INTRODUCTION TO DATABASES

CHAPTER 01

Collin Roberts
University of Waterloo

Chapter Outline

- Motivation for databases
- Characteristics of the Database Approach
- Database basics
- Data independence
- How to access a database
- When Not to Use a DBMS

Suggested Readings

- Textbook Chapter 01 and Chapter 02

Early Data Management

- Data hardcoded inside the program. One program, one data set

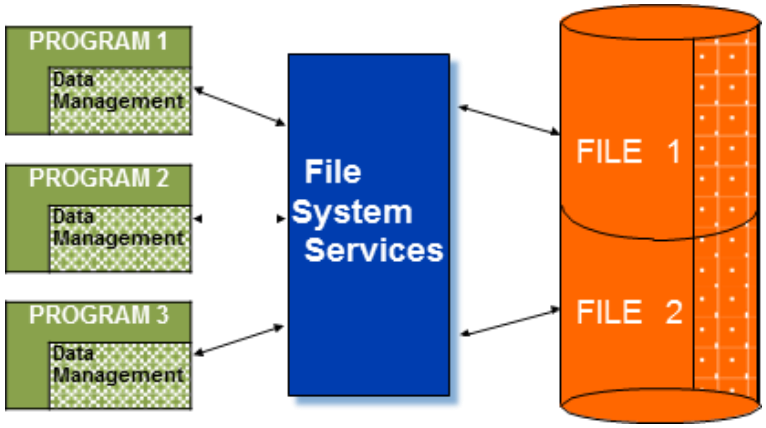

```
int Revenue( )
{
        int sales[ ] = [80, 81, 60, 99, 77];

        … …

}
```

File Processing

- Data are stored in files with interface between programs and files. One file corresponds to one or several programs




```
int Revenue( )
{
        int sales[ ] = Read data from sales file;
        … …
}
```

Problems with Storing Data in a File

- Can a file do the following?
  - Opened by many different applications
  - Displayed in different formats to different people
  - Part of the file is invisible/accessible to certain people but visible to others
    - Redundancy: store the same data in different places
    - Inconsistency: different values to the same attribute
  - Can two or more people modify the same file at the same time?
  - Do you know who accessed the file recently?

Scenario: Same Data Different Apps

- How do they share the same sales report?
  - John uses Adobe product
  - Jane uses MS Word
  - Joe uses Lotus Notes
  - Jess uses WPS

Image Source: http://www.theawl.com/2011/03/how-to-share-big-files

Scenario: Same Data Different Displays

- How do you produce these reports?
  - CFO wants sales data sorted according to region
  - CIO wants sales data sorted according to product
  - CEO wants sales data sorted according to quarter
  - VP production wants a sales report about binders
  - VP R&D wants a sales report for the past 3 months
- Later realized there is a mistake about the price of pencil, how many files/places do you need to make changes?

Traditional File Processing

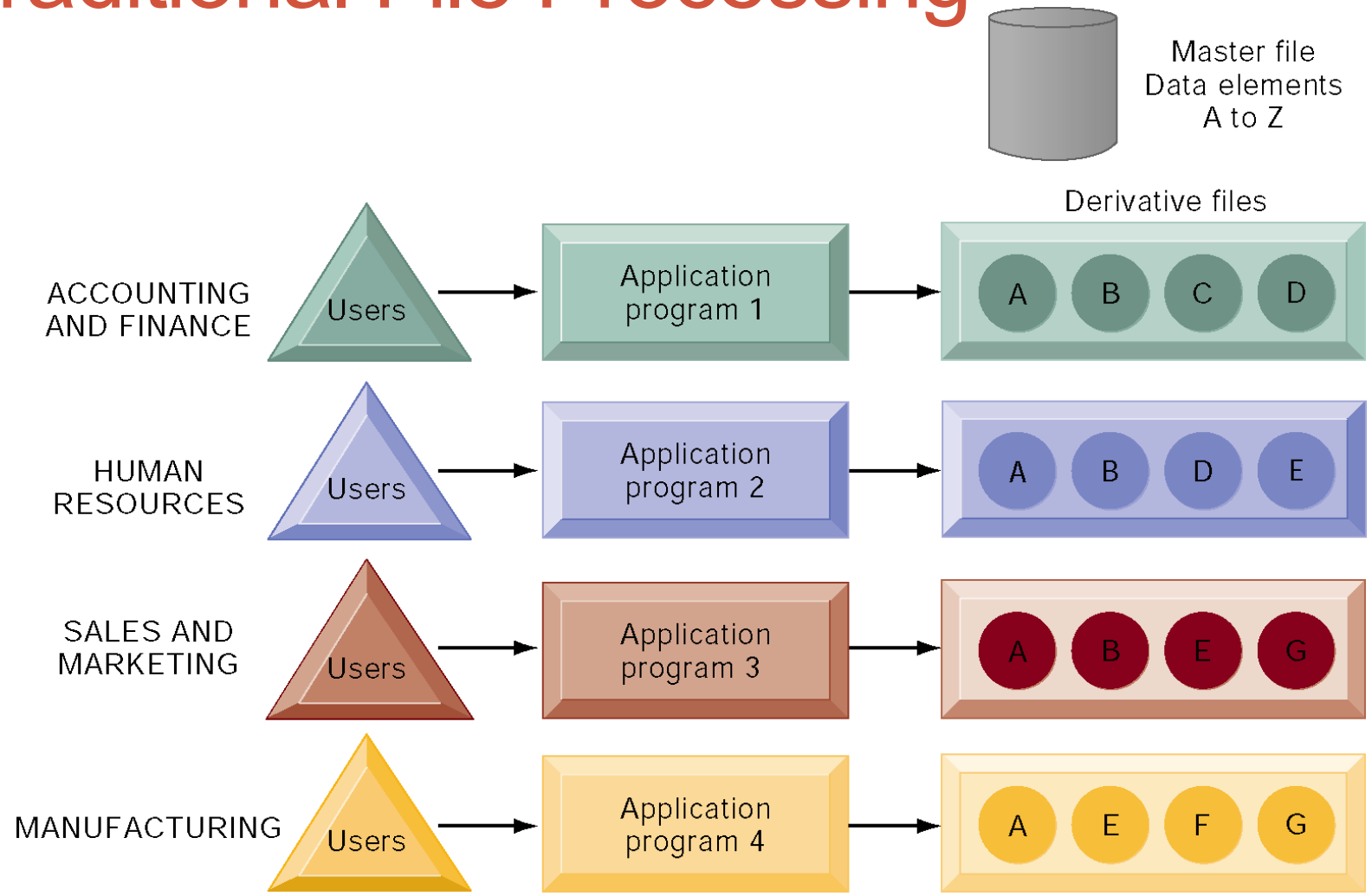


Figure 6-2

Data Redundancy

- **Data redundancy**: The presence of duplicate data in multiple data files so that the same data are stored in more than one place or location
  - **Data inconsistency**: The same attribute may have different values.

Image Source: https://www.shutterstock.com/image-vector/word-cloud-data-integrity-related-items-199701359

Scenario: what if several users access the same file at the same time

- Assumption: no DB used, everything stored in files
- You have an account with TD and the balance is $100.
- It is your birthday.
- You are withdrawing $50 in Waterloo to celebrate with your friends.
- At the same time, your grandpa in Toronto is depositing $90 into your account as your birthday gift.

- What could be your balance after both transactic completed?

Scenario: Deposit and Withdrawal in Details

Bank server in Montreal
Your balance Z = $100

Withdraw at ATM1 in Waterloo

1. X = Read your balance from server
2. X = X – 50
3. Store the new balance X to server

Deposit at ATM2 in Toronto

a. Y = Read your balance from server
b. Y = Y + 90
c. Store the new balance Y to server

Why Databases and DBMS?

Provide data integrity

- Reduce data redundancy and inconsistency

Provide data independence

- Separate program and data

Provide security, concurrent access and crash recovery

- Enable data sharing and high availability

Centralized data administration

- Back up and access
- Enforce policies

Reduce application development time

- Standard software packages readily available in the market

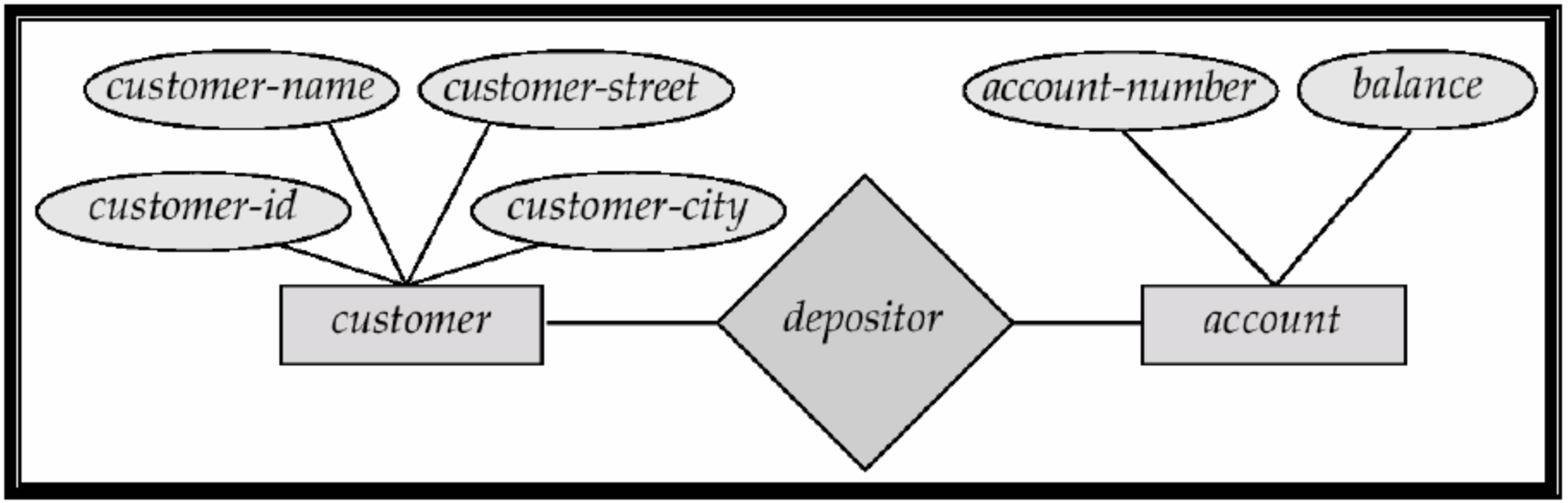
Databases and DBMS

- Database: a collection of related information stored in a structured form
  - The structure of information is described by data model that is expressed in schema
- Database Management System (DBMS): a collection of programs that manipulate a database
  - Set up the storage structures
  - Perform updates on data
  - Process queries (requests for data retrieval) from applications and users
- The DBMS provides a central point of access to the data
- Access 2010: Introduction to Databases

Data Model

- A collection of tools for describing
  - data
  - data relationships
  - data semantics
  - data constraints
- Entity-Relationship model: Example
  - model of real world: entities + relationships
  - widely used for database design
  - Usually converted to design in the relational model
- Other models:
  - Relational model
  - object-oriented model
  - semi-structured data models
  - Older models: network model and hierarchical model

Example:
- Entity-Relationship Model



Example:
- Relational Model

Customer-id	*customer-name*	*customer-street*	*customer-city*	*account-number*
192-83-7465	Johnson	Alma	Palo Alto	A-101
019-28-3746	Smith	North	Rye	A-215
192-83-7465	Johnson	Alma	Palo Alto	A-201
321-12-3123	Jones	Main	Harrison	A-217
019-28-3746	Smith	North	Rye	A-201

Instance and Schema

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - e.g., the database consists of information about a set of students and courses and the enrollment relationship between them)
  - Analogous to type information of a variable in a program
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable

Examples of Instance and Schema

- Schema
  - *Student* (*SID* integer, *name* string, *age* integer, *GPA* float)
  - *Course* (*CID* string, *title* string)
  - *Enroll* (*SID* integer, *CID* integer)
- Instance
  - { ⟨142, Bart, 10, 2.3⟩, ⟨123, Milhouse, 10, 3.1⟩, ...}
  - { ⟨CPS216, Advanced Database Systems⟩, ...}
  - { ⟨142, CPS216⟩, ⟨142, CPS214⟩, ...}

Three Layers of Schema

- Table only describes one layer of a relational database:
- Logical layer: what data to store, and its meaning
- Physical layer: how data is physically stored in storage units
  - Both physical and logical layers are invisible to users
- External view: how data is displayed
  - Only layer visible users
    - This is what you see when accessing a database
    - Often computed at runtime based on search criteria and access right of the user

Example: 3 Layers of University Database

- Logical schema (middle layer):
  - Students(sid: string, name: string, login: string,
  - birthday: date, gpa:real, phone: string)
  - Courses(cid: string, cname:string, credits:integer)
  - Enrolled(sid:string, cid:string, grade:string)

- Physical schema:
  - Data stored as bits and bytes on the disk.
  - How do we store this optimally for the storage medium: Spinning Hard Disk, SSD, Tape? DBMS does that for you
  - Stored in RAID 1 and indexing based primary keys

- External Schema (View):
  - StudentView(name: string, phone: string)
  - Course_info(cid: string, enrollment:integer)

Example: 3 Layers of Birthday Attribute

- Consider the attribute “birthday” in a table and an entry: June 20th 1990
  - At the middle layer (internal schema) of the 3 schema layer hierarchy (i.e tables). It is the easiest to understand
- However there is a layer below (the 0s and 1s on a disk) where data manifests itself physically!
- Above the table there is the external layer where different people/users can “view” the data differently
  - Different display formats for date
  - For example Age is derived from Date of Birth
    - Age is function of date of birth and “today”
    - Can be calculated as needed, thus people who only want to view the age can interpret the logical value of date of birth as needed according to a specified function
    - Age is an application specific concept in this scenario

Why have these three layers?

- Answer: Data independence!
  - Separation of logic, storage and presentation

- What happens if you change where or how your database is stored?
  - You don't have to change any applications that use it
  - Just like file systems (regardless of where the file is stored you can open it, who know how its stored physically?)
- Different people can view the data differently
  - Easier to write software
  - Easier to manage and control (e.g want some users to only see age but not exact date of birth)
- Change software without changing the data and vice versa
- Make applications **independent** of data storage and make display independent of data logic

How to Access a Database: Structured Query Language (SQL)

- The most commonly used database programming language for accessing and managing database in DBMS
  - SQL statements can be embedded in other programming languages (C/C++, Java, Python etc) to enable them to access database
- To create, manage and query a database.
- It is pronounced “ess cue ell”, not SEQUEL
  - SEQUEL is the ancestor of SQL introduced by IBM

SQL Examples

- Data manipulation: ad-hoc queries

```
SELECT        *
FROM  Account
WHERE         Type = "checking ";
```

- Data definition: creation of tables and views

```
CREATE TABLE Account
    (Number          integer  NOT NULL,
     Owner           character,
     Balance         currency,
     Type            character,
    PRIMARY KEY (Number));
```

- Control: assertions to protect data integrity

```
CHECK (Owner IS NOT NULL)
```

Data Definition Language (DDL)

- Language for defining the database schema
  - E.g. **create table** *Enrolled* (
    sid **char**(10), cid **char**(10), grade **char**(2))
- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
  - database schema
  - data *storage and definition* language: specify storage structure and access methods used by the DBMS

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - Procedural – user specifies what data is required and how to get those data
  - Nonprocedural – user specifies what data is required without specifying how to get those data
- SQL is the most widely-used query language

Example: SQL Embedded in Java (FYI)

```
class JDBCTest {
  public static void main (String args [ ])  throws Exception {
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        System.out.println ("Connecting...");
        Connection conn =
              DriverManager.getConnection ("jdbc:mysql://localhost/ML");
        System.out.println ("connected.");

        // Create a statement
        Statement stmt = conn.createStatement ();

        // List all the checking accounts
        ResultSet rset = stmt.executeQuery (“SEECT * FROM Account
                                              WHERE Type = “Checking”");

        … …
  }
}
```

Database Development

- Requirement analysis
  - What data need to be stored
  - What operations performed on the data
- Design
  - Conceptual: ER/EER or UML, DBMS independent
  - Logical: mapping to a logical data model, DBMS dependent
  - Physical: data structure, DBMS dependent
- Implementation
  - Language/DBMS dependent
  - C/C++, Java, SQL interact with DBMS to create table/object/query etc.

When NOT to Use Databases

- More desirable to use flat files for:
  - Small variety of data and/or small amount of data
  - Simple, well-defined applications with no expected changes at all
  - Only single (personal) access to data
  - Stringent, real-time requirements that cannot afford DBMS overhead
- Unlikely that any of these apply to corporate data management.
  - In fact, corporations often maintain many databases across many database systems.

Summary of This Week's Lectures

- Course logistics
- Motivation for databases
- Databases
  - Collection of related data (recorded facts)
  - Data independence
- DBMS
  - Generalized software package for implementing and maintaining a computerized database
  - Provides many services to manage data resources
- Limitations of DB and DBMS