Overview of Data Management

Grant Weddell

Cheriton School of Computer Science
University of Waterloo

CS 348
Introduction to Database Management
Winter 2017
Course Logistics

Webpage
- www.cs.uwaterloo.ca/~gweddell/cs348

Text Book
Course Content

Why do we use databases?

- Functionality provided by a Database Management System
- Database Models: Relational, Network, OO
Course Content

Why do we use databases?
- Functionality provided by a Database Management System
- Database Models: Relational, Network, OO

How do we use a DBMS?
- Relational model, foundational query languages
- SQL
- Application programming
- Transactions and concurrency

How do we design a database?
- Entity-Relationship (ER) modeling
- Dependencies and constraints
- Redundancy and normal forms

How do we administer a DBMS?
- Security and authorization
- Physical design/tuning
Course Content

Why do we use databases?
- Functionality provided by a Database Management System
- Database Models: Relational, Network, OO

How do we use a DBMS?
- Relational model, foundational query languages
- SQL
- Application programming
- Transactions and concurrency

How do we design a database?
- Entity-Relationship (ER) modeling
- Dependencies and constraints
- Redundancy and normal forms
Course Content

Why do we use databases?
- Functionality provided by a Database Management System
- Database Models: Relational, Network, OO

How do we use a DBMS?
- Relational model, foundational query languages
- SQL
- Application programming
- Transactions and concurrency

How do we design a database?
- Entity-Relationship (ER) modeling
- Dependencies and constraints
- Redundancy and normal forms

How do we administer a DBMS?
- Security and authorization
- Physical design/tuning
What is a Database?

Definition (Database)
A large and persistent collection of information organized in a way that facilitates efficient retrieval and modification.

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.
What is a Database?

Definition (Database)

A *large* and *persistent* collection of information organized in a way that facilitates efficient *retrieval* and *modification*.
What is a Database?

Definition (Database)

A *large* and *persistent* collection of information organized in a way that facilitates efficient *retrieval* and *modification*.

Examples:

- a file cabinet
- a library system
- a personnel management system
What is a Database?

**Definition (Database)**

A large and persistent collection of information organized in a way that facilitates efficient retrieval and modification.

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.
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
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
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)

Note

The data maintained by the system are much more important and valuable than the system itself.
Brief History of Data Management: Ancient

2000 BC: Sumerian Records

296 BC: Library of Alexandria

1884: U.S. Census (Hollerith)
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)
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
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
• 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
• 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
Database Management System

Idea

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

Overview of Data Management
Winter 2017
Brief History of Data Management: 1970s

- Edgar Codd proposes relational data model (1970)
  - firm mathematical foundation → declarative queries
- Charles Bachman wins ACM Turing award (1973)
  - “The Programmer as Navigator”
- Peter Chen proposes E-R model (1976)
- Transaction concepts (Jim Gray and others)
- IBM’s System R and UC Berkeley’s Ingres systems demonstrate feasibility of relational DBMS (late 1970s)
Definition (Schema)

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

Definition (Instance)

A database instance is a database (real data) that conforms to a given schema.
Three-level Schema Architecture (cont.)

- **Users/Applications**
  - External Schema
  - Conceptual Schema
  - Internal Schema

- **DBMS**
  - External view
  - Conceptual view
  - Internal view

- **Database**
Data Independence

Idea

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

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

Example:

\[
\begin{align*}
\text{withdraw}(AC,1000) & \quad \text{withdraw}(AC,500) \\
\text{Bal} & := \text{getbal}(AC) & \text{Bal} & := \text{getbal}(AC) \\
\text{if } (\text{Bal}>1000) & \quad \text{if } (\text{Bal}>500) \\
\text{<give-money>} & \quad \text{<give-money>} \\
\text{setbal}(AC,\text{Bal}-1000) & \quad \text{setbal}(AC,\text{Bal}-500)
\end{align*}
\]
Transactions

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

Example:

\[
\begin{align*}
&\text{withdraw(AC,1000)} & \text{withdraw(AC,500)} \\
&\quad \text{Bal := getbal(AC)} & \quad \text{Bal := getbal(AC)} \\
&\quad \text{if (Bal>1000) if (Bal>500)} \\
&\quad \quad \quad <\text{give-money}> & \quad \quad \quad <\text{give-money}> \\
&\quad \quad \quad \text{setbal(AC,Bal-1000)} & \quad \quad \quad \text{setbal(AC,Bal-500)}
\end{align*}
\]

Idea

Every application may think it is the sole application accessing the data. The DBMS should guarantee correct execution.
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
Brief History of Data Management: 1980s

- Development of commercial relational technology
  - IBM DB2, Oracle, Informix, Sybase
- Edgar Codd wins ACM Turing award (1981)
- SQL standardization efforts through ANSI and ISO
- Object-oriented DBMSs
  - persistent objects
  - object id’s, methods, inheritance
  - navigational interface reminiscent of hierarchical model
Brief History of Data Management: 1990s-Present

- Continued expansion of SQL and system capabilities
- New application areas:
  - the Internet
  - On-Line Analytic Processing (OLAP)
  - data warehousing
  - embedded systems
  - multimedia
  - XML
  - data streams
- Jim Gray wins ACM Turing award (1998)
- Relational DBMSs incorporate objects (late 1990s)
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