

Oracle Technical Note

Partition Table

Troubleshooting 시리즈는 필자가 한국오라클 서버지원팀에서 근무하면서 실제 고객들로부터 많은 문의를 받았던 부분들에 대해 단편적인 해결책이 아닌 보다 근본적으로 심도 있게 정리한 것이다. 각 호에서는 각 호마다 다루는 항목에 대한 기본적인 개념과 메커니즘을 설명한 후 업무 과정에서 발생 빈도가 높은 에러를 처리해 나가는 과정을 설명한다.

지난 호까지는 오라클 메모리 부분, 분산 데이터베이스 등에 대해 다루어 보았고, 이번 호에서는 점점 사용량이 증가하고 있는 Partition Table에 대해 다루어 보기로 한다. 이 Partition Table의 초기 개념은 V7.x의 Partition View로 거슬러 올라갈 수 있으나, Partition View에 비해 DML 부분 등의 추가 부분이 상당히 진전된 개념이다. 이에, 먼저 Partition View의 원리와 그 문제점 등에 대해 점검해 보고, V8.x부터 기능이 추가된 Partition Table에 대해 좀 더 알차게 다루어 보기로 한다.

| 글 | 박경희 | 한국오라클 서버지원팀
| khpark@kr.oracle.com |

Partition View

이 부분에서는 Partition Elimination이 가능하게 된 V7.3. 이상의 Partition View에 대해 다루어 본다. 원래 Partition View는 대용량의 데이터를 다루는 Data Warehousing 환경을 지원하고자 하는 것으로,

- 커다란 테이블의 관리보다 테이블을 분리함으로써 손쉽게 관리
- 커다란 전체 테이블을 읽는 것 Scan 보다 분리된 테이블을 읽도록 유도하기 위한 개념이다 .

Partition View의 사용

Partition View를 사용하기 위해서는

1. InitSID.ora 파일에 Partition_view_enabled=TRUE가 설정되어야 한다
2. Partition View를 만들고자 하는 테이블에 시간이나 지역으로 분리가 가능한 칼럼을 선택하여 Partition Column을 결정한다.
3. 테이블 생성시 Partition Column에 반드시 해당 범위의 Check Constraint를 부여한다.
4. 인덱스 생성은 반드시 Partition Column에 필요한 것은 아니나, 생성시 모든 테이블이 반드시 동일한 인덱스를 가져야 한다.
5. 각 테이블에 ANALYZE를 실시하여 정확한 통계를 수집한다.
6. 모든 테이블을 UNION ALL한 Partition View를 생성한다.

Partition View 예제

```
create view sales as
  select * from jan_sales
  union all
  select * from feb_sales
  ...
  union all
  select * from dec_sales
```

이 Partition View는 Jan_sales, Feb_sales, Mar_sales, ..., Dec_sales의 12개 테이블을 이용하여 View를 생성한 것이다. 이때 각 테이블은 반드시 같은 칼럼 이름, 칼럼 데이터 타입, 같은 인덱스로 구성되어 있어야 한다.

또한 반드시 파티션되는 칼럼에는 CHECK Constraint가 있어야 한다. 즉, Jan_sales Table은 Date 칼럼에 JAN 1 ~ JAN 31의 데이터만 입력된다는 조건이어야 한다는 CHECK Constraint 가 있어야 한다.

그리고 이 기본 테이블을 이용한 UNION VIEW가 생성되어야 한다.

관리 및 사용 효율

만일 전체 Sales 테이블이 각 월별로 나뉘지 않고 1년의 전체 데이터로 구성되어 있다면, 데이터 관리자는 해당 월의 모든 데이터를 이 테이블에 올리기 위해 매달 해당 인덱스를 삭제한 후(Load하는 데이터 양이 크기 때문에) 데이터를 올리고 다시 인덱스를 만드는 과정을 거쳐야 하며, 이 작업 동안 해당 테이블은 사용하지 못하는 결과를 얻게 된다.

반면에, Partition View를 사용하게 되면, 데이터 관리자는 매달 새로운 데이터를 새로운 테이블에 넣고, 새로운 인덱스를 생성한 후 새로운 테이블을 포함한 UNION-ALL View를 새롭게 구성하면 된다. 이 경우 인덱스 크기가 상대적으로 작기 때문에 데이터를 올리는(Load) 시간뿐만 아니라 인덱스를 생성하는 시간이 적게 걸리는 효과를 얻을 수 있다.

Performance

아무리 관리 기법이 뛰어나도, Query 효과가 없다면 의미가 없을 것이다.

그러나, UNION -ALL의 Partition View는 최소한 1개의 테이블을 Query하는 것과 속도가 같거나 그보다 더 빠르다 . 이처럼 Performance 효과는

- Partition Elimination 기능
- Parallel 수행 지원

때문에 가능하다.

| 예제 |

```
select order_id,comp_id,date, revenues from sales
where comp_id >20000 and
sales_date between '10-july-2000' and '15-july-2000';
```

이 경우 오라클은 1월과 2월의 파티션만 접근하여 데이터를 가져오는데, 이를 'Partition Elimination'이라 하며(이는 파티션 칼럼에만 영향을 미치는데, 여기서는 sales_date 칼럼이다), Parallel Query Option을 사용하여 그 Degree만큼의 수행 속도를 높일 수 있다.

```
<Execution Plan>
Rows Execution Plan
-----
0 SELECT STATEMENT GOAL: CHOOSE
100 VIEW OF 'SALES'
100 UNION-ALL (PARTITION)
0 FILTER
0 TABLE ACCESS OF 'JAN_SALES'
0 INDEX (RANGE SCAN) OF 'JAN_SALES_IDX'
0 FILTER
0 TABLE ACCESS OF "FEB_SALES"
0 INDEX (RANGE SCAN) OF "JAN_SALES_IDX"
0 FILTER
```

```

0  TABLE ACCESS OF 'MAR_SALES'
0  INDEX (RANGE SCAN) OF 'MAR_SALES_IDX'
0  FILTER
0  TABLE ACCESS OF 'APR_SALES'
0  INDEX (RANGE SCAN) OF 'APR_SALES_IDX'
0  FILTER
0  TABLE ACCESS OF 'MAY_SALES'
0  INDEX (RANGE SCAN) OF 'MAY_SALES_IDX'
0  FILTER
0  TABLE ACCESS OF 'JUN_SALES'
0  INDEX (RANGE SCAN) OF 'JUN_SALES_IDX'
100 TABLE ACCESS OF 'JULY_SALES'
150 INDEX (RANGE SCAN) OF 'JULY_SALES_IDX'
0  FILTER
0  TABLE ACCESS OF 'AUG_SALES'
0  INDEX (RANGE SCAN) OF 'AUG_SALES_IDX'
0  FILTER
0  TABLE ACCESS OF 'SEP_SALES'
0  INDEX (RANGE SCAN) OF 'SEP_SALES_IDX'
0  FILTER
0  TABLE ACCESS OF 'OCT_SALES'
0  INDEX (RANGE SCAN) OF 'OCT_SALES_IDX'
0  FILTER
0  TABLE ACCESS OF 'NOV_SALES'
0  INDEX (RANGE SCAN) OF 'NOV_SALES_IDX'
0  FILTER
0  TABLE ACCESS OF 'DEC_SALES'
0  INDEX (RANGE SCAN) OF 'DEC_SALES_IDX'

```

이처럼 Partition Column이 BETWEEN~ AND~로 비교되었을 때는 접근이 필요 없는 테이블은 Scan하지 않는 Filter Operation이 사용되며, 만일 Where 조건에 Partition Column이 나오지 않는 경우는 Filter 기능이 사용되지 않는 UNION-ALL View 형태로 Access Plan이 사용되어 Performance 저하가 발생된다.

이처럼 부득이한 경우는 ROWNUM 등의 Function 사용으로 TABLE ACCESS를 제한한다.

제약 사항

Partition View의 제약 사항은 다음과 같다.

- 수작업을 통한 파티션 관리
데이터를 파티션 테이블과 인덱스에 분류해야 하는 작업을 사용자가 하여야 하며, 파티션간의 데이터 이동(split, move ...)을 위한 작업 수행시에는 Export/Import 또는 SQL 문으로 하여야 한다.
- DML문의 사용 제한
UNION ALL View에 DML(Insert, Update, Insert) 문장을 사용할 수 없어, 이를 위해서는 파티션을 이루는 기본 테이블에 하여야 한다.
이를 손쉽게 하기 위해 DYNAMIC SQL을 이용하거나 일어날수 있는 모든 경우를 IF-ELSE를 이용한 Logic으로 해결하는 방안 등을 사용하나, 이 경우도 Parsing Overhead나 Partition을 위한 테이블이 추가될 때마다 애플리케이션 프로그램이 변경되어야 하는 단점이 있다.
- Performance 문제
Optimizer가 충분히 인덱스를 활용할 수 없으며, UNION ALL View에 대한 SQL 문장 수행시 동일한 구조의 Table Definition이 중복되어 SGA에 올라오게 되어 SQL Parsing의 낭비가 따른다.
- DDL문(Create, Alter..)의 제한
UNION ALL View에 대해서는 Global Index나 Referential Integrity Constraint를 정의할 수 없다.
- Data Load시의 제한
UNION ALL View에 대해서는 Direct Load를 수행할 수 없다.

Partition Table

기존 Partition View의 제한 사항을 해소하는 개념인 파티션 오브젝트가 V8.0 이상부터는 지난 호에 소개했던 Distributed Database Option처럼 한 부분의 Option으로 분류되어 제공되고 있다.

이는 오라클 제품을 처음 설치시 선택하여야만 사용이 가능함을 일컫는다.

한 테이블당 가능한 파티션은 이론적으로 65535이며, 실제적으로는 1000개까지 테스트가 이루어진 상태이다. 이 파티션 테이블을 만들면서 생성되는 각 파티션은 다른 Segment(Tablespace)에 저장하여 아래와 같은 이점을 활용할 수 있다.

- 시스템 장애 발생시 데이터의 손상 정도를 완화할 수 있다.
- 각 Partition별로 독립적으로 Back Up과 Recovery가 가능하다.
- Disk Drive에 따라 Partition을 Mapping함으로써 I/O Load Balancing이 가능하다.

그리고 Partition Table의 제한 사항은 다음과 같다.

- Partitioning 지원 제한
오라클은 V8.0 이상부터 테이블과 인덱스에 대해 Partitioning을 지원하고 있으며, Clustered Table이나 Clustered Index, Snapshot에 대해서는 Partitioning을 지원하지 않는다.

- 데이터 타입 제한
파티션 테이블은 LONG, LONGRAW 데이터 타입은 불가능하나, 이를 대신하는 LOB(BLOB, CLOB, NCLOB, BFILE) 데이터 타입은 파티션이 가능하다.
- 비트맵 인덱스의 제한
파티션 테이블에 대해 비트맵(Bitmap Index)의 생성이 가능하나, 이는 Local Index로만 생성되어야 하며, Global Index로는 생성할 수 없다.

인덱스

Oracle8의 새로운 기능인 파티션은 인덱스에도 적용되며, Partition Table에서의 인덱스는 Local Index, Global Index로 나눌 수 있다. 우선 Local Index는 기본 테이블 중 하나의 파티션만을 참조하는 경우이며, Global Index는 기본 테이블의 모든 파티션을 참조하는 인덱스이다.

이 파티션되는 인덱스는 생성되는 형태에 따라 몇 가지로 나눌 수 있다 .

LOCAL INDEX

로컬 인덱스란 인덱스를 생성한 테이블과 파티션된 인덱스가 동일하게 파티션된 (Equi-Partition) 경우를 나타낸다. 즉, 인덱스와 테이블은 같은 칼럼에 의해 파티션되며, 하나의 인덱스 파티션이 테이블 파티션 하나와 대응되며, 대응되는 인덱스 파티션과 테이블 파티션은 각각 같은 범위를 갖게 된다. 결국 특정한 하나의 인덱스에 포함된 모든 Key들은 하나의 테이블 파티션 내의 데이터만을 가리키게 된다. 파티션되는 인덱스는 다시 다음과 같이 Prefixed와 Non-Prefixed로 나누어진다.

Local Prefixed Index

인덱스의 맨앞에 위치한 칼럼에 의해 파티션되는 것이며, 뒤에 언급될 Non-Prefixed Index는 인덱스에서 맨앞의 칼럼을 제외한 다른 칼럼에 의해 파티션된 경우이다.

Local Prefixed Index는 Unique/Non-Unique를 모두 허용한다.

Local Prefixed Index는 다음과 같이 생성할 수 있다.

```
CREATE TABLE dept
  (deptno NUMBER NOT NULL,
  dname VARCHAR2(10) NOT NULL,
  loc VARCHAR2(14))
PARTITION BY RANGE (deptno)
(PARTITION part1 VALUES LESS THAN (30),
PARTITION part2 VALUES LESS THAN (MAXVALUE));

CREATE INDEX deptloc1_idx ON dept(deptno) LOCAL;
```

Local non-Prefixed Index

인덱스의 맨앞에 있는 칼럼이 테이블의 파티션을 이루는 키가 아닌 경우 Non-Prefixed Index가 된다. 또한 Local Non-Prefixed Index가 Partitioning Key의 Subset이 아닌 경우에는 Unique Local Non-Prefixed Index를 생성할 수 없

으며, 이는 다음처럼 생성할 수 있다.

```
CREATE INDEX deptloc2_idx ON dept(loc) LOCAL;
```

이러한 Non-Prefixed Index는 Historical한 데이터를 보관하는 테이블의 경우 유용하다. 즉, 날짜에 따라 테이블과 인덱스의 파티션은 이루어지고, 인덱스는 별도의 사원 번호나 제품 번호와 같은 칼럼을 이용하여 생성하는 경우이다.

또한 만일 테이블과 인덱스가 같은 칼럼에 대해 같은 값으로 파티션되어(Equi-Partitioning) 있다면 아래와 같은 이점을 가질 수 있다.

- 기초 테이블에 SPLIT PARTITION을 제외한 Add, Drop의 명령이 수행되는 경우 하나의 인덱스 파티션만이 영향을 받게 된다. 파티션된 테이블이 Local Index만을 가지고 있다면, 이러한 명령들이 수행되는 중에는 해당 파티션만을 사용하지 못하고 다른 파티션은 사용이 가능하다. 즉, Local Index는 각 파티션이 독립적으로 수행이 가능하다는 것을 보장한다.
- Local Index를 사용하여 Equi-Partitioned된 경우에는 보다 나은 Query Access Plan을 생성할 수 있다. Local Index는 Tablespace Incomplete Recovery가 손쉽게 이루어지도록 한다. 시간 단위로 Recovery를 수행하는 경우 테이블과 인덱스는 동시에 Recover되어야 하며, 이를 완벽하게 지원하기 위해서는 Local Index를 사용하여야 한다.

GLOBAL INDEX

Global Prefixed Index

Global Index는 테이블과 다르게 파티션되는 경우이다. 즉 테이블과 같은 칼럼으로 파티션되지만, 그 범위가 틀리거나 혹은 다른 칼럼으로 파티션이 이루어지는 경우이다.

하나의 인덱스 파티션에 있는 모든 인덱스는 모두 하나의 테이블 파티션에 속하게 되지 않고, 두 개 이상의 파티션에 나누어 있을 수 있다.

일반적으로 Global Index는 기초 테이블과 Equi-Partitioned되도록 생성하지는 않지만, Equi-Partitioned Global Index를 생성한 경우에 오라클은 이를 이용하여 Query Plan을 생성하거나 파티션 관리를 위한 명령이 실행되는 경우에도 Equi-Partitioning의 이점을 활용하지 않는다.

따라서 Equi-Partitioned로 인덱스를 생성하고자 하는 경우에는 반드시 LOCAL로 생성하여야 한다. 또는 Global Index의 가장 높은 파티션은 파티션을 분리하는 범위를 나타내는 부분에 MAXVALUE를 가져야 한다. 이렇게 해서 기초가 되는 테이블의 모든 데이터가 인덱스에 모두 반영되는 것을 보장하게 된다.

```
CREATE INDEX dept_idx ON dept(dname)
GLOBAL PARTITION BY RANGE (dname)
(PARTITION p1 VALUES LESS THAN ('N'),
PARTITION p2 VALUES LESS THAN (MAXVALUE));
```

Global Index 생성시 인덱스 칼럼의 맨처음 칼럼을 사용하여 파티션되면, 생성된 인덱스는 Global Prefixed Index이며, 다른 칼럼을 사용하여 파티션 되는 경우에는 Global Non-Prefixed Index이지만, 오라클에서는 Global Prefixed Index만을 지원한다. Global Index는 항상 인덱스의 맨앞 칼럼 값만을 이용하여 파티션된다.

Global Prefixed Index는 Unique/Non-Unique로 생성될 수 있으며, Non-Partitioned Index는 Global Index로 취급된다. 그러면, Global Index가 어느 때 유용한지 다음의 예를 살펴보자.

만일 DATE 칼럼으로 파티션된 커다란 테이블이 있다고 가정한다. 그러나 주로 VCOL이란 칼럼으로 자주 결과를 얻고자 하는 경우, 우리는 VCOL을 이용한 Local Non-Prefixed Index와 VCOL을 이용한 Global Prefixed Index를 만들 수 있으며, 해당 결과를 얻기 위해 VCOL로 파티션된 Unique Global Index를 사용하는 것이 훨씬 좋은 Performance를 얻을 수 있다.

만일 Unique하지 않는 경우는 Parallel Degree의 정도에 따라 Local과 Global의 Performance가 다를 수 있다(이는 Global Index의 관리 비용이 비싸기 때문이다).

Primary Key를 설정하는 경우

파티션 테이블에 Primary Key Constraint를 지정하여 사용하고자 추가하면, 이 때 내부적으로 생성되는 인덱스는 비록 테이블의 데이터가 파티션되어 있다 하더라도, 파티션되지 않고, 하나의 세그먼트로 생성되어 Global Index로 취급되어 불편하다. 이러한 경우는 Local Index를 사용하도록 하기 위해 Local Index를 생성 후 ALTER TABLE 명령어를 이용하여 Constraint를 추가하도록 한다

이때 미리 생성하는 인덱스는 Local이든 Global이든 상관 없으나 반드시 Prefixed Index로 생성되어야 한다(Non-prefixed Index는 하나의 Index Key 값에 해당하는 데이터가 여러 테이블 파티션에 존재할 수 있다는 것을 의미한다).

1. Partitioned Table 생성

```
create table test_a
(col1 number,
col2 number,
col3 varchar2(20))
Partition by range (col1, col2)
(Partition part_test_a_1 values less than (10, 100) tablespace ts0,
Partition part_test_a_2 values less than (20, 200) tablespace ts1,
Partition part_test_a_3 values less than (30, 300) tablespace ts2,
Partition part_test_a_4 values less than (40, 400) tablespace ts3);

column Table_name format a12
column Partition_name format a13
column High_Value format a15
column PP format 99
column TBSN format a5
```

```

select table_name , Partition_name , high_value ,
Partition_position pp, tablespace_name tbsn, initial_extent Ini_Extent ,
next_extent NEXT
from user_tab_Partitions
order by pn

```

Table_name	Partition_name	HighValue	PP	TBSN	Ini_Extent	NEXT
TEST_A	PART_TEST_A_1	10, 100	1	TS0	20480	20480
TEST_A	PART_TEST_A_2	20, 200	2	TS1	20480	20480
TEST_A	PART_TEST_A_3	30, 300	3	TS2	20480	20480
TEST_A	PART_TEST_A_4	40, 400	4	TS3	20480	20480

2. 먼저 Local Prefixed Index 생성

```

create Index ix_test_a on test_a(col1, col2)
Local
(Partition in_test_a_1 tablespace ts0,
Partition in_test_a_2 tablespace ts1,
Partition in_test_a_3 tablespace ts2,
Partition in_test_a_4 tablespace ts3);

```

```

column Index_name format a12
column Partition_name format a13
column hv format a10
column pp format 99
column Status format a8
column Initial_ex format 999999
column NE format 999999
column tbsn format a5

```

```

select Index_name , Partition_name , high_value hv,
Partition_position pp, status Status, tablespace_name tbsn,
initial_extent initial_ex, next_extent NE
from user_ind_Partitions
order by idxn, pn

```

Index_name	Partition_name	HV	PP	Status	TBSN	Initial_ex	NE
IX_TEST_A	IN_TEST_A_1	10, 100	1	USABLE	TS0	20480	20480
IX_TEST_A	IN_TEST_A_2	20, 200	2	USABLE	TS1	20480	20480
IX_TEST_A	IN_TEST_A_3	30, 300	3	USABLE	TS2	20480	20480
IX_TEST_A	IN_TEST_A_4	40, 400	4	USABLE	TS3	20480	20480

```

column Index_name format a12
column Partitioning_type format a10
column pc format 999

```

```
column pkc format 999
```

```
select Index_name, Partitioning_type pt, Partition_count pc,  
Partitioning_key_count pkc, Locality, alignment  
from user_part_Indexes
```

INDEX_NAME	PT	PC	PKC	LOCAL	ALIGN
IX_TEST_A	RANGE	4	2	LOCAL	PREFIXED

PT : Partition Type, PC : Partition Count, PKC : Partition Key Count
Local : Locality, ALIGN : Alignment

3. Primary Key Constraint 생성

```
SQL> alter table test_a add constraint pk_test_a  
2 Primary key ( col1, col2 );
```

```
Table altered.
```

4. 생성되어 있던 IX_TEST_A Index를 Drop하여 본다.

```
SQL> drop Index ix_test_a;  
drop Index ix_test_a  
*  
ERROR at line 1:  
ORA-02429: cannot drop Index used for enforcement of unique/Primary  
key
```

Ora-2429 Error에서 볼 수 있듯이 기존에 생성되어 있던 IX_TEST_A Index가 Primary Key Constraint와 연계되어 사용되는 것을 확인할 수 있다.

이처럼 Primary Key Constraint를 생성할 때 Primary Key로 사용되는 Column들로 이루어진 인덱스가 이미 존재하는 경우, 기존에 존재하는 인덱스를 사용하게 된다.

이때 사용되는 인덱스는 Unique/Non-unique, Local/Global Partitioned Index 여부에 관계없이 사용하게 된다.

위의 예에서 IX_TEST_A Index는 Non-unique Index로 생성되었으며, Primary Key Constraint를 생성할 때 이 인덱스가 사용되었다. 그러나 이 경우 데이터가 Unique한 경우에만 적용되며, 그렇지 않은 경우에는 ora-2437 : Primary Key Violated Error가 발생하게 된다.

Primary Key Constraint를 Drop할 때, 생성되어 있던 인덱스가 Unique Index였다면 함께 Drop되게 되며, 이는 해당 인덱스가 Local이든 Global이든 동일하게 적용된다.

PARTITION TABLE의 관리를 위한 COMMAND

일반 테이블 Partition하기

이를 위해서는 Export/import Method, Subquery를 이용해 데이터를 입력하는 방법 그리고 Partition Exchange 명령어를 이용하는 방법이 있다.

Export/import 방법

1. 테이블을 Export한다.

```
exp usr/pswd tables=numbers file=exp.dmp
```

2. 백업받은 테이블을 제거한다.

```
drop table numbers;
```

3. 파티션 테이블을 생성한다.

```
create table numbers (qty number(3), name varchar2(15))  
Partition by range (qty)  
(Partition p1 values less than (501),  
Partition p2 values less than (maxvalue));
```

4. ignore=y를 사용하여 데이터를 Import한다.

```
imp usr/pswd file=exp.dmp ignore=y
```

Subquery를 이용한 방법

1. 파티션 테이블을 생성한다.

```
create table partbl (qty number(3), name varchar2(15))  
Partition by range (qty)  
(Partition p1 values less than (501),  
Partition p2 values less than (maxvalue));
```

2. Subquery를 이용하여 파티션 테이블에 데이터를 입력한다.

```
insert into partbl (qty, name) select * from origtbl;
```

Partition Exchange 명령어를 사용하는 방법

ALTER TABLE EXCHANGE PARTITION은 파티션 테이블을 일반 테이블로, 또 파티션되어 있지 않는 테이블을 파티션 테이블로도 변경시킬 때 사용 가능하다.

이를 위해서는 중간의 Dummy Table을 이용한다.

```
SQL> CREATE TABLE p_emp  
2 (sal NUMBER(7,2))  
3 PARTITION BY RANGE(sal)  
4 (Partition emp_p1 VALUES LESS THAN (2000),
```

```
5 Partition emp_p2 VALUES LESS THAN (4000));
```

```
Table created.
```

```
SQL> SELECT * FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL
7369	SMITH	CLERK	7902	17-DEC-80	800
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600
7521	WARD	SALESMAN	7698	22-FEB-81	1250
7566	JONES	MANAGER	7839	02-APR-81	2975
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250
7698	BLAKE	MANAGER	7839	01-MAY-81	2850
7782	CLARK	MANAGER	7839	09-JUN-81	2450
7788	SCOTT	ANALYST	7566	19-APR-87	3000
7839	KING	PRESIDENT		17-NOV-81	5000
7844	TURNER	SALESMAN	7698	08-SEP-81	1500
7876	ADAMS	CLERK	7788	23-MAY-87	1100
7900	JAMES	CLERK	7698	03-DEC-81	950
7902	FORD	ANALYST	7566	03-DEC-81	3000
7934	MILLER	CLERK	7782	23-JAN-82	1300

```
14 rows selected.
```

```
SQL> CREATE TABLE dummy_y as SELECT sal  
FROM emp WHERE sal<2000;
```

```
Table created.
```

```
SQL> CREATE TABLE dummy_z as SELECT sal FROM emp WHERE sal  
BETWEEN 2000 AND 3999;
```

```
Table created.
```

```
SQL> alter table p_emp exchange Partition emp_p1  
with table dummy_y;
```

```
Table altered.
```

```
SQL> alter table p_emp exchange Partition emp_p2  
with table dummy_z;
```

```
Table altered.
```

PARTITION TABLE의 관리를 위한 COMMAND

여러 개의 파티션으로 분리된 테이블 중

일부의 파티션만 가진 테이블 생성하기

데이터를 Export한 후 필요한 파티션으로 이루어진 테이블을 생성한다.

```
create table year (col1 date) Partition by range (col1)
(Partition october values less than ('01-NOV-1999) tablespace october,
Partition november values less than ('01-DEC-1999) tablespace november,
Partition december values less than (maxvalue) tablespace december);
```

데이터를 Import한다

```
>imp username/password file=expdat.dmp fromuser=<owner>
touser=<owner>
tables=(year:october,year:november,year:december)
```

Partition을 Add하는 방법

아래와 같이 Partition Table이 생성되어 있다고 가정한다.

```
SQL> create table part_tbl
( in_date char(8) Primary key ,
empno number,
ename varchar2(20),
job varchar2(20) )
Partition by range (in_date)
(Partition part_tbl_03 value less then ('20000331')
tablespace pts_03,
Partition part_tbl_04 value less then ('20000430')
tablespace pts_04,
Partition part_tbl_05 value less then ('20000531')
tablespace pts_05,
Partition part_tbl_06 value less then ('20000630')
tablespace pts_06,
Partition part_tbl_07 value less then ('20000731')
tablespace pts_07,
Partition part_tbl_08 value less then ('20000831')
tablespace pts_08,
Partition part_tbl_09 value less then ('20000930')
tablespace pts_09,
Partition part_tbl_10 value less then ('20001031')
tablespace pts_10 );
```

11월과 12월에 대해 Partition을 Add하고 싶은 경우 다음과 같이 할 수 있다.

```
SQL> alter table part_tbl add Partition part_tbl_11
      values less than ('20001130') tablespace pts_11;
```

```
SQL> alter table part_tbl add Partition part_tbl_12
      values less than ('20001231') tablespace pts_12;
```

PARTITION TABLE의 관리를 위한 COMMAND

특정 Partition을 삭제하는 방법

8월에 해당하는 Partition을 없애고 싶은 경우는 다음과 같이 실행한다.

```
SQL> alter table part_tbl drop Partition part_tbl_08;
```

Drop된 후에 새로 8월에 해당하는 데이터가 입력되면, 9월의 Partition이 less than ('20000930')으로 되어 있으므로 9월에 해당하는 Partition에 저장된다.

Partition을 나누는 방법

1월, 2월에 해당하는 Partition을 생성하려면 Partition을 Add하는 것으로는 불가능하고 기존의 Partition에서 Split해야 한다.

```
SQL> alter table part_tbl split Partition part_tbl_03
      at ('20000229')
      into (Partition part_tbl_02 tablespace pts_02,
           Partition part_tbl_03_1 tablespace pts_03);
```

위와 같이 하면, 기존의 Partition에서 2월 29일을 기준으로 2월과 3월로 Partition이 나뉜다. 그리고 나서 다시 Split해야 한다.

```
SQL> alter table part_tbl split Partition part_tbl_02
      at ('20000131')
      into (Partition part_tbl_01 tablespace pts_01,
           Partition part_tbl_02_1 tablespace pts_02);
```

Partition Name을 변경하는 방법

Partition Name을 바꾸고 싶다면, 다음과 같이 실행한다.

```
SQL> alter table part_tbl rename Partition part_tbl_02_1 to part_tbl_02;
SQL> alter table part_tbl rename Partition part_tbl_03_1 to part_tbl_03;
```

Partition의 Tablespace를 옮기는 방법

Partition part_tbl_10을 저장하는 Tablespace를 pts_10에서 pts_10_1로 바꾸고 싶은 경우 아래와 같은 Command를 사용한다.

```
SQL> alter table part_tbl move Partition part_tbl_10
      tablespace pts_10_1 nologging;
```

특정 Partition의 Data를 Truncate하는 방법

Partition의 Data를 모두 삭제하려면 Truncate하는 방법을 사용할 수가 있는데, Truncate는 Rollback이 불가능하며 특정 Partition 전체를 삭제하므로 주의하여 사용하여야 한다.

```
SQL> alter table part_tbl truncate Partition part_tbl_02;
```

Partition Table의 물리적인 속성 변경

Partition Table은 특정 Partition의 속성만 변경할 수 있고, Table의 속성을 변경하여 전체 Partition에 대해 동일한 변경을 할 수 있다.

```
SQL> alter table part_tbl storage (next 10M);  
-> part_tbl 의 모든 Partition의 Next 값이 변경된다.
```

```
SQL> alter table part_tbl modify Partition part_tbl_05  
storage ( maxextents 1000 );  
-> part_tbl_05 Partition의 Maxextents 값만 변경한다.
```

Index의 관리

위와 같이 Partition Table 관련 작업을 한 후에는 Table에 걸려 있는 Local (Partitioned) Index나 Global Index를 반드시 Rebuild해 주어야 한다.

특정 Partition의 Index를 Rebuild하려면,

```
SQL> alter Index ind_part_tbl rebuild Partition i_part_tbl_02;
```

그리고 Global Index를 Rebuild하려면,

```
SQL> alter Index part_tbl_pk rebuild;
```

으로 처리한다.

Backup과 Recovery 기능

여기서는 Partitioned Table에 대한 Backup(Export/Import) 내용을 예를 통해서 살펴보도록 한다.

Export

Partitioned Table을 위한 Export는 다음과 같이 분류할 수 있다.

Table-level Export

기존의 Table Export처럼 Table 전체를 Export하는 경우이다. 이는 Emp Table(Partitioned 또는 Non-Partitioned) 전체를 Export하는 경우이다.

```
예) $ exp scott/tiger tables=emp file=emp.dmp
```

Partition-level Export

이는 Partition Table의 일부 Partition만을 Export하는 것으로, Full Mode의 Export시에는 사용하지 못하고, Table 단위의 Export시에만 가능하다 .

```
예) $ exp scott/tiger tables=emp:px file=exp_par.dmp
      '.'을 이용하여 Partition 이름을 지정하며 이 경우 Emp Table의 px
      Partition만을 Export.
```

또한 다음과 같이 두 가지 경우를 Level을 혼용하여 사용하는 것도 가능하다.

```
예) $ exp scott/tiger tables=emp:px file=exp_par.dmp
      '.'을 이용하여 Partition 이름을 지정하며 이 경우 Emp Table의 px
      Partition만을 Export.
```

또한 다음과 같이 두 가지 경우를 Level을 혼용하여 사용하는 것도 가능하다.

```
예) $ exp scott/tiger tables=(emp:px, sales) file=both.dmp
      Sales Table은 전부를, Emp Table에서는 px Partition만을 Export.
```

Import

Export와 마찬가지로 Import도 다음의 두 가지로 분류할 수 있다.

Table-level Import

- Partitioned 또는 Non-Partitioned Table 전체를 Import한다.
- 모든 Import Mode (full, user, table)에서 사용된다.

```
예) $ imp scott/tiger file=wookpark.dmp tables=emp
      emp table(Partitioned 또는 non-Partitioned) 전체를 Import.
```

Partition-level Import

- Export Dump File에서(full, user, table 중 어떠한 Mode를 이용하여 Export했건간에) Partitioned Table의 일부 Partition만을 Import한다.
- Table Import Mode에서만 사용 가능하다.

```
예) $ imp scott/tiger file=wookpark.dmp tables=emp:px
      emp table의 px Partition만을 Import.
      '.'을 이용하여 Partition을 지정.
```

테이블 단위의 Import시 우선 Table Creation 문장을 수행하고 Row Insert문을 수행하는 것과 마찬가지로, Partition-level Import도 우선 Partitioned Table의 생성 문장을 수행하고 Row Insert문을 수행하게 된다.

따라서 ignore=y option 등을 적절히 사용하면, Non-Partitioned Table과 Partitioned Table간의 변경, Partitioned Table의 구조 변경 등을 수행할 수 있게 된다. 다음에는 그 중 몇 가지 예이다.

파티션되어 있지 않는 테이블을 Exp, Imp를 이용하여 파티션하는 예

a. 파티션되어 있지 않는 테이블을 Export한다.

```
$ exp scott/tiger file=wookpark.dmp tables=emp
```

b. 해당 Table을 Drop한다.

```
SQL> drop table emp
```

c. 파티션 테이블을 생성한다

```
SQL> create table emp (  
    empno number(4) not null,  
    ... )  
    Partition by range (empno)  
    (Partition emp1 values less than (1000) tablespace ts1,  
    Partition emp2 values less than (2000) tablespace ts2,  
    Partition emp3 values less than (3000) tablespace ts3) ;
```

d. Import한다.

```
$ imp scott/tiger file=wookpark.dmp tables=emp ignore=y
```

Partitioned Table의 Partition들을 exp, imp를 이용하여 Merge하는 예

a. Merge의 대상이 되는 Partition을 Export한다.

```
$ exp scott/tiger file=wookpark.dmp tables=emp:emp2
```

b. Merge의 대상이 되는 Partition을 'alter table...' 문장으로 Drop한다.

```
SQL> alter table emp drop Partition emp2 ;
```

c. Import한다.

```
$ imp scott/tiger file=wookpark.dmp tables=emp:emp2 ignore=y
```

이후 emp Table을 확인하면, emp2 Partition에 있던 Data가 emp3 Partition에 Merge되어 있음을 확인할 수 있다.

Toubleshooting

ORA-14097

ALTER TABLE EXCHANGE PARTITION 명령어를 실행할 때 두 테이블간의 데이터 타입 등이 같지 않기 때문에 발생하는 에러이다.

```
ORA-14097 : Clumn type or size mismatch in ALTER TABLE EXCHANGE PARTITION.
```

```
* Cause : The corresponding columns in the tables specified in the ALTER TABLE EXCHANGE PARTITION are of different type or size.
```

```
* Action : Ensure that the two tables have the same number of columns with the same type and size. on ALTER TABLE EXCHANGE PARTITION.
```

```
SQL> create table ptab (  
  2 c1 integer,  
  3 c2 varchar2(20))  
  4 Partition by range (c1)  
  5 (Partition ptab_10 values less than (10),  
  6 Partition ptab_20 values less than (20),  
  7 Partition ptab_max values less than (maxvalue));  
Table created.
```

```
SQL> create table etab (  
  2 c1 integer,  
  3 c2 varchar2(20),  
  4 c3 integer);  
Table created.
```

이 있다고 가정할 때 다음을 실행하고자 하는 경우

```
SQL> alter table ptab  
  2 exchange Partition ptab_20 with table etab;  
exchange Partition ptab_20 with table etab  
  *  
ERROR at line 2:  
ORA-14097: column type or size mismatch in ALTER TABLE EXCHANGE PARTITION
```

이처럼 ora-14097 오류가 발생한다. 이 경우는

```
SQL> alter table etab set unused column c3;  
Table altered.  
SQL> alter table etab drop unused columns;  
Table altered.
```

로 두 테이블간의 일치하지 않는 칼럼을 Drop 후 실행한다.

```
SQL> alter table ptab
2 exchange Partition ptab_20 with table etab;
Table altered.
```

ORA-14038

ORA-14038 : GLOBAL Partitioned Index must be Prefixed.

Global non-Prefixed Indexes는 생성할 수 없다. 이는 인덱스 생성시 항상 인덱스 맨앞의 칼럼으로 파티션됨을 의미한다.

ORA-376 ERROR, ORA-14404 ERROR

Partitioned Table의 Partition이 포함되어 있는 데이터 파일이 OS Level에서 삭제된 경우, 해당 데이터 파일을 Offline Drop하고 Open한 후 해당 Table을 Access하고자 하는 경우 다음과 같은 Error가 유발된다.

ORA-00376 : File cannot be read at this time

* Cause : An attempt was made to read from a file that is not readable.
The most likely cause is that the file is off line.

* Action : Check the state of the file. Bring the file online, if necessary.

그리고, 해당 Tablespace를 Drop하고자 하면 다음과 같은 Error가 유발된다.

ORA-14404 : Partitioned table contains Partitions in a different tablespace.

* Cause: An attempt was made to drop a tablespace which contains tables

whose Partitions are not completely contained in this tablespace.

* Action: Find tables with Partitions which span the tablespace being dropped and some other tablespace(s). Drop these tables or move Partitions to a different tablespace.

이 경우 다음과 같은 절차로 조치 가능하다.

예를 들어, DEPT라는 Partitioned Table 중에서 PART2라는 Partition이 존재하는 TS_PART2의 Datafile이 유실되었으며, 해당 Datafile을 Offline Drop한 후 Open하였다고 가정한다.

Troubleshooting

1. 가장 간단하게 해당 Partitioned Table 전체를 Drop한 후 Recreate한다.

- drop table DEPT ;
- drop tablespace TS_PART2 including contents ;
- tablespace recreate.
- table rebuild.

이 과정의 문제점은 전체 Partitioned Table을 복구해야 하므로 시간이 오래 소요된다는 것이다.

2. 해당 Partition만을 Drop후 재생성한다.

- alter table DEPT drop Partition PART2 ;
- drop tablespace TS_PART2 including contents ;
- tablespace recreate.
- add Partition or split Partition.
- 해당 Partition만의 Data를 Reload.

Partitioned Table 전체 Data를 Reload하는 것보다 짧은 시간이 소요된다.

3. Temporary Table을 생성하여 Exchange한다.

- DEPT 테이블과 동일한 구조의 Dummy Table을 다른 Tablespace에 생성한다.

```
create table DEPT_TEMP as select * from DEPT where 1=2;
```

- 이 Table을 문제의 Partition과 Exchange한다.

```
alter table DEPT
exchange Partition PART2 with table DEPT_TEMP
without validation ;
```

- drop tablespace TS_PART2 including contents ;
- 새로운 Partition에 필요한 Data를 Load한다. 이때, 제3의 테이블에 Load한 후 동일한 방법으로 Exchange를 하여도 무방하다.

Index Unusable되는 경우

Non-Partitioned Index나 Partitioned Index의 파티션은 일부 오퍼레이션에 의해 인덱스가 Unusable 상태가 될 수 있다. 이처럼 Unusable 상태가 된 인덱스나 인덱스 파티션을 SELECT하거나 DML을 시도하면 오류가 발생하게 된다. 그러나 인덱스가 Unusable에 빠진 인덱스를 읽어야 하는 파티션을 제외한 다른 파티션만을 읽거나 DML을 수행하는 작업은 오류가 발생하지 않는다.

어떤 파티션이 Unusable 상태가 되면 그 파티션을 사용하기 전에 Rebuild하여야 한다. 단, Unusable 상태가 된 인덱스를 다시 만들기 전에 해당 파티션을 Split이나 Rename이 가능하며, Unusable 상태인 Global Index를 Drop하는 것도 가능하다.

파티션을 Index Unusable 상태로 만들 수 있는 작업은 어떠한 것이 있는지 살펴보자.

Direct path Load의 경우

Direct path SQL*Loader 수행 후 인덱스가 테이블의 해당 데이터보다 이전 것이면, Unusable 상태가 된다(Oracle7에서는 인덱스가 Direct Load State가 되었다고 표현한다).

인덱스가 테이블의 데이터보다 이전 상태라는 것은 데이터를 Load한 후 인덱스를 생성하는 중에 Space 부족 등의 원인으로 오류가 발생하였거나, SKIP_INDEX_MAINTENANCE Option을 사용한 경우이다.

ROWID가 변경되는 경우

ALTER TABLE MOVE PARTITION과 같이 ROWID를 변화시키는 작업은 영향받는 Local Index와 전체 Global Index를 Unusable 상태가 되게 한다.

ROWID를 지우는 작업

ALTER TABLE TRUNCATE PARTITION이나 DROP PARTITION과 같이 테이블의 Row를 지우는 경우 영향받는 Local Index Partition과 모든 Global Index Partition을 Unusable 상태로 만든다.

테이블 Partition 정의를 변경하는 경우

ALTER TABLE SPLIT PARTITION은 Local Index의 Partition Definition은 변경시키지만, 자동으로 인덱스를 새로운 Definition에 맞게 Rebuild하지 않기 때문에 영향받는 Local Index Partition을 Unusable 상태로 만든다. 또한 이것은 ROWID를 변경시키기 때문에 모든 Global Index Partition을 Unusable 상태로 만든다.

인덱스 Partition 정의를 변경하는 경우

ALTER INDEX SPLIT PARTITION은 Index의 Definition은 변경시키지만, 영향받은 Partition은 Rebuild시키지 않는다. 이 작업은 영향받는 인덱스 파티션 부분을 Unusable 상태로 만든다. 그러나 Global Index의 경우는 그대로 Usable 상태로 된다.



한국오라클(주)

서울특별시 강남구 삼성동 144-17
삼화빌딩
대표전화 : 2194-8000
FAX : 2194-8001

한국오라클교육센터

서울특별시 영등포구 여의도동 28-1
전경련회관 5층, 7층
대표전화 : 3779-4242~4
FAX : 3779-4100~1

대전사무소

대전광역시 서구 둔산동 929번지
대전둔산사학연금회관 18층
대표전화 : (042)483-4131~2
FAX : (042)483-4133

대구사무소

대구광역시 동구 신천동 111번지
영남타워빌딩 9층
대표전화 : (053)741-4513~4
FAX : (053)741-4515

부산사무소

부산광역시 동구 초량동 1211~7
정암빌딩 8층
대표전화 : (051)465-9996
FAX : (051)465-9958

울산사무소

울산광역시 남구 달동 1319-15번지
정우빌딩 3층
대표전화 : (052)267-4262
FAX : (052)267-4267

광주사무소

광주광역시 서구 양동 60-37
금호생명빌딩 8층
대표전화 : (062)350-0131
FAX : (062)350-0130

고객에게 완전하고 효과적인
정보관리 솔루션을 제공하기 위하여
오라클사는 전 세계 145개국에서
제품, 기술지원, 교육 및
컨설팅 서비스를
제공하고 있습니다.

<http://www.oracle.com/>
<http://www.oracle.com/kr>