[三思笔记]全面学习分区表及分区索引

2008-04-15

- **管理索引分区**
  - 增加索引分区(Adding Index Partitions)
  - 删除索引分区(Dropping Index Partitions)
  - 重编译索引分区(Rebuilding Index Partitions)
  - 重命名索引分区(Renaming Index Partitions)
  - 拆分索引分区(Splitting Index Partitions)
  - 修改索引分区默认属性(Modifying Default Attributes of Index Partitions)
  - 修改索引分区当前属性(Modifying Real Attributes of Index Partitions)

关于分区表和分区索引(About Partitioned Tables and Indexes)

对于 10gR2 而言，基本上可以分成几类：

- Range(范围)分区
- Hash(哈希)分区
- List(列表)分区
- 以及组合分区：Range-Hash,Range-List。

对于表而言(常规意义上的堆组织表)，上述分区形式都可以应用(甚至可以对某个分区指定 compress 属性)，只不过分区依赖列不能是 lob,long 之类数据类型，每个表的分区或子分区数的总数不能超过 1024K-1 个。

对于索引组织表，只能够支持普通分区方式，不支持组合分区，常规表的限制对于索引组织表同样有效，除此之外呢，还有一些其实的限制，比如要求索引组织表的分区依赖列必须是主键才可以等。

注：本篇所有示例仅针对常规表，即堆组织表！

对于索引，需要区分创建的是全局索引，或本地索引：

- 全局索引(global index)：即可以分区，也可以不分区。即可以建 range 分区，也可以建 hash 分区，即可建于分区表，又可创建于非分区表上，就是说，全局索引是完全独立的，因此它也需要我们更多的维护操作。
- 本地索引(local index)：其分区形式与表的分区完全相同，依赖列相同，存储属性也相同。对于本地索引，其索引分区的维护自动进行，就是说你 add/drop/split/truncate 表的分区时，本地索引会自动维护其索引分区。

Oracle 建议如果单个表超过 2G 就最好对其进行分区，对于大表创建分区的好处是显而易见的，这里不多论述 why，而将重点放在 when 以及 how。

WHEN

一、When 使用 Range 分区

Range 分区呢是应用范围比较广的表分区方式，它是以列的值的范围来做为分区的划分条件，将记录存放到列值所在的 range 分区中，比如按照时间划分，2008 年 1 季度的数据放到 a 分区，08 年 2 季度的数据放到 b 分区，因此在创建的时候呢，需要你指定基于的列，以及分区的范围值，如果某些记录暂无法预测范围，可以创建 maxvalue 分区，所有不在指定范围内的记录都会被存储到 maxvalue 所在分区中，并且支持指定多列做为依赖列，后面在讲 how 的时候会详细谈到。

二、When 使用 Hash 分区

通常呢，对于那些无法有效划分范围的表，可以使用 hash 分区，这样对于提高性能还是会有一定的帮助。hash 分区会将表中的数据平均分配到你指定的几个分区中，列所在分区是依据分区列的 hash 值自动分配，因此你并不能控制也不知道哪条记录会被放到哪个分区中，hash 分区也可以支持多个依赖列。

三、When 使用 List 分区

List 分区与 range 分区和 hash 分区都有类似之处，该分区与 range 分区类似的是也需要你指定列的值，但这又不同与 range 分区的范围式列值---其分区值必须明确指定，也不同与 hash 分区---通过明确指定分区值，你能控制记录存储在哪个分区。它的分区列只能有一个，而不能像 range 或者 hash 分区那样同时指定多个列做为分区依赖列，不过呢，它的单个分区对应值可以是多个。

你在分区时必须确定分区列可能存在的值，一旦插入的列值不在分区范围内，则插入/更新就会失败，因此通常建议使用 list 分区时，要创建一个 default 分区存储那些不在指定范围内的记录，类似 range 分区中的 maxvalue 分区。

四、When 使用组合分区

如果某表按照某列分区之后，仍然较大，或者是一些其它的需求，还可以通过分区内再建子分区的方式将分区再分区，即组合分区的方式。

组合分区呢在 10g 中有两种：range-hash，range-list。注意顺序哟，根分区只能是 range 分区，子分区可以是 hash 分区或 list 分区。

提示：11g 在组合分区功能这块有所增强，又推出了 range-range,list-range,list-list,list-hash，这就相当于除 hash 外三种分区方式的笛卡尔形式都有了。为什么会没有 hash 做为根分区的组合分区形式呢，再仔细回味一下第二点，你一定能够想明白~~。

HOW

一、如何创建

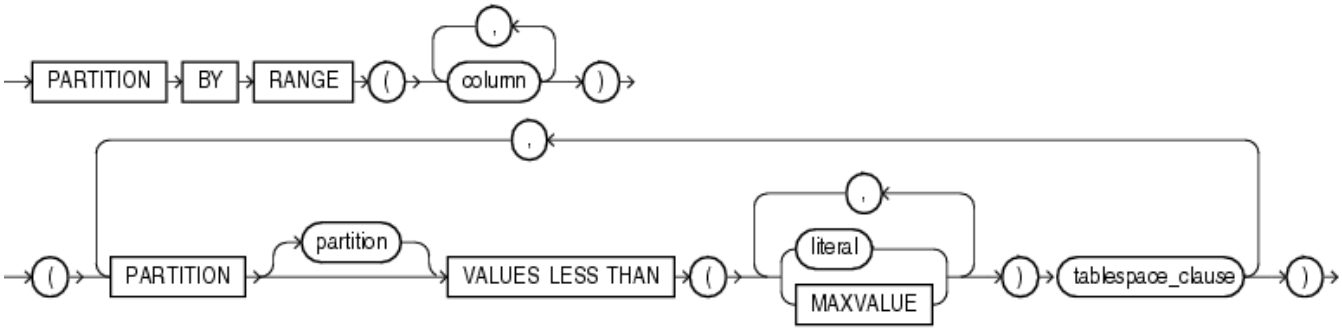
如果想对某个表做分区，必须在创建表时就指定分区，我们可以对一个包含分区的表中的分区做修改，但不能直接将一个未分区的表修改成分区表(起码在 10g 是不行的，当然你可能会说，可以通过在线重定义的方式，但是这不是直接哟，这也是借助临时表间接实现的)。

创建表或索引的语法就不说了，大家肯定比我还熟悉，而想在建表(索引)同时指定分区也非常容易，只需要把创建分区的子句放到";"前就行啦，同时需要注意表的 row movement 属性，它用来控制是否允许修改列值所造成的记录移动至其它分区存储，有 enable|disable 两种状态，默认是 disable row movement，当 disable 时，如果记录要被更新至其它分区，则更新语句会报错。

下面分别演示不同分区方式的表和索引的创建：

1、创建 range 分区

语法如下，图：[range_partitioning.gif]



需要我们指定的有：

- column:分区依赖列(如果是多个，以逗号分隔);

- partition:分区名称;
- values less than:后跟分区范围值(如果依赖列有多个，范围对应值也应是多个，中间以逗号分隔);
- tablespace_clause:分区的存储属性，例如所在表空间等属性(可为空)，默认继承基表所在表空间的属性。

① 创建一个标准的 range 分区表：

```
JSSWEB> create table t_partition_range (id number,name varchar2(50))
  2  partition by range(id)(
  3  partition t_range_p1 values less than (10) tablespace tbspart01,
  4  partition t_range_p2 values less than (20) tablespace tbspart02,
  5  partition t_range_p3 values less than (30) tablespace tbspart03,
  6  partition t_range_pmax values less than (maxvalue) tablespace tbspart04
  7  );

表已创建。
```

要查询创建分区的信息，可以通过查询 user_part_tables,user_tab_partitions 两个数据字典（索引分区、组织分区等信息也有对应的数据字典，后续示例会逐步提及）。

user_part_tables：记录分区的表的信息;

user_tab_partitions：记录表的分区的信息。

例如：

```
JSSWEB> select table_name,partitioning_type,partition_count
  2  From user_part_tables where table_name='T_PARTITION_RANGE';

TABLE_NAME                     PARTITI PARTITION_COUNT
------------------------------ ------- ---------------
T_PARTITION_RANGE              RANGE                 4

JSSWEB> select partition_name,high_value,tablespace_name
  2  from user_tab_partitions where table_name='T_PARTITION_RANGE'
  3  order by partition_position;

PARTITION_NAME                 HIGH_VALUE TABLESPACE_NAME
------------------------------ ---------- --------------------
T_RANGE_P1                     10         TBSPART01
T_RANGE_P2                     20         TBSPART02
T_RANGE_P3                     30         TBSPART03
T_RANGE_PMAX                   MAXVALUE   TBSPART04
```

② 创建 global 索引 range 分区：

```
JSSWEB> create index idx_parti_range_id on t_partition_range(id)
  2  global partition by range(id)(
  3  partition i_range_p1 values less than (10) tablespace tbspart01,
```

```
  4  partition i_range_p2 values less than (40) tablespace tbspart02,
  5  partition i_range_pmax values less than (maxvalue) tablespace tbspart03);

索引已创建。
```

由上例可以看出，创建 global 索引的分区与创建表的分区语句格式完全相同，而且其分区形式与索引所在表的分区形式没有关联关系。

注意：我们这里借助上面的表 t_partition_range 来演示创建 range 分区的 global 索引，并不表示 range 分区的表，只能创建 range 分区的 global 索引，只要你想，也可以为其创建 hash 分区的 global 索引。

查询索引的分区信息可以通过 user_part_indexes、user_ind_partitions 两个数据字典：

```
JSSWEB> select index_name, partitioning_type, partition_count
  2    From user_part_indexes
  3   where index_name = 'IDX_PARTI_RANGE_ID';

INDEX_NAME                     PARTITI PARTITION_COUNT
------------------------------ ------- ---------------
IDX_PARTI_RANGE_ID             RANGE                 3

JSSWEB> select partition_name, high_value, tablespace_name
  2    from user_ind_partitions
  3   where index_name = 'IDX_PARTI_RANGE_ID'
  4   order by partition_position;

PARTITION_NAME                 HIGH_VALUE TABLESPACE_NAME
------------------------------ ---------- --------------------
I_RANGE_P1                     10         TBSPART01
I_RANGE_P2                     40         TBSPART02
I_RANGE_PMAX                   MAXVALUE   TBSPART03
```

③ Local 分区索引的创建最简单，例如：

仍然借助 t_partition_range 表来创建索引

```
--首先删除之前创建的 global 索引
JSSWEB> drop index IDX_PARTI_RANGE_ID;

索引已删除。

JSSWEB> create index IDX_PARTI_RANGE_ID on T_PARTITION_RANGE(id) local;

索引已创建。
```

查询相关数据字典：

```
JSSWEB> select index_name, partitioning_type, partition_count
```

```
  2    From user_part_indexes
  3   where index_name = 'IDX_PARTI_RANGE_ID';

INDEX_NAME                     PARTITI PARTITION_COUNT
------------------------------ ------- ---------------
IDX_PARTI_RANGE_ID             RANGE                 4

JSSWEB> select partition_name, high_value, tablespace_name
  2    from user_ind_partitions
  3   where index_name = 'IDX_PARTI_RANGE_ID'
  4   order by partition_position;

PARTITION_NAME                 HIGH_VALUE TABLESPACE_NAME
------------------------------ ---------- --------------------
T_RANGE_P1                     10         TBSPART01
T_RANGE_P2                     20         TBSPART02
T_RANGE_P3                     30         TBSPART03
T_RANGE_PMAX                   MAXVALUE   TBSPART04
```

可以看出，local 索引的分区完全继承表的分区的属性，包括分区类型，分区的范围值即不需指定也不能更改，这就是前面说的：local 索引的分区维护完全依赖于其索引所在表。

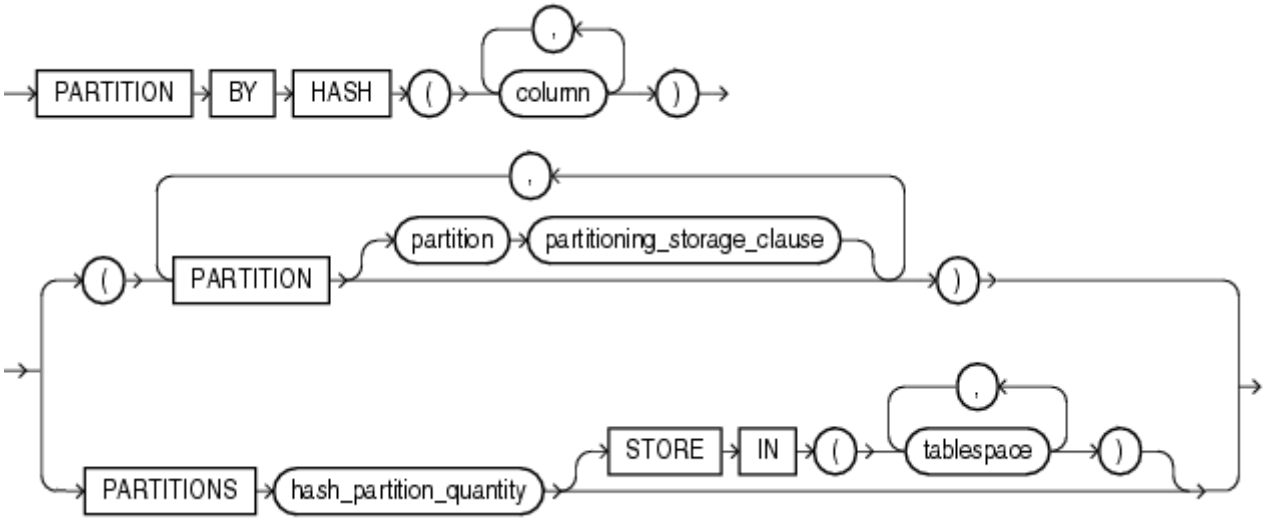
不过呢分区名称，以及分区所在表空间等信息是可以自定义的，例如：

```
SQL> create index IDX_PART_RANGE_ID ON T_PARTITION_RANGE(id) local (
  2  partition i_range_p1 tablespace tbspart01,
  3  partition i_range_p2 tablespace tbspart01,
  4  partition i_range_p3 tablespace tbspart02,
  5  partition i_range_pmax tablespace tbspart02
  6  );

索引已创建。

SQL> select index_name, partitioning_type, partition_count
  2    From user_part_indexes
  3   where index_name = 'IDX_PART_RANGE_ID';

INDEX_NAME                     PARTITI PARTITION_COUNT
------------------------------ ------- ---------------
IDX_PART_RANGE_ID              RANGE                 4

SQL> select partition_name, high_value, tablespace_name
  2    from user_ind_partitions
  3   where index_name = 'IDX_PART_RANGE_ID'
  4   order by partition_position;
```

```
PARTITION_NAME    HIGH_VALUE          TABLESPACE_NAME
--------------- --------------- --------------------
I_RANGE_P1          10                  TBSPART01
I_RANGE_P2          20                  TBSPART01
I_RANGE_P3          30                  TBSPART02
I_RANGE_PMAX       MAXVALUE             TBSPART02
```

2、创建 hash 分区

语法如下：[图:hash_partitioning.gif]



语法看起来比 range 复杂，其实使用起来比 range 更简单，这里需要我们指定的有：

- column:分区依赖列(支持多个，中间以逗号分隔);
- partition:指定分区，有两种方式：
  - 直接指定分区名，分区所在表空间等信息
  - 只指定分区数量，和可供使用的表空间。

① 创建 hash 分区表

```
JSSWEB> create table t_partition_hash (id number,name varchar2(50))
  2  partition by hash(id)(
  3  partition t_hash_p1 tablespace tbspart01,
  4  partition t_hash_p2 tablespace tbspart02,
  5  partition t_hash_p3 tablespace tbspart03);

表已创建。
```

要实现同样效果，你还可以这样：

```
JSSWEB> create table t_partition_hash2 (id number,name varchar2(50))
  2  partition by hash(id)
  3  partitions 3 store in(tbspart01,tbspart02,tbspart03);
```

表已创建。

这就是上面说的，直接指定分区数量和可供使用的表空间。

提示：**这里分区数量和可供使用的表空间数量之间没有直接对应关系。**分区数并不一定要等于表空间数。

要查询表的分区信息，仍然是通过 user_part_tables,user_tab_partitions 两个数据字典，这里不再举例。

② Global 索引 hash 分区

Hash 分区索引的子句与 hash 分区表的创建子句完全相同，例如：

```
JSSWEB> create index idx_part_hash_id on t_partition_hash(id)
  2  global partition by hash(id)
  3  partitions 3 store in(tbspart01,tbspart02,tbspart03);

索引已创建。
```

查询索引的分区信息也仍是通过 user_part_indexes、user_ind_partitions 两个数据字典，不再举例。

③ 创建 Local 索引

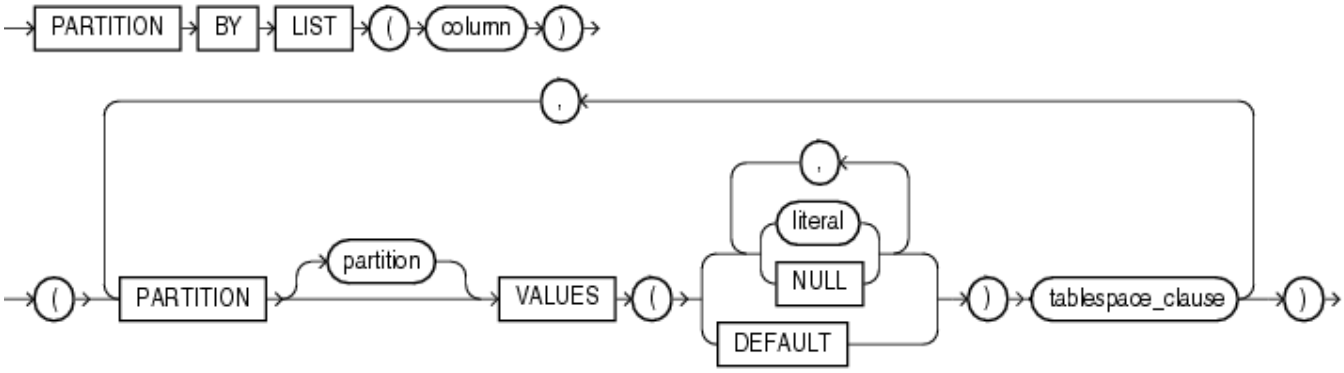
在前面学习 range 分区时，我们已经对 Local 索引的特性做了非常清晰的概述，因此这里也不再举例，如有疑问，建议再仔细复习 range 分区的相关示例，如果还有疑问，当面问我好了:)

综上：

- 对于 global 索引分区而言，在 10g 中只能支持 range 分区和 hash 分区，因此后续示例中不会再提及。
- 对于 local 索引分区而言，其分区形式完全依赖于索引所在表的分区形式，不管从创建语法还是理解难度均无技术含量，因此后续也不再提供示例。
- 注意，在创建索引时如果不显式指定 global 或 local，则默认是 global。
- 注意，在创建 global 索引时如果不显式指定分区子句，则默认不分区(废话)。

3、创建 list 分区

创建语法如下：[图：list_partitioning.gif]



需要我们指定的有：

- column:分区依赖列，注意：只能是一个;
- partition:分区名称;
- literal:分区对应值，注意：每个分区可以对应多个值;
- tablespace_clause:分区的存储属性，例如所在表空间等属性(可为空)，默认继承基表所在表空间的属性。

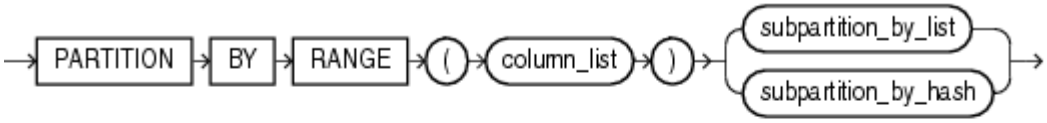
创建 list 分区表示例：

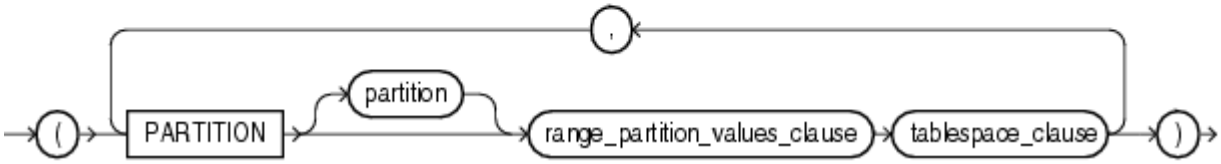
```
JSSWEB> create table t_partition_list (id number,name varchar2(50))
  2  partition by list(id)(
  3  partition t_list_p1 values (1,2,3,4,5,6,7,8,9) tablespace tbspart01,
  4  partition t_list_p2 values (10,11,12,13,14,15,16,17,18,19) tablespace tbspart02,
  5  partition t_list_p3 values (20,21,22,23,24,25,26,27,28,29) tablespace tbspart03,
  6  partition t_list_pd values (default) tablespace tbspart04);

表已创建。
```

上例能够实现与前面 range 分区示例相同的效果，当然针对本示例而言，list 分区显然不好用啊~~~

4、创建 range-hash 组合分区

语法如下：图[composite_partitioning.gif]





需要我们指定的有：

- column_list:分区依赖列(支持多个，中间以逗号分隔);
- subpartition:子分区方式，有两处：
  - Subpartition_by_list:语法与 list 分区完全相同，只不过把关键字 partition 换成 subpartition
  - Subpartition_by_hash:语法与 hash 分区完全相同，只不过把关键字 partition 换成 subpartition
- partition:分区名称;
- range_partition_values_clause:与 range 分区范围值的语法;
- tablespace_clause:分区的存储属性，例如所在表空间等属性(可为空)，默认继承基表所在表空间的属性。

组合分区相对于普通分区，语法上稍稍复杂了一些，但也正因如此，其子分区的创建可以非常灵活，下面分别举几个例子(注：仅示例，并非穷举所有形式)

① 为所有分区各创建 4 个 hash 子分区

```
JSSWEB> create table t_partition_rh (id number,name varchar2(50))
  2  partition by range(id) subpartition by hash(name)
  3  subpartitions 4 store in (tbspart01, tbspart02, tbspart03,tbspart04)(
  4  partition t_r_p1 values less than (10) tablespace tbspart01,
  5  partition t_r_p2 values less than (20) tablespace tbspart02,
  6  partition t_r_p3 values less than (30) tablespace tbspart03,
  7  partition t_r_pd values less than (maxvalue) tablespace tbspart04);

表已创建。

JSSWEB> select partitioning_type,subpartitioning_type,partition_count,def_subpartition_count
  2  From user_part_tables where table_name='T_PARTITION_RH';

PARTITI SUBPART PARTITION_COUNT DEF_SUBPARTITION_COUNT
------- ------- --------------- ---------------------
RANGE   HASH                  4                     4

JSSWEB> select partition_name,subpartition_count,high_value
  2  from user_tab_partitions where table_name='T_PARTITION_RH';

PARTITION_NAME    SUBPARTITION_COUNT HIGH_VALUE
--------------- ------------------ ----------
T_R_P2                             4 20
T_R_P3                             4 30
T_R_PD                             4 MAXVALUE
T_R_P1                             4 10

JSSWEB> select partition_name,subpartition_name,tablespace_name
  2  from user_tab_subpartitions where table_name='T_PARTITION_RH';

PARTITION_NAME    SUBPARTITION_NAME                TABLESPACE_NAME
--------------- ------------------------------ --------------------
T_R_P2            SYS_SUBP140                    TBSPART02
T_R_P2            SYS_SUBP139                    TBSPART02
T_R_P2            SYS_SUBP138                    TBSPART02
T_R_P2            SYS_SUBP137                    TBSPART02
T_R_P3            SYS_SUBP144                    TBSPART03
T_R_P3            SYS_SUBP143                    TBSPART03
T_R_P3            SYS_SUBP142                    TBSPART03
T_R_P3            SYS_SUBP141                    TBSPART03
T_R_PD            SYS_SUBP148                    TBSPART04
T_R_PD            SYS_SUBP147                    TBSPART04
T_R_PD            SYS_SUBP146                    TBSPART04
T_R_PD            SYS_SUBP145                    TBSPART04
```

```
T_R_P1          SYS_SUBP133                    TBSPART01
T_R_P1          SYS_SUBP136                    TBSPART01
T_R_P1          SYS_SUBP135                    TBSPART01
T_R_P1          SYS_SUBP134                    TBSPART01

已选择 16 行。
```

这里我们要学到一个新的数据字典：**user_tab_subpartitions**，用于查询表的子分区信息。

② 对某个分区创建 **hash** 子分区

```
JSSWEB> create table t_partition_rh (id number,name varchar2(50))
  2  partition by range(id) subpartition by hash(name)(
  3  partition t_r_p1 values less than (10) tablespace tbspart01,
  4  partition t_r_p2 values less than (20) tablespace tbspart02,
  5  partition t_r_p3 values less than (30) tablespace tbspart03
  6  (subpartition t_r_p3_h1 tablespace tbspart01,
  7   subpartition t_r_p3_h2 tablespace tbspart02,
  8   subpartition t_r_p3_h3 tablespace tbspart03),
  9  partition t_r_pd values less than (maxvalue) tablespace tbspart04);

表已创建。

JSSWEB> select partitioning_type,subpartitioning_type,partition_count,def_subpartition_count
  2  From user_part_tables where table_name='T_PARTITION_RH';

PARTITI SUBPART PARTITION_COUNT DEF_SUBPARTITION_COUNT
------- ------- --------------- ---------------------
RANGE   HASH                  4                     1

JSSWEB> select partition_name,subpartition_count,high_value
  2  from user_tab_partitions where table_name='T_PARTITION_RH';

PARTITION_NAME    SUBPARTITION_COUNT HIGH_VALUE
--------------- ------------------ ----------
T_R_P1                           1 10
T_R_P2                           1 20
T_R_P3                           3 30
T_R_PD                           1 MAXVALUE

JSSWEB> select partition_name,subpartition_name,tablespace_name
  2  from user_tab_subpartitions where table_name='T_PARTITION_RH';

PARTITION_NAME    SUBPARTITION_NAME                TABLESPACE_NAME
--------------- ------------------------------ --------------------
```

```
T_R_P1          SYS_SUBP149                    TBSPART01
T_R_P2          SYS_SUBP150                    TBSPART02
T_R_P3          T_R_P3_H3                      TBSPART03
T_R_P3          T_R_P3_H2                      TBSPART02
T_R_P3          T_R_P3_H1                      TBSPART01
T_R_PD          SYS_SUBP151                    TBSPART04

已选择 6 行。
```

当然，还可以给各个分区指定不同的子分区

```
JSSWEB> create table t_partition_rh (id number,name varchar2(50))
  2  partition by range(id) subpartition by hash(name)(
  3  partition t_r_p1 values less than (10) tablespace tbspart01,
  4  partition t_r_p2 values less than (20) tablespace tbspart02
  5  (subpartition t_r_p2_h1 tablespace tbspart01,
  6   subpartition t_r_p2_h2 tablespace tbspart02),
  7  partition t_r_p3 values less than (30) tablespace tbspart03
  8   subpartitions 3 store in (tbspart01,tbspart02,tbspart03),
  9  partition t_r_pd values less than (maxvalue) tablespace tbspart04
 10  (subpartition t_r_p3_h1 tablespace tbspart01,
 11   subpartition t_r_p3_h2 tablespace tbspart02,
 12   subpartition t_r_p3_h3 tablespace tbspart03)
 13  );

表已创建。

JSSWEB> select partitioning_type,subpartitioning_type,partition_count,def_subpartition_count
  2  From user_part_tables where table_name='T_PARTITION_RH';

PARTITI SUBPART PARTITION_COUNT DEF_SUBPARTITION_COUNT
------- ------- --------------- ---------------------
RANGE     HASH                4                     1

JSSWEB> select partition_name,subpartition_count,high_value
  2  from user_tab_partitions where table_name='T_PARTITION_RH';

PARTITION_NAME    SUBPARTITION_COUNT HIGH_VALUE
--------------- ------------------ ----------
T_R_P1                           1 10
T_R_P2                           2 20
T_R_P3                           3 30
T_R_PD                           3 MAXVALUE

JSSWEB> select partition_name,subpartition_name,tablespace_name
```

```
  2    from user_tab_subpartitions where table_name='T_PARTITION_RH';

PARTITION_NAME    SUBPARTITION_NAME                TABLESPACE_NAME
--------------- ------------------------------ --------------------
T_R_P1            SYS_SUBP152                     TBSPART01
T_R_P2            T_R_P2_H2                      TBSPART02
T_R_P2            T_R_P2_H1                      TBSPART01
T_R_P3            SYS_SUBP155                     TBSPART03
T_R_P3            SYS_SUBP154                     TBSPART02
T_R_P3            SYS_SUBP153                     TBSPART01
T_R_PD             T_R_P3_H3                      TBSPART03
T_R_PD             T_R_P3_H2                      TBSPART02
T_R_PD             T_R_P3_H1                      TBSPART01

已选择 9 行。
```

提示：由上两例可以看出，未显式指定子分区的分区，系统会自动创建一个子分区。

③ 分区模板的应用

oracle 还提供了一种称为分区模板的功能，在指定子分区信赖列之后，制订子分区的存储模板，各个分区即会按照子分区模式创建子分区，例如：

```
JSSWEB> create table t_partition_rh (id number,name varchar2(50))
  2    partition by range(id) subpartition by hash(name)
  3    subpartition template (
  4      subpartition h1 tablespace tbspart01,
  5      subpartition h2 tablespace tbspart02,
  6      subpartition h3 tablespace tbspart03,
  7      subpartition h4 tablespace tbspart04)(
  8    partition t_r_p1 values less than (10) tablespace tbspart01,
  9    partition t_r_p2 values less than (20) tablespace tbspart02,
 10    partition t_r_p3 values less than (30) tablespace tbspart03,
 11    partition t_r_pd values less than (maxvalue) tablespace tbspart04);

表已创建。

JSSWEB> select partition_name,subpartition_name,tablespace_name
  2    from user_tab_subpartitions where table_name='T_PARTITION_RH';

PARTITION_NAME    SUBPARTITION_NAME                TABLESPACE_NAME
--------------- ------------------------------ --------------------
T_R_P1            T_R_P1_H4                      TBSPART01
T_R_P1            T_R_P1_H3                      TBSPART01
T_R_P1            T_R_P1_H2                      TBSPART01
T_R_P1            T_R_P1_H1                      TBSPART01
```

```
T_R_P2          T_R_P2_H4                       TBSPART02
T_R_P2          T_R_P2_H3                       TBSPART02
T_R_P2          T_R_P2_H2                       TBSPART02
T_R_P2          T_R_P2_H1                       TBSPART02
T_R_P3          T_R_P3_H4                       TBSPART03
T_R_P3          T_R_P3_H3                       TBSPART03
T_R_P3          T_R_P3_H2                       TBSPART03
T_R_P3          T_R_P3_H1                       TBSPART03
T_R_PD          T_R_PD_H4                        TBSPART04
T_R_PD          T_R_PD_H3                        TBSPART04
T_R_PD          T_R_PD_H2                        TBSPART04
T_R_PD          T_R_PD_H1                        TBSPART04

已选择 16 行。
```

5、创建 range-list 组合分区

Range-list 组合分区的创建与 range-hash 极为相似，只是子分区为 list 分区，当然同样也可以应用分区模板，下面也举一个示例：

```
JSSWEB> create table t_partition_rl (id number,name varchar2(50))
  2  partition by range(id) subpartition by list(name)
  3  subpartition template (
  4    subpartition l1 values ('aa') tablespace tbspart01,
  5    subpartition l2 values ('bb') tablespace tbspart02,
  6    subpartition l3 values ('cc') tablespace tbspart03,
  7    subpartition l4 values ('dd') tablespace tbspart04)(
  8  partition t_r_p1 values less than (10) tablespace tbspart01,
  9  partition t_r_p2 values less than (20) tablespace tbspart02,
 10  partition t_r_p3 values less than (30) tablespace tbspart03,
 11  partition t_r_pd values less than (maxvalue) tablespace tbspart04);

表已创建。

JSSWEB> select partition_name,subpartition_name,tablespace_name
  2  from user_tab_subpartitions where table_name='T_PARTITION_RL';

PARTITION_NAME    SUBPARTITION_NAME                TABLESPACE_NAME
--------------- ------------------------------ --------------------
T_R_P1          T_R_P1_L4                       TBSPART01
T_R_P1          T_R_P1_L3                       TBSPART01
T_R_P1          T_R_P1_L2                       TBSPART01
T_R_P1          T_R_P1_L1                       TBSPART01
T_R_P2          T_R_P2_L4                       TBSPART02
```

```
T_R_P2          T_R_P2_L3                    TBSPART02
T_R_P2          T_R_P2_L2                    TBSPART02
T_R_P2          T_R_P2_L1                    TBSPART02
T_R_P3          T_R_P3_L4                    TBSPART03
T_R_P3          T_R_P3_L3                    TBSPART03
T_R_P3          T_R_P3_L2                    TBSPART03
T_R_P3          T_R_P3_L1                    TBSPART03
T_R_PD          T_R_PD_L4                    TBSPART04
T_R_PD          T_R_PD_L3                    TBSPART04
T_R_PD          T_R_PD_L2                    TBSPART04
T_R_PD          T_R_PD_L1                    TBSPART04

已选择 16 行。
```

其它方式的创建对于 range-list 同样好使，这里不再举例，如有不明，请自学复习前章 range_hash 组合分区。

对于复合分区的 local 索引，我们也举一个示例，查看其分区情况：

```
SQL> create index idx_part_rl_id on t_partition_rl(id) local;

索引已创建。
```

又可以学几个数据字典：user_part_indexes、user_ind_partitions 前面已经认识了，user_ind_subpartitions 用来查询索引的子分区信息。

```
SQL> select table_name,partitioning_type,
  2          partition_count,def_subpartition_count
  3    from user_part_indexes
  4   where index_name = 'IDX_PART_RL_ID';

TABLE_NAME                     PARTITI PARTITION_COUNT DEF_SUBPARTITION_COUNT
------------------------------ ------- --------------- ----------------------
T_PARTITION_RL                 RANGE                 4                      4

SQL> select partition_name, subpartition_count, high_value
  2    from user_ind_partitions
  3   where index_name = 'IDX_PART_RL_ID';

PARTITION_NAME    SUBPARTITION_COUNT HIGH_VALUE
--------------- ------------------ ---------------
T_R_P1                             4 10
T_R_P2                             4 20
T_R_P3                             4 30
T_R_PD                             4 MAXVALUE

SQL> select partition_name, subpartition_name, high_value, tablespace_name
```

```
  2    from user_ind_subpartitions
  3   where index_name = 'IDX_PART_RL_ID';

PARTITION_NAME    SUBPARTITION_NAME                HIGH_VALUE
TABLESPACE_NAME
--------------- ------------------------------ --------------- --------------------
T_R_P1            T_R_P1_L1                      'aa'            TBSPART01
T_R_P1            T_R_P1_L2                      'bb'            TBSPART01
T_R_P1            T_R_P1_L3                      'cc'            TBSPART01
T_R_P1            T_R_P1_L4                      'dd'            TBSPART01
T_R_P2            T_R_P2_L1                      'aa'            TBSPART02
T_R_P2            T_R_P2_L2                      'bb'            TBSPART02
T_R_P2            T_R_P2_L3                      'cc'            TBSPART02
T_R_P2            T_R_P2_L4                      'dd'            TBSPART02
T_R_P3            T_R_P3_L1                      'aa'            TBSPART03
T_R_P3            T_R_P3_L2                      'bb'            TBSPART03
T_R_P3            T_R_P3_L3                      'cc'            TBSPART03
T_R_P3            T_R_P3_L4                      'dd'            TBSPART03
T_R_PD            T_R_PD_L1                      'aa'            TBSPART04
T_R_PD            T_R_PD_L2                      'bb'            TBSPART04
T_R_PD            T_R_PD_L3                      'cc'            TBSPART04
T_R_PD            T_R_PD_L4                      'dd'            TBSPART04

已选择 16 行。
```

还是与表的分区格式一样，不管是普通分区还是复合分区，local 索引都没啥自主权啊。

6、公共准则

1、如果选择的分区不能确保各分区内记录量的基本平均，则这种分区方式有可能是不恰当的。

比如对于 range 分区，假设分了 10 个分区，而其中一个分区中的记录数占总记录数的 90%，其它 9 个分区只占总记录数的 10%，则这个分区方式就起不到数据平衡的作用。当然，如果你的目的并不是为了平衡，只是为了区分数据，ok，对于这种情况，我想说的是，你务必要意识到存在这个问题。

2、对于分区的表或索引，其所涉及的所有分区，其块大小必须一致。

最后，建议对于上面创建的表或建表脚本妥善保存并记忆，后面我们需要频繁用到，后续示例将均主要依赖前文中创建的表进行:)

二、如何管理

对于分区的表的操作很多，其中某些操作仅针对某些分区有效，为了避免在演示过程中浪费过多口水标注哪些操作适用于哪些分区，咱们先在这儿列个表，哪个操作适用于哪种分区格式具体可以先参考下面这个表格:

分区表	Range	List	Hash	Range-Hash	Range-List	是否带来IO操作

增加分区 (add partition)	支持	支持	支持	支持	支持	除hash类型外，均不变带来大量IO
收缩分区 (coalesce partitions)	/	/	支持	分区：/ 子分区：支持	/	是
删除分区 (drop partition)	支持	支持	/	分区：支持 子分区：/	支持	无
交换分区 (exchange partition)	支持	支持	支持	支持	支持	无
合并分区 (merge partition)	支持	支持	/	分区：支持 子分区：/	支持	是
修改默认属性 (modify default attributes)	支持	支持	支持	支持	支持	无
修改分区当前属性 (modify partition)	支持	支持	支持	支持	支持	无
List分区增加值 (modify partition add values)	/	支持	/	/	分区：/ 子分区：支持	无
List分区删除值 (modify partition drop values)	/	支持	/	/	分区：/ 子分区：支持	单纯删除操作无，但可能为了实现成功删除，之前的准备操作会带来一定量的IO
修改子分区模板 (set subpartition template)	/	/	/	支持	支持	无
移动分区 (move partition)	支持	支持	支持	分区：支持 子分区：/	分区：支持 子分区：/	有
重命名分区 (rename partition)	支持	支持	支持	支持	支持	无
拆分分区 (split partition)	支持	支持	/	分区：支持 子分区：/	支持	有
截断分区 (truncate partition)	支持	支持	支持	支持	支持	无

注：上述 IO 列的评估建立在假设分区中均存在一定量数据，并忽略修改数据字典可能触发的 IO，忽略造成的索引的重编译带来的 IO。

分区索引的操作也有一张表黑黑，如下：

分区索引	索引类型	Range	List	Hash	组合分区	是否带来IO操作
增加分区 (add partition)	全局	/	/	支持	/	是
	本地	/	/	/	/	
删除分区 (drop partition)	全局	支持	/	/	/	无
	本地	/	/	/	/	
修改默认属性 (modify default attributes)	全局	支持	/	/	/	无
	本地	支持	支持	支持	支持	无
修改分区当前属性	全局	支持	/	/	/	无

(modify partition)	本地	支持	支持	支持	支持	无
重编译分区	全局	支持	/	/	/	有
(rebuild partition)	本地	支持	支持	支持	支持	有
重命名分区	全局	支持	/	/	/	无
(rename partition)	本地	支持	支持	支持	支持	无
拆分分区	全局	支持	/	/	/	有
(split partition)	本地	/	/	/	/	

另外 local 索引前头我们多次提到了，其维护会在 oracle 操作表分区的时候自动进行，需要注意的是 global 索引，当 global 索引所在表执行 alter table 涉及下列操作时，会导至该索引失效：

- ADD PARTITION | SUBPARTITION
- COALESCE PARTITION | SUBPARTITION
- DROP PARTITION | SUBPARTITION
- EXCHANGE PARTITION | SUBPARTITION
- MERGE PARTITION | SUBPARTITION
- MOVE PARTITION | SUBPARTITION
- SPLIT PARTITION | SUBPARTITION
- TRUNCATE PARTITION | SUBPARTITION

因此，建议用户在执行上述操作 sql 语句后附加 update indexes 子句，oracle 即会自动维护全局索引，当然，需要注意这中间有一个平衡，你要平衡操作 ddl 的时间和重建索引哪个时间更少，以决定是否需要附加 update indexes 子句。

分区表的管理

1、增加表分区(add partition)

增加表分区适应于所有的分区形式，其语法是 alter table tbname add partition .....

但是，需要注意对于像 list,range 这种存在范围值的分区，所要增加的分区值必须要大于当前分区中的最大值（如果当前存在 maxvalue 或 default 的分区，add partition 会报错，这种情况只能使用 split，后面会讲到），hash 分区则无此限制。

例如：

```
JSSWEB> create table t_partition_range (id number,name varchar2(50))
  2     partition by range(id)(
  3     partition t_range_p1 values less than (10) tablespace tbspart01,
  4     partition t_range_p2 values less than (20) tablespace tbspart02,
  5     partition t_range_p3 values less than (30) tablespace tbspart03
  6     );

表已创建。

JSSWEB> alter table t_partition_range
  2     add partition t_range_p4 values less than(40);

表已更改。
```

Hash 和 list 的语法与上类似，这里不再举例。

注意：

1、对于 hash 分区，当你执行 add partition 操作的时候，oracle 会自动选择一个分区，并重新分配部分记录到新建的分区，这也意味着有可能带来一些 IO 操作。

2、执行 alter table 时未指定 update indexes 子句：

如果是 range/list 分区，其 local 索引和 global 索引不会受影响;

如果是 hash 分区，新加分区及有数据移动的分区的 local 索引和 glocal 索引会被置为 unuseable，需要重新编译。

3、复合分区完全适用上述所述规则。

2、收缩表分区(coalesce partitions)

Coalesce partition 是个很有意思的分区功能，仅能被应用于 hash 分区或复合分区的 hash 子分区，执行之后，会自动收缩当前的表分区，比如某表当前有 5 个 hash 分区，执行 alter table tbname coalesce partitions 后就变成 4 个，再执行一次就变成 3 个，再执行一次就变 2 个，再执行一次就.....就报错了：)，对于已分区的表至少要有一个分区存在的嘛！

例如：

```
JSSWEB> select table_name,partition_name from user_tab_partitions
  2     where table_name='T_PARTITION_HASH';

TABLE_NAME                     PARTITION_NAME
------------------------------ ------------------------------
T_PARTITION_HASH               T_HASH_P2
T_PARTITION_HASH               T_HASH_P3
T_PARTITION_HASH               T_HASH_P4
T_PARTITION_HASH               T_HASH_P5
T_PARTITION_HASH               T_HASH_P1

JSSWEB> alter table t_partition_hash coalesce partition;

表已更改。

JSSWEB> select table_name,partition_name from user_tab_partitions
  2     where table_name='T_PARTITION_HASH';

TABLE_NAME                     PARTITION_NAME
------------------------------ ------------------------------
T_PARTITION_HASH               T_HASH_P2
T_PARTITION_HASH               T_HASH_P3
T_PARTITION_HASH               T_HASH_P4
T_PARTITION_HASH               T_HASH_P1
```

注意，收缩的只是分区，并不会影响到数据，但是视被收缩分区中数据的多少，收缩表分区也会涉及到 IO 操作。

另外如果你在执行该语句时没有指定 update indexes 子句，收缩过程中有数据改动的分区其 local 索引和 glocal 索引都会失效，需要重新编译。

3、删除表分区(drop partition)

删除表分区包含两种操作，分别是：

- 删除分区：alter table [tbname] drop partition [ptname];
- 删除子分区：alter table [tbname] drop subpartition [ptname];

除 hash 分区和 hash 子分区外，其它的分区格式都可以支持这项操作。

例如，删除分区：

```
JSSWEB> select table_name,partition_name
  2    from user_tab_partitions where table_name='T_PARTITION_LIST';

TABLE_NAME                     PARTITION_NAME
------------------------------ ------------------------------
T_PARTITION_LIST               T_LIST_P1
T_PARTITION_LIST               T_LIST_P2
T_PARTITION_LIST               T_LIST_P3
T_PARTITION_LIST               T_LIST_PD

JSSWEB> alter table t_partition_list drop partition t_list_p2;

表已更改。
```

提示，drop partition 时，该分区内存储的数据也将同时删除，例如：

```
JSSWEB> insert into t_partition_list values (1,'a');
.....
--插入一批记录，分布于当前各个分区
.....
JSSWEB> commit;

提交完成。

JSSWEB> select *from t_partition_list;

        ID NAME
---------- --------------------------------------------------
         1 a
         2 b
        21 a
        22 b
```

```
--单独查询 t_list_p3 分区，当前有数据
JSSWEB> select *from t_partition_list partition(t_list_p3);

        ID NAME
---------- --------------------------------------------------
        21 a
        22 b

--删除 t_list_p3 分区，数据会被同时删除
JSSWEB> alter table t_partition_list drop partition t_list_p3;

表已更改。

JSSWEB> select *from t_partition_list partition(t_list_p3);
select *from t_partition_list partition(t_list_p3)
                                        *
第 1 行出现错误:
ORA-02149: 指定的分区不存在


JSSWEB> select *from t_partition_list;

        ID NAME
---------- --------------------------------------------------
         1 a
         2 b
```

由于是 ddl 操作，这种删除也会是非常迅速的，因此如果你确认某个分区的数据都要被删除，使用 drop partition 会比 delete 更加高效。如果你的本意是希望删除掉指定的分区但保留数据，你应该使用 merge partition，后面也会讲到。

同样，如果你在执行该语句时没有指定 update indexes 子句，也会导致 glocal 索引的失效，至于 local 索引嘛，删除分区时对应的索引分区会被同时删除，但其它分区的 local 索引不会受到影响。

4、交换表分区(Exchange Partitions)

直白的说就是迁移数据。迁移数据的方式很多，为什么要使用 exchange partition 的方式呢，表急，听三思慢慢道来。

Exchange partition 提供了一种方式，让你在表与表或分区与分区之间迁移数据，注意不是将表转换成分区或非分区的形式，而仅只是迁移表中数据(互相迁移)，由于其号称是采用了更改数据字典的方式，因此效率最高(几乎不涉及 io 操作)。Exchange partition 适用于所有分区格式，你可以将数据从分区表迁移到非分区表，也可以从非分区表迁移至分区表，或者从 hash partition 到 range partition 诸如此类吧。

其语法很简单：alter table tbname1 exchange partition/subpartition ptname with table tbname2;

Exchange partition 迁移的方式也很有意思，言语表达怕大家听不明白，下面直接通过示例来表达：

借用前文中创建的空分区表:t_partition_range，并插入几条记录

```
JSSWEB> create table t_partition_range (id number,name varchar2(50))
  2     partition by range(id)(
  3     partition t_range_p1 values less than (10) tablespace tbspart01,
  4     partition t_range_p2 values less than (20) tablespace tbspart02,
  5     partition t_range_p3 values less than (30) tablespace tbspart03,
  6     partition t_range_pmax values less than (maxvalue) tablespace tbspart04
  7     );

表已创建。

JSSWEB> insert into t_partition_range values (11,'a');

已创建 1 行。

JSSWEB> insert into t_partition_range values (12,'b');

已创建 1 行。

JSSWEB> insert into t_partition_range values (13,'c');

已创建 1 行。

JSSWEB> commit;

提交完成。
```

再创建一个非分区表，结构与 t_partition_range 相同

```
JSSWEB> create table t_partition_range_tmp (id number,name varchar2(50));

表已创建。
```

执行交换分区(我们知道刚插入到 range 分区表的数据都在分区 t_range_p2 中，因此这里指定交换该分区)

```
JSSWEB> alter table t_partition_range exchange partition t_range_p2
  2     with table t_partition_range_tmp;

表已更改。
```

看看效果如何：

```
JSSWEB> select * from t_partition_range partition(t_range_p2);

未选定行
```

```
JSSWEB> select * from t_partition_range_tmp;

        ID NAME
---------- --------------------------------------------------
        11 a
        12 b
        13 c
```

记录成功交换到未分区的表中。

我们再执行一次 exchange partition 的命令，看看又会发生什么呢

```
JSSWEB> select *from t_partition_range partition(t_range_p2);

        ID NAME
---------- --------------------------------------------------
        11 a
        12 b
        13 c

JSSWEB> select *from t_partition_range_tmp;

未选定行
```

又交换回来了，有点儿意思。

再做个更加明确的测试，我们往未分区的表中加入一些记录后再执行 exchange partition，看看会发生什么呢：

```
JSSWEB> insert into t_partition_range_tmp values (15,'d');

已创建 1 行。

JSSWEB> insert into t_partition_range_tmp values (16,'e');

已创建 1 行。

JSSWEB> insert into t_partition_range_tmp values (17,'d');

已创建 1 行。

JSSWEB> alter table t_partition_range exchange partition t_range_p2
  2    with table t_partition_range_tmp;

表已更改。

JSSWEB> select *from t_partition_range partition(t_range_p2);
```

```
        ID NAME
---------- --------------------------------------------------
        15 d
        16 e
        17 d

JSSWEB> select *from t_partition_range_tmp;

        ID NAME
---------- --------------------------------------------------
        11 a
        12 b
        13 c
```

这就是前面所说的，互相交换的意思~~

注意：

- 涉及交换的两表之间表结构必须一致，除非附加 with validation 子句;
- 如果是从非分区表向分区表做交换，非分区表中的数据必须符合分区表中指定分区的规则，除非附加 without validation 子句;
- 如果从分区表向分区表做交换，被交换的分区的数据必须符合分区规则，除非附加 without validation 子句;
- Global 索引或涉及到数据改动了的 global 索引分区会被置为 unusable，除非附加 update indexes 子句。

提示：

一旦附加了 without validation 子句，则表示不再验证数据有效性，因此指定该子句时务必慎重。

例如：

```
JSSWEB> insert into t_partition_range_tmp values (8,'g');

已创建 1 行。

JSSWEB> alter table t_partition_range exchange partition t_range_p2
  2    with table t_partition_range_tmp without validation;

表已更改。

JSSWEB> select *from t_partition_range partition(t_range_p2);

        ID NAME
---------- --------------------------------------------------
        11 a
        12 b
        13 c
         8 g
```

虽然新插入的记录并不符合 t_range_p2 分区的范围值，但指定了 without validation 后，数据仍然转换

成功。

5、合并表分区(Merge Partitions)

合并两个分区成一个，适用于除 hash 之外的其它所有分区形式(hash 分区有 coalesce partition 的嘛，前头刚刚讲过)。

语法很简单：alter table tbname merge partitions/subpartitions pt1,pt2 into partition/subpartition pt3;

同样也支持 update indexes 子句以避免单独执行造成索引失效的问题。

需要注意一点，要合并的两个分区必须是连续的，这点是由分区本身的特性所决定的，如例：

```
JSSWEB> alter table t_partition_range merge partitions t_range_p1,t_range_p2
  2    into partition t_range_pnew;

表已更改。

JSSWEB> select table_name,partition_name,high_value from user_tab_partitions
  2    where table_name='T_PARTITION_RANGE';

TABLE_NAME                     PARTITION_NAME                 HIGH_VALUE
------------------------------ ------------------------------ ----------
T_PARTITION_RANGE              T_RANGE_P3                     30
T_PARTITION_RANGE              T_RANGE_PMAX                     MAXVALUE
T_PARTITION_RANGE              T_RANGE_PNEW                     20

JSSWEB> select *from t_partition_range partition(t_range_pnew);

            ID NAME
---------- --------------------------------------------------
            11 a
            12 b
            13 c
             8 g
```

可见，合并分区操作不会造成数据丢失，另外如果你想为新分区指定属性的话，在语句末尾处增加存储属性即可(如果不指定，则新分区默认继续表的存储属性)。例如：

```
JSSWEB> select partition_name,high_value,tablespace_name from user_tab_partitions
  2    where table_name='T_PARTITION_LIST';

PARTITION_NAME       HIGH_VALUE                                     TABLESPACE_NAME
-------------------- ---------------------------------------------- ---------------
T_LIST_P1            1, 2, 3, 4, 5, 6, 7, 8, 9, 10                  TBSPART01
T_LIST_P2            11, 12, 13, 14, 15, 16, 17, 18, 19, 20         TBSPART02
T_LIST_P3            21, 22, 23, 24, 25, 26, 27, 28, 29, 30         TBSPART03
T_LIST_PD             default                                       TBSPART04
```

```
JSSWEB> alter table t_partition_list merge partitions t_list_p2,t_list_p3
  2    into partition t_list_p2 tablespace tbspart02;

表已更改。

JSSWEB> select partition_name,high_value,tablespace_name from user_tab_partitions
  2    where table_name='T_PARTITION_LIST';

PARTITION_NAME     HIGH_VALUE                                        TABLESPACE_NAME
-------------------- ------------------------------------------------- ---------------
T_LIST_P1            1, 2, 3, 4, 5, 6, 7, 8, 9, 10                     TBSPART01
T_LIST_P2            21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 11, 12, 13 TBSPART02
                     , 14, 15, 16, 17, 18, 19, 20
T_LIST_PD            default                                           TBSPART04
```

注意，merge 分区操作与 coalesce 分区操作一样，视被合并的分区数据量多少，都可能涉及到大量的 IO 操作。

其它合并组合分区的操作与上类似，如果要合并组合分区，注意关键字是 merge subpartitions，这里就不做演示了。

6、修改 list 表分区--Add Values

从标题即可得知，此命令仅应用于 list 分区或 list 子分区，语法也非常简单：

Alter table tbname modify partition/subpartition ptname add values (v1,v2....vn);

举个例子：

```
JSSWEB> select partition_name,high_value from user_tab_partitions
  2    where table_name='T_PARTITION_LIST';

PARTITION_NAME          HIGH_VALUE
-------------------- -------------------------------------------------
T_LIST_P1            1, 2, 3, 4, 5, 6, 7, 8, 9, 10
T_LIST_P2            21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 11, 12, 13
                     , 14, 15, 16, 17, 18, 19, 20
T_LIST_PD            default

JSSWEB> alter table t_partition_list modify partition t_list_p1 add values (31,33);

表已更改。

JSSWEB> select partition_name,high_value from user_tab_partitions
  2    where table_name='T_PARTITION_LIST';

PARTITION_NAME          HIGH_VALUE
```

```
------------------- --------------------------------------------------
T_LIST_P1                1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 31, 33
T_LIST_P2                21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 11, 12, 13
                        , 14, 15, 16, 17, 18, 19, 20
T_LIST_PD                default
```

唯一的限制是注意要添加的新 value 值不能存在于当前任何分区中，并且当前表也不能存在记录值为新值的记录，特别是当你创建了 default 分区的时候，有必要先检查一下当前表不存在要添加的值，不然命令执行会出错，例如：

```
JSSWEB> insert into t_partition_list values (32,'a');

已创建 1 行。

JSSWEB> alter table t_partition_list modify partition t_list_p1 add values (32);
alter table t_partition_list modify partition t_list_p1 add values (32)
                                                          *
第 1 行出现错误:
ORA-14324: 所要添加的值已存在于 DEFAULT 分区之中
```

提示，增加新的列表值不会影响到表中原有的记录，因此不会对索引造成影响。

7、修改 list 表分区--Drop Values

与上类似，也是只能应用于 list 分区，不过功能相反，该命令是用来删除指定分区的 value 值，语法如下：

Alter table tbname modify partition/subpartition ptname drop values (v1,v2....vn);

同样在删除 list 分区 value 列值的时候，也必须确认当前分区存在指定的 value 值，但是没有任何应用该值的记录，有点儿饶是吧，脑袋多转几圈就好了。

举个例子：

```
JSSWEB> alter table t_partition_list modify partition t_list_p1 drop values (31);

表已更改。
```

成功执行了是吧，接着来看

```
JSSWEB> alter table t_partition_list modify partition t_list_p1 drop values (31);
alter table t_partition_list modify partition t_list_p1 drop values (31)
*
第 1 行出现错误:
ORA-14313: 值 31 不在分区 T_LIST_P1 中
```

出错了吧，这是其中的一种错误情形，即前面说的，要确保当前分区中存在指定的 value 值，再往下看

```
JSSWEB> alter table t_partition_list modify partition t_list_p1 add values (31);
```

```
表已更改。

JSSWEB> insert into t_partition_list values (31,'b');

已创建 1 行。

JSSWEB> alter table t_partition_list modify partition t_list_p1 drop values (31);
alter table t_partition_list modify partition t_list_p1 drop values (31)
                                *
第 1 行出现错误:
ORA-14518: 分区包含的某些行对应于已删除的值
```

这是另外的一种错误情形，即要确保拆分分区的记录中，没有应用了指定 value 值的记录。

8、拆分表分区(Split Partition)

如果你对我们前面讲到过的 merge partition 还有印象的话，那么学习 Split partition 也不会遇到什么障碍，split partition 的功能与 merge partition 功能正好相反：后者是将两个全区合并成一个，前者则是将一个分区拆分成两个。其用途非常广泛，比如通常见你发现某个分区过大，你就可以通过这种方式将该分区分解成多个小分区，对我而言最常用到的，当然还是 split maxvalue/default 的分区。

该命令的语法针对不同分区会有不同的形式，

- For range partition： alter table tbname split partition ptname at (value) into (partition newpt1 tbs_clause,partition newpt2 tbs_clause);
- For list partition : alter table tbname split partition ptname values (v1,v2...vn) into (partition newpt1 tbs_clause,partition newpt2 tbs_clause);

上述两项，如果是操作子分区，则将 partition 关键字换成 subpartition 即可。旧分区中符合新定义值的记录会存储到指定的第一个分区中，其它的记录存储到第二个分区。

例如,range 分区的示例：

```
JSSWEB> select partition_name,high_value,tablespace_name from user_tab_partitions
  2  where table_name='T_PARTITION_RANGE';

PARTITION_NAME          HIGH_VALUE                              TABLESPACE_NAME
------------------ ---------------------------------------- ------------------
T_RANGE_P3              30                                      TBSPART03
T_RANGE_PMAX              MAXVALUE                                    TBSPART04
T_RANGE_P1              20                                      TBSPART02
```

我们将 t_range_p1 分区分隔到两个分区中，小于 10 的存放新建分区 t_range_p1(已非原 t_range_p1 鸟，只是名称相同而已)，其它数据存入 t_range_p2 分区：

```
JSSWEB> alter table t_partition_range split partition t_range_p1 at (10) into
  2  (partition t_range_p1 tablespace tbspart01,
  3  partition t_range_p2 tablespace tbspart02);
```

```
表已更改。

JSSWEB> select partition_name,high_value,tablespace_name from user_tab_partitions
  2    where table_name='T_PARTITION_RANGE';

PARTITION_NAME          HIGH_VALUE                                TABLESPACE_NAME
------------------- ---------------------------------------- --------------------
T_RANGE_P3              30                                    TBSPART03
T_RANGE_PMAX             MAXVALUE                                  TBSPART04
T_RANGE_P1              10                                    TBSPART01
T_RANGE_P2              20                                    TBSPART02
```

再来演示一个 list 分区的例子：

```
JSSWEB> select partition_name,high_value,tablespace_name from user_tab_partitions
  2    where table_name='T_PARTITION_LIST';

PARTITION_NAME          HIGH_VALUE                                TABLESPACE_NAME
------------------- ---------------------------------------- --------------------
T_LIST_P1               1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 33         TBSPART01
T_LIST_P2               21, 22, 23, 24, 25, 26, 27, 28, 29, 30,   TBSPART02
                        11, 12, 13, 14, 15, 16, 17, 18, 19, 20
T_LIST_PD                default                                       TBSPART04
```

我们将 t_list_p2 分区中分区值是 2 打头的存储到 t_list_p3 分区中，其它值存储到 t_list_p2 分区：

```
JSSWEB> alter table t_partition_list split partition t_list_p2 values
  2    (20,21,22,23,24,25,26,27,28,29) into
  3    (partition t_list_p3 tablespace tbspart03,
  4    partition t_list_p2);

表已更改。

JSSWEB> select partition_name,high_value,tablespace_name from user_tab_partitions
  2    where table_name='T_PARTITION_LIST';

PARTITION_NAME          HIGH_VALUE                                TABLESPACE_NAME
------------------- ---------------------------------------- --------------------
T_LIST_P1               1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 33         TBSPART01
T_LIST_P2               30, 11, 12, 13, 14, 15, 16, 17, 18, 19    TBSPART02
T_LIST_PD                default                                       TBSPART04
T_LIST_P3               20, 21, 22, 23, 24, 25, 26, 27, 28, 29    TBSPART03
```

提示：

- split partition/subpartition 不能用于 hash 分区或 hash 子分区(hash 的话，直接用 add partition 就好了)
- split partition/subpartition 视被分隔的分区数据量多少，可能需要花费不小的代价，相当于该分区数

据的全扫描，我们也许可以形容为：full partition scan:)，除非：

- Split 后的两个分区中，至少有一个是空的，并且非空的那个分区的存储属性与 split 前的存储属性完全相同
- 如果 split 的分区包含 lob 字段，split 后非空的那个分区中该字段的存储属性也必须与 split 前的存储属性完全相同。

这种情况下的 split partition/subpartition 也会非常高效，oracle 会自动进行优化，此时的分区操作类似于 add partition。

- 通常情况下，如果在执行 split partition/subpartition 时，如果没有指定 update indexes 子句，都会造成 local 和 global 索引的失效。注意，我们说的是通常，如果你 split partition/subpartition 的是个空分区，或者没有触发任何数据移动或变化，那么即使不加 update indexes，也不会影响到索引。当然，保险起见，建议你还是执行完之后，查询一下数据字典，确认一下当前索引的状态。

9、截断表分区(Truncate Partition)

Truncate partition 就像 truncate table 一样，直接从头部截断数据，用来删除数据那是效率超高无比。但是如果该表有外键引用的话，ddl 的 truncate 就不好使了，这时候你只能要么使用 delete，要么先 disable 掉外键关联再 truncate 了。同样，在不指定 update indexes 子句的情况下，truncate partition 也会造成分区所在表的 global 索引失效。

语法非常简单：alter table tbname truncate partition/subpartition ptname;

例如：

```
JSSWEB> select *from t_partition_range partition(t_range_p1);

        ID NAME
---------- --------------------
        11 a
        12 b
        13 c

JSSWEB> alter table t_partition_range truncate partition t_range_p1;

表被截断。

JSSWEB> select *from t_partition_range partition(t_range_p1);

未选定行
```

10、移动表分区(Move Partition)

Move partition 与 modify partition 的功能相似，但又比之更加强劲，比如可以修改分区所在表空间等等，与 move table 的操作很类似，某些时间也非常有用，比如降低行迁移。语法很简单：

Alter table tbname move partition/subpartition ptname .....;

例如：

```
JSSWEB> select partition_name,tablespace_name from user_tab_partitions
```

```
  2    where table_name='T_PARTITION_RANGE';

PARTITION_NAME          TABLESPACE_NAME
------------------- ---------------
T_RANGE_P3              TBSPART03
T_RANGE_PMAX              TBSPART04
T_RANGE_P1              WEBTBS

JSSWEB> alter table t_partition_range move partition t_range_p1 tablespace tbspart02;

表已更改。

JSSWEB> select partition_name,tablespace_name from user_tab_partitions
  2    where table_name='T_PARTITION_RANGE';

PARTITION_NAME          TABLESPACE_NAME
------------------- ---------------
T_RANGE_P3              TBSPART03
T_RANGE_PMAX              TBSPART04
T_RANGE_P1              TBSPART02
```

提示：move partition/subpartiton 时会锁表，并且 move partition/subpartiton 视被移动分区中数据量的多少，会带来相应的 IO 操作。同时还需要注意，如果在 move partition/subpartiton 时没有指定 update indexes 子句，则被移动分区所在的 local 索引以及全局索引都会失效，需要手工 rebuilding。

11、重命名表分区(Rename Partition)

就是改名，跟改表名、改列名的操作目的是类似的，语法也很简单：

Alter table tbname rename partition ptname to newptname;

举个例子：

```
JSSWEB> select partition_name from user_tab_partitions where table_name='T_PARTITION_RANGE';

PARTITION_NAME
-------------------
T_RANGE_P3
T_RANGE_PMAX
T_RANGE_PNEW

JSSWEB> alter table t_partition_range rename partition t_range_pnew to t_range_p1;

表已更改。

JSSWEB> select partition_name from user_tab_partitions where table_name='T_PARTITION_RANGE';
```

```
PARTITION_NAME
--------------------
T_RANGE_P3
T_RANGE_PMAX
T_RANGE_P1
```

12、修改表分区默认属性(Modify Default Attributes)

修改表或表中分区的存储参数，对当前表和分区的存储参数没有影响，只有修改过之后，当你下次再添加分区时，在不手工显式指定新分区参数的情况下，新分区默认使用你当前指定的存储参数。

有两种操作方式：

修改表属性，适用于 range,list,hash 分区形式(注意 hash 分区只能修改默认表空间参数)。例如：

```
JSSWEB> alter table t_partition_list modify default attributes tablespace webtbs;

表已更改。
```

修改分区属性，适用于组合分区，例如：

```
JSSWEB> alter table t_partition_rl modify default attributes for partition t_r_p2 tablespace webtbs;

表已更改。
```

13、修改表分区当前属性(Modify Partition)

与上不同，该命令修改的不是默认属性，而是分区当前的存储属性，即修改即生效的那种，虽然号称是修改当前分区属性，但实际上也有限制，比如所在表空间它就改不了(如果你想改，可以用 move partition，后面会讲到)。

存储属性呢，三思一向没有过多关注(也许是因为从未有过因此导致的惨痛教训)，此节跳过，留待有心人自行查询文档:)

14、修改表子分区模板(Set Subpartition Template)

既然是修改子分区模板，自然是只针对复合分区有效。修改分区模式不会改变当前的分区结构，只有当你再增加、合并分区并且未显式指定子分区存储参数时，才会继承新分区模板中的参数。

该命令语法很简单：alter table tbname set subpartition template ....;

下面举个例子：

```
JSSWEB> select subpartition_name,tablespace_name from user_subpartition_templates
  2    where table_name='T_PARTITION_RH';

SUBPARTITION_NAME                TABLESPACE_NAME
-------------------------------- ---------------
```

```
H1                                TBSPART01
H2                                TBSPART02
H3                                TBSPART03
H4                                TBSPART04

JSSWEB> alter table t_partition_rh
  2  set subpartition template(
  3  subpartition h1 tablespace tbspart01,
  4  subpartition h2 tablespace tbspart02,
  5  subpartition h3 tablespace tbspart03);

表已更改。

JSSWEB> select subpartition_name,tablespace_name from user_subpartition_templates
  2  where table_name='T_PARTITION_RH';

SUBPARTITION_NAME                 TABLESPACE_NAME
--------------------------------- ---------------
H1                                TBSPART01
H2                                TBSPART02
H3                                TBSPART03
```

*这里又学到一个数据字典：user_subpartition_templates，用来查询表的分区模板信息。

如果说，想清除某表的分区模板，那就更简单了：

```
JSSWEB> select subpartition_name,tablespace_name from user_subpartition_templates
  2  where table_name='T_PARTITION_RL';

SUBPARTITION_NAME                 TABLESPACE_NAME
--------------------------------- ---------------
L1                                TBSPART01
L2                                TBSPART02
L3                                TBSPART03
L4                                TBSPART04

JSSWEB> alter table t_partition_rl set subpartition template();

表已更改。

JSSWEB> select * from user_subpartition_templates where table_name='T_PARTITION_RL';

未选定行
```

分区索引的管理

1、增加索引分区(Adding Index Partitions)

从语法上来讲，增加索引分区与增加表分区没有什么实际性差别，将 table 换成 index 即可：

Alter index idxname add partition ptname tbs_clause;

需要注意一点 add partition 只能用于 hash 的 global 索引(如果你想为 range 类型的索引增加分区，不要用 add,split 也许能帮你实际你的需求)，并且 add partition 无法新增 local 索引分区，因为 local 分区是由索引所在基表来维护的。

下面举个操作的例子，还记的我们前面演示创建 hash 分区的时候创建的索引吗，这里就以它为例吧：

```
JSSWEB> select partition_name,tablespace_name from user_ind_partitions
  2    where index_name='IDX_PART_HASH_ID';

PARTITION_NAME                 TABLESPACE_NAME
------------------------------ ------------------------------
SYS_P113                       TBSPART01
SYS_P114                       TBSPART02
SYS_P115                       TBSPART03

JSSWEB> alter index idx_part_hash_id add partition i_hash_id_p4 tablespace tbspart04;

索引已更改。

JSSWEB> select partition_name,tablespace_name from user_ind_partitions
  2    where index_name='IDX_PART_HASH_ID';

PARTITION_NAME                 TABLESPACE_NAME
------------------------------ ------------------------------
I_HASH_ID_P4                   TBSPART04
SYS_P113                       TBSPART01
SYS_P114                       TBSPART02
SYS_P115                       TBSPART03
```

看看，就是这么简单。

2、删除索引分区(Dropping Index Partitions)

Drop partition 只能操作 global 索引的 range 分区，语法也很简单：

Alter index idxname drop partition ptname;

看起来很简单对吧，但是，需要注意，索引必须拥有一个 maxvalue 的分区，该分区无法删除。

另外，如果删除的索引分区中包含数据，分区被删除后，会造成相邻的 higher 分区失效，需要手工编译！这个其实很容易理解，索引中数据都是经过排序的，我们 drop partition 删除的只是分区，但其对应的索引数据还需要有地儿存在行啊(不然索引启不就不准确了)，于是就只好存储到比它更高区间值的索引区分里去了，那个分区莫名其妙多了数据，自然状态就为不可用了。

举个例子：

```
JSSWEB> select partition_name,high_value,tablespace_name,status from user_ind_partitions
  2    where index_name='IDX_PART_RANGE_ID';

PARTITION_NAME          HIGH_VALUE         TABLESPACE_NAME          STATUS
------------------- --------------- -------------------- --------
I_RANGE_P1              10                 TBSPART01              USABLE
I_RANGE_P2              40                 TBSPART02              USABLE
I_RANGE_PMAX            MAXVALUE           TBSPART03              USABLE
```

向表中插入几条记录：

```
JSSWEB> insert into t_partition_range values (8,'a');

已创建 1 行。

JSSWEB> insert into t_partition_range values (9,'b');

已创建 1 行。

JSSWEB> commit;

提交完成。
```

执行删除操作

```
JSSWEB> alter index idx_part_range_id drop partition i_range_p1;

索引已更改。

JSSWEB> select partition_name,high_value,tablespace_name,status from user_ind_partitions
  2    where index_name='IDX_PART_RANGE_ID';

PARTITION_NAME          HIGH_VALUE         TABLESPACE_NAME          STATUS
------------------- --------------- -------------------- --------
I_RANGE_P2              40                 TBSPART02              UNUSABLE
I_RANGE_PMAX            MAXVALUE           TBSPART03              USABLE
```

于是，i_range_p2 分区就 unusable 了，继续往下看吧，后面要讲如何重编译索引分区了。

3、重编译索引分区(Rebuilding Index Partitions)

一生不如意，十有八九。碰上索引分区无效也不见得就是撞头彩的运气，这个东西还是黑常见的，比如分区表操作时未指定 update indexes 子句就极有可能造成索引分区的无效，一般情况下，你都可以通过：

Alter index idxname rebuild partition/subpartition ptname;

重新编译。注意 global 索引只支持 range 分区，local 索引无限制。

例如：

```
JSSWEB> alter index idx_part_range_id rebuild partition i_range_p2;

索引已更改。

JSSWEB> select partition_name,high_value,tablespace_name,status from user_ind_partitions
  2  where index_name='IDX_PART_RANGE_ID';

PARTITION_NAME          HIGH_VALUE        TABLESPACE_NAME         STATUS
-------------------- --------------- -------------------- --------
I_RANGE_P2              40                TBSPART02               USABLE
I_RANGE_PMAX            MAXVALUE          TBSPART03               USABLE
```

而对于 local 索引分区，你还可以使用这种命令方式：
alter table tbname modify partition/subpartition ptname rebuild unusable local indexes;

4、重命名索引分区(Renaming Index Partitions)

与表分区中改名功能相同，索引分区重命名也仅只是改个名字而已，语法非常简单：
Alter index idxname rename partition/subpartition ptname to ptnewname;

这个功能没啥可说的，使用也很简单：

```
JSSWEB> alter index idx_part_range_id rename partition i_range_p2 to i_range_p1;

索引已更改。
```

同样需要注意，global 分区只能够支持 range 分区，local 索引无限制。

5、分拆索引分区(Splitting Index Partitions)

Split partiton 操作只能操作 global 索引分区(local 分区会自动维护)，且只能操作 global 索引分区中 range 类型的分区。

语法与表分区的操作很类似：
Alter index idxname split partition ptname at(value) into(partition pt1 tbsclause,partition pt2 tbsclause);

看个例子：

```
JSSWEB> alter index idx_part_range_id split partition i_range_p1 at (10) into
  2  (partition i_range_p1 tablespace tbspart01,
  3  partition i_range_p2 tablespace tbspart02);

索引已更改。

JSSWEB> select partition_name,high_value,tablespace_name,status from user_ind_partitions
  2  where index_name='IDX_PART_RANGE_ID';

PARTITION_NAME          HIGH_VALUE        TABLESPACE_NAME         STATUS
```

```
-------------------- --------------- -------------------- --------
I_RANGE_P1           10              TBSPART01            USABLE
I_RANGE_PMAX           MAXVALUE          TBSPART03            USABLE
I_RANGE_P2           40              TBSPART02            USABLE
```

6、修改索引分区默认属性(Modifying Default Attributes of Index Partitions)

修改索引分区默认属性，与修改表分区的操作没什么区别，不过对于 global 索引，你只能修改 range 分区，local 索引则无此限制。

语法上小有差异:Alter index idxname modify default attributes for partition ptname;

不做演示！

7、修改索引分区当前属性(Modifying Real Attributes of Index Partitions)

同样 global 索引只支持 range 分区的修改，支持所有 local 索引，其它与表分区修改同理，不做演示！

全篇完结之后记

原计划还想花重量篇幅多介绍一些实践，比如分区表的常见使用方式，对于效率提升的比较，不同情况下，使用 global 索引与 local 索引的区别，什么情况下会造成索引失效，什么时候需要重新编译等待~~~待到行笔处却发现千言万语，又不知从何处开始。

越学越觉着 oracle 博大精深，确实不是盖的，小小一个分区就能引出这么多文章，还无法一一道进。因此，原计划起的“深入学习分区表分区表及分区索引”写到最后也是越写也没自信，越写越觉着我所了解到的和我所能介绍的不过都是皮毛罢了，因此，改名吧。。。“全面认识 oracle 分区表及分区索引”就此登场~

本系列全文已打包处理为 pdf，并上传至 pub 论坛：http://www.itpub.net/996554.html，供有心者参阅方便~~

全文不过一家之言，虽在成文之前也多有参考前辈们的精华，但未免仍有纰漏，如有问题，欢迎大家就此与我沟通，交流。

我常在的 QQ 群：59666289，欢迎大家加入讨论:)

[三思笔记]单条 SQL 语句实现复杂逻辑几例
http://www.itpub.net/958526.html

[三思笔记]一步一步学 Dataguard
http://www.itpub.net/958526.html

[三思笔记]使用可传输表空间的特性复制数据
http://www.itpub.net/926949.html

[三思笔记]日期时间及数字的格式化参数大全
http://www.itpub.net/913307.html

[三思笔记]RMAN 高级应用之 Duplicate 复制数据库
http://www.itpub.net/906598.html

[三思笔记]RHEL AS4 下升级 oracle10g 到 10.2.0.3
http://www.itpub.net/896394.html

[三思笔记]RHEL AS4 下安装 32 位 oracle10g
http://www.itpub.net/884137.html

[三思笔记]Statspack 初步学和用
http://www.itpub.net/857807.html

[三思笔记]oracle 著名及非著名函数介绍
http://www.itpub.net/843333.html

[三思笔记]一步一步学 rman
http://www.itpub.net/810100.html

[三思笔记]学习动态性能表
http://www.itpub.net/782892.html