Reference Partitioning in der Praxis

DOAG-Konferenz 2013

Schwerpunkte

- Partitionierung in Oracle – Methoden und ihre Vorteile
- Ein Kundenprojekt
- Besonderheiten und Anforderungen von Reference-Partitionierung
- Migration des Datenmodells und der Daten
- Wartung der Partitionierung
- Neu in Oracle 12c
- Probleme

Vorteile durch Partitionierung

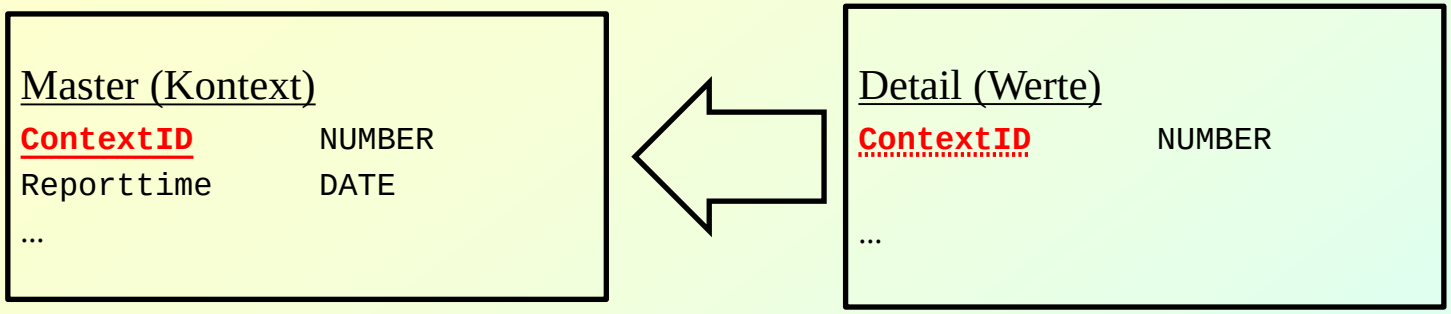
1. Performance
   - Partition Pruning: Es werden nur die Partitionen gelesen, die der WHERE-Klausel entsprechen, unabhängig von der Indizierung.
   - Partitionsweise Joins
   - Verschiedene Indizierungsmöglichkeiten, global und lokal
2. Wartung
   - Partitionen können auf unterschiedliche Tablespaces und damit auf verschiedene Dateien verteilt werden.
   - Partitionierung verbessert das Life Cycle Management für die Daten
   - Wartung mittels Partitionierung basiert auf DDL (z.B. `ALTER TABLE … DROP PARTITION`) ist wesentlich schneller als eine vergleichbare Wartung mit DML (z.B. `DELETE`, weil kein UNDO generiert wird und sich das Logging minimieren läßt.

Nachteil: Partitionierung ist eine kostenpflichtige Option zusätzlich zur Enterprise Edition

Partitionierung – Methodenüberblick

Oracle Release	**Methode**
Oracle 8	Range-Partitionierung
Oracle 8*i*	Hash- und Composite Range-Hash-Partitionierung
Oracle 9*i*	List-Partitionierung
Oracle 9*i* R2	Composite Range-List-Partitionierung
Oracle 11*g* R1	Weitere Composite-Partitionierungen (Alle denkbaren Kombinationen) Reference-Partitionierung Virtual-Column-Partitionierung Interval-Partitionierung System Partitionierung
Oracle 12*c* R1	Interval-Reference-Partitionierung

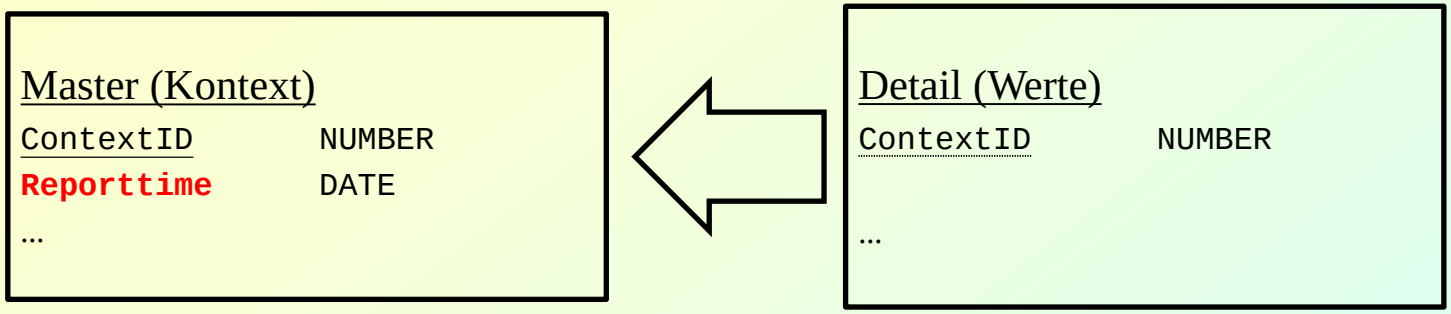
Kundenprojekt - Ausgangssituation



1. Datenmodell
   - Viele (knapp 100) Detail-Tabellen enthalten den Primärschlüssel der Master-Tabelle
   - Das Datenmodell ist in etwa sternförmig
   - Die Tabellen (Master-Tabelle und einige Detail-Tabellen) sind nach der `ContextID` range-partitioniert
2. Probleme
   - Anlegen von Partitionen auf Vorrat, da nicht genau bekannt ist, wie viele ContextID pro Zeiteinheit generiert werden
   - Kein Partition Pruning für zeitbasierte Abfragen
   - Komplizierte Wartung (Löschen und Archivieren von Daten bzw. Partitionen)

Frage: Kann Reference-Partitionierung hier helfen?

Was ist Reference Partitionierung?



Es handelt sich dabei um die Partionierung entlang von Fremdschlüsselbeziehungen. Die abhängigen Tabellen werden nach dem gleichen Kriterium und nach der gleichen **Methode** wie die Mastertabelle partitioniert, ohne daß sie den Partitionierungsschlüssel explizit enthalten müssen. Im vorliegenden Beispiel bedeutet das eine Partitionierung nach der *Reporttime* und nicht nach der *ContextID*. Was sind die Vorteile:

- Tabellen mit Master-Detail-Beziehungen können gleich partitioniert werden, ohne daß der Partitionierungsschlüssel in der abhängigen Tabelle dupliziert werden muß.
- Wartungsmaßnahmen an der Partitionierung der Master-Tabelle werden automatisch in die abhängigen Tabellen kaskadiert, was die Wartung sehr vereinfacht und Erweiterungen des Datenmodells konsistenter und weniger fehleranfällig macht.
- Partionsweise Joins funktionieren auch, wenn das Join-Attribut nicht der Partitionierungsschlüssel ist.
- Reference-Partitionierung ist auch sinnvoll für die Partitionierung von Nested Tables.

Wie wird es gemacht?

Zunächst wird die Master-Tabelle angelegt:

```
  CREATE TABLE "MASTER"
  (
    "CONTEXTID" NUMBER(*,0) NOT NULL,
    "REPORTTIME" TIMESTAMP (6) NOT NULL,
    <weitere Spalten>
    CONSTRAINT "MASTER_PK" PRIMARY KEY ("CONTEXTID")
)
  PARTITION BY RANGE (REPORTTIME)
     (
     partition dez2012 values less than ( to_date( '1.1.2013', 'dd.mm.yyyy') ),
     partition jan2013 values less than ( to_date( '1.2.2013', 'dd.mm.yyyy') ),
     partition maxpart values less than (maxvalue)
     );
```

Dann die abhängigen Tabellen (Beispiel):

```
 CREATE TABLE "DETAIL"
  (
    "CONTEXTID"     NUMBER(*,0) NOT NULL,
    "WAFERMATERIALID" NUMBER(*,0) NOT NULL,
     <weitere Spalten>
    CONSTRAINT "DETAIL_CONTEXT_FK" FOREIGN KEY ("CONTEXTID") REFERENCES "MASTER"
     ("CONTEXTID")
  )
     PARTITION BY REFERENCE ("DETAIL_CONTEXT_FK");
```

Voraussetzungen und Restriktionen

- Die Fremdschlüsselspalten dürfen keine `NULL`-Marken enthalten und müssen explizit als `NOT NULL` deklariert werden.
- Damit man die Reference-Partitionierung in der Detail-Tabelle deklarieren kann, muß man die Fremdschlüssel-Constraints benennen.
- Diese Fremdschlüssel dürfen nicht ausgeschaltet (`DISABLE`) werden.
- Desgleichen müssen diese Constraint einschließlich der referenzierten Primärschlüssel eine transaktionsbezogene Prüfung (`DEFERRABLE`) ausschließen.
- Eine abhängige Tabelle kann Referenzen in mehrere Master-Tabellen besitzen. Für die Reference-Partitionierung kann aber nur immer eine verwendet werden.
- Eine kaskadierende Reference-Partitionierung ist aber möglich.

Fragen im Projekt

- Ist auf allen in Frage kommenden Spalten ein Fremdschlüssel deklariert?
- Gibt es in den abhängigen Tabellen Datensätze, die `NULL`s in der Fremdschlüsselspalte enthalten?
- Sind alle Fremdschlüsselspalten `NOT NULL` deklariert?
- Waren Fremdschlüssel-Constraints auf `DEFERRABLE` gestellt?
- Waren die Fremdschlüssel-Constraints benannt?
- Auf welchem Wege konnten das Datenmodell und die Daten in die neue Struktur migriert werden?

Ansonsten war das Datenmodell durch seine flache Hierarchie und die nahezu ausschließliche Referenz auf eine Master-Tabelle gut für die Reference-Partitionierung geeignet.

Migration eines vorhandenen Datenmodells

Zwei Methoden wurden in Betracht gezogen:

1. Offline durch Neuanlegen des Datenmodells mit Reference-Partitionierung und anschließendem Import der Daten aus einem aktuellen Dump. Beim Import sind die Abhängigkeiten zu berücksichtigen. Er wurde im Modus `DATA_ONLY` und in mehreren Schritten durchgeführt. Von Nachteil ist die von der Datenmenge abhängige Downtime.
2. Online mit Hilfe des Package `DBMS_REDEFINITION`

Letzteres war leider im vorliegenden Fall nicht möglich, wegen des vorliegenden Datenbank-Releases 11.2.0.3.0. Da gibt es aber infolge des Fixes für Bug 9777229 den Bug 13572659, in dessen Konsequenz während der Redefinition die Fremdschlüssel ausgeschaltet werden, was einer der Voraussetzungen für die Reference-Partitionierung widerspricht. (Gefixt ≥ 11.2.0.3.4)

Migration mit DBMS_REDEFINITION

- Anlegen einer Interimstabelle für die Mastertabelle (gleiche Struktur, aber neuer Name) ohne Constraints, aber mit der neuen Partitionierung nach dem Zeitstempel
- Abschalten aller Fremdschlüssel-Constraints, die auf die Mastertabelle zeigen
- Überprüfen, ob die Online-redefinition möglich ist mit `DBMS_REDEFINITION.CAN_REDEF_TABLE`
- Start der Redefinition mit `DBMS_REDEFINITION.START_REDEF_TABLE`
- Transfer der Constraints auf die Interimstabelle mit `DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS`
- Optionale Synchronisation der beiden Tabellen mit `DBMS_REDEFINITION.SYNC_INTERIM_TABLE`
- Statistiken für die neue Tabelle sammeln
- Beenden der Redefinition mit `DBMS_REDEFINITION.FINISH_REDEF_TABLE`
- Löschen der alten Tabelle, die jetzt den Namen der Interimstabelle hat
- Dieser Vorgang wird für die abhängigen Tabellen wiederholt, wobei hier die Interimstabellen mit Reference-Partitionierung angelegt werden.

Wartung

- Durch die Reference-Partitionierung erleichtert sich die Verwaltung der Tabellen enorm. Alle Wartungsmaßnahmen werden allein an der Master-Tabelle durchgeführt, weil z.B. `SPLIT PARTITION` und `DROP PARTITION` direkt an die abhängigen Tabellen propagiert werden. An der Master-Tabelle müssen diese Befehle manuell oder durch ein Skript ausgeführt werden. Das ist der Stand in Oracle 11*g* einschließlich Release 2.
- In Oracle 12c ist es möglich, Reference-Partitionierung mit **Intervall-Partitionierung** zu kombinieren. Dazu braucht man nur für die Master-Tabelle die RANGE-Partitionierung durch die Spezialform Intervall-Partitionierung ersetzen. Für die Deklaration der Detail-Tabellen ändert sich nichts.

```
CREATE TABLE "MASTER"
  (
    "CONTEXTID" NUMBER(*,0) NOT NULL,
    "REPORTTIME" TIMESTAMP (6) NOT NULL,
    <weitere Spalten>
    CONSTRAINT "MASTER_PK" PRIMARY KEY ("CONTEXTID")
)
  PARTITION BY RANGE (REPORTTIME) INTERVAL (NUMTOYMINTERVAL(1,'month'))
(
partition dez2012 values less than ( to_date( '1.1.2013', 'dd.mm.yyyy') ),
partition jan2013 values less than ( to_date( '1.2.2013', 'dd.mm.yyyy') )
);
```

Problem beim SPLIT PARTITION

Benutzer meldeten während des SPLIT PARTITION massive Performanceprobleme. Die ASH- und AWR-Reports zeigten folgendes:

- *Instanz 2* (INSERT durch die Applikation)

Top User Events

Event	Event Class	% Event	Avg Active Sessions
enq: TM - contention	Application	90.43	5.19

Top SQL with Top Events

SQL ID	Planhash	Sampled # of Executions	% Activity	Event	% Event	Top Row Source	% RwSrc	SQL Text
1y619zh0k5zxb	1439585339	532	80.27	enq: TM - contention	79.50	LOAD TABLE CONVENTIONAL	79.50	INSERT INTO DCContext (dcconte...
770d5q6wa07am	2676511830	1	10.93	enq: TM - contention	10.93	UPDATE	10.93	UPDATE TREIS_PE_MODULKLASSIFIZ...

Top Blocking Sessions

Blocking Sid (Inst)	% Activity	Event Caused	% Event
137, 8367(1)	80.46	enq: TM - contention	80.46
23, 8245(1)	6.96	enq: WF - contention	6.96

- *Instanz 1* (SPLIT PARTITION durch Job)

Complete List of SQL Text

SQL Id	SQL Text
0u4n71r93jsxa	CREATE TABLE "MODENG"."TMT3S_PROCESS"PARTITION BY HASH ("DCCONTEXTID") (PARTITION "OCT2013" SEGMENT CREATION IMMEDIATE TABLESPACE "MODDATA" NOCOMPRESS PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0 OBJNO 125513 DATAOBJNO 125513) LOGGING , PARTITION "MAXPART" SEGMENT CREATION IMMEDIATE TABLESPACE "MODDATA" NOCOMPRESS PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0 OBJNO 83155 DATAOBJNO 125512) LOGGING) NOPARALLEL AS SELECT /*+ USE_HASH(t) RELATIONAL(t) */ T.*, L.L$MARKER+1 SYS_PART# FROM SYS."L$301" L, "MODENG"."TMT3S_PROCESS" T WHERE T."DCCONTEXTID" = L."DCCONTEXTID" AND ((DATAOBJ_TO_PARTITION("MODENG"."TMT3S_PROCESS", 120996) <= TBLORIDX$PART$NUM("MODENG"."TMT3S_PROCESS", 0, 0, 0, t.rowid) AND TBLORIDX$PART$NUM("MODENG"."TMT3S_PROCESS", 0, 0, 0, t.rowid) <= DATAOBJ_TO_PARTITION("MODENG"."TMT3S_PROCESS", 120996))) UPDATE GLOBAL INDEXES NOLOGGING

Dinge, die man beachten sollte

- Wartungsmaßnahmen wie das `SPLIT PARTITION` von der vorherigen Folie sollten möglichst präventiv, ohne vorhandene Daten durchgeführt werden.
- Alle Operationen, die zur Bewegung von Zeilen führen, sind mit allergrößter Vorsicht und nach entsprechenden Tests zu verwenden. Das betrifft insbesondere Updates von Partitionierungsschlüsseln in der Master-Tabelle, die zu massenhaften Updates der abhängigen Tabellen führen. (Ungeachtet der Tatsache, daß man dazu das `ROW MOVEMENT` einschalten muß, wenn sich dadurch die Partition ändert.)
- Solche Updates sind auch mit der Gefahr von Deadlocks verbunden.
- Das Entfernen alter Daten durch `DELETE` sollte möglichst vermieden werden.
- Die Fremdschlüsselspalten, über die die Reference-Partitionierung realisiert wird, sollten indiziert werden.
- Die Indizierung der abhängigen Tabellen ist darüber hinaus mit Bedacht vorzunehmen.

Quellen

Oracle® Database VLDB and Partitioning Guide
Oracle® Database SQL Language Reference
Oracle® Database PL/SQL Packages and Types Reference

Dr. Frank Haney
info@haney.it
Tel.: 03641-210224