Oracle Technical Note

Technical Notes
-Troubleshooting 시리즈(4)
백업 / 리커버리(상)

Troubleshooting 시리즈는 필자가 한국오라클 서버지원팀에서 근무하면서 실제 고객들의 문의 사항이 많았던 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 데이타베이스, 파티션 테이블 등에 대해 다루어 보았고, 이번 호와 다음 호에서는 DBA를 책임지는 사람에게 가장 중요하다고 할 수 있는 백업/리커버리(Backup/Recovery)에 대해 다루어 보고자 한다.
이번 호에서는 백업의 종류와 데이타가 손상되는 이유, 데이타베이스가 데이타를 처리하는 과정에서 발생하는 오류에 대해 처리하는 복구의 종류에 대해 살펴본다. 특히, 데이타 손상이 있었을 때 발생하는 에러의 종류와 이의 처리 방안에 대해 살펴보고, 백업을 이용한 리커버리에 대해 알아본다.
그리고 리커버리의 종류 중 백업 파일을 이용한 미디어 리커버리는 지면 관계상 다음 호에서 다루고자 한다.

| 글 / 박경희 | 한국오라클 고객지원실 Server지원팀
| kyeonghee.park@oracle.com |

ORACLE®

오라클 백업

오라클 백업에는 크게 피지컬 백업(Physical Backup) 방식과 로지컬 백업(Logical Backup) 방식이 있다.

피지컬 백업 방식이란 데이타 파일 자체를 카피하는 방식을 말하는데, 이것은 다시 데이타베이스를 셧다운하고 백업하는 방식과 오픈된 상태에서 백업하는 방식으로 나뉜다.

반면, 로지컬 백업 방식이란 오라클에서 제공하는 유틸리티 중의 하나인 Export/Import를 이용하여 백업하는 형식으로, 이 백업 파일을 이용하여 리커버리를 실행하면 Create Object 문장을 이용해 테이블스페이스와 유저, 테이블을 생성하고, Insert Statement를 이용해 특정 테이블에 Insert 작업이 이루어지기 때문에 이처럼 일컫는 것이다.

먼저 이들에 대해 각각 알아보고, 데이타 복구(Recovery)를 위한 기본 개념들을 다루고자 한다. 그리고 리커버리에 대한 더 구체적인 설명은 다음 호에서 다룰 예정이다.

Cold Backup 받는 방법

콜드 백업(Cold Backup)은 데이타베이스가 정상적으로 셧다운된 상태에서 데이타 파일, 로그 파일, 콘트롤 파일을 모두 백업 받는 것을 말한다. 셧다운하지 않고 오픈된 상태에서 백업을 받으면 백업 받은 내용을 나중에 사용할 수가 없으므로 유의해야 한다.

콜드 백업을 위해서 데이타베이스를 셧다운할 때에는 Normal 또는 Immediate 옵션을 사용해야 하며, Abort를 사용해서는 안 된다.

상황이 여의치 않아 Abort를 사용한 경우에는 셧다운 후에 다시 스타트업하고 Normal로 셧다운한 후 백업을 받도록 한다

우선 이들 파일을 백업하기 위해 콘트롤 파일과 데이타 파일 및 로그 파일의 위치를 확인하여 이들을 tar, cpio, dd 등의 명령을 이용하여 백업 받도록 한다.

NT에서는 Copy 명령이나 탐색기를 이용해서 백업을 받으면 된다.

- 콘트롤 파일 확인 방법

  sqlplus에 system/manager로 접속 후
  SQL>select value from v$controlfile;

- 데이타 파일 확인 방법

  SQL>select name from v$datafile;

- 로그 파일 확인 방법

  SQL>select member from v$logfile;

이처럼 각각의 파일 위치를 확인하여 위에서 확인된 3종류의 파일들을 백업 받으면 된다. 이때 오라클은 반드시 셧다운된 상태이어야 한다

항상은 아니지만 필요에 따라서는 패러미터 파일까지 받아두는 것이 좋다. 패러미터 파일은 $ORACLE_HOME/dbs/init〈SID〉.ora에 있는데, 이 초기화 패러미터 파일에 기술되어진 패러미터는 시스템의 성능에 주요한 역할을 하므로 이 파일이 없어

진 경우에 예전의 성능을 유지하려면 이 파일을 필수적으로 백업해야 한다.

NT에서는 C:\ORANT\DATABASE\init〈SID〉.ora 파일이다.

일반적으로 NT의 SID가 처음 만들어질 때 ORCL이므로 이 초기 패러미터 파일 역시 initORCL.ora라고 되어 있다.

여기서 〈SID〉는 오라클 인스턴스 이름으로 환경변수 ORACLE_SID로 지정되어 있다. 이 ORACLE_SID는 Unix의 경우 $env, NT의 경우는 Registry에 지정되어 있으므로 직접 확인해 볼 수 있다.

Hot Backup 받는 방법

핫 백업(Hot Backup)이란 데이타베이스를 셧다운하지 않고 백업하는 방식으로, 일반적인 경우는 적용되지 않고 자신의 데이타베이스를 Archive Mode로 운영하는 경우에만 가능하다.

콜드 백업을 이용하여 데이타를 복구하는 경우에는 콜드 백업을 받은 시점까지만 복구가 가능한 데 반해, 이 Archive 방식으로 데이타베이스를 운영하는 경우에는 사용자가 원하는 시점까지 데이타를 복구할 수 있다. 즉, 원하는 시간을 주어서, SCN 번호를 주어서, 혹은 어느 데이타 파일 한 개만의 복구도 가능하다.

데이타 복구만 생각한다면 모든 데이타베이스를 Archive 방식으로 운영하는 것이 좋겠지만, 이 방식을 사용하는 경우 데이타베이스 관리자가 백업/리커버리에 관하여 많은 지식과 경험을 갖춰야 하며, 또 Archive Log를 계속 유지관리하는 데는 스페이스와 성능의 부하가 따르기 때문에 사용자의 적절한 판단에 의하여 이용하도록 한다.

그러면 Archive Log Mode의 운영 방법에 대해 알아보자.

Archive Log Mode 운영 방법

오라클에서 데이타베이스가 셧다운되지 않은 상태에서 백업을 받거나 문제가 생긴 시점까지의 완벽한 리커버리 작업을 수행하기 위해서는 데이타베이스를 아카이브 로그 모드로 운영하여야 한다.

아카이브 로그 모드로 운영하기 위해서는 다음의 절차에 따라 변경하여야 한다. 1. initSID.ora 파일과 configSID.ora에 다음의 패러미터가 이미 설정되어 있는지 확인한 후에 없을 경우 initSID.ora에 설정한다.

(1) LOG_ARCHIVE_START = TRUE

- 이 패러미터에 의해 데이타베이스가 처음 구동시 ARCH 프로세스가 기동.
- Log Switch 발생시 Automatic Archive를 수행한다.
- 만약 이 패러미터가 False이면 Manual Archive를 실시하여야 한다.

(2) LOG_ARCHIVE_DEST =/home/oracle8/dbs/archive_file/arc

- 아카이브 파일이 생성되는 위치인 디렉토리와 확장자를 포함하지 않는 파일명을 지정.
- 여기에서 archive_file까지는 디렉토리이며 마지막에 있는 arc는 아카이브 로그 파일 이름의 첫 부분이다.

이 아카이브 로그 모드를 설정하면 로그 파일의 스위치가 일어날 때마다 지속적으로 이 Log_Archive_Dest에 지정한 위치에 로그 파일이 쌓이게 되는데, 만일 부주의에 의해 이 아카이브 로그 파일이 깨지거나 분실되는 경우는 그 이후의 데이타 복구가

불가능하므로 주의하여야 한다.

이러한 문제를 줄이기 위해 Oracle8부터는 아카이브 로그 파일을 Mirroring하기 위하여 여러 군데에 보관할 수 있는 방안도 도입되었다. 이는 log_archive_duplex_dest를 설정함으로써 가능하다.

또한 V8i부터는 최대 5군데에 이 로그 파일을 저장할 수 있도록 하여

```
LOG_ARCHIVE_DEST_1="LOCATION=/oracle8/arch_1 MANDATORY"
LOG_ARCHIVE_DEST_2="LOCATION=/oracle8/arch_2 OPTIONAL?
```

처럼 지정이 가능하다

(3) LOG_ARCHIVE_FORMAT = %s.log

- 아카이브 파일의 확장자와 로그 시퀀스 번호의 형식을 지정.
- 이는 (2)에서 정의된 아카이브 로그 파일의 첫 부분 이름과 함께 나타난다.

arc123.log, arc124.log (123과 124는 로그 시퀀스 번호)

와 같은 형태의 파일이 지정한 위치에 생성된다.

2. 다음과 같이 작업하여 아카이브 로그 모드로 변환한다.

```
$ svrmgrl

  SVRMGR> connect internal
  SVRMGR> startup mount              - ■
  SVRMGR> alter database archivelog;   - □
  SVRMGR> archive log list           - ●
  Database log mode              ARCHIVELOG - ◎
  Automatic archival             ENABLED - ◇
  Archive destination            ?/dbs_ar/offline_log/offline - ◆
  Oldest online log sequence     123 - △
  Next log sequence to archive   125 - ▲
  Current log sequence           125 - ▷
  SVRMGR> alter database open;     - ▶
```

■ DB를 Startup Mount까지만 한다.

? 이 커맨드를 이용하여 아카이브 모드로 데이타베이스를 변경한다.

● 아카이브 모드로 변경되었는지를 확인한다.

◎ 데이타베이스가 아카이브 모드임을 나타낸다. 만약 NOARCHIVELOG로 되어 있으면 변경되지 않은 것을 의미한다.

◇ initSID.ora 파일에서 LOG_ARCHIVE_START 패러미터를 TRUE로 정의하였음을 나타내며 False인 경우에는 DISABLED로 나타난다.

◆ initSID.ora 파일의 LOG_ARCHIVE_DEST 패러미터에서 정의한 아카이브할 장소이다.

△ 3개의 Redo Log 중 가장 오래된 리두 로그의 시퀀스가 123임을 의미한다.

▲ 다음에 아카이브 받을 파일의 로그 시퀀스 번호를 나타낸다.

▷ 현재 사용중인 리두 로그의 시퀀스가 125임을 의미한다. 만약 이전부터 아카이브 로그 모드로 운영중이었다면 여기에서 아카이브 로그 파일은 로그 시퀀스 124까지 아카이브되어 있다는 것을 의미한다.

▶ 아카이브 모드로 변경 후 DB를 오픈한다.

No Archive Log Mode로 전환하는 방법

반대로, Archivelog Mode에서 No Archivelog Mode로 전환하는 방법은 다음과 같다.
먼저, 위에서 세팅했던 initSID.ora 파일과 configSID.ora에 있는 다음 패러미터 앞에 #을 넣고 저장한다.

```
#LOG_ARCHIVE_START = TRUE
#LOG_ARCHIVE_DEST = /home/oracle8/dbs/archive_file/arc
#LOG_ARCHIVE_FORMAT = %s.log

$ svrmgrl
SVRMGR> connect internal;
SVRMGR> shutdown immediate
SVRMGR> startup mount
ORACLE instance started.
Database mounted.
SVRMGR> alter database noarchivelog;
Statement processed.
SVRMGR> alter database open;
Statement processed.
```

Hot Backup 하는 방법

데이타베이스를 셧다운하지 않고 데이타 파일을 백업하는 방법이다.
이 방법은 콜드 백업보다 더 복잡하지만 데이타베이스가 오픈되어 있는 도중에 할 수 있고 또한 테이블스페이스 별로 백업할 수 있다는 장점이 있다.
핫 백업은 항상 데이타베이스를 아카이브 로그 모드 상태로 두고 실시한다.

| 방법 | 테이블스페이스 단위로 백업을 실시한다.

```
svrmgrl > CONNECT INTERNAL
svrmgrl > ALTER TABLESPACE SYSTEM BEGIN BACKUP;
시스템 테이블스페이스의 모든 데이타파일에 대해서 OS Backup 한다.
$ tar cvf /dev/rmt0 /usr/oracle8/dbs/syst1ORA8.dbf
$ tar cvf /dev/rmt0 /usr/oracle8/dbs/syst2ORA8.dbf
svrmgrl > ALTER TABLESPACE SYSTEM END BACKUP;
svrmgrl > ALTER TABLESPACE USERS BEGIN BACKUP;
$ tar cvf /dev/rmt0 /usr/oracle8/dbs/user1ORA8.dbf
svrmgrl > ALTER TABLESPACE USERS END BACKUP;
다른 테이블스페이스에 대해서도 같은 방법으로 한다.
```

주의할 점은 BEGIN과 END 사이에는 해당 테이블스페이스의 데이타파일 헤더 정보가 변경되지 않는다는 것이다. 따라서 OS 백업이 종료됨과 동시에 'ALTER TABLESPACE ... END BAKCUP' 커맨드를 실행하여 데이타파일의 헤더가 변경되도록 한다.

이 아카이브 백업 파일을 이용한 복구 방법은 다음 기회에 다루도록 한다.

Export Backup 하는 방법

오라클에서 제공되는 Export 유틸리티는 데이타베이스에 저장된 데이타를 바이너리 형태의 OS 파일로 만들고, 필요시 Import 유틸리티를 이용하여 데이타베이스로 다시 올리는 방식이다 .

이 유틸리티는 각 오브젝트 단위로 처리가 가능하기 때문에 테이블 몇 개만을 복구한다든가 특정 사용자의 테이블들을 다른 테이블스페이스로 옮긴다든가 또는 전체 데이타베이스의 자료를 서로 다른 OS로 옮기는 경우 등에 특히 유리하다.

Export의 종류

이 Export 방안은 시스템과 서로 메시지를 주고받으면서 백업을 할 수 있는 Interactive Mode가 있고, 한 라인의 명령어로 Export시 필요한 옵션을 길게 열거해주는 Command 방식이 있다.

Command 방식은 순간 순간 필요한 옵션을 적어주기만 하면 되기 때문에 여기서는 Interactive 방식을 다루어 보기로 한다

Export의 단위

- Full 단위 : 전체 데이타베이스를 Export 한다.
- User 단위 : 특정 유저 전체 오브젝트를 Export 한다.
- Table 단위 : 특정 테이블을 Export 한다.
- Partition 단위 : 특정 테이블의 파티션을 Export 한다.

Export의 실제 예제

| 예 1 | 전체 데이타베이스의 Export (Interactive Method)

```
$ exp system/manager
Connected to: ORACLE8 Server Release 8.0.5 - Production

With the procedural and distributed options
PL/SQL Release 2.2 - Production
Enter array fetch buffer size : 4096 > 100000 (RETURN)
Export file : expdat.dmp >
(1) E(ntire database), (2) U(sers), (3) T(ables) :  U > e
Export grants (Y/N) : Y > y
Export table data (Y/N) : Y > y
Compress extents (Y/N) : Y > y
About to export the entire database....
. exporting tablespace definitions
. exporting profiles
. exporting user definitions
. exporting role
```

```
. exporting rollback segment definitions
. exporting database links
. exporting sequence numbers
. exporting sequence numbers
. exporting cluster definitions
. exporting stored procedures
. about to export SYSTEM's tables ...
. about to export SCOTT's tables ...
. exporting synonyms
. exporting views
. exporting referential integrity constraints
. exporting triggers

Export terminated successfully without warnings.
```

| 예 2 | 전체 데이타베이스의 EXPORT(Command Line Method)

```
$ exp userid=system/manager full=y file=fullbackup.dmp  buffer=100000
```

| 예 3 | 전체 데이타베이스의 EXPORT(Dynamic Method)

Export 패러미터를 다음과 같은 파일(tusc.par) 형태로 만든다.

```
 system/manager
 full=y
 file=fullbackup.dmp
 buffer=100000
$ exp parfile=tusc.par
```

| 예 4 | User 단위의 Export

```
$ exp system/manager
Connected to: ORACLE8 Server Release 8.0.5 - Production

With the procedural and distributed options
PL/SQL Release 2.2 - Production
Enter array fetch buffer size : 4096 > 100000 (RETURN)
Export file : expdat.dmp >
(1) E(ntire database), (2) U(sers), (3) T(ables) :  U > u
Export grants (Y/N) : Y > y
Export table data (Y/N) : Y > y
Compress extents (Y/N) : Y > y

About to export specified users
User to be exported: (RETURN to quit) > scott
. exporting snapshots
. exporting snapshot log
. exporting database links
. exporting sequence numbers
```

```
. exporting sequence numbers
. exporting cluster definitions
. exporting stored procedures
. about to export SCOTT's tables ...
. exporting synonyms
. exporting views
. exporting referential integrity constraints
. exporting triggers
Export terminated successfully without warnings.
```

| 예 5 | User 단위의 Export (Command Line Method)

```
$ exp system/manager owner=scott file=scott.dmp buffer=100000
```

또는

```
$ exp scott/tiger file=scott.dmp buffer=100000
```

| 예 6 | User 단위의 Export (Dynamic Method)

Export 패러미터를 다음과 같은 파일(tusc.par) 형태로 만든다.

```
 $vi  tusc.par

   scott/tiger
   file=scott.dmp
   buffer=1000000
$ exp parfile=tusc.par
```

| 예 7 | Table 단위의 Export

```
$ exp scott/tiger file=table.dmp tables=emp,dept buffer=100000
```

Technical Notes - Troubleshooting 시리즈(4)

데이타 리커버리

Import를 이용한 데이타 복구

Import 유틸리티를 이용한 데이타 Import는 Export와 비슷한데 항상 Export를 통해 만들어진 dump 파일을 이용하여 Import 한다.

전체 데이타베이스 Export 파일로부터 한 개의 테이블을 Import하는 방법 (Interactive Mode)

```
$ imp system/manager
Connected to: ORACLE8 Server Release 8.0.5 - Production
With the procedural and distributed options
PL/SQL Release 2.2 - Production

Import file: expdat.dmp > (RETURN)
Enter insert buffer size (minimum is 4096) 100000 > (RETURN)
Export file created by EXPORT:V08.05
List contents of import file only (yes/no) : no >
 ( “yes” 이면 Data는 Import 되지 않고 Display만 됨 )
Ignore create errors due to object existence (yes/no) : yes >
 ("yes" : 만일 Table이 존재하면 Record가 추가된다.
  "no"  : 이미 존재한 Object에 대해서 Import를 하지 않는다.)
Import grants (yes/no) : yes >
Import table data (yes/no) : yes >
Import entire export file (yes/no) : yes >  no
User name : SCOTT
Enter table name. Null list name all tables for user
Enter table name or . if done : TEST
 importing SCOTT's objects into SCOTT
 importing table TEST      900 rows imported
Import terminated successfully.
```

주의 : Import 작업은 보통 생성되는 인덱스 수에 따라 1.5~4배의 시간이 걸리는데 Export는 “CREATE INDEX ....” 문만을 파일에 쓰고 Import 시에는 실제로 인덱스를 생성하기 때문이다. 만일 Import 하는 속도를 빠르게 하기 위해 Import 시 인덱스를 생성하지 않고 모든 데이타를 Import 한 후 추가 생성하는 것도 가능하다

```
* Command Line Mode
$ imp system/manager full=n owner=scott tables=test commit=y
buffer=100000
```

주의 : Import 시는 데이타를 데이타베이스에 Write 하기 때문에 Rollback Segment가 사용된다. Import를 수행하는 과정에서 Rollback Segme-nt Error가 나는 것을 피하고자 하는 경우에는 commit=y 옵션을 추가하면 주어진 버퍼 안의 Rows만큼의 데이타를 해당 테이블에 올리고 나서 Commit을 하기 때문에 Rollback Segment가 커지는 것을 피할 수 있다. commit=n이면 테이블 단위로 Import하고 나서 Commit이 되기 때문에 테이블의 사이즈가 큰 경우 커다란 Rollback Segment를 필요로 하고 이에 따라 Rollback Segment Error가 발생할 수 있다.

데이타 손상의 개념

우리가 데이타를 이용해 작업을 하는 과정에서 정상적인 작업이 이루어지지 않는 경우, 또는 ORA-1578이나 ORA-600 에러를 접하는 경우 데이타가 손상(Corrupt)되었다고 한다.

이 데이타 손상 종류는 Physical Corruption과 Logical Corruption으로 나뉘며, Physical의 경우는 하드웨어나 소프트웨어의 결함으로 메모리, I/O 버퍼, 오라클 버퍼 캐시 또는 디스크에 문제가 생긴 경우이다.

이로 인한 에러는 다음과 같다.

- -01578 : "ORACLE data block corrupted (file # %s, block # %s)"
  Cause: The data block indicated was corrupted, mostly due to software errors.
- 첫 번째 Argument가 2000-8000인 ORA-600
- 트레이스 파일에서 내용 중 "Corrupt Block"이 나타나는 경우
- 테이블 분석 후
  ORA-01498: "block check failure - see trace file"
  ORA-01499: "table/index cross reference failure - see trace file"
  의 에러가 나타나는 경우이다.

데이타 리커버리 종류

데이타 리커버리 부문은 자세히 하나씩 모두 열거하고자 하면 너무 길어지므로 이번 호에서는 그 종류에 대해 알아보기로 한다.

블록 단위의 리커버리

우리가 데이타를 입력, 삭제, 수정하는 경우, 오라클 내부에서는 리두 레코드(Redo Record)를 생성한다. 이의 설명은 자칫 어려워질 수 있으므로 간단히 설명하도록 한다.

리두 레코드란 테이블의 데이타 값이 수정되면 해당 데이타가 위치한 블록의 변경에 대한 정보, 이 데이타에 연결된 인덱스 정보, 이 수정에 따른 Rollback 블록에 관한 변경 정보의 총칭이라 이해하면 된다. 또 이들 각각을 'Change Vector'라 칭하므로 리두 레코드란 Change Vector의 모음이라고 생각할 수 있다.

오라클에서는 값이 변경되기 전에 항상 내부적으로 리두 레코드 를 생성하고, 이 리두 레코드는 로그 버퍼에 변경되어지며(Switch), 이 로그 버퍼가 변경되어지거나 시스템의 특정 값에 도달하는 경우 Checkpoint가 발생하는데, 이 체크포인트 발생시 해당 변경 정보들이 실제 Physical Block인 디스크 블록에 변경되어지는 것이다.

그런데 이 과정중에 변경된 데이타가 리두 로그 버퍼에만 반영되고 블록 버퍼에 반영되지 못하고 작업 수행중이던 프로세스가 실패하는 경우 블록 단위의 리커버리가 필요하다

이 블록 단위의 리커버리는 시스템이 현재 사용중인 리두 로그 파일을 이용해 자동으로 수행한다.

단, 로그 파일의 사이즈에 비해 변경 작업이 빠르게 많이 실행되는 경우 'Checkpoint Not Completed' 상태에서 로그 파일이 Switch 된 경우는 체크포인트 이후 시점의 모든 리두 로그 파일이 사용된다.

트랜잭션 리커버리

특정 세션에서 Commit을 만나기 전까지의 SQL 문들을 '트랜잭션'이라 규정하며, 이 트랜잭션 리커버리는 그 일관성을 위해 필요하다.

예를 들어, Insert나 Update 수행 도중 시스템이 중단되는 경우가 발생했을 때 리두 로그의 정보를 이용해 Roll Forward를 실시하고, Rollback Segment를 이용해 Roll Backward를 실시하는 과정을 의미한다.

즉, 각 블록이 수정되기 전에 리두 레코드가 만들어지는데 이 안에는 Rollback의 변경도 모두 만들어진다고 앞에서 언급하였다. 그러므로 이 리두 레코드 정보를 이용해 Roll Forward 하고, 각 데이타 헤더 블록에 들어있는 롤백 정보를 이용해 Roll Backward 한다.

참고로, 데이타 블록 헤더에는 그 블록을 수정한 트랜잭션 정보와 사용한 롤백 세그먼트 정보가 들어 있으므로 이를 이용해 트랜잭션 리커버리가 가능하며 이 리커버리 자체 작업도 리두 레코드를 발생시킨다

즉, 롤백 세그먼트의 정보는 UNDO$를 확인하여 구할 수 있으며, UNDO$ 내의 각 롤백 세그먼트에 포함된 Active Transaction들(아직 처리가 완료되지 않은 트랜잭션)의 정보를 이용해 Rollback 시킨다.

Crash 리커버리

이는 모든 인스턴스가 실패하는 경우로 데이타베이스가 시작될 때 자동으로 수행되며, Online Redo Logs, Current Online Datafiles, Current Control Files을 이용하여 복구되는 방안으로, 각 인스턴스에서 변경한 내용이 캐쉬에는 있으나 아직 디스크 블록에 적지 않은 내용을 반영하는 과정이다.

현재의 데이타 파일에 반영하지 않은 블록 버퍼의 내용을 이미 생성한 리두 레코드 정보를 이용하여 반영하는 것이다.

이때의 리커버리는 온라인 리두 로그 파일의 리두 레코드를 반영하는 과정이며 단 Commit 되지 않은 트랜잭션을 롤백하는 것은 포함하지 않는다.

인스턴스 리커버리

여러 개의 인스턴스를 갖는 시스템(Oracle Parallel Server)에서 한 개 이상의 인스턴스가 실패했지만, 전체 인스턴스 모두가 실패하지 않은 경우이다.

실패한 인스턴스 블록을 DLM이 액세스하고자 하거나, 실패한 인스턴스가 변경한 블록을 다른 인스턴스가 읽고자 하여 그 이전 Lock을 정리를 요청하는 과정에서 이 인스턴스의 실패를 알게 되어 다른 인스턴스나 DLM이 이 실패를 자신의 SMON에게 알려 Thread Open Lock을 획득한 후 Instance Recovery를 하게 된다.

미디어 리커버리

미디어 리커버리는 다른 것과는 달리 자동으로 수행되지 않으며, Recover 명령문에 의해서만 수행된다.

백업 파일을 Restore 한 경우나 특정 파일이 체크포인트 없이 오프라인된 경우, 컨트롤 파일을 다시 만드는 등의 과정으로 Using Backup Controlfile 을 사용한 경우이다. 이 경우는 특정 데이타 파일 헤더와 컨트롤 파일 헤더의 체크포인트 값이 달라 미디어 리커버리가 필요하게 된다.

또한 미디어 리커버리는 해당 파일 헤더에 기록된 가장 작은 SCN부터 복구를 시작한다.

Troubleshooting

ORA-1578 에러 처리

이 에러는 특정 블록의 데이타가 손상되었다는 에러로, 이를 처리하기 위해서는 문제가 발생한 블록을 제외하고 테이블을 카피하는 방법이 있고, Event 10231을 적용하여 해당 블록을 스킵하여 테이블을 백업하는 방법이 있다.

그러면, 각각에 대해 살펴보자.

Copy를 적용한 방법

ORA-1578 에러가 발생하면 데이타 손상이 발생한 파일번호와 블록번호를 알려준다. 여기서는 이 때의 파일번호를 f, 블록번호를 b라고 부르기로 한다.

① 우선 해야 할 일은 어떠한 오브젝트가 손상되었는가를 알아내는 것이다. 다음의 스크립트를 이용하면 알 수 있다.

```
SQL>select segment_name, segment_type
    from dba_extents
    where file_id = f and
    b between block_id and block_id + blocks - 1;
```

② 만약 해당 세그먼트가 인덱스이면 Drop 시키고 다시 생성하면 된다.

③ 만약 해당 세그먼트가 테이블이면 손상된 블록의 데이타는 손상된 것이다.

④ 만약 해당 테이블이 들어 있는 Export 파일이 있다면, 손상된 테이블을 Drop 시키고 Import 받는 것이 제일 간단한 방법이다. 하지만, 만약 Export 받은 파일이 없거나 백업해 놓은 파일도 없다면 해당 테이블에 인덱스가 생성되어 있는 경우에 한해서 다음의 방법을 사용해서 복구를 하도록 한다.

⑤ 해당 테이블에 대한 인덱스가 생성되어 있다면 이를 이용해서 손상된 블록을 피해갈 수 있다. 방법은 다음과 같다.

empno, ename, deptno를 컬럼으로 가지는 emp 테이블이 손상되었다고 가정하자. 그리고, empno 컬럼에 인덱스가 생성되어 있다고 하자. 클러스터화되지 않은 모든 테이블은 Unique 한 rowid를 가진다.

rowid를 varchar2/hexadecimal 형식으로 표현하려면 rowidtochar 함수를 이용한다.

```
SQL>select rowidtochar(rowid) from emp
```

rowid는 총 18자로 Block Address(8자), Dot(1자), Row Address (4자), Dot(1자), File Address(4자)로 구성되어 있다.

```
SQL>select empno, rowid
    from emp
    where empno > 0
```

위의 스크립트를 실행시키면 다음과 같은 결과를 얻게 된다.

```
EMPNO          ROWID
---------------  ---------------------------------
100            00000003.0000.0006
101            00000003.0001.0006
102            00000003.0002.0006
103            00000003.0003.0006
.
.
500            00000004.0000.000A
501            00000004.0001.000A
.
.
755            0000001A.0005.000A
756            0000001A.000C.000A
```

만약 인덱스가 Character 컬럼에 대한 것이었다면 위의 Query 문장을 다음과 같이 바꿀 수 있다.

```
SQL>select empno, rowid
    from emp
    where empno > '';
```

예를 들어, 다음과 같은 에러 메시지가 나왔다고 하자.

```
01578, 00000, "ORACLE data block corrupted (file # 10, block # 4)
```

그러면, 다음의 스크립트를 사용하여 손상된 블록에 있는 Employee에 대한 empno를 구할 수 있다.

```
SQL>select empno from emp
    where empno > 0
    and rowidtochar(rowid) like '00000004.%.000A';
   EMPNO          ROWID
   --------       ------------------
   500            00000004.0000.000A
   501            00000004.0001.000A
```

이제 emp 테이블과 같은 구조를 갖는 새로운 테이블을 만든다.

```
SQL>create table temp
    as select * from emp
    where 1 = 2;
```

그런 다음 손상된 부분을 피해서 새로운 테이블에 손상된 테이블의 데이타를 추가한다.

```
SQL>insert into temp select * from emp where empno < 500;
SQL>insert into temp select * from emp where empno > 501;
```

손상된 테이블을 Drop 시키고 temp 테이블의 이름을 emp로 변경한다. 그리고, 백업된 자료나 문서자료를 통하여 손상된 부분에 대한 정보를 추가한다.

⑥ 손상된 블럭에 여러 개의 Row가 존재하고 있다면 다음의 방법을 이용한다.

```
SQL>create table empnos as
    select empno from emp
    where empno > 0
    and rowidtochar(rowid) not like '00000004.%.000A';
```

이 스크립트를 이용하면 손상된 블록에 포함되지 않는 empno들을 알 수 있다. 다음의 스크립트를 계속 실행시켜 복구한다.

```
SQL>create table temp as select * from emp  where 1 = 2;
SQL>insert into temp
    select emp.empno, emp.ename, emp.deptno
    from emp, empnos
    where emp.empno > 0
    and emp.empno = empnos.empno;
```

⑦ 만약 데이타 디렉토리의 테이블이나 인덱스에서 손상된 블록이 발생했다면 데이타 백업을 이용하여 복구하여야 한다.

Event를 이용하는 방법

Event 10231은 테이블 전제를 읽을 때 문제가 발생한 블록을 건너뛰고 읽어낼 수 있는 방안이다.

이를 이용하면 Export 유틸리티를 이용하여 테이블을 백업 받거나, "Create table as select .." 문장을 이용하여 테이블을 카피할 수 있다. 물론 손상된 블록은 복구할 수 없다.

① 이 Event를 사용하기 위해 데이타베이스를 셧다운한 후 initSID.ora file에 다음의 패러미터를 설정한다.

```
event="10231 trace name context forever, level 10"
```

② 파일을 저장한 후, 패러미터가 설정되었는지 확인하기 위해 다음을 실행한다.

```
SVRMGR>startup restrict;
SVRMGR>show parameter event

NAME    TYPE    VALUE
------  ------  ------------------------------
event   string  10231 trace name context forev
```

③ 다음의 방법으로 테이블을 읽으면서 필요 부분을 읽어낸다.

```
$ exp scott/tiger file=new_table.dmp tables=corrupt_table
```

또는

```
SQL>create table new_table as select * from corrupt_table;
```

④ initSID.ora 파일에서 지정한 Event 패러미터를 제거한 후 다시 데이타베이스를 Startup 한다 .

⑤ 깨진 데이타를 갖는 테이블을 Drop 시키거나 Rename 시키고, 테이블을 재생성한다.

```
SQL>drop table corrupt_table;
```

(혹은 SQL>rename table corrupt_table to temp ;)

```
$imp scott/tiger file=new_table.dmp tables=corrupt_table
```

⑥ '필요시 index, constraint, trigger 등은 재생성하고 권한을 부여한다.

한국오라클(주)
서울특별시 강남구 삼성동 144-17
삼화빌딩
대표전화 : 2194-8000
FAX : 2194-8001

한국오라클교육센타
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