

**NAME**

**PMAN** – partition manager for Oracle range partitions on date column.

**SYNOPSIS****pman**

```
sid=sid_name
table=owner_name.table_name
[next=<number>]
[keep=<number>[T]]
[merge=y|n]
[archive=<number>]
[functional_bitmap_index=y|n]
[show=y]
[help=y]
[debug=y]
[today=YYYYMMDD]
```

**DESCRIPTION**

Oracle table partitioning divides a usually a large table into smaller more manageable units, the partitions. It is a simple “divide and conquer” concept, but offers tremendous benefits.

Partitioning greatly increased the database management capability. Each partition can be individually managed, it can be dropped, truncated, moved, split, merged, and exchanged with a table.

While each partition can be individual queried, you do not have to change your SQL to take advantage of the partitioning. The “partition pruning” is done transparently via Oracle optimizer. The performance increase is similar to that you can find your December bill easily in the folder of that month, instead of going through a pile of paper of the past 5 years’ bills mixed together.

Partitioning, together with locally managed, uniform sized extents, permanently solved the problem of data fragmentation due to the data constantly move in and out of the table. The partitions dropped or truncated release the entire space occupied to the residing tablespace. Those who have spent endless hours during weekends performing object reorganization would appreciate the beauty of the this feature. (However the consultants who bill those hours may not).

Simply speaking, there are 4 types of partitions:

- range partition for continuous values like date and number.
- list partition for discrete values like US fifty states.
- hash partition for random values like license plates.
- composite partition is partition within partition.

This article presents PMAN utility, which handles the range partition based on date column, which is the most common type of partitioning.

The date can be either presented by Oracle internal date type or timestamp type, or by number type, for example, 20070130 for January 30, 2007, or character type such as '20070130'.

**WHAT PMAN DOES**

PMAN manages the partitions through out the life cycle.

PMAN helps you to create partitions when you want to convert a non-partitioned table with many years data to partitioned table.

PMAN helps you to manage the partitions once you have a partitioned table. It creates new partitions, drops old partitions per your specification. It updates statistics, and validate indexes.

PMAN can also help you to merge daily partitions to monthly partitions, and monthly partitions to yearly partitions if you like, and drop the merged (archived) partitions when you do not need them.

**HOW PMAN DOES IT**

PMAN needs to know how the table it operates on is partitioned. It could query the dictionary and gets an idea. But it is easier and more reliable just to look at the label on the partition.

Each partition needs to have a name to identify the partition. Oracle does not care how you name it as long as you name it. The users either don't know what a table partition is, or is agreeable with whatever the name you use as long as her queries continue to run.

So here is PMAN's naming standard:

```

partition_name  partitioned by
=====
Y2004          YEAR   with DATE/TIMESTAMP type
M200407        MONTH  with DATE/TIMESTAMP type
D20040704      DAY    with DATE/TIMESTAMP type
W20040705      WEEK   with DATE/TIMESTAMP type
P2004          YEAR   with NUMBER/CHAR type with date represented as YYYY
P200400        YEAR   with NUMBER/CHAR type with date represented as YYYYMM
P20040000      YEAR   with NUMBER/CHAR type with date represented as YYYYMMDD
P200407        MONTH  with NUMBER/CHAR type with date represented as YYYYMM
P20040700      MONTH  with NUMBER/CHAR type with date represented as YYYYMMDD
P20040704      DAY    with NUMBER/CHAR type with date represented as YYYYMMDD
PW20040705     WEEK   with NUMBER/CHAR type with date represented as YYYYMMDD
=====

```

Standard is good as long as it is easy to follow, and it does not limit the application in any way. With this naming standard, it is quite easy to find how the table is partitioned, and how to add new partitions, drop or merge old partitions, etc.

**PMAN FEATURES**

We would like to use an example go through the life cycle of partition management, and demonstrate the PMAN usage, features, and design decisions along the way.

*01. CREATE INITIAL PARTITIONS*

To start a new partitioned table, just create the table with the earliest partition manually, like this:

```

SQL> CREATE TABLE trade.settlement (
  2   settle_id  NUMBER,
  3   settle_date DATE
  4 )
  5 PARTITION BY RANGE (settle_date)
  6 (PARTITION M200509 VALUES LESS THAN (MAXVALUE));

```

Table created.

```

SQL> ALTER TABLE TRADE.SETTLEMENT
  2> ADD CONSTRAINT pk_settlement PRIMARY KEY (settle_id);

```

Table altered.

And let the PMAN takes care of the rest.

Please note:

- We create the last partition to be able to hold any future rows without upper bound. This is conscious decision for added reliability. In case that you forget to create future partitions, all rows will go to the last partition. This is not a ideal situation, but may be better than failure in most cases.

Then we can use PMAN to create the rest of the partitions. For example, the following command will create two more partitions.

```
$ pman sid:=$ORACLE_SID table=trade.settlement next=2 show=n
```

The result is:

```
SQL> ALTER TABLE TRADE.SETTLEMENT
 2 SPLIT PARTITION M200509
 3 AT (to_date(20051001, 'YYYYMMDD'))
 4 INTO (PARTITION M200509, PARTITION M200510)
 5 UPDATE GLOBAL INDEXES
 6 ;
```

Table altered.

```
SQL> ALTER TABLE TRADE.SETTLEMENT
 2 SPLIT PARTITION M200510
 3 AT (to_date(20051101, 'YYYYMMDD'))
 4 INTO (PARTITION M200510, PARTITION M200511)
 5 UPDATE GLOBAL INDEXES
 6 ;
```

Table altered.

Please note:

- RMAN uses “SPLIT PARTITION” statement to split two partitions as specified.
- The partition column is only needed one time when you create the partitioned table. It is not needed for subsequent management. In other words, the partition column name is not programming parameter and need not to be maintained outside the database.
- show=y option will produce the SQL statements without any changes on the database. This is useful if you want to test PMAN, or use PMAN as a SQL generator. It is tedious and error prone to manually create the SQL code to create many partitions, say, monthly partitions for past 7 years. You can leave the job to PMAN.
- The next=<n> options create next <n> partitions relative to CURRENT partitions, which is the partition a row with current date would go.

We start with one partition M200509 which is the current partition for any dates. “next=2” option creates 2 more partitions on top of M200509 partition, ending up with 3 partitions M200509, M200510, M200511.

## 02. MAINTAIN FUTURE PARTITIONS

Now suppose today is 10/15/2005, the current partition is M200510. If you want to precreate 3 future partitions, you should use option “next=3”. The PMAN command is:

```
$ pman sid:=$ORACLE_SID table=trade.settlement next=3 show=n today=20051015
```

Since there is already one future partitions M200511, PMAN will create just two more partitions. In other words, PMAN will create none, or as many as partitions necessary to maintain the number of future partitions as required. However, PMAN will not drop future partitions if the future partitions are more than what you need.

Suppose today is indeed October 15, 2005, or any day in October 2005 for this example, you do not need to use <today=YYYYMMDD> option. This option is needed only when you pretend today is a different date.

## 03. MAINTAIN PAST PARTITIONS

While next=<n> option is the number of partitions you would like to maintain after the current partition, the keep=<n> is the number of partitions you would like to maintain before the current partition.

The PMAN command to drop all old partitions except the most recent 3 partitions are:

```
$ pman sid:=$ORACLE_SID table=trade.settlement keep=3 show=n today=20060115
```

The result is:

```

SQL> SELECT 'partition='||partition_name
  2 FROM dba_tab_partitions
  3 WHERE table_owner='TRADE'
  4 AND table_name='SETTLEMENT'
  5 ORDER BY 1;

'PARTITION='||PARTITION_NAME
-----
partition=M200509
partition=M200510
partition=M200511
partition=M200512
partition=M200601

SQL> SELECT 'version=' || VERSION from v$instance;

'VERSION='||VERSION
-----
version=10.1.0.5.0

SQL> ALTER TABLE TRADE.SETTLEMENT
  2 DROP          PARTITION M200509
  3 UPDATE GLOBAL INDEXES
  4 ;

```

Table altered.

– We see first PMAN lists all the partitions in the time order. Since today is January 15, 2006, M200601 is the current partition, M200509 through M200512 are 4 past partitions. Since we only to keep 3 month's data, the oldest partition M200509 is dropped.

– We see also that Oracle version is queried. This is because when you drop the partitions with data, most likely there are, the global indexes will become invalid. Oracle version 9i and 10g provide "UPDATE GLOBAL INDEXES" clause to update the indexes simultaneously.

If it is an Oracle database version 8i and below, additional SQL statements like this:

```
ALTER INDEX <owner>.<index> REBUILD COMPUTE STATISTICS;
```

will be executed inside PL/SQL.

– If you specify the option keep=<n>T instead of keep=<n>, PMAN will truncate the unwanted partitions instead of dropping them. This could be useful if the table structure needs to be maintained, or future restore is needed.

– The keep and next options can be used at the same time. This is how jobs are normally setup, to drop one old partition, and precreate a new partition each month for monthly partitioned tables. We separate these two options for illustration purpose.

#### 04. MERGE PAST PARTITIONS

The merge option will merge prior monthly partitions to yearly partitions, and prior daily partitions to monthly partitions. In our example, the following 3 partitions in previous year (relative to the current partition)

```
M200510
M200511
M200512
```

will merge to A2005, where A stands for Archive.

The PMAN command is:

```
$ pman sid:= $\$$ ORACLE_SID table=trade.settlement merge=y show=n today=20060115
```

The result is:

```
SQL> ALTER TABLE TRADE.SETTLEMENT MERGE PARTITIONS
 2 M200510, M200511
 3 INTO PARTITION
 4 M200511a
 5 UPDATE GLOBAL INDEXES
 6 ;
```

Table altered.

```
SQL> ALTER TABLE TRADE.SETTLEMENT MERGE PARTITIONS
 2 M200511a, M200512
 3 INTO PARTITION
 4 M200512a
 5 UPDATE GLOBAL INDEXES
 6 ;
```

Table altered.

```
SQL> ALTER TABLE TRADE.SETTLEMENT RENAME PARTITION M200512a
 2 to A2005;
```

Table altered.

#### 05. DROP MERGED (ARCHIVED) PARTITIONS

The archive option is just like the keep option, but operate on the merged (archived) partitions.

To illustrate this feature, let us fast forward the clock one year: you created all 2006 partitions, and archived them.

The following PMAN commands to simulate this process.

```
$ pman sid:=$ORACLE_SID table=trade.settlement next=12 show=n today=20060115
$ pman sid:=$ORACLE_SID table=trade.settlement merge=y show=n today=20070115
```

The first command will create 12 partitions, bringing the last partition to M200701, and the second command will merge all 2006 partitions.

Now we have two archived partitions A2005 and A2006. Suppose your site policy is to keep the data for 1 year besides the current year, you can use the archive option to drop all archived partitions except the most recent one.

The PMAN command to do this is:

```
$ pman sid:=$ORACLE_SID table=trade.settlement archive=1 show=n today=20070115
```

which simply does:

```
SQL> ALTER TABLE TRADE.SETTLEMENT
 2 DROP PARTITION A2005
 3 UPDATE GLOBAL INDEXES
 4 ;
```

Table altered.

today option would be useful if you need to merge 2006 partitions in the year 2008. You do not need to use the option if you run the command in year 2007.

#### ORACLE CONSIDERATIONS

##### STATISTICS

It was not shown in the example, but PMAN does gather statistics for the partitions it touched, including split partitions, and merged partitions. Example:

```

SQL> BEGIN
2     DBMS_STATS.GATHER_TABLE_STATS (
3         OWNNAME          => 'TRADE' ,
4         TABNAME          => 'SETTLEMENT' ,
5         PARTNAME         => 'M200511' ,
6         GRANULARITY      => 'PARTITION' ,
7         CASCADE          => TRUE ,
8         ESTIMATE_PERCENT => 20
9     );
10 END;
11 /

```

In normal situation, the last partition is future partition and empty. The newly split partitions are therefore empty as well. But zero statistics is meaningful statistics, while no statistics is not.

### *INDEXES*

PMAN generate statements to rebuild local indexes for the partitions it touched:

```

SQL> ALTER TABLE TRADE.SETTLEMENT MODIFY PARTITION M200511
2     REBUILD UNUSABLE LOCAL INDEXES;

```

In normal situation, we split empty partitions which will not make local indexes unusable.

PMAN also generates statements to rebuild global indexes for Oracle 8i and below, and uses “UPDATE GLOBAL INDEXES” for Oracle 9i and above.

### *TABLESPACES*

One of the advantages of partitioning is we can put different partitions on different disks for performance and capacity planning.

PMAN is designed for maintaining a constant of number of partitions occupying constant or slowly growing amount of disk space. It does not use TABLESPACE clause for partitions. As a result, the split table partitions and index partitions remain in the same tablespaces where they split out, respectively.

You can expand its usage by periodically moving the last partitions to new tablespaces; or add TABLESPACE clause to put January partitions on January tablespace, February partitions on January tablespace, etc.

### *FUNCTIONAL BITMAP INDEX*

In normal situation, the last partition should be empty any way. It is more important to keep that way if you are running Oracle prior to Oracle 9.0.1 and have a functional bitmap index.

This is because when we split a non-empty partition, the functional bitmap index (and all other indexes) will become invalid. Rebuilding the functional bitmap index for a partition will cause entire table scanned due to the Oracle bug 1987514. Depending on the size of the table, this could be a problem. So in this case, it is better to drop the index and recreate it.

The bug is about functional local index in general (bitmap index is always local), but we only had experience with functional bitmap index. The bug is supposedly fixed in Oracle 9.0.1.

### **PORTABILITY**

PMAN is written in portable shell, which runs on any platforms that has ksh or bash. It has been tested on various combinations of shell (ksh, bash), Oracle database versions (8i, 9i, and 10g), and operating environments (Solaris, HP-UX, AIX, and Cygwin).

## CAVEATS

Although PMAN naming standard does not pose any limitation of the use of partitions, as said earlier, there would a problem if your organization may use a different standard.

There are two solutions to solve the problem if you want to use pman: to change the partition name to the PMAN standard if the partition name is not an externally exposed interface; or to change the pman to fit your existing standard. If your naming standard does not include the information how the table is partitioned (for example, yearly, monthly, etc), then you have to modify the program to accept the information, creating one more piece of loose part.

It is certainly possible to have a generic program without using the naming convention, just like it is possible, but more difficult, to do premise wiring without color coding standard. However, unless you opt to randomly generate partition names, you have to have a naming convention anyway. It is just a different standard.

The standard difference is not a technical issue. Like culture difference, while it causes confusion, it creates opportunities and makes life more interesting.

## SUMMARY

PMAN can help you to create partitioned tables, or convert regular tables to partitioned tables. It can help you to manage the partitions. It can also be used as a SQL generator, if that is what you want to do.

PMAN is simple, generic, portable, and reliable.

## ACKNOWLEDGMENT

The version 1.0 (2001-12-18) was inspired by our friend David Xiao's PL/SQL code. Throughout the years, enhancements are made and bug are fixed with the help of friends and users. The current version 4.5 (2007-03-17), and future update is available from <<http://www.unixlabplus.com/unix-prog/pman>>.

## REFERENCE

Wang, Michael and Julie Wang. "Date-Related Shell Functions." September 2005. Unix Review. <<http://www.unixreview.com/documents/s=9521/ur0509a/>>, or <[http://www.unixlabplus.com/unix-prog/date\\_function/](http://www.unixlabplus.com/unix-prog/date_function/)>

Oracle Corporation. "Partitioned Tables and Indexes." Oracle Database Concepts. October, 2005. <[http://download-east.oracle.com/docs/cd/B19306\\_01/server.102/b14220.pdf](http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14220.pdf)>.

## AUTHORS

Michael Wang has been a sysadmin, Oracle database admin, and now a Unix programmer. He is an Oracle 10g certified professional. He can be reached at [xw73@columbia.edu](mailto:xw73@columbia.edu).

Julie Wang has managed Unix systems, Lawson Enterprise Systems, and currently Oracle databases. She is an Oracle 10g certified professional. She can be reached at [julie\\_wangye@yahoo.com](mailto:julie_wangye@yahoo.com).

## VERSION HISTORY

### **version 4.6, 2007-03-19.**

- Fixed merge option for P600 case (P20040900).

### **version 4.5, 2007-03-17.**

- Fixed rebuilding global index for 8i with PL/SQL code.

### **version 4.4, 2007-02-15.**

- Added Cygwin support.

**version 4.3, 2007-01-30.**

- Updated generic functions; make the code more portable (ksh88, bash).
- Added today option to pretend we are running on that day.

**version 4.2, 2005-08-11.**

- Added userid option for a major telecom company because the DBE's can not connect as sysdba, and oratab is owned by root.

**version 4.1, 2005-01-22.**

- Added archive option.

**version 4.0, 2005-01-20.**

- Added merge option for MYYYYMM, DYYYYMMDD, PYYYYYMM, PYYYYYMM00, and PYYYYYMMDD to merge the monthly partition belonging to previous year to yearly partition, and daily partition belonging to previous month to monthly partition. The merged partition is named as A<number> indicating archived partition, which do not participate keep and next operation.

**version 3.1, 2004-08-18.**

- Padding vnum for PYYYY00, PYYYY0000, PYYYYMM00.

**version 3.0, 2004-06-24.**

- Added support for global indexes with help from Sheck C:
  - (1) "UPDATE GLOBAL INDEXES" for 9i database;
  - (2) "ALTER INDEX ... REBUILD COMPUTE STATISTICS" for 8i database.
- Added 3 more partition scenarios: PYYYY00, PYYYY0000, PYYYYMM00.
- Changed "ANALYZE" statement to "DBMS\_STATS" per recommendation from Sheck C.

**version 2.1, 2004-04-01.**

- Added the option to truncate the old partion instead of drop.

**version 2.0, 2004-03-19.**

- rewritten without use of GNU date or Perl.
- handle 8 cases.
- renamed from "split\_part" to "pman" for marketing purpose.

**version 1.3, 2002-02-07.**

- Added help=pdf and /usr/perl5/bin checking.

**version 1.2, 2002-01-28.**

- When KEEP=<null>, ignore MIN(partition\_name).

**version 1.1, 2002-01-23.**

- Changed “i++” to use “i=\$(next\_day \$i)”.

**version 1.0, 2001-12-18.**

- Based on PL/SQL code from David Xiao.