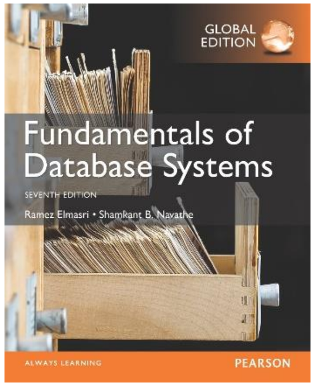
Database Systems

School of Computer Science
Chungbuk National University

Seo-Young Noh

CHAPTER 10: Introduction to SQL Programming Techniques


GLOBAL EDITION
Fundamentals of Database Systems
SEVENTH EDITION
Ramez Elmasri • Shamkant B. Navathe
ALWAYS LEARNING
PEARSON

Chapter 10 Outline

- Database Programming: Techniques and Issues
- ~~Embedded SQL, Dynamic SQL, and SQLJ~~
- Database Programming with Function Calls: SQL/CLI and JDBC
- ~~Database Stored Procedures and SQL/PSM~~
- ~~Comparing the Three Approaches~~

Overview of Database Programming Techniques and Issues

Introduction to SQL Programming Techniques

- **Database applications**
  - Host language
    - Java, C/C++/C#, COBOL, or some other programming language
  - Data sublanguage
    - SQL

- **SQL standards**
  - Continually evolving
  - Each DBMS vendor may have some variations from standard

Database Programming: Techniques and Issues

- **Interactive interface**
  - SQL commands typed directly into a monitor
- **Execute file of commands**
  - *@<filename>*
- **Application programs or database applications**
  - Used as canned transactions by the end users access a database
  - May have **Web interface**

Approaches to Database Programming

- **Embedding database commands in a general-purpose programming language**
  - Database statements identified by a special prefix
  - **Precompiler** or **preprocessor** scans the source program code
    - Identify database statements and extract them for processing by the DBMS
  - Called **embedded SQL**

Approaches to Database Programming (cont’d.)

- **Using a library of database functions**
  - **Library of functions** available to the host programming language
  - **Application programming interface (API)**
- **Designing a brand-new language**
  - **Database programming language** designed from scratch
- **First two approaches are more common**

Impedance Mismatch

- **Differences between database model and programming language model**

- **Binding for each host programming language**
  - Specifies for each attribute type the compatible programming language types

- **Cursor or iterator variable**
  - Loop over the tuples in a query result

Typical Sequence of Interaction in Database Programming

- Open a connection to database server
- Interact with database by submitting queries, updates, and other database commands
- Terminate or close connection to database

Database Programming with Function Calls and Class Libraries: JDBC

Database Programming with Function Calls: JDBC

- **Use of function calls**
  - **Dynamic** approach for database programming

- **Library of functions**
  - Also known as **application programming interface (API)**
  - Used to access database

JDBC: SQL Function Calls for Java Programming

- **JDBC**
  - Java function libraries

- **Single Java program can connect to several different databases**
  - Called data sources accessed by the Java program

- **`Class.forName("oracle.jdbc.driver.OracleDriver")`**
  - Load a **JDBC driver** explicitly

JDBC: SQL Function Calls for Java Programming

- **`Connection` object**
- **`Statement` object has two subclasses:**
  - `PreparedStatement` and `CallableStatement`
- **Question mark (`?`) symbol**
  - Represents a statement parameter
  - Determined at runtime
- **`ResultSet` object**
  - Holds results of query

Figure 10.12 Program segment JDBC1, a Java program segment with JDBC.

```
//Program JDBC1:
import java.io.* ;
import java.sql.*
...
class getEmpInfo {
  public static void main (String args []) throws SQLException, IOException {
    try { Class.forName("oracle.jdbc.driver.OracleDriver")
    } catch (ClassNotFoundException x) {
      System.out.println ("Driver could not be loaded") ;
    }
    String dbacct, passwrd, ssn, lname ;
    Double salary ;
    dbacct = readentry("Enter database account:") ;
    passwrd = readentry("Enter password:") ;
    Connection conn = DriverManager.getConnection
      ("jdbc:oracle:oci8:" + dbacct + "/" + passwrd) ;
    String stmt1 = "select Lname, Salary from EMPLOYEE where Ssn = ?" ;
    PreparedStatement p = conn.prepareStatement(stmt1) ;
    ssn = readentry("Enter a Social Security Number: ") ;
    p.clearParameters() ;
    p.setString(1, ssn) ;
    ResultSet r = p.executeQuery() ;
    while (r.next()) {
      lname = r.getString(1) ;
      salary = r.getDouble(2) ;
      system.out.printline(lname + salary) ;
  } }
}
```

Figure 10.13 Program segment JDBC2, a Java program segment that uses JDBC for a query with a collection of tuples in its result.

```
//Program Segment JDBC2:
import java.io.* ;
import java.sql.*
...
class printDepartmentEmps {
  public static void main (String args [])
      throws SQLException, IOException {
    try { Class.forName("oracle.jdbc.driver.OracleDriver")
    } catch (ClassNotFoundException x) {
      System.out.println ("Driver could not be loaded") ;
    }
    String dbacct, passwrd, lname ;
    Double salary ;
    Integer dno ;
    dbacct = readentry("Enter database account:") ;
    passwrd = readentry("Enter password:") ;
    Connection conn = DriverManager.getConnection
      ("jdbc:oracle:oci8:" + dbacct + "/" + passwrd) ;
    dno = readentry("Enter a Department Number: ") ;
    String q = "select Lname, Salary from EMPLOYEE where Dno = " +
    dno.tostring() ;
    Statement s = conn.createStatement() ;
    ResultSet r = s.executeQuery(q) ;
    while (r.next()) {
      lname = r.getString(1) ;
      salary = r.getDouble(2) ;
      system.out.printline(lname + salary) ;
  } }
}
```