

# Oracle PL/SQL

## Lab Book

## Document Revision History

Date	Revision No.	Author	Summary of Changes
05-Feb-2009	0.1D	Rajita Dhumal	Content Creation
09-Feb-2009		CLS team	Review
02-Jun-2011	2.0	Anu Mitra	Integration Refinements
30-Nov-2012	3.0	Hareshkumar Chandiramani	Revamp of Assignments and Conversion to iGATE format.

## Table of Contents

<i>Document Revision History .....</i>	<i>2</i>
<i>Table of Contents .....</i>	<i>3</i>
<i>Getting Started .....</i>	<i>4</i>
<i>Overview.....</i>	<i>4</i>
<i>Setup Checklist for Oracle 9i.....</i>	<i>4</i>
<i>Instructions .....</i>	<i>4</i>
<i>Learning More (Bibliography if applicable).....</i>	<i>4</i>
<i>Lab 1. Introduction to Data Dictionary .....</i>	<i>5</i>
<i>Lab 2. Introduction to PL/SQL and Cursors.....</i>	<i>6</i>
<i>Lab 3. Exception Handling and Dynamic SQL .....</i>	<i>8</i>
<i>Lab 4. Database Programming.....</i>	<i>10</i>
<i>Lab 5. Case Study 1 .....</i>	<i>14</i>
<i>Lab 6. Case Study 2 .....</i>	<i>16</i>
<i>Lab 7. Handling Files, DBMS_LOB .....</i>	<i>18</i>
<i>Lab 8. SQL *Plus Reports.....</i>	<i>19</i>
<i>Lab 9. SQL Loader .....</i>	<i>23</i>
<i>Appendices .....</i>	<i>28</i>
<i>Appendix A: Oracle Standards .....</i>	<i>28</i>
<i>Appendix B: Coding Best Practices.....</i>	<i>29</i>
<i>Appendix C: Table of Figures.....</i>	<i>30</i>
<i>Appendix D: Table of Examples .....</i>	<i>31</i>

## Getting Started

### Overview

This lab book is a guided tour for learning Oracle 9i. It comprises 'To Do' assignments. Follow the steps provided and work out the 'To Do' assignments.

### Setup Checklist for Oracle 9i

Here is what is expected on your machine in order for the lab to work.

#### Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 95, 98, or NT 4.0, 2k, XP.
- Memory: 32MB of RAM (64MB or more recommended)

#### Please ensure that the following is done:

- Oracle Client is installed on every machine
- Connectivity to Oracle Server

### Instructions

- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory Oracle 9i\_assgn. For each lab exercise create a directory as lab <lab number>.

### Learning More (Bibliography if applicable)

- Oracle10g - SQL - Student Guide - Volume 1 by Oracle Press
- Oracle10g - SQL - Student Guide - Volume 2 by Oracle Press
- Oracle10g database administration fundamentals volume 1 by Oracle Press
- Oracle10g Complete Reference by Oracle Press
- Oracle10g SQL with an Introduction to PL/SQL by Lannes L. Morris-Murphy

## Lab 1. Introduction to Data Dictionary

<b>Goals</b>	Getting the details from various Data Dictionary Objects
<b>Time</b>	45 min

- 1.1: Get the details of all the database objects and their types created by the current user.
- 1.2 Get the details of all the table names owned by current user
- 1.3 Get the details of table names and corresponding column names
- 1.4 Get the details of column names and corresponding constraint names
- 1.5: Get the details of the constraints and corresponding table name.
- 1.6: Get the details of all the View names and corresponding Text of the same.
- 1.7: Get the details of all the Sequence names and their last numbers reached so far.
- 1.8: Get the details of all the Synonym names and their parent object names.
- 1.9: Get the list of all the Index names

## Lab 2. Introduction to PL/SQL and Cursors

<b>Goals</b>	<p>The following set of exercises are designed to implement the following</p> <ul style="list-style-type: none"><li>• PL/SQL variables and data types</li><li>• Create, Compile and Run anonymous PL/SQL blocks</li><li>• Usage of Cursors</li></ul>
<b>Time</b>	1hr 30 min

### 2.1

Identify the problems(if any) in the below declarations:

```
DECLARE
V_Sample1 NUMBER(2);
V_Sample2 CONSTANT NUMBER(2) ;
V_Sample3 NUMBER(2) NOT NULL ;
V_Sample4 NUMBER(2) := 50;
V_Sample5 NUMBER(2) DEFAULT 25;
```

#### Example 1: Declaration Block

### 2.2

The following PL/SQL block is incomplete.

Modify the block to achieve requirements as stated in the comments in the block.

```
DECLARE --outer block
var_num1 NUMBER := 5;
BEGIN

DECLARE --inner block
var_num1 NUMBER := 10;
BEGIN
DBMS_OUTPUT.PUT_LINE('Value for var_num1:' ||var_num1);
--Can outer block variable (var_num1) be printed here.If Yes,Print the same.
END;
--Can inner block variable(var_num1) be printed here.If Yes,Print the same.
END;
```

#### Example 2: PL/SQL block

2.3. Write a PL/SQL block to retrieve all staff (code, name, salary) under specific department number and display the result. (Note: The Department\_Code will be accepted from user. Cursor to be used.)

2.4. Write a PL/SQL block to increase the salary by 30 % or 5000 whichever minimum for a given Department\_Code.

2.5. Write a PL/SQL block to generate the following report for a given Department code

Student_Code	Sudent_Name	Subject1	Subject2	Subject3	Total	Percentage
Grade						

Note: Display suitable error message if wrong department code has entered and if there is no student in the given department.

For Grade:

Student should pass in each subject individually (pass marks 60).

Percent  $\geq 80$  then grade= A

Percent  $\geq 70$  and  $< 80$  then grade= B

Percent  $\geq 60$  and  $< 70$  then grade= C

Else D

2.6. Write a PL/SQL block to retrieve the details of the staff belonging to a particular department. Department code should be passed as a parameter to the cursor.

## Lab 3. Exception Handling and Dynamic SQL

<b>Goals</b>	Implementing Exception Handling ,Analyzing and Debugging
<b>Time</b>	1 hr

3.1: Modify the programs created in Lab2 to implement Exception Handling

3.2 The following PL/SQL block attempts to calculate bonus of staff for a given MGR\_CODE. Bonus is to be considered as twice of salary. Though Exception Handling has been implemented but block is unable to handle the same.

Debug and verify the current behavior to trace the problem.

```
DECLARE
V_BONUS V_SAL%TYPE;
V_SAL STAFF_MASTER.STAFF_SAL%TYPE;

BEGIN
SELECT STAFF_SAL INTO V_SAL
FROM STAFF_MASTER
WHERE MGR_CODE=100006;

V_BONUS:=2*V_SAL;
DBMS_OUTPUT.PUT_LINE('STAFF SALARY IS ' || V_SAL);
DBMS_OUTPUT.PUT_LINE('STAFF BONUS IS ' || V_BONUS);

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('GIVEN CODE IS NOT VALID. ENTER VALID CODE');
END;
```

### Example 3: PL/SQL block

3.3 Rewrite the above block to achieve the requirement.



**3.4**

Predict the output of the following block ? What corrections would be needed to make it more efficient?

```
BEGIN
  DECLARE
    fname emp.ename%TYPE;
  BEGIN
    SELECT ename INTO fname
    FROM emp
    WHERE 1=2;

    DBMS_OUTPUT.PUT_LINE('This statement will print');
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Some inner block error');
  END;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No data found in fname');

  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Some outer block error');
END;
```

**Example 4: PL/SQL Block with Exception Handling****3.5** Debug the above block to trace the flow of control.

Additionally one can make appropriate changes in Select statement defined in the block to check the flow.

**3.6:** Write a PL/SQL program to check for the commission for an employee no 7369. If no commission exists, then display the error message. Use Exceptions.

**3.7:** Write a PL/SQL block to drop any user defined table.

## Lab 4. Database Programming

<b>Goals</b>	The following set of exercises are designed to implement the following <ul style="list-style-type: none"> <li>• Implement business logic using Database Programming like Procedures and Functions</li> <li>• Implement validations in Procedures and Functions</li> <li>• Working with Packages</li> <li>• Performance Tuning</li> </ul>
<b>Time</b>	3.5 Hrs

**Note:** Procedures and functions should handle validations, pre-defined oracle server and user defined exceptions wherever applicable. Also use cursors wherever applicable.

4.1 Write a PL/SQL block to find the maximum salary of the staff in the given department.  
 Note: Department code should be passed as parameter to the cursor.

4.2. Write a function to compute age. The function should accept a date and return age in years.

4.3. Write a procedure that accept staff code and update staff name to Upper case. If the staff name is null raise a user defined exception.

4.4 Write a procedure to find the manager of a staff. Procedure should return the following – Staff\_Code, Staff\_Name, Dept\_Code and Manager Name.

4.5. Write a function to compute the following. Function should take Staff\_Code and return the cost to company.

DA = 15% Salary, HRA= 20% of Salary, TA= 8% of Salary.

Special Allowance will be decided based on the service in the company.

< 1 Year	Nil
>=1 Year< 2 Year	10% of Salary
>=2 Year< 4 Year	20% of Salary
>4 Year	30% of Salary

4.6. Write a procedure that displays the following information of all staff

Staff_Name	Department Name	Designation	Salary	Status
------------	-----------------	-------------	--------	--------

Note: - Status will be (Greater, Lesser or Equal) respective to average salary of their own department. Display an error message Staff\_Master table is empty if there is no matching record.

4.7. Write a procedure that accept Staff\_Code and update the salary and store the old salary details in Staff\_Master\_Back (Staff\_Master\_Back has the same structure without any constraint) table.

Exp < 2 then no Update  
Exp > 2 and < 5 then 20% of salary  
Exp > 5 then 25% of salary

4.8. Create a procedure that accepts the book code as parameter from the user. Display the details of the students/staff that have borrowed that book and has not returned the same. The following details should be displayed

Student/Staff Code	Student/Staff Name	Issue Date	Designation	Expected Ret_Date
--------------------	--------------------	------------	-------------	-------------------

4.9. Write a package which will contain a procedure and a function.

Function: This function will return years of experience for a staff. This function will take the hiredate of the staff as an input parameter. The output will be rounded to the nearest year (1.4 year will be considered as 1 year and 1.5 year will be considered as 2 year).

Procedure: Capture the value returned by the above function to calculate the additional allowance for the staff based on the experience.

Additional Allowance = Year of experience x 3000

Calculate the additional allowance and store Staff\_Code, Date of Joining, and Experience in years and additional allowance in Staff\_Allowance table.

4.10. Write a procedure to insert details into Book\_Transaction table. Procedure should accept the book code and staff/student code. Date of issue is current date and the expected return date should be 10 days from the current date. If the expected return date falls on Saturday or Sunday, then it should be the next working day.

4.11: Write a function named 'get\_total\_records', to pass the table name as a parameter, and get back the number of records that are contained in the table. Test your function with multiple tables.

4.12

**Tune the following Oracle Procedure enabling to gain better performance.**

**Objective:** The Procedure should update the salary of an employee and at the same time retrieve the employee's name and new salary into PL/SQL variables.

```
CREATE OR REPLACE PROCEDURE update_salary (emp_id NUMBER) IS
    v_name  VARCHAR2(15);
    v_newsal NUMBER;
BEGIN
    UPDATE emp_copy SET sal = sal * 1.1
    WHERE empno = emp_id;

    SELECT ename, sal INTO v_name, v_newsal
    FROM emp_copy
    WHERE empno = emp_id;

    DBMS_OUTPUT.PUT_LINE('Emp Name:' || v_name);
    DBMS_OUTPUT.PUT_LINE('Ename:' || v_newsal);
END;
```

**Example 5: Oracle Procedure****4.13**

The following procedure attempts to delete data from table passed as parameter. This procedure has compilation errors. Identify and correct the problem.

```
CREATE or REPLACE PROCEDURE gettable(table_name in varchar2) AS
BEGIN
    DELETE FROM table_name;
END;
```

**Example 6: Oracle Procedure****4.14**

Write a procedure which prints the following report using procedure:

The procedure should take deptno as user input and appropriately print the emp details.

Also display :

Number of Employees,Total Salary,Maximum Salary,Average Salary

**Note:** The block should achieve the same without using Aggregate Functions.

Sample output for deptno 10 is shown below:

```
Employee Name : CLARK
Employee Job : MANAGER
Employee Salary : 2450
Employee Comission :
*****

Employee Name : KING
Employee Job : PRESIDENT
Employee Salary : 5000
Employee Comission :
*****

Employee Name : MILLER
Employee Job : CLERK
Employee Salary : 1300
Employee Comission :
*****

Number of Employees : 3
Total Salary : 8750
Maximum Salary : 5000
Average Salary : 2916.67
-----
```

**Figure 1 :Report**

4.15: Write a query to view the list of all procedures ,functions and packages from the Data Dictionary.

## Lab 5. Case Study 1

<b>Goals</b>	Implementation of Procedures/Functions ,Packages with Testing and Review
<b>Time</b>	2.5hrs

Consider the following tables for the case study.

### Customer\_Masters

Name	Null?	Type
Cust_Id	Not Null	Number(6)
Cust_Name	Not Null	Varchar2(20)
Address		Varchar2(50)
Date_of_acc_creation		Date
Customer_Type		Char(3)

Note: Customer type can be either IND or NRI

### Account\_Masters Table

Name	Null?	Type
Account_Number	Not Null	Number(6)
Cust_ID		Number(6)
Account_Type		Char(3)
Ledger_Balance		Number(10)

Note: Account type can be either Savings (SAV) or Salary (SAL) account.  
 For savings account minimum amount should be 5000.

### Transaction\_Masters

Name	Null?	Type
Transaction_Id	Not Null	Number(6)
Account_Number		Number(6)
Date_of_Transaction		Date
From_Account_Number	Not Null	Number(6)
To_Account_Number	Not Null	Number(6)
Amount	Not Null	Number(10)
Transaction_Type	Not Null	Char(2)

Note: Transaction type can be either Credit (CR) or Debit (DB).

Procedure and function should be written inside a package.  
All validations should be taken care.

5.1 Create appropriate Test Cases for the case study followed up by Self/Peer to Peer Review and close any defects for the same.

5.2 Write a procedure to accept customer name, address, and customer type and account type. Insert the details into the respective tables.

5.3. Write a procedure to accept customer id, amount and the account number to which the customer requires to transfer money. Following validations need to be done

- Customer id should be valid
- From account number should belong to that customer
- To account number cannot be null but can be an account which need not exist in account masters (some other account)
- Adequate balance needs to be available for debit

5.4 Ensure all the Test cases defined are executed. Have appropriate Self/Peer to Peer Code Review and close any defects for the same.

## Lab 6. Case Study 2

<b>Goals</b>	Implementation of Procedures/Functions ,Packages with Testing and Review
<b>Time</b>	2.5hrs

Consider the following table (myEmp) structure for the case study

EmpNo Ename City Designation Salary

-----

The following procedure accepts Task number and based on the same performs an appropriate task.

```

PROCEDURE run_task (task_number_in IN INTEGER)
IS
BEGIN
  IF task_number_in = 1
  THEN
    add_emp;
    --should add new emps in myEmp.
    --EmpNo should be inserted through Sequence.
    --All other data to be taken as parameters.Default location is Mumbai.
  END IF;
  IF task_number_in = 2
  THEN
    raise_sal;
    --should modify salary of an existing emp.
    --should take new salary and empno as input parameters
    --Should handle exception in case empno not found
    --upper limit of raising salary is 30%. should raise exception appropriately
  END IF;
  IF task_number_in = 3
  THEN
    remove_emp;
    --should remove an existing emp
    --should take empno as parameter
  
```



```
--Handle exception if empno not available  
END IF;  
END run_task;
```

### Example 7: Sample Oracle Procedure

However ,it has been observed the method adopted in above procedure is inefficient.

#### 6.1

Create appropriate Test Cases for the case study followed up by Self/Peer to Peer Review and close any defects for the same.

#### 6.2

Recreate the procedure (run\_task) which is more efficient in performing the same.

#### 6.3

Also, create relevant procedures (add\_emp , raise\_sal ,remove\_emp) with relevant logic (read comments)to verify the same.

#### 6.4 Extend the above implementation using Packages

6.5) Ensure all the Test cases defined are executed. Have appropriate Self/Peer to Peer Code Review and close any defects for the same.

## Lab 7. Handling Files, DBMS\_LOB

<b>Goals</b>	Working with UTL_FILE and subprograms of this package
<b>Time</b>	2 hr 30 mins

7.1: The following PL/SQL block creates file “TestFile.txt” with appropriate contents. Enhance the block by reading the contents back from the file and displaying it at SQL prompt.

```

Declare
  TextHandler Utl_File.File_type;
  WriteMessage Varchar2(400);
  ReadMessage Varchar2(400);
Begin
  TextHandler:=Utl_File.Fopen('d:\Sample','TestFile.txt','W');
  WriteMessage:='FOPEN is a Function, which returns the value of type
  File_Type \n UTL_file.PUT_LINE is a procedure in UTL_FILE, which write a line
  to a file,Specific line terminator will be appended \n';

  Utl_file.Putf(Texthandler,writemessage);
  Utl_File.Fflush(Texthandler);
  Utl_File.Fclose(Texthandler);

End;
/
  
```

### Example 8: Block using File Handling operations

7.2 Extend the implementation in the above block by incorporating Exception Handling.

7.3 We need to maintain the above block in database permanently.What can be done for the same ? Rewrite the above to achieve the same.

7.4: Write a PL/SQL block to create a file “EmpDeptDetails.Txt” .The contents in the file map to columns from both the tables Emp and Dept.  
 (Empno,Ename,Job, Sal, Dname, Loc)  
 Also read the contents back from the file and display it on prompt.

## Lab 8. SQL\*Plus Reports

**Note:** Demos are provided for additional reference.

<b>Goals</b>	Use SQL*Plus Reports feature and come up with reports in specified formats.
<b>Time</b>	1 hr

### 8.1: Using Multiple Spacing Techniques

Suppose you have more than one column in your ORDER BY clause and wish to insert space when each column's value changes. Each BREAK command you enter replaces the previous one.

Now consider a scenario where you want to do either of the following:

- to use different spacing techniques in one report, or
- to insert space after the value changes in more than one ordered column

Then you must specify “multiple columns” and “actions” in a single BREAK command.

**Step 1:** Combine the Spacing Techniques.

```
SELECT DEPARTMENT_ID, JOB_ID, LAST_NAME, SALARY FROM
EMP_DETAILS_VIEW
WHERE SALARY>12000
ORDER BY DEPARTMENT_ID, JOB_ID;
```

#### Example 9: Sample Code

Now, to skip a page when the value of DEPARTMENT\_ID changes, and to skip one line when the value of JOB\_ID changes, key in the following command:

```
BREAK ON DEPARTMENT_ID SKIP PAGE ON JOB_ID SKIP 1
```

#### Example 10: Sample Code

To show that SKIP PAGE has taken effect, create a TTITLE with a page number:

```
TTITLE COL 35 FORMAT 9 'Page:' SQL.PNO
```

#### Example 101: Sample Code

Page: 1			
DEPARTMENT_ID	JOB_ID	LAST_NAME	SALARY
20	MK_MAN	Hartstein	13000
Page: 2			
DEPARTMENT_ID	JOB_ID	LAST_NAME	SALARY
80	SA_MAN	Russell	14000
	Partners		13500
Page: 3			
DEPARTMENT_ID	JOB_ID	LAST_NAME	SALARY
90	AD_PREP	King	24000
	AD_VP	Kochhar	17000
	De Haan		17000
6 rows selected.			

**Figure 2: Report**

**Step 2:** Produce a report that does the following when the value of JOB\_ID changes:

- prints duplicate job values,
- prints the average of SALARY, and
- inserts one blank line

Additionally the report should do the following when the value of DEPT\_ID changes:

- prints the sum of SALARY, and
- inserts another blank line

The details should be displayed for all departments respective to jobs. **(To Do)**

DEPT_ID	JOB_ID	ENAME	SALARY
50	SH_CLERK	Taylor	3200
	SH_CLERK	Fleaur	3100
.			
.			
.	SH_CLERK	Gates	2900
*****			
	Avg:		3000
DEPT_ID	JOB_ID	ENAME	SALARY
50	SALESMAN	Perkins	2500
	SALESMAN	Bell	4000
.			
.			
.	SALESMAN	Grant	2600
*****			
	Avg:		3215
DEPT_ID	JOB_ID	ENAME	SALARY
*****			
sum			64300
-			
-			
-			
-			
-			
25 rows selected.			

**Figure 3: Report**

## 8.2: Computing and Printing Subtotals

**Step 1:** Generate SQL Report in the following format.

SALES DEPARTMENT PERSONNEL REPORT		
PERFECT WIDGETS		
		01-JAN-2008 PAGE: 1
DEPARTMENT_ID	LAST_NAME	SALARY
20	Hartstein	13000
80	Russell	14000
80	Partners	13500
90	King	24000
90	Kochhar	17000
90	De Haan	17000
		-----
		98500
COMPANY CONFIDENTIAL		
6 rows selected.		

**Figure 4: SQL Report**

**Step 2:** Generate SQL Report in the following format.

Dept No.	Job Name	No. of Employees	Average Salary/Job
10	SALES	4	\$13,000.00
	CLEARC	2	\$10,666.00
20	MANAGER	3	\$14,000.00
	SALES	6	\$11,000.00
30	CLERK	10	\$13,500.00
	MANAGER	3	\$15,000.00
	SALES	4	\$10,000.00
40	PRESIDENT	1	424,000.00
			-----
Grand Total of Sal:			\$98,560.00
			-----
No. Of Employees:			100

**Figure 5: SQL Report**

## Lab 9. SQL Loader

**Note:** Demos are provided for additional reference.

<b>Goals</b>	<ul style="list-style-type: none"><li>• Use the SQL Loader utility and upload the given files under specified conditions.</li></ul>
<b>Time</b>	1 hr

### 9.1: Load Variable-Length Data

**Step 1:** Crosscheck for the relation. If it does not exist, create the Dept table.

**Step 2:** Create and save the control file named as 'deptcontrol.ctl' in the specified location. Control file should contain following details:

```
12,RESEARCH,"SARATOGA"  
10,"ACCOUNTING",CLEVELAND  
11,"ART",SALEM  
13,FINANCE,"BOSTON"  
21,"SALES",PHILA.  
22,"SALES",ROCHESTER  
42,"INT'L","SAN FRAN
```

Example 112: Control File

**Step 3:** Execute the SQL Loader command at the command prompt and verify the updated Relation by issuing the command given below:

```
SQL>select * from dept;
```

Example 13: Sample Code

9.2: Use the SQL Loader utility. Upload the data file having Fixed-Format Fields, as shown below, into Emp table.

#### Data file

- Given below are a few sample data lines from the file ulcase2.dat.
- Blank fields are automatically set to null.

7782	CLARK	MANAGER	7839	2572.50		10
7839	KING	PRESIDENT		5500.00		10
7934	MILLER	CLERK	7782	920.00		10
7566	JONES	MANAGER	7839	3123.75		20
7499	ALLEN	SALESMAN	7698	1600.00	300.00	30
7654	MARTIN	SALESMAN	7698	1312.50	1400.00	30
7658	CHAN	ANALYST	7566	3450.00		20
7654	MARTIN	SALESMAN	7698	1312.50	1400.00	30

Example 124: Sample Code

9.3: Load combined physical records

#### Control file:

The control file is ulcase4.ctl:

```

LOAD DATA
  INFILE 'ulcase4.dat'
  1) DISCARDFILE 'ulcase4.dsc'
  2) DISCARDMAX 999
  3) REPLACE
  4) CONTINUEIF THIS (1) = '*'
  INTO TABLE emp
  (empno      POSITION(1:4)          INTEGER EXTERNAL,
   ename      POSITION(6:15)         CHAR,
   job        POSITION(17:25)        CHAR,
   mgr        POSITION(27:30)        INTEGER EXTERNAL,
   sal        POSITION(32:39)        DECIMAL EXTERNAL,
   comm       POSITION(41:48)        DECIMAL EXTERNAL,
   deptno     POSITION(50:51)        INTEGER EXTERNAL,
   hiredate   POSITION(52:60)        INTEGER EXTERNAL)
  
```

Example 135: Control file



## Datafile

A sample datafile used for this case, ulcase4.dat, is shown below. Asterisks are in the first position. Further, a newline character, though not visible, is in position 20. Note that clerk's commission is -10, and SQL\*Loader loads the value, converting it to a negative number.

```
*7782 CLARK
MANAGER 7839 2572.50 -10 25 12-NOV-85
*7839 KING
PRESIDENT 5500.00 25 05-APR-83
*7934 MILLER
CLERK 7782 920.00 25 08-MAY-80
*7566 JONES
MANAGER 7839 3123.75 25 17-JUL-85
*7499 ALLEN
SALESMAN 7698 1600.00 300.00 25 3-JUN-84
*7654 MARTIN
SALESMAN 7698 1312.50 1400.00 25 21-DEC-85
*7658 CHAN
ANALYST 7566 3450.00 25 16-FEB-84
*CHEN
ANALYST 7566 3450.00 25 16-FEB-84
*7658 CHIN
ANALYST 7566 3450.00 25 16-FEB-84
```

Example 146: Datafile

## 9.4: Load data into multiple tables.

The control file is ulcase5.ctl.

```
-- Loads EMP records from first 23 characters
-- Creates and loads PROJ records for each PROJNO listed
-- for each employee
LOAD DATA
INFILE 'ulcase5.dat'
BADFILE 'ulcase5.bad'
DISCARDFILE 'ulcase5.dsc'
1) REPLACE
2) INTO TABLE emp
(empno POSITION(1:4) INTEGER EXTERNAL,
ename POSITION(6:15) CHAR,
deptno POSITION(17:18) CHAR,
mgr POSITION(20:23) INTEGER EXTERNAL)
2) INTO TABLE proj
-- PROJ has two columns, both not null: EMPNO and PROJNO
3) WHEN projno != ' '
```

```

(empno POSITION(1:4) INTEGER EXTERNAL,
3) projno POSITION(25:27) INTEGER EXTERNAL) -- 1st proj
2) INTO TABLE proj
4) WHEN projno != ' '
(empno POSITION(1:4) INTEGER EXTERNAL,
4) projno POSITION(29:31) INTEGER EXTERNAL) -- 2nd proj

2) INTO TABLE proj
5) WHEN projno != ' '
(empno POSITION(1:4) INTEGER EXTERNAL,
5) projno POSITION(33:35) INTEGER EXTERNAL) -- 3rd proj

```

Example 157: Control File

**Notes:**

1. REPLACE specifies that if there is data in the tables to be loaded (emp and proj), SQL\*loader should delete the data before loading new rows.
2. Multiple INTO TABLE clauses load two tables, emp and proj. The same set of records is processed three times, using different combinations of columns each time to load table proj.
3. WHEN loads only rows with nonblank project numbers. When projno is defined as columns 25...27, rows are inserted into proj only if there is a value in those columns.
4. When projno is defined as columns 29...31, rows are inserted into proj only if there is a value in those columns.
5. When projno is defined as columns 33...35, rows are inserted into proj only if there is a value in those columns.

**Datafile:**

1234 BAKER	10 9999 101 102	103
1234 JOKER	10 9999 777 888	999
2664 YOUNG	20 2893 425 abc	102
5321 OTOOLE	10 9999 321 55	40
2134 FARMER	20 4555 236 456	
2414 LITTLE	20 5634 236 456	40
6542 LEE	10 4532 102 321	14
2849 EDDS	xx 4555 294	40
4532 PERKINS	10 9999	40
1244 HUNT	11 3452 665 133	456
123 DOOLITTLE	12 9940	132

1453 MACDONALD 25 5532	200
------------------------	-----

## Example 18: Datafile

9.5: Use SQL Loader utility to load specific data from the given flat file into the table. (To Do)

The given flat file contains data in the form of DEPTNO, DEPTNAME, and LOCATION. Using SQL Loader utility, load only DEPTNO and LOCATION from the .dat file into the department table. Do not overwrite the source data present in department table.

12, Research, "Saratoga" 10, "Accounting", Cleveland 11, "Art", Salem 13, Finance, Boston 21, Sales, Phila 22, Sales, Rochester 42, "Int'l", "San Fran"
---

## Example19: Flat file

## Appendices

### Appendix A: Oracle Standards

#### Key points to keep in mind:

1. Write comments in your stored Procedures, Functions and SQL batches generously, whenever something is not very obvious. This helps other programmers to clearly understand your code. Do not worry about the length of the comments, as it will not impact the performance.
2. Prefix the table names with owner names, as this improves readability, and avoids any unnecessary confusion.

#### Some more Oracle standards:

To be shared by Faculty in class

## Appendix B: Coding Best Practices

1. Avoid Dynamic SQL statements as much as possible. Dynamic SQL tends to be slower than Static SQL.
2. Perform all your referential integrity checks and data validations by using constraints (foreign key and check constraints). These constraints are faster than triggers. So use triggers only for auditing, custom tasks, and validations that cannot be performed by using these constraints.
3. Do not call functions repeatedly within your stored procedures, triggers, functions, and batches. For example: You might need the length of a string variable in many places of your procedure. However do not call the LENGTH function whenever it is needed. Instead call the LENGTH function once, and store the result in a variable, for later use.

**Appendix C: Table of Figures**

Figure 1: <a href="#">Report</a> .....	20
Figure 2: Report.....	200
Figure 3: Report.....	211
Figure 4: SQL Report .....	222
Figure 5: SQL Report .....	223

**Appendix D: Table of Examples**

Example 1: Declaration Block.....	6
Example 2: PL/SQL block.....	6
Example 3: PL/SQL block.....	8
Example 4: PL/SQL Block with Exception Handling.....	9
Example 5: Oracle Procedure .....	12
Example 6: Oracle Procedure .....	12
Example 7: Sample Oracle Procedure .....	17
Example 8: Block using File Handling operations .....	18
Example 9: Sample Code .....	19
Example 11: Sample Code .....	19
Example 12: Control File .....	23
Example 14: Sample Code .....	24
Example 15: Control file .....	24
Example 16: Datafile .....	25
Example 17: Control File .....	26