Partitioning pada Oracle 11g

Yuafanda Kholfi Hartono
yuafanda@yahoo.com
http://allofmyjourney.blogspot.com

Partitioning adalah salah satu syarat kunci dari *performance* dan *availability* database yang tinggi, tabel partisi dan indeks split yang lebih kecil, serta komponen-komponen Oracle yang lebih mudah ditangani. Oracle Database 11g menawarkan pilihan terluas metode partisi, termasuk didalamnya adalah metode *interval*, *reference*, *list*, and *range*. Selain itu, Oracle juga menyediakan gabungan partisi dari dua metode, misalnya pada kolom Tanggal Pemesanan (*range*) dan wilayah (*list*) atau wilayah (*list*) dan jenis pelanggan (*list*). *Oracle Partitioning* adalah sebuah opsi pada Oracle Database 11g Enterprise Edition, yang juga merupakan dasar siklus Informasi manajemen strategi Oracle, yang sejalan dengan nilai bisnis informasi untuk meningkatkan efektifitas-biaya proses penyimpanan untuk data warehouse yang besar dan pemrosesan transaksi aplikasi yang tinggi.

Pendahuluan

Partitioning adalah sebuah opsi yang tersedia bagi Database Administrator (DBA) untuk membantu mereka mengefisiensikan area-area kinerja utama database termasuk menyeimbangkan proses I/O, pengurangan *contention*, perbaikan kinerja SQL *statement* dan bahkan availabilitas data. Oracle Database 11g menambahkan beberapa metode baru partisi tabel yang sangat bermanfaat, yang dapat diterapkan untuk membantu DBA untuk mencapai tujuan masing-masing.

Isi

Mengapa Menggunakan Partisi Tabel?

Ada beberapa area-area yang penting dimana DBA cenderung untuk fokus pada pencarian peluang untuk meningkatkan kinerja dalam database mereka. Area-area tersebut antara lain penggunaan memori, kinerja SQL *statement*, persyaratan ruang dan media penyimpanan (*storage*), kemacetan jaringan *(network)*, dan kinerja I/O.

Dianggap sebagai bagian dari fitur pendukung database yang sangat besar, pengertian partisi tabel sendiri adalah opsi yang tersedia untuk Administrator yang dapat membantu setidaknya dalam dua area kinerja utama: kinerja I/O dan SQL *statement*. Dengan mengambil *key* tabel (dan indeks) dengan kapasitas data besar dan tingginya tingkat transaksi dan mempartisi mereka sehingga tersimpan di tablespace yang berbeda, sehingga seorang DBA akan melihat

peningkatan kinerja dalam beberapa area, antara lain :

1. Mengurangi *contention* untuk blok dari tabel atau indeks. Hal ini merupakan hasil dari kepemilikan blok yang didistribusikan dalam tablespace berbeda berdasarkan pilihan partisi yang digunakan.
2. Mengurangi I/O *contention* karena datafiles untuk tablespace yang digunakan oleh tabel partisi secara fisik dapat ditempatkan pada perangkat atau drive yang berbeda.
3. Peningkatan kinerja SQL untuk statement yang tereferensi dengan kolom kunci yang terpartisi pada *where clause* karena optimizer dapat menggunakan partisi untuk 'pemangkasan' (hanya membaca dari partisi tertentu yang cocok dengan data yang diminta).
4. Manfaat lain yang mungkin terjadi adalah ketersediaan data dalam tabel. Tidak hanya dapat menyelesaikan *administrative task* yang dilakukan sementara hanya dengan mengambil subset dari partisi offline, jika sesuatu terjadi pada satu datafile, partisi yang tersisa di tablespace lain masih dapat diakses.

Manfaat

- *Faster Performance* - Menurunkan waktu query dari menit ke detik
- *Increases Availability* - 24 jam dalam 7 hari untuk mengakses informasi penting
- *Improves Manageability* - Mengelola '*chunks*' data yang lebih kecil
- *Enables Information Lifecycle Management* – Penggunaan biaya yang efisien dalam proses penyimpanan

Pada 11g, Oracle telah menambahkan beberapa metode tabel partisi baru yang sangat kuat yang dapat diterapkan untuk mencapai manfaat ini untuk database yang sangat besar (bahkan kecil). Namun, sebelum membahas metode tabel partisi baru, berikut tinjauan singkat sejarah partisi Oracle.

Oracle Database Version	***Partitioning Features***
8.0.5	*Range Partitioning* diperkenalkan
8i	*Hash* dan *composite Range-Hash partitioning* diperkenalkan.
9i	*List Partitioning*, *Composite Range-List partitioning* diperkenalkan.
10g	Memperkenalkan *Range*, *List* and *Hash* partitioning tabel yang telah terindex. Juga memperkenalkan *composite* opsi *partitioning* lainnya.
11g	Memperkenalkan partition extensions: *-Interval partitioning* *-REF partitioning* *-Virtual Column-based partitioning* *- Partition Advisor*.

Metode Partisi Tabel yang Telah Dirilis Sebelumnya

Range Partitioning

Oracle memperkenalkan *range partition* di Oracle Database 8,0 sebagai metode partisi pertama. Ini adalah kemampuan untuk partisi data berdasarkan rentang yang terkait dengan kolom dalam tabel. Hal ini paling sering digunakan untuk membagi data baris di tablespace yang berbeda

berdasarkan nilai tanggal. Sebagai contoh, sebuah tabel yang berisi data penjualan bisa dibagi dengan kuartal, bulan, tahun - atau beberapa kriteria tanggal lain berarti

Hash Partitioning
Metode ini diperkenalkan pada Oracle Database 8.1. Dengan *hash partition*, sebuah tabel bisa dibagi menjadi beberapa partisi berdasarkan *key* yang tidak benar-benar meminjamkan dirinya ke berbagai divisi, seperti misalnya tabel *Customer* yang memiliki data besar. Kita bisa membuat beberapa partisi berdasarkan id pelanggan, dan Oracle akan mendistribusikan *row* data di tablespace berdasarkan hasil kriteria kolom partisi melalui algoritma *hashing* dan menggunakan hasilnya untuk menentukan tempat penyimpanan *row* data tersebut.

Range-Hash Partitioning
Juga ditambahkan dalam 8i, adalah kemampuan untuk melakukan partisi komposit. Hal ini memungkinkan untuk melakukan partisi data pertama berdasarkan *range*, dan kemudian dalam *range* tersebut, dibuat partisi lebih jauh dengan *hash partition*.

List Partitioning
*List Partition d*iperkenalkan pada 9i (R1), mengisi gap yang hilang antara *range* dengan *hash partition*. *List* memungkinkan tabel partisi yang akan dipartisi berdasarkan nilai-nilai yang berbeda seperti kode negara (atau kode provinsi). Kode negara atau *key* lain yang memiliki nilai rahasia yang berbeda, tapi tidak meminjamkan dirinya sendiri untuk *range partition*.

Range-List Partitioning
Sebuah metode komposit partisi baru yang diperkenalkan di 9i (R2) yang aktif tabel yang akan dibagi dengan *range*, dan kemudian disub-partisi dengan daftar nilai diskritnya.

Metode Baru Partisi Tabel Diperkenalkan pada Oracle Database 11g
Interval Partitioning
Pada dasarnya, *Interval Partition* adalah perangkat tambahan untuk *range partition*. Salah satu tantangan dalam berbagai partisi sebelum 11g adalah bahwa tidak ada cara otomatis untuk membuat partisi baru ketika data terkini ditambahkan ke dalam database. Misalnya, jika sebuah tabel sedang dipartisi per bulan, pada awal bulan baru DBA harus secara manual membuat partisi baru dalam data terpisah untuk bulan terakhir.
Dengan *Interval Partition*, Oracle secara otomatis akan menghasilkan partisi baru untuk menampung data baru yang ditambahkan ke dalam tabel. Ketika mendefinisikan tabel dipartisi, opsi baru "interval" telah ditambahkan ke perintah DDL. Selain itu, penggunaan "values less than maxvalue" telah dihapus. Interval partisi hanya bisa dilakukan pada kolom tanggal atau nomor.
Berikut adalah contoh sintaks untuk membuat tabel yang sekaligus melakukan *Interval partition* pada kolom order_date berdasarkan bulan.

```
CREATE TABLE orders_tbl

(order_id number(10),

 order_date date,

 order_mode varchar2(10),

 order_total number(15,2)

 customer_id number(10))

PARTITION BY RANGE (order_date)
```

```
INTERVAL(NUMTOYMINTERVAL(1,'MONTH'))

STORE IN (data01,data02,data03,data04)

(PARTITION JUN10 values less than TO_DATE('01-07-10','dd-mm-yy')),

 PARTITION JUL10 values less than TO_DATE('01-08-10','dd-mm-yy')),

 PARTITION AUG10 values less than TO_DATE('01-09-10','dd-mm-yy')));
```

Setiap data ditambahkan ke dalam kisaran yang telah ditentukan akan terinsert seperti biasa. Dan ketika data yang melampaui akhir Agustus diinsert, sebuah partisi baru akan ditambahkan secara otomatis dengan nama yang dihasilkan oleh system. Anda dapat melakukan query pada DBA_TAB_PARTITIONS untuk melihat nama partisi yang dihasilkan oleh system.
Selain itu, jika Anda memiliki tabel partisi yang sudah ada sebelumnya, Anda dapat dengan cepat melakukan konversi menjadi tabel *Interval partition* menggunakan

```
ALTER TABLE nama_tabel SET INTERVAL (nilai interval);
```

System Partitioning
Jika penempatan data secara khusus dikelola oleh aplikasi, dan dukungan yang dibutuhkan dalam database adalah tabel yang perlu dipecah menjadi partisi yang lebih kecil, partisi sistem mungkin adalah jawabannya.
Dengan partisi sistem, tidak ada tombol partisi, dan beberapa manfaat seperti pemangkasan partisi tidak mungkin dapat direalisasi. Selain itu, beberapa operasi pemeliharaan partisi seperti split partisi tidak akan didukung.

```
CREATE TABLE mypart_tbl
(id number,
desc varchar2(50))
PARTITION BY SYSTEM
(partition part_1 tablespace data01,
partition part_2 tablespace data02,
partition part_3 tablespace data03,
partition part_4 tablespace data04);
```

Statement *insert* dan *merge* ke dalam tabel ini akan membutuhkan sintaks menggunakan partisi-*extended*. Sedangkan statement *delete* dan *update*, tidak membutuhkan sintaks *extended*.

```
INSERT INTO mypart_tab PARTITION part_1 VALUES (1,'Some Text');
```

Virtual Column Partitioning
Virtual kolom yang dikalkulasikan atau yang diturunkan dari data lainnya sekarang dapat digunakan sebagai dasar untuk partisi tabel. Hal ini memungkinkan tabel untuk dipartisi berdasarkan informasi bisnis yang tidak perlu disimpan dalam kolom individu.
Sebagai contoh, katakanlah sebuah bagian table memiliki kolom yang disebut part_id dimana 3, 4, dan 5 karakter merupakan kode produsen dan manajemen sering memerlukan laporan dan informasi untuk produsen tertentu. Hal ini mungkin akan bermanfaat untuk dapat mempartisi tabel berdasarkan subset dari karakter tersebut.

```
CREATE TABLE parts_tbl
(part_id char(10),
 desc varchar2(500),
 cost number(10,2)
 mfr_code as upper((substr(part_id,3,3)))
 PARTITION BY LIST(mfr_code)
 (PARTITION ABC_DEF VALUES ('ABC','DEF'),
 PARTITION GHI_JKL VALUES ('GHI','JKL'),
 PARTITION MNO_PQR VALUES ('MNO','PQR'),
 PARTITION STU_VWX VALUES ('STU','VWR'));
```

Reference Partitioning

Metode partisi terakhir dari metode tabel partisi baru yang diperkenalkan di 11g adalah *Reference Partition*. Partisi ini adalah pilihan yang berguna untuk berurusan dengan dua tabel dalam hubungan *one-to-many* seperti misalnya pada tabel *order* dan *order item*.

Jika tabel pesanan dipartisi dengan *range* atau *interval* berdasarkan urutan tanggal, mungkin masuk akal juga partisi urutan item tabel dilakukan dengan cara yang sama. Di masa lalu, ini akan diperlukan bahwa kita mende-normalkan tabel urutan item dengan menambahkan tanggal sehingga kita bisa mengatur partisi tabel untuk mencocokkan tabel induknya. Salah satu manfaat kunci untuk *Reference Partition* adalah untuk tidak memiliki ruang tambahan yang digunakan, dan tidak harus berurusan dengan isu-isu integritas data potensial.

Untuk menggunakan *Reference Partition*, sebuah *foreign key* formal pada *child* tabel harus didefinisikan dalam database dan tidak akan mungkin untuk menonaktifkan *foreign key constraint*.

```
CREATE TABLE orders_tbl
(order_id number(10),
 order_date date,
 order_mode varchar2(10),
 order_total number(15,2),
 customer_id number(10))
PARTITION BY RANGE (order_date)
INTERVAL(NUMTOYMINTERVAL(1,'MONTH'))
STORE IN (data01,data02,data03,data04)
(PARTITION JUN10 values less than TO_DATE('01-07-10','dd-mm-yy')),
```

```
 PARTITION JUL10 values less than TO_DATE('01-08-10','dd-mm-yy')),

 PARTITION AUG10 values less than TO_DATE('01-09-10','dd-mm-yy')));

CREATE TABLE order_items_tbl

 (order_id NUMBER(10),

 line_id NUMBER(3),

 product_id NUMBER(10),

 price NUMBER(10,2),

 quantity NUMBER(5),

 CONSTRAINT order_items_fk FOREIGN KEY (order_id)

 REFERENCES orders_tbl)

PARTITION BY REFERENCE (order_items_fk);
```

Partition Advisor
SQL Access Advisor yang diperkenalkan pada Oracle 10g, telah di update untuk menyertakan advice pada table partisi yang telah dibuat, materialized view dan index. Fitur ini membantu melakukan *generate* rekomendasi dalam proses partisi, serta menampilkan peningkatan efisiensi performance apabila partisi tersebut diimplementasikan. Fitur ini juga dapat melakukan generate script pembuatan partisi yang efisien, yang dapat disubmit secara manual melalui SQLplus ataupun melalui Enterprise Manager.

Penutup

Partitioning adalah salah satu opsi yang paling dicari setelah opsi untuk data warehousing. Hampir semua data warehouse Oracle menggunakan partisi untuk meningkatkan kinerja query dan juga untuk mengurangi kompleksitas pemeliharaan sehari-hari. Dimulai dengan 11g, pilihan partisi lebih telah disediakan dan hal ini harus mengurangi sebagian besar beban DBA.
Artikel ini bertujuan membantu DBA dan Developer aplikasi yang bekerja terutama di lingkungan data warehouse. Fitur-fitur baru yang disediakan oleh Oracle di 11g harus meningkatkan pilihan partisi dan menyediakan fleksibilitas untuk penggunaan partisi dan pemeliharaan.
Tabel partisi pertama kali diperkenalkan dalam versi 8.0 dan Oracle adalah vendor RDBMS pertama yang mendukung partisi fisik. SQL Server (2000) dan DB2 menyediakan partisi logis (menggunakan pandangan UNION ALL) sedangkan SQL Server 2005 mendukung partisi fisik meskipun tidak secara langsung (dilaksanakan melalui fungsi partisi). Pilihan partisi di Oracle itu sangat diterima oleh komunitas pengguna karena meningkatkan kinerja, pengelolaan dan ketersediaan aplikasi dan yang paling penting, aplikasi DSS. Apresiasi luas fitur ini telah menghasilkan peningkatan yang sering melalui rilis berikutnya.

Referensi

Karen Reliford, “Oracle Database 11g New Features for Table Partitioning”, 2010
http://www.databasejournal.com/features/oracle/article.php/3900241/Oracle-Database-11g-New-Features-for-Table-Partitioning.htm

Oracle-base.com's Team, "Partitioning Enhancements in Oracle Database 11g Release 1",
http://www.oracle-base.com/articles/11g/PartitioningEnhancements_11gR1.php

Oracle Partitioning, 2010
http://www.oracle.com/us/products/database/options/partitioning/index.html

Venurani, "Partitioning in Oracle 11g", 2007
http://www.orafaq.com/node/1912

Biografi Penulis

Yuafanda Kholfi Hartono. Menyelesaikan S1 di Universitas Indonesia. Sebelumnya menamatkan pendidikan di Sekolah Tinggi Akuntansi Negara, minat dan ketertarikan pada bidang IT membuat pada akhirnya ditempatkan pada Direktorat Jenderal Bea dan Cukai Kementerian Keuangan sebagai Database Administrator (DBA).