Statements

Database Connectivity and Access

1

Executing an SQL Statement

- Consider a basic SQL query like:

```
SELECT firstName, lastName FROM people;
```

- How is this executed in code?

2

Considerations

- Varies by implementation of DBMS
- Varies by language
- Some fundamental commonalities:
  - Association with a Connection
  - Passing of an SQL statement (usually)
  - Generation of results
  - Processing the results

3

General Steps

- Association with a Connection
- Passing of an SQL statement (usually)
- Generation of results
- Processing the results

4

Association with Connection

- How this is accomplished varies with language
  - Might create Statement object with a Connection method
  - Might instantiate Statement object with a constructor and then pass in a Connection object
  - Might not use objects at all

5

General Steps

- Association with a Connection
- Passing of an SQL statement (usually)
- Generation of results
- Processing the results

6

Passing the SQL Statement

- Usually just a parameter for the Statement object
  - Might be passed when creating the Statement object
  - Might be assigned with a mutator
- Some advanced approaches provide additional options for this step

7

General Steps

- Association with a Connection
- Passing of an SQL statement (usually)
- Generation of results
- Processing the results

8

Statements to Execute…

- How this is accomplished varies
  - C# .NET
    - ExecuteReader() – SELECT
    - ExecuteNonQuery() – UPDATE, INSERT, DELETE (DDL) CREATE, ALTER, DROP, TRUNCATE, RENAME
  - Java
    - executeQuery() – SELECT
    - executeUpdate() – UPDATE, INSERT, DELETE (DML) (DDL) CREATE, ALTER, DROP, TRUNCATE, RENAME

9

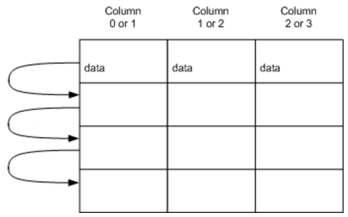
Generation of Results

- May simply result in an integer value
  - For example, UPDATE and DELETE statements
- May be a special 2-dimensional structure called a ResultSet or RecordSet
  - For SELECT statements
  - Nomenclature depends on language

10

The Result

- A fancy two-dimensional array
  - Columns by Rows
  - Called ResultSet (Java) or RecordSet (C#)
  - With some other functionality…
    - Read rows in sequence
    - Iterate across columns



11

General Steps

- Association with a Connection
- Passing of an SQL statement (usually)
- Generation of results
- Processing the results

12

Processing the Result

- ResultSet (or RecordSet) is unique to the Data Layer
  - Must be converted to ArrayList (or List)
- Even when we are selecting a single value from the database results are returned in a 2-d ResultSet
  - Must extract the value the ResultSet and return the scalar value

13

Concepts

- Keep in mind the three key objects/concepts:
  - Connection
  - Statement
  - Results
- All three are connected
- All three have properties that are set
  - Sometimes in the constructor
  - Sometimes with a mutator
- You need all three to execute a statement

14

- But how do we know where we are in the result?

  Cursors, but not the mouse or keyboard kind!

15

Cursor (**CUR**rent **S**et **O**f **R**ow)

- Because data are non-procedural
- Like an iterator over the 2D array
- Maintain current position, manage resources
  - And perhaps provide additional functionality…

16

Cursor Types

- Forward-only
- Scrollable
  - Insensitive
  - Sensitive

17

Forward-only Cursors

- Also called non-scrollable, "firehose"
- Can only traverse result in one direction
- Considerations
  - For small, unlikely to change datasets
  - Very fast
  - Not updateable

18

Scrollable Cursors

- Traversable in either direction
- Sensitive and insensitive
  - Will the cursor detect external data changes?
- Considerations
  - For data that may change while in use (sensitive)
  - Can be updated (sensitive)
  - More overhead than forward-only (both)

19

Data Sensitivity

- Static Changes are not reflected in the cursor
- Keyset Can see changes to existing cursor rows
- Dynamic Changes are reflected in the cursor

20

Varied Cursor Implementation

- Not all cursor features may be available!
- Vary depending on
  - Abstraction API completeness, bugs
  - Driver-layer implementation
  - DBMS support
- "Firehose" is the default

21

Exploration of APIs

- Java
- C#
- PHP

22

Java API

```
Statement stmnt = conn.createStatement();
                THEN
int rc = stmnt.executeUpdate("UPDATE…");
                OR
ResultSet rs = stmnt.executeQuery("SELECT…");

int row = 0;
while (rs.next()){    // Get next row
  for (int i=1; i<=numCols; i++) { // each column
        ary[row][i-1] = rs.getString(i);
  }
  row++;
}
```


23

Getting one value from 'rs'?

- What to do when we only want one value?

```
String stmt = "SELECT AVG(price) from
     sometable";
ResultSet rs=stmt.executeQuery(stmt);
String value = "No data";
if(rs.next()) {  // Any data?
     value = rs.getString(1);
}
```


24

C# API

```
MySqlCommand cmd=new MySqlCommand(sql,conn);
                    THEN
int rc = cmd.ExecuteNonQuery("UPDATE…");
                    OR
MySqlDataReader rdr = cmd.ExecuteReader("SELECT…");

int row = 0;
while (rdr.Read()){    // Get next row
  for (int i=0; i<rdr.FieldCount; i++) { // Col
     ary[row][i] = rdr.getValue(i).ToString();
  }
  row++;
}
```


25

PHP API - mysqli

```
$rs = $conn->query($sql);
                    THEN  (for UPDATE, INSERT, DELETE)
if ($rs > 0) {
 echo 'Changed ' . $conn->affected_rows . ' rows';
}
                    OR
if ($rs->$mysqli->num_rows > 0)
   while( $eachRow = $rs->fetch_assoc() ){
     foreach( $eachRow as $field) {
         echo $field;        //
     }
     echo '---- new row ----';
   }
}
```


26

PHP API - PDO

```
$rs = $conn->query($sql);
                    THEN

if ($rs > 0) …
                    OR
while($r = $rs->fetch(PDO::FETCH_ASSOC)){
  echo $r['some column'];
}
```


27

PHP – PDO vs MySQLi

	PDO	MySQLi
Database support	12 different drivers	MySQL only
API	OOP	OOP + procedural
Connection	Easy	Easy
Named parameters	Yes	No
Object mapping	Yes	Yes
Prepared statements (client side)	Yes	No
Performance	Fast	Fast
Stored procedures	Yes	Yes

https://code.tutsplus.com/tutorials/pdo-vs-mysqli-which-should-you-use--net-24059

28

Data Layer Patterns

- Row Abstraction
  - Attributes mirror table columns
  - Default constructor, id constructor, full parameter constructor
  - Fetch, Post, Put, Delete methods
  - Child manipulation methods possibly
- Table Abstraction
  - Use a Collection object of type row object
  - Both default and parameterized constructors
  - Fetch routine
    - Accept limits
    - Populate collection or return 2-d list

29