

Lab 4 Data Update Operation and PL/SQL Introduction

I. Experiment Objectives

1. Learn to use INSERT, UPDATE, and DELETE to insert, modify, and delete data;
2. Learn to use substitution variables to implement interactive parameterized queries;
3. Master the basic structure and syntax of Procedural SQL (PL/SQL) and be able to write anonymous code blocks with input, check, and output logic to automate small business logic.

II. Experiment Environment

- **Database:** Oracle Database
- **Tool:** SQL Developer
- **Required Database Tables:** EMPLOYEES, CUSTOMERS, ORDERS, SHIPPERS
These tables were created in Lab 3. If not, execute Dummy Database.sql to create the tables and load the data.

III. Experiment Content and Steps

Task 1: Data Update Operations (INSERT/UPDATE/DELETE)

1. Example: Inserting Data (INSERT)

```
-- Insert 2 new employee records
INSERT INTO EMPLOYEES (EMPLOYEEID, LASTNAME, FIRSTNAME, TITLE, HIREDATE,
REPORTSTO)
VALUES (11, 'Zhang', 'San', 'Sales Representative', TO_DATE('2023-01-15','YYYY-MM-
DD'), 5);
INSERT INTO EMPLOYEES (EMPLOYEEID, LASTNAME, FIRSTNAME, TITLE, HIREDATE,
REPORTSTO)
VALUES (12, 'Li', 'Si', 'Sales Representative', TO_DATE('2023-12-17','YYYY-MM-DD'),
5);
```

```
-- Check whether the insertion is successful
SELECT EMPLOYEEID, LASTNAME, FIRSTNAME, TITLE, HIREDATE FROM EMPLOYEES
WHERE EMPLOYEEID IN (11, 12);
```

| | EMPLOYEEID | LASTNAME | FIRSTNAME | TITLE | HIREDATE |
|---|------------|----------|-----------|----------------------|-----------|
| 1 | 11 | Zhang | San | Sales Representative | 15-JAN-23 |
| 2 | 12 | Li | Si | Sales Representative | 17-DEC-23 |

2. Example: Updating Data (UPDATE)

Increase shipping freight by 10% for orders shipped by shipper "Speedy Express" (SHIPVIA = 1).

-- 1. Show the freight of orders shipped by "Speedy Express"(SHIPVIA = 1)

```
SELECT ORDERID, CUSTID, SHIPVIA, FREIGHT
FROM ORDERS
WHERE SHIPVIA = 1;
```

| | ORDERID | CUSTID | SHIPVIA | FREIGHT |
|----|---------|--------|---------|---------|
| 1 | 10249 | TOMSP | 1 | 11.61 |
| 2 | 10258 | ERNSH | 1 | 140.51 |
| 3 | 10265 | BLONP | 1 | 55.28 |
| 4 | 10267 | FRANK | 1 | 208.58 |
| 5 | 10269 | WHITC | 1 | 4.56 |
| 6 | 10274 | VINET | 1 | 6.01 |
| 7 | 10275 | MAGAA | 1 | 26.93 |
| 8 | 10280 | BERGS | 1 | 8.98 |
| 9 | 10282 | ROMEY | 1 | 12.69 |
| 10 | 10284 | LEHMS | 1 | 76.56 |
| 11 | 10288 | REGGC | 1 | 7.45 |
| 12 | 10290 | COMMI | 1 | 79.7 |
| 13 | 10296 | LILAS | 1 | 0.12 |
| 14 | 10309 | HUNGO | 1 | 47.3 |
| 15 | 10317 | LONEP | 1 | 12.69 |
| 16 | 10323 | KOENE | 1 | 4.88 |
| 17 | 10324 | SAVEA | 1 | 214.27 |
| 18 | 10327 | FOLKO | 1 | 63.36 |
| 19 | 10330 | LILAS | 1 | 12.75 |
| 20 | 10343 | LEHMS | 1 | 110.37 |
| 21 | 10349 | SPLIR | 1 | 8.63 |

-- 2. Increase shipping freight for orders shipped by designated carriers

```
UPDATE ORDERS
SET FREIGHT = FREIGHT * 1.10
WHERE SHIPVIA = 1;
```

-- 3. Show the freight of orders shipped by "Speedy Express"(SHIPVIA = 1) again

| | ORDERID | CUSTID | SHIPVIA | FREIGHT |
|----|---------|--------|---------|---------|
| 1 | 10249 | TOMSP | 1 | 12.771 |
| 2 | 10258 | ERNSH | 1 | 154.561 |
| 3 | 10265 | BLONP | 1 | 60.808 |
| 4 | 10267 | FRANK | 1 | 229.438 |
| 5 | 10269 | WHITC | 1 | 5.016 |
| 6 | 10274 | VINET | 1 | 6.611 |
| 7 | 10275 | MAGAA | 1 | 29.623 |
| 8 | 10280 | BERGS | 1 | 9.878 |
| 9 | 10282 | ROMEY | 1 | 13.959 |
| 10 | 10284 | LEHMS | 1 | 84.216 |
| 11 | 10288 | REGGC | 1 | 8.195 |
| 12 | 10290 | COMMI | 1 | 87.67 |
| 13 | 10296 | LILAS | 1 | 0.132 |
| 14 | 10309 | HUNGO | 1 | 52.03 |
| 15 | 10317 | LONEP | 1 | 13.959 |
| 16 | 10323 | KOENE | 1 | 5.368 |
| 17 | 10324 | SAVEA | 1 | 235.697 |
| 18 | 10327 | FOLKO | 1 | 69.696 |
| 19 | 10330 | LILAS | 1 | 14.025 |
| 20 | 10343 | LEHMS | 1 | 121.407 |
| 21 | 10349 | SPLIR | 1 | 9.493 |

3. Example: Deleting Data (DELETE)

-- 1. Query unshipped orders with an order date earlier than '2004-08-01'.

```
SELECT * FROM ORDERS
WHERE SHIPPEDDATE IS NULL
AND ORDERDATE < DATE '2004-08-01';
```

| ORDERID | CUSTID | EMPLOYEEID | ORDERDATE | REQUIREDDATE | SHIPPEDDATE | SHIPVIA | FREIGHT | SHIPNAME | SHIPADDRESS | SHIPCITY | SHIPREGION | SHIPPOSTALCODE | SHIPCOUNTRY |
|---------|-------------|------------|-------------|--------------|-------------|---------|----------|----------|-------------|----------|------------|----------------|-------------|
| 1 | 11000 ANATR | | 7 01-JUL-04 | 15-JUL-04 | (null) | 1 | 0 (null) | (null) | (null) | (null) | (null) | (null) | Mexico |

-- 2. Delete unshipped orders with an order date earlier than '2004-08-01'.

```
DELETE FROM ORDERS
WHERE SHIPPEDDATE IS NULL
AND ORDERDATE < DATE '2004-08-01';
```

-- 3. Query unshipped orders with an order date earlier than '2004-08-01' again.

```
SELECT * FROM ORDERS
WHERE SHIPPEDDATE IS NULL
AND ORDERDATE < DATE '2004-08-01';
```

| ORDERID | CUSTID | EMPLOYEEID | ORDERDATE | REQUIREDDATE | SHIPPEDDATE | SHIPVIA | FREIGHT | SHIPNAME | SHIPADDRESS | SHIPCITY | SHIPREGION | SHIPPOSTALCODE | SHIPCOUNTRY |
|---------|--------|------------|-----------|--------------|-------------|---------|---------|----------|-------------|----------|------------|----------------|-------------|
|---------|--------|------------|-----------|--------------|-------------|---------|---------|----------|-------------|----------|------------|----------------|-------------|

Task 2: Procedural SQL

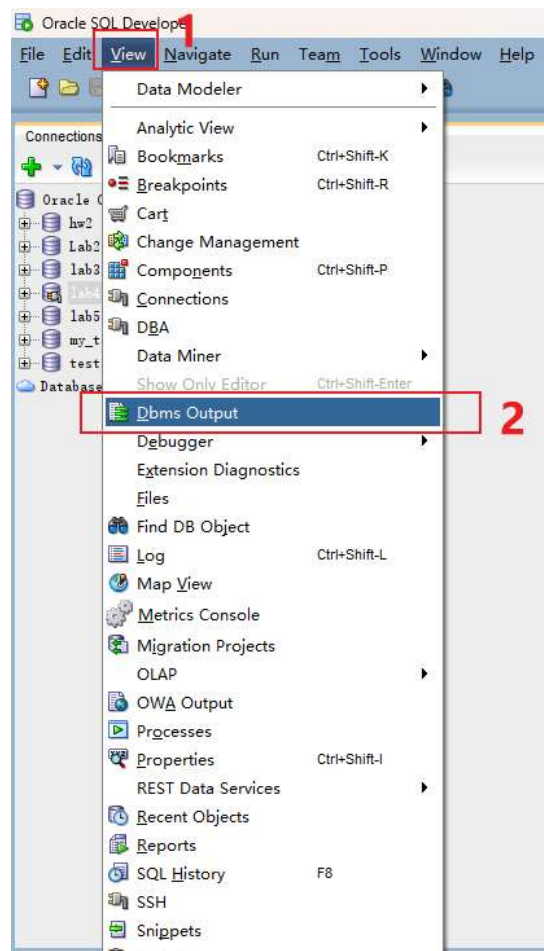
PL/SQL is a procedural language designed to extend SQL. It can organize multiple SQL statements and add program structures such as conditionals and loops to complete complex business tasks. Below we will briefly introduce how to enable the program to read user keyboard input and execute PL/SQL code in SQL Developer, as well as how to view the output results of DBMS_OUTPUT.

1. View the output results of DBMS_OUTPUT.

When running PL/SQL code in SQL Developer, if you want to see the output of DBMS_OUTPUT.PUT_LINE(), you need to open the output panel first.

Steps:

(1) Open **View** → **Dbms Output** in the menu bar:



(2) In the pop-up "DBMS Output" panel, click the "+" button, select the current database connection, and make sure the status is Enabled.



2. Example 1: Counting Order Quantity and Total Shipping Freight by Customer

Business rules: A salesperson wants to enter a customer ID and quickly view the total number of orders and accumulated shipping freight for that customer.

```
SET SERVEROUTPUT ON
DECLARE
v_custid CUSTOMERS.CUSTID%TYPE := '&custid'; -- e.g., ANATR
v_count NUMBER;
v_sum_freight NUMBER;
BEGIN
SELECT COUNT(*), NVL(SUM(FREIGHT), 0)
INTO v_count, v_sum_freight
FROM ORDERS
WHERE CUSTID = v_custid;

DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_custid);
DBMS_OUTPUT.PUT_LINE('Total Orders: ' || v_count);
DBMS_OUTPUT.PUT_LINE('Total Freight: ' || ROUND(v_sum_freight,2));
END;
/
```

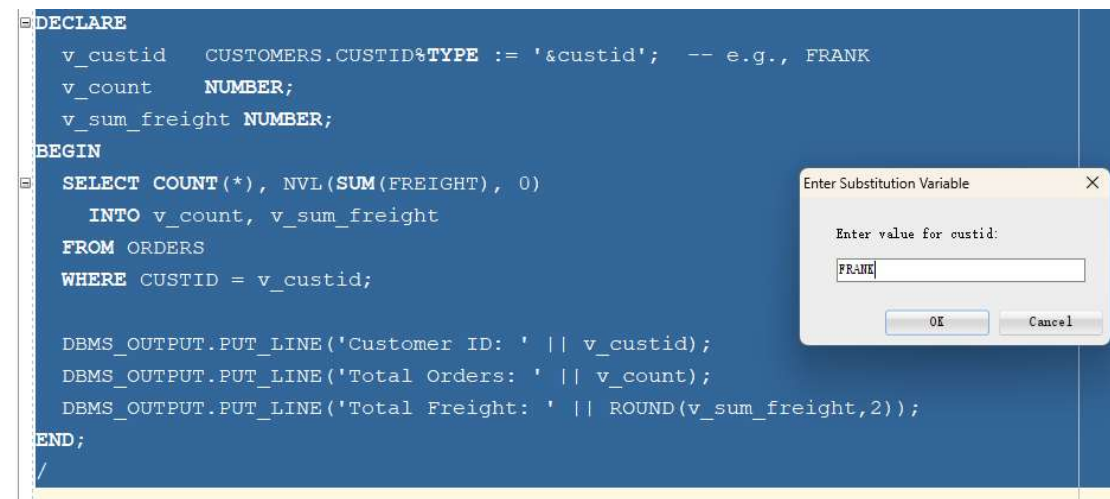
Note that SQL Developer supports variable substitution (a pop-up input box appears during runtime for user input).

- **&variablename**: Prompts for input each time the statement is executed.
- **&&variablename**: Reuses the variable after the initial entry within the current session.
- **UNDEFINE variablename**: Releases the variable, causing it to prompt for input again the next time the statement is executed.

Therefore, to implement **interactive queries** where salespeople enter a customer ID to retrieve information about a specific customer, you need to introduce a **substitution variable**:

```
v_custid CUSTOMERS.CUSTID%TYPE := '&custid'; -- e.g., FRANK
```

Executing the statement will prompt a dialog box for the customer ID. After entering the CUSTID (e.g., "FRANK"), the value you entered will replace the '&custid' portion and be assigned to the variable v_custid:



The screenshot shows a SQL Developer window with a PL/SQL block. The block declares variables v_custid, v_count, and v_sum_freight. It then executes a SELECT statement to retrieve customer information based on the substitution variable &custid. A dialog box titled "Enter Substitution Variable" is open, prompting the user to enter a value for custid. The user has entered "FRANK".

```
DECLARE
v_custid CUSTOMERS.CUSTID%TYPE := '&custid'; -- e.g., FRANK
v_count NUMBER;
v_sum_freight NUMBER;
BEGIN
SELECT COUNT(*), NVL(SUM(FREIGHT), 0)
INTO v_count, v_sum_freight
FROM ORDERS
WHERE CUSTID = v_custid;

DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_custid);
DBMS_OUTPUT.PUT_LINE('Total Orders: ' || v_count);
DBMS_OUTPUT.PUT_LINE('Total Freight: ' || ROUND(v_sum_freight,2));
END;
```

Viewing the DBMS output will display the order quantity and cumulative shipping freight for the customer you entered (FRANK):



The screenshot shows the DBMS Output window with the following output:

```
Customer ID: FRANK
Total Orders: 4
Total Freight: 527.88
```

3. Example 2: Determine if a customer is a high-value customer

Business rules: A salesperson wants to input a customer ID and calculate the

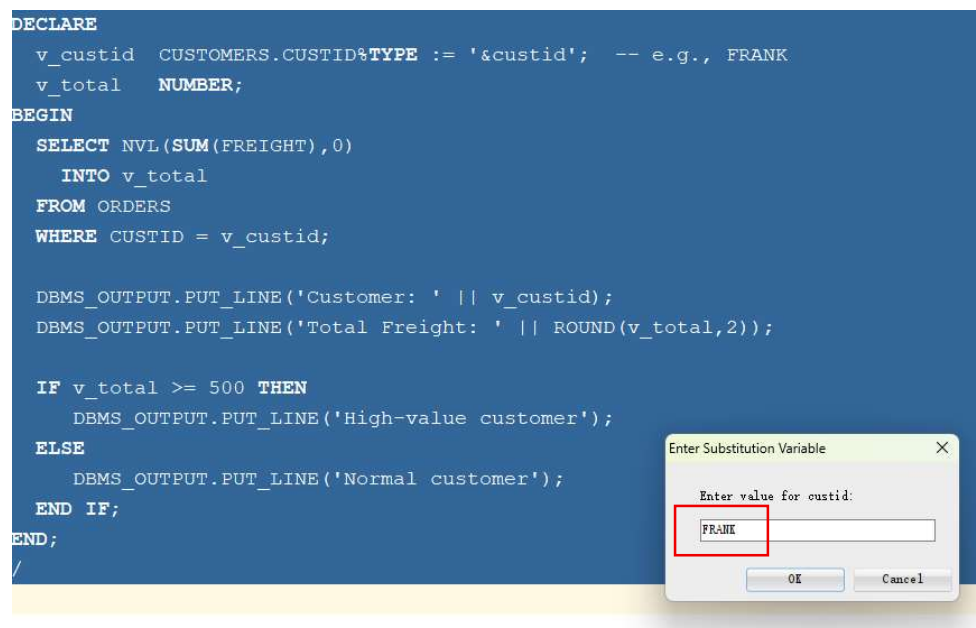
total shipping freight for that customer. If the total shipping freight is not less than 500, then output "High-value customer"; otherwise, output "Normal customer."

```
SET SERVEROUTPUT ON
DECLARE
v_custid CUSTOMERS.CUSTID%TYPE := '&custid'; -- e.g., ANATR
v_total NUMBER;
BEGIN
SELECT NVL(SUM(FREIGHT),0)
INTO v_total
FROM ORDERS
WHERE CUSTID = v_custid;

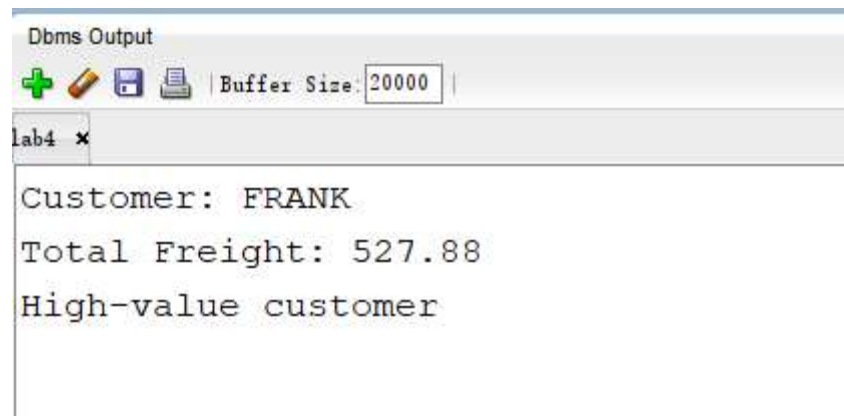
DBMS_OUTPUT.PUT_LINE('Customer: ' || v_custid);
DBMS_OUTPUT.PUT_LINE('Total Freight: ' || ROUND(v_total,2));

IF v_total >= 500 THEN
DBMS_OUTPUT.PUT_LINE('High-value customer');
ELSE
DBMS_OUTPUT.PUT_LINE('Normal customer');
END IF;
END;
/
```

Execute the statement and enter CUSTID (e.g., "FRANK") in the dialog box that pops up:



View the DBMS Output to display the cumulative freight of the customer you entered and determine whether it is a high-value customer:



The screenshot shows a window titled "Dbms Output" with a toolbar containing icons for adding, editing, saving, and printing, along with a "Buffer Size" field set to 20000. Below the toolbar is a tab labeled "lab4". The output text is as follows:

```
Customer: FRANK  
Total Freight: 527.88  
High-value customer
```