Oracle PL/SQL & JDBC

JSP & Servlets

- Example:
  - http://trannew.cs.nmsu.edu:8888/examples/jsp/index.html
  - http://trannew.cs.nmsu.edu:8888/examples/servlets/index.html
- JSP for database
  - http://trannew.cs.nmsu.edu:8888/database/test.jsp
- Test your program:
  - Upload: http://www.cs.nmsu.edu/~tson/classes/fall06-482/uploadCode.html
  - Test your system: http://trannew.cs.nmsu.edu:8888/database/XXX/Code/yourMainPage
  - Let me know if you want to do this, I will need your CS account user id
- I will try to make it run on pilsner but I might not success (incompatibility between version!)

PHP

- Example:
  - http://www.cs.nmsu.edu/~tson/classes/materials/PHP/

JDBC

- *Call-level interfaces*: programming interfaces allowing external access to SQL database manipulation and update commands. (JAVA ←→ SQL/Oracle)
- *Steps*:
  - Establish the connection (JAVA←→Oracle)
  - Send commands to DB
  - Obtain results
  - Process results
  - Close the connection

Example – GpaJDBC.java

```
import java.sql.*;
import oracle.sqlj.runtime.Oracle;
import sqlj.runtime.ref.DefaultContext;

class GpaJDBC {

public static void main (String args[]) throws SQLException
  {
    // body of the main program
  }
}
```

Body of main program

```
// declaration of variables

    Connection conn=null;;
   PreparedStatement ps=null;
   ResultSet rs=null;
   double  gpa;
   int ncourses;

// set the default connection to the URL, user, and password
// specified in your connect.properties file
// create the connection

   Oracle.connect(GpaJDBC.class, "sonconnect.properties");

   conn = DefaultContext.getDefaultContext().getConnection();
```

Body of main program (create statement, …)

```
// preparation - create a statement

Statement stmt = conn.createStatement ();

// Select the grade column from the transcripts
// table – store the result in a ResultSet (an table –
// 2 dimension arrays )

ResultSet rset = stmt.executeQuery ("select grade from
    Transcripts "+ "WHERE studid = " + args[0]);
```

Body of main program (Processing …)

```
// Initialize needed values

    gpa = 0.0;
    ncourses = 0;

// Proceessing the result set (cursor)

    while (rset.next()) {
      if (rset.getString(1).compareTo("A")==0)
       {
          System.out.println("Is an A :-)");
            gpa += 4.0;
          ncourses += 1;
       }
       ……..
    }
```

Close connection

```
// Close the ResultSet
rset.close();

// Close the Statement
stmt.close();

// Close the connection
conn.close();
```

Oracle PL/SQL

Basic Structure – Block

```
DECLARE
    /* Declarative section: variables, types, and local subprograms. */
BEGIN
    /* Executable section: procedural and SQL statements. */
EXCEPTION
    /* Exception handling section: error handling statements. */
END;
.
RUN;
```

Note:

- The last two lines are needed if you store the block as a file and would like to run it by using the command @...
- Blocks can be nested
- Comments: like in C

Variables and Types

- Standard Oracle's types (integer, char(.), etc.)
- NUMBER - generic type
- Type of some database column
- Initial value of every variable is NULL
- Assignment with :=

Example:

```
DECLARE  name VARCHAR(40);
DECLARE  maxenroll  NUMBER;
DECLARE  studGrade Transcripts.Grade%Type;
DECLARE  studTuples Transcripts.%ROWTYPE;
maxenroll := 1;
```

Control Structures

IF <condition>
  THEN <statement_list>
  ELSE <statement_list>
END IF;
IF <condition_1>
  THEN ... ELSIF <condition_2>
  THEN ... ... ELSIF <condition_n>
  THEN ... ELSE ... END IF;

Control Structures

LOOP
  <loop_body> /* A list of statements. */
END LOOP;

At least one of the statements in <loop_body> should be an EXIT statement of the form

EXIT WHEN <condition>;

WHILE <condition>
  LOOP <loop_body> END LOOP;

FOR <var> IN <start>..<finish>
  LOOP <loop_body> END LOOP;

<var>: local variable
<start> and <finish> are constants

Cursor

- A variable that runs through the tuples of some relation. This relation can be a stored table, or it can be the answer to some query.

A PL/SQL Program

```
DECLARE
      gpa NUMBER := 0;
      ncourses NUMBER := 0;
      grd  Transcript.Grade%Type;    /* CHAR(1) */
      CURSOR allRecs IS SELECT T.Grade FROM Transcripts T WHERE T.studid = 111111111;
BEGIN
      OPEN allRecs;
      LOOP
       /* Retrieve each row of the result of the above query into PL/SQL variables: */
      FETCH allRecs INTO grd;
      /* If there are no more rows to fetch, exit the loop: */
      EXIT WHEN allRecs%NOTFOUND;
      IF (grd = 'A') THEN gpa := gpa + 4; ncourses := ncourses + 1; END IF;
      IF (grd = 'B') THEN gpa := gpa + 3; ncourses := ncourses + 1; END IF;
      IF (grd = 'C') THEN gpa := gpa + 2; ncourses := ncourses + 1; END IF;
      IF (grd = 'D') THEN gpa := gpa + 1; ncourses := ncourses + 1; END IF;
      END LOOP;
      CLOSE allRecs;
      IF (ncourses>0) THEN gpa := gpa / ncourses; END IF;
END;
.
RUN;
```

Procedures

CREATE PROCEDURE
  compGPA(id IN NUMBER)
AS
BEGIN
    ….. Code of previous page …
    ….. Declare part between AS and BEGIN
END compGPA;
.
run;

A Procedure

```
CREATE PROCEDURE compGPA(id IN NUMBER)  AS
        gpa NUMBER := 0;
        ncourses NUMBER := 0;
        grd  Transcript.Grade%Type;    /* CHAR(1) */
        CURSOR allRecs IS SELECT T.Grade FROM Transcripts T WHERE T.studid = id;
BEGIN
        OPEN allRecs;
        LOOP
         /* Retrieve each row of the result of the above query into PL/SQL variables: */
        FETCH allRecs INTO grd;
        /* If there are no more rows to fetch, exit the loop: */
        EXIT WHEN allRecs%NOTFOUND;
        IF (grd = 'A') THEN gpa := gpa + 4; ncourses := ncourses + 1; END IF;
        IF (grd = 'B') THEN gpa := gpa + 3; ncourses := ncourses + 1; END IF;
        IF (grd = 'C') THEN gpa := gpa + 2; ncourses := ncourses + 1; END IF;
        IF (grd = 'D') THEN gpa := gpa + 1; ncourses := ncourses + 1; END IF;
        END LOOP;
        CLOSE allRecs;
        IF (ncourses>0) THEN gpa := gpa / ncourses; END IF;
END;
.
RUN;
```

A Procedure with Outputs

```
CREATE PROCEDURE compGPA(id IN NUMBER, gpa OUT NUMBER)  AS
        ncourses NUMBER := 0;
        grd  Transcript.Grade%Type;    /* CHAR(1) */
        CURSOR allRecs IS SELECT T.Grade FROM Transcripts T WHERE T.studid = id;
BEGIN
        gpa := 0;
        OPEN allRecs;
        LOOP
         /* Retrieve each row of the result of the above query into PL/SQL variables: */
        FETCH allRecs INTO grd;
        /* If there are no more rows to fetch, exit the loop: */
        EXIT WHEN allRecs%NOTFOUND;
        IF (grd = 'A') THEN gpa := gpa + 4; ncourses := ncourses + 1; END IF;
        IF (grd = 'B') THEN gpa := gpa + 3; ncourses := ncourses + 1; END IF;
        IF (grd = 'C') THEN gpa := gpa + 2; ncourses := ncourses + 1; END IF;
        IF (grd = 'D') THEN gpa := gpa + 1; ncourses := ncourses + 1; END IF;
        END LOOP;
        CLOSE allRecs;
        IF (ncourses>0) THEN gpa := gpa / ncourses; END IF;
END;
.
RUN;
```

A Function

```
CREATE FUNCTION compGPA(id IN NUMBER) RETURN NUMBER  AS
        ncourses NUMBER := 0;
        gpa NUMBER := 0;
        grd  Transcript.Grade%Type;    /* CHAR(1) */
        CURSOR allRecs IS SELECT T.Grade FROM Transcripts T WHERE T.studid = id;
BEGIN
        OPEN allRecs;
        LOOP
         /* Retrieve each row of the result of the above query into PL/SQL variables: */
        FETCH allRecs INTO grd;
        /* If there are no more rows to fetch, exit the loop: */
        EXIT WHEN allRecs%NOTFOUND;
        IF (grd = 'A') THEN gpa := gpa + 4; ncourses := ncourses + 1; END IF;
        IF (grd = 'B') THEN gpa := gpa + 3; ncourses := ncourses + 1; END IF;
        IF (grd = 'C') THEN gpa := gpa + 2; ncourses := ncourses + 1; END IF;
        IF (grd = 'D') THEN gpa := gpa + 1; ncourses := ncourses + 1; END IF;
        END LOOP;
        CLOSE allRecs;
        IF (ncourses>0) THEN gpa := gpa / ncourses; END IF;
        RETURN gpa;
END;
.
RUN;
```