Introduction
Introduction

- Administration
- Naive DBMS
- CMPT 454 Topics
Administration
Course Website

Assessment

- Assignments – 25%
- Midterm exam in class – 20%
- Final exam – 55%
Simple DBMS
A database stores the data for some organization
- The focus of a conventional database is on recording and querying individual data items
- The data for a very simple system might consist of a single file
  - Which presumably would not require a database

So why use a database?
- Larger organizations require many disparate types of records to be stored
  - Often used by different parts of the organization
  - Consider what information SFU needs to record
- These records need to be accessed by many users at the same time
Let's imagine a naïve implementation of a Database Management System (DBMS)

- The query processor is responsible for accessing data and determining the result of a query

* on hard disk

** in main memory
One file for each table
- Records are separated by newline characters
- Fields in records separated by some special character
  - e.g. the Customer file might store
    - Kent#123#journalist
    - Banner#322#unemployed

Store the database schema in a file
- e.g. the Customer and Account schema
  - Customer#name#STR#id#INT#job#STR
  - Account#acc_id#INT#id#INT#balance#FLOAT
Processing Simple Queries

- Read the file schema to find Customer attributes
  - Check the condition is semantically valid for customers
- Create a new file (T) for the query results
- Read the Customer file, and for each line (i.e. record)
  - Check condition, c
  - If c is true write the line to T
- Add a line for T to the file schema

```
SELECT * 
FROM Customer 
WHERE job = 'journalist'
```
Handling Joins

SELECT balance
FROM Customer C, Account A
WHERE C.name = 'Jones' AND C.id = A.id

- Simple join* algorithm:
  FOR each record c in Customer
    FOR each record a in Account
      IF c and a satisfy the WHERE condition THEN
        add the balance field from Account to result

Note that the query does not specify a join but a Cartesian product followed by a complex selection – so we assume some level of query optimization
Some Problems

- Searching for a subset of records entails reading the entire file
  - There is no efficient method of just retrieving customers who are journalists
  - Or of retrieving individual customers
- There is no efficient way to compute complex queries
  - The join algorithm is relatively expensive
  - Note that every customer is matched to every account regardless of the customer name
  - What is its $O$ notation running time?
More Problems

- Changing a single record entails reading the entire file and writing it back
- What happens when two people want to change account balances at the same time?
  - Either something bad happens
  - Or we prevent one person from making changes
- What happens if the system crashes as changes are being made to data?
  - The data is lost or possibly something worse
CMPT 454

- CMPT 354 – database design, creation, and use
  - ER model and relational model
  - Relational algebra and SQL
  - Implementation of database applications

- CMPT 454 – database management system design
  - How is data stored and accessed?
  - How are SQL queries processed?
  - What is a transaction, and how do multiple users use the same database?
  - What happens if there is a system failure?
How are data stored and accessed?
Data Storage

- Processing occurs in main memory
  - Data is stored in secondary storage and has to be retrieved to be processed
- Reading or writing data in secondary storage is much slower than accessing main memory
  - Typically, the cost metric for DB operations is based around disk access time
Secondary Storage

- Mechanics of disks
  - Access characteristics
  - Organizing related data on disk
  - Algorithms for disk access
- Disk failures
- Improving access and reliability
  - RAID
- Solid State Drives
Files and Records

- Arranging records on a disk
  - Fixed length records
  - Variable length records
- Representing addresses and pointers
- BLOBs
Indexing

- An index is a structure that speeds up access to records based on search criteria
  - For example, finding student data efficiently using student ID
- There are different index structures, with their own strengths and weaknesses
  - B trees
  - Hash tables
  - Specialized index techniques
How are SQL queries processed?
Processing an SQL query requires methods for satisfying SQL operators
- Selections
- Projections
- Joins
- Set operations
- Aggregations
- ...

There is more than one algorithm for each of these operations
SQL is a procedural query language
  - That specifies the operations to be performed to satisfy a query

Most queries have equivalent queries
  - That use different operations, or order of operations, but that return the same result

Query optimization is the process of finding the best equivalent query
  - Or at least one that is good enough
Once equivalent queries are derived they have to be evaluated

- What is the cost metric?
- What is the size of intermediate relations?
  - The result of each operation is a relation
- For multiple relation queries, how does the choice of join order affect the cost of a query?
- How are the results of one operation passed to the next?
What is a transaction, and how do multiple users use the same database?
Transactions

- A transaction is a single logical unit of work
  - Who is the owner of the largest account?
  - Which students have a GPA less than 2.0?
  - Transfer $200 from Bob to Kate
  - Add 5% interest to all accounts
  - Enroll student 123451234 in CMPT 454
  - ...

- Many transactions entail multiple actions
Transferring $200 from one bank account to another is a single transaction

- With multiple actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Bob</th>
<th>Kate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Bob's balance</td>
<td>347</td>
<td></td>
</tr>
<tr>
<td>Read Kate's balance</td>
<td></td>
<td>191</td>
</tr>
<tr>
<td>Subtract $200 from Bob's balance (147)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add $200 to Kate's balance (391)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write Bob's new balance</td>
<td>147</td>
<td></td>
</tr>
<tr>
<td>Write Kate's new balance</td>
<td></td>
<td>391</td>
</tr>
</tbody>
</table>
Concurrency

- A typical OLTP\textsuperscript{1} database is expected to be accessed by multiple users concurrently
  - Consider the Student Information System
- Concurrency increases throughput\textsuperscript{2}
  - Actions of different transactions may be *interleaved* rather than processing each transaction in series
- Interleaving transactions may leave the database in an inconsistent state
- \textsuperscript{1} – Online Transaction Processing
- \textsuperscript{2} – Throughput is a measure of the number of transactions processed over time
...

... without crying please ...

John Edgar
Concurrency Error Example

- **T1** – Transfer $200 from Bob to Kate
- **T2** – Deposit $7,231 in Bob's Account

<table>
<thead>
<tr>
<th>Action</th>
<th>Bob</th>
<th>Kate</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 – Read Bob's balance</td>
<td>347</td>
<td></td>
</tr>
<tr>
<td>T2 – Read Bob's balance</td>
<td>347</td>
<td></td>
</tr>
<tr>
<td>T1 – Read Kate's balance</td>
<td></td>
<td>191</td>
</tr>
<tr>
<td>T2 – Add $7,231 to Bob's balance (7,578)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1 – Subtract $200 from Bob's balance (147)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2 – Write Bob's new balance</td>
<td>7,578</td>
<td></td>
</tr>
<tr>
<td>T1 – Add $200 to Kate's balance (391)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1 – Write Bob's new balance</td>
<td></td>
<td>147</td>
</tr>
<tr>
<td>T1 – Write Kate's new balance</td>
<td></td>
<td>391</td>
</tr>
</tbody>
</table>

Bob is probably not happy

This transaction schedule should be prevented

John Edgar
ACID Transactions

- Transactions should maintain the ACID properties
  - Atomic
  - Consistent
  - Isolated
  - Durable
Concurreny Topics

- Serial and serializable schedules
- Conflict serializability
- Locking
  - Two phase locking
  - Locking scheduler
  - Lock modes
  - Architecture
- Optimistic concurrency control
- Deadlocks
What if there is a system failure?
System Failures

- Processing is performed in main memory
  - But database objects are only persistent when written to long term storage
- Once a transaction has completed it should be persistent
  - If the system crashes after completion but before changes are written to disk those changes are lost
  - Recovery is the process of returning a database to a consistent state after a system crash
Recovery Topics

- Transactions
- Undo logging
- Redo logging
- Undo/Redo logging
- Media failures
Other Topics
Massive datasets are currently maintained across the web
  - Some of these are stored in relational databases
  - And others are not
    - There are many NoSQL data stores

What are the issues in maintaining a distributed database?
Distributed Databases Topics

- Distributed vs. parallel databases
- Horizontal and vertical fragmentation
- Distributed query processing
- Distributed transactions
- Cloud databases