Overview

ISYS2120 Data and Information Management

Overview

Prof Alan Fekete

University of Sydney

Acknowledge: slides from Uwe Roehm and Matloob Khushi, and from the materials associated with reference books (c) McGraw-Hill, Pearson

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of the University of Sydney pursuant to Part VB of the Copyright Act 1968 (**the Act**). The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

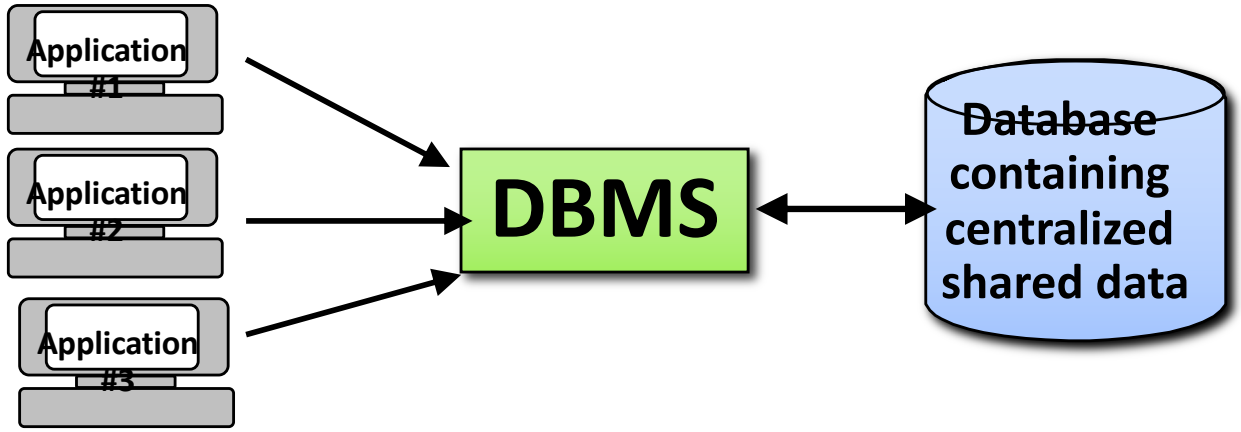
Do not remove this notice.

Outline

- Data and DBMS in context
- The relational data model
  - Terminology
  - Practice understanding relational data
- Careers and people's roles
- DBMS versus other ways to manage shared data
- Some big ideas

Terminology

- Data: Facts that can be recorded
  - Important for users
  - Need to persist (not just temporary values used during a computation)
- Database: A collection of data
  - Usually quite large
  - Usually containing all the data needed to operate an organization (or coherent part of an organization)
- Database Management System (DBMS): software package designed to store and manage one or more databases
  - Allows shared access from many programs



Categories of organizational data

- Personnel
- Inventory
- Orders
- Partners
- Transactions
- Tasks

Importance of data

- The day-to-day operations of an organization depend on accurate data
- The long-term planning in an organization depends on accurate data
- Example: a supermarket chain
  - What data is needed to pay workers?
    - "Carol Jones worked for 7 hours on Monday July 21, as Deli Manager"
  - What data is needed to pay suppliers?
  - What data is needed to decide when to offer a discount on tinned peas?
  - Think of some other activities…
  - Estimate the amount of data…
- Data is a major asset for the organization
  - Think of the losses if the data is missing or wrong
  - Think of the cost to replace the data

Importance of a database

- An asset in its own right
  - Historical data can guide enterprise strategy
  - Of interest to other enterprises
- State of database mirrors state of enterprise
  - Database is persistent
  - Shared:
    all qualified users have access to the same data for use in a variety of activities.

Managing Data

- The usual way to build an information system today, is to write many application programs which all use a shared collection of data
  - The data is stored in a DBMS
  - For core activities in large organizations, the DBMS is usually a commercial one (Oracle, MS SQLServer, IBM DB2, etc)
  - Smaller organizations, or non-mission-critical parts of large organizations, often use Open Source systems (PostgreSQL, MySQL, etc)
    - also, really huge data in large organizations eg Facebook's data on likes, friends, etc
  - Users run application programs which access the data through the DBMS
    - Programs can retrieve data and present it to the users
    - Programs can update the data based on input from the users

The Structure of Data

- A key idea in DBMSs is for the database itself to store descriptions of the format of the data
- This is called the "System Catalogue" or "Data Dictionary"
- Eg, you can find that each employee has
  - Identifier which is integer
  - Name which is a string of up to 30 characters
  - Address which is string of up to 60 characters
  - Salary which is integer
- This sort of information is often called *meta-data*

Relational DBMSs

- Most DBMSs today store data that follows a very simple "relational" structure
- All data is seen by users as tables of related, uniformly structured information
  - No duplicate rows, order is not important
  - Each entry is simple: integer, string, etc
  - Matching values in different tables indicate connections

Supplier:

SuppID	Name	Phone
8703	Heinz	0293514287
8731	Edgell	0378301294
8927	Kraft	0299412020
9031	CSR	0720977632

Product:

ProdID	Descr	SuppID
29012	Peas with Mint, 400g	8731
30086	Peas and Carrots, 450g	8703
31773	Salted Peanuts, 500g	8731

Instance

- An instance is the contents of the database at a single time
  - Specific values, which describe a specific situation in the world
  - Every update changes the instance

Schema

- The schema describes the structure of data in a particular database
  - What tables exist
  - What the columns are called, the type of each
- The instance at any time must fit the pattern of the schema
  - The schema changes rarely, if at all
- Eg **Supplier(SupplD: int, Name:string, Phone: string)**
- The schema can also include *integrity constraints*, which restrict the possible instances
  - Eg each supplier has a different SupplD

Data Design

- A key task when creating an information system is to decide on the schema that will be used for the data
  - Unless there is already a DBMS with the data needed
- The process of making this decision is called "data design"
- It proceeds in stages
  - first produce a conceptual or semantic model,
  - then translate this into a relational schema,
  - then evaluate the schema for quality, and improve it if needed
- We study this in detail in the first part of the semester.

Languages

- **DDL:** Data Definition Language is used to define the schema. It allows one to say tell the DBMS what tables exist, and what structure they have (including integrity constraints).
- **DML:** Data Manipulation Language is used to access the data. It includes commands to update (change) the contents of the database, and also it has commands that can retrieve information from the database ("queries").
- For a relational DBMS, both DML and DDL are in SQL
  - SQL is quite declarative in style (programmer says "what is needed" not "how to find it")
  - SQL is a standard, but each vendor has variations, so one doesn't have complete portability.
  - Applications are written to call the DBMS through SQL commands
- We study this in depth through much of the semester

SQL Example

- The *powerful* command

  SELECT – FROM – WHERE

retrieves data (rows) from one or more tables of a relational database that fulfill a search condition

- Example 1 “What is SuppID and Phone of the supplier named Heinz?”:

```
SELECT SuppID, Phone
  FROM Supplier
 WHERE Name='Heinz'
```

- Example 2 “How many products come from supplier whose SuppID is 8703?”:

```
SELECT COUNT(*)
  FROM Product
 WHERE SuppID = 8703
```

Join in SQL

- The same command

  SELECT – FROM – WHERE

Can connect data from several tables

- Example :

```
SELECT ProdID
   FROM Supplier, Product
  WHERE Product.SuppID = Supplier.SuppID
          AND Supplier.Name='Heinz'
```

- Computes: "Show the ProdID of any product which is supplied by Heinz"

Declarative Query

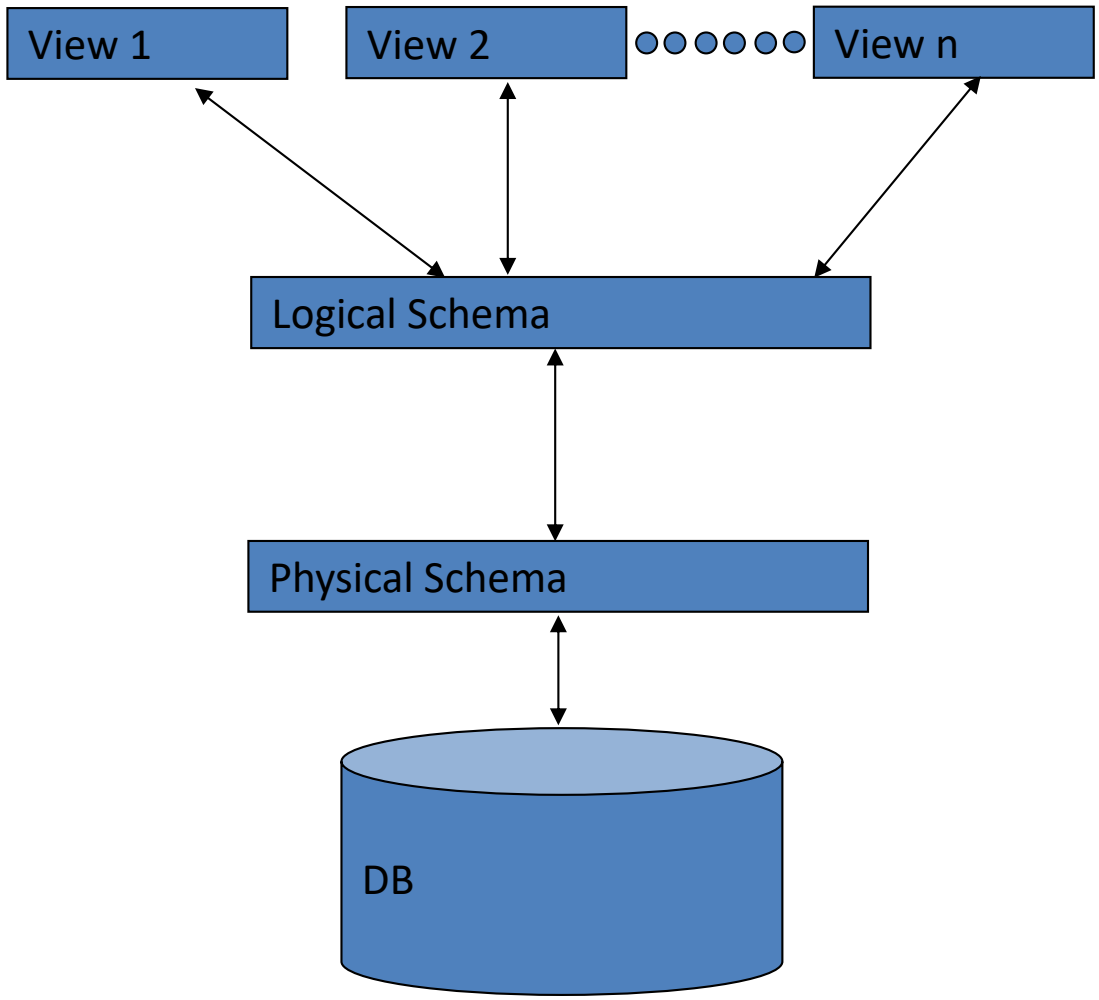
- Consider

```
SELECT ProdID
   FROM Supplier, Product
 WHERE Product.SuppID = Supplier.SuppID
         AND Supplier.Name='Heinz'
```

- Computes: "Show the ProdID of any product which is supplied by Heinz"
- How to compute this?
  - Look through Product table, row by row
    - For each product, take the SuppID in the row, and then find it in Supplier table, then check Name; if OK, output this ProdID
  - Or, Look through Supplier table, row by row
    - For each supplier, check Name; if OK, take the SuppID, then go through Product table checking each row to see if this SuppID is there
  - Or,… (other possibilities too)
- Given a query provided by user, DBMS should find a computation that is both correct and fast, and then do it that way to give user the answer!

Levels of Abstraction

- Many views, single logical schema and physical schema
  - A View describes how a user sees the data
  - Logical schema defines the structure of data as it is shared among all users
  - Physical schema describes the files and indexes used for storage on disk



Data Independence

- Applications are insulated from how data is structured and stored
- Logical data independence: Protection from changes in logical schema (eg from introducing an extra column in a table)
- Physical data independence: Protection from changes in physical structure and location of data
- Data independence is one of the most important benefits of using a DBMS

Roles with DBMS

- End users
- DB application programmers
- Database administration (DBA)
- DBMS Vendor Staff

- This course should help you appreciate the approach and concerns of each of these groups of people
  - So you can work in teams with them
  - Also, it can provide a foundation for career development in each of these directions

End Users

- End users are are people who do something that advances the organization's purpose
- They often are unaware that they are dealing with data in a DBMS
  - They simply run applications that present the data to them, and allow them to make (controlled) changes
- Categories
  - Off-line naïve users who get reports etc
    - Eg store manager gets weekly profit report
  - Parametric naïve users who execute pre-written applications
    - Eg deli manager who runs application to reorder some item
  - Ad hoc users who explore the data
    - Eg Division manager looking for trends

Application Programmers

- IT Professionals who produce the applications that end users can run
  - This usually fits into a broader software development process, with systems analysts, project managers, testers, etc
- Programmers need to understand how to create an application that accesses data through a DBMS
  - As well, they need a range of software engineering skills, including quality assurance, SDLC process, user interfaces, etc

DBA

- The Database Administrator is responsible to organization's management for effective and efficient use of resources in providing access to data
  - Does day-to-day operations, and makes important decisions
- Example tasks
  - Design logical/physical schemas
    - Make trade-offs between different choices, to get good performance for all users, at reasonable cost in hardware and software
  - Handle security and authorization
    - Set up accounts and permissions
  - Data availability, crash recovery
    - Make sure backups are taken, and used when needed
  - Database tuning
    - Monitor performance and adjust parameters or redesign schemas as needed

DBMS Vendor Staff

- Each vendor (Oracle, IBM, Microsoft etc) employ staff to help sell more installations, in particular by helping organizations that have bought the software
  - DBMS Implementors
  - Tools Development
  - Training Staff
  - Sales support staff
  - Operational support staff

A Database without a DBMS?

- Can one have a database without a DBMS to store it in?
- Yes; the information can be stored in files, which are accessed directly by all the programs that need to use the data
  - Files are good at keeping data for a long time
- This was common for business data in the 1950s, and is still common in science today
  - Eg geologist collects samples, and does analysis of chemical composition, and produces a file with the information
  - This file can be sent to other scholars as email attachment, or made available for download

File vs. DBMS

- Why are DBMSs now the industry-standard approach in business applications which need to deal with valuable, long-lasting shared data?

- Let's have an example:
  Comparison of an information system implemented on top of the file system (with C/Java/…) versus a realisation based on a DBMS
  - Scenario: Sales system

    Data: customers, orders, products, …
    Applications: orderings, payments & delivery, marketing,…

File vs. DBMS

Sharing metadata

File System:

```
TYPE customer  = RECORD
    cid  : INTEGER;
    name : ARRAY [1..20] OF CHAR;
    address: CHAR;
    END;
TYPE product  = RECORD
    pid   : INTEGER;
    title : ARRAY [1..30] of CHAR;
    weight: REAL;
    price : REAL;
    stock : INTEGER;
    END;
TYPE order = RECORD
    pid         : INTEGER;
    cid         : INTEGER;
    quantity    : INTEGER;
    order_date : DATE;
    END;
```

Data definitions repeated in each program…

Database System:

```
CREATE TABLE customers (
    cid INTEGER CHECK (cid>0),
    name VARCHAR(20),
    address VARCHAR(50)
   )
CREATE TABLE products (
    pid INTEGER CHECK (pid>0),
    title VARCHAR(30),
    weight FLOAT,
    price FLOAT,
    stock INTEGER CHECK (stock>=0)
    )
CREATE TABLE orders (
   pid INTEGER CHECK (pid>0),
   cid INTEGER CHECK (cid>0),
   quantity   INTEGER,
   order_date DATE,
   )
```

Data definitions once in the central data dictionary of a DBMS.
=> Same for all applications

File vs. DBMS

Programming for access to data

File System:

```
VAR o : order;
    p : product;
    fh1, fh2: file_handle;
fh1 = Open(order_file);
WHILE NOT EOF(fh1) DO
    o := getnext(fh1);
    IF o.month = 10 AND o.day >= 18
    THEN
       fh2 = Open(product_file);
       WHILE NOT eof(fh2) DO
         p := getnext(fh2);
         IF p.pid=o.pid AND
    p.stock<100 THEN
           WriteCard(o.pid,
    o.quantity);
        END;
       END;
       close(fh2):
    END;
END; close(fh1);
```

You have to program it by hand,
over and over and over …

Database System:

```
SELECT pnr, quantity
FROM   orders o, products p
WHERE  o.month = 10 AND o.day >= 18
  AND  o.pid = p.pid
  AND  p.stock<100
```

Declarative queries

- easy to read and maintain
- usable from different applications
- efficient evaluation due to automatic optimization

File vs. DBMS

Ensuring integrity

1. Product IDs and product titles should be unique.

File System:

Possible violation of data integrity due to incorrect input or missing synchronisation between different programs…

Database System:

```
CREATE TABLE products (
   pid INTEGER CHECK (pid>0)
       PRIMARY KEY,
   title VARCHAR(30) UNIQUE,
   …)
```

⇒ No integrity violation possible

2. There must be a corresponding product for each order.

File System:

Violation of integrity constraints possible, if a record is deleted from the file with the products without checking the orders first.

Database System:

```
CREATE TABLE orders (
   …,
   FOREIGN KEY(pid) REFERENCES
   products ON DELETE NO ACTION)
```

Data integrity ensured.

File vs. DBMS

Evolution of the system?

If you find you also need to know the phone number of each customer

- File System: rewrite every program that reads customer records

  You need to keep track of which programs use which files, and modify all those programs when the file structure changes

- DBMS: rewrite one table definition; probably don't need to change the queries that access customer, unless they need to use the phone number

  You only have to change the description in the database, not in the programs that use the data

File vs. DBMS

Access control

File System:

provides only a coarse granularity for access control:

- files
- on many systems, just three levels of control: owner / group / world

Database System:

declarative access control with regard to individual database users and user groups; views

```
CREATE VIEW ordersSYD AS
   SELECT * FROM orders
   WHERE cid IN ( SELECT
   cid FROM customers
   WHERE city = 'Sydney')
GRANT SELECT ON ordersSYD
   TO agentSmith
```

Comparison

- Drawbacks of using file systems to store data:
  - Data redundancy and inconsistency
    - Multiple file formats, duplication of information in different files
  - Difficulty in accessing data
    - Need to write a new program to carry out each new task
  - No central authority: different programmers might want different choices of files and formats, and there is no easy way to enforce organizational control over the valuable data
  - Integrity problems
    - Integrity constraints (e.g. account balance > 0) become part of program code
    - Hard to add new constraints or change existing ones

Comparison

- Drawbacks of using file systems (cont.)
  - Atomicity of updates: either all occur, or nothing occurs
    - Failures may leave database in an inconsistent state with partial updates carried out
    - E.g. transfer of funds from one account to another should either complete or not happen at all
  - Concurrent access by multiple users
    - Concurrent accessed needed for performance
    - Uncontrolled concurrent accesses can lead to inconsistencies
      - E.g. two people reading a balance and updating it at the same time
  - Security problems

➢ Database systems offer solutions to all the above problems

- ➢ But, this comes at a price in performance for simple processing
- ➢ And also in money
  - ➢ Licence fee for use for commercial platforms
  - ➢ Salaries for expert admins, always
- ➢ Finally, DBMSs are targeted at the most common sorts of information; they often work poorly on specialized data (eg gene sequences, images, etc)

Why organizations use a DBMS?

- Data independence and efficient access
- Reduced application development time
- Enhanced data integrity and security
- Hopes for extracting value by connecting data from different sources
- Uniform data administration
- Concurrent access, recovery from crashes

Relational Data Model

- The relational model is the most common in today's DBMSs
- It does very well for representing typical organizational data, such as employees and the names and salaries, products and their descriptions and quantity-in-stock, etc
  - Many facts of each kind, but each kind is simple (a few strings, integers etc)
- It does not do so well for richer information (images, designs, time series, instructions, etc)
  - There is a very active research community looking at other data models

Some big ideas

- There are some powerful ideas that we will see during the semester; they shape the way data management is done
  - These also apply (with variations) to non-relational data storage platforms
- Understanding these is a valuable outcome from this unit, in any career where one deals with important shared data
  - Even if one is working with data in files, try to adopt these ideas as much as possible

Meta-data as data

- The schema, and other meta-data, is essential to working with the data
  - Data is useless if one can't interprete it
  - Eg Alan Fekete,162,47,447
    - Is 162 weight in pounds, salary in $/hrs, height in cms, room number, or something else?
  - Over time, the people may leave who know what a value or string means
- Meta-data should be stored with the data it describes
  - And there should be some facilities for asking about and updating the meta-data

“What” not “how”

- It is convenient to indicate declaratively *what* information is needed, and leave it to the system to work out *how* to process through the data to extract what you need
  - Programming is hard, and choosing between different computations is hard
- Users should be offered a way to express their requests declaratively
  - A query language can be based on logic
  - Select…where…

Administrative services

- The performance that users get, and the security of the whole system, can be greatly affected by the way the data is stored, and the settings of system configurations
- What helps one user may be bad for others
  - But some changes have wide benefits
- The needs of the whole organization can be considered by people who are responsible to the whole, rather than to individuals or subsets
- A system should offer an interface for administrators to adjust parameters, structures etc
  - And one for users to provide hints that can help the administrators

Summary

- How information systems are built today: a DBMS is used to maintain and query large datasets, that are shared among many application programs
  - Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security
  - Data independence
- Overview of the relational model
  - Most important ideas and terminology
- The main roles of professionals who need to know about DBMS
- Some big ideas
  - Store the schema
  - Declarative queries
  - System Administration

References

- UW(3ed) Ch 1.1, 2.1, 2.2
  - Missing: comparison with file-based info system, roles of workers
- KBL(2ed) Ch 1.1-1.3, 2.1, 2.2, 3.1, 3.2
  - Missing: comparison with file-based info system
- RG(3ed) Ch 1.1-1.6, 1.9, 3.1
- SKS(6ed) Ch 1.1-1.7, 1.12