PL/SQL

1. PL/SQL is a very secure **functionality tool** for manipulating, controlling, validating, and restricting unauthorized access data from the SQL database.
2. Using PL/SQL we can improve **application performance**. It also allows to **deal** with **errors** so we can provide **user friendly error messages**.
3. PL/SQL have a **great functionality** to display multiple records from the multiple tables at the same time.
4. PL/SQL is **capable** to **send** entire block of statements and execute it in the **engine** at once.
5. Procedural language support: PL/SQL is a **development tools** not only for data manipulation futures but also provide the conditional checking, looping or branching operations same as like **other programming language**.
6. Reduces network traffic: This one is **great advantages** of PL/SQL. Because PL/SQL nature is **entire block** of SQL statements execute into **oracle engine** all at once so it's main benefit is **reducing** the **network traffic**.
7. Error handling: PL/SQL is dealing with **error handling**, It's permits the smart way **handling the errors** and giving **user friendly** error messages, when the errors are encountered.
8. Declare variable: PL/SQL gives you control to **declare variables** and access them **within the block**. The declared variables can be used at the time of **query processing**.

9. Intermediate Calculation: Calculations in PL/SQL done quickly and efficiently without using Oracle engines. This **improves** the transaction performance.
10. Portable application: Applications are written in PL/SQL are **portable** in any **Operating system**. PL/SQL applications are **independence program** to run any computer.
11. PL/SQL is structured as it consists of blocks of code and hence streamlined. This makes PL/SQL highly productive.
12. It is highly portable, has immense error handling mechanisms.
13. High performance as lines of code can be sent to oracle. This reduces traffic.
14. With the user of stored procedures, PL/SQL is highly secured.
15. Extremely flexible and easy to learn with syntaxes like SELECT, INSERT, UPDATE etc.
16. Support SQL data manipulation.
17. Provide facilities like conditional checking, branching and looping.
18. Provide fast code execution since it sends SQL statement as a block to the oracle engine.

Data type of PL/SQL

There are various type of data types:

1. Row id Data types
2. Character Data types
3. Numeric Data types
4. Date/Time Data types

https://docs.oracle.com/cd/A97630_01/appdev.920/a96624/03_types.htm

https://docs.oracle.com/cd/B28359_01/appdev.111/b28370/datatypes.htm#CIHCEDIA

Row id Datatypes:

Data type Syntax	Oracle 10g	Explanation
Rowid	The format of the rowid is: BBBBBBB.RRRR.FFFFF Where BBBBBBB is the block in the database file; RRRR is the row in the block; FFFFF is the database file.	Fixed-length binary data. Every record in the database has a physical address or rowid.
Urowid(size)		Universal rowid. Where size is optional.

Character Datatypes :

Data Type Syntax	Oracle 10g	Explanation
char(size)	Maximum size of 2000 bytes.	Where size is the number of characters to store. Fixed-length strings. Space padded.
nchar(size)	Maximum size of 2000 bytes.	Where size is the number of characters to store. Fixed-length NLS string Space padded.
nvarchar2(size)	Maximum size of 4000 bytes.	Where size is the number of characters to store. Variable-length NLS string.
Varchar2(size)	Maximum size of 4000 bytes.	Where size is the number of characters to store. Variable-length string.
Long	Maximum size of 2GB.	Variable-length strings. (backward compatible)
Raw	Maximum size of 2000 bytes.	Variable-length binary strings
Long Raw	Maximum size of 2GB.	Variable-length binary strings. (backward compatible)

Numeric Data type:

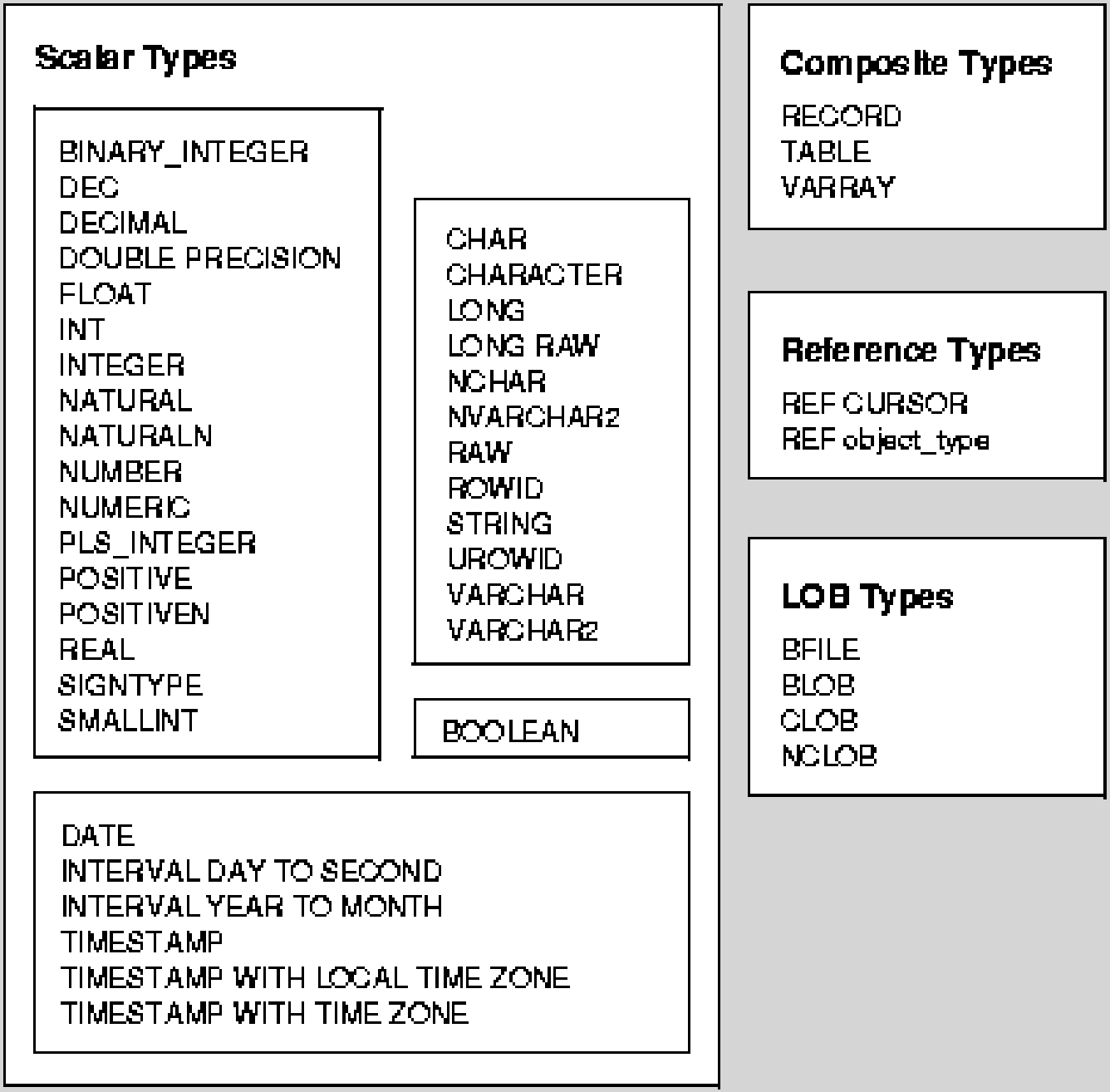
Data Type Syntax	Oracle 10g	Explanation
number(p,s)	Precision can range from 1 to 38. Scale can range from -84 to 127.	Where p is the precision and s is the scale. For example, number(7,2) is a number that has 5 digits before the decimal and 2 digits after the decimal.
numeric(p,s)	Precision can range from 1 to 38.	Where p is the precision and s is the scale. For example, numeric(7,2) is a number that has 5 digits before the decimal and 2 digits after the decimal.
dec(p,s)	Precision can range from 1 to 38.	Where p is the precision and s is the scale. For example, dec(3,1) is a number that has 2 digits before the decimal and 1 digit after the decimal.
decimal(p,s)	Precision can range from 1 to 38.	Where p is the precision and s is the scale. For example, decimal(3,1) is a number that has 2 digits before the decimal and 1 digit after the decimal.

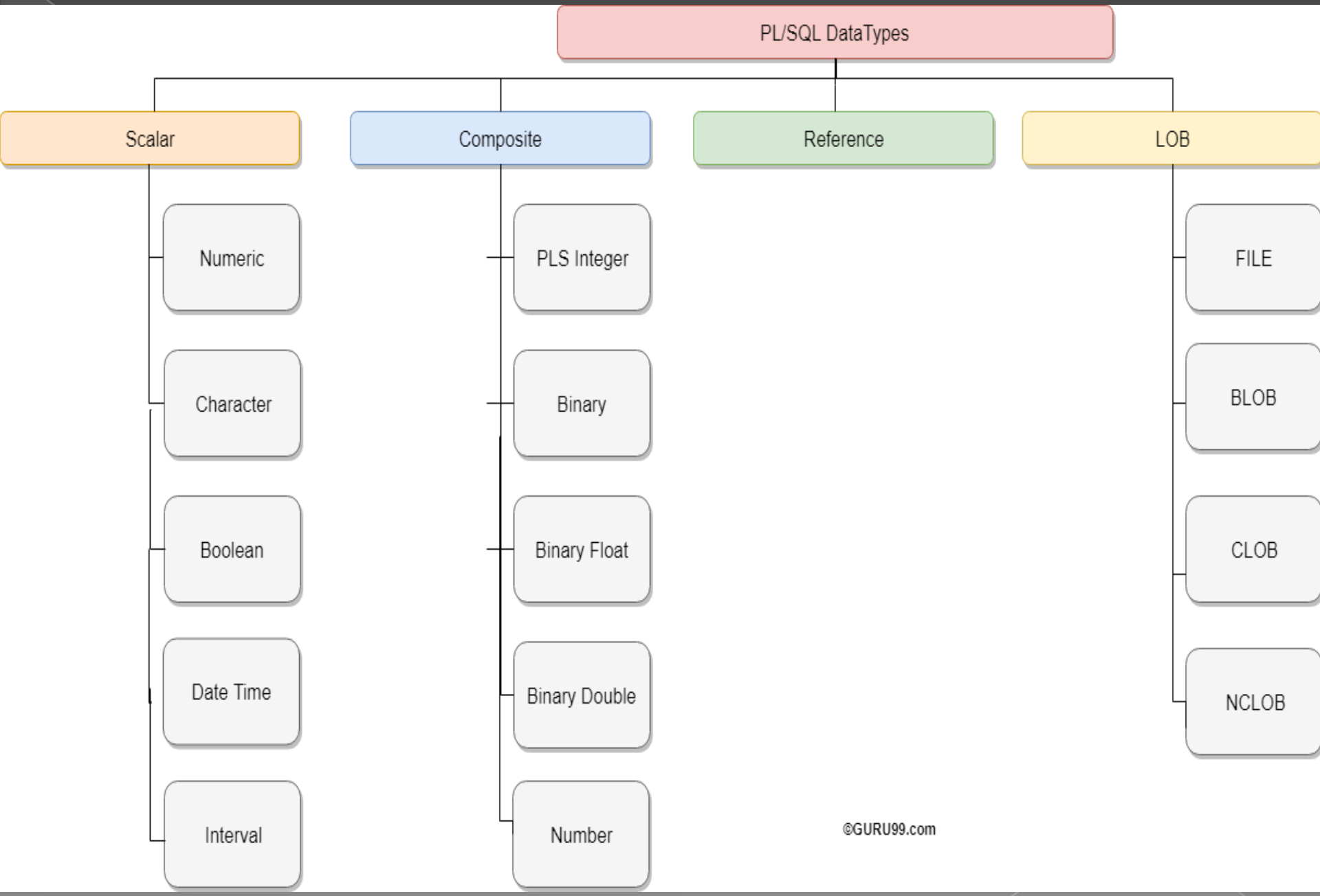
It also contains integer ,float ,int data types to store numeric values .

Date/Time Datatypes:

Data Type Syntax	Oracle 10g	Explanation
Date	A date between Jan 1, 4712 BC and Dec 31, 9999 AD.	
timestamp (fractional seconds precision)	Fractional second's precisionmust be a number between 0 and 9. (default is 6)	Includes year, month, day, hour, minute, and seconds. For example: timestamp(6)

Built-in Data types




PL/SQL DataTypes
Scalar
Composite
Reference
LOB
Numeric
Character
Boolean
Date Time
Interval
PLS Integer
Binary
Binary Float
Binary Double
Number
FILE
BLOB
CLOB
NCLOB
©GURU99.com

PL/SQL - Variables

- Each variable in PL/SQL has a specific data type, which determines the size and the layout of the variable's memory; the range of values that can be stored within that memory and the set of operations that can be applied to the variable.

  variable_name [CONSTANT] datatype [NOT NULL] [:= | DEFAULT initial_value]

```
sales number(10, 2);
pi CONSTANT double precision := 3.1415;
name varchar2(25);
address varchar2(100);
```

https://www.ibm.com/support/knowledgecenter/en/SS6NHC/com.ibm.swg.im.dashdb.apdv.plsql.doc/doc/c0053860.html

Initializing Variables in PL/SQL

- During the declaration, using either of the following:
- The **DEFAULT** keyword
- The **assignment** operator

counter binary_integer := 0;

greetings varchar2(20) DEFAULT 'Have a Good Day';

http://www.plsqltutorial.com/plsql-variables/

```
DECLARE
a integer := 10;
b integer := 20;
c integer;
f real;
BEGIN
c := a + b;
dbms_output.put_line('Value of c: ' || c);
f := 70.0/3.0; dbms_output.put_line('Value
of f: ' || f);
END;
/
```

PLSQL: Literals

- Literals (text, integer, and number)

1. **Text Literals**

- Text literals are always surrounded by single quotes (').

  For example:

  'Hewlett Packard' '28-MAY-13'

2. **Integer Literals**

- Integer literals can be up to 38 digits.
- Integer literals can be either positive numbers or negative numbers.
- If you do not specify a sign, then a positive number is assumed.

  23

  +23

  -23

3. **Number Literals**

- Number literals can be up to 38 digits.
- Number literals can be either positive or negative numbers.
- If you do not specify a sign, then a positive number is assumed.

  25

  +25

  -25

  25e-04

  25.607

- **Date/Time Literals**
- Date and time are enclosed in single quotes (').

  For example:

  '2015-04-30'

  '2015-04-30 08:13:24‘
- When dealing with date/time values, you will want to use the TO_DATE function to convert a literal to a date.

  For example:

  SELECT TO_DATE('2015/04/30', 'yyyy/mm/dd') FROM dual;

  This example will take a literal value of '2015/04/30' and convert it to a date.

Advantages of PL/SQL

- PL/SQL is development tool that not only supports SQL data manipulation but also provide facilities of conditional checking, branching and looping.
- PL/SQL sends an entire block of statements to the Oracle engine at one time. This is turn reduces network traffic. The Oracle engine gets the SQL statements as a single block, and hence it processes this code much faster than if it got the code one sentence at a time. There, is a definite improvement in the performance time of the Oracle engine. As an entire block of code is passed to the DBA at one time for execution, all changes made to the data in the table are done or undone, in one go.

1. PL/SQL also permits dealing with errors as required, and facilitates displaying user-friendly messages, when errors are encountered.
2. PL/SQL allows declaration and use of variables in blocks of code. These variables can be used to store intermediate results of a query for later processing, or calculate values and insert them into an Oracle table later. PL/SQL variables can be used anywhere, either in SQL statements or in PL/SQL blocks.
3. Via PL/SQL, all sorts of calculations can be done quickly and efficiently without the use of the Oracle engine. This considerably improves transaction performance.
4. Applications written in PL/SQL are portable to any computer hardware and operating system, where Oracle is operational. Hence, PL/SQL code blocks written for a DOS version of Oracle will run on its UNIX version, without any modifications at all.
5. Support for SQL
6. Support for object-oriented programming
7. Better performance
8. Higher productivity
9. Full portability
10. Tight integration with Oracle
11. Tight security

PL/SQL offers the following advantages over any other procedural language:

- support for SQL,
- closer integration with Oracle leading to better performance, and
- support for object-oriented programming.

Among the other advantages, performance of PL/SQL is mainly due to its architecture. Without PL/SQL, Oracle must process SQL statements one at a time. Each SQL statement results in a new call to Oracle and leads to higher performance overhead. In case of a network environment, this means a transmit and receive for each SQL statement, increasing the overhead significantly.

However, with PL/SQL, an entire block of statements can be sent to Oracle one at a time. A PL/SQL engine resides either at the server-end or at the client-end in an application development tool. In the former case processing of an entire block of PL/SQL statements takes place at the server end and hence drastically reduces the communication between the application and Oracle server.

The main disadvantages of PL/SQL is its lack of portability. This was not viewed as a major drawback in this proof-of-concept experiment. As discussed in the conclusion other languages e.g. C++ and JDBC should be considered in the future.

Advantages of PL/SQL

1. ***Block Structures*:** PL SQL consists of blocks of code, which can be nested within each other. Each block forms a unit of a task or a logical module. PL/SQL Blocks can be stored in the database and reused.
2. ***Procedural Language Capability*:** PL SQL consists of procedural language constructs such as conditional statements (if else statements) and loops like (FOR loops).
3. ***Better Performance*:** PL SQL engine processes multiple SQL statements simultaneously as a single block, thereby reducing network traffic.
4. ***Error Handling*:** PL/SQL handles errors or exceptions effectively during the execution of a PL/SQL program. Once an exception is caught, specific actions can be taken depending upon the type of the exception or it can be displayed to the user with a message.

PL/SQL Control Statements

- PL/SQL has three categories of control statements:
- **Conditional selection statements**, which run different statements for different data values. The conditional selection statements are IF and CASE.
- **Loop statements**, which run the same statements with a series of different data values. The loop statements are the basic LOOP, FOR LOOP, and WHILE LOOP.

  The EXIT statement transfers control to the end of a loop. The CONTINUE statement exits the current iteration of a loop and transfers control to the next iteration. Both EXIT and CONTINUE have an optional WHEN clause, where you can specify a condition.
- **Sequential control statements**, which are not crucial to PL/SQL programming. The sequential control statements are GOTO, which goes to a specified statement, and NULL, which does nothing.

IF THEN Statement

```
DECLARE
  PROCEDURE p (
    sales  NUMBER,
    quota  NUMBER,
    emp_id NUMBER
  )
  IS
    bonus    NUMBER := 0;
    updated  VARCHAR2(3) := 'No';
  BEGIN
    IF sales > (quota + 200) THEN
      bonus := (sales - quota)/4;

      UPDATE employees
      SET salary = salary + bonus
      WHERE employee_id = emp_id;

      updated := 'Yes';
    END IF;

    DBMS_OUTPUT.PUT_LINE (
      'Table updated?  ' || updated || ', ' ||
      'bonus = ' || bonus || '.'
    );
  END p;
BEGIN
  p(10100, 10000, 120);
  p(10500, 10000, 121);
END;
/
```

IF THEN ELSE Statement

```
DECLARE
  PROCEDURE p (
    sales  NUMBER,
    quota  NUMBER,
    emp_id NUMBER
  )
  IS
    bonus  NUMBER := 0;
  BEGIN
    IF sales > (quota + 200) THEN
      bonus := (sales - quota)/4;
    ELSE
      bonus := 50;
    END IF;

    DBMS_OUTPUT.PUT_LINE('bonus = ' || bonus);

    UPDATE employees
    SET salary = salary + bonus
    WHERE employee_id = emp_id;
  END p;
BEGIN
  p(10100, 10000, 120);
  p(10500, 10000, 121);
END;
/
```