PL/SQL: Verwendung von SQL und RECORDS

Stephan Karrer

Variablen: Wertzuweisung mit SELECT INTO

```
DECLARE
  emp_id employees.employee_id%TYPE := 100;
  emp_name employees.last_name%TYPE;
  wages NUMBER(7,2);
BEGIN
  SELECT last_name, salary + (salary *
                                              nvl(commission_pct,0))
        INTO emp_name, wages
        FROM employees
        WHERE employee_id = emp_id;

  DBMS_OUTPUT.PUT_LINE
    ('Employee ' || emp_name || ' might make ' || wages);
END;
```

Das %TYPE -Attribut

```
CREATE TABLE employees_temp (empid NUMBER(6) NOT NULL PRIMARY KEY,
  deptid NUMBER(6) CONSTRAINT check_deptid CHECK (deptid BETWEEN
                                                     100 AND 200),
  deptname VARCHAR2(30) DEFAULT 'Sales');

DECLARE
  v_empid employees_temp.empid%TYPE;
  v_deptid employees_temp.deptid%TYPE;
  v_deptname employees_temp.deptname%TYPE;
BEGIN
  v_empid := NULL; /*this works, null constraint is not inherited*/
  /* v_empid := 10000002; invalid, number precision too large */
  v_deptid := 50; /*this works, check constraint is not inherited*/
  /* the default value is not inherited in the following */
  DBMS_OUTPUT.PUT_LINE('v_deptname: ' || v_deptname);
END;
```

Verwendung von DML-Anweisungen in PL/SQL

```
CREATE TABLE employees_temp
     AS SELECT first_name, last_name FROM employees;
DECLARE
  x VARCHAR2(20) := 'my_first_name';
  y VARCHAR2(25) := 'my_last_name';
BEGIN
  INSERT INTO employees_temp VALUES(x, y);
  UPDATE employees_temp SET last_name = x WHERE first_name = y;
  DELETE FROM employees_temp WHERE first_name = x;
  COMMIT;
END;
```

Verwendung der RETURNING-Klausel

```
CREATE TABLE employees_temp
  AS SELECT employee_id, first_name, last_name FROM employees;

DECLARE
  emp_id employees_temp.employee_id%TYPE;
  emp_first_name employees_temp.first_name%TYPE;
  emp_last_name employees_temp.last_name%TYPE;
BEGIN
  INSERT INTO employees_temp VALUES(299, 'Bob', 'Henry');
  UPDATE employees_temp
  SET first_name = 'Robert' WHERE employee_id = 299;
  DELETE FROM employees_temp WHERE employee_id = 299
      RETURNING first_name, last_name
      INTO emp_first_name, emp_last_name;
  COMMIT;
  DBMS_OUTPUT.PUT_LINE( emp_first_name || ' ' || emp_last_name);
END;
```

Verwendung des SQL%ROWCOUNT-Attributs

```
CREATE TABLE employees_temp AS SELECT * FROM employees;

BEGIN
     UPDATE employees_temp
     SET salary = salary * 1.05 WHERE salary < 5000;
     DBMS_OUTPUT.PUT_LINE('Updated ' || SQL%ROWCOUNT || '
                                        salaries.');
END;
```

Verwendung von SQL-Funktionen

```
DECLARE
  job_count NUMBER;
  emp_count NUMBER;
BEGIN
  SELECT COUNT(DISTINCT job_id)
        INTO job_count
        FROM employees;

  SELECT COUNT(*)
        INTO emp_count
        FROM employees;
END;
```

Zusammengesetzte Datentypen

- RECORD: Daten unterschiedlicher Typen als logische Einheit
- Collections: Daten gleichen Datentyps als logische Einheit
    - TABLE INDEX BY
    - NESTED TABLE
    - VARRAY

Zusammengesetzte Datentypen: Records

```
DECLARE
  TYPE emprec_type IS RECORD
    (lname VARCHAR2(25) := 'Karrer',
     jobid VARCHAR2(10),
     sal   NUMBER(8,2)
     );
  emprecord emprec_type;
BEGIN
  DBMS_OUTPUT.PUT_LINE( emprecord.lname );
  emprecord.jobid := 'IT_PROG';
  emprecord.sal := 5000.00;

  /* ... */
END;
```

Das %ROWTYPE -Attribut

```
DECLARE
    emp_rec employees%ROWTYPE;

BEGIN

    SELECT * INTO emp_rec
        FROM employees
        WHERE ROWNUM < 2;

    IF emp_rec.department_id = 20
            AND emp_rec.last_name = 'JOHNSON'
        THEN  emp_rec.salary := emp_rec.salary * 1.15;
    END IF;

END;
```

Records: ein komplexeres Beispiel

```
DECLARE
  TYPE DeptRecType IS RECORD (
    deptid NUMBER(4) NOT NULL := 99,
    dname departments.department_name%TYPE,
    reg regions%ROWTYPE );
  dept_rec DeptRecType;
BEGIN
  SELECT r.region_id, r.region_name INTO dept_rec.reg
    FROM regions r
         JOIN countries c ON c.region_id = r.region_id
         JOIN locations l ON l.country_id = c.country_id
         JOIN departments d ON d.location_id = l.location_id
    WHERE department_id = 50;
  -- ...
END;
```

Zusammengesetzte Datentypen: mit Records Tabelleneinträge aktualisieren

```
DECLARE
       dept_info departments%ROWTYPE;

BEGIN
       dept_info.department_id := 300;
       dept_info.department_name := 'Personnel';
       dept_info.location_id := 1700;

       INSERT INTO departments VALUES dept_info;

       UPDATE departments SET ROW = dept_info
              WHERE department_id = 300;
END;
```