SQL in applications; NoSQL

Rasmus Pagh
Today’s lecture

• Based on sections 8.1-8.5

• SQL in applications
  – Focus on ODBC/JDBC

• NoSQL technologies, by example
  – MapReduce (Hadoop)
  – BigTable (and descendants)
  – Graph databases (Neo4J)
  – Overview article: www.infoworld.com/print/167400
SQL in applications

• Most SQL databases are at the back end of applications
  – important to know how this works.

• On the surface, a very boring subject
  – How to move data from A to B doing suitable translation, etc.

• Also a very interesting topic!
  – Focus of a lot of research and development.

• In this course we will stay pretty much at the surface...
SQL in applications

Several flavors:
• Dynamic or static SQL?
• Library or “native” support?

Things that need to be addressed
• How to deal with DBMS errors?
• How to specify transactions?
• How to access query results?
• DBMS independence?
JDBC connection

```java
String url = "jdbc:mysql://localhost/";
String dbName = "imdb";
String driver = "com.mysql.jdbc.Driver";
String userName = "root";
String password = "";

try {
    Class.forName(driver);
    Connection conn = DriverManager.getConnection(url+dbName,userName,password);
    System.out.println("Connected to MySQL");
    conn.close();
    System.out.println("Disconnected from MySQL");
} catch (Exception e) {
    e.printStackTrace();
}
```
Use caution when creating SQL based on user input!
Hi, this is your son’s school. We’re having some computer trouble.

Oh, dear – did he break something? In a way –

Did you really name your son Robert’); DROP TABLE Students;-- ?

Oh, yes. Little Bobby Tables, we call him.

Well, we’ve lost this year’s student records. I hope you’re happy.

And I hope you’ve learned to sanitize your database inputs.

xkcd.com
**JDBC static SQL**

```java
PreparedStatement insertPerson =
    conn.prepareStatement("INSERT INTO person VALUES (?,?,?,null,null,null)"); // Create prepared statement
insertPerson.setInt(1, 123456);
insertPerson.setString(2, "John Doe");
insertPerson.setString(3, "M");
insertPerson.setDate(4, new java.sql.Date(1606176000000L)); // Set date, given in milliseconds
insertPerson.setNull(6, java.sql.Types.INTEGER); // Set to NULL
insertPerson.executeUpdate(); // Execute prepared statement with current parameters
```
Efficiency issues

• Connection takes time to establish – use 1 connection for many operations.
• It takes time to parse dynamic SQL – prepared statements start executing faster.
• ORDER BY may force creation of full result within the DBMS before any output reaches the application.
  • Why is this not just usual? (Answer in next slide.)
Cursors

• Common to not generate full results of queries, but provide a "cursor" that allows the result to be traversed.

• JDBC examples:
  
  – `Statement s = con.createStatement(ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_READ_ONLY)`
  
  – `Statement s = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE)`
Four examples

1. Movies by year – imperative way
2. Movies by year – SQL centric way
3. Iterating through a large result set
4. Iterating through a filtered result set
Automatic code generation

• Instead of dealing directly with JDBC, one can automatically generate code to make objects “persistent” in a database.
  – E.g. Nhibernate

• Advantage: Tedious code made with very little effort.

• Disadvantage: Little and indirect control over efficiency issues.
Language integration

• “Little languages” with tight database integration.

• New query sublanguages for mainstream languages such as C#.
  - If used with conventional DBMS: Automatically translated to SQL.
NO-SQL

• Silly to indentify technologies with what it is not.
• Better: Not Only SQL.

But what is it?
• Lemire: Programmer’s revolt against database administrators.
• Common reason: Independence from very expensive large DBMSs.
XML databases

• 8-10 years ago believed to be the up-and-coming database technology.

• Status now:
  - XML is mainly a textual data format.
  - XML support built into relational DBMSs.
  - XML database systems eXist, but have small market share.

• Much more info in 3, 4, and 5 weeks!
MapReduce

- Google system for distributed queries on line-based data.
- Runs on a cluster of networked machines (can be 1000s).
- Open source version: Hadoop
- Builds on distributed file system: Does not deal with transactions.
MapReduce in terms of SQL

- SELECT myFunction1
  FROM myFunction2(R)
  GROUP BY key

- Mapper transform the input into lines with keys and values.
- Reducer transforms a group of values with the same key into an output.
- Language for mapper and reducer not specified (typical: Python, Java).
MapReduce examples

1. Word count
   Mapper: Transform text lines into pairs \((w,1)\).
   Reducer: Add the occurrences of each word.

2. \(R_1\) NATUAL JOIN \(R_2\)
   Mapper: Make the join attribute key of each tuple.
   Reducer: For each key value, output cartesian product of tuples in \(R_1\) and \(R_2\).

More complex queries can often be made using several MapReduce passes.
BigTable

- Google system for storing data persistently in a distributed system.
- Many similar systems since then (distributed hash tables, Cassandra,...).
- Data model generalizes the relational model. System stores a function
  \((\text{rowId: string, column: string, time: int}) \rightarrow \text{string}\)
- Only simple queries:
  - E.g. lookup string using rowId and column.
- Only simple transactions:
  Modify a single row.
BigTable discussion

• Many DBMSs are mainly used to store data persistently – only need simple updates and queries.

• If data set is large and/or high reliability is desired, a distributed solution is desirable (all data replicated for availability).

• Often distributed storage systems offer relaxed consistency compared to a DBMS (e.g. “eventual consistency”).
Neo4J

• Database especially oriented towards storing graphs (in the sense of computer science).

• Query language specifies way of traversing graph to compute result – inspired by Xpath query language.

• Common graph search algorithms built – hard or inefficient to simulate using a traditional DBMS implementation.
Social data (customer: brand-name social network)

Graph example 1
Graph example 2

Spatial data (customer: large telecom company)

1. Name = "Omni Hotel"
   Lat = 3492848
   Long = 283823423

2. Name = ...
   Length = 3 miles

3. Name = ...
   Lat, Long = ...
   Length = 7 miles

4. Name = ...

5. Name = "Swedland"
   Lat = 23410349
   Long = 2342348852
Social AND spatial data (customer: LBS)

name = “Omni Hotel”
lat = 3492848
long = 283823423

name = “The Tavern”
lat = 1295238237
long = 234823492

name = “Emil”
beer_qual = expert

name = “Maria”
age = 30
beer_qual = non-existant

name = “Peter”

length = 3 miles

ROAD

LIKES

ROAD

SIBLING

KNOWS
Graph database discussion

- Any graph can be modeled in a relational DB, but not vice versa.
- A relational DBMS can store relations with two attributes as a graph adjacency list (using an index).
- But SQL is not made with typical “graph queries” in mind.
  - Example: Two-link references in IMDB.
- **Open**: Will relational DBMS providers create special functionality for relations that contain graph data?
Guest lecture

• On November 8, Claus Samuelsen from IBM will give a guest lecture.
• He will talk about how IBM use traditional database and NoSQL technologies in projects.