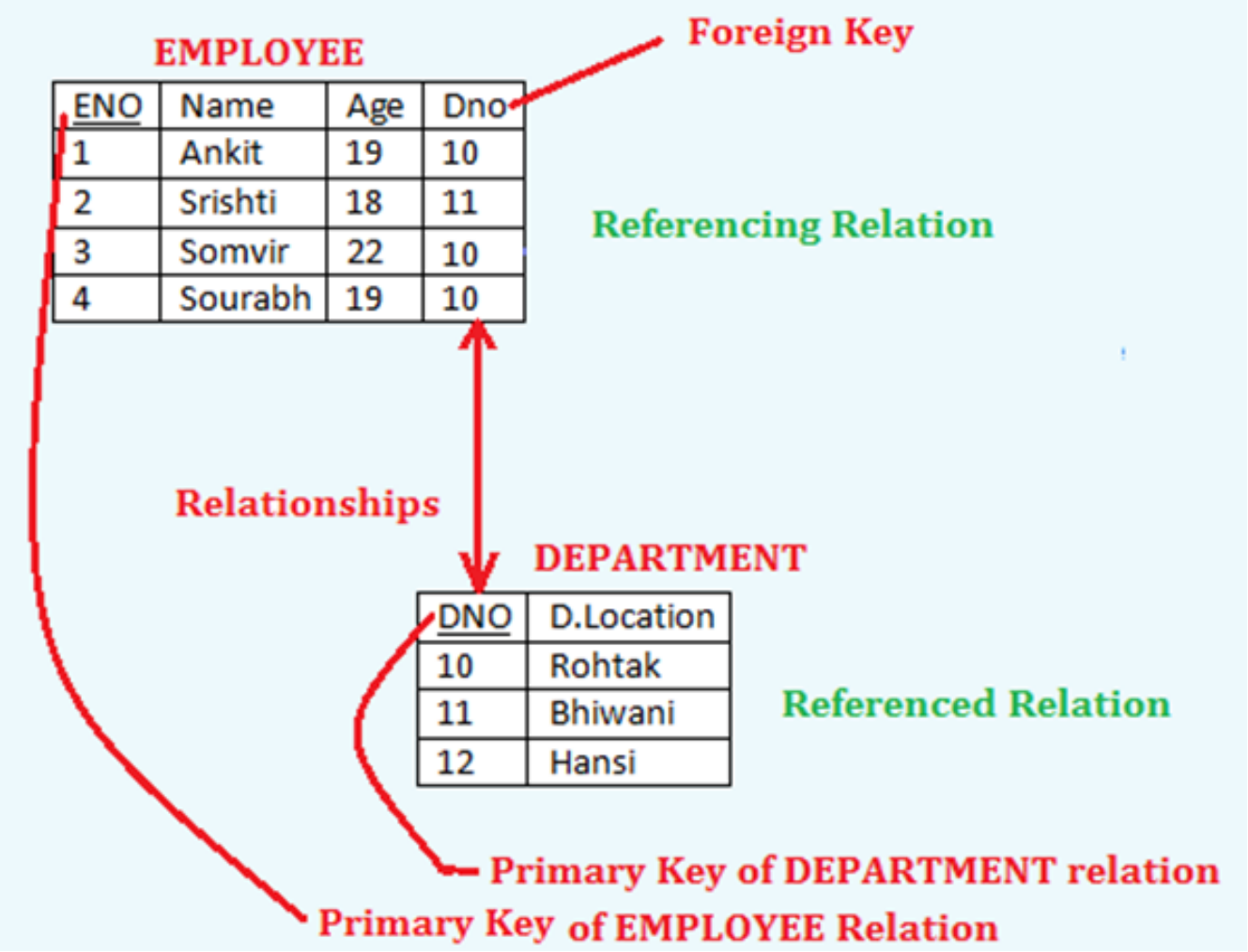


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Advantages of Using the DBMS Approach (cont'd.)

- Enforcing **integrity constraints**
  - **Referential integrity** constraint
    - Every section record must be related to a course record
  - **Key** or **uniqueness** constraint
    - Every course record must have a unique value for Course_number
  - **Business rules**
  - **Inherent rules** of the data model

Constraints (Example)



DOMAIN CONSTRAINTS

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10
5	Pooja	10	11

DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

Insertion into EMPLOYEE table is not allowed, Because this insertion violates the domain constraints as the employee age cannot be less than 18 years.

Key Constraints

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

3	Anuja	19	11

Insertion into EMPLOYEE table is not allowed, Because this insertion violates the key constraints as an employee with ENO(Primary Key) 3 already exists.

Entity Constraints

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

NULL	Kavya	21	10

Insertion into EMPLOYEE table is not allowed, Because this insertion violates the Entity Integrity constraints or Integrity Rule 1 as the primary key(ENO) cannot contain a null value.

Referential Integrity Constraints

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

6	Ajit	19	16

Insertion into EMPLOYEE table is not allowed, Because this insertion violates the Referential Integrity constraints or Integrity Rule 2 as there is no row or tuple with DNO=15 exists in DEPARTMENT relation.

Advantages of Using the DBMS Approach (cont'd.)

- Permitting inferencing and actions using rules
  - **Deductive database systems**
    - Provide capabilities for defining deduction **rules**
    - Inferencing new information from the stored database facts

- **What is a deductive database system?**

A deductive database can be defined as an advanced database augmented with an inference system.

By evaluating rules against facts, new facts can be derived, which in turn can be used to answer queries. It makes a database system more powerful.

Deductive databases: Example

Facts: parent(x, y) means that y is x's parent

```
parent(peter,mary).
parent(peter,paul).
parent(mary,john).
parent(paul,joan).
```

Rules: ancestor(x, y) means that y is x's ancestor

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z),ancestor(Z,Y).
```

Queries: (1) ancestors of Peter, (2) descendants of Joan

```
?- ancestor(peter,?).
?- ancestor(?,joan).
```

DEDUCTIVE DATABASE

- **DATALOG**: This is a programming language used in deductive database
  - It is part of another language called Prolog and incorporates basic logic principles for data integration, database queries, etc.
  - Datalog has found new application in Data integration, Declarative Networking, Program Analysis, and Commercial and Academic Systems2

DATALOG

- The general syntax for Datalog rule is given below

ancestor(X,Y) :- parent(X,Y).

ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).

- These two lines define two facts, i.e. things that always hold "the parent of mary is bill and the parent of john is mary".

parent(bill, mary).

parent(mary, john).

Advantages of Using the DBMS Approach (cont'd.)

- **Trigger**
  - Rule activated by updates to the table
  - Mostly used for maintaining the integrity constraint

SQL Triggers

- To monitor a database and take a corrective action when a condition occurs
  - Examples:
    - Limit the salary increase of an employee to no more than 5% raise

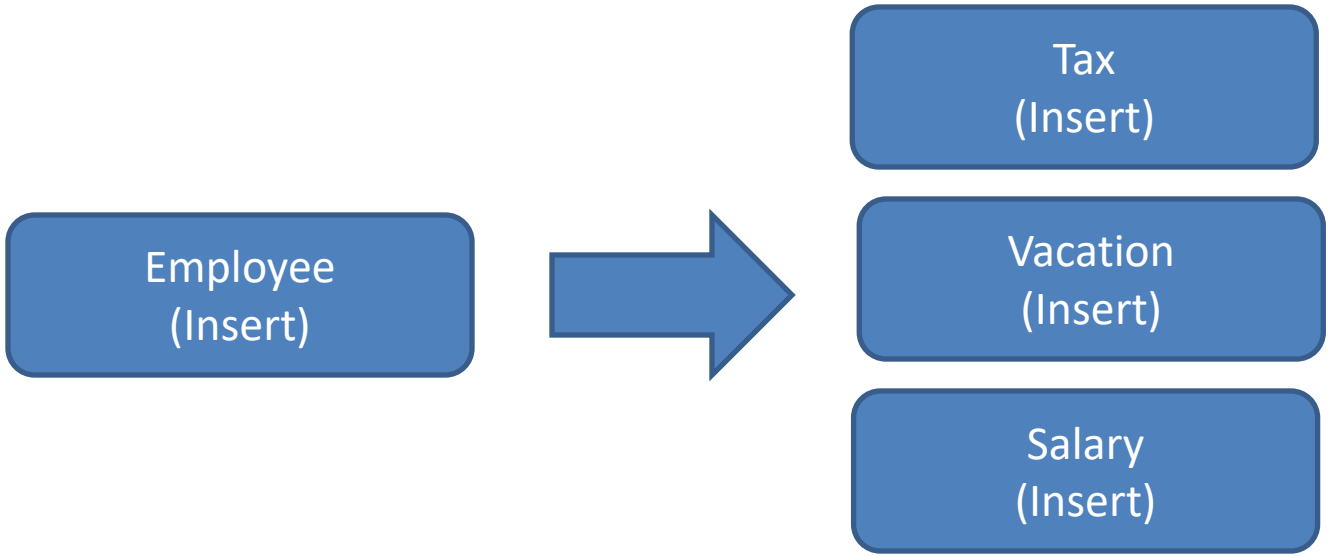
```
CREATE TRIGGER trigger-name
    trigger-time trigger-event
    ON table-name
    FOR EACH ROW
        trigger-action;
```

trigger-time $\in$ {BEFORE, AFTER}

trigger-event $\in$ {INSERT,DELETE,UPDATE}

Trigger (Example)

- When a new record (representing a new worker) is added to the employees table, new records should also be created in the tables of the taxes, vacations and salaries.



SQL Triggers: Another Example

- Create a trigger to update the total salary of a department when a new employee is hired:

```
mysql> delimiter |
mysql> create trigger update_salary
    -> after insert on employee
    -> for each row
    -> begin
    ->       if new.dno is not null then
    ->           update deptsal
    ->           set totalsalary = totalsalary + new.salary
    ->           where dnumber = new.dno;
    ->       end if;
    -> end |
Query OK, 0 rows affected (0.06 sec)

mysql> delimiter ;
```

- The keyword “new” refers to the new row inserted

Advantages of Using the DBMS Approach (cont'd.)

- **Stored procedures**
  - More involved procedures to enforce rules

Stored Procedures & Functions

- **What is stored procedure?**
  - Piece of code stored inside the DBMS
  - SQL allows you to define **procedures** and **functions** and store them inside DBMS

- **Advantages**
  - **Reusability**: do not need to write the code again and again
  - Programming language-like environment
    - Assignment, Loop, For, IF statements
  - Call it whenever needed
    - From select statement, another procedure, or another function

Creating A Stored Procedure

If exists, then drop it and create it again

'IS' or 'AS' both are valid

```
CREATE [OR REPLACE] PROCEDURE <procedureName> (<paramList>) [IS| AS]
      <localDeclarations>
Begin
      <procedureBody>;
End;
/
```

A parameter in the paramList is specified as:

<name> <mode> <type>

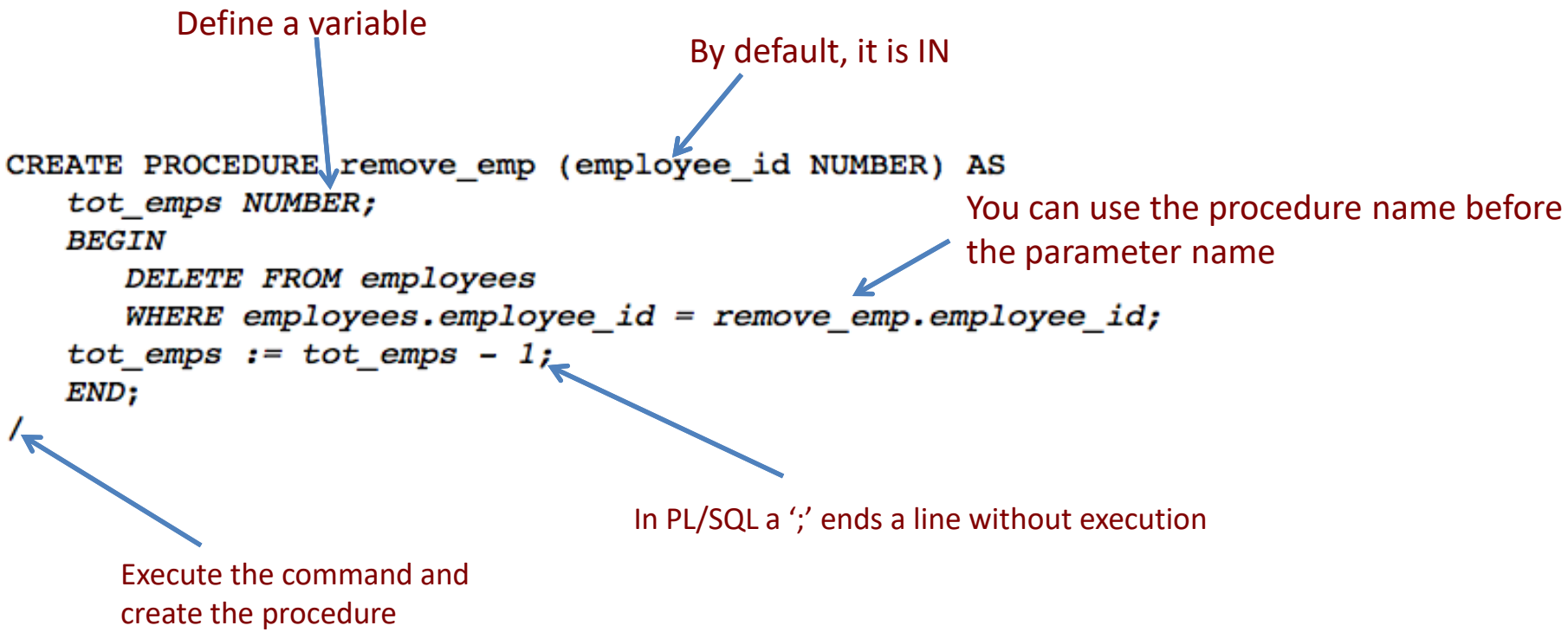
Mode:

IN ➔ input parameter (default)

OUT ➔ output parameter

INOUT ➔ input and output parameter

Example I


Define a variable
By default, it is IN
CREATE PROCEDURE remove_emp (employee_id NUMBER) AS
tot_emps NUMBER;
BEGIN
DELETE FROM employees
WHERE employees.employee_id = remove_emp.employee_id;
tot_emps := tot_emps - 1;
END;
/
You can use the procedure name before the parameter name
In PL/SQL a ';' ends a line without execution
Execute the command and create the procedure

Advantages of Using the DBMS Approach (cont'd.)

- Additional implications of using the database approach
  - Reduced application development time
  - Flexibility
  - Availability of up-to-date information
  - Economies of scale

Introduction to Databases

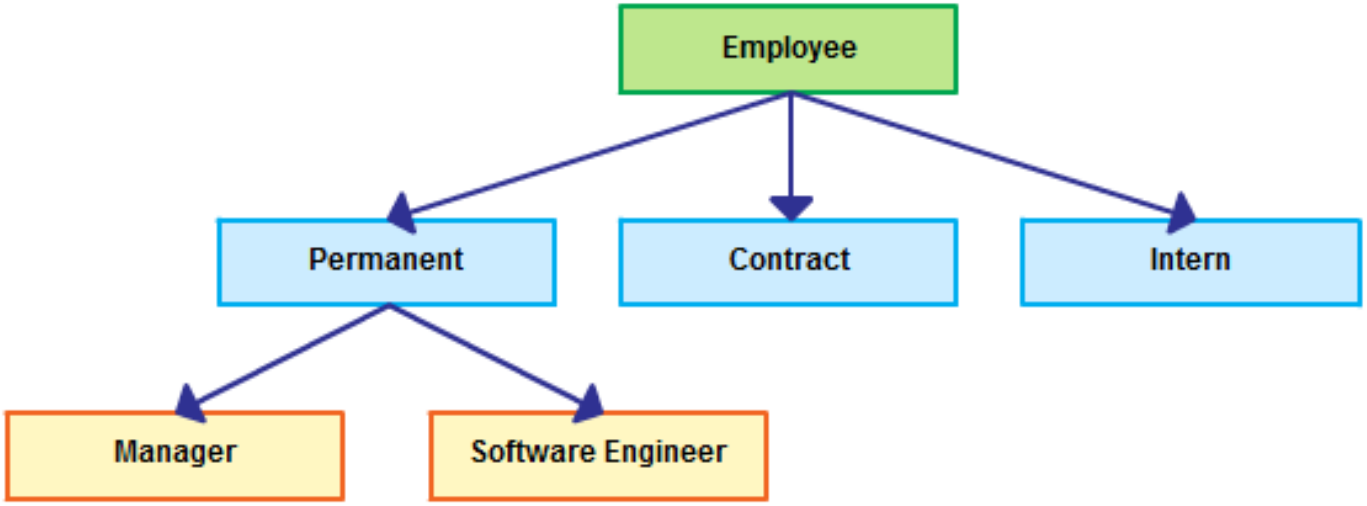
- Introduction
- An Example
- Characteristics of the Database Approach
- Actors on the Scene
- Workers behind the Scene
- Advantages of Using the DBMS Approach
- A Brief History of Database Applications
- When Not to Use a DBMS

A Brief History of Database Applications

- Early database applications using hierarchical and network systems
  - Large numbers of records of similar structure

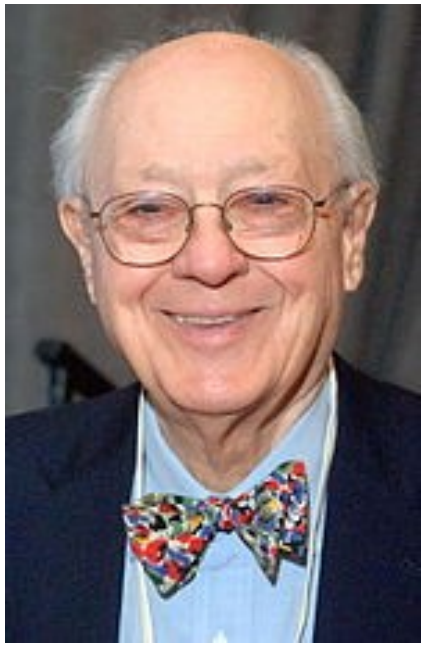
Hierarchical Model

- Developed by IBM in the 1960s
- Used in early mainframe
- Records' relationships form a treelike model

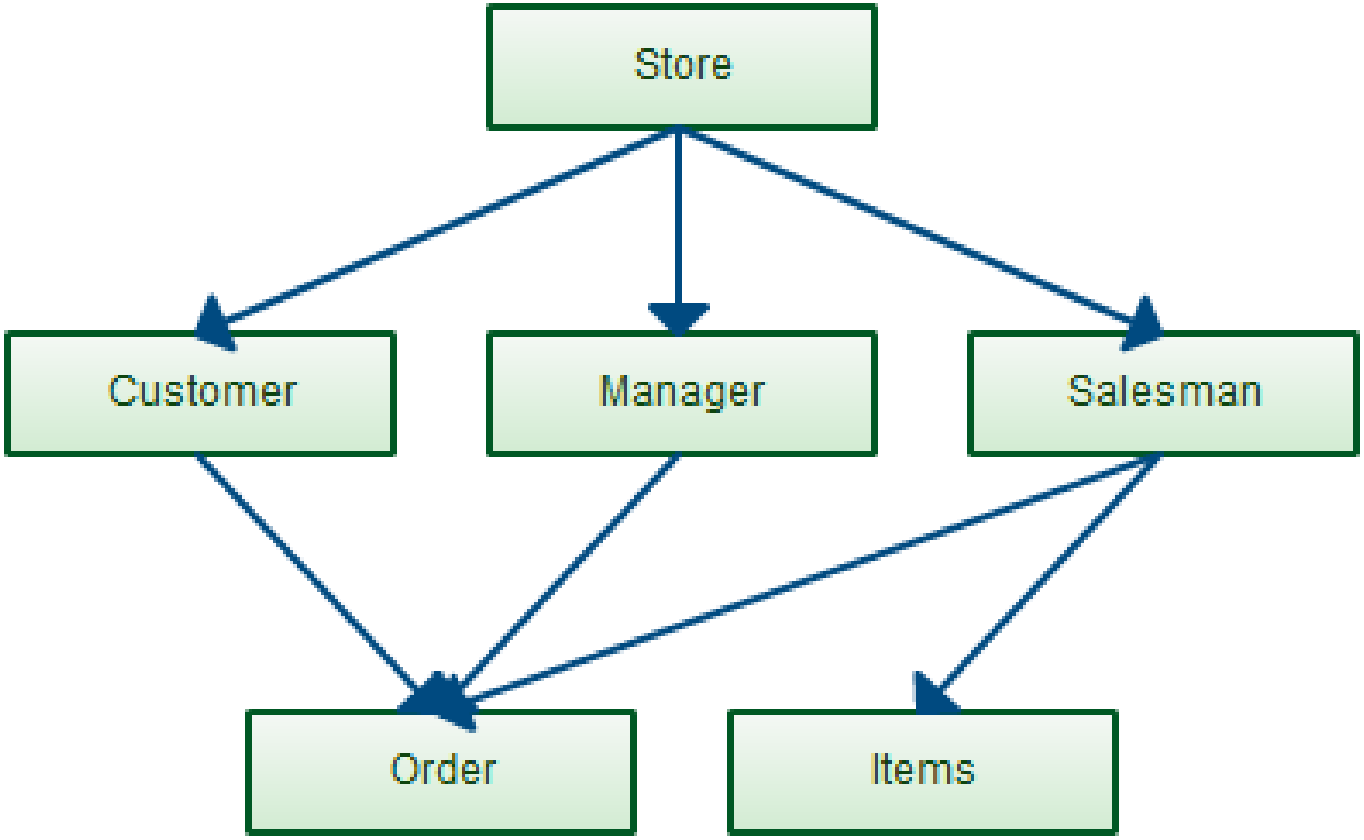


Network Model

- Proposed By Charles Bachman
- Published in CODASYL 1969
- Graph
  - Nodes: Object Types
  - Edges: Relationship Types



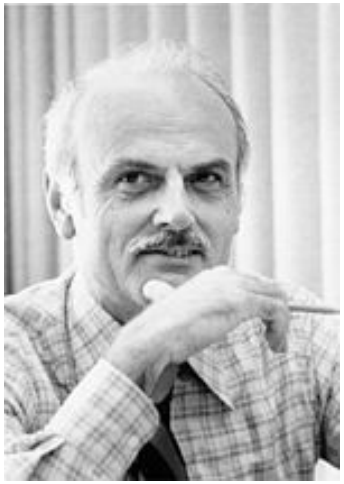
Network Model



A Brief History of Database Applications

- Providing data abstraction and application flexibility with relational databases
  - Separates physical storage of data from its conceptual representation
  - Provides a mathematical foundation for data representation and querying

Relational Model



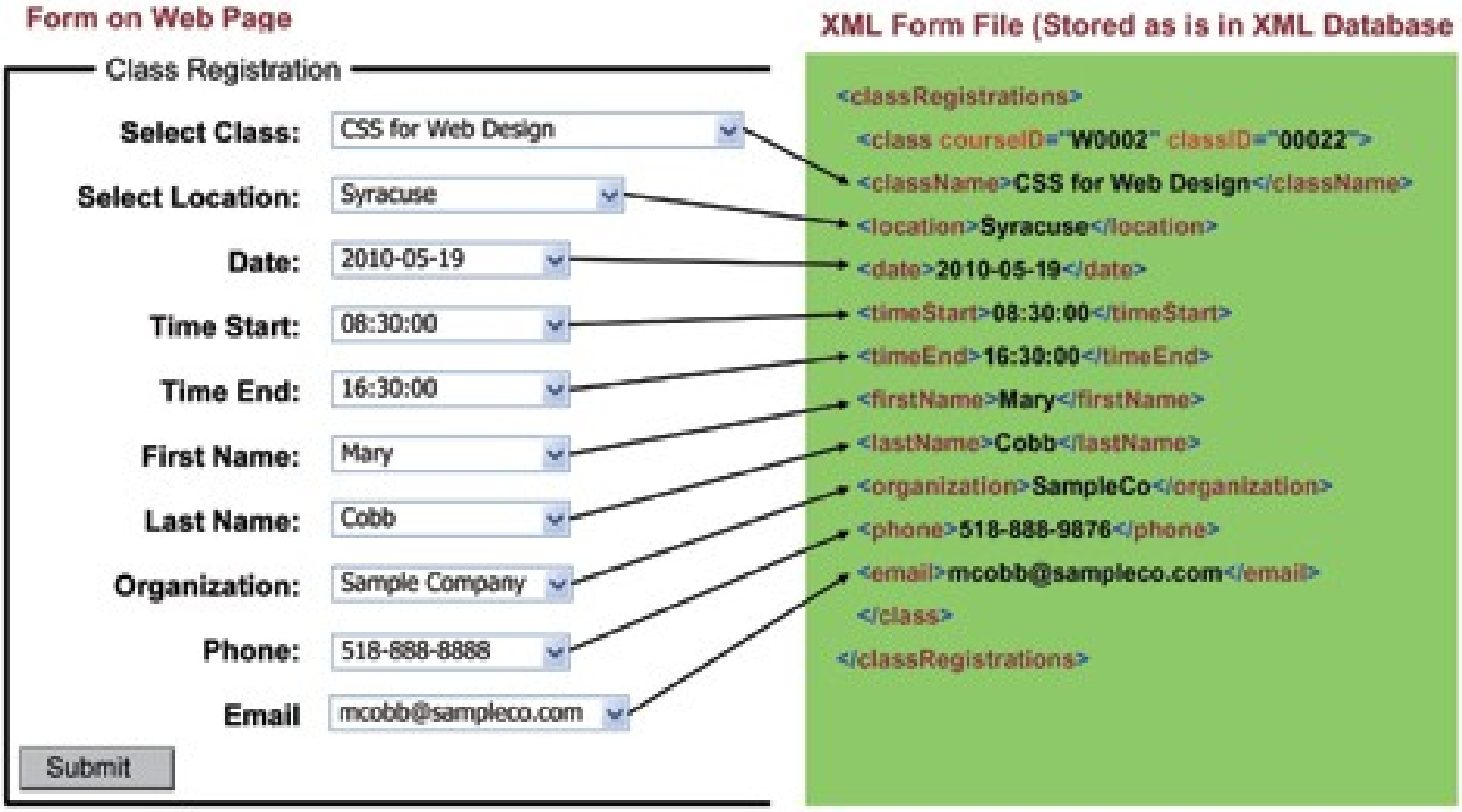
- Described in 1969 by Edgar F. Codd
- Fundamental assumption:
  - All data is represented as mathematical

    n-ary relations:
    - an n-ary relation being a subset of the Cartesian product of n domains.

A Brief History of Database Applications (cont'd.)

- Object-oriented applications and the need for more complex databases
  - Used in specialized applications: engineering design, multimedia publishing, and manufacturing systems
- Interchanging data on the Web for e-commerce using XML
  - Extended markup language (XML) primary standard for interchanging data among various types of databases and Web pages

XML


Form on Web Page
Class Registration
Select Class:
CSS for Web Design
Select Location:
Syracuse
Date:
2010-05-19
Time Start:
08:30:00
Time End:
16:30:00
First Name:
Mary
Last Name:
Cobb
Organization:
Sample Company
Phone:
518-888-8888
Email
mcobb@sampleco.com
Submit
XML Form File (Stored as is in XML Database
<classRegistrations>
<class courseID="W0002" classID="00022">
<className>CSS for Web Design</className>
<location>Syracuse</location>
<date>2010-05-19</date>
<timeStart>08:30:00</timeStart>
<timeEnd>16:30:00</timeEnd>
<firstName>Mary</firstName>
<lastName>Cobb</lastName>
<organization>SampleCo</organization>
<phone>518-888-9876</phone>
<email>mcobb@sampleco.com</email>
</class>
</classRegistrations>

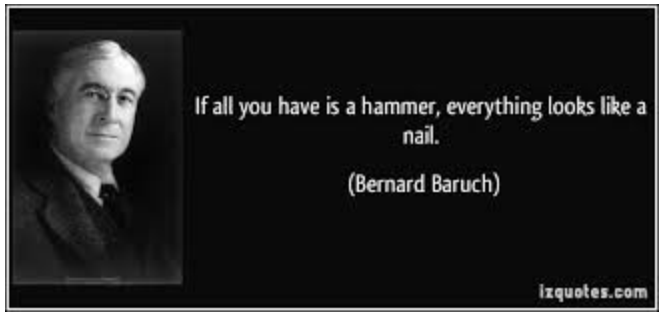
A Brief History of Database Applications (cont'd.)

- Extending database capabilities for new applications
  - Extensions to better support specialized requirements for applications
  - **Enterprise resource planning (ERP)**
  - **Customer relationship management (CRM)**
- Databases versus information retrieval
  - **Information retrieval (IR)**
    - Deals with books, manuscripts, and various forms of library-based articles

Information Retrieval

Introduction to Databases

- Introduction
- An Example
- Characteristics of the Database Approach
- Actors on the Scene
- Workers behind the Scene
- Advantages of Using the DBMS Approach
- A Brief History of Database Applications
- When Not to Use a DBMS


If all you have is a hammer, everything looks like a nail.
(Bernard Baruch)
izquotes.com

When Not to Use a DBMS

- More desirable to use regular files for:
  - Simple, well-defined database applications not expected to change at all
  - Real-time requirements that may not be met because of DBMS overhead
  - Embedded systems with limited storage capacity
  - No multiple-user access to data

Summary

- Database
  - Collection of related data (recorded facts)
- DBMS
  - Generalized software package for implementing and maintaining a computerized database
- Several categories of database users
- Database applications have evolved
  - Current trends: IR, Web

Why take this class?

A. Database systems are at the core of CS
B. They are incredibly important to society
C. The topic is intellectually rich
D. A capstone course for undergrad
E. It isn't that much work
F. Looks good on your resume

Let's spend a little time on each of these

Why take this class?

A. Database systems are the core of CS

- Shift from computation to information
  - True in corporate computing for years
  - Web, p2p made this clear for personal computing
  - Increasingly true of scientific computing
- Need for DB technology has exploded in the last years
  - Corporate: retail swipe/clickstreams, "customer relationship mgmt", "supply chain mgmt", "data warehouses", etc.
  - Web:not just "documents". Search engines, e-commerce, blogs, wikis, other "web services".
  - Scientific: digital libraries, genomics, satellite imagery, physical sensors, simulation data
  - Personal: Music, photo, & video libraries. Email archives. File contents ("desktop search").

Why take this class?

B. DBs are incredibly important to society

- "Knowledge is power." -- Sir Francis Bacon

- "With great power comes great responsibility." -- SpiderMan's Uncle Ben

Policy-makers should understand technological possibilities.
Informed Technologists needed in public discourse on usage.

Why take this class?

C. The topic is intellectually rich.

- representing information
  - data modeling
- languages and systems for querying data
  - complex queries & query semantics*
  - over massive data sets
- concurrency control for data manipulation
  - controlling concurrent access
  - ensuring transactional semantics
- reliable data storage
  - maintain data semantics even if you pull the plug

* semantics: the meaning or relationship of meanings of a sign or set of signs

Why take this class?

D. The course is a capstone.

- We will see
  - Algorithms and cost analyses
  - System architecture and implementation
  - Resource management and scheduling
  - Computer language design, semantics and optimization
  - Applications of AI topics including logic and planning
  - Statistical modeling of data

Why take this class?

~~**E. It isn't that much work.**~~

- Bad news: It is a lot of work.

- Good news:
  - Load balanced with most other classes

Why take this class?

F. Looks good on my resume.

- Yes, but why? This is not a course for:
  - Oracle administrators
  - IBM DB2 engine developers
    - Though it's useful for both!
- It is a course for well-educated computer scientists
  - Database system concepts and techniques increasingly used "outside the box"
    - Ask your friends at Microsoft, Yahoo!, Google, Apple, etc.
    - Actually, they may or may not realize it!
  - A rich understanding of these issues is a basic and (un?)fortunately unusual skill.

Quiz

- Imagine a mini-world of three entities:
  - Student, Course, Professor
- Explain in details the relationships and constraints among these three entities.
- Draw diagrams if needed.