Partitioning in der Datenbank 12*c*: Was ist neu?

Jan Ott, Trivadis AG

Die neuen Features sollen die tägliche Wartung der Datenbank vereinfachen, die Verfügbarkeit erhöhen und die Performance verbessern – soweit die Ankündigungen. Dabei fallen Worte wie „Asynchronous", „Online", „Partial Indexes" und „Single Operation". Der Artikel zeigt, was es damit auf sich hat, was die neuen Partitioning-Features dazu beitragen und ob die Ziele im praktischen Einsatz erreicht werden.

Eine Herausforderung bei der täglichen Arbeit ist der Umgang mit Partitionen. Die Datenmengen werden größer und daher auf mehr Partitionen verteilt. Partitionen einfach und effizient verwalten zu können, ist von zentraler Bedeutung. Zum Beispiel lassen sich ältere Daten aus mehreren Partitionen zusammenlegen und gleichzeitig komprimieren oder Partitionen, die zu groß wurden, auf mehrere verteilen. Neu in der Datenbank 12*c* ist nun die Möglichkeit, mit einer Operation mehrere Partitionen zu erstellen (ADD), zu löschen (DROP/TRUNCATE), zusammenzuführen (MERGE) und aufzuteilen (SPLIT).

ADD – Multiple Partition

Mehrere Partitionen lassen sich am Ende einer bestehenden Partition oder neuer Listen mit einer Operation einfügen. Dieses Feature ist für die folgenden Partitionierungsarten verfügbar:

- Range/Composite
- Liste; nur, wenn keine Default-Partition existiert

Listing 1 zeigt ein Beispiel zum Einfügen mehrerer Jahre, pro Jahr eine Partition. Die Partitionen sind durch Komma getrennt aufzulisten.

MERGE – Multiple Partition

Dieses Feature zum Zusammenführen mehrerer Partitionen zu einer mit nur einer Operation ist für folgenden Partitionierungsarten verfügbar:

- Range; die Bereiche müssen aneinander grenzen

```
ALTER TABLE t_partition_range
ADD
  PARTITION p_2007
    VALUES LESS THAN (TO_DATE('01.01.2008','dd.mm.yyyy')),
  PARTITION p_2008
    VALUES LESS THAN (TO_DATE('01.01.2009','dd.mm.yyyy')),
  PARTITION p_2009
    VALUES LESS THAN (TO_DATE('01.01.2010','dd.mm.yyyy'))
```

Listing 1

```
ALTER TABLE t_partition_list
MERGE PARTITIONS p_fr, p_it, p_de INTO PARTITION p_weu
```

Listing 2

```
ALTER TABLE t_partition_list
MERGE PARTITIONS p_b TO p_d INTO PARTITION p_b_to_d
```

Listing 3

```
ALTER TABLE t_partition_range
SPLIT PARTITION p_2000 INTO (
  PARTITION p_2000_q1
    VALUES LESS THAN (TO_DATE('01.04.2000','dd.mm.yyyy')),
  PARTITION p_2000_q2
    VALUES LESS THAN (TO_DATE('01.07.2000','dd.mm.yyyy')),
  PARTITION p_2000_q3
    VALUES LESS THAN (TO_DATE('01.09.2000','dd.mm.yyyy')),
  PARTITION p_2000_q4
)
```

Listing 4

- List; die Listen werden zusammengeführt (UNION). Wenn eine der Partitionen die Default-Partition ist, wird sie erweitert
- Partition und Subpartition

In unserem Beispiel sind die einzelnen Partitionen durch Komma getrennt aufgelistet (siehe Listing 2). Ein Bereich ist durch zwei Partitionsnamen abgegrenzt (siehe Listing 3).

```
ALTER TABLE t_partition_range
DROP PARTITIONS p_2003, p_2004, p_2005
```

Listing 5

```
ALTER TABLE t_partition_range
TRUNCATE PARTITIONS p_2003, p_2004, p_2005
```

Listing 6

```
ALTER TABLE t_partition_range
DROP PARTITION p_2003 ,p_2004 UPDATE INDEXES;

Danach ist der globale Index VALID, hat jedoch Waisen (ORPHA-
NED_ENTRIES).
SELECT index_name, status, orphaned_entries
  FROM user_indexes
 WHERE index_name = 'PK_PARTITION_RANGE';

INDEX_NAME          STATUS   ORPHANED_ENTRIES
------------------- -------- -----------------
PK_PARTITION_RANGE  VALID    YES
```

Listing 7

```
BEGIN
  dbms_part.cleanup_gidx
    (schema_name_in => USER
    ,table_name_in => 'T_PARTITION_RANGE'
    );
END;
/

PL/SQL procedure successfully completed.

SELECT index_name, status, orphaned_entries
  FROM user_indexes
 WHERE index_name = 'PK_PARTITION_RANGE';

INDEX_NAME          STATUS   ORPHANED_ENTRIES
------------------- -------- -----------------
PK_PARTITION_RANGE  VALID    NO
```

Listing 8

SPLIT – Multiple Partition

Dieses Feature zum Verteilen der Daten einer Partition auf mehrere Partitionen in einer Operation ist für die folgenden Partitionierungsarten verfügbar:

- Range/List
- Partition/Subpartition

Listing 4 zeigt dazu ein Beispiel.

DROP – Multiple Partition

Dieses Feature zum Entfernen mehrerer Partitionen in einer Operation ist für folgende Partitionierungsarten verfügbar:

- Range/List
- Partition/Subpartition

Ein Beispiel dazu ist in Listing 5 zu sehen.

TRUNCATE – Multiple Partition

Dieses Feature zum Löschen mehrerer Partitionen in einer Operation ist für folgende Partitionierungsarten verfügbar:

- Range/List
- Partition/Subpartition

Listing 6 zeigt dazu ein Beispiel.

Alle Ziele erreicht?

Das Ziel der vereinfachten Wartung wurde erreicht. Es ist mit der neuen Syntax einfacher, aus einer Partition mehrere zu erstellen (SPLIT). Auch die Performance ist besser bei einem SPLIT, da die Ausgangspartition nur einmal gelesen wird. Zudem erfolgt alles in einem Schritt. Die Verfügbarkeit wird leider nur bedingt erreicht. Das System ist zwar schneller wieder verfügbar, bei hoher Last ist jedoch immer noch ein Wartungsfenster erforderlich.

Im Design steht man immer wieder vor der Frage „Können wir globale Indizes auf partitionierten Tabellen einsetzen?“ Ein „TRUNCATE“ oder „DROP“ einer Partition setzt den globalen Index auf „unusable“. Danach ist der Zugriff über diesen Index nicht mehr möglich; es wird ein „full table scan“ verwendet. Dabei kommt das System zum Stillstand. In der Version 11*g* gab es dafür die beiden Möglichkeiten, entweder keine globalen Indizes zu verwenden oder „TRUNCATE/DROP“ durch „DELETE“ zu ersetzen (was die Last enorm erhöht) und bei „DROP“ die Partition im Wartungsfenster zu entfernen.

Oracle geht diese Herausforderung jetzt an. Es bleibt ein globaler Index verfügbar („usable“). Bei diesem wird markiert, dass er Waisen („orphan“) besitzt. Diese verwaisten Einträge werden zu einem späteren Zeitpunkt aufgeräumt. Die Transaktion ist ein DDL-Statement und dauert daher, wie von „TRUNCATE/DROP“ gewohnt, nur sehr kurz. Es gelten folgende Verfügbarkeit und Einschränkungen:

- Range/List
- Partitionen/Subpartitionen
- Nur „Heap“-Tabellen, keine „Index only“-Tabellen (IOT)
- Keine Objekt-Types und Domain-Indizes

```
ALTER TABLE t_partition_range
MOVE PARTITION p_2003 ONLINE UPDATE INDEXES

dbms_part.cleanup_online_op(
  schema_name    => USER,
  table_name     => 'T_PARTITION_RANGE',
  partition_name => 'P_2003'
);
```

Listing 9

```
CREATE TABLE t_partition_range(
   id      INTEGER NOT NULL CONSTRAINT pk_partition_range
                                    PRIMARY KEY USING IN-
DEX
  ,name    VARCHAR2(100)
  ,time_id DATE
)
-- setzen für die Tabelle => default für die Partitionen
INDEXING OFF
PARTITION BY RANGE (time_id) (
  -- nicht spezifiziert => übernimmt das Setting der Tabelle
  PARTITION p_2000
    VALUES LESS THAN (TO_DATE('01.01.2001','dd.mm.yyyy'))
  -- kann gesetzt werden, nicht nötig, das gleich wie Tabelle
 ,PARTITION p_2001
    VALUES LESS THAN (TO_DATE('01.01.2002','dd.mm.yyyy'))
    INDEXING OFF
  -- für die Partition ist das Indexieren eingeschaltet
 ,PARTITION p_2002
    VALUES LESS THAN (TO_DATE('01.01.2003','dd.mm.yyyy'))
  INDEXING ON)
```

Listing 10

```
ALTER TABLE t_partition_range
ADD PARTITION p_2003
  VALUES LESS THAN (TO_DATE('01.01.2004','dd.mm.yyyy'))
  INDEXING ON

SELECT table_name, partition_name, indexing
  FROM user_tab_partitions
 WHERE table_name = 'T_PARTITION_RANGE'
 ORDER BY partition_name;

TABLE_NAME          PARTITION_NAME  INDEXING
------------------- --------------- ----------
T_PARTITION_RANGE           P_2000        OFF
T_PARTITION_RANGE           P_2001        OFF
T_PARTITION_RANGE           P_2002        ON
T_PARTITION_RANGE           P_2003        ON
```

Listing 11

Bei „DROP“ oder „TRUNCATE“ muss neu „UPDATE INDEXES“ verwendet werden. Der Index wird wie bisher erstellt. Die Maintenance des Index übernimmt Oracle automatisch durch den Job „SYS.PMO_DEFERED_GIDX_MAINT_JOB“. Dieser läuft per Default um zwei Uhr in der Nacht. Der Zeitpunkt lässt sich anpassen. Zudem kann die Wartung mit dem „DBMS_PART“-Paket jederzeit injiziert werden. Listing 7 zeigt dazu ein Beispiel (bitte „UPDATE INDEXES“ beachten). In Listing 8 wird der Index aufgeräumt und nicht dem Wartungsjob überlassen.

Das Ziel „Asynchronous Global Index Maintenance” ist auf der ganzen Linie erreicht. Globale Indizes können nun vermehrt verwendet werden. Die Performance liegt im Subsekunden-Bereich, da das DDL-Statement eine Data-Dictionary-Änderung ist. Der Index wird zu einem späteren Zeitpunkt aufgeräumt. Dies kann in einem Wartungsfenster erledigt oder dem automatischen Job „PMO_DEFERED_GIDX_MAINT_JOB“ überlassen werden. Somit ist auch die Verfügbarkeit erhöht, da der Index immer online ist.

Online Move Partition

Die Oracle-Datenbank muss verfügbar sein, die Wartungsfenster werden immer kürzer. Die Möglichkeit, Partitionen während des Betriebs zu reorganisieren, um Platz freizugeben, sie in einen neuen Tablespace zu verschieben, zu komprimieren und/oder Parameter zu verändern, ist daher sehr willkommen. Wenn es zu Problemen kommt, stellt Oracle das Paket „DBMS_PART“ zur Verfügung. Listing 9 zeigt ein Beispiel. Zu beachten ist das Schlüsselwort „ONLINE“. Mit „UPDATE INDEXES“ wird erreicht, dass die Indizes verfügbar bleiben.

Auch hier ist das Ziel erreicht. Die Daten bleiben während der Reorganisation verfügbar. Ein Wartungsfenster ist nicht erforderlich – es besteht nur eine kleine Einschränkung. Auf der Partition wird ein exklusives „Lock“ benötigt. Dies kann bei Systemen mit hoher Last ein Ding der Unmöglichkeit sein, zum Beispiel, um Partitionen mit Vergangenheitsdaten zu kompri-

```
CREATE INDEX idx_parr_name_partial ON t_partition_range
(name)
GLOBAL INDEXING PARTIAL
```

Listing 12

```
SELECT index_name, status, num_rows
  FROM user_indexes
 WHERE index_name IN (‚PK_PARTITION_RANGE‘
                     ,‘IDX_NAME_GLOBAL_PARTIAL‘);
```

Listing 13

```
INDEX_NAME                STATUS   NUM_ROWS
------------------------- -------- ----------
PK_PARTITION_RANGE        VALID    48
IDX_NAME_GLOBAL_PARTIAL   VALID    24
```

Listing 14

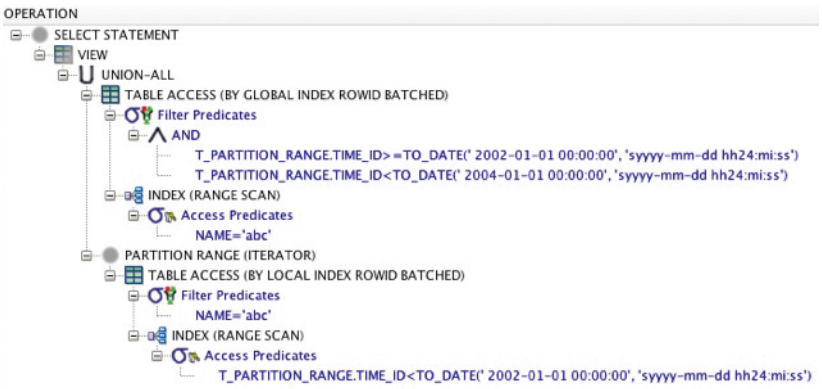

Abbildung 1: Ein Explain Plan im SQL Developer, Partial Index 1

```
CREATE TABLE t_partition_parent (
  pk INTEGER NOT NULL CONSTRAINT pk PRIMARY KEY USING INDEX,
  i  INTEGER
)
PARTITION BY RANGE (pk) INTERVAL (10)
(PARTITION p_1 VALUES LESS THAN (10))
```

Listing 15

mieren oder in einen anderen Tablespace auslagern, da auf die Daten weniger zugegriffen wird. Es gilt, ein Zeitfenster mit kleiner Last zu finden, zum Beispiel am Wochenende oder in der Nacht, um die Daten dann online zu reorganisieren. Da das System weiterläuft, können vereinzelte Benutzer arbeiten und werden nicht von einem Wartungsfenster behindert.

Partial Indexes for Partitioned Tables

Es wird immer wieder die Frage gestellt, wie viele Indizes man verwenden sollte. Ein Grund dafür ist der Platzbedarf. Bei den neuen partiellen Indizes kann der Index pro Partition ein- oder ausgeschaltet werden. Die Partition des ausgeschalten Index ist leer, braucht somit keinen Platz und muss nicht nachgeführt werden. Es gelten folgende Verfügbarkeit und Einschränkungen:

- Globale Indizes
- Lokale Indizes

Listing 10 zeigt dazu ein Beispiel. Im ersten Schritt sind die Tabelle und die Partitionen entsprechend zu definieren. Bei neuen Partitionen kann man das Indexierungsverhalten angeben, dabei wird der Index eingeschaltet (siehe Listing 11). Nun ist der Index partiell zu definieren, in Listing 12 als globaler Index.

Die Anweisung in Listing 13 zeigt, was im partiellen Index steht. Die Tabelle enthält 48 Zeilen, 12 pro Partition. Der normale Index „PK_PARTITION_RANGE“ hat 48 Zeilen. Doch im partiellen Index sind nur 24 (siehe Listing 14).

Interessant ist, wie Oracle die Daten selektiert (siehe Abbildung 1): „SELECT * FROM t_partition_range WHERE name = ‚abc‘;“. Der Optimizer löst das vorherige Select mit einem „UNION-ALL“ auf. Er verwendet den globalen partiellen Index für die Partitionen mit dem eingeschalteten Index. Für den Rest der Daten macht er einen Scan auf die Partitionen, die nicht im Index sind.

Partielle Indizes sind hilfreich für Tabellen, in denen zusätzliche Indizes für bestimmte Bereiche einer Tabelle gebraucht werden. Es ist nun möglich, diese individueller dem Abfrage-Muster anzupassen. Leider gibt es nur ein „ON/OFF“-Muster für alle partiellen Indizes einer Tabelle.

Interval-Reference-Partitioning

Interval-Partitioning ist seit der Version 11*g* eine Erleichterung. Es kann ein Intervall angegeben werden und Oracle fügt die benötigten Partitionen automatisch an. Das Intervall lässt sich nun auch auf eine Kind-Tabelle übertragen. Im ersten Schritt erstellt man eine „INTERVAL“-Vater-Tabelle (siehe Listing 15). Der zweite Schritt ist das

```
CREATE TABLE t_partition_child(
  fk INTEGER NOT NULL,
  i  INTEGER,
  CONSTRAINT part_child_fk FOREIGN KEY (fk)
                           REFERENCES t_partition_parent(pk)
)
PARTITION BY REFERENCE(part_child_fk)
```

Listing 16

Erstellen einer Kind-Tabelle (siehe Listing 16). Die Referenz auf die Vater-Tabelle war in der Version 11*g* nicht möglich.

Interval-Reference-Partitioning ist sicher eine nützliche Ergänzung und kann die Wartung vereinfachen, da man beispielsweise Vater-Kind in einer Operation austauschen kann („exchange"). Neue Partitionen werden automatisch angelegt und synchron zwischen Vater- und Kind-Partitionen gehalten. Oracle führt darüber hinaus mit der Version 12*c* noch drei weitere Features ein:

- „TRUNCATE TABLE" nun mit der „CASCADE"-Option
- XML DB und Domain Index unterstützen Hash-Partitionierung
- Statistiken unterstützen inkrementelle Statistiken für die „Exchange"-Partition

Fazit

Die Partitionierung wurde von Oracle in die richtige Richtung entwickelt. Die Wartung ist vereinfacht, da globale Indizes auf partitionierte Tabellen erstellt und die effizienten Befehle „DROP" und „TRUNCATE" verwendet werden können. Ein weiterer Vorteil ist das Interval-Reference-Partitioning, wobei die Datenbank die Erstellung der benötigten Partitionen der Vater- und Kind-Tabelle automatisch übernimmt.

Die Performance ist durch die „SPLIT/MERGE"-Partition erhöht, so muss zum Beispiel beim Split die Partition nicht mehrmals gelesen werden. Auch die Verfügbarkeit ist gestiegen, „Online Move"-Partitionen lassen sich nun ohne Wartungsfenster verschieben.

Eine Sache stört allerdings immer noch: Es muss der richtige Zeitpunkt abgewartet werden. Bei hoher Last erhält die Datenbank das „exklusive Lock" nicht und der Prozess bricht ab. Bei hoher Last auf der Partition braucht es also weiterhin ein Wartungsfenster.

Immerhin sind einige Ziele erreicht. Wer mit Partitionen arbeitet, wird die kleinen, aber feinen Verbesserungen zu schätzen wissen.

Jan Ott
jan.ott@trivadis.com

Oracle Data Integrator 12c mit neuer Benutzeroberfläche und Golden Gate mit Mandantenfähigkeit-Unterstützung

Mit dem Major Update seines Data-Integrations-Portfolios bringt Oracle die Version 12c von Data Integrator (ODI) und Golden Gate heraus. Die Entwicklung vom Warehouse Builder indes wurde eingestellt.

Oracle hat dem Data Integrator eine neue Benutzeroberfläche verpasst. Mit ihrem deklarativen, Flow-basierten Ansatz sorgt sie für eine einfache Bedienung. Zudem kann die Mapping-Logik wiederverwendet werden, was für kürzere Entwicklungszeiten sorgt. In Sachen Performance hat Oracle an der Parallelität der Daten-Integrationsprozesse gearbeitet.

Oracle hat auch die Interoperabilität zwischen dem Data Integrator und dem Oracle Warehouse Builder (OWB) – dessen Entwicklung eingestellt wurde – verbessert. Dies soll OWB-Kunden eine einfachere Migration nach ODI ermöglichen. Die enge Integration mit dem Enterprise Manager 12c sorgt für ein besseres, zentrales Monitoring. Darüber hinaus ist auch ein engeres Zusammenspiel zwischen ODI und Golden Gate vorgesehen. Dies ermöglicht ein effizienteres Laden und Transformieren von Real-Time-Daten.

Bei Oracle Golden Gate hat Oracle vor allem an der Optimierung für Oracle Database 12c gearbeitet. Dabei unterstützt das Produkt die mandantenfähige Architektur der neuen Datenbank und die Cloud-basierte Echtzeit-Replikation.

Für eine bessere Performance und Skalierbarkeit liefert Oracle extra für Oracle Golden Gate eine einfache Streaming-API.

Im Bereich Security hat das Development-Team die Integration mit Oracle Credential Store und Oracle Wallet weiterentwickelt, die den Anwndern das Speichern und Zurückholen von verschlüsselten Benutzernamen und Passwörtern ermöglicht.

In puncto Hochverfügbarkeit hilft die Integration mit dem Data Guard und Fast-Start-Fail-Over (FSFO) für mehr Ausfallsicherheit.

Darüber hinaus bringt Golden Gate 12c eine bessere Unterstützung von Echtzeit-Replikation von Daten in heterogenen Umgebungen mit MySQL, Microsoft SQL Server, Sybase nutzen IBM DB2 mit sich.