Performance-Gewinn durch Techniken der Oracle Datenbank (Teil 3)

Im dritten Teil unserer Serie über grundlegende Techniken der Oracle-Datenbank in Data Warehouse Systemen geht es um das Thema „Partitionierung“. Durch den richtigen Einsatz von Partitionierung können trotz steigendem historischem Datenvolumen weitestgehend konstante Laufzeiten sowohl für Abfragen – zum Beispiel in Berichten und Dashboards – als auch für ETL-Prozesse erzielt werden.
Bei der Partitionierung werden die Daten von Tabellen und Indizes anhand einer Tabellenspalte, dem Partitionsschlüssel, aufgeteilt und in physisch separate Objekte gespeichert. Beim Zugriff per SQL-Befehl werden sie jedoch weiterhin als eine logische Einheit angesprochen. Wird im SELECT-Befehl in der WHERE-Bedingung ein Filter auf den Partitionsschlüssel angegeben, kann der Oracle Optimizer das Partition Pruning anwenden. Anhand des Filterkriteriums für den Partitionsschlüssel wird der Zugriff auf die relevanten Partitionen der Tabellen und Indizes beschränkt. Nicht relevante Partitionen werden erst gar nicht gelesen, so dass die Laufzeit deutlich reduziert werden kann.
Im Ausführungsplan ist die Anwendung des Partition Pruning anhand der Werte für Pstart und Pstop zu erkennen. Diese Werte geben die Partitionsnummern an, die gelesen werden. Die folgenden Beispiele verdeutlichen das. In den Beispielen wird aus einer Faktentabelle (F_VERKAUF) der Umsatz für einen bestimmten Zeitraum ermittelt. Die Tabelle ist RANGE-partitioniert. Als Partitionsschlüssel wird die Spalte DATUM verwendet.

Im ersten Beispiel wird nur der Umsatz eines einzelnen Tages ausgewertet – entsprechend wird auch nur eine einzelne Partition (PARTITION RANGE SINGLE) gelesen. Es handelt sich dabei um die Partition mit der Nummer 262, wie an den Werten für Pstart und Pstop zu erkennen ist. *(siehe Listing 1)*

Im zweiten Beispiel wird nur der Umsatz eines Zeitraums über mehrere Tage ausgewertet. Folglich werden auch mehrere aufeinander folgende Partition (PARTITION RANGE ITERATOR) gelesen. Es handelt sich dabei um die Partitionen mit der Nummer 262 bis 265. *(siehe Listing 2)*

Wird der Umsatz für mehrere, nicht aufeinander folgende Tage ausgewertet, müssen einzelne ebenfalls nicht aufeinander folgende Partitionen (PARTITION RANGE INLIST) gelesen werden. Im Ausführungsplan werden dann für Pstart und Pstop nicht die einzelnen Partitionsnummern, sondern der Wert KEY(I) angezeigt. *(siehe Listing 3)*

Wird über den Partitionsschlüssel gefiltert, kann Partition Pruning nur dann angewendet werden, wenn im Filter auf den Partitionsschlüssel keine Funktionen angewendet werden. Die folgende Abfrage liefert das gleiche Ergebnis wie die erste Abfrage, jedoch kann aufgrund der verwendeten Funktion TO_CHAR kein Partition Pruning angewendet werden. Somit werde alle Partitionen (PARTITION RANGE ALL) gelesen. Die Anwendung solcher Funktionen ist bei Filtern auf dem Partitionsschlüssel möglichst zu vermeiden. *(siehe Listing 4)*

Die komplette Serie zum Thema „Performance-Tuning“:

Performance-Gewinn durch Techniken der Oracle Datenbank (Teil 1)

Performance-Gewinn durch Techniken der Oracle Datenbank (Teil 2)

Performance-Gewinn durch Techniken der Oracle Datenbank (Teil 3)

Performance-Gewinn durch Techniken der Oracle Datenbank (Teil 3)

Listing 1:

```
select datum, sum (umsatz)
  from f_verkauf
 where datum = to_date ('19.09.2014', 'dd.mm.yyyy')
 group by datum;

-------------------------------------------------------------- ... -----------------
| Id | Operation               | Name      | Rows | Bytes | ... | Pstart| Pstop |
-------------------------------------------------------------- ... -----------------
|  0 | SELECT STATEMENT        |           |  755K|    15M| ... |       |       |
|  1 |  SORT GROUP BY NOSORT   |           |  755K|    15M| ... |       |       |
|  2 |   PARTITION RANGE SINGLE|           |  755K|    15M| ... |   262 |   262 |
|* 3 |    TABLE ACCESS FULL    | F_VERKAUF |  755K|    15M| ... |   262 |   262 |
-------------------------------------------------------------- ... -----------------

Predicate Information (identified by operation id):
---------------------------------------------------
  3 - filter("DATUM"=TO_DATE(' 2014-09-19 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
```

Listing 2:

```
select datum, sum (umsatz)
  from f_verkauf
 where datum between to_date ('19.09.2014', 'dd.mm.yyyy')
                 and to_date ('22.09.2014', 'dd.mm.yyyy')
 group by datum;

-------------------------------------------------------------- ... -----------------
| Id | Operation               | Name      | Rows | Bytes | ... | Pstart| Pstop |
-------------------------------------------------------------- ... -----------------
|  0 | SELECT STATEMENT        |           | 2196K|    46M| ... |       |       |
|  1 |  PARTITION RANGE ITERATOR|           | 2196K|    46M| ... |   262 |   265 |
|  2 |   SORT GROUP BY         |           | 2196K|    46M| ... |       |       |
|* 3 |    TABLE ACCESS FULL    | F_VERKAUF | 2196K|    46M| ... |   262 |   265 |
-------------------------------------------------------------- ... -----------------

Predicate Information (identified by operation id):
---------------------------------------------------
  3 - filter("DATUM"<=TO_DATE(' 2014-09-22 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
```

Performance-Gewinn durch Techniken der Oracle Datenbank (Teil 3)

Listing 3:

```
select datum, sum (umsatz)
  from f_verkauf
 where datum = to_date ('19.09.2014', 'dd.mm.yyyy')
    or datum = to_date ('22.09.2014', 'dd.mm.yyyy')
 group by datum;

------------------------------------------------------------ ... -----------------
| Id | Operation              | Name      | Rows | Bytes | ... | Pstart| Pstop |
------------------------------------------------------------ ... -----------------
|  0 | SELECT STATEMENT       |           | 1262K|    26M| ... |       |       |
|  1 |  PARTITION RANGE INLIST|           | 1262K|    26M| ... |KEY(I) |KEY(I) |
|  2 |   SORT GROUP BY        |           | 1262K|    26M| ... |       |       |
|* 3 |    TABLE ACCESS FULL   | F_VERKAUF | 1262K|    26M| ... |KEY(I) |KEY(I) |
------------------------------------------------------------ ... -----------------

Predicate Information (identified by operation id):
---------------------------------------------------
  3 - filter("DATUM"=TO_DATE(' 2014-09-19 00:00:00', 'syyyy-mm-dd hh24:mi:ss') OR
             "DATUM"=TO_DATE(' 2014-09-22 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
```

Listing 4:

```
select datum, sum (umsatz)
  from f_verkauf
 where to_char (datum, 'dd.mm.yyyy') = '19.09.2014'
 group by datum;

------------------------------------------------------------ ... -----------------
| Id | Operation           | Name      | Rows | Bytes | ... | Pstart| Pstop |
------------------------------------------------------------ ... -----------------
|  0 | SELECT STATEMENT    |           |  623K|    13M| ... |       |       |
|  1 |  PARTITION RANGE ALL|           |  623K|    13M| ... |     1 |   503 |
|  2 |   SORT GROUP BY     |           |  623K|    13M| ... |       |       |
|* 3 |    TABLE ACCESS FULL| F_VERKAUF |  623K|    13M| ... |     1 |   503 |
------------------------------------------------------------ ... -----------------

Predicate Information (identified by operation id):
---------------------------------------------------
  3 - filter(TO_CHAR(INTERNAL_FUNCTION("DATUM"),'dd.mm.yyyy')='19.09.2014')
```