DATABASE DESIGN I - 1DL300

Summer 2008

An introductory course on database systems

http://user.it.uu.se/~udbl/dbt-sommar08/
alt. http://www.it.uu.se/edu/course/homepage/dbastekn/st08/

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden
Database API:s

(Elmasri/Navathe ch. 9)
(Padron-McCarthy/Risch ch 20)

Kjell Orsborn

Department of Information Technology
Uppsala University, Uppsala, Sweden
Database user interfaces

• Textual interfaces
  – Such as BSQL for Mimer

• Graphical interfaces
  – Most well-known is QBE (Query-By-Example) originally developed by IBM. MS Access uses a QBE variant.

• SQL application programming interfaces
  – Requires management of sessions, sql statements and some control of query optimization.
  – Call-level interfaces
  – Embedded SQL
Call-Level Interfaces

• Vendor-specific call-level interfaces
  – An SQL API usually for one or several host languages like C, C++, Java, Fortan, COBOL etc.
  – Support to manage sessions, SQL statements and data conversions

• SQL Call Level Interface (CLI),
  – The Call Level Interface (CLI) is a standard SQL API created by The Open Group. The API is defined for C and COBOL only. ISBN: 1-85912-081-4, X/Open Document Number: C451, 1995.

• SQL/CLI
  – Call-Level Interface (SQL/CLI) is an implementation-independent CLI to access SQL databases. SQL/CLI is an ISO standard ISO/IEC 9075-3:1995 Information technology -- Database languages -- SQL -- Part 3: Call-Level Interface (SQL/CLI). The current SQL/CLI effort is adding support for SQL:1999.

• ODBC
  – (Microsoft) Open Database Connectivity is a standard SQL API. ODBC is based on the Call Level Interface (CLI) specifications from SQL, X/Open (now part of The Open Group), and the ISO/IEC. ODBC was created by the SQL Access Group and released Sept, 1992.

• JDBC - Java Database Connectivity
  – JDBC is an SQL API for Java (to be strictly correct, JDBC is not an acronym).
The ODBC architecture

- ODBC API is independent of any one programming language, database system or operating system.
The JDBC architecture

- JDBC API is independent of (relational) DBMS and operating system
Alt. JDBC architecture (JDBC-ODBC bridge)

- Makes ODBC accessible from JDBC such that no special JDBC drivers are required.
Programming with SQL CLI interfaces
JDBC example

• The JDBC API (Application Program Interface) is a set of Java interfaces that allow database applications to:
  – open connections to a database,
  – execute SQL statements, and
  – process the results.

• These include:
  – `java.sql.DriverManager`, which loads the specific drivers and supports creating new database connections
  – `java.sql.Connection`, which represents a connection to a specific database
  – `java.sql.Statement`, which allows the application to execute a SQL statement
  – `java.sql.PreparedStatement`, which represents a pre-compiled SQL statement
  – `java.sql.ResultSet`, controls access to rows resulting from executing a statement
import java.sql.*;

public class JDBCExample {

    public static void main(String args[]) {

        String url = "jdbc:mySubprotocol:myDataSource";
        Connection con;
        String query = "SELECT NAME FROM EMPLOYEE WHERE INCOME > 10000";
        Statement stmt;

        try {
            Class.forName("myDriver.ClassName");
        }catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }

        try {
            con = DriverManager.getConnection(url, "myLogin", "myPassword");
            stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            while (rs.next()) {
                String s = rs.getString("NAME");
                System.out.println(s);
            }
            rs.close();
            stmt.close();
            con.close();
        }catch(SQLException ex) {
            System.err.print("SQLException: ");
            System.err.println(ex.getMessage());
        }}}}
import java.sql.*;

public class JDBCExample {
    public static void main(String args[]) {

        String url = "jdbc:mySubprotocol:myDataSource";
        Connection con;
        String query = "SELECT NAME FROM EMPLOYEE WHERE INCOME > ?;"
        Int incomeLimit;
        PreparedStatement stmt;

        try {
            Class.forName("myDriver.ClassName");
        }catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }

        try {
            con = DriverManager.getConnection(url, "myLogin", "myPassword");
            stmt = con.prepareStatement(query);
            while(....) {
                .... // Code to read lower income limit into incomeLimit
                stmt.setInt(1,incomeLimit);
                ResultSet rs = stmt.executeQuery();
                while (rs.next()) {
                    System.out.println(rs.getString("NAME"));
                }
            rs.close();
            stmt.close();
            con.close();
        }catch(SQLException ex) {
            System.err.print("SQLException: ");
            System.err.println(ex.getMessage());
        }
    }
}
Embedded SQL

- Host language include embedded and specially marked SQL statements.
- Embedded statements are extracted by preprocessor, translated and replaced by database calls, precompiled (prepared) and stored on server.
- The preprocessed application is then compiled normally.
- Supports dynamic recompilation.
- Reduces optimization cost and can be somewhat simpler than CLI programming.