Tipps & Tricks: DBMS_TRACE

Bereich:	PL/SQL	Erstellung:	01/2002 HA
Versionsinfo:	9.2, 10.2, 11.1	Letzte Überarbeitung:	06/2009 HA

PL/SQL-Tracing

Neben der altbekannten Möglichkeit, SQL-Aufrufe zu tracen, bietet das Package **DBMS_TRACE** ab Version 8i die Möglichkeit, auch PL/SQL-Aufrufe und Exceptions zu tracen. Voraussetzung dafür ist, dass im Schema SYS das Skript `tracetab.sql` ausgeführt wird. Dadurch werden zwei Tabellen und eine Sequenz (SYS.PLSQL_TRACE_RUNNUMBER) zur Erzeugung der eindeutigen RUNID angelegt. Dem jeweiligen User müssen außerdem die entsprechenden Rechte auf Tabellen und Sequenz gegranted werden. Die Tabelle SYS.PLSQL_TRACE_RUNS zeigt an, welcher User wann Tracing durchgeführt hat, und welche RUNID ihm dafür zugewiesen wurde. Die eigentlichen Informationen finden sich dann unter dieser RUNID in der Tabelle SYS.PLSQL_TRACE_EVENTS.

Mögliche Levels sind:

- **trace_all_calls / trace_enabled_calls**: Aufrufe innerhalb von PL/SQL werden mitprotokolliert - aufrufendes Programm (EVENT_UNIT_OWNER, EVENT_UNIT, EVENT_UNIT_KIND) , aufgerufenes Programm (PROC_OWNER, PROC_UNIT, PROC_UNIT_KIND), Zeilennummer (EVENT_LINE) und Stack-Tiefe (STACK_DEPTH)
- **trace_all_exceptions / trace_enabled_exceptions**: Exceptions werden mitprotokolliert, ob user_defined (USER_EXCP=1, EXCP=0) oder predefined (USER_EXCP=0, EXCP=0RA-Fehlernummer), in welcher Zeile (EVENT_LINE) sie aufgetreten sind (EVENT_COMMENT = "Exception raised", EVENT_KIND = 52), und (ggf.) in welcher Zeile sie abgefangen wurden (EVENT_COMMENT = "Exception handled", EVENT_KIND = 53).
- **trace_all_sql / trace_enabled_sq**l: SQL-Befehle werden mitprotokolliert (EVENT_COMMENT = abgesetzter SQL-Befehl, EVENT_KIND = 54). Nicht protokolliert werden Befehle, die über dynamisches SQL bearbeitet wurden (getestet mit DBMS_SQL).
- **trace_all_lines / trace_enabled_lines**: Jede einzelne Zeile, die abgearbeitet wird, wird mitprotokolliert (EVENT_COMMENT = "New line executed", EVENT_KIND = 51).

Mit Hilfe dieser Flags kann auch eingestellt werden, ob nur bestimmte Programmeinheiten ("**_enabled_**") oder alles ("**_all_**") protokolliert werden soll. Eine Programmeinheit wird "enabled", wenn sie mit der DEBUG-Option (re)kompiliert wird. Die Flags sind beliebig miteinander kombinierbar. Soll die Anzahl der gesammelten Informationen begrenzt werden, kann zusätzlich das Flag **trace_limit** gesetzt werden. Dadurch werden nur die letzten ca. 8000-9000 Einträge beibehalten (eine Angabe der Obergrenze kann nicht gemacht werden). Beginn und Ende des Tracing sowie Änderung der Flags werden immer in eigenen Einträgen mitprotokolliert.

Beispiel:

```
-- Als User SYS:
@?\rdbms\admin\tracetab.sql
GRANT SELECT ON PLSQL_TRACE_EVENTS TO SCOTT;
GRANT SELECT ON PLSQL_TRACE_RUNS TO SCOTT;
GRANT SELECT ON PLSQL_TRACE_RUNNUMBER TO SCOTT;

--Als User SCOTT:
ALTER PROCEDURE TRACE_TEST COMPILE DEBUG;
```

```
ALTER PROCEDURE test_proc COMPILE DEBUG;

PROCEDURE TRACE_TEST
   IS
   test_ex EXCEPTION;
BEGIN
   -- Tracing beginnen:
   DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.trace_enabled_exceptions);
   DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.trace_all_calls );
   /*
   alternativ:
   DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.trace_enabled_exceptions +
                  DBMS_TRACE.trace_all_calls);
   */
   test_proc;
   RAISE test_ex;
   -- Tracing beenden (wird hier aber wegen RAISE nicht abgearbeitet)
   DBMS_TRACE.CLEAR_PLSQL_TRACE;
EXCEPTION
   WHEN test_ex THEN
      DBMS_TRACE.CLEAR_PLSQL_TRACE;
   WHEN OTHERS THEN
      DBMS_TRACE.CLEAR_PLSQL_TRACE;
END;


SELECT *
  FROM SYS.PLSQL_TRACE_EVENTS
 WHERE RUNID=(SELECT MAX(RUNID)
              FROM SYS.PLSQL_TRACE_RUNS
              WHERE RUN_OWNER = 'SCOTT')
 ORDER BY event_seq;
-- oder:
SELECT EVENT_TIME, EVENT_KIND,
      EVENT_UNIT, EVENT_LINE, EVENT_PROC_NAME,
      PROC_NAME, PROC_LINE,
      USER_EXCP, EXCP ,EVENT_COMMENT
  FROM SYS.PLSQL_TRACE_EVENTS
 WHERE RUNID=(SELECT MAX(RUNID)
              FROM SYS.PLSQL_TRACE_RUNS
              WHERE RUN_OWNER = 'SCOTT')
 ORDER BY event_seq;
```