Lab Guide for RDBMS Essentials

Education and Research

Infosys®

POWERED BY INTELLECT
DRIVEN BY VALUES

Author(s)	Rengarajan Ramanujam, Shinu Thomas
Authorized by	Srikantan Moorthy
Creation/Revision Date	June 2010
Version	1.4

Document Revision History

Version	Date	Author(s)	Reviewer(s)	Comments
1.0	16-Mar-09	Rengarajan, Shinu Thomas	Kiran RK	Initial Draft
1.1	12-May-09	Rengarajan, Shinu Thomas	Kiran RK	Review comments incorporated after Pilot
1.2	15-Jun-09	Rengarajan, Shinu Thomas	Kiran RK	Review comments incorporated after Pilot
1.3	24-Mar-2010	Shinu Thomas	Rengarajan	IP checking and incorporation
1.4	1-Jun-2010	Giri Babu Boyani, Amit Behl	Rengarajan, Shinu Thomas, Amit Behl	Conversion of assignments to Devsquare platform

Contents

1 Background

This document contains assignments to be completed as part of the hands on for the subject RDBMS Essentials **(Course code: DB91).**

Note: In order to complete the course, assignments in this document must be completed in the sequence mentioned.

1 Assignments for Day 3 of RDBMS Essentials

1.1 Demo 1: Usage of %ROWTYPE anchored declaration

Objective: To be able to declare row type variables using %ROWTYPE.

Platform: Use SQL*PLUS to solve this assignment. Create the tables and insert the data using the “CourseRegistrationDBDesign.sql” script.

Problem Description:

Write a PL/SQL program accept the department details and display the same.
Create a single row type variable to accept the values using %ROWTYPE and display it.

```
SET SERVEROUTPUT ON;

DECLARE
  v_department department%ROWTYPE;
 BEGIN
  v_department.departmentid :=&g_deptid;
  v_department.departmentname:=’&g_deptname’;
  v_department.headofdepartment:=’&g_hod’;
  DBMS_OUTPUT.PUT_LINE(‘Department Id: ‘||v_department.departmentid);
  DBMS_OUTPUT.PUT_LINE(‘Department Name:
‘||v_department.departmentname);
  DBMS_OUTPUT.PUT_LINE(‘Department Head:
‘||v_department.headofdepartment);
END;
/
OUTPUT:
```

```
Enter value for g_deptid: 10
old   4:   v_department.departmentid :=&g_deptid;
```

```
new   4:   v_department.departmentid :=10;
Enter value for g_deptname: Computer Science
old   5:   v_department.departmentname:='&g_deptname';
new   5:   v_department.departmentname:='Computer Science';
Enter value for g_hod: I105
old   6:   v_department.headofdepartment:='&g_hod';
new   6:   v_department.headofdepartment:='I101';
Department Id: 10
Department Name: Computer Science
Department Head: I101
```

1.2 Assignment 1: Usage of %ROWTYPE anchored declaration

Objective: To be able to declare row type variables using %ROWTYPE.

Platform: Use SQL*PLUS to solve this assignment. Make sure that all the necessary tables are present in your login id.

How to check the list of all tables present in my login id?

Use the below command in SQL> prompt to check the list of tables present in your login id.

SQL>SELECT table_name FROM tabs;

Problem Description:

1. Write a PL/SQL program to accept the branch details and display the same.
   Create a single row type variable to accept the values using %ROWTYPE and display it.

   [Hint: Please refer to the branch table description in the database]

2. Write a PL/SQL program to accept the student details and display the same.
   Create a single row type variable to accept the values using %ROWTYPE and display it.

   [Hint: Please refer to the student table description in the database]

1.3 Demo 2: Using SQL in PL/SQL

Objective: To be able to use DMLs of SQL in PL/SQL blocks.

Platform: Use SQL*PLUS to solve this assignment. Make sure that all the necessary tables are present in your login id.

Problem Description:

Write a PL/SQL program to display the details of a department.

```
SET VERIFY OFF;

DECLARE
  v_department department%ROWTYPE;
  v_departmentid department.departmentid%TYPE;
BEGIN
  v_departmentid:= &v_departmentid;
  SELECT * INTO v_department FROM department WHERE
departmentid=v_departmentid;
  DBMS_OUTPUT.PUT_LINE('Dept ID: '||v_department.departmentid);
  DBMS_OUTPUT.PUT_LINE('Dept Name: '||v_department.departmentname);
  DBMS_OUTPUT.PUT_LINE('Dept HOD: '||v_department.headofdepartment);
END;
/
OUTPUT:
```

```
Enter value for v_departmentid: 20
Dept ID: 20
Dept Name: Electronics
Dept HOD: I104
```

1.4 *Assignment 2: Using SQL in PL/SQL*

Objective: To be able to use DMLs of SQL in PL/SQL blocks.

Platform: Use Devsquare platform to solve the following. Make use of the URL shared by the educator or Batch Owner to login to Devsquare.

Problem Description:

1. Write a PL/SQL program to accept a studentid and display the **applicantid, currentsemester, branchid, userid, password** and **residentialstatus** in separate lines.

   Note: In Devsquare, type the solution for this problem in D3_Assignment2_1.sql

   Use the variable ***v_studentid*** to accept **studentid**

Use the following format to display the data.

<applicantid>
<currentsemester>
<branchid>
<userid>
<password>
<residentialstatus>

2. Write a PL/SQL program update the hostelfee by 10% for a given hostelid. Display the message **“Updated Successfully”.**

   Note: In Devsquare, type the solution for this problem in D3_Assignment2_2.sql

   Use the variable ***v_hostelid*** to accept hostelid .

NOTE:
Message should be displayed exactly as given in the problem statement. Any change in the message may throw an error.

3. Write a PL/SQL program to accept the registrationid, projectscore, assignmentscore, internalscore and semester score.

   Note: In Devsquare, type the solution for this problem in D3_Assignment2_3.sql

   Use the following variable names :
   v_registrationid
   v_projectscore
   v_assignmentscore
   v_internalscore
   v_semester

NOTE:

a. Check whether the project score is between 0 and 20, if not display a message **Invalid project score**

b. Check whether the assignment score is between 0 and 10 if not display a message **Invalid assignment score**

c. Check whether the internal score is between 0 and 20 if not display a message **Invalid internal**

score

d. Check whether the semester score is between 0 and 50, if not display a message **Invalid semester score**

Do not try to implement the above logic in a loop until a correct score is given as input, as you cannot do so using PL/SQL

If all the scores are valid, update the registration table with all valid values for the given registrationid and display a message **Scores updated successfully**

NOTE: 1: While using Devsquare platform to implement the above PL/SQL block you will not be able to see the updated values.

2: While using SQL* PLUS platform to implement the above PL/SQL block you will be able to see the updated values.

4. Write a PL/SQ L program to delete a branch by branchid 'B4'. After successful deletion display a message **Branch deleted successfully**.

   Platform: Use SQL*PLUS to solve this assignment.

1.5 Assignment 3: Exception handling – Predefined Exceptions

Objective: To be able to handle predefined exceptions in PL/SQL blocks.

Platform: Use SQL*PLUS to solve this assignment. Make sure that all the necessary tables are present in your login id.

Problem Description:

1. Write a PL/SQL block to verify whether the given applicantid exists or not. If exists display the applicant name, email id and overall percentage of the applicant else display the message **The applicant id does not exist**
   [Hint: NO_DATA_FOUND exception]

2. Write a PL/SQL block to verify whether the given studentid exists or not. If exists display the userid, password and branch id of the student else display the message **The student id does not exist**
   [Hint: NO_DATA_FOUND exception]

3. Write a PL/SQL block to display the course name and branch id of courses which are registered by a given student. If the student has registered only for one course

display the course name and branch id else display the message, **Student has opted for more than one course**

NOTE:

Do not use cursors, display appropriate message in case of exceptions

[Hint: TOO_MANY_ROWS exception]

4. Write a PL/SQL block to divide two numbers and display the result. Give the following inputs and incase of exceptions display the message **Division by Zero** in case one of the operand is Zero.

Trials	Number1	Number2
1	10	20
2	30	0
3	0	40
5	15	20

[Hint: ZERO_DIVIDE exception]

5. Write a PL/SQL block to insert the following details to the branch table. Observe the exception raised. Handle the exception raised and display the message **Branch with this branchid already exists in the branch table**

NOTE:
The branch id B1 is already present in the branch table

Column Name	Value
branchid	B1
branchname	Mechanical
departmentid	10
seatsavailable	5

[Hint:DUP_VAL_ON_INDEX exception]

1.6 Demo 3: Exception handling – Predefined Oracle Server Exceptions

Objective: To be able to handle predefined exceptions in PL/SQL blocks.

Problem Description:

Write a PL/SQL block to add a new branch in the branch table. Accept the branchid, branchname, departmentid and seatsavailable. The branchid should be unique and departmentid should be an existing departmentid in department table. Display appropriate messages in case of errors or exceptions else insert the values and display a success message.

```
DECLARE
      v_branchid branch.branchid%TYPE:='&v_branchid';
      v_branchname branch.branchname%TYPE:='&v_branchname';
      v_departmentid branch.departmentid%TYPE:=&v_departmentid;
      v_seatsavailable branch.seatsavailable%TYPE:=&v_seatsavailable;
      v_deptid department.departmentid%TYPE;
BEGIN
      SELECT departmentid INTO v_deptid FROM department WHERE
      departmentid=v_departmentid;
      INSERT INTO branch
VALUES(v_branchid,v_branchname,v_departmentid,v_seatsavailable);
      DBMS_OUTPUT.PUT_LINE('Successfully added the branch');
EXCEPTION
      WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('No such department exists');
      WHEN DUP_VAL_ON_INDEX THEN
            DBMS_OUTPUT.PUT_LINE('Branch id already exists');
      WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Some Error!!!');
END;

OUTPUT
```

```
SQL> SET VERIFY OFF;
SQL> /
Enter value for v_branchid: B4
Enter value for v_branchname: MECH
Enter value for v_departmentid: 20
Enter value for v_seatsavailable: 9
Successfully added the branch

PL/SQL procedure successfully completed.
```

```
SQL> /
Enter value for v_branchid: B5
Enter value for v_branchname: ENV
Enter value for v_departmentid: 50
Enter value for v_seatsavailable: 5
No such department exists

PL/SQL procedure successfully completed.

Enter value for v_branchid: B1
Enter value for v_branchname: Telecom
Enter value for v_departmentid: 20
Enter value for v_seatsavailable: 5
Branch id already exists

PL/SQL procedure successfully completed.
Enter value for v_branchid: ME
Enter value for v_branchname: DDDDD
Enter value for v_departmentid: 10
Enter value for v_seatsavailable: 3
Some Error!!!
```

Explore:
Analyze why the exception is thrown for the last inputs. Why is it caught in OTHERS exception handler?

1.7 Demo 4: Exception handling – Non Predefined Oracle Server Exceptions

Objective: To be able to handle non-predefined Oracle server exceptions in PL/SQL blocks.

Problem Description:

Write a PL/SQL block to add a new instructor with the following values:
It will throw a non-predefined Oracle server exception. Handle the exception and display appropriate outputs.

NOTE:
The value 50 is not present in the department.departmentid, so it will throw a referential integrity violation error. This has to be handled in the PL/SQL block.

Column Name	Value
Instructorid	I110
Instructorname	Steve Benhur
Dateofjoining	10-Jan-2009
Departmentid	50
Remaininghours	40

NOTE:
-2291 is the error number thrown by the SQL for Referential integrity violation.

```
DECLARE
      e_integrity EXCEPTION;
      PRAGMA EXCEPTION_INIT(e_integrity,-2291);
      v_instructorid   instructor.instructorid%TYPE:='&v_instructorid';
      v_instructorname instructor.instructorname%TYPE:=
'&v_instructorname';
      v_dateofjoining instructor.dateofjoining%TYPE:=
'&v_dateofjoining';
      v_departmentid instructor.departmentid%TYPE:= &v_departmentid;
      v_remaininghours instructor.remaininghours%TYPE:=
&v_remaininghours;

BEGIN
      INSERT INTO instructor VALUES ( v_instructorid, v_instructorname,
v_dateofjoining, v_departmentid,v_remaininghours);

EXCEPTION
```

```
      WHEN e_integrity THEN
           DBMS_OUTPUT.PUT_LINE('Invalid department');
      WHEN OTHERS THEN
           DBMS_OUTPUT.PUT_LINE('Some Error!!!');
END;

OUTPUT:
```

```
Enter value for v_instructorid: I110
Enter value for v_instructorname: Steve Benhur
Enter value for v_dateofjoining: 10-jan-2009
Enter value for v_departmentid: 50
Enter value for v_remaininghours: 40
Invalid department

PL/SQL procedure successfully completed.
```

1.8 Assignment 4: Exception handling – Non Predefined Oracle Server Exceptions

Objective: To be able to handle non-predefined Oracle server exceptions in PL/SQL blocks.

Platform: Use SQL*PLUS to solve this assignment. Make sure that all the necessary tables are present in your login id.

Problem Description:

1. Write a PL/SQL block to delete a course from the course table. If the given course id is not a valid course id then display appropriate message else try to delete the course. If it raises any error/exception handle it and display user friendly messages.

2. Write a PL/SQL block to update the department id of a given branch id in branch table. If the given branch id does not exist, display appropriate message. If it is not updatable handle the exception/error and display appropriate messages.

3. Write a PL/SQL block to insert a record to the table hostel with all NULL values and handle the exceptions and display appropriate messages.

1.9 Demo 5: Exception handling – User defined Exceptions

Objective: To be able to create user defined exceptions in PL/SQL blocks and handle the same.

Problem Description:

Write a PL/SQL block to add a new branch in the branch table. Accept the branchid, branchname, departmentid and seatsavilable.

- The branchid should be unique and if it is not starting with 'B' display an error message "Invalid Branch Id".
- departmentid should be an existing departmentid in department table.
- Display appropriate messages in case of errors or exceptions
- Else insert the values and display a success message.

```
DECLARE
      v_branchid branch.branchid%TYPE:='&v_branchid';
      v_branchname branch.branchname%TYPE:='&v_branchname';
      v_departmentid branch.departmentid%TYPE:=&v_departmentid;
      v_seatsavailable branch.seatsavailable%TYPE:=&v_seatsavailable;
      v_deptid department.departmentid%TYPE;
      e_invalidbranchid EXCEPTION;
BEGIN
      SELECT departmentid INTO v_deptid FROM department WHERE
      departmentid=v_departmentid;
      IF v_branchid NOT LIKE 'B%' THEN
            RAISE e_invalidbranchid;
      END IF;
      INSERT INTO branch
VALUES(v_branchid,v_branchname,v_departmentid,v_seatsavailable);
      DBMS_OUTPUT.PUT_LINE('Successfully added the branch');

EXCEPTION
      WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('No such department exists');
      WHEN DUP_VAL_ON_INDEX THEN
            DBMS_OUTPUT.PUT_LINE('Branch id already exists');
      WHEN e_invalidbranchid THEN
            DBMS_OUTPUT.PUT_LINE('Invalid Branch ID');
      WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Some Error!!!');
END;
/
```

```
OUTPUT:

Enter value for v_branchid: K10
Enter value for v_branchname: MECH
Enter value for v_departmentid: 10
Enter value for v_seatsavailable: 4
Invalid Branch ID

PL/SQL procedure successfully completed.

SQL> /
Enter value for v_branchid: B1
Enter value for v_branchname: ENV
Enter value for v_departmentid: 30
Enter value for v_seatsavailable: 3
Branch id already exists

PL/SQL procedure successfully completed.

SQL> /
Enter value for v_branchid: B4
Enter value for v_branchname: Mech
Enter value for v_departmentid: 20
Enter value for v_seatsavailable: 4
Successfully added the branch

PL/SQL procedure successfully completed.
```

1.10 Assignment 5: Exception handling – User defined Exceptions

Objective: To be able to create user defined exceptions in PL/SQL blocks and handle the same.

Platform: Solve the following problems both in devsquare and SQL*PLUS. Make use of the URL shared by the educator or Batch Owner to login to devsquare.

Problem Description:

1. Write a PL/SQL block to update the project score, assignment score and internal score for a given studentid and courseid.
   a. If the studentid is invalid then display **Invalid Student Id**
   b. If the courseid is invalid then display **Invalid Course Id**

c. If the student has not registered for the course then display **Student NOT Registered for this Course**
d. If the project score is <0 and >20 then display **Invalid Project Score**
e. If the assignment score is <0 and >10 then display **Invalid Assignment Score**
f. If the internal score is <0 and >20 then display **Invalid Internal Score**
g. If all the values are proper then update the values to the respective record and display **Updated Successfully**

Use the following variable names:
v_studentid
v_courseid
v_projectscore
v_assignmentscore
v_internalscore

NOTE:

1: While using Devsquare platform to implement the above PL/SQL block you will not be able to see the updated values.

2: While using SQL* PLUS platform to implement the above PL/SQL block you will be able to see the updated values.

2. Write a PL/SQL block to update the rooms available, hostel type and hostel fee of hostel table by giving a hostel id as input. Consider the following business requirements.
   a. If the hostelid is invalid then display **Invalid Hostel Id**
   b. If the hosteltype is other than ‘M’ or ‘F’ then display **Invalid Hostel Type**
   c. If the hostelfee is <0 or >5000 then display **Invalid Fee Structure**
   d. If all the values are proper then update the values to the respective record and display **Updated Successfully**

   Use the following variable names:
   v_hostelid
   v_hosteltype
   v_hostelfee
   v_roomsavailable

NOTE:
1: While using Devsquare platform to implement the above PL/SQL block you will not be able to see the updated values.

2: While using SQL* PLUS platform to implement the above PL/SQL block you will be able to see the updated values.

3. Implement the exception to all the Day2 PL/SQL blocks.

NOTE: Use ONLY SQL * PLUS to run this PL/SQL block

1.11 Assignment 6: Exception handling – RAISE_APPLICATION_ERROR and usage of SQLCODE and SQLERRM

Objective: To be able to raise application errors in PL/SQL blocks and handle the SQLCODE and SQLERRM in PL/SQL blocks.

Platform: Use SQL*PLUS to solve this assignment. Make sure that all the necessary tables are present in your login id.

Problem Description:

1. Modify the today's Assignment 5(1) and 5(2) of by raising an application error instead of raising a user defined exception.
2. Implement the SQLCODE and SQLERRM too in the Assignment 5(1) and 5(2).