PL/SQL

Functies

DE HOGESCHOOL MET HET NETWERK

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be

Ingebouwde functies

Bekend vanuit de lessen SQL

Vb SELECT sysdate FROM dual;
SELECT SUBSTR(last_name, 1, 4) FROM employees;

Er bestaan functies met of zonder parameters.

Een functie geeft **altijd** 1 resultaat terug.

Creating Functions

The PL/SQL block must have at least one `RETURN` statement.

```
CREATE [OR REPLACE] FUNCTION function_name
 [(parameter1 [mode1] datatype1, . . .)]
RETURN datatype IS|AS
 [local_variable_declarations;
  . . .]
BEGIN
  -- actions;
  RETURN expression;
END [function_name];
```

PL/SQL Block

- RETURN datatype → geen lengte meegeven
- De mode is altijd IN (default)

Functie: voorbeeld zonder parameters

```
CREATE OR REPLACE FUNCTION sysdate2
RETURN VARCHAR2
AS
      v_tekst      VARCHAR2(30);
BEGIN
      v_tekst := to_char(sysdate, 'fmDay, dd month yyyy');
      RETURN v_tekst;
END;
/
```

Return-type: geen lengte!!

Functie

- / → creatie functie
- de broncode wordt in ieder geval in de data dictionary opgeslagen
- als foutloze code: gecompileerde versie → databank
- als code met fouten:
  `Melding: 'created with compilation errors'.`
  Fouten opvragen → `show errors`
- DESC functienaam → overzicht van de functie

```
SQL> desc netto
FUNCTION netto RETURNS VARCHAR2
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 P_BRUTO                        NUMBER(8,2)             IN
```

Oproepen functie

Vanuit een anomiem blok, andere functie of procedure:

v_datum := sysdate2;

Vanuit SQL*Plus:

```
SQL> SELECT sysdate2
       FROM dual;

of

SQL> EXECUTE DBMS_OUTPUT.PUT_LINE(SYSDATE2)
```

Functie: voorbeeld met parameters

```
CREATE OR REPLACE FUNCTION fulldate
(p_date  IN  DATE)
RETURN VARCHAR2
AS
BEGIN
  RETURN TO_CHAR(p_date, 'fmDay, dd month yyyy');
END;
/
```

Oproepen functie met parameter

Vanuit een anomiem blok, andere functie of procedure:

```
v_tekst := fulldate(v_datum);
```

Vanuit SQL*Plus:

```
SQL>  SELECT fulldate(hire_date)
      FROM employees;
```

Creating and Invoking a Stored Function Using the CREATE FUNCTION Statement: Example

```
CREATE OR REPLACE FUNCTION get_sal
 (p_id employees.employee_id%TYPE) RETURN NUMBER IS
  v_sal employees.salary%TYPE := 0;
BEGIN
  SELECT salary
  INTO   v_sal
  FROM   employees
  WHERE  employee_id = p_id;
  RETURN v_sal;
END get_sal; /
```

```
FUNCTION get_sal Compiled.
```

```
-- Invoke the function as an expression or as
-- a parameter value.

EXECUTE dbms_output.put_line(get_sal(100))
```

```
anonymous block completed
24000
```

Using Different Methods for Executing Functions

```
-- As a PL/SQL expression, get the results using host variables

VARIABLE b_salary NUMBER
EXECUTE :b_salary := get_sal(100)
```

```
anonymous block completed
b_salary
-----
24000
```

```
-- As a PL/SQL expression, get the results using a local
-- variable

DECLARE
  sal employees.salary%type;
BEGIN
  sal := get_sal(100);
  DBMS_OUTPUT.PUT_LINE('The salary is: '|| sal);
END;/
```

```
anonymous block completed
The salary is: 24000
```

Using Different Methods for Executing Functions

```
-- Use as a parameter to another subprogram

EXECUTE dbms_output.put_line(get_sal(100))
```

```
anonymous block completed
24000
```

```
-- Use in a SQL statement (subject to restrictions)

SELECT job_id, get_sal(employee_id) FROM employees;
```

```
JOB_ID     GET_SAL(EMPLOYEE_ID)
---------- --------------------
SH_CLERK   2600
SH_CLERK   2600
AD_ASST    4400
MK_MAN     13000
   . . .
SH_CLERK   3100
SH_CLERK   3000

107 rows selected
```

Functie verwijderen

- Syntax:

```
DROP FUNCTION function_name
```

- Voorbeeld: `DROP FUNCTION get_aantal_dienstjaren;`
  - Alle privileges betreffende de functie worden mee verwijderd.
  - De `CREATE OR REPLACE` syntax is equivalent aan het verwijderen en opnieuw creëren van de functie. Toegekende privileges i.v.m. de functie blijven bestaan als deze syntax gebruikt wordt.

Opvragen kenmerken (Data Dictionary)

Alle informatie over bestaande PL/SQL procedures en functies worden bewaard in de databank. Je kan hiervoor gebruik maken van volgende Oracle data dictionary views:

- **USER_OBJECTS**: deze view bevat informatie over ALLE databankobjecten van de eigen user, dus alle zelf-gecreëerde tabellen, indexen, sequences, functies, procedures,….

- **USER_SOURCE**: hierin zit de code van bepaalde objecten

```
SELECT object_name
FROM   user_objects
WHERE  object_type = 'FUNCTION';

SELECT text
FROM   user_source
WHERE  name = 'RAISE_SALARY_DEPT';
```