

# Enterprise Applications

CMSC 838p

# Logistics

- Check with me if you aren't officially registered
  - need a mailing list
- homework 1 due Wednesday
  - in class; bring a print-out

# Grading

- Homeworks 20%
  - simply stuff to keep you on your toes
- Class participation 30%
- Projects 30%
- Exams 20%

# Class organization

- Small class
- I expect activate participation and help in developing course material
- SnipSnap page is a group effort
  - I expect you to provide content

# Infrastructure

- mysql database on verve
  - temporary measure
- 4 dual-processor x86 linux boxes
  - 2 have arrived, 2 more expected next week
  - just for our class
- Other resources available:
  - 24-processor, 24 gigabyte Sun SMP
  - various larger clusters
- Access to prerelease builds of Java 1.5

# Projects

- Creating and evaluating real projects
  - might bring some customers in
- Looking at standard examples and benchmarks
  - Pet Store
  - Spec JAppServer2001

What is an enterprise  
application?

# Enterprise applications

- E-bay
- Amazon
- Campus course registration system

# Features of Enterprise Applications

- Persistent data
  - databases
  - integrity
  - transactions
- Interfaces
  - web clients
  - web services
  - legacy applications

# More features

- Distribution and scaling
  - some enterprise applications may be deployed in low activity environment
  - should be possible to scale them if demand warrants it
  - often means distribution across multiple machines
  - often geographically distributed
- Resiliency to failure
  - one machine failure cannot shutdown an enterprise application

# Databases

- Relational databases the standard mechanism
  - scales well to very large data sizes
  - handles transactions and failure well
- SQL the standard language for talking to them

# JDBC

- Java standard for talking to databases
- Allows efficient mechanisms for talking SQL to databases
  - doesn't just read/write text streams

# Persistent Objects

- It is a pain to have to use SQL queries whenever you want to touch persistent state
- Better to have objects that reflect persistent state
  - and can check integrity rules and other business logic

# Persistent Object solutions

- EJB (Enterprise Java Beans)
  - BMP - Beans managed persistence
  - CMP - Container managed persistence
    - AppServer provides mapping between beans and database
- Java Data Objects
  - Good for small stuff

# Databases in one lecture

# Transactions

- **A**tomicity - indivisible set of actions
- **C**onsistency - respects database invariants
  - typically domain specific
- **I**solation - ignorant of other, uncommitted transactions
- **D**urability - persistent, even in case of system failure

# A song database

- Album:
  - albumID*
  - title
  - category
  - year
- Artist:
  - artistID*
  - name
- Song:
  - songID*
  - albumID
  - title
  - length
- Performance:
  - songID
  - artistID

# Relational databases

- Normalized form
  - 1st: All attributes are single valued
  - 2nd: non-key entries are functionally dependent on primary key
  - 3rd: No transitive dependencies exist
  - Also 4th and 5th normal forms

# Locking

- Transaction 1 reads current balance
  - sees \$100
- Transaction 2 reads current balance
  - sees \$100
- Transaction 2 adds \$10
  - sets current balance to \$110
- Transaction 1 adds \$10
  - sets current balance to \$110

# How to handle

- In threads, we would just use locks
- In databases, can use pessimistic or optimistic concurrency
  - pessimistic concurrency similar to standard locking
  - optimistic: don't lock
    - fail when committing if anyone else changed data you depended on