Oracle 12C 新特征
- - - 分区表

王光辉
QQ: 83767582
微信:15147123801

CONTENTS

01

分区概述

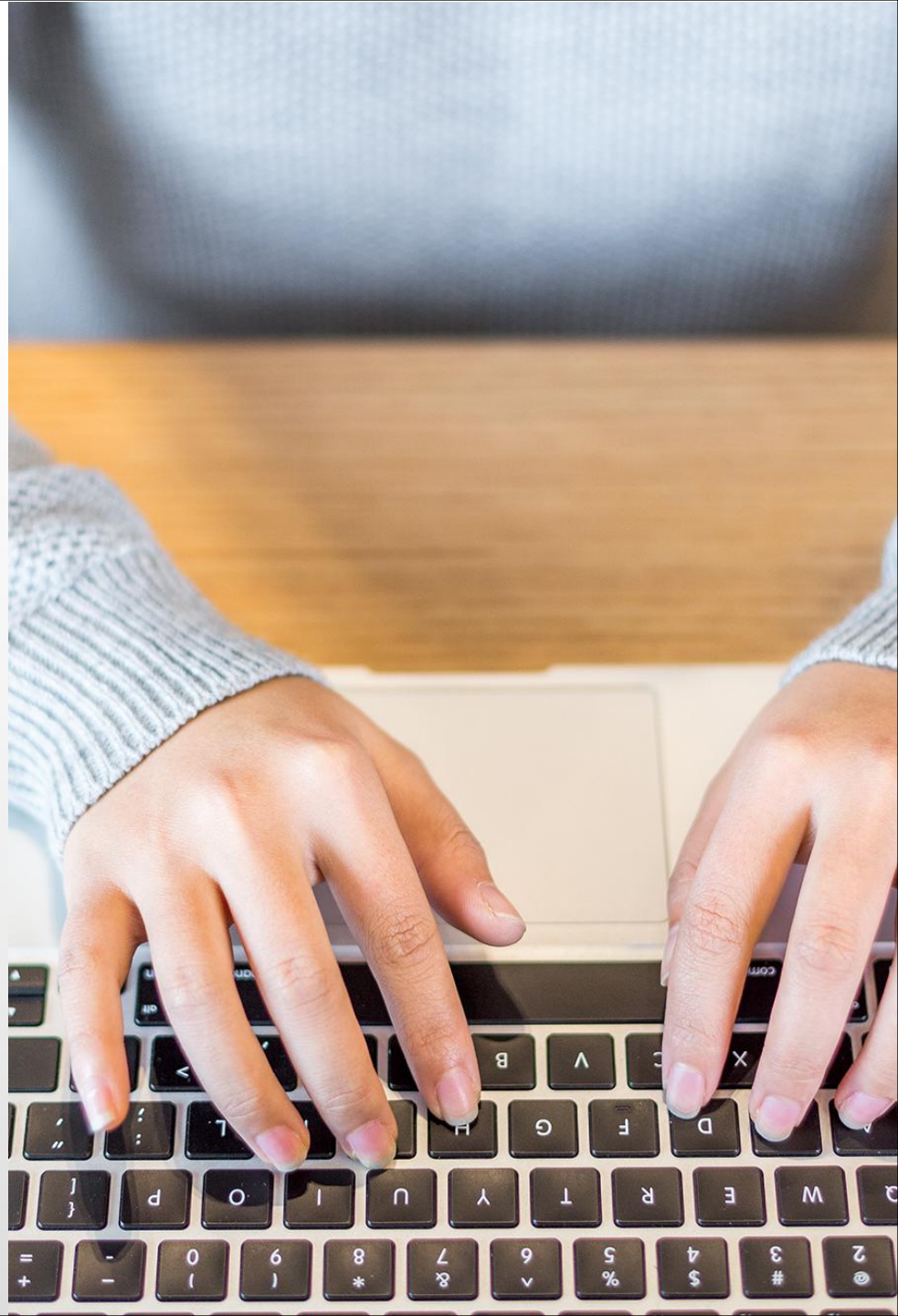
02

分区表收集统计信息

分区表收集统计信息

Optimizer statistics are a collection of data that describe more details about the database and the objects in the database.

These statistics are used by the query optimizer to choose the best execution plan for each SQL statement.



分区表收集统计信息

Optimizer statistics include the following:

- Table statistics
  - Number of rows
  - Number of blocks
  - Average row length
  - Global statistics (if requested)
- Column statistics
  - Number of distinct values (NDV) in column

分区表收集统计信息-What are Statistics in Oracle?

- Index statistics
  - Number of leaf blocks
  - Levels
  - Clustering factor
- System statistics
  - CPU performance and utilization
  - I/O performance and utilization

分区表收集统计信息-Collecting Statistics

- **Database statistics** are produced by:
  - Using the DBMS_STATS package (recommended)
  - Running the ANALYZE command
- **System statistics** are produced by using the DBMS_STATS package.
- Statistics are stored in the data dictionary tables and cached in the shared pool.
- CBO converts statistics into costs prior to use in costing execution plans.
- You can gather statistics for tables, indexes, and table columns separately.

分区表收集统计信息-Collecting Statistics

- DBMS_STATS 是 ANALYZE 的增强
- ANALYZE 的优缺点
  - 不能并行
  - 不能正确收集分区表的统计信息，即 global statistics
  - 可以收集跟optimizer 不相关的一些额外信息，如：
    CHAIN_CNT，VALIDATE STRUCTURE 等
- DBMS_STATS 的缺陷
  - 只能收集跟optimizer相关的统计信息

分区表收集统计信息-Display Statistics

- sosi.txt
- SOSI = Show Optimizer Statistics Information

- sosi.txt 来源于 MOS 上的文章 “SCRIPT - Select to show Optimizer Statistics for CBO [ID 31412.1]” 里所附带的脚本，它的使用方法非常简单，只需要运行一下 sosi.txt，并指定要查看统计信息的表名就可以了。sosi.txt 支持分区表，它的显示分为三部分，分别是表级别的统计信息，分区级别的统计信息和子分区级别的统计信息，其典型的显示结果为如下所示。

分区表收集统计信息-Table Statistics

- DBA_TABLES and DBA_TAB_[SUB]PARTITIONS
- Note: Table statistics are physically stored in the SYS.TAB$, SYS.TABPART$, SYS.TABSUBPART$ data dictionary table.
- Used to determine:
  - Table and (sub)partition access cost
  - Join cardinality
  - Join order
- Some of the statistics generated are:
  - Row count (NUM_ROWS)
  - Block count (BLOCKS)

分区表收集统计信息
---Table Statistics Inaccuracy Example

- 表统计信息不准导致SQL性能问题的实例
- Notes：再导入大量数据后应及时收集统计信息

分区表收集统计信息-Column Statistics

DBA_TAB_COL_STATISTICS and DBA_TAB_HISTOGRAMS
DBA_[SUB]PART_COL_STATISTICS
DBA_[SUB]PART_HISTOGRAMS

- Count of distinct values of the column (NUM_DISTINCT)
- Low value (LOW_VALUE)
- High value (HIGH_VALUE)
- Number of NULLS (NUM_NULLS)

分区表收集统计信息-Global Statistics

- What are global statistics
- How to gather global statistics
- How to check in dictionary views if global statistics are gathered
- global statistics 就是指直接从对象本身这一级收集到的统计信息，而不是从下一级对象“推导”或“汇总”出来的统计信息。
- global statistics 对优化器来说是非常重要的，一个SQL，除非其查询条件限定了数据只在部分分区上，否则在大多数情况下需要global statistics才能得到正确的执行计划。
- 有的统计值可以从对象的下一级对象进行汇总后得到，比如表的总行数，可以通过各分区的行数相加得到。但有的统计值则不能从下一级对象得到，比如列上的唯一值数量（distinct value）以及密度值（density）。

分区表收集统计信息-Collecting Statistics

How to Gather Global Statistics

- global statistics 只能通过 dbms_stats包来收集。
- 使用dbms_stats 收集统计信息时，参数granularity（比如 gather_table_stas过程）指定了那个级别上的统计信息会被收集
  - Global - - Table（表）
  - Partition - - Partition（分区）
  - SubPartition - - SubParttion（子分区）
  - Default - - Table + Partition（表 + 分区）
  - All - - Table + Partition + Subpartition（表 + 分区 + 子分区）
  - Auto - - Table + Partition + Subpartition（10g / 11g，表 + 分区，当子分区是list分区时还包括子分区）

分区表收集统计信息-Collecting Statistics

How to Check Global Statistics

Value:

- Yes means Global Statistics are gathered
- No means statistics are derived by aggregation from a lower level

At the Table Level

- [DBA|ALL|USER]_TABLES
- [DBA|ALL|USER]_TAB_COL_STATISTICS

At the Partition Level

- [DBA|ALL|USER]_TAB_PARTITIONS
- [DBA|ALL|USER]_PART_COL_STATISTICS

At the SubPartition Level

- [DBA|ALL|USER]_TAB_SUBPARTITIONS
- [DBA|ALL|USER]_SUBPART_COL_STATISTICS

分区表收集统计信息-Collecting Statistics

分区统计信息收集注意：

- 收集优化器统计信息，动态采样不是适用于Datawarehousing环境中的表中典型的大量数据的适当解决方案。
- 如果可行，可对空表运行查询（在数据加载之前）以填充列使用情况。这为以后的统计信息收集提供了有价值的信息。
- 在数据加载到表之后，但在创建索引之前（Oracle在创建索引时自动收集统计信息），应该进行统计信息收集。
- 对于估计百分比：在11g中，使用自动样本大小，它有一个新的哈希算法，非常准确的NDV（不同值的数量），同时又花费最少的时间。
- 对于10g中的样本大小，如果可能，使用100％，如果您知道数据偏斜，请使用直方图，否则谨慎。 （自动样本大小倾向于使用小样本，给出NDV的不良估计，由于列使用统计，常常获得直方图，如果这些列曾经被用于谓词条件）。
- 在10g和11g中使用增量统计信息收集
- 分区表受益于全局统计信息，没有全局统计信息，CBO估算来自单个分区的统计信息，NDV可能不准确

分区表收集统计信息-Collecting Statistics

Incremental statistics collection in 10g：

在10.2.0.4 + Patch 6526370和10.2.0.5新值中，GATHER_TABLE_STATS过程的GRANULARITY参数的'APPROX_GLOBAL AND PARTITION'模拟全局统计信息的增量维护。

我们更新表的NDV。 对于分区列，我们将NDV更新为分区级别的NDV之和。 另外，我们将本地唯一索引列的NDV设置为表的行数。 对于非分区列，我们不会在全局级别更新索引的NDV和键值的distinct的数量。

在下面的示例中，我们考虑由time_id分区的表SALES_INC，并收集统计信息。 最后一个分区为空SALES_Q4_2001。加载分区SALES_Q4_2001后，我们通过为GRANULARITY指定“APPROX_GLOBAL AND PARTITION”来收集统计信息。

加载分区SALES_Q4_2001之前：

USER_TABLES对列NUM_ROWS和Last_Analyzed显示以下值：

```
TABLE_NAME        NUM_ROWS Last Analyzed                                    GLOBAL_ST
--------------- ---------- ---------------------------------------------- ---------
SALES_INC           849094 02/20/2012 14:24                               YES
```

Incremental statistics collection in 10g

- USER_TAB_PARTITIONS显示分区
  SALES_Q4

```
TABLE_NAME      PARTITION_NAME  Last Analyzed
GLOBAL_ST   NUM_ROWS AVG_ROW_LEN HIGH_VALUE
--------------- --------------- ------------------------------------------------- -----
---- ---------- ---------- ------------------------------------------------- ...
SALES_INC       SALES_Q4_2000   02/20/2012 14:24
YES           55984          28 TO_DATE(' 2001-01-01 00:00:00', 'SYYYY-MM-DD HH24:
SALES_INC       SALES_Q4_2001   02/20/2012 14:24
YES              0           0 TO_DATE(' 2002-01-01 00:00:00', 'SYYYY-MM-DD HH24:
```

Incremental statistics collection in 10g

- USER_TAB_COLUMNS显示表的列的以下NDV：

TABLE_NAME	COLUMN_NAME	NUM_DISTINCT
SALES_INC	AMOUNT_SOLD	3296
SALES_INC	CHANNEL_ID	4
SALES_INC	CUST_ID	7056
SALES_INC	PROD_ID	72
SALES_INC	PROMO_ID	4
SALES_INC	QUANTITY_SOLD	1
SALES_INC	TIME_ID	1368

Incremental statistics collection in 10g

- 然后将数据导入分区SALES_Q4_2001中。统计数据已于14时50分收集

```
EXEC DBMS_STATS.GATHER_TABLE_STATS (ownname => 'SH', tabname
=>'SALES_INC', partname=>'SALES_Q4_2001', GRANULARITY =>
'APPROX_GLOBAL AND PARTITION');
```

Incremental statistics collection in 10g

- USER_TAB_PARTITIONS显示针对SALES_Q4_2001收集的分区级别统计信息（将列NUM_ROWS和LAST_ANALYZED与上面获得的结果进行比较）

```
TABLE_NAME       PARTITION_NAME  Last Analyzed
GLOBAL_ST   NUM_ROWS AVG_ROW_LEN HIGH_VALUE
---------------- --------------- ------------------------------------------------ -----
---- ---------- ----------- ------------------------------------------------ ...
SALES_INC        SALES_Q4_2000   02/20/2012 14:24
YES             55984           28 TO_DATE(' 2001-01-01 00:00:00', 'SYY
SALES_INC        SALES_Q4_2001   02/20/2012 14:50
YES             67872           28 TO_DATE(' 2002-01-01 00:00:00', 'SYY
```

Incremental statistics collection in 10g

- 显示表的全局统计信息的最后更新时间戳。

```
TABLE_NAME        NUM_ROWS Last Analyzed                                    GLOBAL_ST
--------------- ---------- ---------------------------------------------- ---------
SALES_INC           916966 02/20/2012 14:50                               YES
```

- 注意：USE
  的NDV已更

```
TABLE_NAME      COLUMN_NAME          NUM_DISTINCT
--------------- -------------------- ------------
SALES_INC       AMOUNT_SOLD                  3296
SALES_INC       CHANNEL_ID                      4
SALES_INC       CUST_ID                      7056
SALES_INC       PROD_ID                        72
SALES_INC       PROMO_ID                        4
SALES_INC       QUANTITY_SOLD                   1
SALES_INC       TIME_ID                      1460
```

Incremental statistics collection in 11g

- 如果满足以下所有条件，Oracle将通过仅扫描已更改的分区或子分区而不是整个表来更新全局表统计信息：
- 分区表的INCREMENTAL属性的值设置为TRUE（默认值为false）

```
DBMS_STATS.SET_TABLE_PREFS(<owner>,<tablename>,'INCREMENTAL','TRUE');
```

```
DBMS_STATS.SET_TABLE_PREFS(<owner>,<tablename>,'PUBLISH','TRUE');
```

- 分区表的PUBLISH属性的值设置为TRUE（默认值）

```
estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE
granularity      => 'AUTO'
```

Incremental statistics collection in 11g

- 设置为AUTO的粒度不会收集哈希子分区表的统计信息，
  因此当子分区类型为哈希时不会发生增量统计信息

03

12C 新特征

Incremental statistics collection in 12c

并发统计数据收集

- 此功能允许并发收集表内的多个分区（或子分区）上的统计信息。
- 通过在CONCURRENT设置为true时对分区表调用DBMS_STATS.GATHER_TABLE_STATS过程，Oracle会为表中的每个分区（或子分区）创建单独的统计信息收集作业。

增量统计的增强

- 已增强增量统计以支持分区交换加载。加载到非分区表中的数据可以与表中的分区交换，Oracle可以使用来自未分区表的统计信息，自动和准确地计算现有分区级统计信息和分区表的全

Incremental statistics collection in 12c

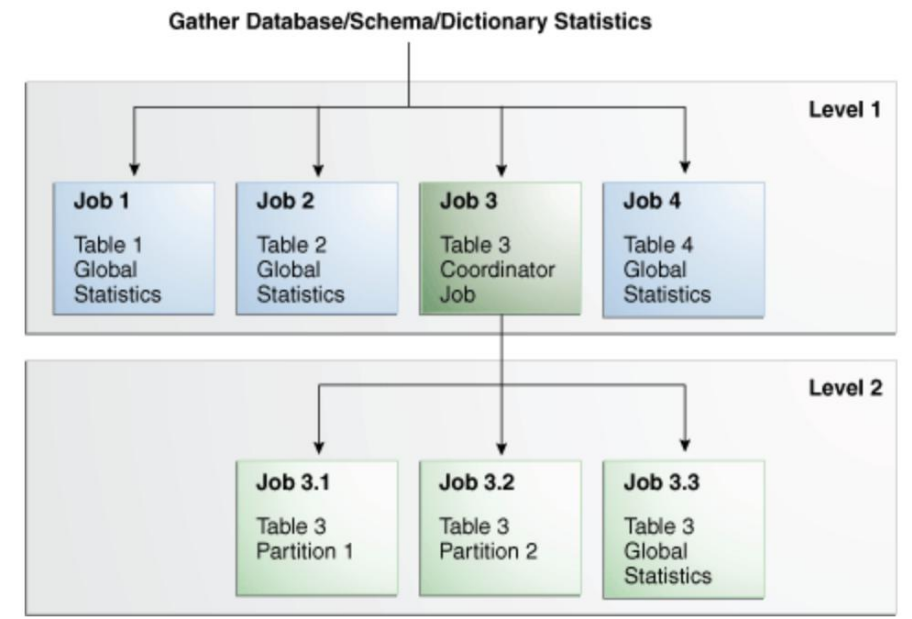
并发统计数据收集

- 并发统计收集使用户能够并发地收集表中模式（或数据库）中的多个表和表中的多个分区（或子分区）的统计信息。Oracle使用Oracle作业计划程序和高级队列组件来同时创建和管理多个统计信息收集作业。
- 如果在CONCURRENT设置为true时对分区表调用DBMS_STATS.GATHER_TABLE_STATS过程，则Oracle会为表中的每个分区（或子分区）创建单独的统计信息收集作业。Oracle作业计划程序根据可用的系统资源决定这些作业中有多少作业并行运行，以及有多少作业排队。随着当前运行的作业完成，更多作业将出队并执行，直到所有分区（或子分区）已收集其统计信息。

Incremental statistics collection in 12c

并发统计数据收集

```
BEGIN
  DBMS_STATS.SET_GLOBAL_PREFS('CONCURRENT','ALL');
END;
/
```



Identity Columns

12c 中的新特性，Identity Column，中文通常译为“标识列”，主要的功能是在一行数据被插入的同时，标识列的值自动+1，在实际的应用中有很多方便的用途，例如：

- 1. 自动成为表的主键
- 2.快速定位一行数据
- 3.方便表连接操作

Oracle主要用意在迁移支持identity columns列的数据库的数据到Oracle中来。

INCREMENTAL_STALENESS preference

这是一种增强功能，可以在增量统计信息收集模式中轻量DML更改的分区上跳过统计信息收集。

此修复程序为DBMS_STATS引入了一个新的INCREMENTAL_STALENESS首选项，它控制如何决定分区或子分区是否过时。

它的枚举值为：

'USE_STALE_PERCENT'，'USE_LOCKED_STATS'，'USE_STALE_PERCENT，USE_LOCKED_STATS'

- ‘USE_STALE_PERCENT'：分区/子分区不被视为stale，如果DML更改少于stale_percent首选项值
- 'USE_LOCKED_STATS'：锁定的分区/子分区统计不认为是陈旧的，不管dml变化
- null - 这是默认值，这意味着只要分区/子分区有任何DML更改，就认为该分区/子分区已失效。 当使用默认值时，以增量模式收集的统

Identity Columns

```
CREATE TABLE t1 (c1 NUMBER GENERATED BY DEFAULT
ON NULL AS IDENTITY, c2 VARCHAR2(10));
```

```
INSERT INTO t1(c2) VALUES ('abc');
INSERT INTO t1 (c1, c2) VALUES (null, 'xyz');

SQL> select * from t1;

        C1 C2
---------- ----------
         1 abc
         2 xyz
```

Identity Columns

- ALWAYS: If you specify ALWAYS, then Oracle Database always uses the sequence generator to assign a value to the column. If you attempt to explicitly assign a value to the column using INSERT or UPDATE, then an error will be returned. This is the default.
- BY DEFAULT: If you specify BY DEFAULT, then Oracle Database uses the sequence generator to assign a value to the column by default, but you can also explicitly assign a specified value to the column. If you specify ON NULL, then Oracle Database uses the sequence generator to assign a value to the column when a subsequent INSERT statement attempts to assign a value that evaluates to NULL.

Automatic List-Partitioned Table

- 自动列表分区方法允许按需创建列表分区。自动列表分区表类似于常规列表分区表，除了该分区表更容易管理。您可以仅使用已知的分区键值创建自动列表分区表。当数据加载到表中时，如果加载的分区键值不对应于任何现有分区，则数据库自动创建新分区。因为分区是根据需要自动创建的，所以自动列表分区方法在概念上类似于现有的间隔分区方法。

```
CREATE TABLE sales_auto_list
(
   salesman_id    NUMBER(5),
   salesman_name  VARCHAR2(30),
   sales_state    VARCHAR2(20),
   sales_amount   NUMBER(10),
   sales_date     DATE
)
  PARTITION BY LIST (sales_state) AUTOMATIC
 (PARTITION P_CAL VALUES ('CALIFORNIA')
);
```

Automatic List-Partitioned Table

- 对值非常频繁变化的数据类型的自动列表分区不太适合此方法，除非可以调整数据。
- 例如，具有日期值的SALES_DATE字段（当格式未删减时）将每秒增加一次。每个SALES_DATE值，如05-22-2016 08:00:00，05-22-2016 08:00:01等，都会生成自己的分区。为了避免创建大量分区，您必须知道将要输入的数据并相应地进行调整。例如，您可以将SALES_DATE日期值截断为某天或某些其他时间段，具体取决于所需的分区数。
- CREATE和ALTER TABLE SQL语句使用附加子句进行更新，以指定AUTOMATIC或MANUAL列表分区。自动列表分区表在创建时必须至少有一个分区。因为会为新的和未知的分区键值自动创建新分区，所以自动列表分区不能具有DEFAULT分区。

Multi-column List-Partitioned Table

- 多列列表分区允许您基于多列的列表值进行分区。 与单列列表分区类似，单个分区可以包含列表值的集合。
- 多列列表分区表只能有一个DEFAULT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)
```

Read-Only Partitions or Subpartitions

- 您可以将表，分区和子分区设置为只读状态，以保护数据免受任何用户或触发器的无意DML操作。在设置为只读的分区或子分区中更新数据的任何尝试都会导致错误，同时在设置为读写成功的分区或子分区中才能更新数据。

```
CREATE TABLE orders_read_write_only
( order_id NUMBER (12),
  order_date DATE CONSTRAINT order_date_nn NOT NULL,
  state VARCHAR2(2)
) READ WRITE
  PARTITION BY RANGE (order_date)
    SUBPARTITION BY LIST (state)
     ( PARTITION order_p1 VALUES LESS THAN (TO_DATE ('01-
DEC-2015','DD-MON-YYYY')) READ ONLY
      ( SUBPARTITION order_p1_northwest VALUES ('OR',
'WA'),
        SUBPARTITION order_p1_southwest VALUES ('AZ', 'UT',
'NM')
      ),
     PARTITION order_p2 VALUES LESS THAN (TO_DATE ('01-
MAR-2016','DD-MON-YYYY'))
      ( SUBPARTITION order_p2_northwest VALUES ('OR',
'WA'),
        SUBPARTITION order_p2_southwest VALUES ('AZ', 'UT',
'NM') READ ONLY
      ),
     PARTITION order_p3 VALUES LESS THAN (TO_DATE ('01-
JUL-2016','DD-MON-YYYY'))
```

Partitioned External Table

- 对于范围分区和散列分区表，最多可以指定16个分区键列。当分区键由多个列组成时，使用多列分区，后续列定义的粒度高于前面的列。最常见的情况是分解的DATE或TIMESTAMP键，由年，月和日的单独列组成。

- 在评估多列分
目标分区时使
正确分区时使
全匹配并且为
确的分区。仅
个界限完全匹
边界值完全匹
有绑定值完全
下一个分区的

```
CREATE TABLE sales_ext (loc_id number, prod_id number,
cust_id number, amount_sold number, quantity_sold number)
   ORGANIZATION EXTERNAL
   (TYPE oracle_loader
    DEFAULT DIRECTORY load_d1
    ACCESS PARAMETERS
    ( RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
      NOBADFILE
      LOGFILE log_dir:'sales.log'
      FIELDS TERMINATED BY ","
     )
   )
    REJECT LIMIT UNLIMITED
  PARTITION BY RANGE (loc_id)
  (PARTITION p1 VALUES LESS THAN (1000) LOCATION
 ('california.txt'),
    PARTITION p2 VALUES LESS THAN (2000) DEFAULT DIRECTORY
 load_d2 LOCATION ('washington.txt'),
```

Deferred Segment Creation for Automatic List Partitions and Interval Subpartitions

- 自动列表复合分区表和间隔子分区仅在存在数据时创建子分区; 此操作可节省空间。 在按需创建新分区时推迟创建子分区段确保仅在插入第一个匹配行时创建子分区段。

Conversion of a Non-Partitioned Table to a Partitioned Table

- 未分区表可以转换为已添加到ALTER TABLE SQL语句中的MODIFY子句的分区表。 此外，可以指定关键字ONLINE，从而在转换正在进行时启用并发DML操作。

```
ALTER TABLE employees_convert MODIFY
  PARTITION BY RANGE (employee_id) INTERVAL (100)
  ( PARTITION P1 VALUES LESS THAN (100),
    PARTITION P2 VALUES LESS THAN (500)
   ) ONLINE
  UPDATE INDEXES
 ( IDX1_SALARY LOCAL,
   IDX2_EMP_ID GLOBAL PARTITION BY RANGE (employee_id)
  ( PARTITION IP1 VALUES LESS THAN (MAXVALUE))
);
```

Conversion of a Non-Partitioned Table to a Partitioned Table

当使用UPDATE INDEXES子句时，请注意以下内容。

- 此子句可用于更改要转换的索引的索引和存储属性的分区状态。
- UPDATE INDEXES子句的规范是可选的。用于在线和离线转换到分区表，同时维护索引。
- 此子句不能更改定义原始索引列表的列。
- 此子句不能更改索引或任何其他索引属性的唯一性属性。
- 如果不为任何索引指定表空间，那么将应用以下表空间默认值。
  - 转换后的本地索引与表分区并置。
  - 转换后的全局索引驻留在未分区表上的原始全局索引的同一表空间中。
- 如果未指定INDEXES子句或INDEXES子句未指定原始未分区表上的所有索引，则以下默认行为适用于所有未指定的索引。
  - 全局分区索引保持不变，并保留原始分区形状。
  - 非前缀索引成为全局非分区索引。

Filtering Maintenance Operations

- 分区维护操作支持添加数据过滤，实现分区和数据维护的组合。过滤的分区维护操作仅保留满足数据过滤的数据作为分区维护的一部分。数据过滤的能力适用于MOVE PARTITION，MERGE PARTITION和SPLIT PARTITION。
- 过滤谓词必须位于分区表上。可以在线执行的所有分区维护操作（MOVE和SPLIT）也可以作为过滤的分区维护操作执行。指定ONLINE时，允许对正在维护的分区执行DML操作。

```
ALTER TABLE orders_move_part
  MOVE PARTITION q1_2016 TABLESPACE open_orders COMPRESS
ONLINE
  INCLUDING ROWS WHERE order_state = 'open';
```

Using Table Compression with Partitioned Tables

- 对于堆组织的分区表，可以使用表压缩来压缩一些或所有分区。 可以为表空间，表或表的分区声明压缩属性。 无论何时未指定compress属性，都会像任何其他存储属性一样继承。

```
CREATE TABLE costs_demo (
    prod_id      NUMBER(6),     time_id      DATE,
    unit_cost    NUMBER(10,2), unit_price   NUMBER(10,2))
PARTITION BY RANGE (time_id)
   (PARTITION costs_old
       VALUES LESS THAN (TO_DATE('01-JAN-2003', 'DD-MON-
YYYY')) COMPRESS,
    PARTITION costs_q1_2003
       VALUES LESS THAN (TO_DATE('01-APR-2003', 'DD-MON-
YYYY')),
    PARTITION costs_q2_2003
       VALUES LESS THAN (TO_DATE('01-JUN-2003', 'DD-MON-
YYYY')),
    PARTITION costs_recent VALUES LESS THAN (MAXVALUE));
```

Using Key Compression with Partitioned Indexes

- 您可以使用键压缩来压缩B树索引的某些或所有分区。 密钥压缩仅适用于B树索引。 默认情况下，位图索引以压缩方式存储。 使用键压缩的索引消除了键列前缀值的重复出现，从而节省了空间和I / O。
- 以下示例为所有分区创建了一个本地分区索引，除了最近一个分区被压缩：

```
CREATE INDEX i cost1 ON costs demo (prod id) COMPRESS
LOCAL
   (PARTITION costs_old, PARTITION costs_q1_2003,
    PARTITION costs_q2_2003, PARTITION costs_recent
NOCOMPRESS);
```

谢谢！