Database Application Development

SQL Integration Approaches

- Create special API to call SQL commands
  - API = application programming interface
  - JDBC, PHP
- Embed SQL in the host language = extend language
  - Embedded SQL, SQLJ
- Move (part of) application code into database
  - Stored procedures, object-relational extensions, …

Overview

- SQL API
  - Example 1: PHP
  - Example 2: JDBC
- Embedded SQL
  - Basics; Cursors; Dynamic SQL – based on Example 1: C
  - Example 2: SQLJ
- Stored procedures

DB APIs

- Library with database calls
  - Pass SQL string from language
  - present results in language-friendly way
- Representatives:
  - PHP: “Private Home Page” -> “PHP Hypertext Processor”
  - JDBC: Java SQL API (Sun Microsystems)
    - *cf. ODBC by Microsoft*

PHP and (My)SQL

www.php.net

- PHP calls embedded within HTML as special tag
  - <?php *php-statement-sequence* ?>
- Execution (server-side!) of PHP statements generates text which substitutes PHP code snippets; all then is forwarded by Web server:

```
<h1><?php echo "Hello World"; ?></h1>
```

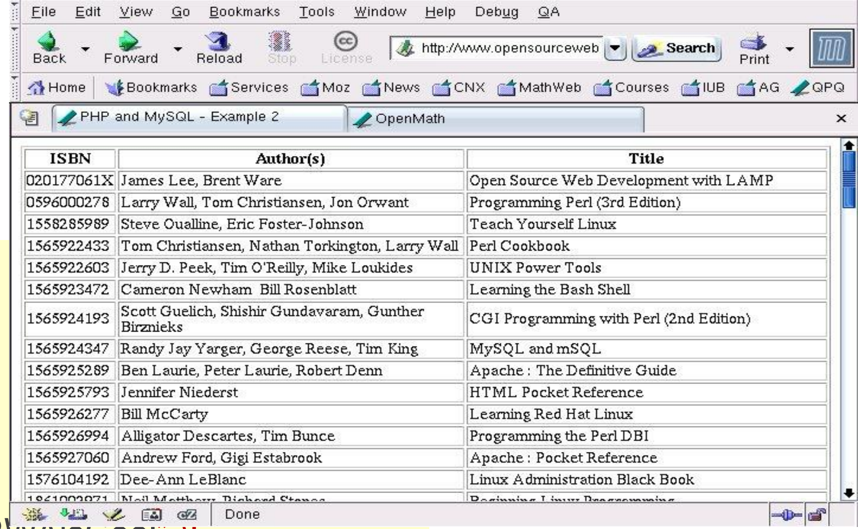
$\longrightarrow$

```
<h1>Hello World</h1>
```

- Example: connecting to mysql server on localhost

```
<?php
   $mysql = mysql_connect( "localhost", "apache", "DBWAisCool" )
      or die( "cannot connect to mysql" );
?>
```

PHP, HTML, and (My)SQL


File Edit View Go Bookmarks Tools Window Help Debug QA
Back Forward Reload Stop License
http://www.opensourceweb
Search Print
Home Bookmarks Services Moz News CNX MathWeb Courses IUB AG QPQ
PHP and MySQL - Example 2
OpenMath
Done

ISBN	Author(s)	Title
020177061X	James Lee, Brent Ware	Open Source Web Development with LAMP
0596000278	Larry Wall, Tom Christiansen, Jon Orwant	Programming Perl (3rd Edition)
1558285989	Steve Oualline, Eric Foster-Johnson	Teach Yourself Linux
1565922433	Tom Christiansen, Nathan Torkington, Larry Wall	Perl Cookbook
1565922603	Jerry D. Peek, Tim O'Reilly, Mike Loukides	UNIX Power Tools
1565923472	Cameron Newham Bill Rosenblatt	Learning the Bash Shell
1565924193	Scott Guelich, Shishir Gundavaram, Gunther Birznieks	CGI Programming with Perl (2nd Edition)
1565924347	Randy Jay Yarger, George Reese, Tim King	MySQL and mSQL
1565925289	Ben Laurie, Peter Laurie, Robert Denn	Apache : The Definitive Guide
1565925793	Jennifer Niederst	HTML Pocket Reference
1565926277	Bill McCarty	Learning Red Hat Linux
1565926994	Alligator Descartes, Tim Bunce	Programming the Perl DBI
1565927060	Andrew Ford, Gigi Estabrook	Apache : Pocket Reference
1576104192	Dee-Ann LeBlanc	Linux Administration Black Book
1861002971	Neil Matthew, Richard Stones	Beginning Linux Programming

```
<html>
  <head>
    <title>PHP and MySQL Example</title>
  </head>
  <body>
    <?php $mysql = mysql_connect( “localhost”, “apache”, “DBWAIsCool” );
      $result = mysql_db_query( “books”, “SELECT isbn, author, title FROM book_info
                                        WHERE author=‘" . $_GET("author") . "’ " )
        or die( “query failed - “ . mysql_errno() . “: " . mysql_error(); )
    ?>
    <table>
      <tr>   <th>ISBN</th>  <th>Author(s)</th>   <th>Title</th>   </tr>
      <?php while ( $array = mysql_fetch_array($result) ); ?>
      <tr><td><?php echo $array[ "isbn"     ]; ?></td>
          <td><?php echo $array[ "author" ]; ?></td>
          <td><?php echo $array[ "title"     ]; ?></td>
      </tr>
      <?php endwhile; ?>
     </table>
     <?php mysql_close($mysql); ?>
  </body>
</html>
```

open

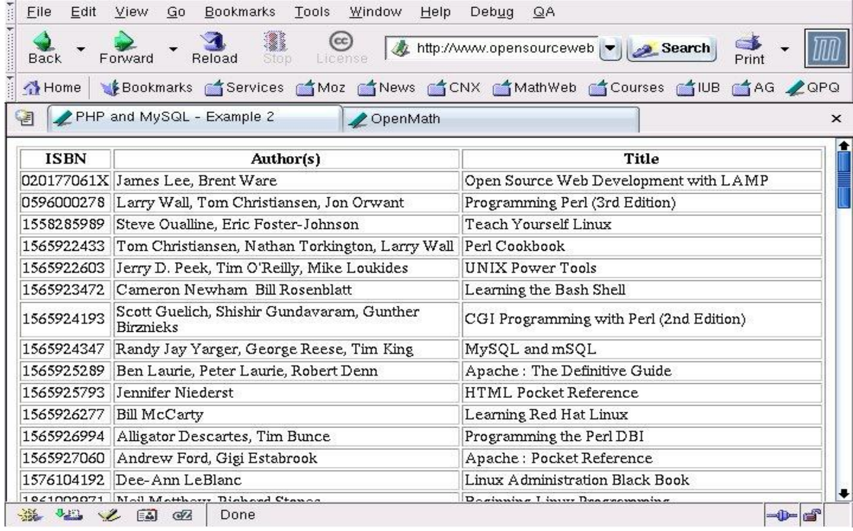
query

iterate over result set

close

Python and (My)SQL

- Different approach
  - Python code is prime, not HTML
  - Flask for invocation from Web server



File Edit View Go Bookmarks Tools Window Help Debug QA
Back Forward Reload Stop License
http://www.opensourceweb
Search Print
Home Bookmarks Services Moz News CNX MathWeb Courses IUB AG QPQ
PHP and MySQL - Example 2
OpenMath
Done

ISBN	Author(s)	Title
020177061X	James Lee, Brent Ware	Open Source Web Development with LAMP
0596000278	Larry Wall, Tom Christiansen, Jon Orwant	Programming Perl (3rd Edition)
1558285989	Steve Oualline, Eric Foster-Johnson	Teach Yourself Linux
1565922433	Tom Christiansen, Nathan Torkington, Larry Wall	Perl Cookbook
1565922603	Jerry D. Peek, Tim O'Reilly, Mike Loukides	UNIX Power Tools
1565923472	Cameron Newham Bill Rosenblatt	Learning the Bash Shell
1565924193	Scott Guelich, Shishir Gundavaram, Gunther Birznieks	CGI Programming with Perl (2nd Edition)
1565924347	Randy Jay Yarger, George Reese, Tim King	MySQL and mSQL
1565925289	Ben Laurie, Peter Laurie, Robert Denn	Apache : The Definitive Guide
1565925793	Jennifer Niederst	HTML Pocket Reference
1565926277	Bill McCarty	Learning Red Hat Linux
1565926994	Alligator Descartes, Tim Bunce	Programming the Perl DBI
1565927060	Andrew Ford, Gigi Estabrook	Apache : Pocket Reference
1576104192	Dee-Ann LeBlanc	Linux Administration Black Book

- Python code example (schematic):

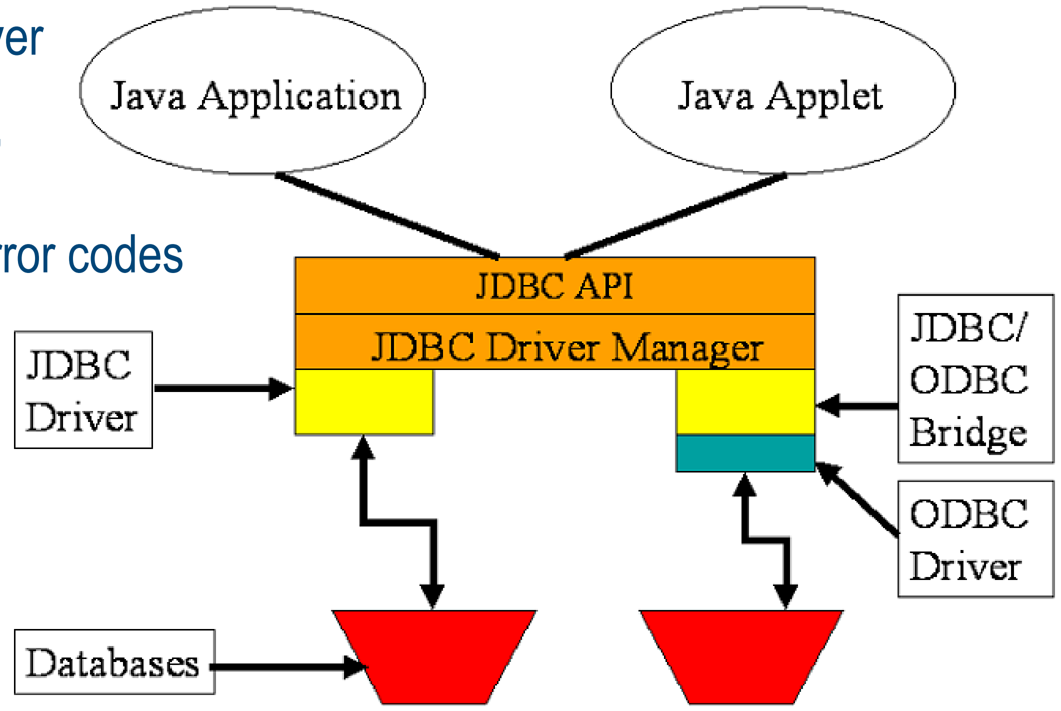
```
from flask import Flask, request, render_template
import mysql.connector
import sys
app = Flask( __name__ )
@app.route( '/booklist', method=[ 'GET‘ ] )
def booklist() :
    connection = mysql.connector.connect( … )
    cursor = connection.cursor()
    cursor.execute( "SELECT isbn, author, title FROM book_info WHERE author=‘?’ ", author )
    result = cursor.fetchall()
    connection.close()
    return render_template( 'booklist.html', result = result )
```

Overview

- SQL API
  - Example 1: PHP
  - Example 2: JDBC
- Embedded SQL
  - Basics; Cursors; Dynamic SQL – based on Example 1: C
  - Example 2: SQLJ
- Stored procedures

JDBC: Architecture

- Four architectural components:
  - Application: initiates / terminates connections, submits SQL statements
  - Driver manager: load JDBC driver
  - Driver: connects to data source, transmits requests, returns/translates results and error codes
  - Data source: processes SQL statements



Prepared Statement: Example

```
String sql = “INSERT INTO Sailors VALUES(?,?,?,?)”;
PreparedStatement pstmt=con.prepareStatement( sql );

pstmt.clearParameters();            // reset parameter list
pstmt.setInt( 1, sid );             // set attr #1 to value of sid
pstmt.setString( 2, sname );        // set attr #2 to sname
pstmt.setInt( 3, rating );          // set attr #3 to rating
pstmt.setFloat( 4, age );           // set attr #4 to age

// INSERT belongs to the family of UPDATE operations
// (no rows are returned), thus we use executeUpdate()
int numRows = pstmt.executeUpdate();
```

- Two methods for query execution:
  - PreparedStatement.executeUpdate() returns *number* of affected records
  - PreparedStatement.executeQuery() returns *data*

Result Sets

- Class ResultSet (aka cursor) for returning data to application

```
ResultSet rs = pstmt.executeQuery( sql );        // rs is a cursor
while ( rs.next() )
{
    System.out.println( rs.getString(“name") + " has rating " + rs.getDouble(“rating") );
}
```

- …but a very powerful cursor:

previous()	moves one row back
absolute(int num)	moves to the row with the specified number
relative (int num)	moves forward or backward
first() and last()	moves to first or last row, resp.

Overview

- SQL API
  - Example 1: PHP
  - Example 2: JDBC
- Embedded SQL
  - Basics; Cursors; Dynamic SQL – based on Example 1: C
  - Example 2: SQLJ
- Stored procedures

Embedded SQL

- Prefix tells preprocessor what to translate (“source-to-source”)
- Connecting to a database:
  - EXEC SQL CONNECT
- Declaring variables:
  - EXEC SQL BEGIN DECLARE SECTION
    …
    EXEC SQL END DECLARE SECTION
- Statements:
  - EXEC SQL Statement

```
EXEC SQL include sqlglobals.h;
EXEC SQL include "externs.h“

EXEC SQL BEGIN DECLARE SECTION;
        long rasver1;
        long schemaver1;
        char *myArchitecture = RASARCHITECTURE;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT ServerVersion, IFVersion
       INTO :rasver1, :schemaver1
       FROM RAS_ADMIN
       WHERE Architecture = :myArchitecture;
if (SQLCODE != SQLOK)
{      if (SQLCODE == SQLNODATAFOUND) …;
}
```

Cursors

- Problem: How to iterate over result sets
  when procedural languages do not know “sets”?
- Cursor = aka generic iterator (C++!)
  - on relation or query statement generating a result relation
- Can open cursor,
  and repeatedly fetch a tuple then move the cursor,
  until all tuples have been retrieved
- special clause ORDER BY to control order in which tuples are returned
  - Fields in ORDER BY clause must also appear in SELECT clause
- Can also modify/delete tuple pointed to by a cursor
  - …but no update of attributes mentioned in ORDER BY clause (obviously)

Names of sailors who have reserved a red boat, in alphabetical order

```
EXEC SQL DECLARE sinfo CURSOR FOR
        SELECT  S.sname
        FROM  Sailors S, Boats B, Reserves R
        WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color=‘red’
        ORDER BY  S.sname
```

- Cursor + query = 1 statement, embedded in host language

CONSTRUCTOR UNIVERSITY

Embedding SQL in C: An Example

```
long SQLCODE;
EXEC SQL BEGIN DECLARE SECTION
        char c_sname[20]; short c_minrating; float c_age;
EXEC SQL END DECLARE SECTION
c_minrating = random();         /* just for fun */

EXEC SQL DECLARE sinfo CURSOR FOR
        SELECT S.sname, S.age
        FROM Sailors S
        WHERE S.rating > :c_minrating
        ORDER BY S.sname;
do
{       EXEC SQL FETCH sinfo INTO :c_sname, :c_age;
        if ( SQLCODE == 0 )
                printf("%s is %d years old\n", c_sname, c_age);
} while ( SQLCODE >= 0 );
EXEC SQL CLOSE sinfo;
```

Note ":" prefix! Precompiler needs that hint to distinguish program from SQL variables

Overview

- SQL API
  - Example 1: PHP
  - Example 2: JDBC
- Embedded SQL
  - Basics; Cursors; Dynamic SQL – based on Example 1: C
  - Example 2: SQLJ
- Stored procedures

CONSTRUCTOR UNIVERSITY

SQLJ

- SQLJ = Java + embedded JDBC database access, nicely wrapped
  - ISO standard
  - eliminates JDBC overhead
    → compact & elegant database code, less programming errors
- SQLJ program ----[SQLJ translator]----> std Java source code
  - embedded SQL statements → calls to SQLJ runtime library
- (semi-) static query model: Compiler does
  - syntax checks, strong type checks
  - consistency wrt. schema
- Primer: http://archive.devx.com/dbzone/articles/sqlj/sqlj02/sqlj012102.asp

SQLJ Code Example

```
Int sid; String name; Int rating;
#sql iterator Sailors( Int sid, String name, Int rating );
Sailors sailors;

#sql sailors =
   { SELECT sid, sname INTO :sid, :name FROM Sailors WHERE rating = :rating };

while (sailors.next())
{    System.out.println( sailors.sid + ": " + sailors.sname) );
}

sailors.close();
```

SQLJ vs. JDBC

```
String vName; int vSalary; String vJob;
Java.sql.Timestamp vDate;

...

#sql { SELECT Ename, Sal
       INTO :vName, :vSalary
       FROM Emp
       WHERE Job = :vJob and HireDate = :vDate };
```

```
String vName; int vSalary; String vJob;
Java.sql.Timestamp vDate;
...
PreparedStatement stmt =
   connection.prepareStatement(
       "SELECT Ename, Sal "      +
       "INTO :vName, :vSalary "  +
       "FROM Emp "               +
       "WHERE Job = :vJob and HireDate = :vDate");
stmt.setString(1, vJob);
stmt.setTimestamp(2, vDate);
ResultSet rs = stmt.executeQuery();
rs.next();
vName   = rs.getString(1);
vSalary = rs.getInt(2);
rs.close();
```

simplified:
no result set iteration

Overview

- SQL API
  - Example 1: PHP
  - Example 2: JDBC
- Embedded SQL
  - Basics; Cursors; Dynamic SQL – based on Example 1: C
  - Example 2: SQLJ
- Stored procedures

Stored Procedures

- What is a stored procedure?
  - Program executed through a single SQL statement
  - Executed in the process space of the server
- Advantages:
  - Can encapsulate application logic while staying “close” to the data
  - Reuse of application logic by different users
  - Avoid tuple-at-a-time return of records through cursors

SQL/PSM

- Most DBMSs allow users to write stored procedures in a simple, general-purpose language (close to SQL)
  - SQL/PSM standard: “Persistent Storage Module”
  - Other languages: see vendor manuals
- Procedural constructs: procs/functions, variables, branches, loops
  - computationally complete
- Ex: User-Defined Function (UDF) = server-side linked code:

```
CREATE PROCEDURE TopSailors( IN num INTEGER )
   LANGUAGE JAVA
   EXTERNAL NAME “file:///c:/storedProcs/rank.jar”
```

SQL/PSM Example

```
CREATE FUNCTION rateSailor (IN sailorId INTEGER) RETURNS INTEGER
        DECLARE rating INTEGER
        DECLARE numRes INTEGER

        SET numRes = (SELECT COUNT(*)
                             FROM Reserves R
                             WHERE R.sid = sailorId)

        IF (numRes > 10)
        THEN rating =1;
        ELSE rating = 0;
        END IF;

        RETURN rating;
```

Calling Stored Procedures from Client

- Embedded SQL:
  - EXEC CALL IncreaseRating(:sid, :rating);
- JDBC:
  - CallableStatement cstmt = con.prepareCall(“{call ShowSailors}”);
- SQLJ:
  - #sql showsailors = { CALL ShowSailors };

Summary: Connecting PL & DBMS

- Coupling techniques
  - API: library with DBMS calls = layer of abstraction between application and DBMS
  - Embedded SQL: extend PL with SQL statements
  - Stored procedures: execute application logic directly at the server
- Cursor mechanism for record-at-a-time traversal
  - bridge impedance mismatch
- Query flexibility
  - (parametrized) static queries, checked a compile-time
  - Dynamic SQL: ad-hoc queries within host language
- Never forget error handling!