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

```
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");
```

Database operations

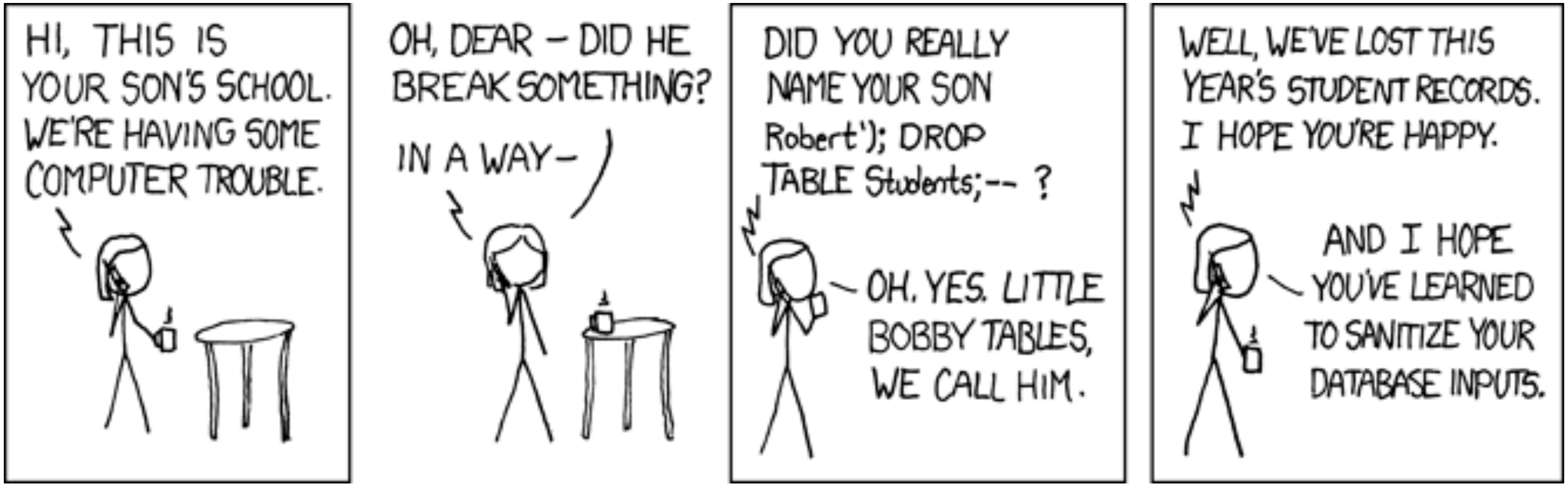
```
    conn.close();
    System.out.println("Disconnected from MySQL");
} catch (Exception e) {
    e.printStackTrace();
}
```

JDBC dynamic SQL

```
try {
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery("SELECT gender, count(*) FROM person GROUP BY gender");
    while (rs.next()) {
        System.out.println(rs.getString("gender")+": "+rs.getInt(2));
    }

    st.executeUpdate("DROP TABLE IF EXISTS JDBCtest");
    st.executeUpdate("CREATE TABLE JDBCtest(id int, string varchar(10))");
    st.executeUpdate("INSERT INTO JDBCtest VALUES (1,\"Tada!\")");
}
catch(SQLException s){
    System.out.println(s.toString());
}
```

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

```
PreparedStatement insertPerson =
conn.prepareStatement("INSERT INTO person VALUES (?,?,?,?,?,?)"); // Create preparea
insertPerson.setInt(1, 123456);
insertPerson.setString(2, "John Doe");
insertPerson.setString(3, "M");
insertPerson.setDate(4, new java.sql.Date(160617600000)); // Set date, given in mili
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.
  - E.g. ”Ruby on Rails”, `http://en.wikipedia.org/wiki/Ruby_on_Rails`
- New query sublanguages for mainstream languages such as C#.
  - E.g. LINQ, `http://en.wikipedia.org/wiki/Language_Integrated_Query`
  - If used with conventional DBMS: Automatically translated to SQL.

NO-SQL

- Silly to indentify technologies with what it is not.
- Better: **N**ot **O**nly **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 $\overbrace{\text{in}}^{\text{(simplified)}}$ terms of SQL

- ```
  SELECT myFunction1        Reducer
  FROM myFunction2(R)       Mapper
  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
  `(rowId:string,column:string,time:int) → string`
- Only simple queries:
  - E.g. lookup string using rowId and column.
- Only simple transactions:
  Modify a single row.

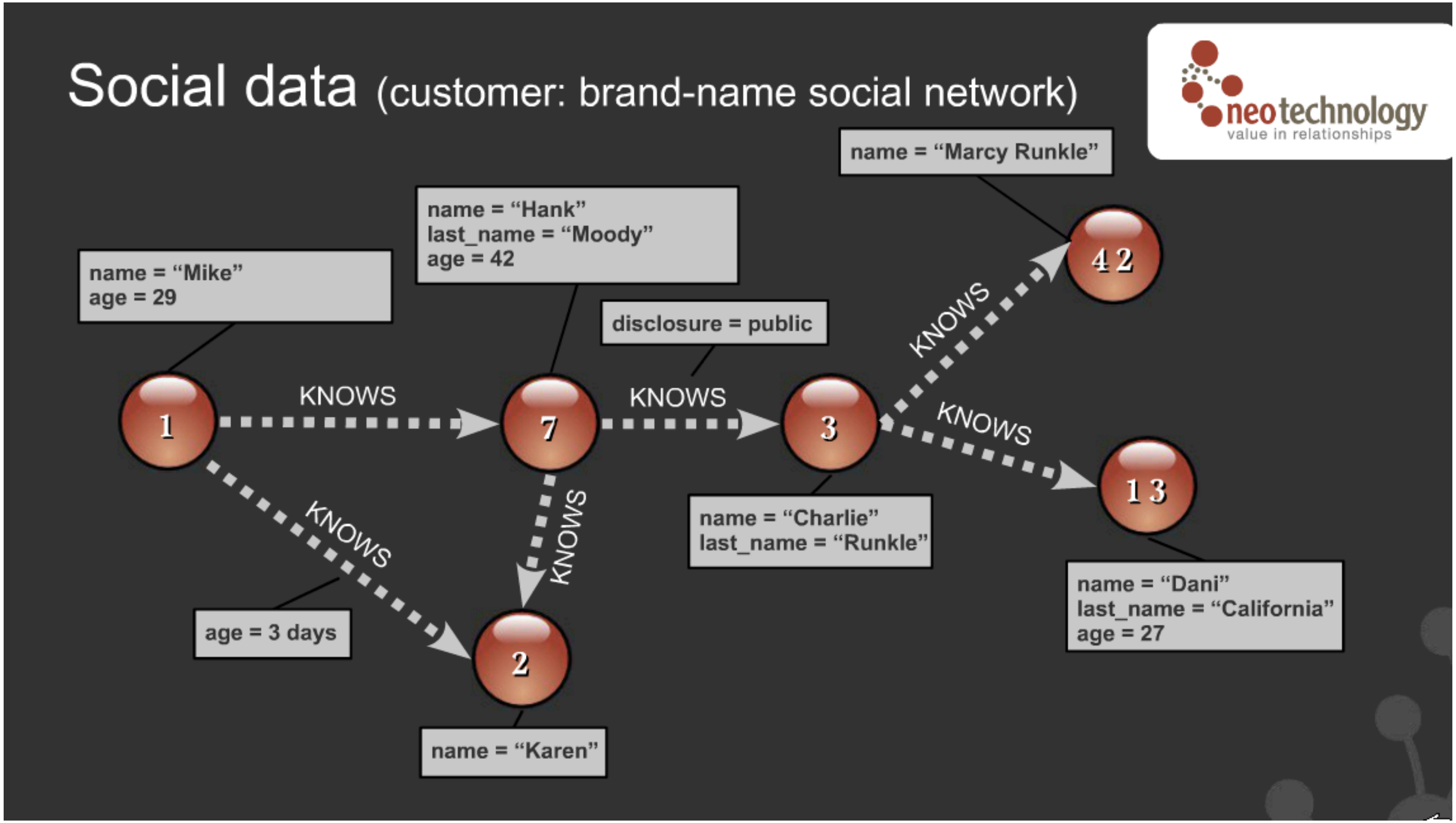
BigTable discussion

- 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”).

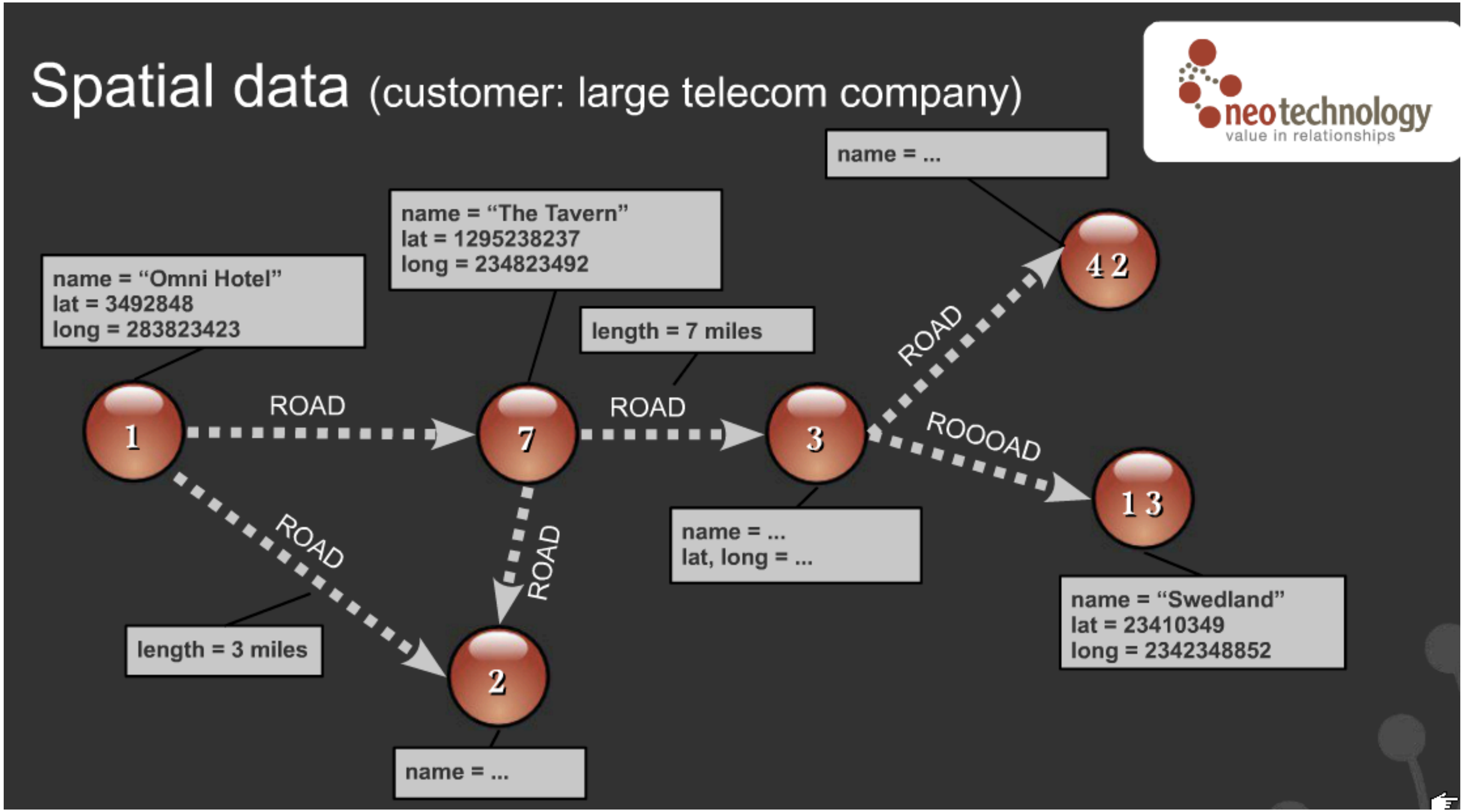
Neo4J

- 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.

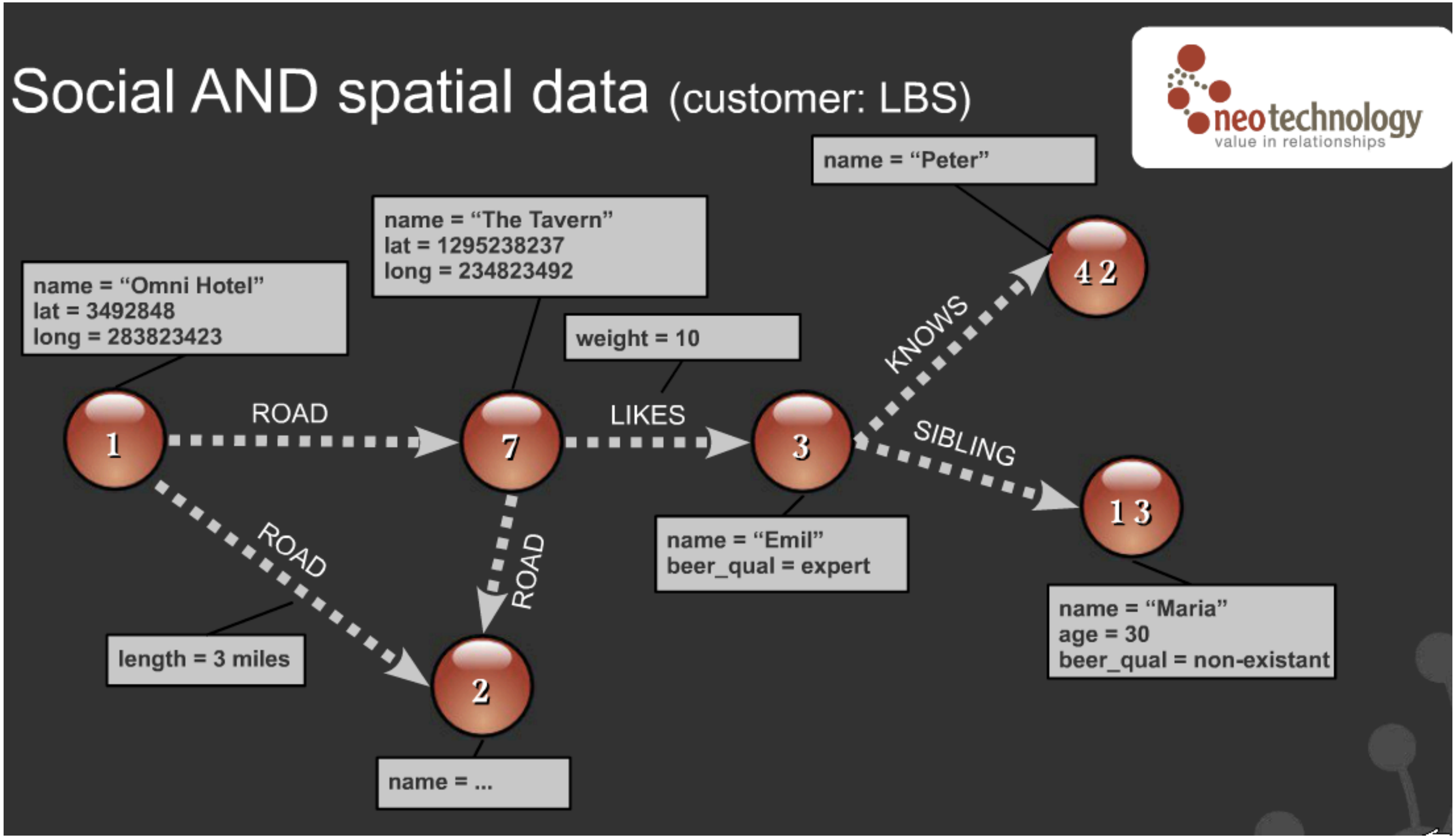
Graph example 1



Graph example 2



Graph example 3



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.