

oracle 表分区详解

从以下几个方面来整理关于分区表的概念及操作：

1. 表空间及分区表的概念
2. 表分区的具体作用
3. 表分区的优缺点
4. 表分区的几种类型及操作方法
5. 对表分区的维护性操作

1.表空间及分区表的概念

表空间：

是一个或多个数据文件的集合，所有的数据对象都存放在指定的表空间中，但主要存放的是表， 所以称作表空间。

分区表：

当表中的数据量不断增大，查询数据的速度就会变慢，应用程序的性能就会下降，这时就应该考虑对表进行分区。表进行分区后，逻辑上表仍然是一张完整的表，只是将表中的数据在物理上存放到多个表空间(物理文件上)，这样查询数据时，不至于每次都扫描整张表。

2.表分区的具体作用

Oracle 的表分区功能通过改善可管理性、性能和可用性，从而为各式应用程序带来了极大的好处。通常，分区可以使某些查询以及维护操作的性能大大提高。此外,分区还可以极大简化常见的管理任务，分区是构建千兆字节数据系统或超高可用性系统的关键工具。

分区功能能够**将表、索引或索引组织表进一步细分为段**，这些数据库对象的段叫做分区。每个分区有自己的**名称**，还可以选择自己的**存储特性**。从数据库 管理员的角度来看，一个分区后的对象具有多个段，这些段既可进行集体管理，也可单独管理，这就使数据库管理员在管理分区后的对象时有相当大的灵活性。但 是，**从应用程序的角度来看，分区后的表与非分区表完全相同，使用 SQL DML 命令访问分区后的表时，无需任何修改。**

什么时候使用分区表：

- 1) **表的大小超过 2GB。**
- 2) **表中包含历史数据，新的数据被增加都新的分区中。**

3.表分区的优缺点

优点：

- 1) **改善查询性能**：对分区对象的查询可以仅搜索自己关心的分区，提高检索速度。
- 2) **增强可用性**：如果表的某个分区出现故障，表在其他分区的数据仍然可用；
- 3) **维护方便**：如果表的某个分区出现故障，需要修复数据，只修复该分区即可；
- 4) **均衡 I/O**：可以**把不同的分区映射到磁盘以平衡 I/O**，改善整个系统性能。

缺点：

分区表相关，**已经存在的表没有方法可以直接转化为分区表**。不过 Oracle 提供了**在线重定义表**的功能。

4.表分区的几种类型及操作方法

1.范围分区

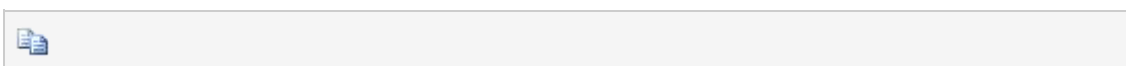
范围分区将数据基于范围映射到每一个分区，这个范围是你在创建分区时指定的分区键决定的。这种分区方式是最为常用的，并且分区键经常采用日期。举个例子：你可能会将销售数据按照月份进行分区。

当使用范围分区时，请考虑以下几个规则：

- 1) 每一个分区都必须有一个 **VALUES LESS THEN** 子句，它指定了一个**不包括在该分区中的上限值**。分区键的任何值**等于或者大于**这个上限值的记录都会被加入到下一个高一些的分区中。
- 2) 所有分区，除了第一个，都会有一个**隐式的下限值**，这个值就是**此分区的前一个分区**的上限值。
- 3) 在最高的分区中，**MAXVALUE** 被定义。**MAXVALUE** 代表了一个不确定的值。这个值高于其它分区中的任何分区键的值，也可以理解为高于任何分区中指定的 **VALUE LESS THEN** 的值，同时**包括空值**。

例 1：

假设有一个 **CUSTOMER** 表，表中有数据 200000 行，我们将此表通过 **CUSTOMER_ID** 进行分区，每个分区存储 100000 行，我们将每个分区保存到单独的表空间中，这样数据文件就可以跨越多个物理磁盘。下面是创建表和分区的代码，如下：



```
CREATE TABLE CUSTOMER

(

    CUSTOMER_ID NUMBER NOT NULL PRIMARY KEY,

    FIRST_NAME VARCHAR2 (30) NOT NULL,

    LAST_NAME VARCHAR2 (30) NOT NULL,

    PHONE VARCHAR2 (15) NOT NULL,

    EMAIL VARCHAR2 (80),

    STATUS CHAR (1)

)

PARTITION BY RANGE (CUSTOMER_ID)

(

    PARTITION CUS_PART1 VALUES LESS THAN (100000) TABLESPACE CUS_TS01,
```

```
    PARTITION CUS_PART2 VALUES LESS THAN (200000) TABLESPACE CUS_TS02

)
```



例 2：按时间划分



```
CREATE TABLE ORDER_ACTIVITIES
```

```
(
```

```
    ORDER_ID    NUMBER(7) NOT NULL,
```

```
    ORDER_DATE  DATE,
```

```
    TOTAL_AMOUNT NUMBER,
```

```
    CUSTOTMER_ID NUMBER(7),
```

```
    PAID        CHAR(1)
```

```
)
```

```
    PARTITION BY RANGE (ORDER_DATE)
```

```
(
```

```
    PARTITION ORD_ACT_PART01 VALUES LESS THAN (TO_DATE('01-MAY-2003','DD-MON-YYYY')) TABLESPACE ORD_TS01,
```

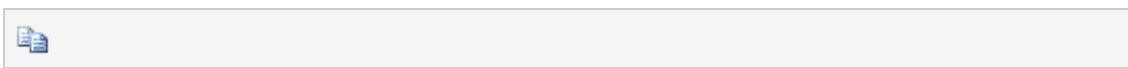
```
    PARTITION ORD_ACT_PART02 VALUES LESS THAN (TO_DATE('01-JUN-2003','DD-MON-YYYY')) TABLESPACE ORD_TS02,
```

```
    PARTITION ORD_ACT_PART02 VALUES LESS THAN (TO_DATE('01-JUL-2003','DD-MON-YYYY')) TABLESPACE ORD_TS03
```

)



例 3: MAXVALUE



```
CREATE TABLE RangeTable
```

```
(
```

```
  idd INT PRIMARY KEY ,
```

```
  iNAME VARCHAR(10) ,
```

```
  grade INT
```

```
)
```

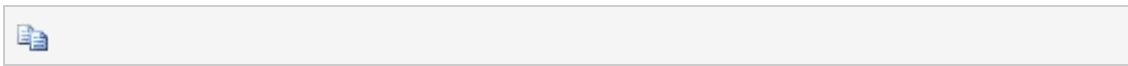
```
  PARTITION BY RANGE (grade)
```

```
(
```

```
  PARTITION part1 VALUES LESS THEN (1000) TABLESPACE Part1_tb,
```

```
  PARTITION part2 VALUES LESS THEN (MAXVALUE) TABLESPACE Part2_tb
```

```
);
```



2.列表分区:

该分区的特点是某列的值只有几个，基于这样的特点我们可以采用列表分区。

例 1



```
CREATE TABLE PROBLEM_TICKETS
```

```
(
```

```
    PROBLEM_ID NUMBER(7) NOT NULL PRIMARY KEY,
```

```
    DESCRIPTION VARCHAR2(2000),
```

```
    CUSTOMER_ID NUMBER(7) NOT NULL,
```

```
    DATE_ENTERED DATE NOT NULL,
```

```
    STATUS VARCHAR2(20)
```

```
)
```

```
PARTITION BY LIST (STATUS)
```

```
(
```

```
    PARTITION PROB_ACTIVE VALUES ('ACTIVE') TABLESPACE PROB_TS01,
```

```
    PARTITION PROB_INACTIVE VALUES ('INACTIVE') TABLESPACE PROB_TS02
```



例 2



```
CREATE TABLE ListTable
```

```
(
```

```
    id INT PRIMARY KEY ,
```

```
    name VARCHAR (20) ,
```

```
    area VARCHAR (10)
```

```

)

PARTITION BY LIST (area)

(

    PARTITION part1 VALUES ('guangdong','beijing') TABLESPACE Part1_tb,

    PARTITION part2 VALUES ('shanghai','nanjing') TABLESPACE Part2_tb

);

)

```



3.散列分区:

这类分区是在列值上使用散列算法，以确定将行放入哪个分区中。当列的值没有合适的条件时，建议使用散列分区。

散列分区为通过指定分区编号来均匀分布数据的一种分区类型，因为通过在 I/O 设备上进行散列分区，使得这些分区大小一致。

例 1:



```

CREATE TABLE HASH_TABLE

(

    COL NUMBER(8),

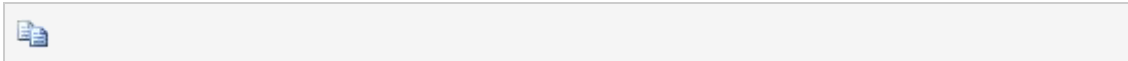
    INF VARCHAR2(100)

)

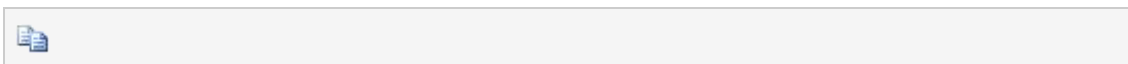
PARTITION BY HASH (COL)

```

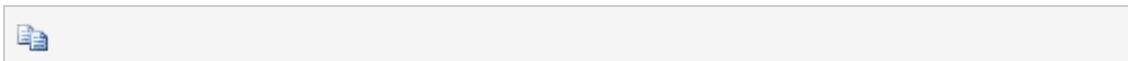
```
(  
  
    PARTITION PART01 TABLESPACE HASH_TS01,  
  
    PARTITION PART02 TABLESPACE HASH_TS02,  
  
    PARTITION PART03 TABLESPACE HASH_TS03  
  
)
```



简写:




```
CREATE TABLE emp  
  
(  
  
    empno NUMBER (4),  
  
    ename VARCHAR2 (30),  
  
    sal NUMBER  
  
)  
  
PARTITION BY HASH (empno) PARTITIONS 8  
  
STORE IN (emp1,emp2,emp3,emp4,emp5,emp6,emp7,emp8);
```



hash 分区最主要的机制是根据 hash 算法来计算具体某条纪录应该插入到哪个分区中,hash 算法中最重要的是 hash 函数, Oracle 中如果你要使用 hash 分区, 只需指定分区的数量即可。建议分区的数量采用 2 的 n 次方, 这样可以使得各个分区间数据分布更加均匀。

4.组合范围散列分区

这种分区是基于范围分区和列表分区，表首先按某列进行范围分区，然后再按某列进行列表分区，分区之中的分区被称为子分区。



```
CREATE TABLE SALES

(

PRODUCT_ID VARCHAR2 (5) ,

SALES_DATE DATE ,

SALES_COST NUMBER (10) ,

STATUS VARCHAR2 (20)

)

PARTITION BY RANGE (SALES_DATE) SUBPARTITION BY LIST (STATUS)

(

PARTITION P1 VALUES LESS THAN (TO_DATE ('2003-01-01', 'YYYY-MM-DD')) TABLESPACE rptfact2009
```

```

(
    SUBPARTITION P1SUB1 VALUES ('ACTIVE') TABLESPACE rptfact2009,

    SUBPARTITION P1SUB2 VALUES ('INACTIVE') TABLESPACE rptfact2009

),

PARTITION P2 VALUES LESS THAN (TO_DATE('2003-03-01','YYYY-MM-DD')) TABLESPACE rptfact2009

(
    SUBPARTITION P2SUB1 VALUES ('ACTIVE') TABLESPACE rptfact2009,

    SUBPARTITION P2SUB2 VALUES ('INACTIVE') TABLESPACE rptfact2009

)
)

```



5.复合范围散列分区:

这种分区是基于范围分区和散列分区，表首先按某列进行范围分区，然后再按某列进行散列分区。



```

create table dinya_test

(
    transaction_id number primary key,

    item_id number(8) not null,

    item_description varchar2(300),

    transaction_date date
)

```

```
)
```

```
partition by range(transaction_date) subpartition by hash(transaction_id) subpartitions 3 store in  
(dinya_space01,dinya_space02,dinya_space03)
```

```
(
```

```
partition part_01 values less than(to_date('2006-01-01','yyyy-mm-dd')),
```

```
partition part_02 values less than(to_date('2010-01-01','yyyy-mm-dd')),
```

```
partition part_03 values less than(maxvalue)
```

```
);
```



5.有关表分区的一些维护性操作

1) 添加分区

以下代码给 SALES 表添加了一个 P3 分区

```
ALTER TABLE SALES ADD PARTITION P3 VALUES LESS THAN(TO_DATE('2003-06-01','YYYY-MM-DD'));
```

注意：以上添加的分区界限应该高于最后一个分区界限。

以下代码给 SALES 表的 P3 分区添加了一个 P3SUB1 子分区

```
ALTER TABLE SALES MODIFY PARTITION P3 ADD SUBPARTITION P3SUB1 VALUES('COMPLETE');
```

2) 删除分区

以下代码删除了 P3 表分区：

```
ALTER TABLE SALES DROP PARTITION P3;
```

在以下代码删除了 P4SUB1 子分区：

```
ALTER TABLE SALES DROP SUBPARTITION P4SUB1;
```

注意：如果删除的分区是表中唯一的分区，那么此分区将不能被删除，要想删除此分区，必须删除表。

3) 截断分区

截断某个分区是指删除某个分区中的数据，并不会删除分区，也不会删除其它分区中的数据。当表中即使只有一个分区时，也可以截断该分区。通过以下代码截断分区：

```
ALTER TABLE SALES TRUNCATE PARTITION P2;
```

通过以下代码截断子分区：

```
ALTER TABLE SALES TRUNCATE SUBPARTITION P2SUB2;
```

4) 合并分区

合并分区是将相邻的分区合并成一个分区，结果分区将采用较高分区的界限，值得注意的是，不能将分区合并到界限较低的分区。以下代码实现了 P1 P2 分区的合并：

```
ALTER TABLE SALES MERGE PARTITIONS P1,P2 INTO PARTITION P2;
```

5) 拆分区

拆分区将一个分区拆分两个新分区，拆分后原来分区不再存在。注意不能对 HASH 类型的分区进行拆分。

```
ALTER TABLE SALES SBLIT PARTITION P2 AT (TO_DATE('2003-02-01','YYYY-MM-DD')) INTO (PARTITION P21,PARTITION P22);
```

6) 接合分区(coalesca)

结合分区是将散列分区中的数据接合到其它分区中，当散列分区中的数据比较大时，可以增加散列分区，然后进行接合，值得注意的是，接合分区只能用于散列分区中。通过以下代码进行接合分区：

```
ALTER TABLE SALES COALESCE PARTITION;
```

7) 重命名表分区

以下代码将 P21 更改为 P2

```
ALTER TABLE SALES RENAME PARTITION P21 TO P2;
```

8) 相关查询

跨分区查询



```
select sum( *) from
```

```
(select count(*) cn from t_table_SS PARTITION (P200709_1)
```

```
union all
```

```
select count(*) cn from t_table_SS PARTITION (P200709_2)
```

```
);
```



查询表上有多少分区

```
SELECT * FROM user_TAB_PARTITIONS WHERE TABLE_NAME='tableName'
```

查询索引信息



```
select object_name,object_type,tablespace_name,sum(value)

from v$segment_statistics

where statistic_name IN ('physical reads','physical write','logical reads')and object_type='INDEX'

group by object_name,object_type,tablespace_name

order by 4 desc
```



--显示数据库所有分区表的信息:

```
select * from DBA_PART_TABLES
```

--显示当前用户可访问的所有分区表信息:

```
select * from ALL_PART_TABLES
```

--显示当前用户所有分区表的信息:

```
select * from USER_PART_TABLES
```

--显示表分区信息 显示数据库所有分区表的详细分区信息:

```
select * from DBA_TAB_PARTITIONS
```

--显示当前用户可访问的所有分区表的详细分区信息:

```
select * from ALL_TAB_PARTITIONS
```

--显示当前用户所有分区表的详细分区信息:

```
select * from USER_TAB_PARTITIONS
```

--显示子分区信息 显示数据库所有组合分区表的子分区信息:

```
select * from DBA_TAB_SUBPARTITIONS
```

--显示当前用户可访问的所有组合分区表的子分区信息:

```
select * from ALL_TAB_SUBPARTITIONS
```

--显示当前用户所有组合分区表的子分区信息:

```
select * from USER_TAB_SUBPARTITIONS
```

--显示分区列 显示数据库所有分区表的分区列信息:

```
select * from DBA_PART_KEY_COLUMNS
```

--显示当前用户可访问的所有分区表的分区列信息:

```
select * from ALL_PART_KEY_COLUMNS
```

--显示当前用户所有分区表的分区列信息:

```
select * from USER_PART_KEY_COLUMNS
```

--显示子分区列 显示数据库所有分区表的子分区列信息:

```
select * from DBA_SUBPART_KEY_COLUMNS
```

--显示当前用户可访问的所有分区表的子分区列信息:

```
select * from ALL_SUBPART_KEY_COLUMNS
```

--显示当前用户所有分区表的子分区列信息:

```
select * from USER_SUBPART_KEY_COLUMNS
```

--怎样查询出 oracle 数据库中所有的分区表

```
select * from user_tables a where a.partitioned='YES'
```


--删除一个表的数据是

```
truncate table table_name;
```

--删除分区表一个分区的数据是

```
alter table table_name truncate partition p5;
```