Overview of Database Management

Ashraf Aboulnaga

David R. Cheriton School of Computer Science
University of Waterloo

CS 348
Introduction to Database Management
Winter 2013

Notes

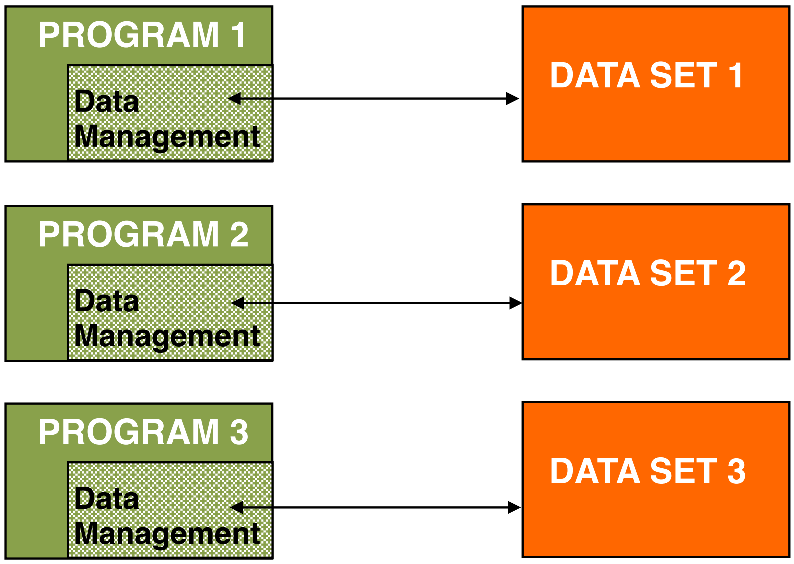
What is “Data”?

- ANSI definition of data:
  1. A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by humans or by automatic means.
  2. Any representation such as characters or analog quantities to which meaning is or might be assigned. Generally, we perform operations on data or data items to supply some information about an entity.
- Volatile vs persistent data
  - Our concern is primarily with persistent data

Notes

Early Data Management – Ancient History

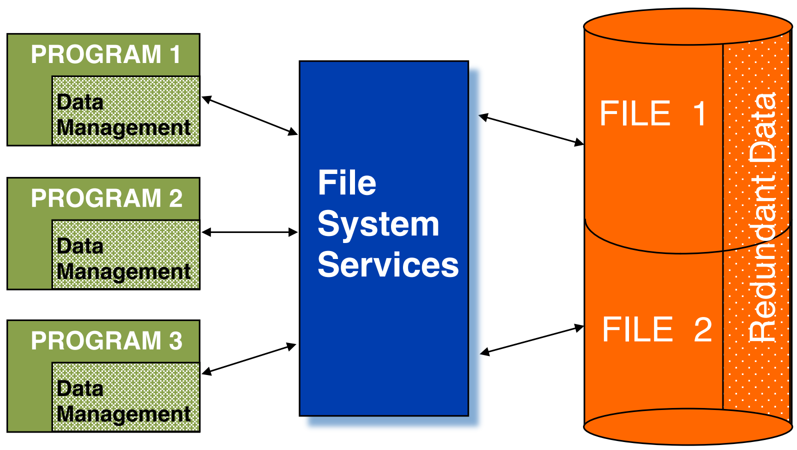
- Data are not stored on disk
- One data set per program. High data redundancy



Notes

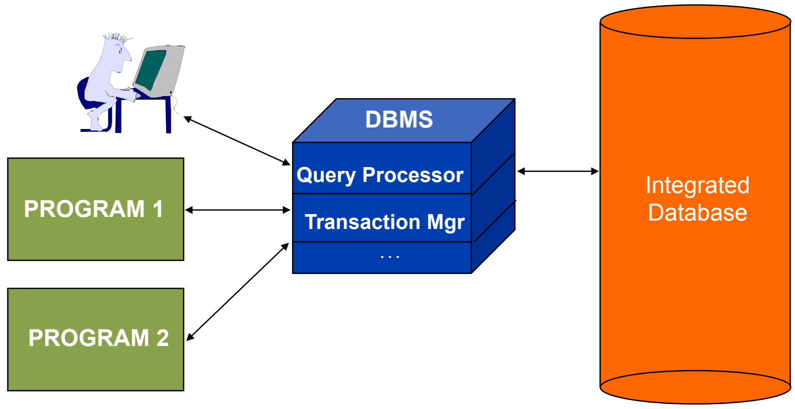
File Processing – More Recent History

- Data are stored in files with interface between programs and files.
- Various access methods exist (e.g., sequential, indexed, random).
- One file corresponds to one or several programs.



Notes

Database Approach



Notes

What is a Database?

Definition (Database)

A *large* and *persistent* collection of (more-or-less similar) pieces of information organized in a way that facilitates efficient *retrieval* and *modification*.

The structure of the database is determined by the abstract data model that is used

Examples:

- a file cabinet
- a library system
- a personnel management system

Definition (Database Management System (DBMS))

A program (or set of programs) that manages details related to storage and access for a database.

Notes

Data Model

- Formalism that defines what the structure of the data are $\Longrightarrow$ schema
  - within a file
  - between files
- File systems can at best specify data organization within one file
- Alternatives for business data
  - Hierarchical; network
  - Relational
  - Object-oriented (more recently object-relational)

Notes

Schema and Instance

Definition (Schema)

A **schema** is a description of the data interface to the database (i.e., how the data is organized).

Definition (Instance)

A **database instance** is a database (real data) that conforms to a given schema.

A schema can have many instances.

Notes

Example – Relational

- Schema
  - EMP(ENO, ENAME, TITLE)
  - PROJ(PNO, PNAME, BUDGET)
  - WORKS(ENO, PNO, RESP, DUR)
- Instance

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

WORKS

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Notes

Application of Databases

Original
- inventory control
- payroll
- banking and financial systems
- reservation systems

More recent
- computer aided design (CAD)
- software development (CASE, SDE/SSE)
- telecommunication systems
- e-commerce
- dynamic/personalized web content

Notes

Application of Databases (cont'd)

Common Circumstances:

- There is lots of data (mass storage)
- Data is formatted
- Requirements:
  - persistence and reliability
  - efficient and concurrent access
- Issues:
  - many files with different structure
  - shared files or replicated data
  - need to exchange data (translation programs)

Notes

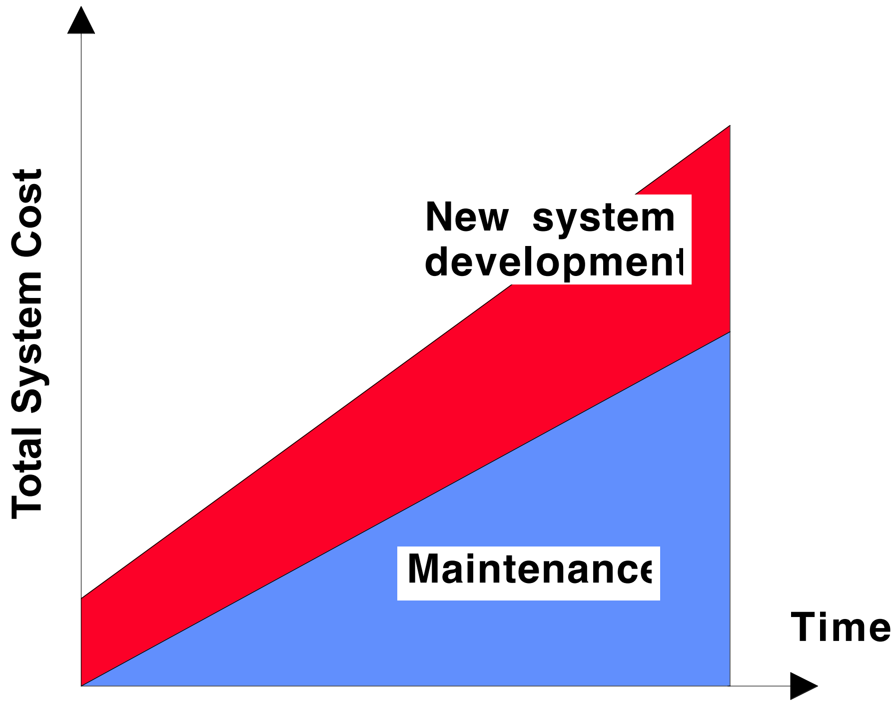
Why Database Technology?

- Data constitute an organizational asset $\Longrightarrow$ Integrated control
  - Reduction of redundancy
  - Avoidance of inconsistency
  - Data integrity
  - Sharability
  - Improved security
- Programmer productivity $\Longrightarrow$ Data Independence
  - Programmers do not have to deal with data organization

Notes

Programmer Productivity

- High data independence



Notes

Brief History of Data Management: Ancient

- 2000 BC: Sumerian Records
- 350 BC: Syllogisms (Aristotle)
- 296 BC: Library of Alexandria
- 1879: Modern Logic (Frege)
- 1884: U.S. Census (Hollerith)
- 1941: Model Theory (Tarski)

Notes

Brief History of Data Management: 1950s

First generation 50's and 60's

- batch processing
- sequential files and tape
- input on punched cards

Second generation (60's)

- disk enabled random access files
- new access methods (ISAM, hash files)
- mostly batch with some interactivity
- independent applications with separate files
- growing applications base

Notes

Brief History of Data Management: 1960s (cont'd)

As the application base grows, we end up with

- many shared files
- a multitude of file structures
- a need to exchange data among applications

This causes a variety of problems

- redundancy: multiple copies
- inconsistency: independent updates
- inaccuracy: concurrent updates
- incompatibility: multiple formats
- insecurity: proliferation
- inauditability: poor chain of responsibility
- inflexibility: changes are difficult to apply

Notes

Brief History of Data Management: 1960s (cont'd)

- Hierarchical data model
  - IBM's Information Management System (IMS): concurrent access
  - only allows 1:N parent-child relationships (i.e. a tree)
  - hierarchy can be exploited for efficiency
  - queries navigate up and down trees—one record at a time
  - data access language embedded in business processing language
  - difficult to express some queries
- Network data model
  - Charles Bachman's Integrated Data Store (IDS)
  - model standardized by Conference On DAta SYstems Languages (CODASYL)
  - data organized as collections of sets of records
  - separation of physical data representation from users' view of data
  - pointers between records represent relationships
  - set types encoded as lists
  - queries navigate between records—still one record at a time

Notes

Database Management System

Idea

Abstracts common functions and creates a uniform well defined interface for applications accessing data.

1. Data Model
   all data stored in a well defined way
2. Access control
   only authorized people get to see/modify it
3. Concurrency control
   multiple concurrent applications access data
4. Database recovery
   nothing gets accidentally lost
5. Database maintenance

Notes

Three Level Schema Architecture

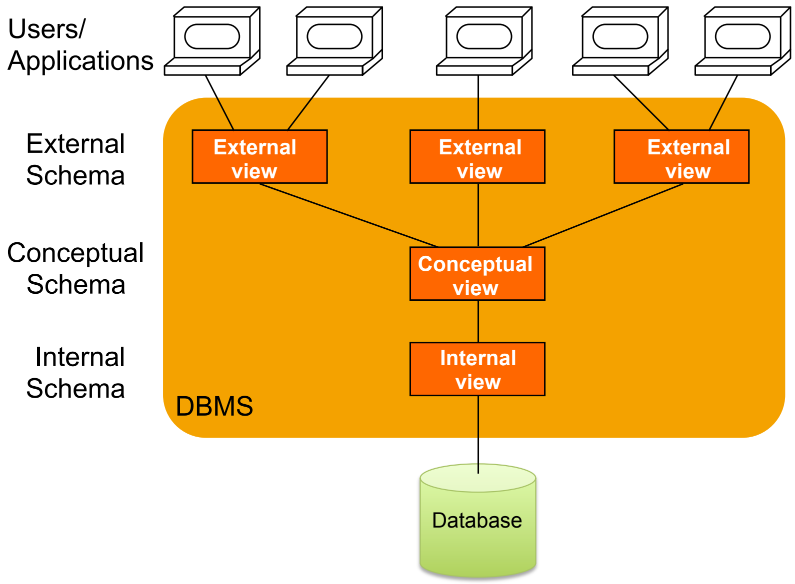
Definition (Schema)

A **schema** is a description of the data interface to the database (i.e., how the data is organized).

1. External schema (view): what the application programs and user see. May differ for different users of the same database.
2. Conceptual schema: description of the logical structure of *all* data in the database.
3. Physical schema: description of physical aspects (selection of files, devices, storage algorithms, etc.)

Notes

Three-level Schema Architecture (cont.)



Notes

Data Independence

Idea

Applications do not access data directly but, rather through an abstract data model *provided by the DBMS.*

Two kinds of data independence:

- **Physical:** applications immune to changes in storage structures
- **Logical:** applications immune to changes in data organization

Note

One of the most important reasons to use a DBMS!

Notes

Interfacing to the DBMS

Data Definition Language (DDL): for specifying schemas

- may have different DDLs for external schema, conceptual schema, internal schema
- information is stored in the **data dictionary**, or **catalog**

Data Manipulation Language (DML): for specifying queries and updates

- **navigational** (procedural)
- **non-navigational** (declarative)

Notes

Types of Database Users

End user:

- Accesses the database indirectly through forms or other query-generating applications, or
- Generates ad-hoc queries using the DML.

Application developer:

- Designs and implements applications that access the database.

Database administrator (DBA):

- Manages conceptual schema.
- Assists with application view integration.
- Monitors and tunes DBMS performance.
- Defines internal schema.
- Loads and reformats database.
- Is responsible for security and reliability.

Notes

Transactions

When multiple applications access the same data, undesirable results occur.

Example:

```
withdraw(AC,1000)         withdraw(AC,500)
  Bal := getbal(AC)
                            Bal := getbal(AC)
  if (Bal>1000)             if (Bal>500)
    <give-money>              <give-money>
    setbal(AC,Bal-1000)
                              setbal(AC,Bal-500)
```

Idea

Every application may think it is the sole application accessing the data. The DBMS should guarantee correct execution.

Notes

Transactions (cont'd)

Definition (Transaction)

An application-specified atomic and durable unit of work.

Properties of transactions ensured by the DBMS:

- **Atomic:** a transaction occurs entirely, or not at all
- **Consistency:** each transaction preserves the consistency of the database
- **Isolated:** concurrent transactions do not interfere with each other
- **Durable:** once completed, a transaction's changes are permanent

Notes

Summary

Using a DBMS to manage data helps:

- to remove common code from applications
- to provide uniform access to data
- to guarantee data integrity
- to manage concurrent access
- to protect against system failure
- to set access policies for data

Notes