
Kapitel 4

Dynamisches SQL

Folien zum Datenbankpraktikum
Wintersemester 2010/11 LMU München

© 2008 Thomas Bernecker, Tobias Emrich
unter Verwendung der Folien des Datenbankpraktikums aus dem Wintersemester 2007/08 von Dr. Matthias Schubert

- Der Einsatz von dynamischen SQL ist notwendig, wenn Anweisungen ausgeführt werden sollen, die in PL/SQL nicht als statische Anweisungen ausführbar sind, wie z.B. :
 - DDL-Anweisungen,
 - Kontrollanweisungen (GRANT, etc.)
 - Zur Laufzeit zusammengestellte dynamische Anweisungen
- Dynamische SQL-Anweisungen sind nicht in Quellcode eingebettet, sondern werden als Zeichenkette zur Laufzeit generiert oder als Parameter einer PL/SQL-Funktion oder -Prozedur übergeben.
- Es können Anfragen mit einem oder mehreren Ergebnissätzen formuliert werden
- Ist eine SQL-Anweisung zur Übersetzungszeit (beim Compilieren) der Anwendung nicht vollständig bekannt (mit Ausnahme von Parametern), dann muss dynamisches SQL verwendet werden.
- *ORACLE* bietet zwei Möglichkeiten dynamische SQL-Anweisungen in einem PL/SQL – Block zu implementieren:
 - DBMS_SQL Package
 - Natives dynamisches SQL

Übersicht

4.1 DBMS_SQL Package

4.2 Natives dynamisches SQL

4.3 Gegenüberstellung

DBMS_SQL Package

- o Beispiel: dynamisches Einfügen in eine Tabelle
(Variablenübergabe durch *Call by Value*)

```
CREATE OR REPLACE PROCEDURE einfuegen (  
    tab_name      VARCHAR2,  
    telnr        INTEGER,  
    name         VARCHAR2) IS  
    cursor_hdl    INTEGER;  
    sql_anweisung VARCHAR(100);  
    rows_processed INTEGER;  
  
BEGIN  
    sql_anweisung := 'INSERT INTO ' || tab_name ||  
        ' VALUES (:var1, :var2)';  
    cursor_hdl := DBMS_SQL.open_cursor;  
    DBMS_SQL.parse(cursor_hdl, sql_anweisung, DBMS_SQL.native);  
    DBMS_SQL.bind_variable(cursor_hdl, ':var1', telnr);  
    DBMS_SQL.bind_variable(cursor_hdl, ':var2', name);  
    rows_processed := DBMS_SQL.execute(cursor_hdl);  
    DBMS_SQL.close_cursor(cursor_hdl);  
END;
```

Übersicht

4.1 DBMS_SQL Package

4.2 Natives dynamisches SQL

4.3 Gegenüberstellung

Anfragen mit einem Ergebnissatz:

```
EXECUTE IMMEDIATE <Anweisung>
  [INTO <ErgebnisVariable>, ... <ErgebnisVariable>]
  [USING <ParameterVariable>, ... <ParameterVariable>];
```

Beispiel:

```
DECLARE
  Name          VARCHAR(20);
  Nr            INTEGER := 9332;
  sql_anweisung VARCHAR(100);
BEGIN
  sql_anweisung := 'SELECT Name FROM Mitarbeiter WHERE TelNr = :1';
  EXECUTE IMMEDIATE sql_anweisung INTO Name USING Nr;
  sql_anweisung := 'INSERT INTO Mitarbeiter VALUES (:1, :2)';
  EXECUTE IMMEDIATE sql_anweisung USING Name, Nr;
END;
```

Anfragen mit mehreren Ergebnissätzen:

```
OPEN <Cursorvariable> FOR <SELECT-Anweisung>
  [USING <ParameterVariable>, ... <ParameterVariable>];
LOOP
  FETCH <Cursorvariable> INTO <ErgebnisVariable>, ...
                                <ErgebnisVariable>;
  EXIT WHEN <Cursorvariable>%NOTFOUND;
END LOOP;
CLOSE <Cursorvariable>;
```

- Die SELECT-Anweisung wird zur Laufzeit zusammengestellt.
- Eventuelle Parameter werden mit USING gebunden.
- Die Bearbeitung erfolgt wie bei einem normalen Cursor in einer LOOP-Schleife,
- die über FETCH die einzelnen Ergebnissätze in Variablen ablegt.

Beispiel: DDL mit dynamischem SQL

- ```
CREATE OR REPLACE PROCEDURE tab_erstellen (tab_name VARCHAR2) IS
BEGIN
-- Erstellen einer Tabelle. Name der Tabelle im Parameter tab_name
EXECUTE IMMEDIATE
 'CREATE TABLE ' || tab_name || ' (
 t_id NUMBER(4) NOT NULL,
 name VARCHAR(15),
 beruf VARCHAR(10),
 einkommen NUMBER(6,2),
 telnr NUMBER(10)
)';
END;
```
- ```
CREATE OR REPLACE PROCEDURE tab_loeschen (tab_name VARCHAR2) IS
BEGIN
EXECUTE IMMEDIATE 'DROP TABLE ' || tab_name;
END;
```

Übersicht

4.1 DBMS_SQL Package

4.2 Natives dynamisches SQL

4.3 Gegenüberstellung

Vergleich: DBMS_SQL – Natives dynamisches SQL

DBMS SQL Package

```
CREATE OR REPLACE PROCEDURE einfuegen (  
    tab_name  VARCHAR2,  
    telnr     INTEGER,  
    name      VARCHAR2) IS  
    cursor_hdl  INTEGER;  
    sql_anweisung  VARCHAR(100);  
    rows_processed  INTEGER;  
BEGIN  
    sql_anweisung := 'INSERT INTO ' ||  
                    tab_name ||  
                    ' VALUES (:var1, :var2)';  
    cursor_hdl := DBMS_SQL.open_cursor;  
    DBMS_SQL.parse(cursor_hdl, sql_anweisung, 1);  
    DBMS_SQL.bind_variable(cursor_hdl, ':var1', telnr);  
    DBMS_SQL.bind_variable(cursor_hdl, ':var2', name);  
    rows_processed := DBMS_SQL.execute(cursor_hdl);  
    DBMS_SQL.close_cursor(cursor_hdl);  
END;
```

Natives dynamisches SQL

```
CREATE OR REPLACE PROCEDURE einfuegen (  
    tab_name  VARCHAR2,  
    telnr     INTEGER,  
    name      VARCHAR2) IS  
    sql_anweisung  VARCHAR(100);  
BEGIN  
    sql_anweisung :=  
        'INSERT INTO ' ||  
        tab_name ||  
        ' VALUES (:var1, :var2)';  
    EXECUTE IMMEDIATE sql_anweisung  
        USING telnr, name;  
END;
```

Beispiel: Dynamische Ausführung eines PL/SQL-Blocks

<pre>CREATE OR REPLACE PROCEDURE event_handler_1 (param number) IS BEGIN ... END; CREATE OR REPLACE PROCEDURE event_handler_2 (param number) IS BEGIN ... END; CREATE OR REPLACE PROCEDURE event_handler_3 (param number) IS BEGIN ... END; CREATE OR REPLACE PROCEDURE dispatcher (event number, param number) IS BEGIN IF (event = 1) THEN EVENT_HANDLER_1(param); ELSIF (event = 2) THEN EVENT_HANDLER_2(param); ELSIF (event = 3) THEN EVENT_HANDLER_3(param); END IF; END;</pre>	<pre>CREATE OR REPLACE PROCEDURE dispatcher (event NUMBER, param NUMBER) IS BEGIN EXECUTE IMMEDIATE 'BEGIN EVENT_HANDLER_' to_char(event) '(:1); END; ' USING param; END;</pre>
--	---

Vergleich: Dynamisches SQL – Statisches SQL

- Statisches SQL umfasst SQL-Anweisungen, die man zur Übersetzungszeit exakt kennt (d.h. die als Zeichenketten vorgegeben sind)
- Sie enthalten insbesondere keine Variablen für Tabellen- oder Spaltennamen
- Solche Anweisungen können schon zur Übersetzungszeit vorbereitet werden
- Man kennt auch die Anzahl und Typen der Ergebnisspalten der SQL-Anweisung, die bei jedem Programmablauf in exakt gleicher Weise ausgeführt wird.
- Statisches SQL ist effizient, da die Anweisungen vor ihrer Ausführung einmal vorbereitet werden und dann (mehrere Male) ausgeführt werden können.
- Die Anwendung, die statisches SQL verwendet, ist allerdings nach der Übersetzung an eine spezielle Datenbank gebunden.