

Servlet DB Connection

Qiang Yang
Simon Fraser University

Thanks: Alon Halevy, UW Seattle

1/29/01 DB Connection 1

JDBC

- Java Database Connectivity
 - Conforms to ANSI SQL 1992 with
 - Select, update, insert and delete, etc
 - Portable (for different db systems)
- JDBC 2.0
 - Need driver
 - Package: javax.sql
 - DriverManager, Connection, Statement, ResultSet
- Steps
 - Load db driver;
 - Get connection 3.
 - SQL statement 4.
 - Process ResultSet (row-based)

1/29/01 DB Connection 2

Steps 1 and 2. Driver and Connection

- Load


```
Class.forName("oracle.jdbc.driver.OracleDriver")
```
- Connection


```
String driverName="Oracle.jdbc.driver.OracleDriver";
String Url = "jdbc:oracle:thin:<servername>:<port>:<dbname>";
String user "qyang";
String passwd <password>;

Class.forName(driverName);
Connection myConn= DriverManager.getConnection(URL, user, passwd)
```

1/29/01 DB Connection 3

Steps 3 and 4: get the result

- SQL queries


```
Statement myStmt = myConn.createStatement();
ResultSet rs = myStmt.executeQuery("Select * From <tablename>");
```
- Process Result in Row Order
 - Cursor: a pointer at the current row
 - Initially before the first row


```
While (rs.next()) {
    System.out.println(rs.getString(2)); print value of second column
}
```
- Exceptions
 - SQLException and ClassNotFoundException (for driver)

1/29/01 DB Connection 4

Processing Result Set

- next() method of ResultSet
 - Moves cursor forward one row in result set
 - Returns false when no more rows to move

Initial cursor →

1/29/01 DB Connection 5

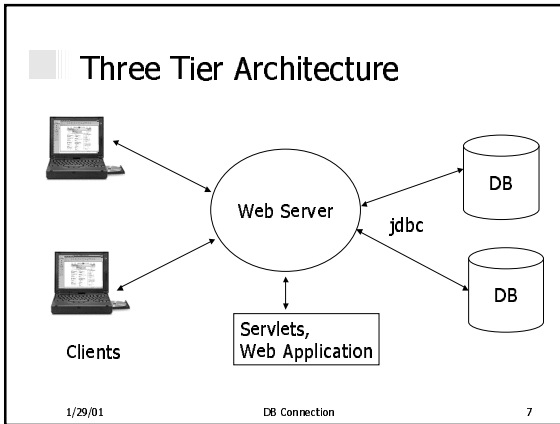
Processing Resultset in a loop

StudentID	StudentName
001	Mary

```
Statement myStmt = myConn.createStatement();
ResultSet myRs = myStmt.executeQuery(
    "Select * From StudentTable");

// print out a list of student names
While (myRs.next()) {
    out.println("<h3>" + "Student Name" + "</h3><p>");
    out.println(myRs.getString("StudentName"));
}
myRs.close();
myStmt.close();
```

1/29/01 DB Connection 6



- ### An Example of a 3-tier Application (Chap29.7 of DDN Bok)
- Application: user fills out a GuestBook
 - Front End: guestbook.html
 - HTML Form
 - Middle Tier: GuestBookServlet.class
 - Connects to a database GuestBook.mdb
 - Process form data
 - Inserts into Database
 - Backend: Microsoft Access DB
- 1/29/01 DB Connection 8

SQL Introduction

Standard language for querying and manipulating data

Structured Query Language

Many standards out there: SQL92, SQL2, SQL3.
Vendors support various subsets of these, but all of what we'll be talking about.

Basic form: (many many more bells and whistles in addition)

```
Select attributes
From relations (possibly multiple, joined)
Where conditions (selections)
```

1/29/01 DB Connection 9

- ### SQL Outline
- select-project-join
 - attribute referencing, select distinct
 - nested queries
 - grouping and aggregation
 - updates
 - view
- 1/29/01 DB Connection 10

Selections

```
SELECT *
FROM Company
WHERE country="USA" AND stockPrice > 50
```

You can use:

- attribute names of the relation(s) used in the FROM.
- comparison operators: =, <>, <, >, <=, >=
- apply arithmetic operations: stockprice*2
- operations on strings (e.g., "||" for concatenation).
- Lexicographic order on strings.
- Pattern matching: s LIKE p
- Special stuff for comparing dates and times.

1/29/01 DB Connection 11

Projections and Ordering Results

Select only a subset of the attributes

```
SELECT name, stockprice
FROM Company
WHERE country="USA" AND stockprice > 50
```

Rename the attributes in the resulting table

```
SELECT name AS company, stockprice AS price
FROM Company
WHERE country="USA" AND stockprice > 50
ORDERBY country, name
```

1/29/01 DB Connection 12

Joins

```
SELECT name, store
FROM Person, Purchase
WHERE name=buyer
      AND city="Seattle"
      AND product="gizmo"
```

Product (name, price, category, maker)
Purchase (buyer, seller, store, product)
Company (name, stock price, country)
Person(name, phone number, city)

1/29/01

DB Connection

13

Disambiguating Attributes

Find names of people buying telephony products:

```
SELECT Person.name
FROM Person, Purchase, Product
WHERE Person.name=buyer
      AND product=Product.name
      AND Product.category="telephony"
```

Product (name, price, category, maker)
Purchase (buyer, seller, store, product)
Person(name, phone number, city)

1/29/01

DB Connection

14

Tuple Variables

Find pairs of companies making products in the same category

```
SELECT product1.maker, product2.maker
FROM Product AS product1, Product AS product2
WHERE product1.category=product2.category
      AND product1.maker <> product2.maker
```

Product (name, price, category, maker)

1/29/01

DB Connection

15

Union, Intersection, Difference

```
(SELECT name
FROM Person
WHERE City="Seattle")
```

UNION

```
(SELECT name
FROM Person, Purchase
WHERE buyer=name AND store="The Bon")
```

Similarly, you can use INTERSECT and EXCEPT.
You must have the same attribute names (otherwise: rename).

1/29/01

DB Connection

16

Subqueries

```
SELECT Purchase.product
FROM Purchase
WHERE buyer =
      (SELECT name
FROM Person
WHERE social-security-number = "123 - 45 - 6789");
```

In this case, the subquery returns one value.

If it returns more, it's a run-time error.

1/29/01

DB Connection

17

Subqueries Returning Relations

Find companies who manufacture products bought by Joe Blow.

```
SELECT Company.name
FROM Company, Product
WHERE Company.name=maker
      AND Product.name IN
      (SELECT product
FROM Purchase
WHERE buyer = "Joe Blow");
```

You can also use: s > ALL R
s > ANY R
EXISTS R

1/29/01

DB Connection

18

Correlated Queries

Find movies whose title appears more than once.

```
SELECT title
FROM Movie AS Old
WHERE year < ANY
      (SELECT year
       FROM Movie
       WHERE title = Old.title);
```

Movie (title, year, director, length)
Movie titles are not unique (titles may reappear in a later year).

Note scope of variables

1/29/01

DB Connection

19

Removing Duplicates

```
SELECT DISTINCT Company.name
FROM Company, Product
WHERE Company.name=maker
      AND (Product.name,price) IN
      (SELECT product, price)
FROM Purchase
WHERE buyer = "Joe Blow");
```

1/29/01

DB Connection

20

Conserving Duplicates

The UNION, INTERSECTION and EXCEPT operators operate as sets, not bags.

```
(SELECT name
FROM Person
WHERE City="Seattle")
```

UNION ALL

```
(SELECT name
FROM Person, Purchase
WHERE buyer=name AND store="The Bon")
```

1/29/01

DB Connection

21

Aggregation

```
SELECT Sum(price)
FROM Product
WHERE manufacturer="Toyota"
```

SQL supports several aggregation operations:

SUM, MIN, MAX, AVG, COUNT

Except COUNT, all aggregations apply to a single attribute

```
SELECT Count(*)
FROM Purchase
```

1/29/01

DB Connection

22

Grouping and Aggregation

Usually, we want aggregations on certain parts of the relation.

Find how much we sold of every product

```
SELECT product, Sum(price)
FROM Product, Purchase
WHERE Product.name = Purchase.product
GROUPBY Product.name
```

1. Compute the relation (I.e., the FROM and WHERE).
2. Group by the attributes in the GROUPBY
3. Select one tuple for every group (and apply aggregation)

SELECT can have (1) grouped attributes or (2) aggregates.

1/29/01

DB Connection

23

HAVING Clause

Same query, except that we consider only products that had at least 100 buyers.

```
SELECT product, Sum(price)
FROM Product, Purchase
WHERE Product.name = Purchase.product
GROUPBY Product.name
HAVING Count(buyer) > 100
```

HAVING clause contains conditions on aggregates.

1/29/01

DB Connection

24

Modifying the Database

We have 3 kinds of modifications: insertion, deletion, update.

Insertion: general form --

```
INSERT INTO R(A1,..., An) VALUES (v1,..., vn)
```

Insert a new purchase to the database:

```
INSERT INTO Purchase(buyer, seller, product, store)
VALUES (Joe, Fred, wakeup-clock-espreso-machine,
"The Sharper Image")
```

If we don't provide all the attributes of R, they will be filled with NULL.

We can drop the attribute names if we're providing all of them in order.

1/29/01

DB Connection

25

Data Definition in SQL

So far, SQL operations on the data.

Data definition: defining the schema.

- Create tables
- Delete tables
- Modify table schema

But first:

Define data types.

Finally: define indexes.

1/29/01

DB Connection

26

Data Types in SQL

- Character strings (fixed of varying length)
- Bit strings (fixed or varying length)
- Integer (SHORTINT)
- Floating point
- Dates and times

Domains will be used in table declarations.

To reuse domains:

```
CREATE DOMAIN address AS VARCHAR(55)
```

1/29/01

DB Connection

27

Creating Tables

```
CREATE TABLE Person(
```

```
    name          VARCHAR(30),
    social-security-number INTEGER,
    age           SHORTINT,
    city          VARCHAR(30),
    gender        BIT(1),
    Birthdate     DATE
```

```
);
```

1/29/01

DB Connection

28

Creating Indexes

```
CREATE INDEX ssnIndex ON Person(social-security-number)
```

Indexes can be created on more than one attribute:

```
CREATE INDEX doubleindex ON
    Person (name, social-security-number)
```

Why not create indexes on everything?

1/29/01

DB Connection

29

Defining Views

Views are relations, except that they are not physically stored.

They are used mostly in order to simplify complex queries and to define conceptually different views of the database to different classes of users.

View: purchases of telephony products:

```
CREATE VIEW telephony-purchases AS
SELECT product, buyer, seller, store
FROM Purchase, Product
WHERE Purchase.product = Product.name
AND Product.category = "telephony"
```

1/29/01

DB Connection

30

A Different View

```
CREATE VIEW Seattle-view AS
```

```
SELECT buyer, seller, product, store
FROM Person, Purchase
WHERE Person.city = "Seattle" AND
      Person.name = Purchase.buyer
```

We can later use the views:

```
SELECT name, store
FROM Seattle-view, Product
WHERE Seattle-view.product = Product.name AND
      Product.category = "shoes"
```

What's really happening when we query a view??

1/29/01

DB Connection

31

Updating Views

How can I insert a tuple into a table that doesn't exist?

```
CREATE VIEW bon-purchase AS
SELECT store, seller, product
FROM Purchase
WHERE store = "The Bon Marche"
```

If we make the following insertion:

```
INSERT INTO bon-purchase
VALUES ("the Bon Marche", Joe, "Denby Mug")
```

We can simply add a tuple
("the Bon Marche", Joe, NULL, "Denby Mug")
to relation Purchase.

1/29/01

DB Connection

32