JDBC IV

IS 313
1.30.2003
Outline

- Batch processing
- Transactions
- Homework #2
- Examples
Omission

- Not every ResultSet is updatable
  - usually when the query is a join
Distributed Application

![Diagram showing a client program communicating with a database through JDBC and a driver.](image-url)
Problem

- Cost of moving data between client and server
- One solution
  - Stored procedure: move computation to data
- Another
  - Try to reduce the cost by reducing overhead
  - Communication costs are “chunky”
Non-linear Communication Costs

![Graph showing non-linear communication costs]

- Communication time vs. Data quantity
Example

- Suppose 12 small operations
- Execution time
  - $= 12 \times (\text{op\_time} + \text{comm\_in} + \text{comm\_out})$
- If grouped together
  - $= 12 \times \text{op\_time} + \text{comm\_in} + \text{comm\_out}$
Steps

1. Turn off auto commit
2. For each update, call addBatch instead of executeUpdate
3. when complete, call executeBatch
Batch processing

```java
con.setAutoCommit(false);
... JDBC calls ...
Statement stmt = con.prepareStatement ("INSERT ...;");
... set parameters ...
// ready to update
stmt.addBatch();  // instead of executeUpdate()
... more stuff ...
stmt2.addBatch ();
... more stuff ...
stmt3.addBatch ();
// updates complete
con.executeBatch();
con.executeBatch();
con.setAutoCommit(true);
```
Summary - Batch Processing

- Useful when there are many updates

- Benefits
  - Lower communication overhead to database

- Problems
  - Each statement treated separately
    - No chance to handle exceptions that affect later updates
Transactions

- Goal
  - In the face of machine failure, keep the database consistent
Transactions

- Atomicity
  - No partial success
- Consistency
  - End point is consistent
- Isolation
  - Transaction invisible until completed
- Durability
  - Committed actions survive system failure
Example

- Reservation is added
- Actions
  - Record inserted into Reservation table
  - Capacity table updated with new reservation count
Steps

- 1. Set auto commit to false
- 2. perform updates as normal
- 3. Commit transaction
- 4. Set auto commit back to true
- If there is a failure, rollback the transaction
  - typically exception handler
Example

```java
try {
    conn.setAutoCommit (false);
    addReservation (conn);
    updateCapacity (conn);
    conn.commit ();
} catch (SQLException e) {
    try {
        conn.rollback ();
        DBUtilities.displaySQLException (e);
    } catch (SQLException e2) { DBUtilities.displaySQLException (e2); }
} finally {
    try {
        conn.setAutoCommit (true);
    } catch (SQLException e3) { DBUtilities.displaySQLException (e3); }
}
```
DatabaseMetaData metadata = con.getMetaData();
ResultSet tables = metadata.getTables(null, null, null, null, null);
while (rs.next())
{
    System.out.println (tables.getString ("name"));
}
Homework #2

- Hotel reservations
- Database
- Interactive text mode
Interaction

% java HotelTextMode
  ➢ add
  Enter first name: Jack
  Enter last name: Spratt
  Enter date of arrival (mm/dd/yy): 1/30/03
  Enter room type: 0
  Enter # of guests: 2
  Enter number of days: 5
  ➢ date
  Enter date to search (mm/dd/yy): 1/30/03
  Reservation Id: 45
  Customer: Jack Spratt
  Room: 0
  Arrival: 1/30/03
  2 guests, staying 5 days
  ➢ delete
  Enter reservation id: 45
  ➢ exit
%
Commands

- add
- find by date
- help
- (exit)
- (capacity)
- delete
- find by name
- show all
How to design?
Adaptability

- What are the most likely vectors of change?
Command Pattern

- Represent users action with an object
- Framework in which command objects are created and executed
Central Loop

- Loop
  - Create command object
  - Set parameters
  - Execute
Command Line Interaction

% java HotelTextMode delete 37
%

Command Hierarchy
Advice

- Start right away
- Don’t use JDK 1.3
- Close statements and result sets
- Start right away
Programming example

- Exit command
- Capacity command