# Tipps & Tricks: Natives dynamisches SQL

| Bereich: | PL/SQL | Erstellung: | 04/2001 HA |
|---|---|---|---|
| Versionsinfo: | 8.1.7, 9.2, 10.2, 11.1 | Letzte Überarbeitung: | 05/2009 HA |

# Natives dynamisches SQL

## Dynamisches SQL mit REF Cursorn

REF Cursor waren bereits unter Version 8.0 bekannt, doch konnte die OPEN FOR-Klausel nur ein statisches SELECT-Statement enthalten. Seit Version 8.i sind dynamische SELECT-Anweisungen möglich, die erst zur Laufzeit erzeugt werden.

Syntax:

```
OPEN  <cursor-variable>
  FOR <select_anweisung>
  [USING bind_variable1[, bind_variable2...]];
```

Achtung: Bind-Variablen sind nicht zulässig für Spalten- oder Tabellennamen, sondern nur als Platzhalter für Werte.

Beispiel:

```
CREATE  OR REPLACE PROCEDURE ref_bsp
     (tname IN VARCHAR2,
      cname IN VARCHAR2,
      wert  IN VARCHAR2 DEFAULT  NULL)
IS
   TYPE r_cur_t IS REF Cursor;
   r_cur     r_cur_t;
   sqlstring VARCHAR2(2000);
BEGIN
   sqlstring  := 'SELECT  '||cname||'  FROM  '||tname;
   OPEN  r_cur FOR sqlstring;
   --......
   CLOSE r_cur;
   sqlstring  := 'SELECT  '||cname||'
               FROM  '||tname||'
               WHERE  '||cname||' = :ein_name';
   OPEN  r_cur FOR sqlstring USING wert;
   --......
   CLOSE r_cur;
END;
/

EXEC ref_bsp('emp', 'ename', 'KING')
```

## EXECUTE IMMEDIATE

Innerhalb von PL/SQL sind nur SELECT INTO und DML unmittelbar zulässig, eine Einschränkung, die vor 8i nur mittels DBMS_SQL umgangen werden konnte.
Die Anweisung EXECUTE IMMEDIATE bietet seit Version 8i die Möglichkeit, einen beliebigen, dynamisch erzeugten SQL-Befehl sofort auszuführen (Ausnahme: ein SELECT-Befehl, der keine oder mehr als eine Zeile zurückliefert => REF Cursor). Auch ein kompletter PL/SQL-Block kann an diese Anweisung übergeben werden; dieser muss allerdings im Unterschied zu einer SQL-Anweisung mit einem Semikolon abgeschlossen werden.

Syntax:

```
EXECUTE  IMMEDIATE sql_string
  [INTO {define_variable[, define_variable]... | record}]
  [USING [IN | OUT | IN OUT] bind_argument
      [, [IN | OUT | IN OUT]  bind_argument]...]
  [{RETURNING | RETURN} INTO bind_argument[, bind_argument]...];
```

Die USING-Klausel wird (analog zu Ref-Cursorn) zur Übergabe von Bind-Variablen verwendet.

Die INTO-Klausel wird nur im Zusammenhang mit SELECT verwendet, z.B.:

```
DECLARE
   v_name EMP.ename%TYPE;
BEGIN
   EXECUTE IMMEDIATE
   'SELECT ename FROM emp WHERE empno = 7369'
   INTO v_name;
   DBMS_OUTPUT.PUT_LINE(v_name);
END;
```

RETURNING|RETURN wird bei DML-Anweisungen mit RETURNING-Klausel benötigt:

```
DECLARE
    v_name    EMP.ename%TYPE;
    v_empno   EMP.empno%TYPE := &nummer;
BEGIN
   EXECUTE IMMEDIATE 'DELETE FROM emp
                       WHERE  empno = :1
                      RETURNING  ename INTO :2'
                USING v_empno
                RETURNING INTO v_name;
   DBMS_OUTPUT.PUT_LINE('gelöscht: '||v_name);
END;
```

Die Variable v_name kann in diesem Beispiel nicht unmittelbar im DML-Befehl referenziert werden, weil sie innerhalb des an EXECUTE IMMEDIATE übergebenen Strings "unbekannt" ist. Stattdessen wird sie über die RETURNING-Klausel als Bind-Variable übergeben.

Weitere Beispiele:

```
BEGIN
   EXECUTE  IMMEDIATE
   'CREATE TABLE temp (nr NUMBER, t_user VARCHAR2(100))';
END;
/
BEGIN
   EXECUTE  IMMEDIATE
    'CREATE INDEX indy ON emp(ename)';
END;
/
BEGIN
   EXECUTE IMMEDIATE 'ALTER SESSION SET nls_language ='''GERMAN''';
END;
/
DECLARE
   sqlstring VARCHAR2(2000);
BEGIN
   sqlstring:=
        'DECLARE
            dummy  NUMBER;
         BEGIN
            dummy  := 10;
            DBMS_OUTPUT.PUT_LINE(''dummy  =''||dummy);
         END;';
   EXECUTE IMMEDIATE sqlstring;
END;
/
DECLARE
   sqlstring VARCHAR2(2000);
   v_table   VARCHAR2(30) := '&Tabelle';
BEGIN
   sqlstring :='DROP TABLE '||v_table;
   EXECUTE IMMEDIATE sqlstring;
END;
/
```