Outline

- Announcements
- MySQL case (cont’d)
- Databases
- Student Presentation - Akamai
Announcements

- Reading for 3/1
  - Messerchmitt 11.2 (pp.333-335)
  - Akamai Case (reader pp.217-236)

- Forthcoming Student Presentations
  - 3/1
    - Miles Carvalho Dreszer Bus Proj: Jamba Juice
    - Jonathan Apostol Jangar Bus Proj: Target
  - 3/3
    - Vidya Menon Kulavil Bus Proj: Walmart Mar 03
    - Owen Mathys Bus Proj: FedEx

- News Folio #3 due today
mySQL Case
**MySQL**

What does MySQL make?

**How Successful is MySQL?**

- Visibility: Fortune magazine, more mentions on www
- Reaction from giants
- Revenue growth 2001 700k, 2002 6.2m, 2003 10m
- Good performance reviews
- Recent SAP alliance
- But Market share tiny:
  - $10 million out of $10 billion market!

**Why Success?**

- Good Technology
- Large DBMS bloated with features most don't need
- Innovative OSS model
mySQL

How does OSS work?

Two Types of License:

- **GPL**
  - Free
  - No Support
  - Any software that uses MySQL as a module must itself be made GPL

- **Commercial License**
  - Support
  - Could be distributed with non-open source software
  - Not Free:
    - MySQL: Classic $250, Pro $495 (for ~ 50 users)
    - Compare to:
      - MSFT $3150 single proc for 50 users
      - IBM $33000 single proc for 50 users
      - Oracle $40000 single proc for 50 users
Aside: DB’s in different software stacks

- Which companies are competitors?
- Which are complimentary to each other?
- Which are both?
MySQL

- Which segments of market is MySQL strong in?
  - Large Companies or Small Companies?
  - Web applications or Critical Enterprise data?

- Why would a major enterprise want to pay so much more for an Oracle or IBM DB?
### My SQL: market

<table>
<thead>
<tr>
<th>Enterprise wide data 90%</th>
<th>Small 20%</th>
<th>Medium 30%</th>
<th>Large 50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft</td>
<td></td>
<td></td>
<td>Oracle IBM</td>
</tr>
<tr>
<td>Reliability</td>
<td></td>
<td></td>
<td>Support</td>
</tr>
<tr>
<td>Scalability</td>
<td></td>
<td></td>
<td>Longevity</td>
</tr>
<tr>
<td>Web Sites 10%</td>
<td>My SQL Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How should mySQL grow in order to meet it’s stated goal of getting to $100 million in revenue?

Figure Adapted from “Teaching Note for MySQL Open Source Database,” 6/1/04, Stanford GSB.
## My SQL: Growth Strategy

- Lack of Brand identity in this segment
- MySQL lacks the organization to offer support
- Large enterprises have high switching costs

<table>
<thead>
<tr>
<th></th>
<th>Small 20%</th>
<th>Medium 30%</th>
<th>Large 50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise wide data</td>
<td>Microsoft</td>
<td>Oracle</td>
<td>IBM</td>
</tr>
<tr>
<td>Web Sites</td>
<td>My SQL Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90%</td>
<td></td>
<td>Reliability</td>
<td></td>
</tr>
<tr>
<td>10%</td>
<td></td>
<td>Scalability</td>
<td>Support</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Longevity</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure Adapted from “Teaching Note for MySQL Open Source Database,” 6/1/04, Stanford GSB.
## My SQL: Growth Strategy

<table>
<thead>
<tr>
<th>Enterprise wide data</th>
<th>Small 20%</th>
<th>Medium 30%</th>
<th>Large 50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft</td>
<td></td>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IBM</td>
</tr>
<tr>
<td>Web Sites</td>
<td>My SQL</td>
<td></td>
<td>Reliability</td>
</tr>
<tr>
<td></td>
<td>Cost</td>
<td></td>
<td>Scalability</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Support</td>
</tr>
<tr>
<td>Stay Put?</td>
<td></td>
<td></td>
<td>Longevity</td>
</tr>
</tbody>
</table>

- Not a big enough market to reach stated $100 million goal.

Figure Adapted from “Teaching Note for MySQL Open Source Database,” 6/1/04, Stanford GSB.
My SQL: Growth Strategy

- Many of these customers already using MySQL with websites
- Less emphasis on global organization
- Leverage SAP alliance
- Up against Microsoft.

Figure Adapted from “Teaching Note for MySQL Open Source Database,” 6/1/04, Stanford GSB.
**My SQL: Growth Strategy**

<table>
<thead>
<tr>
<th>Enterprise wide data 90%</th>
<th>Small 20%</th>
<th>Medium 30%</th>
<th>Large 50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Sites 10%</td>
<td>My SQL</td>
<td>Microsoft</td>
<td>Oracle IBM</td>
</tr>
<tr>
<td></td>
<td>Cost</td>
<td>90%</td>
<td>Reliability, Scalability, Support, Longevity</td>
</tr>
</tbody>
</table>

- + builds on existing brand and strengths
- - Market not so big

Figure Adapted from “Teaching Note for MySQL Open Source Database,” 6/1/04, Stanford GSB.
Database Management
Databases & MySQL Quiz

1) In a relational table, each record is represented by
   a) a row   b) a column   c) either a row or a column

2) Who are the top three RDBMS competitors of MySQL?

3) Linux is
   a) a proprietary operating system 
   b) a proprietary DBMS
   c) an open source operating system 
   d) an open source DBMS
Databases

- **Treat data as a separate asset**
  - May be shared by multiple applications

- **Provide protection and integrity features appropriate to mission-critical data**
  - Access control
  - Integrity constraints
  - Persistence
  - etc.
Two capabilities

**Aggregation**: accessing multiple databases

**Sharing**: two or more applications accessing the same databases

Databases
## Relational table

<table>
<thead>
<tr>
<th>Table</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Address</td>
</tr>
<tr>
<td>Record</td>
<td></td>
</tr>
</tbody>
</table>

Field/Attribute
Remember: Data properties

<table>
<thead>
<tr>
<th>EMPLOYERS</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP_ID</td>
<td>EMP_NAME</td>
<td>POSITION</td>
<td>DEPT_ID</td>
</tr>
<tr>
<td>100</td>
<td>Alice</td>
<td>Manager</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>Bob</td>
<td>Programmer</td>
<td>1</td>
</tr>
<tr>
<td>102</td>
<td>Chris</td>
<td>Manager</td>
<td>2</td>
</tr>
<tr>
<td>103</td>
<td>David</td>
<td>Accountant</td>
<td>2</td>
</tr>
</tbody>
</table>

Field name

Table name

Field Value

Field Type
e.g. EMP_ID is INTEGER

EMPL_NAME is STRING
The Relational Model

<table>
<thead>
<tr>
<th>NAME</th>
<th>POSITION</th>
<th>DEPARTMENT</th>
<th>DEPT.</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Manager</td>
<td>IT</td>
<td></td>
<td>San Jose</td>
</tr>
<tr>
<td>Bob</td>
<td>Programmer</td>
<td>IT</td>
<td></td>
<td>San Jose</td>
</tr>
<tr>
<td>Chris</td>
<td>Manager</td>
<td>Finance</td>
<td></td>
<td>New York</td>
</tr>
<tr>
<td>David</td>
<td>Accountant</td>
<td>Finance</td>
<td></td>
<td>New York</td>
</tr>
</tbody>
</table>

- Not all data are stored in the same table
  - Avoid data replication/redundancy
  - Enable faster/efficient computations over data
## The Relational Model

### Employers

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>NAME</th>
<th>POSITION</th>
<th>DEPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Alice</td>
<td>Manager</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>Bob</td>
<td>Programmer</td>
<td>1</td>
</tr>
<tr>
<td>102</td>
<td>Chris</td>
<td>Manager</td>
<td>2</td>
</tr>
<tr>
<td>103</td>
<td>David</td>
<td>Accountant</td>
<td>2</td>
</tr>
</tbody>
</table>

### Departments

<table>
<thead>
<tr>
<th>DEPT_ID</th>
<th>DEPARTMENT</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IT</td>
<td>San Jose</td>
</tr>
<tr>
<td>2</td>
<td>Finance</td>
<td>New York</td>
</tr>
</tbody>
</table>
# The Relational Model

## Employers

<table>
<thead>
<tr>
<th>EMPL_ID</th>
<th>EMPL_NAME</th>
<th>EMPL_POSITION</th>
<th>DEPT.ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Alice</td>
<td>Manager</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>Bob</td>
<td>Programmer</td>
<td>1</td>
</tr>
<tr>
<td>102</td>
<td>Chris</td>
<td>Manager</td>
<td>2</td>
</tr>
<tr>
<td>103</td>
<td>David</td>
<td>Accountant</td>
<td>2</td>
</tr>
</tbody>
</table>

## Departments

<table>
<thead>
<tr>
<th>DEPT.ID</th>
<th>DEPARTMENT</th>
<th>DEPT. ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IT</td>
<td>San Jose</td>
</tr>
<tr>
<td>2</td>
<td>Finance</td>
<td>New York</td>
</tr>
</tbody>
</table>
The Relational Model

<table>
<thead>
<tr>
<th>EMPLOYERS</th>
<th>EMPLOYERS</th>
<th>EMPLOYERS</th>
<th>EMPLOYERS</th>
<th>EMPLOYERS</th>
<th>EMPLOYERS</th>
<th>EMPLOYERS</th>
<th>EMPLOYERS</th>
<th>EMPLOYERS</th>
<th>EMPLOYERS</th>
<th>EMPLOYERS</th>
<th>EMPLOYERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPL_ID</td>
<td>EMPL_NAME</td>
<td>EMPL_POSITION</td>
<td>DEPT.ID</td>
<td>EMPL_ID</td>
<td>EMPL_NAME</td>
<td>EMPL_POSITION</td>
<td>DEPT.ID</td>
<td>EMPL_ID</td>
<td>EMPL_NAME</td>
<td>EMPL_POSITION</td>
<td>DEPT.ID</td>
</tr>
<tr>
<td>100</td>
<td>Alice</td>
<td>Manager</td>
<td>1</td>
<td>101</td>
<td>Bob</td>
<td>Programmer</td>
<td>1</td>
<td>102</td>
<td>Chris</td>
<td>Manager</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>Bob</td>
<td>Programmer</td>
<td>1</td>
<td>102</td>
<td>Chris</td>
<td>Manager</td>
<td>2</td>
<td>103</td>
<td>David</td>
<td>Accountant</td>
<td>2</td>
</tr>
</tbody>
</table>

DEPARTMENTS

<table>
<thead>
<tr>
<th>DEPT. ID</th>
<th>DEPARTMENT</th>
<th>DEPT. ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IT</td>
<td>San Jose</td>
</tr>
<tr>
<td>2</td>
<td>Finance</td>
<td>New York</td>
</tr>
</tbody>
</table>

Each table should have a unique identifier for each record: KEY.

Tables are connected using these KEYS.
## The Relational Model

### EMPLOYERS

<table>
<thead>
<tr>
<th>EMPL_ID</th>
<th>EMPL_NAME</th>
<th>EMPL_POSITION</th>
<th>DEPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Alice</td>
<td>Manager</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>Bob</td>
<td>Programmer</td>
<td>1</td>
</tr>
<tr>
<td>102</td>
<td>Chris</td>
<td>Manager</td>
<td>2</td>
</tr>
<tr>
<td>103</td>
<td>David</td>
<td>Accountant</td>
<td>2</td>
</tr>
</tbody>
</table>

### DEPARTMENTS

<table>
<thead>
<tr>
<th>DEPT. ID</th>
<th>DEPARTMENT</th>
<th>DEPT. ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IT</td>
<td>San Jose</td>
</tr>
<tr>
<td>2</td>
<td>Finance</td>
<td>New York</td>
</tr>
</tbody>
</table>
The Relational Model

- Relational Model
  - Based on relational algebra
  - Specifies a suite of operations/operators that can be performed on tables
  - These operations are expressed by SQL commands
SQL interface

- SQL (Structured Query Language)
- Forms an “interface” between an application and the DBMS
  - For manipulating, and extracting data from database
- Standardized, not vendor specific

- Encapsulates various internal details
  - Data partitioning and replication
  - Host mapping
  - File representation
  - etc.
Database operations

Each operation results in a new table
# Database Operations

## EMPLOYERS

<table>
<thead>
<tr>
<th>EMPL_ID</th>
<th>EMPL_NAME</th>
<th>EMPL_POSITION</th>
<th>DEPT. ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Alice</td>
<td>Manager</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>Bob</td>
<td>Programmer</td>
<td>1</td>
</tr>
<tr>
<td>102</td>
<td>Chris</td>
<td>Manager</td>
<td>2</td>
</tr>
<tr>
<td>103</td>
<td>David</td>
<td>Accountant</td>
<td>2</td>
</tr>
</tbody>
</table>

## DEPARTMENTS

<table>
<thead>
<tr>
<th>DEPT. ID</th>
<th>DEPARTMENT</th>
<th>DEPT. ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IT</td>
<td>San Jose</td>
</tr>
<tr>
<td>2</td>
<td>Finance</td>
<td>New York</td>
</tr>
</tbody>
</table>

## EMPL_DEPT

<table>
<thead>
<tr>
<th>EMPL_ID</th>
<th>EMPL_NAME</th>
<th>EMPL_POSITION</th>
<th>DEPT.</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Alice</td>
<td>Manager</td>
<td>IT</td>
<td>San Jose</td>
</tr>
<tr>
<td>101</td>
<td>Bob</td>
<td>Programmer</td>
<td>IT</td>
<td>San Jose</td>
</tr>
<tr>
<td>102</td>
<td>Chris</td>
<td>Manager</td>
<td>Finance</td>
<td>New York</td>
</tr>
<tr>
<td>103</td>
<td>David</td>
<td>Accountant</td>
<td>Finance</td>
<td>New York</td>
</tr>
</tbody>
</table>
Application Logic and Tables

- Application logic (programming language)
- SQL
- Database
mySQL Case
Two applications might try to access & update the same resource concurrently.
Algorithms and protocols

Adapted from
David G. Messerschmitt
Algorithm

Specified sequence of steps that
- accomplish a designated task
- in a finite number of steps

Representation:
- simple algorithm: flowchart
- complicated algorithm: program
Example: one turn at monopoly

Start turn → Throw dice → Move token number of squares indicated on dice → Land on “go to jail”? → Yes → Move to “jail” square → No → Do not move; follow policies for square (like “pay rent”) → Finish turn
Algorithm building blocks

- **Sequence**
  - Start
  - Action
  - Action
  - Action
  - Finish

- **Selection**
  - Start
  - Decision
  - Action
  - Action
  - Finish

- **Loop**
  - Start
  - Action
  - Test
  - Programming languages support these three building blocks
  - Finish

**Algorithm building blocks**

- Start
- Action
- Action
- Action
- Finish

- Start
- Decision
- Action
- Action
- Finish

- Start
- Action
- Test
- Finish

- Start
Protocol

- Distributed algorithm ...
- Realized by two or more modules to coordinate their actions or accomplish some shared task
- Module interoperability requires a protocol
  - Prescribed order of method invocations
  - Part of interface documentation
Monopoly players protocol

This is a protocol interaction diagram

One-turn algorithm

Time
Internally, the network uses protocols to implement the services it provides.
Example:

HEADQUARTERS
Airline Dataserver

Airline Intranet

HHC Server

Wireless Link

HHC

Airline
Layered Protocols Example

- **HHC Server**
  - HHC Server Application
  - Windows OS
  - Networking Infrastructure
  - Break Messages into Packets

- **HHC Application**
  - HHC Application
  - Palm OS
  - Networking Infrastructure

- **Application Level Protocol**
  - Request Pass. Data
  - Send Pass. Data As Message
  - Send Packet
  - Acknowledge Packet

- **Link Level Protocol**
Three simple protocols

One-way message: send-receive
Two-way interaction: request-response
Push: publish-subscribe
Send - Receive

Client → Server
send

Server ← Client
receive

Time
Request - Response
Send - Acknowledge
Example: HTTP (Hyper Text Transfer Protocol)

1. User activates URL
2. HTTP request
3. HTTP response (embedded document)
4. Browser displays document (if HTML) or invokes “helper application”
Locating things

by

David G. Messerschmitt
Three ways of locating things

Name
- "Joe Bloe"

Address
- "1299 Hearst St, Berkeley, CA"

Reference
- "Postmaster of Berkeley CA"
Name

- Symbolic (character string) representation
- Easy for people to remember or guess
- Identifies, but
  - Does not locate directly
    - Distinction important for mobile entities
- Not unique: entities can have more than one name (called aliases)
Hierarchical names

Hierarchy makes names easier to remember or guess

Host domain names:
- “info.sims.berkeley.edu”
- designates administrative hierarchy

File names:
- “c:\My Documents\Docs\Resume.doc”
- designates folder hierarchy
Address

- Route or path to entity
  - is directly specified, or
  - can be inferred

- Independent of who or what is locating entity

- Topological specification
Example

Path from [color] to [color] is (R,D,D,D,R,R,R,R)

Is (R,D,D,D,R,R,R,R) an address?
No! -- not an address, because it depends on starting point.
Example

Address of is (6,5)

Route from can be inferred
Reference

Abstract representation of an entity

Interaction is with representation
- infrastructure arranges redirection to actual entity
- especially appropriate for things that move

Example
- A Cell phone number is a reference.
- A Wired phone number is an address.
Name services

1. Name
2. Address or reference
3. Interaction