SQL and programming languages

SET08104 Database Systems

Copyright Napier University
Pure SQL

Pure SQL: Queries typed at an SQL prompt.

- SQL is a non-procedural language.
- SQL specifies WHAT, not HOW.
- Pure SQL is good for:
  - defining database structure
  - generating low-volume, ad hoc queries
  - prototyping
- Sophisticated applications are often implemented by using SQL in combination with a programming language.
Embedded SQL

- SQL can be embedded within procedural programming languages.
- These languages include C/C++, Java, Perl, Python, and PHP.
- Embedded SQL supports:
  - Highly customised applications.
  - Background applications running without user intervention.
  - Combining database tools with programming tools.
  - Databases on the WWW.
Two types of embedding

Low-level embedding (eg. C/C++):

▶ SQL and program compiled into a single executable.
▶ Very efficient link.

ODBC - Open Database Connectivity (eg. PHP/Java):

▶ SQL query sent from the program to the database as a string.
▶ Results returned as an array or list.
▶ Independence of program and database:
  ▶ Each language has one DBI (database interface) for all DBMS types. (For example, JDBC for Java.)
  ▶ Separate database drivers (DBD) for each DBMS type.
Low-level embedding (eg. C/C++)

- Queries consist of a mixture of SQL and special commands.
- A cursor steps through the resulting rows one at a time.

For example:

```
EXEC SQL SELECT empname INTO :ename
FROM employee WHERE eno = :eno;
```
Cursors

- A pointer to the current item in a query result set.
- Starts with the first item.
- Steps through the results one at a time.
- Some cursor implementations allow to step back up as well.
ODBC database connections

- Connect to the database.
- Prepare a query (as a string).
- Execute the query.
- Fetch the results (as an array of rows).
- Finish the query (so that DB can clean up its buffers).
- Disconnect from the database.
For example: Java

- import the DBI libraries
  Class.forName("oracle.jdbc.OracleDriver")

- connect to the database
  Connection con = DriverManager.getConnection("jdbc:oracle:DatabaseName", "myLogin", "myPassword");

- Execute a query
  ResultSet rs = stmt.executeQuery("SELECT empno, surname FROM employee");

- Cursor points to the first row
  rs.next()
while (rs.next()) {
    int emp = rs.getInt("empno");
    String surn = rs.getString("surname");
    System.out.println(emp + " " + surn);
}

or

while (rs.next()) {
    int emp = rs.getInt(1);
    String surn = rs.getString(2);
    System.out.println(emp + " " + surn);
}
For example: PHP

- connect to the database
  
  ```
  $link = mysql_connect('hostname','uname', 'passwd');
  ```

- Select database
  ```
  mysql_select_db('test');
  ```

- Execute a query
  ```
  $result = mysql_query('select * from test');
  ```

- Fetch the result
  (See next slide)

- Finish the query
  ```
  mysql_free_result($result);
  ```

- Disconnect the database
  ```
  mysql_close($link);
  ```

MySQL commands might throw errors, which should be caught:

... or die('Error message ' . mysql_error());
 Fetching the result (PHP)

```php
echo "<table>
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
    echo "<tr>",
    echo "<td>" , $line['firstfield'] , "</td>" ;
    echo "<td>" , $line['secondfield'] , "</td>" ;
    echo "<td>" , $line['thirdfield'] , "</td>" ;
    echo "</tr>
}
echo "</table>";
```
Security Warning!

▶ Using MySQL and PHP on the web is a potential severe security risk.
▶ There is a lot of nonsense information about how to use MySQL with PHP on the web.
▶ It is especially dangerous to take any user input (i.e. form variables) and use them directly in an SQL query.
▶ For an experienced programmer, PHP provides a lot of support for writing secure code (but that is beyond this lecture).
▶ Inexperienced programmers should not use MySQL with PHP.
This is a statement found in a PHP forum:

“At first my remote connection to Mysql did not work, but then I discovered I only had to stop my firewall and it worked fine.”
Security Warning continued

This is what a hacker might type into a textfield written by the user on the previous slide:

\[ 0; \text{SELECT} \ * \text{ from mysql.user; } - - \]