International J. Eng. Tech 6(3): 530-538, September 2009
A publication of "G-Science Implementation & publication" website: www.gscience.net

WORKING WITH PL/SQL FOR DATABASE APPLICATION DEVELOPMENT

DIPONKAR PAUL[1]*, U. K. PRODHAN[2], YEASIR FATHAH RUMI[2] and MD. ASRAF ALI[2]

ABSTRACT

Oracle 81 is a database designed for the Internet. It provides a secure foundation for building for managing large enterprise application, lava programs and information from the Internet. The introduction of Oracle 8I (The I stands for Internet), now extends the capability and advantages of Oracle as the most powerful database software. Oracle 81 changes the way information is managed and accessed to meet the demands of the Internet age. It provides the advanced tools to manage all types of data in web sites. For the Java developers it turned out a new client side programming interface allowing them to deploy applications, which have easy access to any type of database. Java stored procedures compiled and stored in Oracle 8I database provide an efficient way to maintain business logic on the server. This makes it possible to build robust and reliable industrial application using Java. The industry leading Oracle 81 data server provides a secure foundation to build and manage large enterprise applications. With the introduction of SQLJ, Oracle 81 is designed to access and manage all our data using the style and information of the Internet. Oracle 8I is the most complete and comprehensive platform for building, deploying and managing Internet and traditional applications. This research work was conducted at the Department of Electrical and Electronic Engineering in Rajshahi University of Engineering and Technology, Rajshahi, Bangladesh during April 2003 to April 2004.

Keywords: JVM, Trigger, Class path and SQL.

INTRODUCTION

Oracle Corporation was the first company to offer a true relational DBMS (Data base management system) commercially, and has continually led innovation in the field of RDBMS (Relational database management system). The Oracle Corporation strategy of offering RDBMS that is portable, compatible and connectable resulted in a very powerful tool for users. The Oracle system uses the non-procedural structure query language (SQL) to communicate with its database kernel. Oracle is divided into two fundamental segments: Server and client. The server portion of the package includes the core database itself and it contains the background processor that keep the entire database functioning. The client portion usually includes a Graphical User Interface (GUI) that makes it easier to use. Object oriented technology is fast becoming the standard for all today's application development efforts. The swiftness towards object oriented technology has mainly been in the development of front-end or client applications. The main purpose of the object-oriented approach is to enable us to build components (Objects) that the real-life business processes. These objects become meaningful and reliable and help to enhance the development. Oracle8I enhances object technology without compromising the performance of the relational database system. Oracle BI is a hybrid database that supports both objects oriented concept and the Relational worlds. In recent months, it has energized as a powerful object oriented based RDBMS. SQL is the most powerful data retrieval language amongst the ones available today (Thomas Kyte, 2006). This can be considered to be apex of database technology available. But SQL cannot do every thing that a procedural language can do. PL/SQL reaches out and links the power of SQL with the flexibility of a procedural language. There is other works PL/SQL is an extension of SQL. PL/SQL is an application development language that contains numerous (Amar, 2003) procedural statement and commands along with SQL statements and commands along with SQL commands. This language bridges the gap between database technology and procedural programming language. Basically, an application development too], PL/SQL uses the facilities of the sophisticated Oracle RDBMS and extends the standard SQI, database language. PL/SQL permits the use of all SQL data manipulation statement including INSERT, UPDATE, DELETE and SELECT as well as transaction processing statements like COMMIT, ROLLBACK and SAVEPOINT. PL/SQL not only allow us to manipulate data using SQL statements, but it also lets us process the data using flow control statements such as iterative loops and conditional branching which could not he done in SQL.

[1]Assistant Professor, Department of Electrical & Electronic Engineering, [2]Senior Lecturer , Department of Computer Science and Engineering, Bangladesh University, Dhaka-1207, Bangladesh. *Corresponding author's Email: DIPO0001@ntu.edu.sg.

Working principal

The program can be written in the following way. This is given in below:

DECLARE ---- THIS IS THE DECLERATION SECTIUN, THTS IS THE OPTIONAL SECTION---------------- V _ENAME VARCHAR2(30); V _JOB VARCI-TAR2(30); V_SAL NUMBER(9); V_HIREDATE DATE; V.BOOL BOOLEAN; V_EMPNU F_MP.EMPNO%TYPE NOT NULL: =7369; V_U1rI''I'NU C'ONS"1'AN"1' L:MP.CMPNO'%'CYPL:=20; -------- -----DECLERATTON SECTION IS ENDED --------- BEGIN SELECT ENAME, JOB, SAL, HTREDATE, ET\4PNO INTO V_ENAME,yJUB,V_SAL,VHIREDATE,yEMPNO FROM EMP WIIERE ENAME='&ENAME'; DBMS_OUTPUT.PUT LINE(yENAMEII'TS THE '!IV JUBII'OF THE COMPANY AND GET $'l,IV_SALII'PER MONTH'); ------------HERE WE CAN APPLY THE EXCEPTION HANDLING OPTIONAL SECTION - -------- --- END;	The output of the program is given in below: Enter value for ename: SCOTT old 14: NVI-IERE ENAME='RcENAME'; new 1=I: WHERE ENAM1r='S('()TT'; SC'.UTTIS THE ANr1LYSTUF THE COMPANY AND GET $i()UUPER MUNTI I PL/SQL procedure successfully completed. Using blind variables: To reference a blind variable in PL/SQL, we must prefix its name with a colonl '~ . Example SQL> variable SUM VARC(`IAR?(6U) SQL> DECLARE 2 V'-NAME %VAR(7I1AR2(50)ˉ.='TODAY WE ARE STARTING PROGRAMMING UNIT', 3 BEGIN 4 :SUM:ˉV_NAME; 5 END 6; 7/ fig:1:Copyrit Oracle corporation, 1999. All rights reserved. Printing Blind variables: SQL> PRINT SUM; The output of the program is given below: SUM

Just like any other programming language, statement can be grouped in PLSQL also if these groups are named, they are named, they are called subprograms and they are assumed then they are called anonymous blocks. Blocks can contain more blocks and these blocks are referred to as nested blocks or sub-blocks. These are nested in an enclosing block. Nesting is permitted in the executable and exception handling hart. but not in the declarative part, but not in the declarative part. A maximum of '200 levels of" nesting is permitted. Figure 2 illustrates a nested block (sub-block). The program can be written in nested PL/LQL Black given below:

```
DECLARE
V_ENAME VARCHAR?(1 >);
V.EMPNO NUMBER(S);
V _JOB SEMIP.JOB%TYPE;
V_SAL SEMP. SAL%TYPE;
V _DATE DATE;
V_COMM NUJMBER(5);
V_DEP'TNU SEMP.DEPTNU%'I'YPE;
V ROWS VARCHAR2(9),
BEGIN DECLARE
V ROWS VARCHAR2(9);
BEGIN
INSERT INTO SEMP(EMPNU,ENAME,JOB, SAL,HIREDATE,DEPTINI0,COMiM)
VALUES(5555,'WALTUN','ANALYST',3000,SYSDATE, 10, NULL);
V_ROWS: =SQL%RUWCUUNT;
DBMS.OUTPUT. PUT_ LI1vTE('NO OF ROWS'||V_ROWS);
END;
SELECT EMPNU,ENAME,SAL,JOB,DEPTNO,HIREDATE,CUMM INTO V EMPNU,V _ENANIE,V SAL,V JUB,V
DEPTNO,V_DATE,V CUMM FROM SEMP WHERE ENAME='WALTON';
DBMS_OUTPUT.PUT_ LINE(V ENAME||'HAS JOINED IN THE COMPANY ON '|}|V_ DATEII'AS A'||V_ JOB ||''WITH
SALARY $'|| V_SALL|| 'PER MONTIH');
E
N
D
;
 /
```

The output of the program is given below:

```
NO OF ROWS 1
WALTUNI-IAS JOINED IN 'THF COMPANY ON04-JAN-O1ASA ANALYSTWITH SALARY $3000PER MONTH
```

PL/SQL procedure successfully completed.

To process a SQL statement, PI,/SQL opens a work area in the memory called mi cursor area. PL/SQL uses the context area to excess the SQL statement and store processing information. A PL/SQL construct called 'cursor' allows us to name a context area, access its information and in some cases, control its processing. It keeps track of' which raw is currently being processed. This set of rows returned by any query can consist of zero, one, or many rows depending on the number of rows that meet the query's search conditions. When a query means a multiple rows, a cursor can be explicitly defined to:

- Process beyond the first row returned by the query.
- Keep track of which row is currently being processed.

The set of rows, returned by a multi-row query, is called the active set. PL/SQL supports two types of cursors, implicit and explicit cursors.

Implicit cursors: Oracle automatically opens a context area to process each SQL statement 1:N-en when the SQL statement is not a query, there is useful information stored in Oracle context -ca. To access this information PI_/SQI_ refers to the most recently opened context area as "SQL%" cursor.

Explicit cursor: If a query returns multiple rows, a cursor can be defined to keep track of which row is being processed. This is known as an explicit cursor. These cursors are like variables to hold more than one record, variable they also need to be defined (Shah Nitesh, 2002). We can initialize cursor parameters to default values. We can pass different numbers of' actual parameters to a cursor i1ccePliny or overwriting the default values as se please.

The scope of cursor parameters is local to the cursor. The values of cursor parameters are used by the associated query when the cursor is opened. A cursor is defined in the declaration part of the 1,/SQL block by naming it and associating it with a query. To define a cursor we can use the following tatement. Cursor <cursorname > IS
< SELECT statement>
For example,

```
declare
cursor dept cursor is
select deptno,dname,loc trom dept order by deptno;
cursor emp cursor(v deptna number) is
select * from emp where deptno=v_deptno; emp.record
emp%rrnvtype;
n.deptno dcpt.deptno%type; v dname dept.dname%type;
v_loc dept.loc%type;
v.linc varchar2(100); BEGiN
v line:='-___________________________-; open dept
cursor;
Ioop
fetch dept cursor into n_deptno,v_dname,v_loc;
exit when dept cursor%notfound;
dbms_output.put_line('departmentno'll'….'||'department
name'||v_dname||'loc'||v_loc);
dbms output.put line(v_line);
if emp_cursor%isopen then
close emp.cursor;
end if,
open emp_cursor(n_deptno);
loop
fetch emp_cursor into emprecord;
exit when emp_cursor%notfound;
dhms_cnrtput.put_line(emp_record.enamelI'
              'lEemp_record.jobll' '!!emp_record.salll'
'llemp_recurd.hiredate);
end loop, end loop; close emp_cursor; close dept_cursar;
end;
/
```

```
The output of this program is given in below:
departmentno IOdepartment name:
ACCOUNTfNG1ocNEW YORK ------------------------
---
CLARK MANAGER 2450 09-JUN-81 KING
PRESIDENT 5000 17-NUV-81 MILLER CLERK
1100 ?_s-JAN-B? depart rnentno 20department
name:RESI?:\RC.'F11ocDALLi•S -----------------------
-
SM1TH CLERK 800 17-DEC-80 JONES MANAGER
?975 02-APR-81 SCOTT ANALYST 3000 19-APR-
87 ADAMS              CLERK 1 100 23-MAY-87
FORD ANALYST 3000 0i-DEC-81 departmentno
30department name: SAI,ESIocCHICAGO
---------------------------
ALLEN SALESMAN 1600 20-FEB-81 WARD
SALESMAN 1250 22-I•L;13-81 MARTIN
SALESMAN 1 250 ?8-SEP-81 BLAKE MANAGER
?850 O 1-MAY-81 TURNER SALESMAN 1500 08-
SEP-B I JAMES CLERK 950 03-DEC-81
departmentno 40department name-
OPERATIONSlocBOSTON
PL/SQL procedure successfully completed.
```

Three statements are used to manipulate a cursor: i. OPEN ii. FETCH and iii. CLOSE

The open statement: The cursor must be initialized or opened with the OPEN statement before any rows are returned by the query. Opening the cursor executes the query and identities the action set, which consists of rows that need the query search criteria. The OPEN statement IS used as follows: OPEN <cursorname >;

The FETCH statement: A fetch statement is used to receive the rows in the active set one at a time. 1 has to be executed repeatedly for processing multiple records. The format of the FETCH statement is as follows: FETCH <cursornamc > INTO variable 1, variable 2, ...,variable N;
or FETCH <cursorname> INTO record_variable.

The close statement: When all the rows of the cursor have been processed, the cursor must he closed with a CLOSE statement and this makes the active set undefined. The format of the CLOSE statement is as follows: CLOSE <cursorname>

Every cursor has four attributes that can be used to access the cursor's context area. To use these tour attributes, we simply append it to the name of the cursor. The flour attributes are:

i. %NOTFOUND, ii. %FOUND iii. 3 %ROWCOUNT and iv. %OPEN

The % NOTFOUND attribute is evaluates to TRUE if the last FETCH failed because no more rows were available or to FALSE if the last FETCH returned a row.

The %FOUND attribute is evaluates to TRUE if the last FETCH returned a row, or to FALSE if the last FETCH failed if no more rows were available.

The %ROWCOUNT Attribute is returns the number of rows FETCHED from the active set till row.

Cursor FOR Loop

A cursor FOR loop implicitly declares its loop index as a record OF %ROWTYPE opens the cursor, repeatedly fetches rows of values from the active set into fields in the record, then it closes the cursor when ass rows have been processed or when tile loop is excited Parameterized Cursors. A cursor can also receive parameters. These parameters can be only as input parameters to tile SELECT statement of the cursor. They cannot be used to return data from the cursor like functions. A cursor parameters can appear in a query wherever a constant can appear.

Cursor for update of and current of the word SQL is reserved by PL/SQL for use as the default name for Oracle context areas and cannot be used in cursor declaration. The CURRENT OF classes can be used in an UPDATE or DELETE statement to refer to the current row, provided the select statement contains the FOR UPDATE OF clause. Although the OPEN and FETCH statements cannot be used to manipulate the SQL% cursor, the cursor attributes can be used to access its context area. Before Oracle opens The SQL cursor automatically, the implicit cursor attributes evaluate NULL.

Exceptions are nothing but error handlers, they are written to handle the error that might occur in a PL/SQL block. Runtime errors occur during the execution of the PL/SQL block. Runtime errors arise from design faults, coding mistakes, hardware failure and many other sources. A PL/SQL block can handle the runtime errors that occur. Exception can be either internally defined or user-designed. When an errors, an exception is raised whereby normal program execution stops and the control is transferred to the exception handling routine oh the PL/SOL. block. Internal exceptions are implicitly raised, whereas user-defined exceptions are explicitly raised, by using the RAISE statement PL/SQL makes it easy to detect and process error conditions called `exception'. An exception is an condition, not an object. When an error occurs, an exception is raised. That is, normal execution stops and controls transfers to the exception-handling part of the PL/SQL block or subprogram. To handle raised exceptions, we write separate routines called 'exception handlers'. For example, if you try to divide a number hy zero, the predefined exception ZERO _DIVIDE is raised automatically. A package is a database object that groups logically related PL/SQL types, objects and subprograms. Alike every other database objects we need to declare it before using it (Shi Wenyauan, 2005).

```
declare
v -rows varchar2(I2);
eimp_record emp%row type;
cursor emp_cursor is select * from emp;
BEGIN
open emp cursor;
for i in 1..14 loop
fetch emp_cursor into emp_record;
dbms_output.put_line('The employee ' | |
lemp_record.enamel||J'get $'llemp_record.sal II'as
a'llemp_recvrd.job);
if emp record.cvmm is not null then
dbms_output.pul_ line('And get commision per month'),
else
dbms_output.put_line('and never get commision');
 v_rows: =emp_cursor%rowcount;
dbms_output.put_ line(v_rows);
end if;
end loop;
close emp_cursor;
end;
```

```
/ The output of this program is given below
The employee SMITHbet $ I 382.4as aCLERK and never
get commision
I
The employee AI_I_T:N(;et $276-t.Sas aSALPSM.IN
And get commision per month
The employee WARDget $2160as aSALESMAN And
get commision per month
The employee JONES-et $5145.9gas aMANAGER and
never get commision
4
The employee Martinet $2160as aSALESMAN And get
coninitsion per month
The employee BLAKEget $4924.8as aMANAGER and
never "et cornrnisicm
The employee CLARKget $4?33.0as aMANAGER and
never get commision
7
The employee SCOTT-et $5184as aANALYST and
never get cornrnision
8
The employee KPVGget $8640as aPRESIDENT and
never get conuniaion
O
The employee TURNL:Rget $25c)2as aSALL;SMAN
And get commisicm per month
The employee ADAMSget $1900.4as aCLT:RK and
never( et cornmision
11
The employee JAMES-et $164 I .6as aCLERK and never
get commision
12
The employee Forget $10125.01 as aANALYST and
never get commision
13
The employee MILLFRget $?246.4as aCLERK and
never get commision
14
PL/SQL procedure successfully completed.
```

Packages usually have two parts: i. Specification and ii. Body

The specification serves as an interface to the applications it declares the types, variables, constants, exceptions, errors and subprograms available for use. The fully defines cursors and subprograms and so implements the specification. The specification holds public declarations, which are visible and accessible to the applications. The body implementation details and private declarations in which ae is hidden and accessible from the applications. The specification can be thought of an operational interface (i.e. applications make calls to procedure and function given in the specifications) and the body as a "block body" (i.e. the application does not know what happens within the package body, it only receives the result of a procedure or function that executes in the body). A package body can be replaced, enhanced or replaced without changing the interface to the package body. Only the declarations in the package specification are visible and accessible to applications. Implementation details in the package are hidden and inaccessible. So the body can be changed without having to recompile calling programs.

```
Create or replace package over_load is
procedure add-value
(v_deptno dept.deptno%type,
V_dname dept.dname%type,
v loc dept. loc%type);
 procedure add value
(v deptno dept.deno%tyype;
V_dname dept.dname%type);
procedure add_value
(v_deptno dept.deptno%type);
end;
/
```

The output of this program is given in below:

Package created: A database trigger is a stored PL/SQL program unit associated with specific database table. Oracle executes the database trigger automatically whereas a given SQL operati0n affects the table. So these triggers are invoked implicitly. Triggers can supplement the standard capabilities of Oracle to provide a highly customized database management system. For example, a trigger can restrict D.[1]01operation against n table to these issued during regular business hours. A trigger could also DML operations to occur only at certain times during weekdays. '1' riggers can also be used to:

- Automatically generate derived column values.
- Prevent invalid transactions
- Enforce complex security authorization
- Enforce complex business rules
- Maintain synchronous table replicates
- Gather statistics on table access
- Derive column values automatically

Triggers are useful term customizing a database. However, they should be used only when necessary.

A database trigger has three parts: i. The triggering event, ii. An optional trigger constraint and iii. A trigger action.

When an event occurs, the database trigger fired and an anonymous PL/SQL block performs the action. Database triggers fire with the packages of the owner, not the current user. So an owner must have appropriate access to all objects referenced by all trigger action.

```
create or replace trigger secure_emp
before insert or update or delete on emp I BEGIN
if(to char(sysdate,'day') in ('saturday','sunday')) or (to_char(sysdate,'hh am') not between '08 am' and
'O5 pm')
then raise application error(-20316,'you have no permission to access out of normal office hour');
end
if,
end
; /
```

The output of this program is trigger created.

METHODOLOGY

Unlike JDBC, SQLJ provides strong typing of query outputs and return values. The variables that are declared in java, to receive data from queries or function calls are type checked against SQL data types on during compilation. It provides concise and less error prone codes when compared to dynamic SQL because the database access in SQLJ is made through static SQL. This research was done in the Department of Electrical and Electronic engineering in Rajshahi University of Engineering and Technology, Rajshahi, Bangladesh during April 2003 to April 2004. With the advent of Oracle 81, it is now possible to write stared procedure in either Java or PL/SQL. This is because the Aurora JVM (Java Virtual Machine) has been incorporated with the SQL and PL/SQL engine. As a result programmers are now able to develop efficient and robust application using the programming resources of both Java and PL/SQL .Programs created in SQLJ are saved with sqlj extension. The SQLI translator converts the embedded SQ1.J statement to JDBC calls 'The JDRC driver should he installed to provide connected y to the database. SQLJ is compiled and executed. The source code written in SQLJ contains embedded static SQL statements. The translator processes the source code and converts all the SQL statements to JDi3C calls. In the process the SQLJ file is converted to a Java file. The Java file is converted to corresponding class file by the compiler. The class file contains Java byte codes with JDI3C calls. The Java byte codes are responsible for interaction between the user and the database. .At runtime the JDBC driver interacts with the Java byte codes and the database. The instructions are real from the class file and data is retrieved form the database. The Java byte codes format the data and send it back to the user. .A SQLJ code consists of a combination of Java and SQLJ declaration and statements. SQLJ declarations begin with the token #token followed by declarations. There are two kinds of SQI.J declarations. They are iterator declarations; these declaration' define iterator classes. The iterator classes are implemented as iterator objects, which are used for receiving rnulti query results from the database.

• Connection context declarations: These declarations define connection context classes which help the SQLJ program to connect to different database schemas (Ress Hardman and Michael Michaughlin, 2006).

Connect method: The Oracle connect method needs certain information to be able to connect to the database. The information that needs to be specified are the type of JD13C driver, the URI, of the database, the usernarne and the password. Since these information is needed for every SQLJ program we create, they can be stored in a file connect properties.

```
Sqlj.url¨= jdbc.oracle::thin@ 127,0.0.1¨1521-ORCL
Sqlj.user=scott
Sqlj.password=tiger
```

The actual process of compiling and executing a SQI-J program components. They are: SQLJ translator and SQLJ Runtime.

SQLJ translator: The process of compilation of a SQI_J program file is handled by SQI,J translator component. This component is written in pure Java. It invokes two process, translator and compilation. .The translator process converts all embedded SQL statements into JDBC calls. The resulting file has an extension Java. This Java tile is compiled 1) (fie compiler to give a class File. In the first phase of translation, while the S (11.J translator parses the SQLJ file, it invokes a Java parser and a SQLJ parser. The Java checks the syntax of the java host variables and expressions within the SQLJ operations. The SQLJ parser checks the syntax of the SQL statements. The SQL syntax checking is done during the semantics checking. Once the source code is verified syntactically the translator invokes the semantics checker. The semantics checker verifies the validity of' he Java types used in the SQI, operations. It connects to the database to check the compatibility of the .lava types and the SQL types.

SQL runtime: Each time a user wants to execute a SQLJ application the SQLJ runtime component is invoked (Li Xuedang, 2007). It implements the desired SQL operation on the database using the .1DBC driver. The SQLJ runtime component reads information from the profile file and passes the information to the JDBC driver. As we have sent earlier during compilation, a profile tile is generated with an extension .ser. This file is Java resources file, which stores information above the SQI, operation in the input file. The profile riles are Java resource files that are used by the SQLJ runtime during the execution of an application. A profile is generated for each connection context we use in our application. Connection contexts are meant by which a SQLJ program connects to a data base. A profile describes the operation to be performed on the database objects such as tables, views and stored procedure. It exits as a Java serialized object within the resource file. The resource file uses a .ser extension. The generated SQL.J profiles names is qualified with the sqlj input file name followed by the string

At first we create tables in the following way in Oracle 81 enterprise edition 8, i.7 version.

```
 create table SALES( URDERNU NUMBER(4), PRODUCT.11") NUMBER(S), COPIES NUMBER(3), AMOUNT
PAYABLE NUMBER(6,2), PAYMENT MODE CHAR(] 0), ORDERJ_DATE            DATE,
DEL_DATE DATE
}}
 create table BOOK _DE'T( BOOK ID NUMBER(5), NAME VARCHAR2(3t)), AUTHOR VARCHAR2(30),
BOOK _SOLD                    NUMBER(8), EDITION DATE,
BOOKS TYPE                    CHAR(I 0), QTY.ON.HAND NUMBER(4),
REORDER_LAVEL Ni1MBER(3), PRICE NUMIIER(l 01)
INSERT INTO BOOK DT?'I' VALUES( I001,"TUXIN','RUBIN CU(.)K', I 58,'US-()('.'I'99','FIC'I'1UN',842,25U,24S);
INSERT INTO SALES VALUES( l, I UUU 1,2,590.00,'CFIEQUE','08-SEP-00','08-UCTUU');
```

40 Then we can create class file and path in the following way. Classpath:

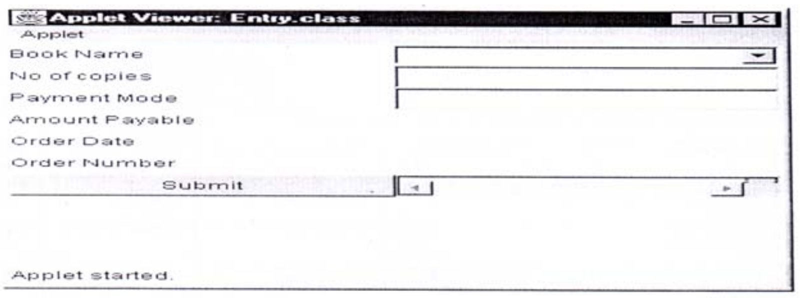
```
D:\Oracle\Ora81\jdbc\lib\classes111.zip;D:\Oracle\Ora81\sqlj\lib\translator.zip;D:\Oracle\Ora81\sqlj\lib\runtime.zip;%path%;
Path:
D:\Oracle\Ora81\jdbc\lib\classes111.zip;D:\Oracle\Ora81\bin;D:\Oracle\Ora81\sqlj\lib\translator.zip;D:\Oracle\Ora81\sqlj\lib\runtime.zip;d:\jdk1.3\bin
```

In the next step, we create a.sqlj file in the following format:

```
import java.sql.*;
import oracle. sdlj.wntirne.()racle;
public class DatnEntry(
public static int id,sold,qty,rlvl,max;
public static void main(String args[J) throws SQI-
Exception {
      max=934; try{
long pr;
String narne,auth,hty,dt,dt I; java.sqLDate edi;
Oracle.cotinect(DataEtitry.class, "connect properties");
              #scll{
ND,REURDER LAVEL,PRICEINTO:
id,:name,:auth,:sold,:edi,:bty,:cjty,:rlvl,:pr FROM
BOUK_DET WHERE BOOK.ID ¯-IUUUI };
#sd1{SEI_FC'.T MAX(orderno)¯+¯I INTO :max FROM
SALES); Ifsyl{ST;'I" :dt-sys(latc};
#sql{SET :dtl=(sysdate+7)};
              #sql{INSERTINTOSALES-RI
#sql{CUMMIT};
System. out. print ln("\n=SALES Table
Upgraded=====");
catch(SQLErception se) {
System. out. println("Error is occured for Data Retrive
in orcl"); System. err. print] n("ErrorError"+ se);
```

In the ultimate step, we can create a Java application program.

```
import DataEntry; import java.sql.*; import java.awt.*; import
java.applet.*; import java.awt.event.
*;
public class Entry l extends Applet implements ActionListener(
private Button submit;
private Label Ib I,Ib2,Ib3,Ib4,1b5,1b6,1b7,1b8,1b9,1b 10;
TextField tl,t2;
Choice ch; int a;
public void init(){
Ib I U=new Label("",2); DataEntry DE=new DataEntry(); try{
a=DE.doSomething(); Ib 10. setText("a="+a);
   t 1 =new TextField("a="+a); }catch(SQLrxception se)
{1bt0.setText(""-f-se);}
//setBackground(Color.red); setl,ayout(new Grid Layout(7,2));
lbl=new Label("Book Name:"); add(Ib 1);
ch=new Choice();
ch.add("Javn in 21 days"); ch.add("Sun Java");
ch. add("C-r4"); ch.add("DB with SQLJ");
ch.add("Uracle8i"); add(ch);
lb2=new Label("No of Copies:"); add(lb2);
tl-new Text Field(20); add(t 1);
163=new Label("Payment Mode:"); add(lb3);
t2=new TextField(20); add(t2);
IM¯new 1.ahel("Amcwnt Payable"); add(11A);
16S=new Label(""); add(IbS);
lb6=new Label("Urder Date"); add(Ib6);
Ib7=new Label(""); add(Ib7);
1b8=new Label("Urder Number"); add(Ib8);
Ib9=new Label(""); add(Ih9);
submit=new Button(" Submit"); add(submit);
//1610-new Label(" "+DE. max); add(Ib 10);
public void act ionPerformed(ActionEvent ae){ /I
,
}
}
```



```
Applet Viewer: Entry.class
Applet
Book Name
No of copies
Payment Mode
Amount Payable
Order Date
Order Number
Submit
Applet started.
```

Fig. 1. Java applet entry output.

The output of the program is run successfully. We got an idea of how a program written in SQLJ is complied. It is seen that a SQLJ program consists of java codes and embedded SQL statements. When translated, a SQLJ program is converted into a java program which is then compiled and executed. The process of translation actually converts the embedded SQL statements into java statements which keeping the java codes unchanged. We have learnt how java expressions are bridged with embedded SQL statements in SQLJ code. Java expression include list of operators, variables and methods. These Java expressions are used for controlling the execution of the program. Java expressions include a. Host expression, b. Content expression and c. Result expression. We have learnt how to retrieve and manipulate data from the database using a special java class called iterator.

CONCLUSION

With the advent of Oracle 8i, it is now possible to write procedure in either Java or PL/SQL. This result is a new language-SQLJ. This language facilitates SQL operation to be embedded in Java program. The basics of SQLJ programming includes the advantages that it enjoys over similar database access tools like JDBC. SQLJ supports blind variables directly with the SQL statement.

REFERENCES

Amar, K. P. 2003. Execute emmediate option for Dynamics SQL and PL/SQL.

Li Xuedang, 2007. The technical character of PL/SQL and its application in the software developing cfhi technology No. 6.

Ress Hardman and Michael Michaughlin. 2006. Expert Oracle PL/SQL Design and Develop advanced PL/SQL Solutions. Beijing: Tsinghua University press.

Shah, Nitesh, 2002. Database systems using Oracle : a simplifies guide to SQL & PL/SQL*. Prentice Hall.

Shi Wenyuan, 2005. Research and Implementation of some key technologies for PL/SQL Engine. Changsha: National university of Defence technology.

Thomas Kyte, 2006. Programming and of Oracle 9I & 10g. Designing: people's post and telecommunication publishing.