

## Database Programming with PL/SQL

### 12-2: Improving PL/SQL Performance

#### Practice Activities

##### Vocabulary

Identify the vocabulary word for each definition below:

	passes arguments by reference rather than by value, and usually speeds up the execution of SQL statements.
	provides bulk processing for DML activity
	provides bulk processing for SELECT and FETCH statements
	means that the same input value will always produce the same output value, and must be used to create a function-based index on your own functions.
	allows the retrieval of data modified by a DML statement without triggering a separate context switch
	fetches all the rows in a single call to the SQL Engine.

##### Try It / Solve It

1. Run this code to load 25,000 records into a local nested table and pass these values to two local procedures that do nothing. Notice the call to the subprogram using NOCOPY. What are the results?

```
CREATE OR REPLACE PACKAGE nocopy_test AS
TYPE EmpTabTyp IS TABLE OF employees%ROWTYPE;
emp_tab EmpTabTyp := EmpTabTyp(NULL);
PROCEDURE get_time (t OUT NUMBER);
PROCEDURE do_nothing1 (tab IN OUT EmpTabTyp);
PROCEDURE do_nothing2 (tab IN OUT NOCOPY EmpTabTyp);
END nocopy_test;
```

```

CREATE OR REPLACE PACKAGE BODY nocopy_test AS
PROCEDURE get_time (t OUT NUMBER) IS
BEGIN
t := DBMS_UTILITY.get_time;
END;
PROCEDURE do_nothing1 (tab IN OUT EmpTabTyp) IS
BEGIN
NULL;
END;
PROCEDURE do_nothing2 (tab IN OUT NOCOPY EmpTabTyp) IS
BEGIN
NULL;
END;
END nocopy_test;

```

```

DECLARE
t1 NUMBER;
t2 NUMBER;
t3 NUMBER;
BEGIN
SELECT * INTO nocopy_test.emp_tab(1) FROM EMPLOYEES
WHERE employee_id = 100;
nocopy_test.emp_tab.EXTEND(49999, 1); -- Copy element 1 into 2..50000
nocopy_test.get_time(t1);
nocopy_test.do_nothing1(nocopy_test.emp_tab); -- Pass IN OUT parameter
nocopy_test.get_time(t2);
nocopy_test.do_nothing2(nocopy_test.emp_tab); -- Pass IN OUT NOCOPY parameter
nocopy_test.get_time(t3);
DBMS_OUTPUT.PUT_LINE ('Call Duration (secs)');
DBMS_OUTPUT.PUT_LINE ('-----');
DBMS_OUTPUT.PUT_LINE ('Just IN OUT: ' || TO_CHAR((t2 - t1)/100.0));
DBMS_OUTPUT.PUT_LINE ('With NOCOPY: ' || TO_CHAR((t3 - t2)/100.0));
END;

```

2. Run the following PL/SQL program which increases the salary for employees with IDs 100, 102, 104, or 110. The FORALL statement bulk-binds the collection. What are the results?

```

CREATE OR REPLACE PROCEDURE raise_salary (p_percent NUMBER) IS
    TYPE numlist_type IS TABLE OF NUMBER
    INDEX BY BINARY_INTEGER;
    v_id          numlist_type; -- collection
BEGIN
    v_id(1) := 100;
    v_id(2) := 102;
    v_id(3) := 104;
    v_id(4) := 110;
    -- bulk-bind the associative array
    FORALL i IN v_id.FIRST .. v_id.LAST
        UPDATE employees
            SET salary = (1 + p_percent / 100) * salary
            WHERE employee_id = v_id (i);
END;

```

Execute the following SELECT statement to find out salaries before executing the raise\_salary procedure:

```

SELECT salary
FROM employees
WHERE employee_id = 100 OR employee_id = 102
   OR employee_id = 104 OR employee_id = 100;

```

Execute the raise\_salary procedure and verify the results.

```

BEGIN
    raise_salary(10);
END;

```

```

SELECT salary
FROM employees
WHERE employee_id = 100 OR employee_id = 102
   OR employee_id = 104 OR employee_id = 100;

```

3. Create and execute a procedure called `get_departments` that obtains all rows from the `DEPARTMENTS` table for a specific location using the `BULK COLLECT` clause.
4. Create and execute an anonymous block containing the `BULK COLLECT` and `RETURNING` clause that deletes all employees in `department_id` 20 from the `EMP_TEMP` table. Create the `EMP_TEMP` table from the `EMPLOYEES` table. Your anonymous block should produce results that look similar to this (your results may vary depending on previous changes you may have made to the `EMPLOYEES` table):

Results	Explain	Describe
---------	---------	----------

Deleted 7 rows:

Employee #201: Hartstein

Employee #202: Fay

Employee #215: Steiner

Employee #217: TAYLOR

Employee #219: Stocks

Employee #228: Safwah

Employee #235: Newton

1 row(s) deleted.