1 分区技术概述

分区技术（Partitioning）可以将大表、大索引分解为更小、易于管理的块，这些块称为分区（Partition），通过分区技术可以有效的解决大表、大索引带来的问题，对分区表执行的SQL查询或DML语句与普通数据表的语句一样。但是定义了分区后，DML语句可以访问、操作一个单独的分区，而不是整个表或索引，这样通过分区技术就能简化对大数据库对象的管理工作。

使用分区技术，有以下优势：

- 提高数据的可用性；
- 将大段分解为小段，从而减轻管理的负担；
- 改善某些查询的性能（分区修剪）；
- 可以把数据的修改分布到多个分区上，从而减少I/O竞争；

2 分区表的分类

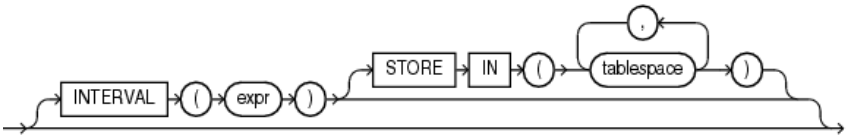
2.1 范围分区

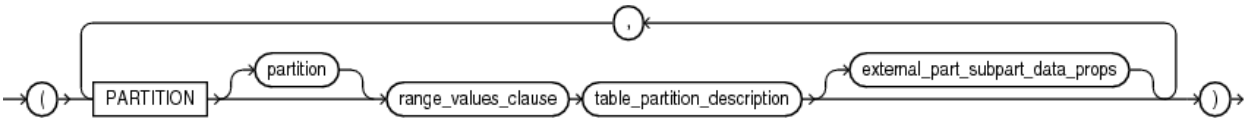
1）范围分区介绍

创建范围分区的关键字是Range，创建该分区后，其中的数据可以根据分区键指定的范围进行分布，当数据在范围内均匀分布时，性能最好。

当表结构采用范围分区时，首先要考虑分区的列应该符合范围分区的方法；其次要考虑列的数据值的取值范围；最后考虑列的边界问题。

2）语法结构





3）示例

```
--创建范围分区表
create table T_RANGE
(
  ID      NUMBER,
  NAME    VARCHAR2(100),
  CDATE   DATE default sysdate,
  CREATOR VARCHAR2(20) default 'system'
)
partition by range (CDATE)
(
  partition P1 values less than (TO_DATE('2018-08-01', 'YYYY-MM-DD')),
  partition P2 values less than (TO_DATE('2018-08-02', 'YYYY-MM-DD')),
  partition P3 values less than (TO_DATE('2018-08-03', 'YYYY-MM-DD')),
  partition P4 values less than (TO_DATE('2018-08-04', 'YYYY-MM-DD')),
  partition P5 values less than (MAXVALUE)
);
--创建序列
create sequence seq_range;
--初始化范围分区表数据
BEGIN
 FOR j IN 1..10 LOOP
   INSERT INTO t_range(ID,NAME,cdate) VALUES(seq_range.nextval,'Range
'||j,SYSDATE-2+j);
 END LOOP;
```

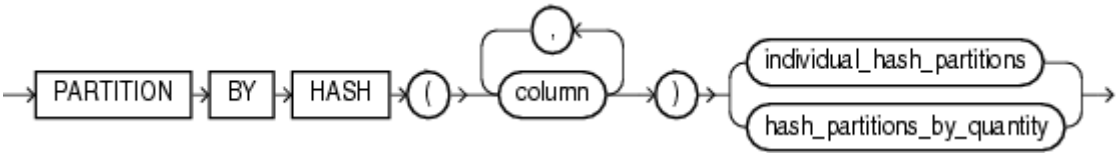
```
 COMMIT;
END;
--查看表数据
SELECT * FROM t_range;

SELECT * FROM t_range PARTITION(p2);
--查看分区信息
SELECT * FROM user_part_tables t WHERE t.table_name = 'T_RANGE';

SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_RANGE';

SELECT * FROM user_segments t WHERE t.segment_name = 'T_RANGE';
```

2.2 散列分区

1）散列分区介绍

对于散列分区表，Oracle会使用一个散列函数对每一条插入数据的分区列值计算出其散列值，以此确定数据应当放在N个分区中的哪个分区中。Oracle建议N是2的一个幂（2、4、8、16等），这样表中的数据才能最好的分布在所有的分区上。散列分区可用于大强度更新、具有高度争用的环境，使用散列分区后，更新不在集中于一个热点段中，而是分散到16个分区上，每个分区都能进行数据的修改。

2）语法结构



3）示例

```
--创建哈希分区表
create table T_HASH
(
  id      NUMBER,
```

```
  name    VARCHAR2(100),
  cdate   DATE default SYSDATE,
  creator VARCHAR2(100) default 'system'
)
partition by hash (ID)
(
  partition P1
    tablespace USERS,
  partition P2
    tablespace TEST
);
--创建序列
CREATE SEQUENCE seq_hash;
--初始化散列表
BEGIN
 FOR j IN 1..10 LOOP
   INSERT INTO t_hash(ID,NAME) VALUES(seq_hash.nextval,'HASH'||j);
 END LOOP;
 COMMIT;
END;
--查看表数据
SELECT * FROM t_hash;
SELECT * FROM t_hash PARTITION(p1);
--查看分区信息
SELECT * FROM User_Part_Tables t WHERE t.table_name='T_HASH';
SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_HASH';
SELECT * FROM User_Segments t WHERE t.segment_name='T_HASH';
```

4）注意事项

- 使用哈希分区，无法控制一行最终放在哪个分区中，Oracle会应用散列函数，并依据散列的结果来确定行会放在哪里；
- 改变散列分区的个数，数据会在所有分区中重新分布，向一个散列分区表增加或删除一个分区时，将导致所有数据都会重写，因为现在每一行都可能属于一个不同的分区；

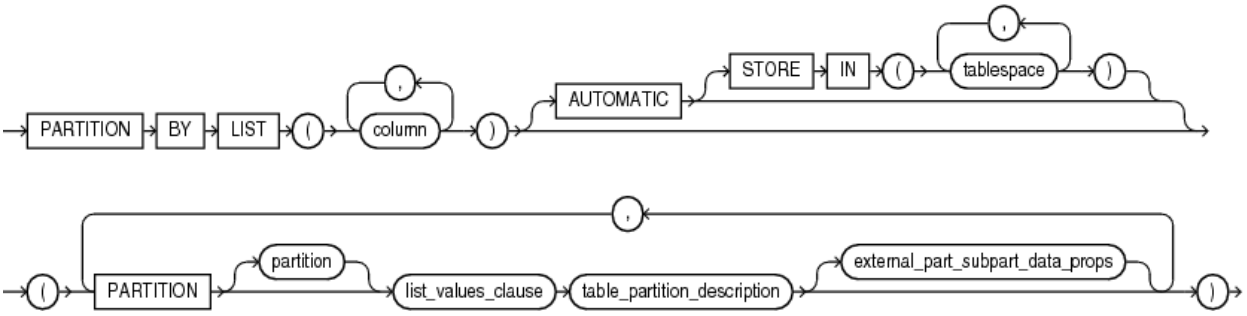
- 分区数应该是2的幂。如果分区数是2的幂，那么分区将会均匀分布，如果不是，那么分区将会不均匀分布。

2.3 列表分区

1）列表分区介绍

列表分区为每个分区指定了一个离散值列表，每行数据根据其分区键值所属列表归到相应的分区中。

2）语法结构



3）示例

```
--创建列表分区
create table T_LIST
(
  id      NUMBER,
  city    VARCHAR2(50),
  cdate   DATE default SYSDATE,
  creator VARCHAR2(20) default 'system'
)
partition by list (CITY)
(
  partition P1 values ('Henan'),
  partition P2 values ('Shandong'),
  partition P3 values ('Guangdong'),
  partition P4 values ('Hunan')
);
```

```
--创建序列
CREATE SEQUENCE seq_list;
--初始化数据
BEGIN
  FOR j IN 1 .. 15
  LOOP
    IF MOD(j, 5) = 1
    THEN
      INSERT INTO t_list(ID,city)VALUES(seq_list.nextval,'Henan');
    ELSIF MOD(j, 5) = 2
    THEN
      INSERT INTO t_list(ID,city)VALUES(seq_list.nextval,'Shandong');
    ELSIF MOD(j, 5) = 3
    THEN
      INSERT INTO t_list(ID,city)VALUES(seq_list.nextval,'Guangdong');
    ELSE
      INSERT INTO t_list (ID,city)VALUES(seq_list.nextval,'Hunan');
    END IF;
  END LOOP;
  COMMIT;
END;
--查看表数据
SELECT * FROM t_list;
SELECT * FROM t_list PARTITION(p1);
--查看分区信息
SELECT * FROM User_Part_Tables t WHERE t.table_name='T_LIST';
SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_LIST';
SELECT * FROM User_Segments t WHERE t.segment_name='T_LIST';
```

思考： 新插入一个列表中没有的省份，会出现什么情况?

```
SQL> INSERT INTO t_list (ID,city)VALUES(seq_list.nextval,'Jilin');
INSERT INTO t_list (ID,city)VALUES(seq_list.nextval,'Jilin')
                                   *
ERROR at line 1:
```

ORA-14400: inserted partition key does not map to any partition

解决：

SQL> alter table t_list add partition pd values(default);

Table altered.

SQL> INSERT INTO t_list (ID,city)VALUES(seq_list.nextval,'Jilin');

1 row created.

SQL> commit;

Commit complete.

注：一旦列表分区表有一个Default分区，就不能再向这个表增加更多的分区了，此时必须删除Default分区，然后再增加分区，再加回Default分区：

SQL> alter table t_list add partition p5 values('Heilongjiang');
alter table t_list add partition p5 values('Heilongjiang')
*
ERROR at line 1:
ORA-14323: cannot add partition when DEFAULT partition exists

SQL> ALTER TABLE t_list DROP PARTITION pd;

Table altered.

SQL> alter table t_list add partition p5 values('Heilongjiang');

Table altered.

SQL> alter table t_list add partition pd values(default);

Table altered.

补充，关于列表分区，Oracle 12c后增加了一些新特性，具体如下：

1）自动列表分区

该分区类型与Interval分区类似，如果创建的列表不在已存在的列表中，则会自动创建新的分区：

```
create table t_auto_list
(
  id      NUMBER,
  city    VARCHAR2(50),
  cdate   DATE default SYSDATE,
  creator VARCHAR2(20) default 'system'
)
partition by list (CITY) Automatic
(
  partition P1 values ('Henan')
);

INSERT INTO t_auto_list(ID,city)VALUES(seq_list.nextval,'Henan');
INSERT INTO t_auto_list(ID,city)VALUES(seq_list.nextval,'Shandong');
COMMIT;

SELECT * FROM User_Part_Tables t WHERE t.table_name='T_AUTO_LIST';
SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_AUTO_LIST';
SELECT * FROM User_Segments t WHERE t.segment_name='T_AUTO_LIST';
```

2）多列列表分区

多列列表分区可以基于多列的列表值对表进行分区：

```
CREATE TABLE sales_by_region_and_channel
```

```
(deptno           NUMBER,
 deptname         VARCHAR2(20),
 quarterly_sales  NUMBER(10,2),
 state            VARCHAR2(2),
 channel          VARCHAR2(1)
)
PARTITION BY LIST (state, channel)
(
 PARTITION q1_northwest_direct VALUES (('OR','D'), ('WA','D')),
 PARTITION q1_northwest_indirect VALUES (('OR','I'), ('WA','I')),
 PARTITION q1_southwest_direct VALUES (('AZ','D'),('UT','D'),('NM','D')),
 PARTITION q1_ca_direct VALUES ('CA','D'),
 PARTITION rest VALUES (DEFAULT)
);
```

2.4 间隔分区

1）间隔分区介绍

间隔分区（interval partition）是Oracle 11g Release 1及以上版本才有的特性，与范围分区类似，使用关键字interval进行分区。

11g之前，维护分区需要手工。11g之后使用interval来实现自动扩展分区，简化了维护：

- 根据年：INTERVAL(NUMTOYMINTERVAL(1,'YEAR'))；
- 根据月：INTERVAL(NUMTOYMINTERVAL(1,'MONTH'))；
- 根据天：INTERVAL(NUMTODSINTERVAL(1,'DAY'))；
- 根据时分秒: NUMTODSINTERVAL(n, { 'DAY'|'HOUR'|'MINUTE'|'SECOND'})

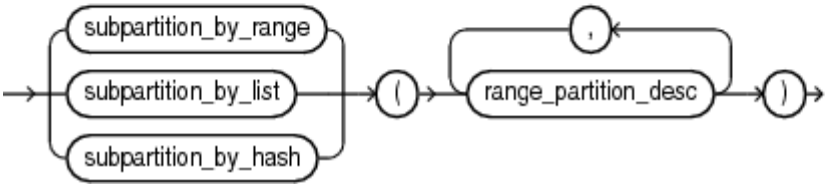
2）示例

```
--创建分区表
create table T_INTERVAL
(
  id      NUMBER,
  name    VARCHAR2(100),
  cdate   DATE default Sysdate,
```

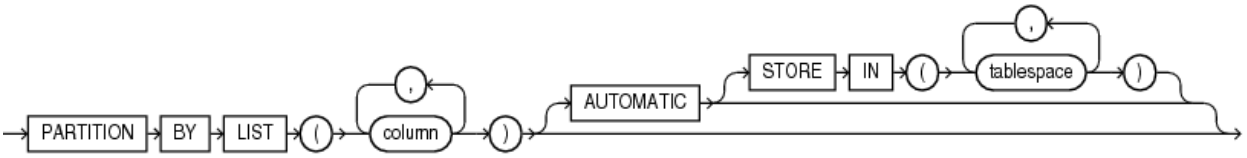
```
  creator VARCHAR2(20) default 'system'
)
partition by range (CDATE)INTERVAL(NUMTODSINTERVAL(1,'DAY'))
(
  partition P1 values less than (TO_DATE('2018-05-22', 'YYYY-MM-DD'))
);
--创建序列
CREATE SEQUENCE seq_interval;
--初始化数据
BEGIN
  FOR j IN 1..5 LOOP
    INSERT INTO t_interval(ID,NAME,cdate) VALUES(seq_interval.nextval,'Inter
'||j,SYSDATE +j);
  END LOOP;
  COMMIT;
END;
--查看数据
SELECT * FROM T_INTERVAL;
SELECT * FROM t_interval PARTITION(SYS_P461);
--查看分区信息
SELECT * FROM User_Part_Tables t WHERE t.table_name='T_INTERVAL';
SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_INTERVAL';
SELECT * FROM User_Segments t WHERE t.segment_name='T_INTERVAL';
```

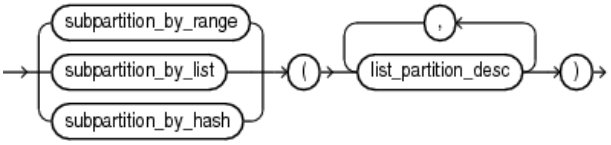
2.5 组合分区

1）组合范围分区

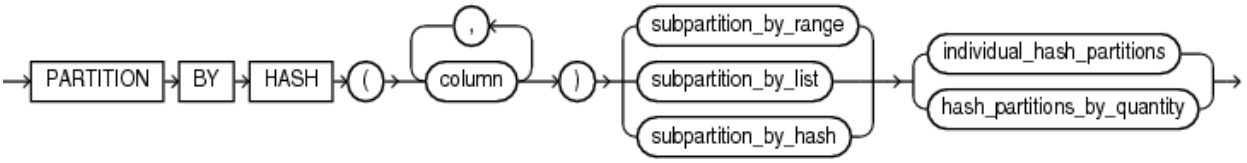


2）组合列表分区





3）组合哈希分区



3 索引分区

3.1 本地前缀分区索引

1）解释本地和前缀

- 本地：所谓本地指索引的分区方法和对应表的分区方法一样；
- 前缀索引：所谓前缀索引是指分区字段是索引字段的前缀。

2）创建本地前缀分区索引

```
CREATE INDEX idx_range_cdate ON t_range(cdate) LOCAL;
CREATE INDEX idx_range_cdate_name ON t_range(cdate,NAME) LOCAL;
```

3）查看分区索引信息

```
SELECT * FROM User_Part_Indexes t WHERE t.table_name='T_RANGE';
SELECT * FROM user_ind_partitions t WHERE t.index_name='IDX_RANGE_CDATE';
```

4）优点

使用本地前缀分区索引查询数据时，往往会极大提高查询性能，因为这样的查询会首先锁定一个索引分区，这样就比使用一个大索引减少了查询时间，而后通过分区索引定位表分区，继而找到需要的数据；
表分区被删除或合并后，Oracle将自动维护索引分区，使得本地前缀分区索引依然有效，从而提高表的可用性。

3.2 本地非前缀分区索引

1）说明

本地非前缀分区索引与本地前缀分区索引的差异就在于分区索引的前缀字段，如果创建的本地分区索引不使用表分区的分区键作为前缀就是本地非前缀分区索引。

2）创建索引

```
CREATE INDEX idx_range_name ON t_range(NAME) LOCAL;
```

3）查看索引信息

```
SELECT * FROM User_Part_Indexes t WHERE t.table_name='T_RANGE';
SELECT * FROM user_ind_partitions t WHERE t.index_name='IDX_RANGE_NAME';
```

4）优点

当执行查询时，Oracle会检索所有的索引分区。提高了索引访问的可用性，即在表分区被drop或merge后，本地非前缀分区索引依然有效。

注：如果历史数据整理很频繁，不能承受全局分区索引重建的长时间等待，则最好使用本地非前缀分区索引。

3.3 全局分区索引

1）说明

全局的含义是索引的分区和表的分区没有关系。全局分区索引有两种类型，一种是全局范围分区索引，一种是全局哈希分区索引。全局范围分区索引使用某个字段作为分区键，可以有效控制索引的分布；全局哈希分区索引的创建由Oracle自动完成，适用于大并发量和批量数据加载的情况。

2）创建索引

```
CREATE INDEX idx_range_id ON t_range(ID)
GLOBAL PARTITION BY HASH(ID)
(
PARTITION p1,
PARTITION p2,
PARTITION p3,
PARTITION p4
);
```

3）查看索引信息

```
SELECT * FROM User_Part_Indexes t WHERE t.table_name='T_RANGE';
SELECT * FROM user_ind_partitions t WHERE t.index_name='IDX_RANGE_ID';
```

4）缺点

如果表分区被删除或合并，全局分区索引无效，需要重建，如果用户的数据无法容忍在分区变动之后长时间的索引重建过程，最好不要使用全局分区索引，因为重建索引占用的很长时间大大降低了数据的可访问性。

4 分区维护

4.1 维护前准备

1）创建分区表

```
CREATE TABLE t_maintain(
owner,
object_name,
object_id,
data_object_id,
object_type,
created,
last_ddl_time,
TIMESTAMP,
status)PARTITION BY RANGE(object_id)
(PARTITION p_3000 VALUES LESS THAN (3000) TABLESPACE USERS,
PARTITION p_6000 VALUES LESS THAN (6000) TABLESPACE USERS,
PARTITION p_max VALUES LESS THAN (MAXVALUE) TABLESPACE USERS
) AS
SELECT owner,
       object_name,
       object_id,
       data_object_id,
       object_type,
       created,
       last_ddl_time,
       TIMESTAMP,
       status
  FROM dba_objects;
```

2）查看分区信息

```
SELECT * FROM User_Part_Tables t WHERE t.table_name='T_MAINTAIN';
SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_MAINTAIN';
SELECT * FROM User_Segments t WHERE t.segment_name='T_MAINTAIN';
```

3）创建全局索引和本地前缀分区索引

对于本地分区索引不会受到表分区的变化影响，本地分区索引会自动维护；对于全局分区索引，一旦表分区变化则失效，需要重建。

```
CREATE INDEX idx_maintain_owner ON t_maintain(owner) GLOBAL;
CREATE INDEX idx_maintain_objectid ON t_maintain(object_id) LOCAL;
```

4）查看索引状态

```
SELECT * FROM User_Part_Indexes t WHERE t.table_name='T_MAINTAIN';

SELECT * FROM user_ind_partitions t WHERE
t.index_name='IDX_MAINTAIN_OBJECTID';

SELECT t.index_name,t.status,t.* FROM User_Indexes t
WHERE t.table_name='T_MAINTAIN';
```

4.2 新增分区

1）新增分区

如果分区边界不是maxvalue，则可以直接添加新的分区，否则，需删除原有分区，再添加，或者采用分区的拆分，如果没有删除边界分区而直接增加分区，则出错。

```
ALTER TABLE t_maintain ADD PARTITION p_9000 VALUES LESS THAN(9000)
TABLESPACE USERS;
```

ORA-14074: partition bound must collate higher than that of the last partition

2）删除默认分区再增加分区

```
ALTER TABLE t_maintain DROP PARTITION p_max;
ALTER TABLE t_maintain ADD PARTITION p_9000 VALUES LESS THAN(9000)
TABLESPACE USERS;
```

3）增加默认分区

```
ALTER TABLE t_maintain ADD PARTITION p_max VALUES LESS THAN(MAXVALUE)
TABLESPACE USERS;
```

4）查看新增后分区信息

```
SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_MAINTAIN';
```

5）查看索引状态

```
SELECT t.index_name,t.status,t.* FROM User_Indexes t
WHERE t.table_name='T_MAINTAIN';
```

从查询结果可知，IDX_MAINTAIN_OWNER索引状态不可用，所以需要重建全局索引。

6）重建索引

```
ALTER INDEX IDX_MAINTAIN_OWNER REBUILD;

SELECT t.index_name,t.status,t.* FROM User_Indexes t
WHERE t.table_name='T_MAINTAIN';
```

4.3 移动分区

1）移动分区

移动分区是将分区从一个表空间移动到另外一个表空间，比如当我们的一个磁盘需要更换或维修时，并且该磁盘存放了分区对象的数据，此时就可以通过移动分区将分区迁移到别的表空间。

2）创建新的表空间

```
CREATE TABLESPACE maintain DATAFILE
'/u01/app/oracle/oradata/orcl/maintain01.dbf' SIZE 100M;
```

3）移动分区到新表空间

```
ALTER TABLE T_MAINTAIN MOVE PARTITION P_6000 TABLESPACE maintain;
```

4）查看分区

```
SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_MAINTAIN';
```

5）查看索引状态

```
SELECT t.index_name,t.status,t.* FROM User_Indexes t
WHERE t.table_name='T_MAINTAIN';
```

6）重建索引

```
ALTER INDEX IDX_MAINTAIN_OWNER REBUILD;

SELECT t.index_name,t.status,t.* FROM User_Indexes t
WHERE t.table_name='T_MAINTAIN';
```

4.4 截断分区

1）截断分区

截断（truncate）相对删除操作很快，数据仓库中的大量数据的批量数据加载可能会用到，截断分区同样会自动维护局部分区索引，同时会使全局索引不可用，所以需要重建全局分区

索引。

ALTER TABLE T_MAINTAIN TRUNCATE PARTITION p_3000;

2）查看分区数据

SELECT * FROM t_maintain PARTITION(P_3000);

3）查看索引状态

SELECT t.index_name,t.status,t.* FROM User_Indexes t
WHERE t.table_name='T_MAINTAIN';

4）重建索引

ALTER INDEX IDX_MAINTAIN_OWNER REBUILD;

SELECT t.index_name,t.status,t.* FROM User_Indexes t
WHERE t.table_name='T_MAINTAIN';

4.5 删除分区

1）删除分区

使用Drop指令直接删除分区，依然会造成全局索引失效。

ALTER TABLE T_MAINTAIN DROP PARTITION P_6000;

2）查看分区

SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_MAINTAIN';

3）查看索引状态

SELECT t.index_name,t.status,t.* FROM User_Indexes t

WHERE t.table_name='T_MAINTAIN';

4）重建索引

ALTER INDEX IDX_MAINTAIN_OWNER REBUILD;

4.6 拆分分区

1）拆分分区

是把一个分区拆成两个分区，比如把p_9000分区拆为p_6000，用户存放object_id>=3000and object_id < 6000；p_9000用户存放object_id>=6000and object_id < 9000的记录，利用split技术可实现这种拆分。

```
DROP TABLE t_maintain;
CREATE TABLE t_maintain(
owner,
object_name,
object_id,
data_object_id,
object_type,
created,
last_ddl_time,
TIMESTAMP,
status)PARTITION BY RANGE(object_id)
(PARTITION p_3000 VALUES LESS THAN (3000) TABLESPACE USERS,
PARTITION p_9000 VALUES LESS THAN (9000) TABLESPACE USERS,
PARTITION p_max VALUES LESS THAN (MAXVALUE) TABLESPACE USERS
) AS
SELECT owner,
       object_name,
       object_id,
       data_object_id,
       object_type,
       created,
```

```
    last_ddl_time,
    TIMESTAMP,
    status
  FROM dba_objects;

SELECT '<3000',  sum(CASE WHEN t.object_id<3000 THEN 1 ELSE 0 END ) qty1 ,
'3000-6000',sum(CASE WHEN t.object_id>=3000 AND t.object_id<6000 THEN 1
ELSE 0 END ）qty2,
'3000-9000',sum(CASE WHEN t.object_id>=3000 AND t.object_id<9000 THEN 1
ELSE 0 END ）qty3,
'6000-9000',sum(CASE WHEN t.object_id>=6000 AND t.object_id<9000 THEN 1
ELSE 0 END ）qty4,
'>9000',sum(CASE WHEN t.object_id>=9000 THEN 1 END ）qty5,
COUNT(1) total
FROM T_MAINTAIN t;
```

2）查看分区

```
SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_MAINTAIN';
```

3）拆分分区

```
ALTER TABLE T_MAINTAIN SPLIT PARTITION P_9000 AT(6000) INTO (
PARTITION p_6000 TABLESPACE USERS,
PARTITION p_9000 TABLESPACE maintain);
```

4）再次查看分区信息

```
SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_MAINTAIN';
```

5）查询分区的边界

```
SELECT MIN(object_id),MAX(object_id) FROM t_maintain PARTITION(p_6000);
```

4.7 合并分区

1）合并分区

相邻的分区可以合并为一个分区，新分区的下边界为原来边界值较低的分区，上边界为原来边界值较高的分区，原先的局部索引相应也会合并，全局索引会失效，需重建。

ALTER TABLE T_MAINTAIN MERGE PARTITIONS p_6000,p_9000 INTO PARTITION p_9000;

2）查看分区信息

SELECT * FROM User_Tab_Partitions t WHERE t.table_name='T_MAINTAIN';

3）查看分区边界

SELECT MIN(object_id),MAX(object_id) FROM t_maintain PARTITION(p_9000);

4.8 交换分区

1）交换分区

分区交换可以把一个表和分区表中的一个分区中的数据进行对换，分区的交换只是一个数据字典的操作，因此操作速度很快。

```
DROP TABLE t_maintain;
CREATE TABLE t_maintain(
owner,
object_name,
object_id,
data_object_id,
object_type,
created,
last_ddl_time,
TIMESTAMP,
status)PARTITION BY RANGE(object_id)
```

```
(PARTITION p_3000 VALUES LESS THAN (3000) TABLESPACE USERS,
PARTITION p_6000 VALUES LESS THAN (6000) TABLESPACE USERS,
PARTITION p_9000 VALUES LESS THAN (9000) TABLESPACE USERS,
PARTITION p_max VALUES LESS THAN (MAXVALUE) TABLESPACE USERS
) AS
SELECT owner,
       object_name,
       object_id,
       data_object_id,
       object_type,
       created,
       last_ddl_time,
       TIMESTAMP,
       status
  FROM dba_objects;

ALTER TABLE  t_maintain TRUNCATE PARTITION p_6000;
```

2）构建一个表，表结构和分区表相同

```
CREATE TABLE t_6000
 AS
SELECT owner,
       object_name,
       object_id,
       data_object_id,
       object_type,
       created,
       last_ddl_time,
       TIMESTAMP,
       status
  FROM dba_objects t
  WHERE t.object_id>=3000 AND t.object_id<6000;
```

3）使用Exchange指令交换分区

```
ALTER TABLE t_maintain EXCHANGE PARTITION p_6000 WITH TABLE t_6000;
```

4）查看分区p_6000

```
SELECT COUNT(1) FROM t_maintain PARTITION (p_6000);
```

5）验证原表t_6000没有数据

```
SELECT COUNT(1) FROM t_6000;
```

注：如果交换的表中包含的记录不符合分区的规定，可以使用without validation字句跳过检查，具体参照下面的演示：

```
ALTER TABLE  t_maintain TRUNCATE PARTITION p_6000;
DROP TABLE t_6000;
CREATE TABLE t_6000
 AS
SELECT owner,
       object_name,
       object_id,
       data_object_id,
       object_type,
       created,
       last_ddl_time,
       TIMESTAMP,
       status
  FROM dba_objects t
  WHERE t.object_id>=3000 AND t.object_id<7000;
ALTER TABLE t_maintain EXCHANGE PARTITION p_6000 WITH TABLE t_6000;
ORA-14099: all rows in table do not qualify for specified partition
ALTER TABLE t_maintain EXCHANGE PARTITION p_6000 WITH TABLE t_6000
WITHOUT validation;--不建议采用此种方式
SELECT COUNT(1) FROM t_maintain PARTITION (p_6000);
```

4.9 非分区表转换为分区表

Oracle 12c的新特性，可以将非分区表转为分区表，通过指定ONLINE，在转换过程中可以继续进行DML并发操作。

```
CREATE TABLE t_partition AS SELECT *FROM Dba_Objects;

CREATE INDEX idx_partition_owner ON t_partition(owner);
CREATE INDEX idx_partition_objectid ON t_partition(object_id);

ALTER TABLE t_partition MODIFY
  PARTITION BY RANGE (OBJECT_ID)
  ( PARTITION P1 VALUES LESS THAN (3000),
    PARTITION P2 VALUES LESS THAN (6000),
    PARTITION P3 VALUES LESS THAN (9000),
    PARTITION P4 VALUES LESS THAN (MAXVALUE)
   ) ONLINE
  UPDATE INDEXES
 ( idx_partition_objectid LOCAL,
   idx_partition_owner GLOBAL
 );
```