Anleitung zur nachträglichen Partitionierung von Tabellen

In unserem Beitrag „Performance-Gewinn durch Techniken der Oracle Datenbank (Teil 3)“ wurde der Einsatz der Partitionierung vorgestellt. Doch was geschieht, wenn man eine Tabelle noch gar nicht partitioniert hat und diese nachträglich partitionieren möchte?
Um dieses Problem zu lösen, möchten wir in diesem Artikel kurz das Package „dbms_redefinition“ der Oracle Datenbank vorstellen.
Das Package hilft dem Anwender online eine Tabelle zu kopieren. Ergänzend dazu werden weitere Objekte der Tabelle (Constraints, Indizes etc.) automatisch erzeugt. Die Funktionsweise wird anhand des folgenden Beispiels beschrieben:

Als Ausgangspunkt wird eine Tabelle betrachtet, die nachträglich partitioniert werden soll. *(siehe Listing 1)*

Exemplarisch wird ein Primary Key, ein Kommentar sowie ein Default-Wert festgelegt. *(siehe Listing 2)*

Aufgabe: Die Tabelle „part_table“ soll „liefern“ als Partitionierungsschlüssel besitzen.

Zu Beginn des Prozesses muss eine leere neue Tabelle (hier: Suffix _copy) manuell erzeugt werden, die die Struktur der ursprünglichen Tabelle besitzt. Dabei muss darauf geachtet werden, dass diese partitioniert ist. *(siehe Listing 3)*

Somit ist eine leere Kopie der Tabelle ohne Schlüssel, Constraints und Spaltenkommentare entstanden. Durch folgenden Aufruf wird abgefragt, ob man die Tabelle „part_table“ mit dem Package „dbms_redefinition“ kopieren kann. *(siehe Listing 4)*

Wird von der Prozedur kein Fehler gemeldet, ist die Tabelle „part_table“ bereit, online kopiert zu werden.
Im nächsten Schritt wird nun die Tabelle kopiert. *(siehe Listing 5)*

Der erste Funktionsaufruf (Teil 1) gibt der Datenbank die Information, dass die Tabellen nun kopiert werden soll. Anschließend (Teil 2) werden die Daten kopiert. Dazu werden auch alle Schlüssel, Constraints etc. übernommen. Es gibt die Möglichkeit durch weitere Eingabeparameter Einstellungen des Kopiervorgangs zu ändern, falls bspw. Triggers nicht übernommen werden sollen (copy_triggers => false). Nach diesem Schritt entspricht „part_table_copy“ der Tabelle „part_table“ nur, dass sie partitioniert ist. Zum Schluss (Teil 3) werden die Namen der Tabellen ausgetauscht.

Durch das Abfragen auf die Repository-Tabellen user_col_comments, user_constraints und user_indexes kann man sehen, dass die neue Tabelle auch die Spaltenkommentare, Constraints und Indizes der Ursprungstabelle besitzt. Dabei besitzen die Indizes jetzt auch den Namen der Indizes der Ursprungstabelle, während die Indizes der ursprünglichen Tabelle durch einen Präfix TMP$$_ umbenannt wurden.

Fazit

Das Package dbms_redefinition ist eine Möglichkeit, um Tabellen online nachträglich zu partitionieren, dabei steht die Tabelle während dieses Vorgangs immer zur Verfügung. Ein weiterer Vorteil ist, dass durch wenige Zeilen Code, das gesamte DDL für die Erzeugung und Umbenennung aller abhängigen Tabellenobjekte, wie Spaltenkommentare, Constraints und Indizes der generiert wird.

Unvorteilhaft hingegen ist es, dass man nachträglich noch die Defaultwerte angeben muss. Des Weiteren sind auch die Constraints im Status NOT VALIDATED. Durch einen entsprechenden Aufruf „alter table user.part_table modify constraint pk_part_table enable validate“ kann das Problem im obigen Beispiel behoben werden.

Zudem ist zu beachten, dass nach der Kopie sowohl die neue Tabelle als auch die Ursprungstabelle den gleichen Datenbestand beinhalten. Bis zum manuellen Löschen der Ursprungstabelle (drop table user.part_table_copy) wird somit der doppelte Speicherplatz benötigt.

Das in diesem Artikel beschrieben Verfahren kann ebenfalls angewendet werden, wenn man die Spalten einer Tabelle umsortieren will.

Anleitung zur nachträglichen Partitionierung von Tabellen

Listing 1:

```
create table user.part_table as
select level id,
       mod(level,100) liefernr,
       trunc(sysdate,'dd') stichtag,
       'Beschreibung '||level Beschreibung,
       sysdate erstellt_am, sys_context('USERENV', 'OS_USER') erstellt_von
from dual
connect by level <=100000;
```

Listing 2:

```
alter table user.part_table  add constraint pk_part_table primary key (id, liefernr);
comment on column user.part_table.liefernr is 'Enthält die aktuelle Liefernr';
alter table user.part_table modify (liefernr default 0);
```

Listing 3:

```
create table user.part_table_copy
      PARTITION BY RANGE (liefernr)
      INTERVAL (1)
      (PARTITION VALUES LESS THAN (1))
      enable row movement
 as select * from user.part_table where 1=0;
```

Listing 4:

```
exec dbms_redefinition.can_redef_table('USER', 'PART_TABLE');
```

Listing 5:

```
declare
  num_errors pls_integer := 0;
begin
  --Teil 1
  dbms_redefinition.start_redef_table('USER', 'PART_TABLE', 'PART_TABLE_COPY');
  --Teil 2
  dbms_redefinition.copy_table_dependents(uname=>'USER', orig_table=>'PART_TABLE',
      int_table=>'PART_TABLE_COPY',num_errors=>num_errors);
  --Teil 3
  dbms_redefinition.finish_redef_table('USER', 'PART_TABLE', 'PART_TABLE_COPY');
end;
```