Partitioned Tables in Oracle

Overview

Oracle supports "partitioning" of tables. Each partition is an independent segment in the data dictionary, but DML statements don't need to know about the structure of partitions. Individual partitions can be dropped without affecting the rest of the table. Indexes can also be partitioned by declaring them "local".

Of the partitioning schemes available, the one most interesting for tables in the FlightOps schema is "range" partitioning. In this scheme, the partitions are bounded by ranges of values of a column in the table. Both numeric and date/time ranges are supported.

Partitioning existing tables

A partition's data segment can be "exchanged" with a non-partitioned table with the same structure. This operation is a direct manipulation of the data dictionary, and therefore is unaffected by the amount of data in the table or partition. This feature allows us to convert an existing "monolithic" table to a partitioned table as follows:

- Create a new, partitioned table with the same columns as the existing table and one partition.
- Create corresponding partition-local indices on the new table.
- Exchange the data segments of the existing table and the new partition.
- Rename the existing table and global index out of the way
- Rename the partitioned table and local index into place.
- (Optional) Drop the old table/index.

I've used the following script to partition the existing V3HKVALCHFIELDS table in ISOC_NIGHTLY:

```
-- create a new parallel table with one partition
create table v3hkvalchfields_partition(
        valtlmsrc_fk    integer,
        usecs           timestamp,
        value           number,
        validity        smallint
)
partition by range(usecs)(
partition M200805 values less than (maxvalue)
);

-- create a new parallel index that's partition-local
create index v3hkvalch_idx1_local on v3hkvalchfields_partition(valtlmsrc_fk, usecs) local;

-- now exchange data segments between the single partition
-- of the new table and the whole of the existing table
alter table v3hkvalchfields_partition exchange partition M200805
with table v3hkvalchfields without validation update global indexes;

-- we've swapped out the data, so rename the old table/index out of the way
rename v3hkvalchfields to v3hkvalchfields_nonpart;
alter index v3hkvalch_idx1 rename to v3hkvalch_idx1_global;

-- now rename the new table/index
rename v3hkvalchfields_partition to v3hkvalchfields;
alter index v3hkvalch_idx1_local rename to v3hkvalch_idx1;
```

To run this in ISOC_FLIGHT, I'd probably throw in a few "lock table ... in exclusive mode;" statements to avoid tripping over cases when the automated processing is loading data.

Managing partitions

In Oracle 10g and earlier, there is no way to have the database automatically create new range partitions as needed. Oracle 11g has an extension to range partitioning called "interval" partitions that automates the creation of new partitions as data arrives.

But all is not lost in our 10g world; some kind souls have developed a script called "pman.sh" that does the heavy lifting. The script relies on a naming convention for partitions that encodes the partitioning interval and column type. In this convention, the "M200805" I've used above means that the partitions should be monthly and the range columnt is a date/timestamp type. See the script documentation for other options. I did have to modify the script a bit for local conditions; it originally assumed that it could "connect / as sysdba", and tried to discover the version of Oracle. My local mods allow it to run in a "walleted" environment with a non-dba account, and force the version (and thus the generated SQL syntax) to Oracle v10 conventions.

The script can both create new future partitions and drop old partitions. After transforming the original monolithic V3HKVALCHFIELDS table, running the script as `pman.sh alias=isocnightly table=isoc_nightly.v3hkvalchfields next=6 show=n` generated and executed the following DML:

```
ALTER TABLE ISOC_NIGHTLY.V3HKVALCHFIELDS
SPLIT PARTITION M200805
AT (to_date(20080601,   'YYYYMMDD'))
INTO (PARTITION M200805, PARTITION M200806)
UPDATE GLOBAL INDEXES
;

ALTER TABLE ISOC_NIGHTLY.V3HKVALCHFIELDS MODIFY PARTITION M200805
REBUILD UNUSABLE LOCAL INDEXES;

BEGIN
  DBMS_STATS.GATHER_TABLE_STATS (
    OWNNAME          => 'ISOC_NIGHTLY',
    TABNAME          => 'V3HKVALCHFIELDS',
    PARTNAME         => 'M200805',
    GRANULARITY      => 'PARTITION',
    CASCADE          => TRUE,
    ESTIMATE_PERCENT => 20
  );
END;
/

ALTER TABLE ISOC_NIGHTLY.V3HKVALCHFIELDS
SPLIT PARTITION M200806
AT (to_date(20080701,   'YYYYMMDD'))
INTO (PARTITION M200806, PARTITION M200807)
UPDATE GLOBAL INDEXES
;

... and so forth for M200808, M200809, etc.
```

Since there was already data in the original M200805 partition, the `ALTER TABLE ... SPLIT PARTITION...` and `ALTER TABLE... REBUILD...` statements take some time to execute. I've sent all this DML off to Ian so he can assess the impact on the database before I try it "in anger".

After experimenting a bit with pman.sh, the structure of ISOC_NIGHTLY.V3HKVALCHFIELDS is:

```
SQL> select table_name, partition_name, high_value, num_rows from user_tab_partitions order by partition_name;

TABLE_NAME                     PARTITION_NAME
HIGH_VALUE                                                                       NUM_ROWS
------------------------------ ------------------------------
-------------------------------------------------------------------------------- ----------
V3HKVALCHFIELDS                M200712                        TIMESTAMP'2008-01-01 00:00:
00'                                                          420725
V3HKVALCHFIELDS                M200801                        TIMESTAMP'2008-02-01 00:00:
00'                                                           0
V3HKVALCHFIELDS                M200802                        TIMESTAMP'2008-03-01 00:00:
00'                                                           0
V3HKVALCHFIELDS                M200803                        TIMESTAMP'2008-04-01 00:00:
00'                                                           0
V3HKVALCHFIELDS                M200804                        TIMESTAMP'2008-05-01 00:00:
00'                                                         2081325
V3HKVALCHFIELDS                M200805                        TIMESTAMP'2008-06-01 00:00:
00'                                                           0
V3HKVALCHFIELDS                M200806                        TIMESTAMP'2008-07-01 00:00:
00'                                                         1604110
V3HKVALCHFIELDS                M200807                        TIMESTAMP'2008-08-01 00:00:
00'                                                         4778975
V3HKVALCHFIELDS                M200808                        TIMESTAMP'2008-09-01 00:00:
00'                                                         1801765
V3HKVALCHFIELDS                M200809                        TIMESTAMP'2008-10-01 00:00:
00'                                                           0
V3HKVALCHFIELDS                M200810                        TIMESTAMP'2008-11-01 00:00:
00'                                                          809642
V3HKVALCHFIELDS                M200811                        TIMESTAMP'2008-12-01 00:00:
00'                                                           0
V3HKVALCHFIELDS                M200812
MAXVALUE                                                                            0

13 rows selected.
```

Once we get the initial partitioning done (provide Ian approves), then we'll set up a monthly cron job to run pman.sh with the `next=3` and `keep=6` options to create future partitions and drop those older than six months.

If we ever to to 11g, we can take advantage of the new-partition-generation feature on the existing table with an `ALTER TABLE V3HKVALCHFIELDS SET INTERVAL NUMTOYMINTERVAL(1, 'month');`.