So you’ve taken DB systems?

- Then you know what this is:

- But, that’s not a system

- This is:
Course Goals

- Learn about database systems
  - How to build them, not how to use them
- This class is built around transactions (historically)
  - How a database operates at the storage/memory level
- DB Systems are broader than transactions
  - Physical design, indexing, replication
- Transactions are bigger than DB systems
  - TP monitors, micro-transactions, parallel computing
- Plus my interests have grown
  - Spatial data, query optimization, selectivity estimation
- We’ll cover whatever I think is important/fun/cool
  - But, I won’t/can’t rename the course. A better title would be
    - DBs (and other systems) that Use Transactions Sometimes
Topics

  - Actually, just page model transaction processing, not object model
  - We will go light on some of the more theoretical material

- You will understand ARIES
  - This is on the path to enlightenment

- Hot Database Systems topics
  - E.g., databases for multi-core architectures, column stores

- Database technology spillover
  - E.g., micro-transactions in operating systems, declarative networking, declarative interfaces to parallel programming, pseudo-DBs
Your Job

- Come to class (only if you want)
  - It’s fun and educational!
  - You will be responsible for all in-class material
- 4 or 5 problem sets
- 1 programming project
  - I’m getting rid of ARIES, 5 times is enough
  - We’ll make it up as we go along
  - I’m thinking that we should implement a column store
- 2 midterms (one masquerading as a final exam)
Transactions

- *What is a transaction?*
Transactions

- **What is a transaction?**
  - A computing abstraction!
  - A programming model
  - Typically defined by ACID semantics, but this process can be misleading

- **What’s the abstraction then?**
  - Collect many dependent or independent together and execute as one
  - TxN properties: available, indivisible, recoverable
  - Consider government bills with riders by way of analogy

- **What are transactions good for?**
  - Model real-world interactions (barter/exchange)
  - Create dependencies among multiple operations
  - Encode independent computations for parallel execution
A Canonical Example

- OLTP in a single database

```c
void main () {
    EXEC SQL BEGIN DECLARE SECTION
    int b /*balance*/, a /*accountid*/, amount;
    EXEC SQL END DECLARE SECTION;
    /* read user input */
    scanf ("%d %d", &a, &amount);
    /* read account balance */
    EXEC SQL Select Balance into :b From Account
    Where Account_Id = :a;
    /* add amount (positive for debit, negative for credit) */
    b = b + amount;
    /* write account balance back into database */
    EXEC SQL Update Account
    Set Balance = :b Where Account_Id = :a;
}
```
## Concurrent Executions

<table>
<thead>
<tr>
<th>P1</th>
<th>Time</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Select Balance Into :b&lt;sub&gt;1&lt;/sub&gt;</strong>&lt;br&gt;From Account&lt;br&gt;Where Account_Id = :a</td>
<td>1</td>
<td><strong>Select Balance Into :b&lt;sub&gt;2&lt;/sub&gt;</strong>&lt;br&gt;From Account&lt;br&gt;Where Account_Id = :a</td>
</tr>
<tr>
<td>/* b&lt;sub&gt;1&lt;/sub&gt;=0, a.Balance=100, b&lt;sub&gt;2&lt;/sub&gt;=0 */</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>b&lt;sub&gt;1&lt;/sub&gt; = b&lt;sub&gt;1&lt;/sub&gt;-50</strong>&lt;br&gt; /* b&lt;sub&gt;1&lt;/sub&gt;=100, a.Balance=100, b&lt;sub&gt;2&lt;/sub&gt;=100 */</td>
<td>2</td>
<td><strong>b&lt;sub&gt;2&lt;/sub&gt; = b&lt;sub&gt;2&lt;/sub&gt; +100</strong></td>
</tr>
<tr>
<td>/* b&lt;sub&gt;1&lt;/sub&gt;=50, a.Balance=100, b&lt;sub&gt;2&lt;/sub&gt;=100 */</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Update Account</strong>&lt;br&gt;S&amp;t Balance = :b&lt;sub&gt;1&lt;/sub&gt;&lt;br&gt;Where Account_Id = :a</td>
<td>3</td>
<td><strong>Update Account</strong>&lt;br&gt;S&amp;t Balance = :b&lt;sub&gt;2&lt;/sub&gt;&lt;br&gt;Where Account_Id = :a</td>
</tr>
<tr>
<td>/* b&lt;sub&gt;1&lt;/sub&gt;=50, a.Balance=100, b&lt;sub&gt;2&lt;/sub&gt;=200 */</td>
<td>4</td>
<td>/* b&lt;sub&gt;1&lt;/sub&gt;=50, a.Balance=200, b&lt;sub&gt;2&lt;/sub&gt;=200 */</td>
</tr>
<tr>
<td><strong>b&lt;sub&gt;2&lt;/sub&gt; = b&lt;sub&gt;2&lt;/sub&gt; +100</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/* b&lt;sub&gt;1&lt;/sub&gt;=50, a.Balance=200, b&lt;sub&gt;2&lt;/sub&gt;=200 */</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
Think Big!

- Transactional guarantees need to be bigger than a single database
- Consider buying a song through iTunes
  - Multiple parties involved (you, credit card company, iTunes)
  - Transaction demands
    - I want the song = availability
    - I won’t pay unless I get it = indivisibility
    - If my computer fails during download, I need to re-download = recoverability
- *Is this example actually covered by transactional semantics?*
E-Commerce Example

Shopping at Internet book store:
- client connects to the book store's server and starts browsing and querying the store's catalog
- client fills electronic shopping cart
- upon check-out client makes decision on items to purchase
- client provides information for definitive order (including credit card or cyber cash info)
- merchant's server forwards payment info to customer's bank credit or card company or cyber cash clearinghouse
- when payment is accepted, shipping of ordered items is initiated by the merchant's server and client is notified

- distributed, heterog. system with transactional effects
  - Amazon's fulfillment has changed the world w.r.t. our expectations for e-commerce. They separate ordering and fulfillment in their systems.
Workflow Example

Workflows are (the computerized part of) business processes, consisting of a set of (automated or intellectual) activities with specified control and data flow between them (e.g., specified as a state chart or Petri net).

Conference travel planning:
- Select a conference, based on subject, program, time, and place.
  If no suitable conference is found, then the process is terminated.
- Check out the cost of the trip to this conference.
- Check out the registration fee for the conference.
- Compare total cost of attending the conference to allowed budget, and decide to attend only if the cost is within the budget.

Observations: activities spawn transactions on information servers, workflow state must be failure-resilient, long-lived workflows are not isolated.
Example: Travel Planning Workflow

- Select Conference
  - / Budget := 1000; Trials := 1;
  - [ConfFound]
    - / Cost := 0
  - [!ConfFound]
- Check Conf Fee
- Select Tutorials
- Compute Fee
- Check Airfare
- Check Hotel
- Check Travel Cost
- Check Cost
  - / Cost = Conf Fee + Travel Cost
  - [Cost ≤ Budget]
  - Go
  - [Cost > Budget & Trials ≥ 3]
  - No
- [Cost > Budget & Trials < 3] / Trials++
Some Observations

- Transactions are a powerful concept
  - They model complex, real-world interactions in computers
  - They provide a simple, fault-tolerant programming model that is easily parallelizable
  - E.g., if all file systems updates were transactional, we would not need file system checkers

- Complex systems often implement partial transaction systems and hybrid models
  - Transactions can be expensive over networks and/or with multiple independent parties
  - So, relaxing transaction semantics is as important as transactions
  - E.g., file systems use transactions for metadata updates, but not data
Transaction Processing Systems

All of the following:

- Databases: lower levels/physical layer
  - People often conflate DB=Transactions. This is just wrong.
  - DBs do much more and many applications don’t require TxNs.

- TP Monitor: transaction systems for business applications that either:
  - Manage their own data representation, e.g. SAP
  - Do not require a full relational model, e.g. some STOCK tickers

- Logging and recover in file systems
- Workflow systems, e.g. Lotus Notes
- Scientific workbenches, e.g. Emulab
Transactions in DBs

Clients

Database Server

- Language & Interface Layer
- Query Decomposition & Optimization Layer
- Query Execution Layer
- Access Layer
- Storage Layer

Requests

Request Execution Threads

Data Accesses

Database

Lecture 1: Introduction to Transaction Processing
Some More Observations

- TxNs are (mostly) independent of the data model
  - Do not depend on relational nature of data

- TxNs are becoming ubiquitous
  - The semantics of isolation are helpful in managing parallelism, multi-core architectures, etc.

- Operating systems are the most active area of research on transaction processing systems
  - Not DBs. There it’s solved.