Neue Funktionen und Prozeduren in Oracle 11g R2

Ulrike Schwinn, ORACLE Deutschland B.V. & Co. KG

Ein Blick in das Handbuch „PL/SQL Packages and Types References 11*g* Release 2" gibt im Kapitel „What's New" einen ersten Überblick über die Veränderungen auf Package-Ebene. Doch wie kann man nun Neuigkeiten im Bereich der Unterprogramme finden, um neue und interessante Funktionen und Prozeduren in die eigenen Datenbank-Anwendungen einzuarbeiten? Der folgende Beitrag zeigt eine Möglichkeit, die Liste der neuen Funktionen und Prozeduren zu erstellen, und illustriert an einem Beispiel das neue Package DBMS_EXECUTE_PARALLEL zur Parallelisierung von Operationen.

Das genannte Handbuch enthält keine einheitliche Zusammenstellung der neuen Funktionen und Prozeduren. Es existiert nur eine Liste der neuen und veränderten Packages. Detaillierte Informationen lassen sich allerdings über ein SQL-Skript erzeugen. Damit die Liste die einzelnen Unterprogramme der entsprechenden Packages enthält, ist eine Abfrage auf DBA_OBJECTS auf ALL_ARGUMENTS zu erweitern. Dabei sollen nur diejenigen Objekte von Interesse sein, die dem User „PUBLIC" zur Verfügung stehen. Eine SELECT-Operation listet alle Funktionen und Prozeduren der aktuellen Datenbank auf (siehe Listing 1).

Diese Abfrage in einer „11*g* Release 2"-Installation liefert mehr als 7.800 Prozeduren und Funktionen! Um die neuen Unterprogramme in Release 2 im Unterschied zu Release 1 zu erhalten, ist ein Database-Link (hier o11) zu einer Release-1-Datenbank notwendig. Danach muss die Abfrage aus Listing 1 nur noch um eine MINUS- Operation erweitert werden. Listing 2 zeigt das Ergebnis.

```
select package_name, subprogram from (
   select distinct package_name, object_name subprogram from
all_arguments
   where package_name in (
         select name from (
           select object_name name from dba_objects
           where object_type = 'PACKAGE' and object_name in (
             select table_name from dba_synonyms where owner =
'PUBLIC'))))
PACKAGE_NAME                   SUBPROGRAM
------------------------------ ------------------------------
…
DBMS_ADVISOR                   CREATE_TASK
```

Listing 1: Liste aller Funktionen und Prozeduren

```
/* Abfrage aus Listing 1 */
…
MINUS
select package_name, subprogram from (
select distinct package_name, object_name subprogram from all_arguments@o11
where package_name in (
    select name from (
      select object_name name from dba_objects@o11
      where object_type = 'PACKAGE' and object_name in (
        select table_name from dba_synonyms@o11 where owner = 'PUBLIC'))))
PACKAGE_NAME                   SUBPROGRAM
------------------------------ ------------------------------
…
DBMS_UTILITY                   WAIT_ON_PENDING_DML
…
```

Listing 2: Die Liste der neuen Unterprogramme

Die Ausgabe ergibt insgesamt noch 663 Zeilen. Die Liste der Packages beziehungsweise der Unterprogramme gibt einen guten Überblick über die neuen Features in 11*g* Release 2. Beispielsweise enthält das Package DBMS_DBFS_CONTENT die Neuerungen des Database Filesystems, das Package DBMS_COMPRESSION implementiert den neuen Compression Advisor und DBMS_PARALLEL_EXECUTE die aktuellen Funktionen zur Parallelisierung. Zusätzlich gibt es neue Funktionen in existierenden Packages wie zum Beispiel im Package DBMS_UTILITY das Unterprogramm WAIT_ON_PENDING_DML, das über den Status („offen" oder „pending") einer Transaktion Auskunft geben kann. Alle neuen Programme zu erläutern, würde den Rahmen dieses Beitrags sprengen, daher liegt der Fokus im Folgenden auf der Beschreibung des Packages DBMS_PARALLEL_EXECUTE, das ein Hilfsmittel zur Prozess-Parallelisierung darstellt.

Automatische Parallelisierung mit DBMS_PARALLEL_EXECUTE

Parallelisierung von DML-Operationen (Parallel DML) – beispielsweise um eine große Menge an Daten mit einer UPDATE-Operation anzupassen

oder einen monatlichen Datenrefresh mit MERGE durchzuführen – ist seit jeher mit den entsprechenden „ALTER SESSION“-Einstellungen möglich. Diese Art der Ausführung führt allerdings zu einigen Einschränkungen: So lassen sich keine weiteren seriellen oder parallelen Operationen in der gleichen Transaktion durchführen. Integritäts-Constraints, Trigger oder Clustertabellen werden ebenfalls nicht unterstützt. Eine vollständige Liste und detaillierte Beschreibungen zu den Einschränkungen stehen im Handbuch „Oracle Database VLDB and Partitioning Guide 11*g* Release 2 (11.2)“.

Eine Alternative zu dieser Vorgehensweise ist die manuelle Aufteilung der Tabelle in mehrere „Chunks“, zum Beispiel über feste ROWID-Bereiche und anschließende Verarbeitung in parallelen Sessions.

In 11*g* Release 2 lässt sich diese Aufgabe durch das neue Package

DBMS_PARALLEL_EXECUTE automatisieren. Die Idee ist, dass ein spe-

```
 DECLARE
  l_task     VARCHAR2(30) := ‚test_task‘;
  l_sql_stmt VARCHAR2(32767);
  l_try      NUMBER;
  l_status   NUMBER;
BEGIN
  DBMS_PARALLEL_EXECUTE.create_task (task_name => l_task);
  DBMS_PARALLEL_EXECUTE.create_chunks_by_rowid(task_name   =>
l_task,
                                              table_owner =>
‚SCOTT‘,
                                              table_name  =>
‚TEST_TAB‘,
                                              by_row      =>
TRUE,
                                              chunk_size  =>
10000);
  l_sql_stmt := ‚UPDATE /*+ MONITOR */test_tab t SET t.num_col =
t.num_col
                 + 10 WHERE rowid BETWEEN :start_id AND :end_id‘;
  DBMS_PARALLEL_EXECUTE.run_task(task_name      => l_task,
                                 sql_stmt       => l_sql_stmt,
                                 language_flag  => DBMS_SQL.NA-
TIVE,
                                 parallel_level => 10);
  DBMS_PARALLEL_EXECUTE.drop_task(l_task);
END;
/
```

Listing 3: Beispielanwendung

```
SQL> select job_name, start_rowid, end_rowid, status
     from dba_parallel_execute_chunks order by 1

JOB_NAME        START_ROWID         END_ROWID           STATUS
--------------- ------------------ ------------------ ------------------
--
TASK$_1300_1    AAAWkpAAFAAFTIAAAA AAAWkpAAFAAFTIxCcP ASSIGNED
TASK$_1300_1    AAAWkpAAFAAFS2yAAA AAAWkpAAFAAFS3jCcP PROCESSED
TASK$_1300_1    AAAWkpAAFAAFTAAAAA AAAWkpAAFAAFTAxCcP PROCESSED
TASK$_1300_1    AAAWkpAAFAAFStkAAA AAAWkpAAFAAFSt/CcP PROCESSED
TASK$_1300_10   AAAWkpAAFAAFS2AAAA AAAWkpAAFAAFS2xCcP PROCESSED
…
TASK$_1300_9    AAAWkpAAFAAFS3kAAA AAAWkpAAFAAFS3/CcP PROCESSED
TASK$_1300_9    AAAWkpAAFAAFS/kAAA AAAWkpAAFAAFS//CcP PROCESSED
TASK$_1300_9    AAAWkpAAFAAFTJkAAA AAAWkpAAFAAFTJ/CcP PROCESSED
                AAAWkpAAFAAFSbAAAA AAAWkpAAFAAFSbHCcP UNASSIGNED
                AAAWkpAAFAAFTQyAAA AAAWkpAAFAAFTRjCcP UNASSIGNED
                AAAWkpAAFAAFSbQAAA AAAWkpAAFAAFSbXCcP UNASSIGNED
                AAAWkpAAFAAFTQAAAA AAAWkpAAFAAFTQxCcP UNASSIGNED
                AAAWkpAAFAAFTOAAAA AAAWkpAAFAAFTOxCcP UNASSIGNED
...
73 rows selected.
```

Listing 4: Monitoring über DBA_PARALLEL_EXECUTE_CHUNKS

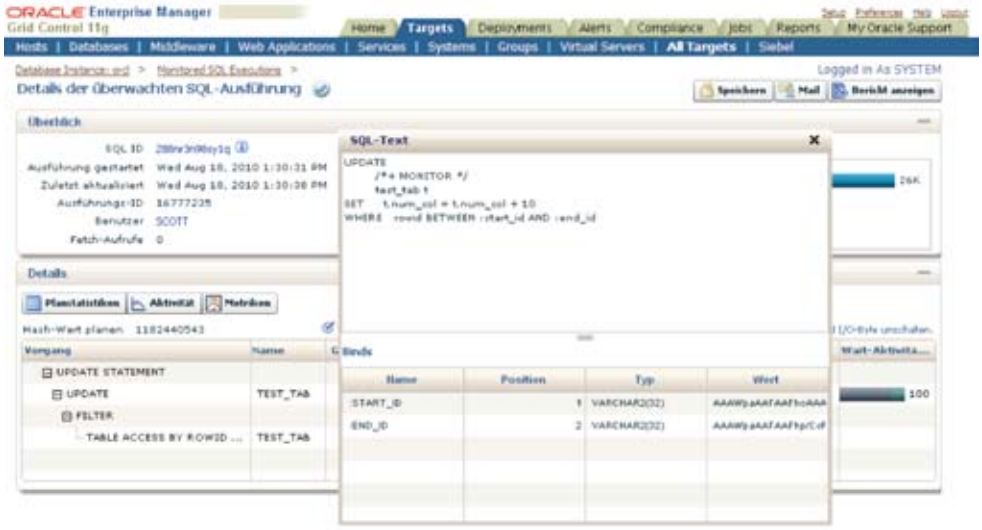

Abbildung 1: SQL Monitoring im Enterprise Manager

zieller Scheduler-Prozess die Tabelle automatisch in Teilbereiche aufteilt, parallel abarbeitet und eine COMMIT-Operation nach jedem abgearbeiteten Teilbereich durchführt. Folgende drei Aufrufe sind dafür notwendig:

1. Das Anlegen einer Aufgabe (auch Task) mit CREATE_TASK
2. Die Aufteilung der Tabelle in Teilbereiche (auch Chunks) durch eine definierte Methode. Zur Verfügung stehen die ROWID-, die Spaltewert- oder die SELECT- Methode.
3. Der Ablauf des Statements mit RUN_TASK in den definierten Chunk-Bereichen

Um im Fehlerfall die Operation automatisch wieder zu starten, steht die Prozedur RESUME_TASK zur Verfügung, die zusätzlich verwendet werden kann. Folgendes Beispiel zeigt eine einfache Anwendung eines parallelisierten UPDATE-Statements (siehe Listing 3). Die Tabelle besteht aus 500.000 Zeilen und soll mit einem Parallelisierungsgrad von 10 bearbeitet werden. Die definierten Chunks haben dabei jeweils die Größe von 10.000 Zeilen.

Während der Ausführung kann man beobachten, dass zusätzliche Job-Prozesse mit Namen „ora_j00x_orcl“ gestartet werden – in unserem Fall handelt es sich um zehn weitere Prozesse. Detailliert überwachen lässt sich der Ablauf der Verarbeitung mit neuen Data Dictionary Views. Mit der View DBA_PARALLEL_EXECUTE_CHUNKS lässt sich zum Beispiel der aktuelle Stand der Ausführung überprüfen (siehe Listing 4). Wie man leicht erkennen kann, werden die ROW ID-Teilbereiche von den einzelnen Job-Prozessen abgearbeitet und befinden sich jeweils in einem unterschiedlichen Prozess-Status. Durch die Verwendung des Hints „MONITORW (siehe Listing 3) lässt sich zusätzlich die Abarbeitung im Enterprise Manager im Bereich „SQL Monitoring“ überwachen.

Die Vorteile der Nutzung sind offensichtlich. Durch die parallele Abarbeitung der Scheduler Job-Slaves kann eine schnelle Durchführung gewährleistet werden. Außerdem wird durch die zusätzliche Möglichkeit, den Status der Bearbeitung abzufragen und einen Wiederanlauf im Fehlerfall zu starten, eine erfolgreiche Erledigung sichergestellt. Möchte man eine parallele und skalierbare DML-Verarbeitung großer Tabellen ohne die Restriktionen von Parallel DML durchführen, stellt die Verwendung von DBMS_PARALLEL_EXECUTE somit eine gute Alternative dar.

Weitere Informationen

- Oracle Database PL/SQL Packages and Types Reference 11g Release 2 (11.2): http://download.oracle.com/docs/cd/E11882_01/appdev.112/e10577/toc.htm
- Deutschsprachige DBA-Community: http://www.oracle.com/global/de/community/dbadmin/index.html

Kontakt:
Ulrike Schwinn
Ulrike.Schwinn@oracle.com