References


Ruby PostgreSQL, http://ruby.scripting.ca/postgres/rdoc/
Few More SQL Commands

mysql> ALTER TABLE students ADD column foo CHAR (40);  
Query OK, 1 row affected (0.03 sec)  
Records: 1  Duplicates: 0  Warnings: 0

mysql> DROP TABLE students;  
Query OK, 0 rows affected (0.01 sec)

mysql> DROP DATABASE lectureexamples;  
Query OK, 0 rows affected (0.00 sec)
An Example

PostgreSQL Version
CREATE TABLE faculty (  
    name  CHAR(20) NOT NULL,
    faculty_id  SERIAL   PRIMARY KEY
);

MySQL Version
CREATE TABLE faculty (  
    name  CHAR(20) NOT NULL,
    faculty_id  INTEGER AUTO_INCREMENT   PRIMARY KEY
);
Indices

Indices make accessing faster

Primary keys automatically have an index

The CREATE INDEX command creates indices

    CREATE INDEX faculty_name_key on faculty (name);
Adding Values

INSERT INTO faculty (name) VALUES ('Whitney');
INSERT INTO faculty (name) VALUES ('Beck');
INSERT INTO faculty (name) VALUES ('Anantha');
INSERT INTO faculty (name) VALUES ('Vinge');

select * from faculty;

Result

<table>
<thead>
<tr>
<th>name</th>
<th>faculty_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>1</td>
</tr>
<tr>
<td>Beck</td>
<td>2</td>
</tr>
<tr>
<td>Anantha</td>
<td>3</td>
</tr>
<tr>
<td>Vinge</td>
<td>4</td>
</tr>
</tbody>
</table>

(4 rows)
## Second Table

<table>
<thead>
<tr>
<th>start_time</th>
<th>end_time</th>
<th>day</th>
<th>faculty_id</th>
<th>office_hour_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:00</td>
<td>11:00</td>
<td>Wed</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8:00</td>
<td>12:00</td>
<td>Mon</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>17:00</td>
<td>18:30</td>
<td>Tue</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>9:00</td>
<td>10:30</td>
<td>Tue</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9:00</td>
<td>10:30</td>
<td>Thu</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>15:00</td>
<td>16:00</td>
<td>Fri</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>faculty_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>1</td>
</tr>
<tr>
<td>Beck</td>
<td>2</td>
</tr>
<tr>
<td>Anantha</td>
<td>3</td>
</tr>
<tr>
<td>Vinge</td>
<td>4</td>
</tr>
</tbody>
</table>
Generating Second Table

PostgreSQL
CREATE TABLE office_hours (  
    start_time    TIME NOT NULL,  
    end_time    TIME NOT NULL,  
    day    CHAR(3) NOT NULL,  
    faculty_id INTEGER REFERENCES faculty,  
    office_hour_id    SERIAL    PRIMARY KEY  
);  

MySQL
CREATE TABLE office_hours (  
    start_time    TIME NOT NULL,  
    end_time    TIME NOT NULL,  
    day    CHAR(3) NOT NULL,  
    faculty_id INTEGER REFERENCES faculty,  
    office_hour_id INTEGER AUTO_INCREMENT PRIMARY KEY  
);
Adding Office Hours

Simple Insert

```sql
INSERT
    INTO office_hours ( start_time, end_time, day, faculty_id )
VALUES ( '10:00:00', '11:00:00', 'Wed', 1 );
```

The problem is that we need to know the id for the faculty
Adding Office Hours

Using Select

```
INSERT INTO office_hours (start_time, end_time, day, faculty_id)
SELECT '8:00:00' AS start_time,
      '12:00:00' AS end_time,
      'Mon' AS day,
      faculty_id AS faculty_id
FROM faculty
WHERE name = 'Beck'
```
Selecting Office Hours

SELECT
    name, start_time, end_time, day
FROM
    office_hours, faculty
WHERE
    faculty.faculty_id = office_hours.faculty_id;

<table>
<thead>
<tr>
<th>name</th>
<th>start_time</th>
<th>end_time</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>10:00:00</td>
<td>11:00:00</td>
<td>Wed</td>
</tr>
<tr>
<td>Beck</td>
<td>08:00:00</td>
<td>12:00:00</td>
<td>Mon</td>
</tr>
<tr>
<td>Whitney</td>
<td>17:00:00</td>
<td>18:30:00</td>
<td>Tue</td>
</tr>
<tr>
<td>Whitney</td>
<td>15:00:00</td>
<td>16:00:00</td>
<td>Fri</td>
</tr>
<tr>
<td>Anantha</td>
<td>09:00:00</td>
<td>10:30:00</td>
<td>Tue</td>
</tr>
<tr>
<td>Anantha</td>
<td>09:00:00</td>
<td>10:30:00</td>
<td>Thu</td>
</tr>
</tbody>
</table>
SELECT
  name AS Instructor,
  TEXT(start_time) || ' to ' || TEXT(end_time) AS Time,
  day AS Day
FROM
  office_hours, faculty
WHERE
  faculty.faculty_id = office_hours.faculty_id
ORDER BY
  Name

<table>
<thead>
<tr>
<th>Instructor</th>
<th>Time</th>
<th>Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anantha</td>
<td>09:00:00 to 10:30:00</td>
<td>Tue</td>
</tr>
<tr>
<td>Anantha</td>
<td>09:00:00 to 10:30:00</td>
<td>Thu</td>
</tr>
<tr>
<td>Beck</td>
<td>08:00:00 to 12:00:00</td>
<td>Mon</td>
</tr>
<tr>
<td>Whitney</td>
<td>10:00:00 to 11:00:00</td>
<td>Wed</td>
</tr>
<tr>
<td>Whitney</td>
<td>17:00:00 to 18:30:00</td>
<td>Tue</td>
</tr>
<tr>
<td>Whitney</td>
<td>15:00:00 to 16:00:00</td>
<td>Fri</td>
</tr>
</tbody>
</table>
Sample Selection

SELECT  
    name, start_time, end_time, day  
FROM  
    office_hours, faculty  
WHERE  
    faculty.faculty_id = office_hours.faculty_id  
AND  
    start_time > '09:00:00'  
AND  
    end_time < '16:30:00'  
ORDER BY  
    Name;

<table>
<thead>
<tr>
<th>name</th>
<th>start_time</th>
<th>end_time</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>10:00:00</td>
<td>11:00:00</td>
<td>Wed</td>
</tr>
<tr>
<td>Whitney</td>
<td>15:00:00</td>
<td>16:00:00</td>
<td>Fri</td>
</tr>
</tbody>
</table>
## Joins

### People

<table>
<thead>
<tr>
<th>id</th>
<th>first_name</th>
<th>last_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Roger</td>
<td>Whitney</td>
</tr>
<tr>
<td>2</td>
<td>Leland</td>
<td>Beck</td>
</tr>
<tr>
<td>3</td>
<td>Carl</td>
<td>Eckberg</td>
</tr>
</tbody>
</table>

### Email_Addresses

<table>
<thead>
<tr>
<th>id</th>
<th>user_name</th>
<th>host</th>
<th>person_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>beck</td>
<td>cs.sdsu.edu</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>whitney</td>
<td>cs.sdsu.edu</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>whitney</td>
<td>rohan.sdsu.edu</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>foo</td>
<td>rohan.sdsu.edu</td>
<td></td>
</tr>
</tbody>
</table>
Inner Join

Only uses entries linked in two tables

<table>
<thead>
<tr>
<th>first_name</th>
<th>last_name</th>
<th>user_name</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leland</td>
<td>Beck</td>
<td>beck</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>rohan.sdsu.edu</td>
</tr>
</tbody>
</table>

```sql
select first_name, last_name, user_name, host
from people, email_addresses
where people.id = email_addresses.person_id;
```
```sql
select first_name, last_name, user_name, host
from people inner join email_addresses
on (people.id = email_addresses.person_id);
```
Outer Left Join

Use all entries from the left table

<table>
<thead>
<tr>
<th>first_name</th>
<th>last_name</th>
<th>user_name</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leland</td>
<td>Beck</td>
<td>beck</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>rohan.sdsu.edu</td>
</tr>
<tr>
<td>Carl</td>
<td>Eckberg</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
select first_name, last_name, user_name, host
from people left outer join email_addresses
on (people.id = email_addresses.person_id);
```
## Right Outer Join

Use all entries from the right table

<table>
<thead>
<tr>
<th>first_name</th>
<th>last_name</th>
<th>user_name</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leland</td>
<td>Beck</td>
<td>beck</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>rohan.sdsu.edu</td>
</tr>
<tr>
<td></td>
<td></td>
<td>foo</td>
<td>rohan.sdsu.edu</td>
</tr>
</tbody>
</table>

```sql
select first_name, last_name, user_name, host
from people right outer join email_addresses
on (people.id = email_addresses.person_id);
```
A right outer join B == B left outer join A

The following two statements are equivalent

```
select first_name, last_name, user_name, host
from people right outer join email_addresses
on (people.id = email_addresses.person_id);
```

```
select first_name, last_name, user_name, host
from email_addresses left outer join people
on (people.id = email_addresses.person_id);
```
Normal forms

Defined by Dr. E. F. Codd in 1970

Reduce redundant data and inconsistencies
**First Normal Form (1NF)**

An entity is in the first normal form when all its attributes are single valued.

<table>
<thead>
<tr>
<th>Name</th>
<th>OfficeHour1</th>
<th>OfficeHour2</th>
<th>OfficeHour3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>10:00-11:00 W</td>
<td>17:00-18:30 Tu</td>
<td>15:00-16:00 Fri</td>
</tr>
<tr>
<td>Beck</td>
<td>8:00-12:00 M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anantha</td>
<td>9:00-10:30 Tu</td>
<td>9:00-10:30 Thu</td>
<td></td>
</tr>
</tbody>
</table>

What if someone has more than 3 office hours?
Wasted space for those that have fewer office hours

Not is 1NF since office hours are repeated.
### In 1NF

**Faculty**

<table>
<thead>
<tr>
<th>name</th>
<th>faculty_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>1</td>
</tr>
<tr>
<td>Beck</td>
<td>2</td>
</tr>
<tr>
<td>Anantha</td>
<td>3</td>
</tr>
</tbody>
</table>

**Office Hours**

<table>
<thead>
<tr>
<th>start_time</th>
<th>end_time</th>
<th>day</th>
<th>faculty_id</th>
<th>office_hour_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:00</td>
<td>11:00</td>
<td>Wed</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8:00</td>
<td>12:00</td>
<td>Mon</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>17:00</td>
<td>18:30</td>
<td>Tue</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>9:00</td>
<td>10:30</td>
<td>Tue</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9:00</td>
<td>10:30</td>
<td>Thu</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>15:00</td>
<td>16:00</td>
<td>Fri</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>
## Second Normal Form (2NF)

<table>
<thead>
<tr>
<th>cd_title</th>
<th>artist</th>
<th>music_type</th>
<th>cd_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Songs from the Trilogy</td>
<td>Glass</td>
<td>Modern Classical</td>
<td>1</td>
</tr>
<tr>
<td>I Stoten</td>
<td>Falu Spelmanslag</td>
<td>Swedish</td>
<td>2</td>
</tr>
<tr>
<td>Photographer</td>
<td>Glass</td>
<td>Modern Classical</td>
<td>3</td>
</tr>
</tbody>
</table>

An entity is in the second normal form if:

- It is in 1NF and
- All non-key attributes must be fully dependent on the entire primary key

Table is not in 2NF since different CDs

- Can have the same artists
- Can have same music type
At SDSU the schedule number uniquely identifies a course in a semester
So the term and schedule number uniquely identifies a course at SDSU
We can use term and schedule as the primary key

The table is in 1NF but not 2NF

Name, Time and Days are not fully dependent on the primary key
## Schedule in 2NF

### Schedule

<table>
<thead>
<tr>
<th>course_id</th>
<th>time_id</th>
<th>term_id</th>
<th>schedule_number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>09461</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>09472</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>09483</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>09494</td>
</tr>
</tbody>
</table>

### Courses

<table>
<thead>
<tr>
<th>course</th>
<th>title</th>
<th>name_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS635</td>
<td>Adv Obj Orient Dsgn Prog</td>
<td>1</td>
</tr>
<tr>
<td>CS651</td>
<td>Adv Multimedia Systems</td>
<td>2</td>
</tr>
<tr>
<td>CS683</td>
<td>Emerging Technologies</td>
<td>3</td>
</tr>
<tr>
<td>CS696</td>
<td>Writing Device Drivers</td>
<td>4</td>
</tr>
</tbody>
</table>

### Term

<table>
<thead>
<tr>
<th>semester</th>
<th>year</th>
<th>term_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall</td>
<td>2000</td>
<td>1</td>
</tr>
<tr>
<td>Spring</td>
<td>2001</td>
<td>2</td>
</tr>
</tbody>
</table>

### Time

<table>
<thead>
<tr>
<th>start_time</th>
<th>end_time</th>
<th>days</th>
<th>time_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:00:00</td>
<td>18:15:00</td>
<td>MW</td>
<td>1</td>
</tr>
<tr>
<td>18:30:00</td>
<td>19:45:00</td>
<td>MW</td>
<td>2</td>
</tr>
<tr>
<td>15:30:00</td>
<td>16:45:00</td>
<td>MW</td>
<td>3</td>
</tr>
<tr>
<td>15:30:00</td>
<td>16:45:00</td>
<td>TTh</td>
<td>4</td>
</tr>
<tr>
<td>Etc.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comments about Previous Slide

The schedule table is now in 2NF

What about the other tables?

If not how would you fix them?

Can you find a better way to decompose the original table?
### Third Normal Form (3NF)

#### Customer

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>City</th>
<th>State Name</th>
<th>State abbreviation</th>
<th>zip</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An entity is in third normal form if

It is in 2NF and
All non-key attributes must only be dependent on the primary key

State abbreviation depends on State Name

Table is not in 3NF
import java.sql.*;

public class SampleConnection
{
    public static void main (String args[]) throws Exception
    {
        String dbUrl = "jdbc:mysql://rugby.sdsu.edu:3306/test";
        String user = "whitney";
        String password = "mylittleSecret";
        System.out.println("Load Driver!");
        
        Class.forName("com.mysql.jdbc.Driver");
        Connection rugby;
        rugby = DriverManager.getConnection( dbUrl, user, password);
        Statement getTables = rugby.createStatement();
        ResultSet tableList =
            getTables.executeQuery("SELECT * FROM name");
        while (tableList.next() )
            System.out.println("Last Name: " + tableList.getString(1) + \t +
                "First Name: " + tableList.getString("first_name"));
        rugby.close();
    }
}
Documentation

MySQL
http://dev.mysql.com/doc/

PostgreSQL
http://www.postgresql.org/docs/
MySQL jdbc driver
http://dev.mysql.com/downloads/connector/j/5.0.html

Drivers must be in your classpath

PostgreSQL jdbc driver
http://jdbc.postgresql.org/index.html

Course accounts use PostgreSQL 7.4
JDBC Drivers

Java supports four types of JDBC drivers

JDBC-ODBC bridge plus ODBC driver
Java code access ODBC native binary drivers
ODBC driver accesses databases
ODBC drivers must be installed on each client

Native-API partly-Java driver
Java code accesses database specific native binary drivers

JDBC-Net pure Java driver
Java code accesses database via DBMS-independent net protocol

Native-protocol pure Java driver
Java code accesses database via DBMS-specific net protocol
JDBC URL Structure

jdbc:<subprotocol>:<dbname>

<subprotocol>
  Name of the driver or database connectivity mechanism

<dbname>
  Depends on the <subprotocol>, can vary with vendor

PostgreSQL

jdbc:postgresql:database
jdbc:postgresql://host/database
jdbc:postgresql://host:port/database

MySQL

jdbc:mysql://[host][,failoverhost...][:port][,[database]
[?propertyName1]=propertyValue1][&propertyName2]
[=propertyValue2]...
Loading Driver

In your code
Class.forName("com.mysql.jdbc.Driver");

Command line
java -Djdbc.drivers=org.postgresql.Driver
yourProgramName
DriverManager.getConnection

Three forms:

getConnection(URL, Properties)
getConnection(URL, userName, Password)
getConnection(URLWithUsernamePassword)

Form 1
    static String ARS_URL = "jdbc:oracle:@PutDatabaseNameHere";

    DriverManager.getConnection(ARS_URL, "whitney","secret");

Form 2
    DriverManager.getConnection(
        "jdbc:oracle:whitney/secret@PutDatabaseNameHere");

Form 3
    java.util.Properties info = new java.util.Properties();
    info.addProperty ("user","whitney");
    info.addProperty ("password","secret");

    DriverManager getConnection (ARS_URL ,info );
java.sql verses javax.sql

java.sql
DriverManager

javax.sql
DataSource
  Connection Pools
  Distributed
  Transactions
  Requires JNDI
JNDI
Java Naming and Directory Interface

Need JNDi Service Provider

http://java.sun.com/docs/books/tutorial/jndi/overview/index.html
Queries

executeUpdate
   Use for INSERT, UPDATE, DELETE or SQL that return nothing

executeQuery
   Use for SQL (SELECT) that return a result set

execute
   Use for SQL that return multiple result sets
   Uncommon
ResultSet

ResultSet - Result of a Query

JDBC returns a ResultSet as a result of a query

A ResultSet contains all the rows and columns that satisfy the SQL statement

A cursor is maintained to the current row of the data

The cursor is valid until the ResultSet object or its Statement object is closed

next() method advances the cursor to the next row

You can access columns of the current row by index or name

ResultSet has getXXX methods that:

have either a column name or column index as argument

return the data in that column converted to type XXX
**getObject**

A replacement for the getXXX methods

Rather than

```java
ResultSet tableList =
    getTables.executeQuery("SELECT * FROM name");
String firstName = tableList.getString( 1);
```

Can use

```java
ResultSet tableList =
    getTables.executeQuery("SELECT * FROM name");
String firstName = (String) tableList.getObject( 1);
```

ggetObject( int k) returns the object in the k’th column of the current row

ggetObject( String columnName) returns the object in the named column
<table>
<thead>
<tr>
<th>SQL type</th>
<th>Java type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>String</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>String</td>
</tr>
<tr>
<td>LONGVARCHAR</td>
<td>String</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>java.math.BigDecimal</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>java.math.BigDecimal</td>
</tr>
<tr>
<td>BIT</td>
<td>boolean</td>
</tr>
<tr>
<td>TINYINT</td>
<td>byte</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>short</td>
</tr>
<tr>
<td>INTEGER</td>
<td>int</td>
</tr>
<tr>
<td>BIGINT</td>
<td>long</td>
</tr>
<tr>
<td>REAL</td>
<td>float</td>
</tr>
<tr>
<td>FLOAT</td>
<td>double</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>double</td>
</tr>
<tr>
<td>BINARY</td>
<td>byte[]</td>
</tr>
<tr>
<td>VARBINARY</td>
<td>byte[]</td>
</tr>
<tr>
<td>LONGVARBINARY</td>
<td>byte[]</td>
</tr>
<tr>
<td>DATE</td>
<td>java.sql.Date</td>
</tr>
<tr>
<td>TIME</td>
<td>java.sql.Time</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>java.sql.Timestamp</td>
</tr>
</tbody>
</table>

Data Conversion
Some Result Set Issues

What happens when we call next() too many times?

What happens if we try to access data before we call next?

In both cases an java.sql.SQLException is thrown
Mixing ResultSets

Can't have two active result sets on same statement

```java
Connection rugby;
rugby = DriverManager.getConnection( dbUrl, user, password);
Statement getTables = rugby.createStatement();
ResultSet count =
    getTables.executeQuery("SELECT COUNT(*) FROM name");
ResultSet tableList =
    getTables.executeQuery("SELECT * FROM name");

while (tableList.next() )
    System.out.println("Last Name: " + tableList.getObject(1) + '\t' +
                       "First Name: " + tableList.getObject( "first_name");)

    // Raises java.sql.SQLException
    count.getObject(1);

    rugby.close();
```

disc this can happen when two threads have access to the same statement
Two Statements on one Connection work

Connection rugby;
rugby = DriverManager.getConnection(dbUrl, user, password);
Statement getTables = rugby.createStatement();
Statement tableSize = rugby.createStatement();

ResultSet count =
    getTables.executeQuery("SELECT COUNT(*) FROM name");
ResultSet tableList =
    tableSize.executeQuery("SELECT * FROM name");

while (tableList.next() )
    System.out.println("Last Name: " + tableList.getObject(1) + \t +
                      "First Name: " + tableList.getObject("first_name");

( "first_name"));
count.next();
System.out.println("Count: " + count.getObject(1) );
count.close();
tableList.close();
rugby.close();
Threads & Connections

Some JDBC drivers are not thread safe

If two threads access the same connection results may get mixed up

PostgreSQL & MySql drivers are thread safe

When two threads make a request on the same connection

The second thread blocks until the first thread get its results

Can use more than one connection but

Each connection requires a process on the database
Ruby MySQL

Documentation & Directions
http://www.kitebird.com/articles/ruby-mysql.html
Ruby PostgreSQL

Install

gem install ruby-postgres --rdoc

Docs

http://ruby.scripting.ca/postgres/rdoc/

Examples (Unix)

/usr/lib/ruby/gems/1.8/gems/ruby-postgres-0.7.1.2005.12.21/sample
require "postgres"

cs580 = PGconn.connect('bismarck.sdsu.edu',5432, nil, nil, 'cs580whitney', 'cs580whitney', 'password')

cs580.exec("DROP TABLE test") rescue nil
cs580.exec("CREATE TABLE test (first_name VARCHAR(20), last_name VARCHAR(20))")
cs580.exec("INSERT INTO test VALUES ('Roger', 'Whitney')")
cs580.exec("INSERT INTO test VALUES ('Roger', 'Rabbit')")

result = cs580.exec("SELECT * FROM test")
for field in result.fields
  printf("%-15s",field)
end
printf("\n")
result.result.each do |tuple|
  tuple.each do |fld|
    printf("%-15s",fld)
  end
  printf("\n")
end