CMPT 354

Introduction

Course Website

- http://www.cs.sfu.ca/CourseCentral/354/johnwill/

Assessment

- Assignments – 30%
- Midterm exam in class – 20%
- Final exam – 50%

Data and Databases

What is a Database?

- A database is a collection of information
  - Databases of one sort or another have existed since the dawn of civilization
- In Computing Science a database is a collection of data
  - That is managed by a Database Management System (DBMS)
  - DBMS commonly use a relational model to represent data

History of Databases

- http://www.computerhistory.org/revolution/memory-storage/8/265/2207

Data, Data, Everwhere

- Early databases were primarily used by large organizations to store textual data
  - In 1975 there were some 301 databases containing about 52 million records,
  - By 1998 there were 11,339 databases holding nearly 12.05 billion records!
    - Martha E. Williams (1998), "State of Databases Today: 1999," in *Gale Directory of Databases*, L. Kumar, ed.
- Databases are now used to store many different types of data
  - Images, sounds, …

Data in the Current Millenium

- The amount of data stored has increased dramatically
  - As has the variety of types of data
- Consider statistics for Walmart in 2012
  - Handled more than 1 million customer transactions per hour
  - Imported into databases estimated to contain more than 2.5 petabytes of data

Kilobyte	– 2^{10} bytes
Megabyte	– 2^{20} bytes
Gigabyte	– 2^{30} bytes
Terabyte	– 2^{40} bytes
Petabyte	– 2^{50} bytes
Exabyte	– 2^{60} bytes
Zettabyte	– 2^{70} bytes

What's a Zettabyte?

- A zettabyte is
  - Often spelt zetabyte
  - 2^{70} bytes
    - 1,180,591,620,717,411,303,424 bytes
- That's a big number
  - There are estimated to be in the order of 100 billion stars in the Milky Way (our) galaxy
    - 100,000,000,000 = 0.0000000000847 zettabytes
  - Estimates of the number of stars in the observable universe vary wildly, here is one
    - 10,000,000,000,000,000,000,000,000 = 847 zettabytes

Big Data

- 2012 statistics*
  - 2.7 zettabytes of data in the world
  - 235 terabytes of data were collected by the Library of Congress in 2011
  - Facebook stores, accesses and analyzes over 30 petabytes of user generated data
  - The largest AT&T database is 312 terabytes containing 1.9 trillion rows
  - Decoding the human genome originally took 10 years to process – it could now be achieved in a week
- *Wikibon Blog

Database Applications

- Any application that has to store large amounts of data probably needs a database
  - Financial industry
  - Government agencies
  - Airlines
  - Universities
  - Utilities
  - High street retailers
  - On-line retailers
  - Manufacturing
  - Human resources
  - Social media sites
  - MMORPGs
  - …

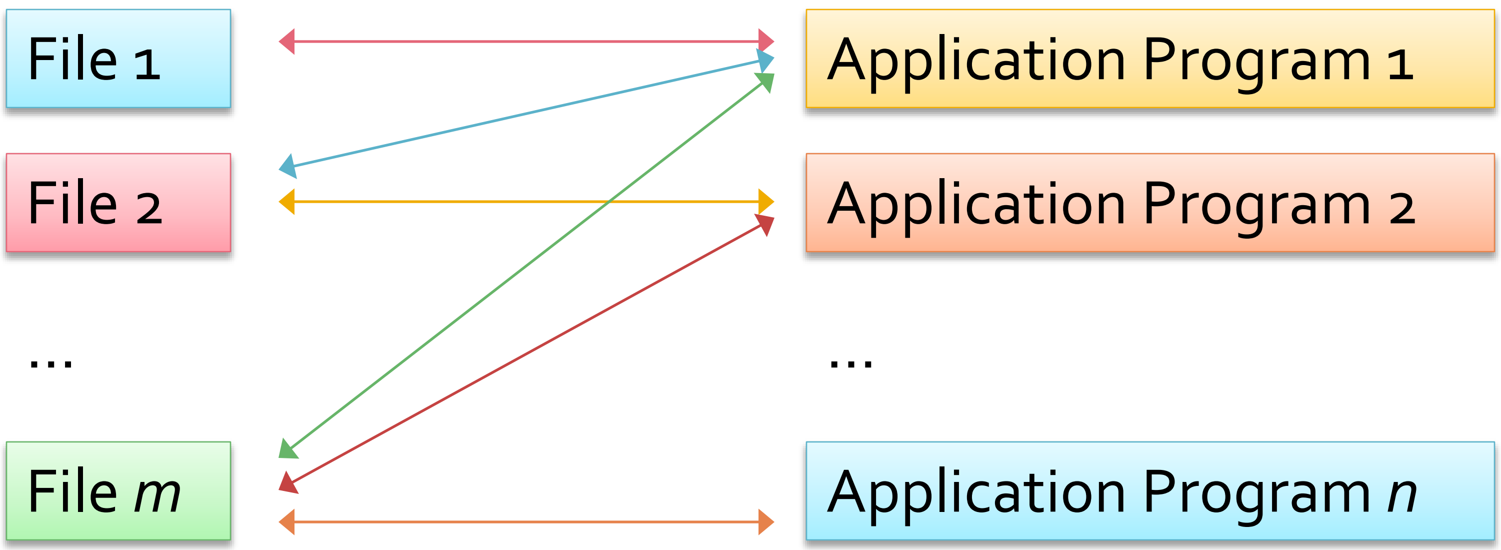
What is a Database System?

- A database system consists of two components
  - Database (DB) and
  - Database Management System (DBMS)
- The DB contains the data
- The DBMS is software that stores, manages and retrieves the information in the DB

Why Use a Database?

Data Storage Without DBMS

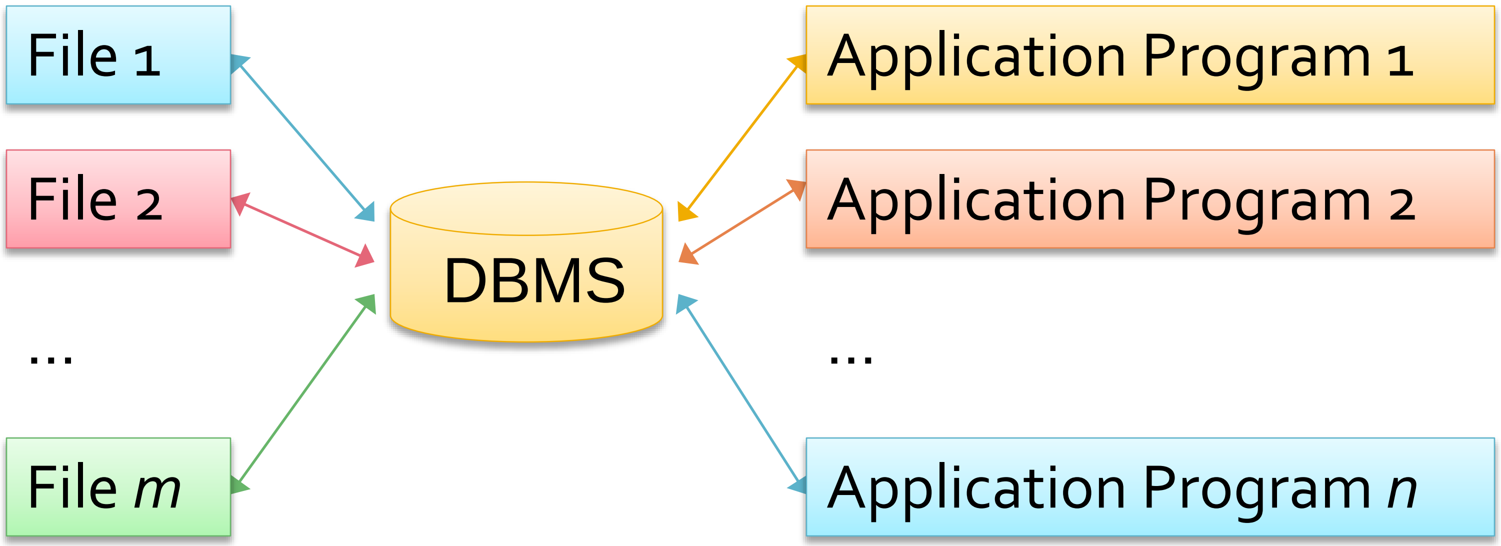
- Data would be collected in many different files and
- Used by many application programs



What Happens If …

- An attribute is added to one of the files?
- Information that is in more than one file is changed by a program that only interacts with one file?
- We need to repeatedly access a single record out of millions of records?
- We need to retrieve data stored in multiple files?
- Several programs need to access *and modify* the same record at the same time?
- The system crashes while one of the application programs is running?

Data Storage with a DBMS



DBMS Functions

- All access to data is centralized and managed by the DBMS
- Design and implementation advantages
  - Logical data independence
  - Physical data independence
  - Reduced application development time
- Use advantages
  - Efficient access
  - Data integrity and security
  - Concurrent access and concurrency control
  - Crash recovery

Data Abstraction

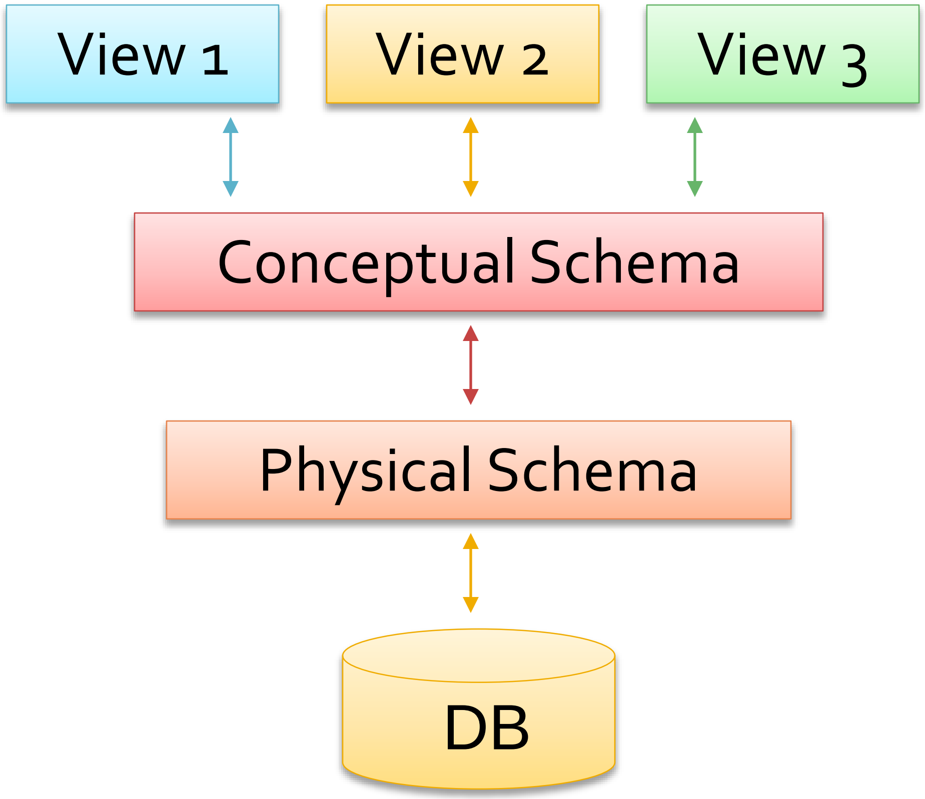
Data Models - Definitions

- A database models a real-world enterprise
- A *data model* is a formal language for describing data
  - A *schema* is a description of a particular collection of data using a particular data model
- The most widely used data model is the *relational data model*
  - The main concept of this model is a *relation*, or set
    - Which can be represented by a table with rows and columns

Data Abstraction

- Data can be described at three levels of abstraction
- Physical schema
  - The lowest level schema
  - Describes how data are stored and indexed
- Conceptual (or logical) schema
  - What (not how) data are stored
  - Describes data in terms of the data model
- External (or view) schema
  - The highest level schema
  - Describes how some users access the data
  - There can be many different views

Levels of Abstraction



Data Independence

- The levels of a DB allow a schema at one level to be modified without affecting the others
- Allows application programs to be relatively independent from the data
  - They do not need to be modified as a result of changes to the database structure or storage
- Resulting in reduced application development and maintenance time

Types of Data Independence

- Physical data independence
  - Allows the physical schema to be modified without rewriting application programs
  - Usually to improve performance
    - e.g. adding or removing an index or moving a file to a different disk
- Logical data independence
  - Shields users from changes in the logical schema – i.e. their views remain unchanged
  - Allows the logical schema to be modified without rewriting application programs
    - e.g. adding an attribute to a relation

Views

- One major purpose of a database is to allow users to view data
  - Without requiring knowledge of how the data are stored
- A single database can support many different views of the same data for different users
  - Example – Student Information System

Transactions

Database Advantages

- There are a number of advantages of using a DBMS to record data
- Records can be efficiently accessed
- Transactions are processed so that the integrity of the database is maintained
  - Even if multiple users access the database concurrently or
  - The system crashes during processing

Efficient Access

- What happens when a user wants to find one record out of millions?
  - An index structure maps the desired attribute values to the address of the record
  - The desired records can be retrieved without scanning the whole relation
  - This makes query processing efficient

Introduction

- A transaction is a single execution of a user program in a DBMS
  - e.g. Transferring money between two bank accounts
    - If money is transferred between two bank accounts, *both* accounts' records must be updated
- A single *transaction* may result in *multiple actions*
  - For performance reasons, these actions may be interleaved with actions of another transaction
- Transactions should have the **ACID** properties

ACID Properties

- Transactions should be **atomic**
  - Either all or none of a transaction's actions are carried out
- A transaction must preserve DB **consistency**
  - The responsibility of the DB designers and the users
- A transaction should make sense in **isolation**
  - Transactions should stand on their own and
  - Should be protected from the effects of other concurrently executing transactions
- Transactions should be **durable**
  - Once a transaction is successfully completed, its effects should persist, even in the event of a system crash

Consistency

- A DB is in a consistent state if it satisfies all of the constraints of the DB schema
  - For example key constraints, foreign key constraints
- A DBMS typically cannot detect all inconsistencies
  - Inconsistencies often relate to domain specific information
  - And may be represented by triggers, or policies
- Users are responsible for transaction consistency
  - This includes programs that access the DB
    - Casual users should only interact with a DB via programs that are consistency preserving
    - Expert users often interact directly with a DB

Examples

- Entering a new Patient record with a duplicate MSP
  - Allowing this transaction would leave the DB in an inconsistent state so the DB rejects the transaction
- Customers are only allowed if they have an account, accounts are only allowed if they have a customer
  - Entering a new customer who owns a new account is impossible if the DB attempts to maintain this constraint
  - The constraint can be modeled by an application program
- Employees are not allowed to be customers, but SIN is not recorded for customers
  - The constraint must be maintained by policy

Isolation

- Multiple transactions may be interleaved
  - That is, processed concurrently
  - The net effect of processing the transactions should be the same as executing them in *some* serial order
    - There is no guarantee *which* serial order is chosen
- Consistency and isolation
  - If transactions leave the DB in a consistent state, and
  - If transactions are executed serially then,
  - The resulting DB should be in a consistent state

Atomicity

- A transaction that is interrupted may leave the DB in an inconsistent state
  - Transactions consist of multiple actions, and are consistent only when considered in their entirety
  - Transactions must therefore be atomic
    - All or Nothing!
- A DB will remain consistent if transactions are
  - Consistent
  - Processed as if they occur in some serial order
  - Atomic, ensuring that no partial transactions occur

Atomicity and Consistency

- Consider transferring money between accounts
  - The database will only remain consistent if both sides of the transfer are processed
  - If the transfer is interrupted there is a risk that one half is processed and the other is not
- Note that this principle applies to almost any accounting entry
  - Most accounting systems use *double-entry bookkeeping*
- Many other DB transactions are composed of multiple actions

Durability

- Transactions are first processed in main memory
  - And later written to disk
    - For example when the affected memory pages are replaced
  - If the system crashes in between these two processes there is a possibility that the transaction is lost
- A DBMS maintains a *log* that records information about all writes to the database
  - In the event of a system crash the log is used to restore transactions that were not written to disk
- To achieve durability it is also necessary to maintain backups of the database

Database Languages

Database Languages

- A database language is divided into two parts
  - Data definition language (DDL)
  - Data manipulation language (DML)
- Structured query language (SQL) is both a DDL and a DML
  - Most traditional commercial databases use SQL and we will cover it in detail in this course

Data Definition Language

- The DDL allows entire databases to be created, and allows integrity constraints to be specified
  - Domain constraints
  - Referential integrity
  - Assertions
  - Authorization
- The DDL is also used to modify existing DB schema
  - Addition of new tables
  - Deletion of tables
  - Addition of attributes

Data Manipulation Language

- The DML allows users to access or change data in a database
  - Retrieve information stored in the database
  - Insert new information into database
  - Delete information from the database
  - Modify information stored in the database
- There are two basic types of DMLs
  - *Procedural* – users specify what data is required and how it should be retrieved
  - *Declarative* (nonprocedural) – users specify what data is required without specifying how it should be retrieved

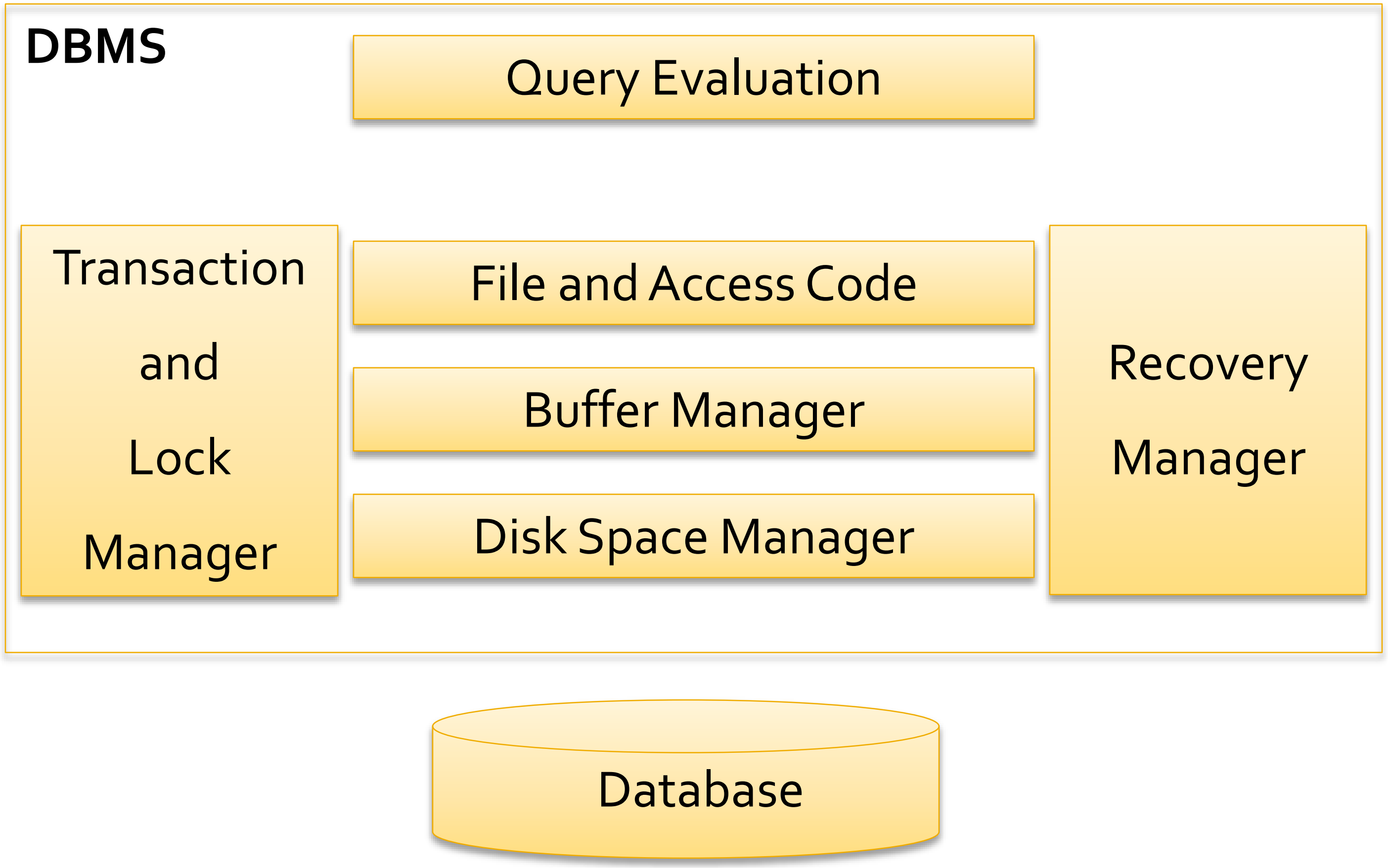
Database Users

Database Users

- End users
  - May have specialized knowledge (CAD etc.) and may be familiar with SQL
  - The majority have no DB knowledge
- DB Administrators
  - Have central control over data and programs that access that data
- Database Application Programmers
  - Write programs that need to interact with the DB
- DB Implementers and Vendors
  - Build and sell DB products

Database Components

Typical DBMS Structure



Database Components

- Diskspace (storage) manager – responsible for interaction with the OS file system
  - Allows other levels of the DBMS to consider the data as a collection of pages
- Buffer manager – responsible for bringing pages into main memory from disk
  - Including the management of a replacement policy when main memory is full
- File and access code allows the query evaluation system to request data from lower levels

Database Components

- Query evaluation – most modern DBMSs will optimize queries
  - There are often multiple equivalent queries
  - The query optimizer determines an efficient execution plan for a query
- Transaction lock manager – responsible for allowing concurrent access
  - While maintaining data integrity
- Recovery manager – responsible for maintaining a log and restoring the system after a crash

Topics

CMPT 354 and 454

CMPT 354 and 454 Topics

- CMPT 354 – DB specification and implementation
  - Database design – the relational model and the ER model
  - Creating and accessing a database
    - Relational algebra
    - Creating and querying a DB using SQL
  - Database application development
- CMPT 454 – DBMS Issues
  - Disk and buffer management and storage
  - Query evaluation
  - Transactions and recovery
  - Advanced topics

CMPT 354 Topics

- Designing a DB using the Entity Relationship model
  - Entity Relationship diagrams
- The relational model, converting an ERD into an SQL database
- Relational algebra, the basis of SQL
- SQL
- Specifying constraints on a database
- Database applications
- Normalization
- Other topics