CS 580 Client-Server Programming
Spring Semester, 2007
Doc 9 SQL, Normalization, JDBC
Feb 20, 2007

References

Oracle Design, Ensor & Stevenson, O'Reilly & Associates, Inc., 1997

MySQL On-line Manual http://www.mysql.com/doc/en/Reference.html

PostgreSQL Commands http://www.postgresql.org/idocs/index.php?sql-commands.html

http://java.sun.com/j2se/1.5/docs/guide/jdbc/index.html Sun's on-line JDBC Tutorial & Documentation

Connector-J, MySql JDBC Driver Documentation, http://www.mysql.com/products/connector/j/

PostgreSQL JDBC Documentation, http://jdbc.postgresql.org/documentation/docs.html

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
);
```

name	faculty_id
Whitney	1
Beck	2
Anantha	3

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

```
          name           |  faculty_id
-------------------------+--------------
 Whitney                 |            1
 Beck                    |            2
 Anantha                 |            3
 Vinge                   |            4
(4 rows)
```

Second Table

start_time	end_time	day	faculty_id	office_hour_id
10:00	11:00	Wed	1	1
8:00	12:00	Mon	2	2
17:00	18:30	Tue	1	3
9:00	10:30	Tue	3	4
9:00	10:30	Thu	3	5
15:00	16:00	Fri	1	6

name	faculty_id
Whitney	1
Beck	2
Anantha	3
Vinge	4

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

```
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;
```

name	start_time	end_time	day
Whitney	10:00:00	11:00:00	Wed
Beck	08:00:00	12:00:00	Mon
Whitney	17:00:00	18:30:00	Tue
Whitney	15:00:00	16:00:00	Fri
Anantha	09:00:00	10:30:00	Tue
Anantha	09:00:00	10:30:00	Thu

PostgreSQL only

```
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
```

Instructor	Time	Day
Anantha	09:00:00 to 10:30:00	Tue
Anantha	09:00:00 to 10:30:00	Thu
Beck	08:00:00 to 12:00:00	Mon
Whitney	10:00:00 to 11:00:00	Wed
Whitney	17:00:00 to 18:30:00	Tue
Whitney	15:00:00 to 16:00:00	Fri

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;
```

name	start_time	end_time	day
Whitney	10:00:00	11:00:00	Wed
Whitney	15:00:00	16:00:00	Fri

Joins

People

id	first_name	last_name
1	Roger	Whitney
2	Leland	Beck
3	Carl	Eckberg

Email Addresses

id	user_name	host	person_id
1	beck	cs.sdsu.edu	2
2	whitney	cs.sdsu.edu	1
3	whitney	rohan.sdsu.edu	1
4	foo	rohan.sdsu.edu	

Inner Join

Only uses entries linked in two tables

first_name	last_name	user_name	host
Leland	Beck	beck	cs.sdsu.edu
Roger	Whitney	whitney	cs.sdsu.edu
Roger	Whitney	whitney	rohan.sdsu.edu

```
select
        first_name, last_name, user_name, host
from
        people, email_addresses
where
        people.id = email_addresses.person_id;
```

```
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

first_name	last_name	user_name	host
Leland	Beck	beck	cs.sdsu.edu
Roger	Whitney	whitney	cs.sdsu.edu
Roger	Whitney	whitney	rohan.sdsu.edu
Carl	Eckberg		

```
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

first_name	last_name	user_name	host
Leland	Beck	beck	cs.sdsu.edu
Roger	Whitney	whitney	cs.sdsu.edu
Roger	Whitney	whitney	rohan.sdsu.edu
		foo	rohan.sdsu.edu

```
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

Name	OfficeHour1	OfficeHour2	OfficeHour3
Whitney	10:00-11:00 W	17:00-18:30 Tu	15:00-16:00 Fri
Beck	8:00-12:00 M		
Anantha	9:00-10:30 Tu	9:00-10:30 Thu	

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

name	faculty_id
Whitney	1
Beck	2
Anantha	3

Office Hours

start_time	end_time	day	faculty_id	office_hour_id
10:00	11:00	Wed	1	1
8:00	12:00	Mon	2	2
17:00	18:30	Tue	1	3
9:00	10:30	Tue	3	4
9:00	10:30	Thu	3	5
15:00	16:00	Fri	1	6

Second Normal Form (2NF)

cd_title	artist	music_type	cd_id
Songs from the Trilogy	Glass	Modern Classical	1
I Stoten	Falu Spelmanslag	Swedish	2
Photographer	Glass	Modern Classical	3

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

Example 2

Name	Time	Days	Term	Schedule Number
CS635	1700-1815	MW	Spring01	09461
CS651	1700-1815	MW	Spring01	09472
CS672	1700-1815	MW	Spring01	09483
CS683	1830-1945	MW	Spring01	09494
CS696	1530-1645	MW	Spring01	09505
CS696	1830-1945	MW	Spring01	09516
CS696	1530-1645	TTh	Spring01	09520

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

course_id	time_id	term_id	schedule_number
1	1	2	09461
2	1	2	09472
3	1	2	09483
4	2	2	09494

Term

semester	year	term_id
Fall	2000	1
Spring	2001	2

Courses

course	title	name_id
CS635	Adv Obj Orient Dsgn Prog	1
CS651	Adv Multimedia Systems	2
CS683	Emerging Technologies	3
CS696	Writing Device Drivers	4

Time

start_time	end_time	days	time_id
17:00:00	18:15:00	MW	1
18:30:00	19:45:00	MW	2
15:30:00	16:45:00	MW	3
15:30:00	16:45:00	TTh	4
Etc.			

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

Name	Address	City	State Name	State abbreviation	zip	id

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

Java – Connecting To Database

```
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

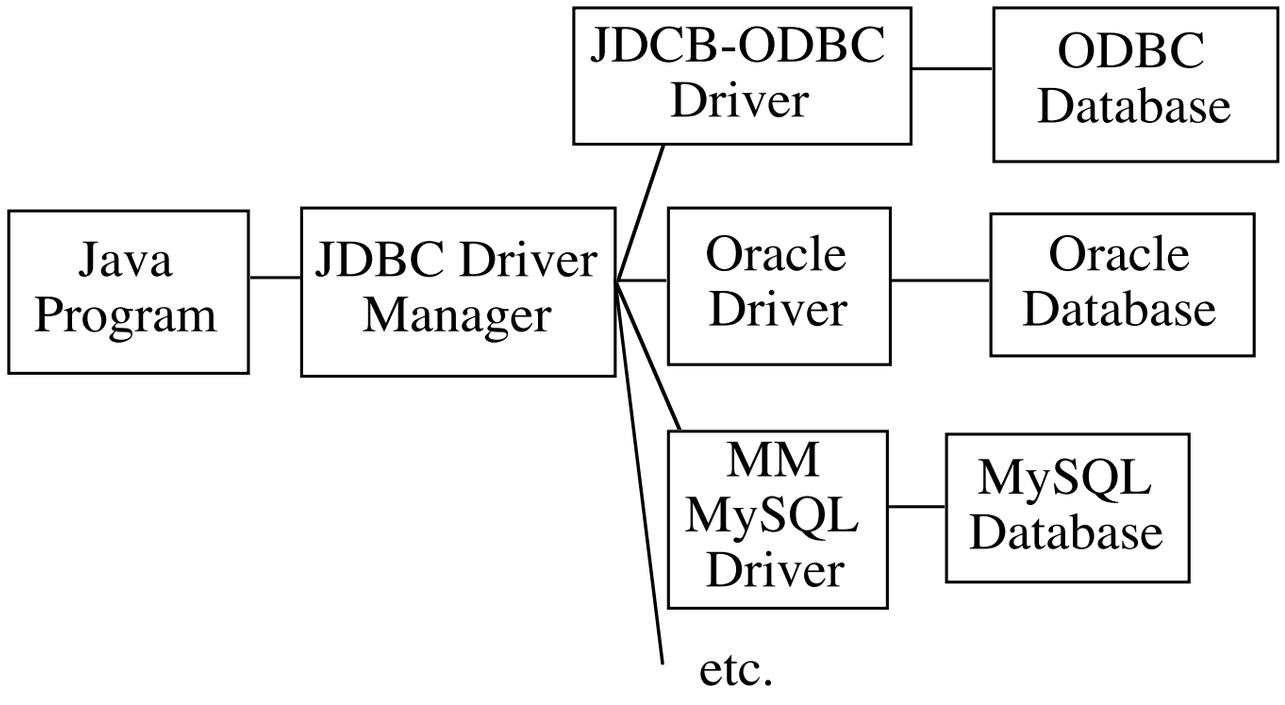
MySQL

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

PostgreSQL

http://www.postgresql.org/docs/

JDBC



MySQL jdbc driver

http://dev.mysql.com/downloads/connector/j/5.0.html

PostgreSQL jdbc driver

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

Drivers must be in your classpath

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>:<subname>

<subprotocol>

Name of the driver or database connectivity mechanism

<subname>

Depends on the <subprotocol>, can vary with vender

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

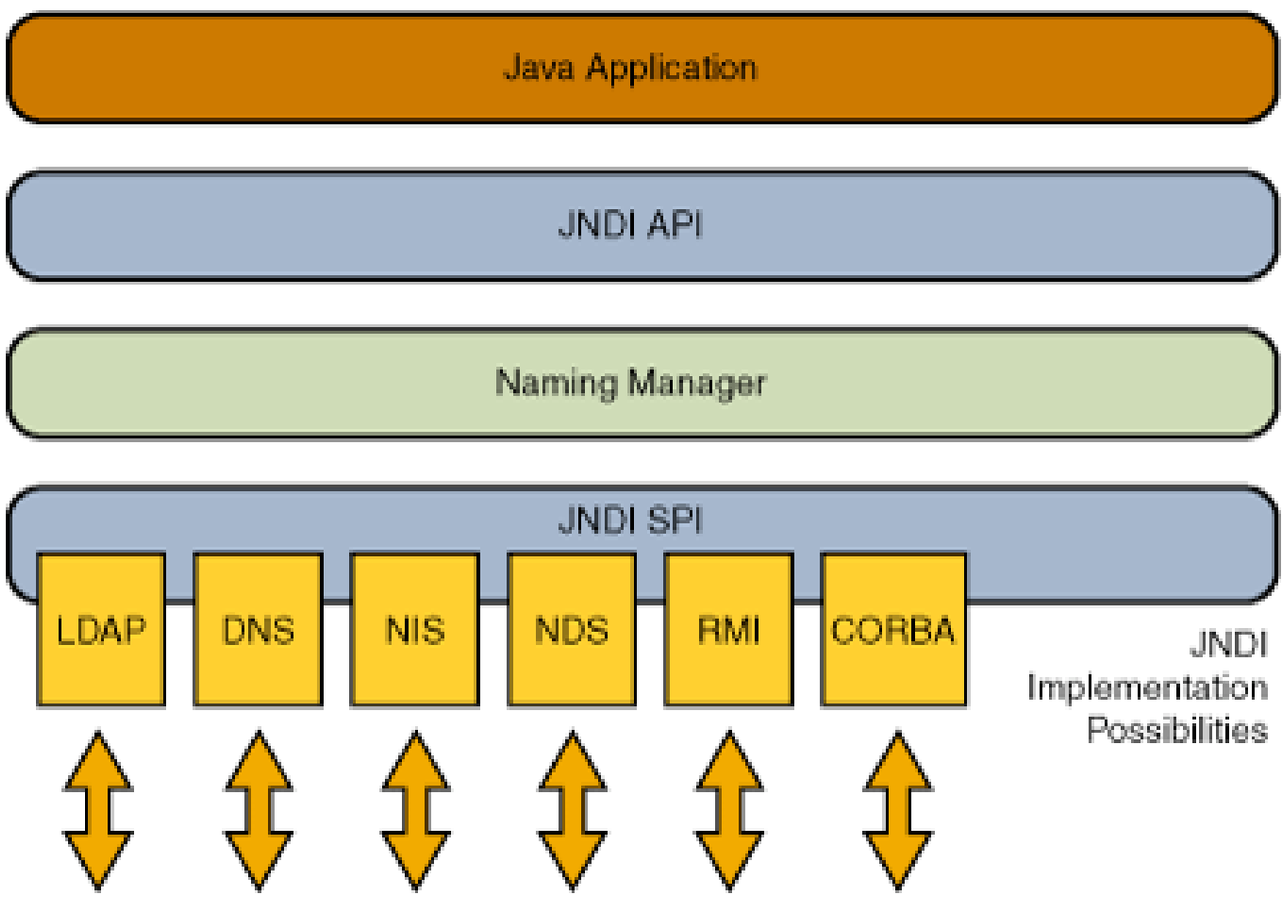
java.sql
DriverManager

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

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

Can use

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

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

getObject(String columnName) returns the object in the named column

Data Conversion

SQL type	Java type
CHAR	String
VARCHAR	String
LONGVARCHAR	String
NUMERIC	java.math.BigDecimal
DECIMAL	java.math.BigDecimal
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT	double
DOUBLE	double
BINARY	byte[]
VARBINARY	byte[]
LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp

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

```
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();
```

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"));
                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 it 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

Ruby PostgreSQL Example

```
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
```