

Tipps & Tricks: Performance-Vergleich

Bereich:	PL/SQL	Erstellung:	06/2001 HA
Versionsinfo:	8.1.7, 9.2, 10.2, 11.1	Letzte Überarbeitung:	05/2009 HA

 [Als PDF Downloaden!](#)

Vergleich Bulk Binds, FOR-Schleife und DBMS_SQL

SELECT - Beispiel

Das folgende Beispiel geht von "aufgeblähten" emp-Tabellen von jeweils 131.072 Datensätzen aus; es wurden drei verschiedene Tabellen benutzt, damit der Befehl jedes Mal sicher neu geparkt werden muss. Ähnliche Ergebnisse wurden auch mit einer realen großen Tabelle erzielt. Auch für DBMS_SQL wurde mit Array-Fetch statt mit zeilenweisem Holen gearbeitet.

```
CREATE OR REPLACE PROCEDURE BULK_PERFORMANCE (anzahl IN NUMBER)
IS
  dummy1      NUMBER;
  dummy2      NUMBER;
  dummy3      INTEGER;
  sqlstring   VARCHAR2 (2000);
  v_cur       INTEGER;
  v_empno     DBMS_SQL.NUMBER_TABLE;  -- vordefinierte Index By-Tabelle
BEGIN
  dummy1 := DBMS_UTILITY.get_time;

  SELECT empno
     BULK COLLECT INTO v_empno
   FROM scott.bigemp
  WHERE ROWNUM < anzahl;

  dummy2 := DBMS_UTILITY.get_time;
  DBMS_OUTPUT.put_line ('Zeit Bulk: ' || (dummy2 - dummy1));
  dummy1 := DBMS_UTILITY.get_time;

  FOR i IN (SELECT empno
            FROM scott.bigemp2
            WHERE ROWNUM < anzahl)
  LOOP
    v_empno (SQL%ROWCOUNT) := i.empno;
  END LOOP;

  dummy2 := DBMS_UTILITY.get_time;
  DBMS_OUTPUT.put_line ('Zeit einzeln: ' || (dummy2 - dummy1));
  sqlstring := 'SELECT empno FROM SCOTT.BIGEMP3
               WHERE ROWNUM < :anzahl';
  dummy1 := DBMS_UTILITY.get_time;
  v_cur := DBMS_SQL.open_cursor;
```

```

DBMS_SQL.parse (v_cur, sqlstring, DBMS_SQL.native);
DBMS_SQL.bind_variable (v_cur, ':anzahl', anzahl);
DBMS_SQL.define_array (v_cur, 1, v_empno, anzahl - 1, 1);
dummy3 := DBMS_SQL.EXECUTE (v_cur);
dummy3 := DBMS_SQL.fetch_rows (v_cur);
DBMS_SQL.COLUMN_VALUE (v_cur, 1, v_empno);
DBMS_SQL.close_cursor (v_cur);
dummy2 := DBMS_UTILITY.get_time;
DBMS_OUTPUT.put_line ('Zeit DBMS_SQL: ' || (dummy2 - dummy1));
END; -- Procedure BULK_PERFORMANCE
/

```

Ergebnis in 8i (auf sehr langsamer Hardware):

```

Anzahl = 1000:
Zeit Bulk: 8
Zeit einzeln: 24
Zeit DBMS_SQL: 12

Anzahl = 10000:
Zeit Bulk: 16
Zeit einzeln: 106
Zeit DBMS_SQL: 16

Anzahl = 100000
Zeit Bulk: 148
Zeit einzeln: 1073
Zeit DBMS_SQL: 52

```

DML - Beispiel

Voraussetzungen waren auch hier die drei gleich großen Tabellen.
 Für DBMS_SQL wurde mit Array-Bind gearbeitet.

```

CREATE OR REPLACE Procedure FORALL_PERFORMANCE (anzahl NUMBER)
IS
    v_empno    DBMS_SQL.NUMBER_TABLE;  -- vordefinierte Index By-Tabelle
    v_ename    DBMS_SQL.VARCHAR2_TABLE; -- vordefinierte Index By-Tabelle
    dummy      INTEGER;
    dummy1     NUMBER;
    dummy2     NUMBER;
    sqlstring  VARCHAR2(2000);
    v_cur      INTEGER;
BEGIN
    FOR i IN 1 .. anzahl
    LOOP
        v_empno (i) := i;
        v_ename (i) := 'Num ' || i;
    END LOOP;
    dummy1 := DBMS_UTILITY.get_time;
    FORALL j IN v_empno.FIRST .. v_empno.LAST

```

```
INSERT INTO scott.bigemp(empno, sal, ename )
VALUES (1, v_empno(j), v_ename(j));
dummy2 := DBMS_UTILITY.get_time;
DBMS_OUTPUT.PUT_LINE('forall: ' || (dummy2 - dummy1));
ROLLBACK;

dummy1 := DBMS_UTILITY.get_time;
FOR j IN v_empno.FIRST .. v_empno.LAST
LOOP
INSERT INTO scott.bigemp2(empno, sal, ename)
VALUES (1, v_empno(j), v_ename(j));
END LOOP;
dummy2 := DBMS_UTILITY.get_time;
DBMS_OUTPUT.PUT_LINE('for-Schleife: ' || (dummy2 - dummy1));
ROLLBACK;

sqlstring :=
'INSERT INTO scott.bigemp3 (empno, sal, ename)
VALUES (1, :empno, :ename)';
dummy1 := DBMS_UTILITY.get_time;
v_cur := DBMS_SQL.open_cursor;
DBMS_SQL.parse(v_cur, sqlstring, DBMS_SQL.native);
DBMS_SQL.bind_array(v_cur, ':empno', v_empno);
DBMS_SQL.bind_array(v_cur, ':ename', v_ename);
dummy := DBMS_SQL.EXECUTE(v_cur);
DBMS_SQL.close_cursor(v_cur);
dummy2 := DBMS_UTILITY.get_time;
DBMS_OUTPUT.PUT_LINE('DBMS bulk: ' || (dummy2 - dummy1));
ROLLBACK;
END; -- Procedure FORALL_PERFORMANCE
/
```

Ergebnis in 8i (auf sehr langsamer Hardware):

```
Anzahl = 100:
forall: 2
for-Schleife: 8
DBMS bulk: 4
```

```
Anzahl = 1000:
forall: 4
for-Schleife: 38
DBMS bulk: 56
```

```
Anzahl = 10000:
forall: 119
for-Schleife: 1532
DBMS bulk: 1615
```

```
Anzahl = 10000 (2. Versuch):
forall: 258
for-Schleife: 1478
DBMS bulk: 1727
```

Fazit:

DBMS_SQL ist um einiges umständlicher und unübersichtlicher zu programmieren als FORALL bzw. BULK COLLECT, vor allem bei langen Spaltenlisten (aus gutem Grund wurden hier nur maximal zwei Spalten berücksichtigt). Bei DML-Befehlen ist FORALL eindeutig die schnellste Variante, aber bei einem SELECT, der viele Zeilen betrifft, kann DBMS_SQL nach wie vor eine Alternative sein, wenn Performanceverbesserung wichtig ist.