分而治之 -
Oracle 最新分区特性优化实践

杨廷琨

个人介绍

- 杨廷琨(yangtingkun)
  - Oracle ACE Director
  - ITPUB数据库管理区版主
  - ACOUG核心会员
  - 参与编写《Oracle数据库性能优化》、《Oracle DBA手记》、《Oracle DBA手记3》和《Oracle性能优化与诊断案例精选》
  - 十八年的一线DBA经验
  - 个人BLOG中积累了2500篇原创技术文章
  - 云和恩墨CTO

目录

- 分区基本概念
- 分区演进历史
- 分区最佳实践
- 分区最新特性

分区概述

- 定义
  - 根据内部定义的规则，将一张表的数据拆分到多个数据段中。
  - 对应用透明，程序可以不做任何额外调整
  - 可以通过分区列上的条件访问指定分区的数据，也可以通过分区扩展语句显式的访问
  - 分区级别上提供删除，截断，迁移，索引的能力

分区概述

- 分区的优点
  - 可维护性
  - 可用性增强
  - OLTP：降低共享资源争用
  - OLAP：提升查询性能

目录

- 分区基本概念
- 分区演进历史
- 分区最佳实践
- 分区最新特性

Oracle分区演进历史

版本	功能	性能	管理性
Oracle 8.0	范围分区 本地、全局范围索引	静态分区剪裁	基本维护功能：ADD、DROP、EXCHANGE
Oracle 8.1	哈希分区 范围哈希分区	分区智能连接 动态分区剪裁	扩展维护功能：MERGE
Oracle 9.0	列表分区		全局索引维护
Oracle 9.2	范围列表分区	快速分区SPLIT	
Oracle 10.1	全局哈希索引分区		本地索引维护
Oracle 10.2	单表允许1百万分区	多维度分区剪裁	快速DROP TABLE
Oracle 11.1	虚拟列分区 多重复合分区方式 参考分区		间隔分区 分区建议 增量统计信息收集
Oracle 11.2	HASH复合分区 扩展参考分区	“AND”分区剪裁	多分支执行

Oracle分区演进历史

版本	功能	性能	管理性
Oracle 12.1	间隔参考分区 范围分区支持哈希聚簇	多个分区的分区维护操作 异步全局索引维护	在线分区移动 级联截断 部分分区索引
Oracle 12.2	多列列表分区 自动列表分区 间隔子分区 外部表分区	多种在线分区操作 非分区表与分区表在线转换 DDL操作减少游标失效	过滤分区维护操作 只读分区 创建交换分区表
Oracle 18		并行分区连接性能增强	在线修改分区策略 在线合并分区

目录

- 分区基本概念
- 分区演进历史
- 分区最佳实践
- 分区最新特性

范围分区

- 适用场景
  - 时间属性
  - 关注近期数据
- 最佳实践
  - 数据生命周期和访问范围
  - 明确时间条件限定
  - 定期清理过期分区
  - 非主键优先本地索引
- 优势
  - 数据分布平均，分区数量可控
  - DDL清理过期数据
  - 查询仅访问个别分区
  - 表大小相对稳定
  - 索引大小相对稳定

范围分区

- 面临挑战
  - 分区清理逻辑复杂
  - DELETE效率低下
  - DELETE无法释放空间
  - 数据量迅速膨胀
  - 分区数量不断增长

- 解决方案
  - INSERT + EXCHANGE方式
  - MERGE分区减少分区数量
  - 避免DELETE效率低下
  - 避免空间无法释放

范围分区

```
SQL> SELECT TEMPORARY, COUNT(*) FROM T_PART PARTITION (P3) GROUP BY TEMPORARY;

T   COUNT(*)
- ----------
Y         30
N      87261

SQL> LOCK TABLE T_PART PARTITION (P3) IN EXCLUSIVE MODE;

Table(s) Locked.

SQL> INSERT INTO T_INTER SELECT * FROM T_PART PARTITION (P3) WHERE TEMPORARY = 'Y';

30 rows created.

SQL> ALTER TABLE T_PART EXCHANGE PARTITION P3 WITH TABLE T_INTER;

Table altered.

SQL> SELECT TEMPORARY, COUNT(*) FROM T_PART PARTITION (P3) GROUP BY TEMPORARY;

T   COUNT(*)
- ----------
Y         30
```

范围分区

- 面临挑战
  - 子表不包含分区时间列
  - 主子表时间列字段含义不同
  - 数据清理破坏主子表依赖关系
  - 子表需要冗余主表分区字段
  - 主外键约束禁止DDL操作

- 解决方案
  - 主表时间字段范围分区
  - 子表外键建立参考分区
  - 参考分区不会破坏主外键依赖关系

参考分区

- 适用场景
  - 主子表采用相同的数据清理策略
  - 子表没有合适的分区字段
  - 主子表经常关联访问

```
SQL> CREATE TABLE T_PRIMARY (ID NUMBER PRIMARY KEY, NAME VARCHAR2(128), CREATED DATE)
  2  PARTITION BY RANGE (CREATED)
  3  (PARTITION P1 VALUES LESS THAN (TO_DATE('201512', 'YYYYMM')),
  4  PARTITION P2 VALUES LESS THAN (MAXVALUE));

Table created.

SQL> CREATE TABLE T_FOREIGN (ID NUMBER, FID NUMBER NOT NULL, NAME VARCHAR2(128),
  2  CONSTRAINT FK_FID FOREIGN KEY (FID) REFERENCES T_PRIMARY (ID))
  3  PARTITION BY REFERENCE (FK_FID);

Table created.
```

哈希分区

- 适用场景
  - 没有时间属性
  - 缺少区分数据的业务字段
- 最佳实践
  - 分区键值列选择重复度不高的字段
  - 分区数量应为2的幂
  - 多创建全局索引
  - 哈希分区索引可解决索引热点块问题
- 优势
  - 分区没有业务特点的数据
  - 数据均匀分布
  - 有效解决递增索引的热点块问题

哈希全局分区索引

- 批量插入导致索引热点块争用
- RAC全局等待加重热点块问题
- 逆键索引不支持范围扫描

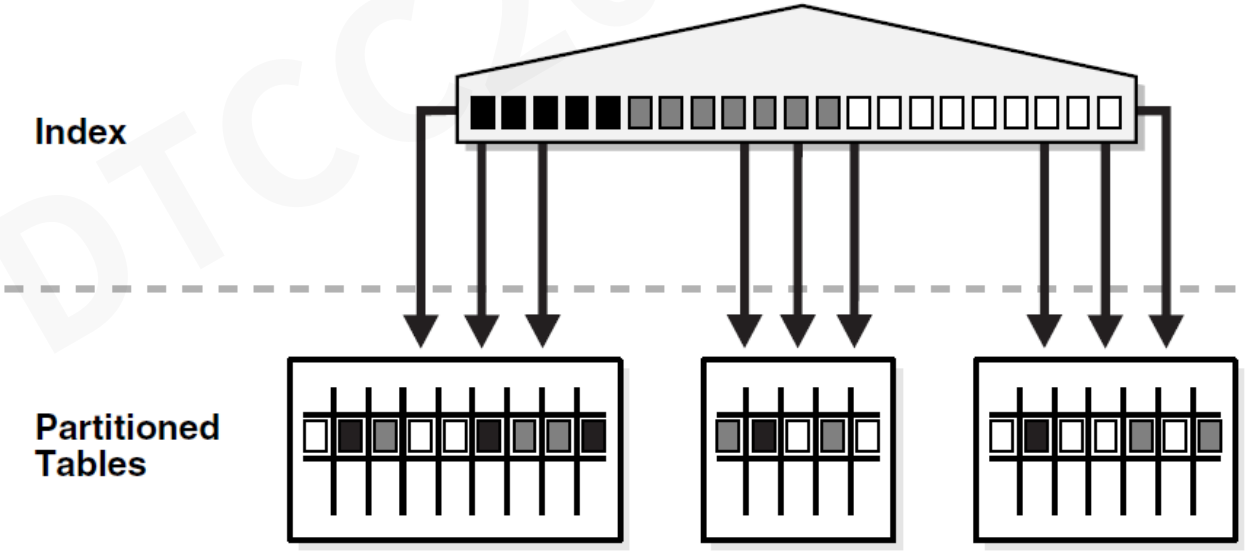
```
SQL> CREATE TABLE T_PART (ID NUMBER, NAME VARCHAR2(30), CREATED DATE);

Table created.

SQL> CREATE INDEX IND_PART_CREATED ON T_PART(CREATED) GLOBAL
  2  PARTITION BY HASH (CREATED)
  3  PARTITIONS 32;

Index created.
```

列表分区

- 适用场景
  - 地域，类型等有限的业务属性
  - 访问一种或几种业务属性
  - 通过业务属性可平均拆分数据
- 最佳实践
  - 地区字段是常见候选
  - 数据分布和访问方式确定分区键值划分
  - 设定DEFAULT分区
- 优势
  - 分区方式和业务匹配度更好
  - 数据如何在分区中存放的选择度更高
  - 分区键值与分区的对应更加明确

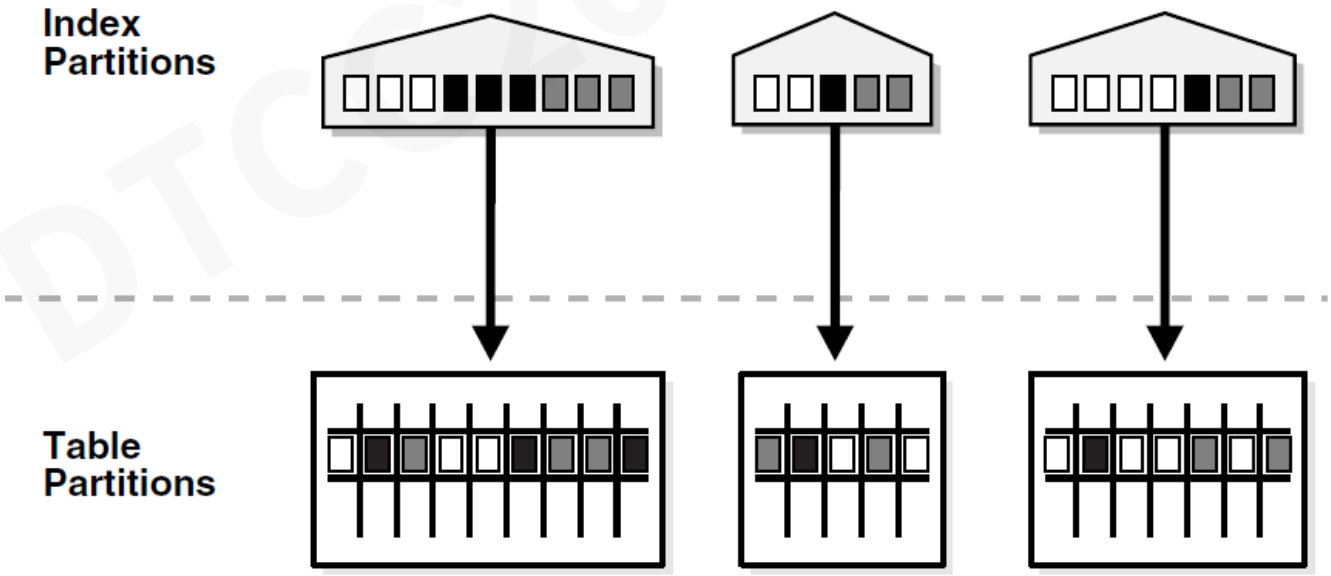
分区索引

- 全局索引
  - 索引扫描不会导致逻辑读增加
  - 大部分分区维护操作会导致索引失效
  - 对于并行执行没有帮助



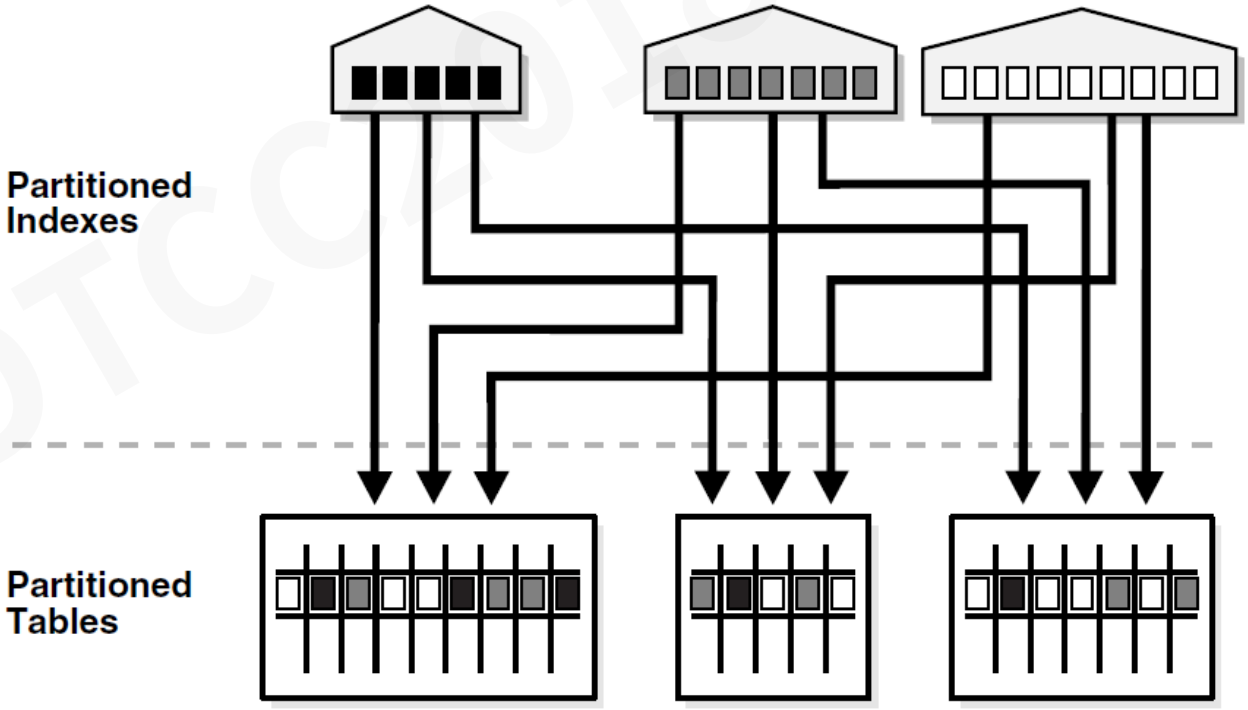
分区索引

- 本地索引
  - 大部分分区维护操作不会导致索引失效
  - 支持并行扫描和创建
  - 唯一索引必须包含分区键值列
  - 未限定分区的查询将扫描全部索引分区



分区索引

- 全局哈希分区索引
  - 具有分散热点数据的能力
  - 索引范围扫描将扫描全部索引分区



目录

- 分区基本概念
- 分区演进历史
- 分区最佳实践
- 分区最新特性

12C分区新特性

版本	功能	性能	管理性
Oracle 12.1	间隔参考分区 范围分区支持哈希聚簇	多个分区的分区维护操作 异步全局索引维护	在线分区移动 级联截断 部分分区索引
Oracle 12.2	多列列表分区 自动列表分区 间隔子分区 外部表分区	多种在线分区操作 非分区表与分区表在线转换 DDL操作减少游标失效	过滤分区维护操作 只读分区 创建交换分区表
Oracle 18		并行分区连接性能增强	在线修改分区策略 在线合并分区

12.1新特性

- 部分分区索引

```
SQL> CREATE TABLE T_PARTIAL_INDEX (ID NUMBER, NAME VARCHAR2(30), CREATED DATE)
  2  PARTITION BY RANGE (CREATED)
  3  (PARTITION P201701 VALUES LESS THAN (TO_DATE('2017-1-1', 'YYYY-MM-DD')),
  4  PARTITION P201704 VALUES LESS THAN (TO_DATE('2017-4-1', 'YYYY-MM-DD')),
  5  PARTITION P201707 VALUES LESS THAN (TO_DATE('2017-7-1', 'YYYY-MM-DD')),
  6  PARTITION P201710 VALUES LESS THAN (TO_DATE('2017-10-1', 'YYYY-MM-DD')),
  7  PARTITION P201801 VALUES LESS THAN (TO_DATE('2018-1-1', 'YYYY-MM-DD')),
  8  PARTITION P201804 VALUES LESS THAN (TO_DATE('2018-4-1', 'YYYY-MM-DD')),
  9  PARTITION P201807 VALUES LESS THAN (TO_DATE('2018-7-1', 'YYYY-MM-DD')),
 10  PARTITION P201810 VALUES LESS THAN (TO_DATE('2018-10-1', 'YYYY-MM-DD')),
 11  PARTITION PMAX VALUES LESS THAN (MAXVALUE));

Table created.

SQL> INSERT INTO T_PARTIAL_INDEX SELECT ROWNUM, SUBSTR(OBJECT_NAME, 1, 30), CREATED FROM DBA_OBJECTS;

74281 rows created.

SQL> COMMIT;

Commit complete.
```

12.1新特性

- 部分分区索引

```
SQL> ALTER TABLE T_PARTIAL_INDEX MODIFY PARTITION P201701 INDEXING OFF;

Table altered.

SQL> ALTER TABLE T_PARTIAL_INDEX MODIFY PARTITION P201704 INDEXING OFF;

Table altered.

SQL> ALTER TABLE T_PARTIAL_INDEX MODIFY PARTITION P201707 INDEXING OFF;

Table altered.

SQL> ALTER TABLE T_PARTIAL_INDEX MODIFY PARTITION P201710 INDEXING OFF;

Table altered.

SQL> CREATE INDEX IND_PARTIAL_NAME ON T_PARTIAL_INDEX(NAME) LOCAL INDEXING PARTIAL;

Index created.

SQL> CREATE INDEX IND_PARTIAL_ID ON T_PARTIAL_INDEX(ID) INDEXING PARTIAL;

Index created.
```

12.1新特性

- 部分分区索引

```
SQL> SELECT INDEX_NAME, PARTITION_NAME, STATUS FROM USER_IND_PARTITIONS WHERE INDEX_NAME = 'IND_PARTIAL_NAME';

INDEX_NAME       PARTITION_ STATUS
---------------- ---------- ----------------
IND_PARTIAL_NAME P201701    UNUSABLE
IND_PARTIAL_NAME P201704    UNUSABLE
IND_PARTIAL_NAME P201707    UNUSABLE
IND_PARTIAL_NAME P201710    UNUSABLE
IND_PARTIAL_NAME P201801    USABLE
IND_PARTIAL_NAME P201804    USABLE
IND_PARTIAL_NAME P201807    USABLE
IND_PARTIAL_NAME P201810    USABLE
IND_PARTIAL_NAME PMAX       USABLE

9 rows selected.
```

12.1新特性

- 部分分区索引

```
SQL> SELECT ID, NAME FROM T_PARTIAL_INDEX WHERE ID = 10000;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		1	31	5 (0)	00:00:01		
1	VIEW	VW_TE_2	2	60	5 (0)	00:00:01		
2	UNION-ALL							
* 3	TABLE ACCESS BY GLOBAL INDEX ROWID BATCHED	T_PARTIAL_INDEX	1	39	2 (0)	00:00:01	ROWID	ROWID
* 4	INDEX RANGE SCAN	IND_PARTIAL_ID	1		1 (0)	00:00:01		
5	PARTITION RANGE ITERATOR		1	39	3 (0)	00:00:01	1	4
* 6	TABLE ACCESS FULL	T_PARTIAL_INDEX	1	39	3 (0)	00:00:01	1	4

```
   3 - filter("T_PARTIAL_INDEX"."CREATED">=TO_DATE(' 2017-10-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss') OR
             "T_PARTIAL_INDEX"."CREATED" IS NULL)
   4 - access("ID"=10000)
   6 - filter("ID"=10000)
```

12.1新特性

- 部分分区索引

```
SQL> SELECT ID, NAME FROM T_PARTIAL_INDEX WHERE ID = 10000 AND CREATED = TO_DATE('2017-4-28', 'YYYY-MM-DD');

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name            | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |     1 |    39 |    62   (0)| 00:00:01 |       |       |
|   1 |  VIEW                         | VW_TE_2         |     2 |    60 |    62   (0)| 00:00:01 |       |       |
|   2 |   UNION-ALL                   |                 |       |       |            |          |       |       |
|*  3 |    FILTER                     |                 |       |       |            |          |       |       |
|   4 |     PARTITION RANGE SINGLE    |                 |     1 |    39 |    62   (0)| 00:00:01 |     3 |     3 |
|*  5 |      TABLE ACCESS FULL        | T_PARTIAL_INDEX |     1 |    39 |    62   (0)| 00:00:01 |     3 |     3 |
|   6 |    PARTITION RANGE SINGLE     |                 |     1 |    39 |    62   (0)| 00:00:01 |     3 |     3 |
|*  7 |     TABLE ACCESS FULL         | T_PARTIAL_INDEX |     1 |    39 |    62   (0)| 00:00:01 |     3 |     3 |
-------------------------------------------------------------------------------------------------------------

   3 - filter(NULL IS NOT NULL)
   5 - filter("CREATED"=TO_DATE(' 2017-04-28 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND "ID"=10000)
   7 - filter("CREATED"=TO_DATE(' 2017-04-28 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND "ID"=10000)
```

12.1新特性

- 部分分区索引

```
SQL> SELECT ID, NAME FROM T_PARTIAL_INDEX WHERE ID = 10000 AND CREATED = TO_DATE('2018-4-28', 'YYYY-MM-DD');
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		1	34	2 (0)	00:00:01		
1	VIEW	VW_TE_2	2	60	2 (0)	00:00:01		
2	UNION-ALL							
* 3	TABLE ACCESS BY GLOBAL INDEX ROWID BATCHED	T_PARTIAL_INDEX	1	34	2 (0)	00:00:01	7	7
* 4	INDEX RANGE SCAN	IND_PARTIAL_ID	1		1 (0)	00:00:01		
* 5	FILTER							
* 6	TABLE ACCESS BY GLOBAL INDEX ROWID BATCHED	T_PARTIAL_INDEX	1	34	2 (0)	00:00:01	7	7
* 7	INDEX RANGE SCAN	IND_PARTIAL_ID	1		1 (0)	00:00:01		

```
3 - filter("CREATED"=TO_DATE(' 2018-04-28 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
4 - access("ID"=10000)
5 - filter(NULL IS NOT NULL)
6 - filter("CREATED"=TO_DATE(' 2018-04-28 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
7 - access("ID"=10000)
```

12.1新特性

- 部分分区索引

```
SQL> SELECT ID, NAME FROM T_PARTIAL_INDEX WHERE NAME = 'TEST';
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		2	62	19 (0)	00:00:01		
1	VIEW	VW_TE_2	4	120	19 (0)	00:00:01		
2	UNION-ALL							
3	VIEW	VW_ORE_27726257	3	90	16 (0)	00:00:01		
4	UNION-ALL							
5	PARTITION RANGE ITERATOR		2	78	12 (0)	00:00:01	5	9
*6	TABLE ACCESS BY LOCAL INDEX ROWID BATCHED	T_PARTIAL_INDEX	2	78	12 (0)	00:00:01	5	9
*7	INDEX RANGE SCAN	IND_PARTIAL_NAME	2		11 (0)	00:00:01	5	9
8	PARTITION RANGE SINGLE		1	39	4 (0)	00:00:01	9	9
*9	TABLE ACCESS BY LOCAL INDEX ROWID BATCHED	T_PARTIAL_INDEX	1	39	4 (0)	00:00:01	9	9
*10	INDEX RANGE SCAN	IND_PARTIAL_NAME	2		3 (0)	00:00:01	9	9
11	PARTITION RANGE ITERATOR		1	39	3 (0)	00:00:01	1	4
*12	TABLE ACCESS FULL	T_PARTIAL_INDEX	1	39	3 (0)	00:00:01	1	4

```
   6 - filter("T_PARTIAL_INDEX"."CREATED" IS NOT NULL)
   7 - access("NAME"='TEST')
   9 - filter("T_PARTIAL_INDEX"."CREATED" IS NULL AND LNNVL("T_PARTIAL_INDEX"."CREATED">=TO_DATE(' 2017-10-01 00:00:00', 'syyyy-mm-dd
hh24:mi:ss')))
  10 - access("NAME"='TEST')
  12 - filter("NAME"='TEST')
```

12.1新特性

- 全局索引异步维护

```
SQL> CREATE TABLE T_PART_GLO_IND (ID NUMBER, NAME VARCHAR2(30), CREATED DATE)
  2  PARTITION BY RANGE (CREATED) INTERVAL (INTERVAL '1' MONTH)
  3  (PARTITION P1 VALUES LESS THAN (TO_DATE('2016-1-1', 'YYYY-MM-DD')));

Table created.

SQL> INSERT INTO T_PART_GLO_IND SELECT ROWNUM, SUBSTR(OBJECT_NAME, 1, 30), TO_DATE('2015-12-30') + ROWNUM/10000
  2  FROM DBA_OBJECTS A, (SELECT 1 FROM DUAL CONNECT BY ROWNUM < 100) B;

7355106 rows created.

SQL> COMMIT;

Commit complete.

SQL> CREATE INDEX IND_GLO_NAME ON T_PART_GLO_IND(NAME);

Index created.
```

12.1新特性

- 全局索引异步维护

```
SQL> SELECT PARTITION_NAME, PARTITION_POSITION FROM USER_TAB_PARTITIONS
  2  WHERE TABLE_NAME = 'T_PART_GLO_IND';

PARTITION_NAME                 PARTITION_POSITION
------------------------------ ------------------
P1                                              1
SYS_P3481                                       2
SYS_P3482                                       3
SYS_P3483                                       4
SYS_P3484                                       5
SYS_P3485                                       6
SYS_P3486                                       7
SYS_P3487                                       8
SYS_P3488                                       9
.
.
.
SYS_P3505                                      26

26 rows selected.
```

12.1新特性

- 全局索引异步维护

```
SQL> SELECT COUNT(*) FROM T_PART_GLO_IND PARTITION FOR (TO_DATE('2017-1-4', 'YYYY-MM-DD'));

  COUNT(*)
----------
    310000

SQL> SELECT COUNT(*) FROM T_PART_GLO_IND PARTITION FOR (TO_DATE('2017-3-5', 'YYYY-MM-DD'));

  COUNT(*)
----------
    310000

SQL> ALTER TABLE T_PART_GLO_IND TRUNCATE PARTITION FOR (TO_DATE('2017-1-4', 'YYYY-MM-DD'));

Table truncated.

Elapsed: 00:00:00.18
```

12.1新特性

- 全局索引异步维护

```
SQL> SELECT INDEX_NAME, STATUS FROM USER_INDEXES WHERE TABLE_NAME = 'T_PART_GLO_IND';

INDEX_NAME                     STATUS
------------------------------ ----------------
IND_GLO_NAME                   UNUSABLE

SQL> ALTER INDEX IND_GLO_NAME REBUILD;

Index altered.

SQL> ALTER TABLE T_PART_GLO_IND TRUNCATE PARTITION FOR (TO_DATE('2017-3-5', 'YYYY-MM-DD')) UPDATE INDEXES;

Table truncated.

Elapsed: 00:00:00.07

SQL> SELECT INDEX_NAME, STATUS FROM USER_INDEXES WHERE TABLE_NAME = 'T_PART_GLO_IND';

INDEX_NAME                     STATUS
------------------------------ ----------------
IND_GLO_NAME                   VALID
```

12.1新特性

- 全局索引异步维护

```
SQL> SELECT OBJECT_NAME, SUBOBJECT_NAME, OBJECT_TYPE, TO_DATE(LAST_DDL_TIME, 'YYYY-MM-DD HH24:MI:SS') DDL_TIME
  2  FROM USER_OBJECTS WHERE OBJECT_NAME = 'T_PART_GLO_IND' ORDER BY 4 DESC;

OBJECT_NAME          SUBOBJECT_NAME       OBJECT_TYPE     DDL_TIME
-------------------- -------------------- --------------- -------------------
T_PART_GLO_IND                            TABLE           2018-05-01 21:43:01
T_PART_GLO_IND       SYS_P3495            TABLE PARTITION 2018-05-01 21:43:01
T_PART_GLO_IND       SYS_P3493            TABLE PARTITION 2018-05-01 21:40:47
T_PART_GLO_IND       SYS_P3505            TABLE PARTITION 2018-05-01 21:31:21
T_PART_GLO_IND       SYS_P3504            TABLE PARTITION 2018-05-01 21:31:20
T_PART_GLO_IND       SYS_P3503            TABLE PARTITION 2018-05-01 21:31:19
T_PART_GLO_IND       SYS_P3502            TABLE PARTITION 2018-05-01 21:31:18
T_PART_GLO_IND       SYS_P3501            TABLE PARTITION 2018-05-01 21:31:17
.
.
.
T_PART_GLO_IND       SYS_P3481            TABLE PARTITION 2018-05-01 21:31:01
T_PART_GLO_IND       P1                   TABLE PARTITION 2018-05-01 21:28:52

27 rows selected.
```

12.2新特性

- 自动列表分区

```
SQL> CREATE TABLE T_LIST_AUTO (ID NUMBER, NAME VARCHAR2(30), TYPE VARCHAR2(30))
  2  PARTITION BY LIST (TYPE) AUTOMATIC
  3  (PARTITION P1 VALUES ('TABLE'));

Table created.

SQL> INSERT INTO T_LIST_AUTO VALUES (1, 'V_VIEW', 'VIEW');

1 row created.

SQL> SELECT TABLE_NAME, PARTITION_NAME, HIGH_VALUE FROM USER_TAB_PARTITIONS WHERE TABLE_NAME = 'T_LIST_AUTO';

TABLE_NAME      PARTITION_NAME  HIGH_VALUE
--------------- --------------- ------------------------------
T_LIST_AUTO     P1              'TABLE'
T_LIST_AUTO     SYS_P1366       'VIEW'
```

12.2新特性

- 非分区表在线转换分区表

```
SQL> CREATE TABLE T_ONLINE_PART (ID NUMBER, NAME VARCHAR2(128), CREATED DATE);

Table created.

SQL> INSERT INTO T_ONLINE_PART SELECT ROWNUM, OBJECT_NAME, CREATED FROM DBA_OBJECTS;

94154 rows created.

SQL> COMMIT;

Commit complete.

SQL> ALTER TABLE T_ONLINE_PART MODIFY PARTITION BY RANGE (CREATED)
  2  (PARTITION P1 VALUES LESS THAN (TO_DATE('201512', 'YYYYMM')),
  3  PARTITION P2 VALUES LESS THAN (TO_DATE('201601', 'YYYYMM')),
  4  PARTITION PMAX VALUES LESS THAN (MAXVALUE)) ONLINE;

Table created.
```

18 新特性

- 在线修改分区策略

```
SQL> CREATE TABLE T_PART (ID NUMBER, NAME VARCHAR2(30), CREATED DATE)
  2  PARTITION BY RANGE (CREATED)
  3  (PARTITION P1 VALUES LESS THAN (TO_DATE('2018-1-1', 'YYYY-MM-DD')),
  4  PARTITION P2 VALUES LESS THAN (MAXVALUE));

Table created.

SQL> INSERT INTO T_PART SELECT ROWNUM, SUBSTR(OBJECT_NAME, 1, 30), CREATED FROM DBA_OBJECTS;

74238 rows created.

SQL> COMMIT;

Commit complete.
```

18 新特性

- 在线修改分区策略

```
SQL> CREATE INDEX IND_PART_ID ON T_PART(ID) LOCAL;

Index created.

SQL> CREATE INDEX IND_PART_NAME ON T_PART(NAME);

Index created.

SQL> CREATE INDEX IND_PART_CREATED ON T_PART(CREATED) LOCAL;

Index created.

SQL> ALTER TABLE T_PART MODIFY PARTITION BY HASH (ID) PARTITIONS 8 ONLINE
  2  UPDATE INDEXES (IND_PART_ID LOCAL,
  3  IND_PART_NAME GLOBAL PARTITION BY HASH (NAME) PARTITIONS 4);

Table altered.
```

18 新特性

- 在线修改分区策略

```
SQL> SELECT INDEX_NAME, PARTITION_NAME, STATUS FROM USER_IND_PARTITIONS
  2  WHERE INDEX_NAME IN (SELECT INDEX_NAME FROM USER_INDEXES WHERE TABLE_NAME = 'T_PART') ORDER BY 1, 2;

INDEX_NAME                     PARTITION_NAME                 STATUS
------------------------------ ------------------------------ ----------------
IND_PART_CREATED               SYS_P3153                      USABLE
IND_PART_CREATED               SYS_P3154                      USABLE
IND_PART_CREATED               SYS_P3155                      USABLE
IND_PART_CREATED               SYS_P3156                      USABLE
IND_PART_CREATED               SYS_P3157                      USABLE
IND_PART_CREATED               SYS_P3158                      USABLE
IND_PART_CREATED               SYS_P3159                      USABLE
IND_PART_CREATED               SYS_P3160                      USABLE
IND_PART_ID                    SYS_P3153                      USABLE
IND_PART_ID                    SYS_P3154                      USABLE
IND_PART_ID                    SYS_P3155                      USABLE
IND_PART_ID                    SYS_P3156                      USABLE
IND_PART_ID                    SYS_P3157                      USABLE
IND_PART_ID                    SYS_P3158                      USABLE
IND_PART_ID                    SYS_P3159                      USABLE
IND_PART_ID                    SYS_P3160                      USABLE
IND_PART_NAME                  SYS_P3161                      USABLE
IND_PART_NAME                  SYS_P3162                      USABLE
IND_PART_NAME                  SYS_P3163                      USABLE
IND_PART_NAME                  SYS_P3164                      USABLE
```

18 新特性

- 在线MERGE分区

```
SQL> create table t_merge_partition (id number, name varchar2(30), created date)
  2  partition by range (created)
  3  (partition p1 values less than (to_date('2018-1-1', 'yyyy-mm-dd')),
  4  partition p2 values less than (to_date('2018-2-1', 'yyyy-mm-dd')),
  5  partition p3 values less than (to_date('2018-3-1', 'yyyy-mm-dd')),
  6  partition pmax values less than (maxvalue));

Table created.

SQL> insert into t_merge_partition select rownum, substr(object_name, 1, 30), created from dba_objects;

74272 rows created.

SQL> commit;

Commit complete.

SQL> alter table t_merge_partition merge partitions p1, p2 into partition p2 online;

Table altered.
```

THANKS


HANA
Hadoop
SQL
MySQL
DB2
BigDate
Spark
SQL Server
NewSQL
Oracle
MPP
NoSQL
OLAP

讲师申请
联系电话（微信号）：18612470168

关注“ITPUB”更多
技术干货等你来拿~

与百度外卖、京东、魅族等先后合作系列分享活动

微学堂

让学习更简单

微学堂是以ChinaUnix、ITPUB所组建的微信群为载体，定期邀请嘉宾对热点话题、技术难题、新产品发布等进行移动端的在线直播活动。

截至目前，累计举办活动期数60+，参与人次40000+。

ITPUB学院

ITPUB学院是盛拓传媒IT168企业事业部（ITPUB）旗下
企业级在线学习咨询平台
历经18年技术社区平台发展
汇聚5000万技术用户
紧随企业一线IT技术需求
打造全方式技术培训与技术咨询服务
提供包括企业应用方案培训咨询（包括企业内训）
个人实战技能培训（包括认证培训）
在内的全方位IT技术培训咨询服务

ITPUB学院讲师均来自于企业
一些工程师、架构师、技术经理和CTO
大会演讲专家1800+
社区版主和博客专家500+

培训特色

无限次免费播放
随时随地在线观看
碎片化时间集中学习
聚焦知识点详细解读
讲师在线答疑
强大的技术人脉圈

八大课程体系

基础架构设计与建设
大数据平台
应用架构设计与开发
系统运维与数据库
传统企业数字化转型
人工智能
区块链
移动开发与SEO

联系我们

联系人：黄老师
电　话：010-59127187
邮　箱：edu@itpub.net
网　址：edu.itpub.net
培训微信号：18500940168