Embedded SQL, Dynamic SQL and SQLJ

OUTLINES

- What is SQL
- What is Embedded SQL
- Cursors
- Dynamic SQL
- SQLJ
- Summary

WHAT IS SQL?

- Structured Query Language (SQL) is a standardized language used to manipulate database objects and the data they contain.
- It comprised of several different statements that are used manipulate data values.
- SQL being a nonprocedural is not a general-purpose programming language.

```
UPDATE EMPLOYEE SET LASTNAME = 'Jones' WHERE
EMPID = '001'
```

WHAT IS EMBEDDED SQL?

- As a result, database applications are usually developed by combining capabilities of a high-level programming language with SQL.
- The simplest approach is to embed SQL statements directly into the source code file(s) that will be used to create an application. This technique is referred to as embedded SQL programming.
- sqlca.h – header file to be included.

EMBEDDED SQL

- High-level programming language compilers cannot interpret, SQL statements.
- Hence source code files containing embedded SQL statements must be preprocessed before compiling.
- Thus each SQL statement coded in a high-level programming language source code file must be prefixed with the keywords **EXEC SQL** and terminated with either a semicolon or the keywords **END_EXEC**.

EMBEDDED SQL

- Likewise, the Database Manager cannot work directly with high-level programming language variables.
- Instead, it must use special variables known as **host variables** to move data between an application and a database.
- Two types of Host variables:-
  1. Input Host Variables – Transfer data to database
  2. Output Host Variables – receives data from database

EMBEDDED SQL

- Host variables are ordinary programming language variables.
- To be set apart, they must be defined within a special section known as a **declare section**.

```
EXEC SQL BEGIN DECLARE SECTION
char EmployeeID[7];
double Salary;
EXEC SQL END DECLARE SECTION
```

- Each host variable must be assigned a unique name even though declared in different declaration section.

EMBEDDED SQL

```
main() {
 EXEC SQL BEGIN DECLARE SECTION;
 int OrderID, CustID;
 char SalesPerson[10], Status[6];
 EXEC SQL END DECLARE SECTION;

 printf ("Enter order number: ");
 scanf ("%d", &OrderID);

 EXEC SQL SELECT CustID, SalesPerson, Status FROM
Orders WHERE OrderID = :OrderID INTO :CustID,
:SalesPerson, :Status;

 printf ("Customer number: %d \n", CustID);
 printf ("Salesperson: %s \n", SalesPerson);
 printf ("Status: %s \n", Status);
}
```

EMBEDDED SQL

Connecting to Database using embedded sql

```
EXEC SQL CONNEC  :userid IDENTIFIED BY :passwd;
         T
EXEC SQL CREATE TABLE Test (a int);
EXEC SQL INSERT INTO Test VALUES (1);
EXEC SQL SELECT MAX (a) INTO :value from R;
printf ("Max value=%d\n",value);
```

CURSOR

- CAN DECLARE A CURSOR ON A QUERY STATEMENT WHICH GENERATES A
- RELATION.
- CAN OPEN A CURSOR, REPEATEDLY FETCH A TUPLE, MOVE THE CURSOR, UNTIL ALL TUPLES HAVE BEEN RETRIEVED.
- CONTROL ORDER: ORDER BY, IN QUERIES THAT ARE ACCESSED THROUGH A CURSOR
- CAN ALSO MODIFY/DELETE TUPLE POINTED TO BY CURSOR.
- MUST CLOSE CURSOR AT END.

CURSOR

- EXEC SQL DECLARE MYCURSOR CURSOR FOR SELECT BID
- FROM RESERVATIONS;

- EXEC SQL OPEN MYCURSOR;
- EXEC SQL WHENEVER NOT FOUND DO BREAK;
- WHILE (1) {
    - EXEC SQL FETCH MYCURSOR INTO :NUM;
- }

- EXEC SQL CLOSE MYCURSOR;

DYNAMIC SQL

- Dynamic SQL means composing and executing new (not previously compiled) SQL statements at run-time.
- Although static SQL statements are relatively easy to incorporate, Dynamic SQL statements are much more flexible as they can be constructed at run time.
- Dynamic queries can be complex because the type and number of retrieved attributes are unknown at compile time.

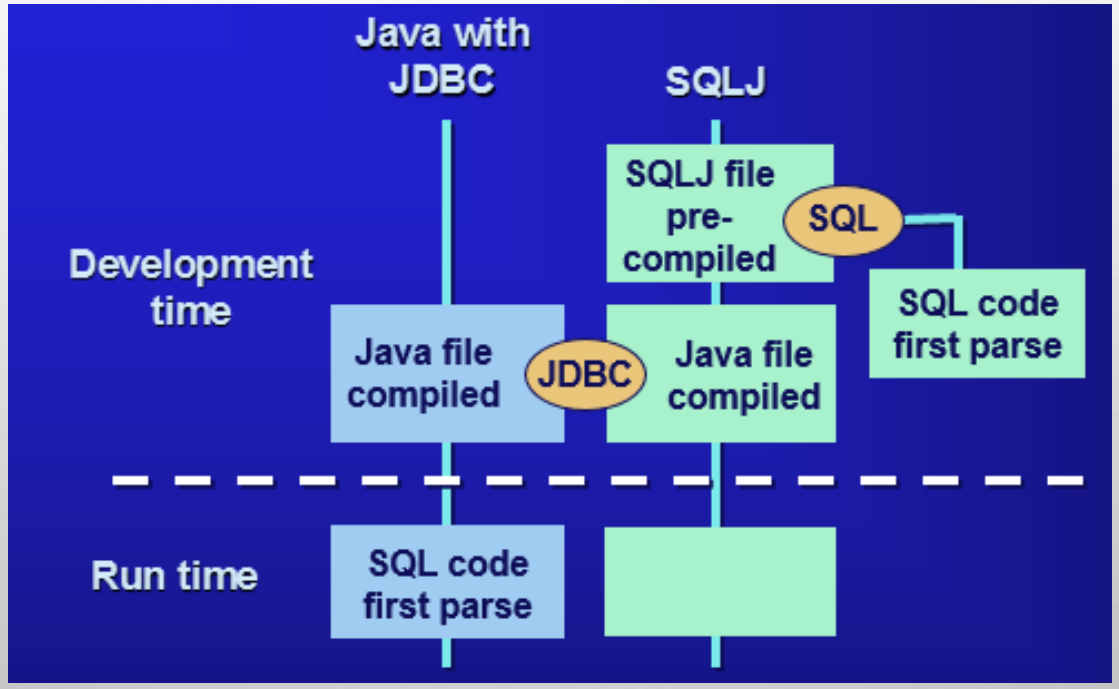
```
INSERT INTO EMPLOYEES VALUES (?, ?)
DELETE FROM DEPARTMENT WHERE DEPTID = ?
```

SQLJ

- SQLJ – Standard for embedding SQL in Java
- Similar to existing SQL extensions provided for C, FORTRAN, and other programming languages.
- IBM, Oracle, and several other companies proposed SQLJ as a standard and as a simpler and easier-to-use alternative to JDBC.
- An SQLJ translator converts SQL statements into Java
- These are executed through the JDBC interface
- Certain classes have to be imported E.g., java.sql

SQLJ

- SQLJ precompiles SQL code in a Java program.
- Provides greater compile-time checking of SQL statements.
- Reduces the amount of code needed to execute SQL from within Java.



SQLJ V/S JDBC

```
// SQLJ
int n;
#sql { INSERT INTO emp VALUES (:n)};

// JDBC
int n;
Statement stmt = conn.prepareStatement
("INSERT INTO emp VALUES (?)");
stmt.setInt(1,n);
stmt.execute ();
stmt.close();
```

SQLJ

```
import java.sql.*;
import sqlj.runtime.*;
import sqlj.runtime.ref.*;

class X {
void myJavaMethod() {
  try {
    #sql{update EMP set SAL = SAL + 100
           where SAL < 1500};
  }
  catch (SQLException e) {…}
}
```

Imports Needed

SQLJ statement begins with #sql

SQL statement placed in braces can throw SQLException

SQLJ

Loading the JDBC Driver

SQLJ requires that the JDBC driver class is loaded. This can be performed in the same way as for JDBC

```
try
{
   Class.forName("oracle.jdbc.driver.OracleDriver");
}

catch (ClassNotFoundException e)
{
  System.out.println("Could not load driver");
}
```

SQLJ

Specifying a Connection Context

- All SQLJ statements execute in a “connection context”
- Plays similar role as a Connection object does in JDBC.
- Establishes the database we are connecting to, the user name, and the password.

```
try
{
Class.forName("oracle.jdbc.driver.OracleDriver");
DefaultContext.setDefaultContext(new DefaultContext(
        "jdbc:oracle:thin:@HOSTID:1521:ORCL",
        "theUser", "thePassword") );
}
```

Passing Host Variables into a SQLJ Statement

- Prefix the java variable name with a colon (:)

```
#sql {delete from EMP where SAL >= :amt};
```

SQLJ

Dealing with Query Result Sets

- SQLJ can be used to execute queries that return a result set .
- To process the result set, define an “iterator” type that specifies the data type of each column

```
#sql iterator MyIter(String ENAME, String JOB);
class MyClass {
    MyIter iter;
    #sql iter = { select ENAME, JOB from EMP };
    while(iter.next()) {
      String ename = iter.ENAME();
      String job   = iter.JOB();
    }
```

WHEN TO WHICH ?

How do applications connect to a database?

- App ↔ DBMS: Embedded SQL
- App ↔ Driver ↔ DBMS: JDBC/ODBC or SQLJ

What mechanisms exist to retrieve/modify data?

- Static Queries: Embedded SQL, SQLJ
- Dynamic Queries: JDBC/ODBC, Dynamic SQL