5주차 교육 세미나

2023. 08. 11
DQA1-3 이윤재

I. 지난주 피드백

II. Physical Storage Structure

III. Logical Storage Structure

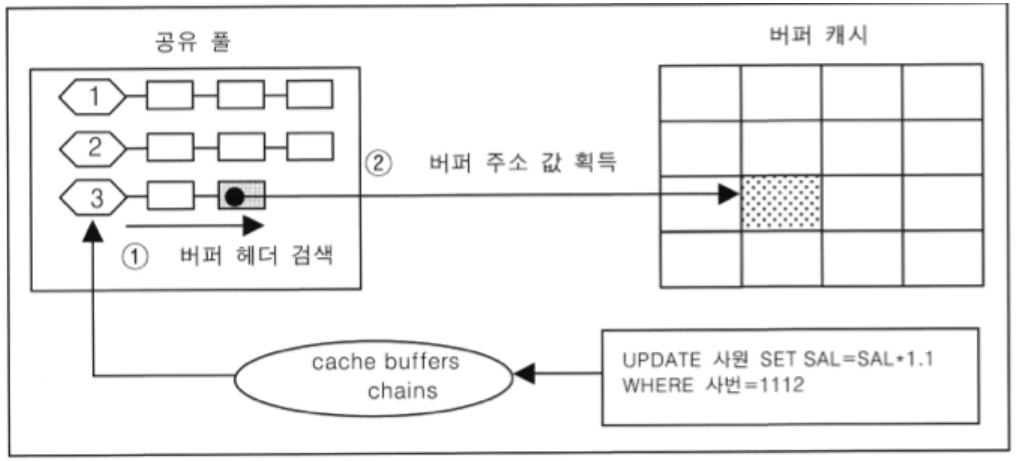
IV. Oracle Database Instance

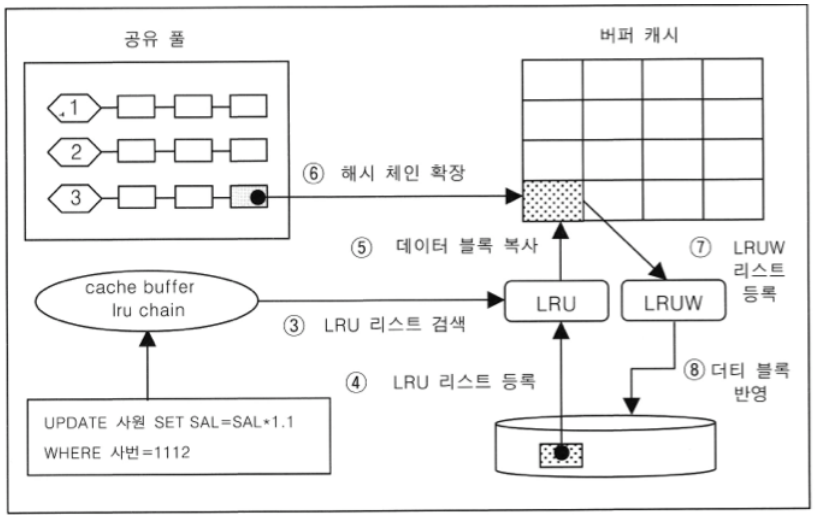
Ⅰ. 지난주 피드백

- 데이터 버퍼 접근 순서
- 재시작 가능한 백그라운드 프로세스
- Checkpoint와 SCN
- SCN record solution
- Hint
- OPTIMIZER_MODE
- Active Session History(ASH)

지난주 피드백

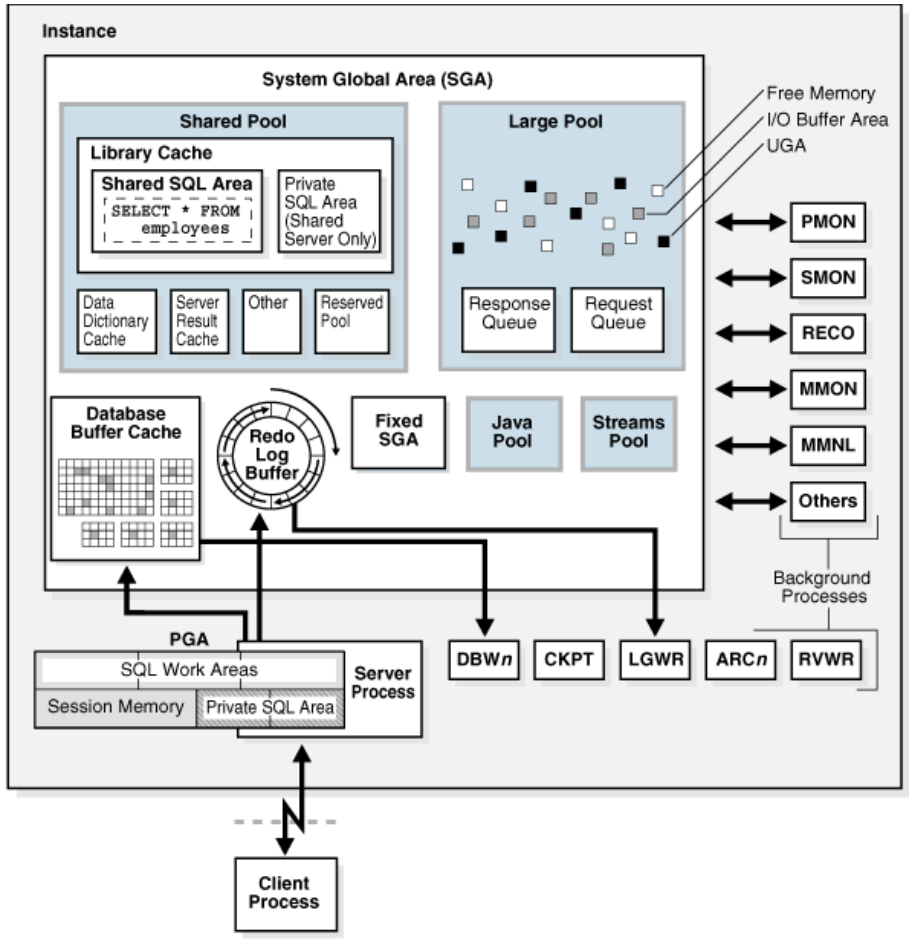
데이터 버퍼 접근 순서





지난주 피드백

BUFFER
HASH
CHAIN



UNDO
SEGMENT

지난주 피드백

재시작 가능한 백그라운드 프로세스

PMON, SMON, DBW, LGWR...

```
[ora19c@moti ora19c]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Thu Aug 10 17:27:47 2023
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to an idle instance.
```

ARCn

```
[ora19c@moti ora19c]$ ps -ef | grep ora_arc
ora19c    750755       1  0 17:28 ?        00:00:00 ora_arc0_DB19
ora19c    750791       1  0 17:28 ?        00:00:00 ora_arc1_DB19
ora19c    750793       1  0 17:28 ?        00:00:00 ora_arc2_DB19
ora19c    750795       1  0 17:28 ?        00:00:00 ora_arc3_DB19
ora19c    751111  747231  0 17:29 pts/4    00:00:00 grep --color=auto ora_arc
[ora19c@moti ora19c]$ kill 750795
```

```
[ora19c@moti ora19c]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Thu Aug 10 17:29:15 2023
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
```

지난주 피드백

Checkpoint와 SCN

select checkpoint_change#, current_scn from v$database;

```
CHECKPOINT_CHANGE# CURRENT_SCN
------------------ -----------
           3646663     3646799

SQL> /

CHECKPOINT_CHANGE# CURRENT_SCN
------------------ -----------
           3646663     3646800

SQL> /

CHECKPOINT_CHANGE# CURRENT_SCN
------------------ -----------
           3646663     3646801
```

```
SQL> alter system checkpoint;

System altered.

SQL> select checkpoint_change#, current_scn from v$database;

CHECKPOINT_CHANGE# CURRENT_SCN
------------------ -----------
           3647549     3647551
```

지난주 피드백

SCN record solution

1. 컨트롤 파일 헤더
   - Checkpoint 발생 때
   - Resetlogs 발생 때

2. Data blocks(cache layer)
   - Block cleanout시 마지막 scn을 각 block에 기록

3. Data blocks(ITL entries)
   - Data block의 transaction layer 안에 있는 ITL entries에 commit된 SCN 정보를 기록

4. Data file headers : 모든 데이터 파일 헤더에 scn 기록
   - Begin backup 수행
   - 복구 되었을 때 마지막 scn 기록

5. Redo record / log buffer
   - Commit이 수행되면 commit record에 scn을 포함하여 저장

지난주 피드백

Hint

123 DEPTNO	ABC DNAME	ABC LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Select * from dept

ABC JOB	123 DEPTNO
PRESIDENT	10
MANAGER	30
MANAGER	10
MANAGER	20
ANALYST	20
CLERK	20
SALESMAN	30
SALESMAN	30
SALESMAN	30
SALESMAN	30
CLERK	30
CLERK	10

Select job, deptno from emp

```
SELECT *
FROM dept
WHERE deptno IN (SELECT deptno
                 FROM emp
                 WHERE job = 'CLERK');
```

지난주 피드백

Hint

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

JOB	DEPTNO
PRESIDENT	10
MANAGER	30
MANAGER	10
MANAGER	20
ANALYST	20
CLERK	20
SALESMAN	30
SALESMAN	30
SALESMAN	30
SALESMAN	30
CLERK	30
CLERK	10

```
|   0 | SELECT STATEMENT            |         |     3 |   147 |     6  (17)| 00
:00:01 |
|   1 |  NESTED LOOPS               |         |     3 |   147 |     6  (17)| 00
:00:01 |
|   2 |   NESTED LOOPS              |         |     3 |   147 |     6  (17)| 00
:00:01 |
|   3 |    SORT UNIQUE              |         |     3 |    57 |     3   (0)| 00
:00:01 |
|*  4 |     TABLE ACCESS FULL       | EMP     |     3 |    57 |     3   (0)| 00
:00:01 |
|*  5 |    INDEX UNIQUE SCAN        | PK_DEPT |     1 |       |     0   (0)| 00
:00:01 |
|   6 |   TABLE ACCESS BY INDEX ROWID| DEPT    |     1 |    30 |     1   (0)| 00
:00:01 |
```

```
SELECT *
FROM dept
WHERE deptno IN (SELECT deptno
                 FROM emp
                 WHERE job =
                 'CLERK');
```

지난주 피드백

Hint

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

JOB	DEPTNO
PRESIDENT	10
MANAGER	30
MANAGER	10
MANAGER	20
ANALYST	20
CLERK	20
SALESMAN	30
SALESMAN	30
SALESMAN	30
SALESMAN	30
CLERK	30
CLERK	10

```
|   0 | SELECT STATEMENT                    |          |     4 |   120 |     5
  (0)| 00:00:01 |

|*  1 |  FILTER                             |          |       |       |
     |          |

|   2 |   TABLE ACCESS FULL                 | DEPT     |     4 |   120 |     3
  (0)| 00:00:01 |

|*  3 |   TABLE ACCESS BY INDEX ROWID BATCHED| EMP      |     1 |    19 |     2
  (0)| 00:00:01 |

|*  4 |    INDEX RANGE SCAN                 | EMP_IX01 |     1 |       |     1
  (0)| 00:00:01 |
```

SELECT *
FROM DEPT
WHERE deptno IN (SELECT **/*+ no_unnest */**
deptno FROM emp WHERE job='CLERK');

지난주 피드백

Hint

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

JOB	DEPTNO
PRESIDENT	10
MANAGER	30
MANAGER	10
MANAGER	20
ANALYST	20
CLERK	20
SALESMAN	30
SALESMAN	30
SALESMAN	30
SALESMAN	30
CLERK	30
CLERK	10

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     4 |   120 |     6   (0)| 00:00:01 |
|*  1 |  FILTER            |      |       |       |            |          |
|   2 |   TABLE ACCESS FULL| DEPT |     4 |   120 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| EMP  |     1 |    19 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

```
SELECT *
FROM DEPT
WHERE deptno IN (SELECT /*+ no_unnest full(emp)
*/ deptno FROM emp WHERE job='CLERK');
```

지난주 피드백

OPTIMIZER_MODE

- FIRST_ROWS_n : 첫 n 개만큼의 row를 반환하는 시간을 줄이는 것을 목표로 cost-based 접근을 사용함
- FIRST_ROWS : cost-based와 rule-based를 섞어 적은 양의 row를 빠르게 전달하는 것이 목표이다. 이전 버전의 호환성을 위해 남겨둔 모드
- ALL_ROWS : 모든 SQL 질의에 대해 cost-based 접근을 사용하고, 가장 적은 자원을 사용하는 것을 목표로 한다(best throughput).

지난주 피드백

Active Session History(ASH)

ASH는 현재 접속해서 활동 중인 Active 세션 정보를 1초에 한번씩 샘플링해서 ASH 버퍼에 저장한다

ASH 버퍼가 가득 차게 되면 MMNL이 AWR로 내려쓰게된다.

```
SQL> select * from v$sgastat where name = 'ASH buffers';

POOL         NAME                            BYTES     CON_ID
------------ -------------------------- ---------- ----------
shared pool  ASH buffers                   25165824          1
```

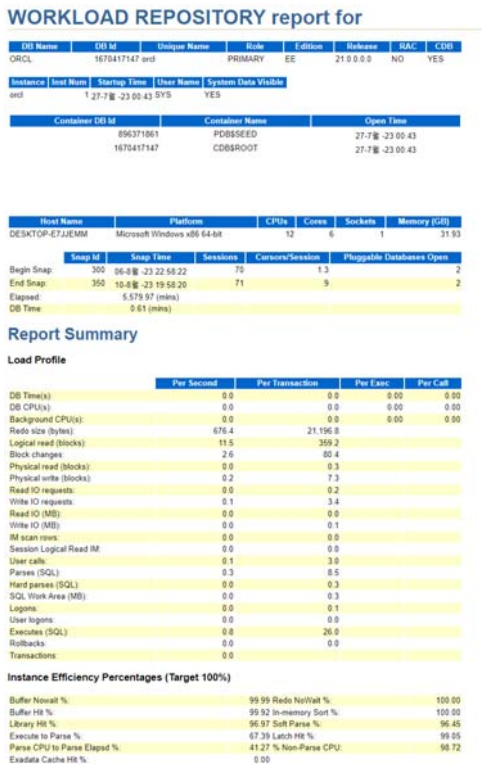
SAMPLE_ID	SAMPLE_TIME	SESSION_ID
1,248,329	2023-08-11 00:31:51.629	611
1,248,200	2023-08-11 00:29:42.628	611
1,248,114	2023-08-11 00:28:16.630	491
1,248,078	2023-08-11 00:27:40.629	611
1,247,948	2023-08-11 00:25:30.633	611
1,247,815	2023-08-11 00:23:17.628	611
1,247,797	2023-08-11 00:22:59.633	1,221
1,247,782	2023-08-11 00:22:44.628	1,221
1,247,693	2023-08-11 00:21:15.627	611
1,247,620	2023-08-11 00:20:02.633	246
1,247,614	2023-08-11 00:19:56.625	246
1,247,441	2023-08-11 00:17:03.628	611

지난주 피드백

Active Session History(ASH)

```
SQL> @?/rdbms/admin/awrrpt.sql

Specify the Report Type
~~~~~~~~~~~~~~~~~~~~~~~
AWR reports can be generated in the following formats.  Please enter the
name of the format at the prompt.  Default value is 'html'.

'html'          HTML format (default)
'text'          Text format
'active-html'   Includes Performance Hub active report

report_type의 값을 입력하십시오: html
```

```
Listing the last 3 days of Completed Snapshots
Instance     DB Name      Snap Id       Snap Started    Snap Level
------------ ------------ ---------- ------------------ ----------

orcl         ORCL              311  08 8월  2023 00:58   1
                               312  08 8월  2023 01:59   1
                               313  08 8월  2023 02:58   1
                               314  08 8월  2023 03:58   1
                               315  08 8월  2023 04:58   1
                               316  08 8월  2023 05:58   1
                               317  08 8월  2023 06:58   1
                               318  08 8월  2023 19:44   1
                               319  08 8월  2023 20:58   1
                               320  08 8월  2023 21:59   1
                               321  08 8월  2023 22:58   1
                               322  08 8월  2023 23:58   1
                               323  09 8월  2023 00:58   1
                               324  09 8월  2023 01:58   1
                               325  09 8월  2023 02:58   1
                               326  09 8월  2023 03:58   1
                               327  09 8월  2023 04:58   1
                               328  09 8월  2023 05:58   1
```



WORKLOAD REPOSITORY report for

DB Name	DB Id	Unique Name	Role	Edition	Release	RAC	CDB
ORCL	1670417147	orcl	PRIMARY	EE	21.0.0.0.0	NO	YES

Instance	Inst Num	Startup Time	User Name	System Data Visible
orcl	1	27-7월 -23 00:43	SYS	YES

Container DB Id	Container Name	Open Time
896371861	PDB$SEED	27-7월 -23 00:43
1670417147	CDB$ROOT	27-7월 -23 00:43

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
DESKTOP-E7JJEMM	Microsoft Windows x86 64-bit	12	6	1	31.93

	Snap Id	Snap Time	Sessions	Cursors/Session	Pluggable Databases Open
Begin Snap:	300	06-8월 -23 22:58:22	70	1.3	2
End Snap:	350	10-8월 -23 19:58:20	71	9	2
Elapsed:		5,579.97 (mins)			
DB Time:		0.61 (mins)			

Report Summary

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	0.0	0.0	0.00	0.00
DB CPU(s):	0.0	0.0	0.00	0.00
Background CPU(s):	0.0	0.0	0.00	0.00
Redo size (bytes):	676.4	21,196.8		
Logical read (blocks):	11.5	359.2		
Block changes:	2.6	80.4		
Physical read (blocks):	0.0	0.3		
Physical write (blocks):	0.2	7.3		
Read IO requests:	0.0	0.2		
Write IO requests:	0.1	3.4		
Read IO (MB):	0.0	0.0		
Write IO (MB):	0.0	0.1		
IM scan rows:	0.0	0.0		
Session Logical Read IM:	0.0	0.0		
User calls:	0.1	3.0		
Parses (SQL):	0.3	8.5		
Hard parses (SQL):	0.0	0.3		
SQL Work Area (MB):	0.0	0.3		
Logons:	0.0	0.1		
User logons:	0.0	0.0		
Executes (SQL):	0.8	26.0		
Rollbacks:	0.0	0.0		
Transactions:	0.0			

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	99.99	Redo NoWait %:	100.00
Buffer Hit %:	99.92	In-memory Sort %:	100.00
Library Hit %:	96.97	Soft Parse %:	96.45
Execute to Parse %:	67.39	Latch Hit %:	99.05
Parse CPU to Parse Elapsd %:	41.27	% Non-Parse CPU:	98.72
Exadata Cache Hit %:	0.00		

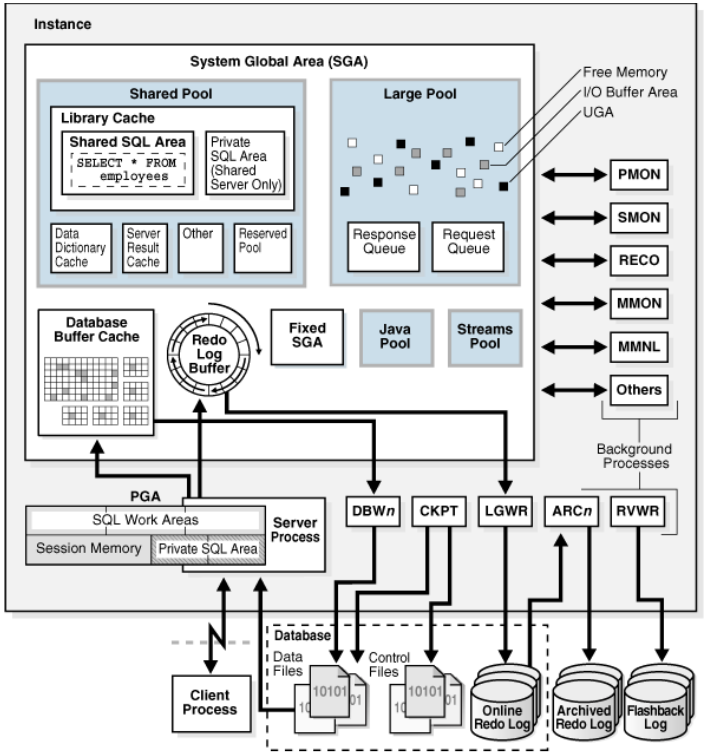
Ⅱ. Physical Storage Structure

- Intruduction to Physical Storage Structure
- Overview of Datafile
- Overview of Control Files
- Overview of Online Redo log

Physical Storage Structure

Physical Storage Structure Overview

- Data file and temp file
- Control file
- Online redo log file



Physical Storage Structure

Mechanism for Storing Database Files

- Oracle ASM
- 운영체제 파일 시스템
- 클러스터 파일 시스템

Physical Storage Structure

Oracle Automatic Storage Management (Oracle ASM)

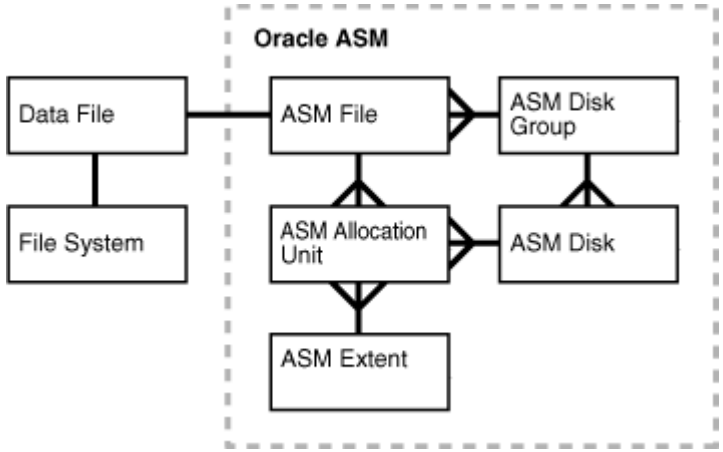
Oracle ASM은 Oracle Database 파일을 위한 관리하기 쉬운 고성능 스토리지 솔루션이다.

- 데이터베이스 생성 및 배치, 디스크 공간 관리와 같은 스토리지 관련 작업 간소화
- 물리적 Disk 전반에 데이터를 분산하여 핫 스팟을 제거하고 Disk 전반에 걸쳐 균일한 성능 제공
- 스토리지 구성 변경 후 자동으로 데이터 균형 재조정

Physical Storage Structure

Oracle ASM Storage Component

- ASM Disk : 물리적 Disk, 파티션, 네트워크 연결 파일
- ASM Disk Group : 논리적인 ASM Disk의 모음.
- ASM File : ASM Disk 그룹에 저장된 파일.
  - 데이터 파일, redo log 파일, control file 등등..
- ASM Allocation Unit : ASM Disk group에서의 최소 연속 Disk 공간 단위
- ASM Extent : ASM Allocation Unit의 집합



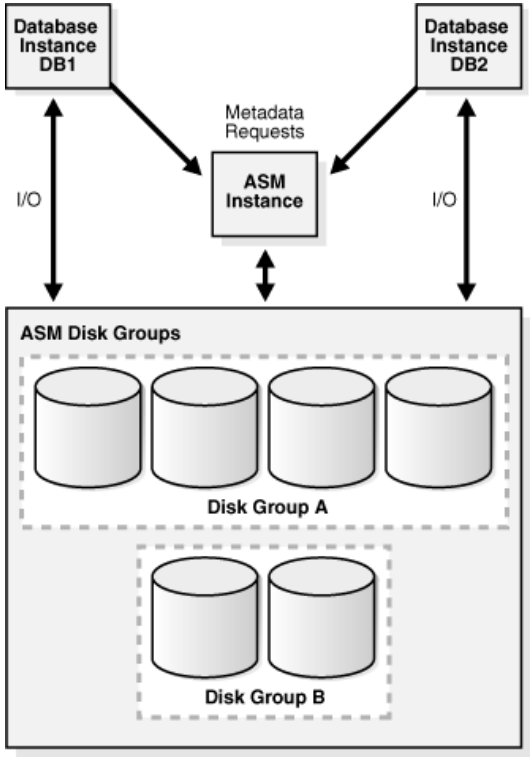
Physical Storage Structure

Oracle ASM Instance

ASM Instance는 DB Instance가 공유 자원에 접근할 수 있도록 통제해주는 역할을 한다.

ASM Instance는 DB Instance의 요청에 따라 자원을 할당하고, DB Instance는 직접 파일을 조작한다.

대신 ASM Instance는 DB Instance의 작업이 끝나면 DB Instance 대신 Commit을 수행한다.



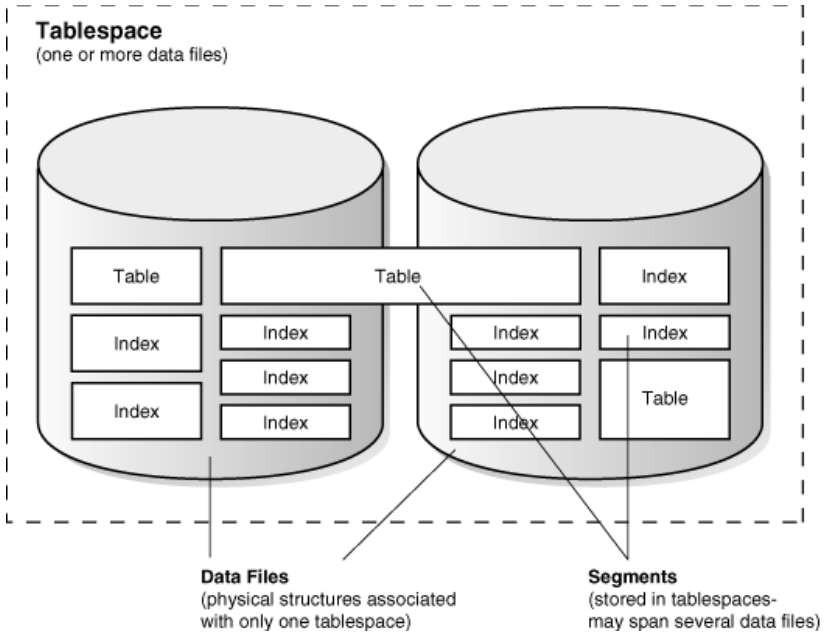
Physical Storage Structure

Overviews of Data Files

분할되지 않은 각 스키마 개체와 개체의 각 파티션은 하나의 테이블 스페이스에만 속하는 고유한 세그먼트에 저장된다.

테이블 스페이스와 데이터 파일은 다음과 같은 차이를 보인다 :

- 각각의 테이블스페이스는 하나 이상의 데이터 파일을 가진다.
- 세그먼트는 하나 이상의 데이터파일에 저장될 수 있다.



Physical Storage Structure

Permanent and Temporary Data File

영구 테이블스페이스는 일반적인 테이블 스페이스와 같다.

임시 테이블 스페이스는 세션 동안 지속되는 데이터를 저장하고, 정렬, 해쉬 같은 작업에 사용되고, 메모리 공간이 부족할 때 결과 셋을 저장한다. 임시 파일은 NOLOGGGING 모드로 설정되는데, redo 로그가 남지 않아 복구할 때 무시된다.

```
select * from v$tempfile;
```

FILE#	CREATION_CHANGE#	CREATION_TIME	TS#	RFILE#	STATUS	ENABLED	BYTES	BLOCKS	CREATE_BYTES	BLOCK_SIZE	NAME	CON_ID
1	1921092	31-JUL-23	3	1	ONLINE	READ WRITE	33554432	4096	20971520	8192	/app/ora19c/oradata/ORCL/temp01.dbf	0

Physical Storage Structure

Online and Offline Data Files

모든 데이터 파일은 온라인이거나 오프라인이다.

개별적인 데이터 파일과 임시 파일을 오프라인으로 바꾸거나 온라인으로 바꿀 수 있지만, 오프라인으로 바꾸게 되면 데이터베이스는 접근할 수 없다. 또한 데이터베이스가 데이터 파일에 접근할 수 없으면 데이터 파일을 오프라인으로 만든다.

오프라인 하는 이유 :
- 오프라인 백업
- 블록 손상 차단

```
SQL> select * from orcl_user.test;
select * from orcl_user.test
                        *
ERROR at line 1:
ORA-00376: file 7 cannot be read at this time
ORA-01110: data file 7: '/app/ora19c/oradata/ORCL/users01.dbf'
```

Physical Storage Structure

Online and Offline Data Files

```
SQL> select tablespace_name, file_id, file_name, online_status from dba_data_files where tablespace_name like '%USERS%';

TABLESPACE_NAME                                FILE_ID
------------------------------------------------------------ ----------
FILE_NAME
--------------------------------------------------------------------------------
ONLINE_STATUS
--------------
USERS                                               7
/app/ora19c/oradata/ORCL/users01.dbf
RECOVER
```

복구 필요

```
SQL> alter database datafile '/app/ora19c/oradata/ORCL/users01.dbf' online;
alter database datafile '/app/ora19c/oradata/ORCL/users01.dbf' online
*
ERROR at line 1:
ORA-01113: file 7 needs media recovery
ORA-01110: data file 7: '/app/ora19c/oradata/ORCL/users01.dbf'
```

Media recovery 이전에는 복구 불가능

```
SQL> RECOVER  DATAFILE '/app/ora19c/oradata/ORCL/users01.dbf';
Media recovery complete.
SQL> alter database datafile '/app/ora19c/oradata/ORCL/users01.dbf' online;

Database altered.
```

복구 완료

```
SQL> DESC ORCL_USER.TEST;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(38)
```

Physical Storage Structure

Data File Structure

오라클 데이터베이스는 테이블 스페이스를 위한 데이터파일을 데이터 파일 헤더와 함께 생성한다.

데이터 파일 헤더는 데이터 파일의 메타 데이터를 포함한다 :

- 체크포인트 SCN
- 데이터 파일 크기
- 절대적 파일 번호(데이터베이스에서 유일)
- 상대적 파일 번호(테이블스페이스에서 유일)

Physical Storage Structure

Data File Structure

오라클은 미래에 사용할 공간을 예약해 두고, 테이블스페이스가 커지면 빈 공간을 사용하게 된다.

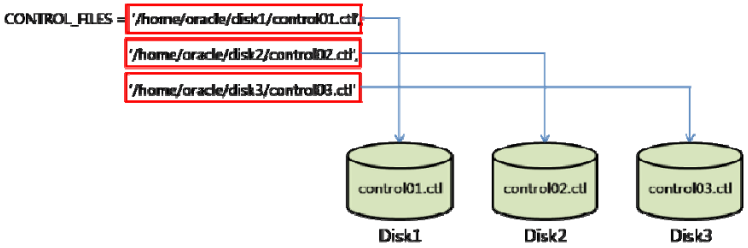
오브젝트의 수정과 삭제는 테이블스페이스에 빈 공간을 만들 수 있는데 이를 fragmented free space라고 한다.

Physical Storage Structure

Overview of Control Files

데이터베이스 컨트롤 파일은 하나의 데이터베이스에 속한 바이너리 파일이다.
각각의 데이터베이스는 하나의 유일한 컨트롤 파일을 가진다.

컨트롤 파일은 데이터 파일의 위치와 마지막으로 기록된 체크포인트의 SCN을 가지고 있기 때문에 다중화를 통해 유실에 대비해야한다.




```
select name from v$controlfile;
```

Select all rows | Save as: CSV

NAME
/app/ora19c/oradata/ORCL/control01.ctl
/app/ora19c/oradata/ORCL/control02.ctl

Physical Storage Structure

Use of Control Files

컨트롤 파일은 다음과 같은 정보를 포함한다 :

- Database name과 database unique identifier
- Database 생성 시간
- Data file, online redo log files, archived redo log files 의 정보
- Tablespace information

NAME	CREATED	DB_UNIQUE_NAME
ORCL	2023-07-19 23:00:11.000	orcl

TYPE	MEMBER
ONLINE	C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO03.LOG
ONLINE	C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO02.LOG
ONLINE	C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO01.LOG

CREATION_TIME	CHECKPOINT_CHANGE#	NAME
2021-09-28 04:24:06.000	5,224,248	C:\ORACLE_BASIC\ORCL_DATA\ORCL\SYSTEM01.DBF
2021-09-28 04:25:07.000	5,224,248	C:\ORACLE_BASIC\ORCL_DATA\ORCL\SYSAUX01.DBF
2021-09-28 08:39:43.000	5,224,248	C:\ORACLE_BASIC\ORCL_DATA\ORCL\UNDOTBS01.DBF
2021-09-28 04:26:32.000	5,224,248	C:\ORACLE_BASIC\ORCL_DATA\ORCL\USERS01.DBF

Physical Storage Structure

Control File Structure

데이터베이스에 대한 정보는 컨트롤 파일의 서로 다른 섹션에 저장되어 있다. 각각의 섹션은 record의 집합으로 이루어져 있다.

컨트롤 파일은 다음과 같은 타입의 record를 가지고 있다 :

- Circular reuse records : 덮어씌워져도 괜찮을 정도로 상대적으로 덜 중요한 정보를 포함한다.
  - Archived redo log files에 대한 정보

- Noncircular reuse records : 아주 가끔씩 변경되고 덮어씌워지면 안되는 중요한 정보를 포함한다.
  - 테이블스페이스, 데이터 파일, online redo log에 대한 정보를 저장한다.
  - 오브젝트가 테이블 스페이스로부터 drop 되기 전까지 저장한다.

컨트롤 파일을 읽고 쓰는 것은 버퍼 캐시를 사용하는 것이 아닌, 디스크에서 PGA영역으로 직접 읽고 쓰는 것이다.

Physical Storage Structure

Use of the Online Redo Log

데이터베이스는 데이터 유실을 막기 위해 online redo log file을 관리한다.

인스턴스가 실패했을 때, online redo log file은 commit 됐지만 데이터 파일에 쓰지 않은 데이터를 복구한다.

서버 프로세스는 모든 트랜잭션에 동기적으로 redo log buffer에 쓰기 작업을 하는데, LGWR 프로세스가 online redo log에 쓰기 작업을 하게 된다.

Undo segment에 변경이 이뤄지더라도 online redo log에 변경점을 기록한다. 따라서 online redo log에는 영구적인 객체에 대한 undo 데이터가 포함된다.

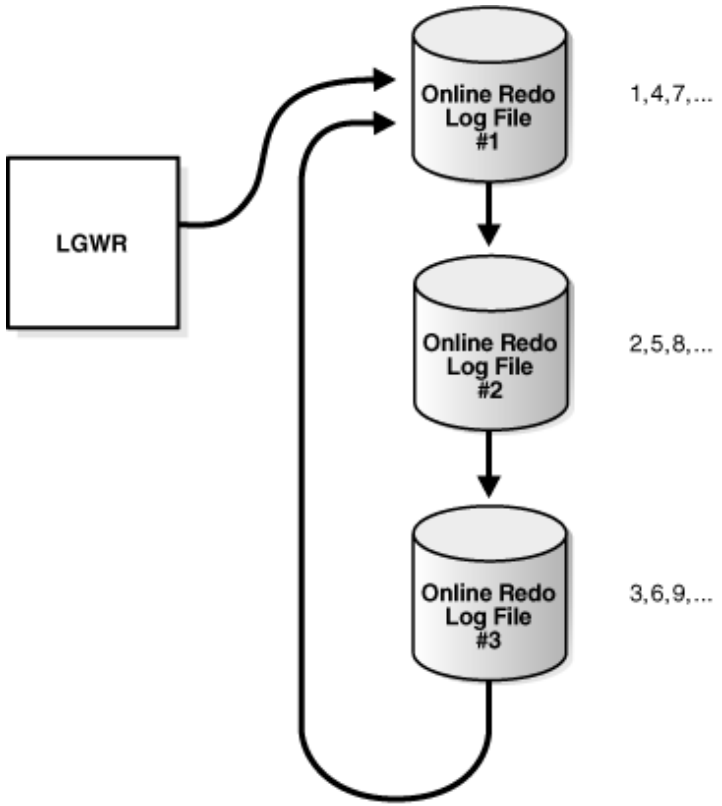
Physical Storage Structure

Online Redo Log Switches

LGWR 프로세스가 online redo log file 하나를 가득 채우면 log switch가 발생한다.

하지만 redo log의 사용량에 상관없이 일정 간격으로 강제적으로 log switch를 발생시킬 수 있다.

Redo log file은 log switch가 발생할 때마다 증가하는 log sequence number를 가지는데, archive log를 기록할 때 사용한다.

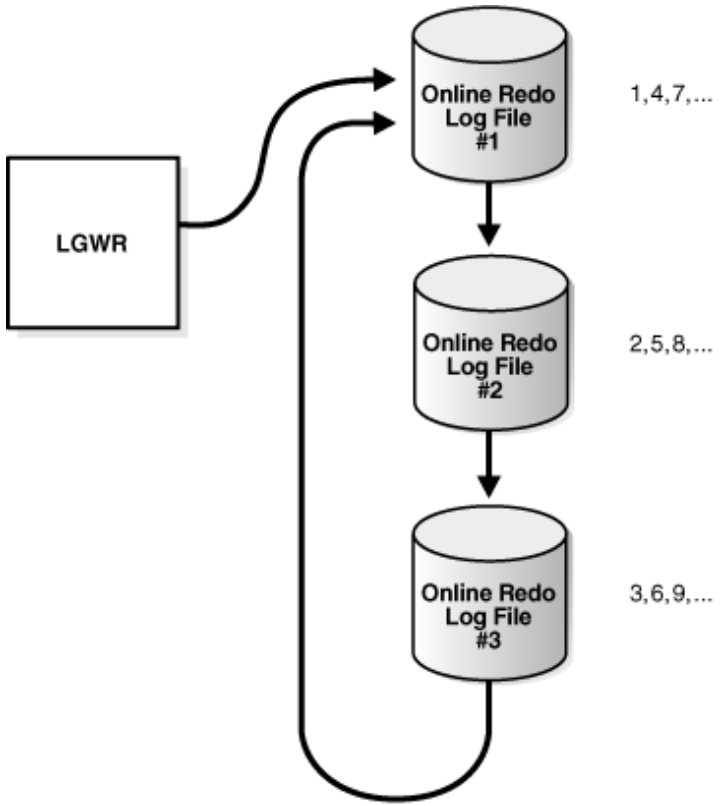


Physical Storage Structure

Online Redo Log Switches

가득 쓰여진 redo log file은 archiving mode에 따라서 사용 가능한 시점이 달라진다 :

- NOARCHIVELOG MODE : 체크포인트되어 DBW에 의해 디스크에 기록된 이후에 사용 가능해진다.
- ARCHIVELOG MODE : 데이터 파일에 기록되고 파일이 아카이브 된 이후에 사용 가능해진다.



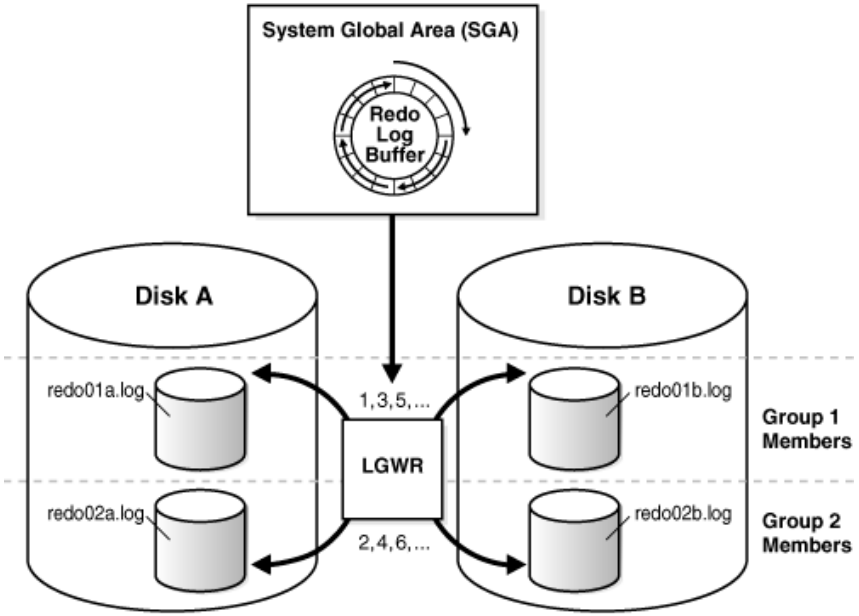
Physical Storage Structure

Multiple Copies of Online Redo Log Files

오라클 데이터베이스는 자동적으로 두개 이상의 동일한 online redo log를 다른 공간에 저장할 수 있다.

```
ALTER DATABASE ADD logfile MEMBER 'C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO01_2.LOG' TO GROUP 1;
ALTER DATABASE ADD logfile MEMBER 'C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO02_2.LOG' TO GROUP 2;
ALTER DATABASE ADD logfile MEMBER 'C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO03_2.LOG' TO GROUP 3;
```

123 GROUP#	ABC MEMBER	123 BYTES	ABC STATUS
1	C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO01.LOG	209,715,200	INACTIVE
1	C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO01_2.LOG	209,715,200	INACTIVE
2	C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO02_2.LOG	209,715,200	CURRENT
2	C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO02.LOG	209,715,200	CURRENT
3	C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO03_2.LOG	209,715,200	INACTIVE
3	C:\ORACLE_BASIC\ORCL_DATA\ORCL\REDO03.LOG	209,715,200	INACTIVE

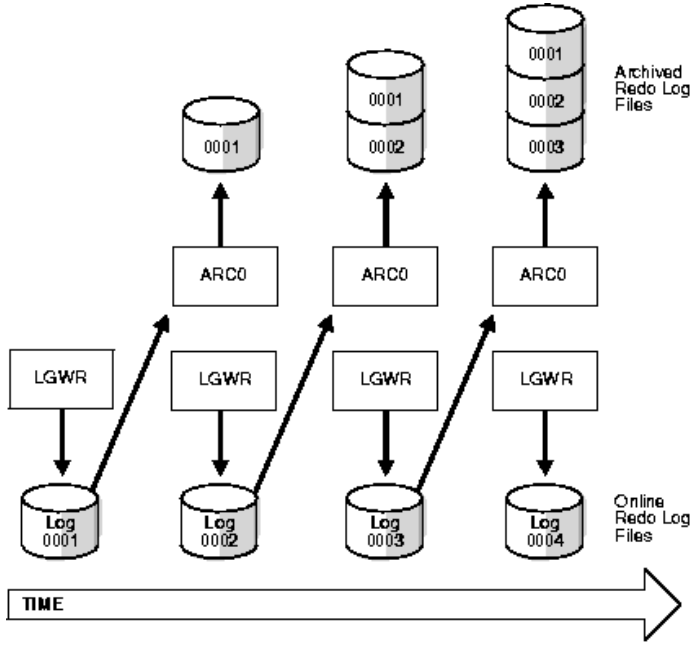


Physical Storage Structure

Archived Redo Log Files

Archived redo log file은 online redo log group의 멤버의 복사본이다.

Automatic archiving이 활성화 되어 있다면, ARCn 프로세스가 log switch가 일어난 redo log file을 archived redo log로 기록한다.

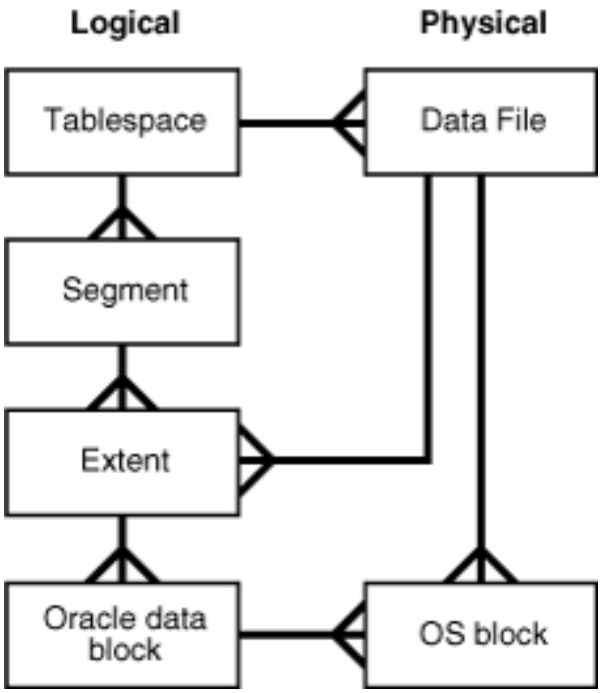


Ⅲ. Logical Storage Structure

- Introduction to Logical Storage Structures
- Data Block
- Extent
- Segment
- Tablespace

Logical Storage Structure

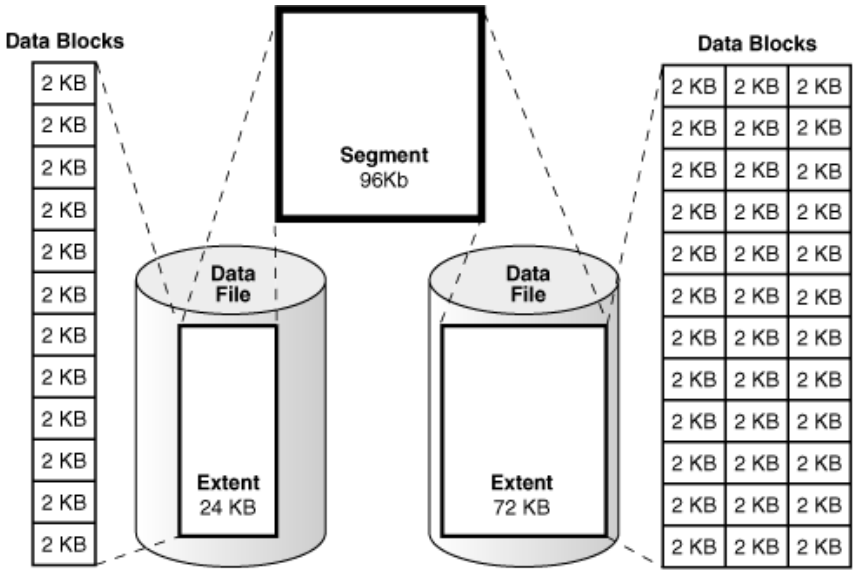
Introduction to Logical Storage Structure



Logical Storage Structure

Logical Storage Hierarchy

- Data block : 가장 작은 논리적 단위로, OS Block의 배수로 설정되어야 공간 낭비가 적음
- Extent : 연속적인 데이터 블록 집합. 디스크 할당의 최소 단위이다.
- Segment : 특정 데이터베이스 오브젝트에 할당된 Extent 집합
- Tablespace : 한 개 이상의 Segment를 가지는 스토리지

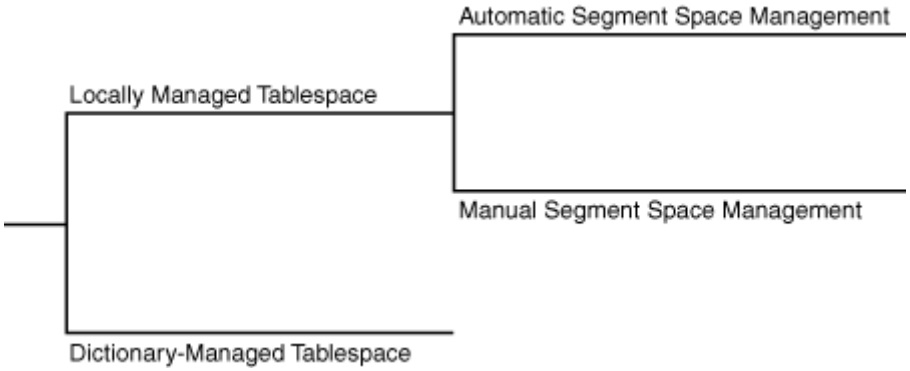


Logical Storage Structure

Logical Space Management

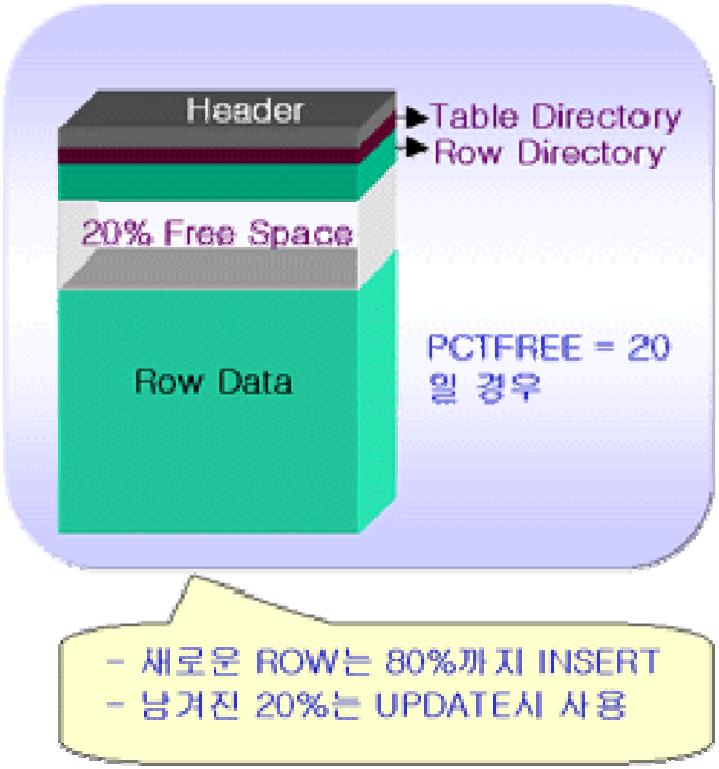
테이블 스페이스에 extent를 할당하고 추적하기 위해 logical space management를 사용해야한다.

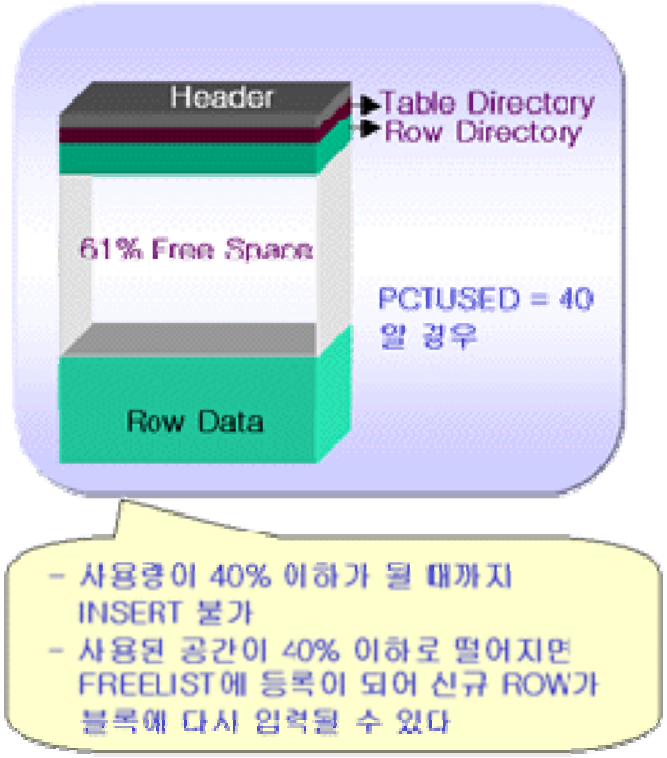
- ASSM : 수동 지정을 지양하고 PCTFREE 파라미터만 조정하여 관리한다.
- MSSM : 세그먼트를 할당하고, 사용하기 위해 다양한 파라미터를 수동으로 지정. Free list를 공간 관리를 위해 사용
- LMT : Extent의 사용 여부를 데이터 파일 헤더에서 비트맵으로 관리
- DMT : Data Dictionary에서 공간 할당을 총괄하기 대문에 경합이 생기기 쉬움



Logical Storage Structure

Logical Space Management

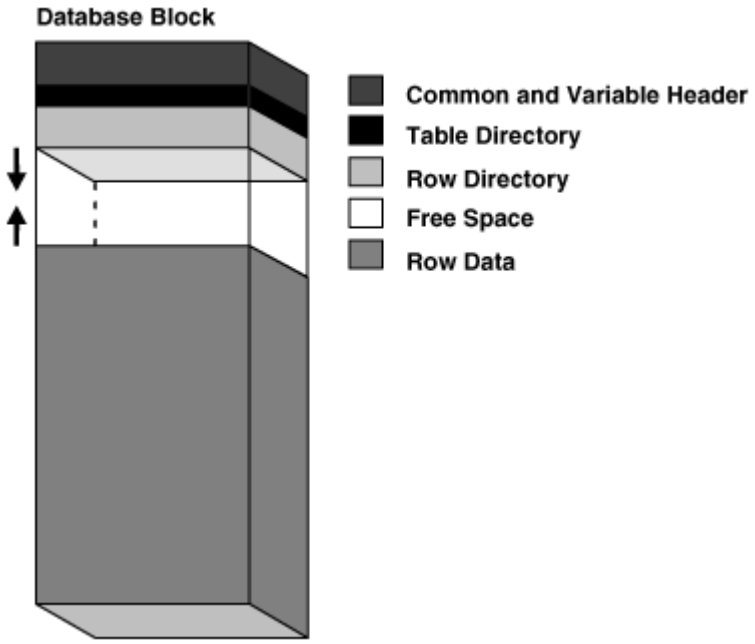




Logical Storage Structure

Data Block Format

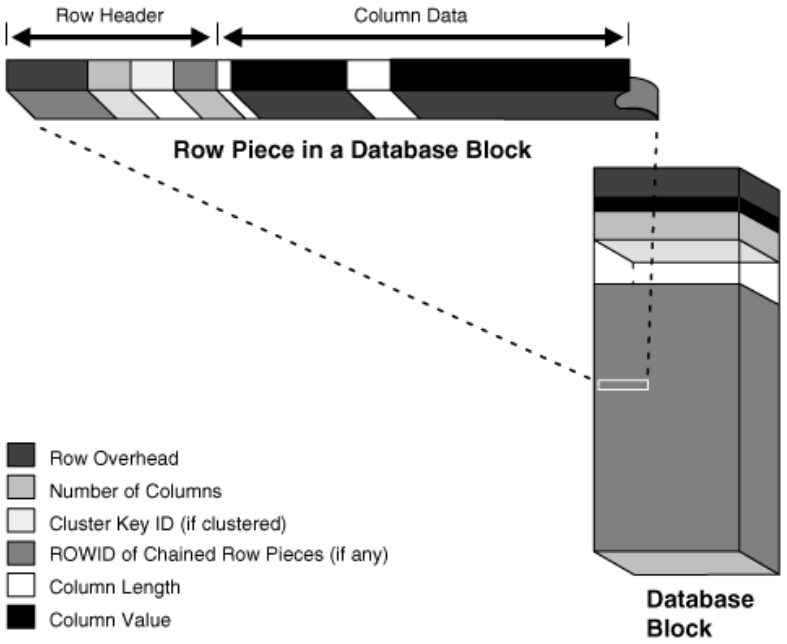
- Block header
  - Cache layer
    - Data Block Address
    - Block Type(table,index..)
    - SCN
  - Transaction Layer
    - ITL Entries
    - Free List Link 정보
    - Last Block Cleanout 시간
- Table Directory : 클러스터에서 오브젝트를 구분하기 위해 사용
- Row Directory : row의 위치 값 저장



Logical Storage Structure

Row Format

- Row Header
  - Row piece에 저장되는 column 정보
  - 다른 블록에 넘어가서 저장된 row piece 정보
  - 테이블 클러스터를 위한 클러스터 키
- Column Data
- Rowid Format



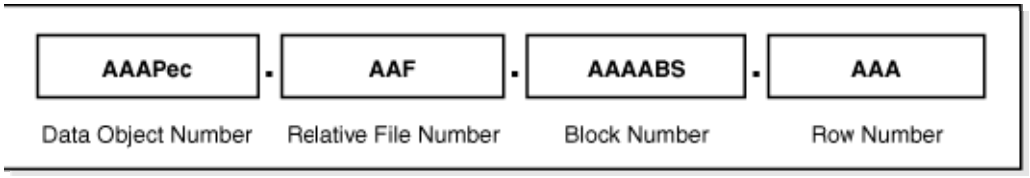
Logical Storage Structure

Row Format

- Rowid Format
  - Data Object Number : 데이터베이스 세그먼트 식별자
  - Relative File Number : 테이블스페이스 내에서 상대적인 파일 번호
  - Block Number : 데이터 파일 내에서 상대적인 블록 번호
  - Row Number : 블록 내의 row 번호

AAATd8AAHAAAAFjAAA
AAATd8AAHAAAAFjAAB
AAATd8AAHAAAAFjAAC
AAATd8AAHAAAAFjAAD
AAATd8AAHAAAAFjAAE
AAATd8AAHAAAAFjAAF
AAATd8AAHAAAAFjAAG
AAATd8AAHAAAAFjAAH
AAATd8AAHAAAAFjAAI
AAATd8AAHAAAAFjAAJ
AAATd8AAHAAAAFjAAK
AAATd8AAHAAAAFjAAL
AAATd8AAHAAAAFjAAM
AAATd8AAHAAAAFjAAN

655	AAATd8AAHAAAAFjAKO
656	AAATd8AAHAAAAFjAKP
657	AAATd8AAHAAAAFjAKQ
658	AAATd8AAHAAAAFjAKR
659	AAATd8AAHAAAAFjAKS
660	AAATd8AAHAAAAFjAKT
661	AAATd8AAHAAAAFkAAA
662	AAATd8AAHAAAAFkAAB
663	AAATd8AAHAAAAFkAAC
664	AAATd8AAHAAAAFkAAD



Logical Storage Structure

Optimization by Increasing Free Space

다음과 같은 질의는 남는 공간을 증가시킨다

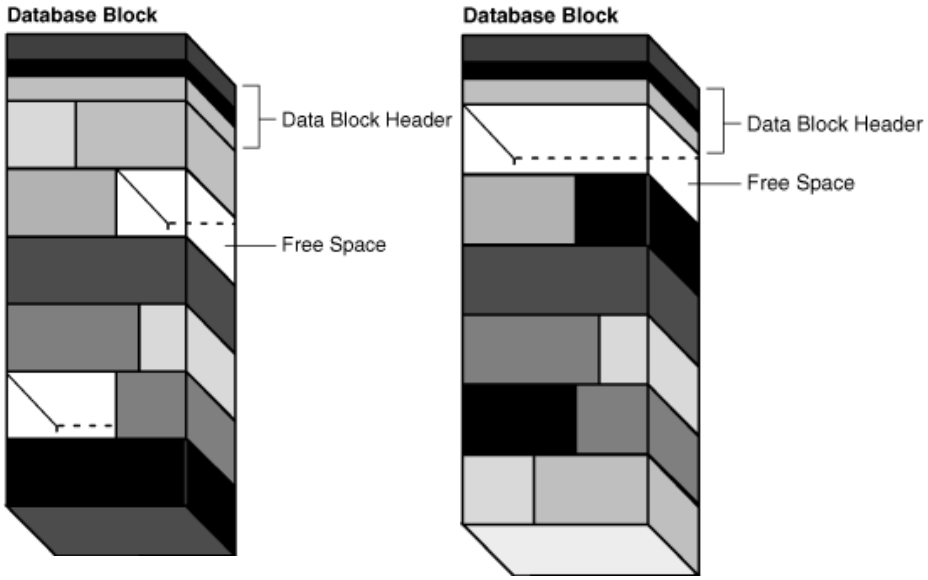
- DELETE
- 이전 값 보다 작은 값이거나 row migration을 유도하는 UPDATE
- Row compression을 사용하는 INSERT문

Logical Storage Structure

Optimization by Coalescing Fragmented Space

오라클은 다음 조건이 충족되는 경우에만 데이터 블록의 사용 가능한 공간을 자동으로 병합한다 :

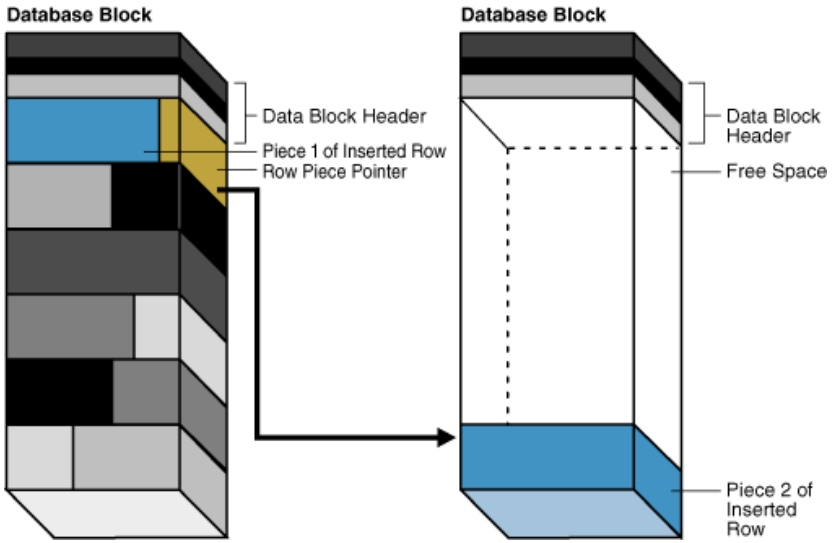
- INSERT나 UPDATE 질의가 row piece가 충분히 들어가는 여유 공간이 포함된 블록을 사용하려고 함
- 여유 공간이 단편화 되어 있어 row piece가 삽입될 수 있다.



Logical Storage Structure

Chained and Migrated Rows

Row Chaining은 크기가 큰 row를 저장할 때 한 개 이상의 블록을 사용하여 저장하는 것을 말한다.

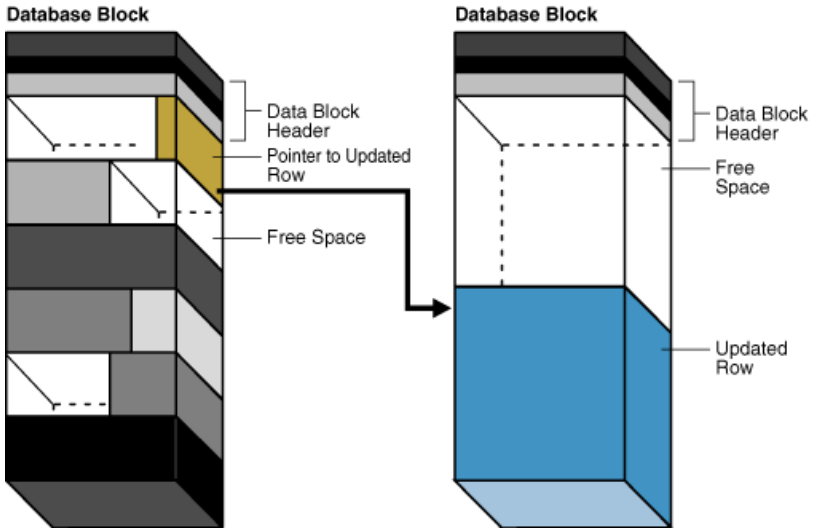


Logical Storage Structure

Chained and Migrated Rows

Row Migration은 데이터 변경으로 인해 전체적은 row의 길이가 길어져 블록의 여유 공간보다 커졌을 때 발생한다.

Row Migration과 Row Chaining 모두 블록을 읽는데 추가 I/O가 발생한다.



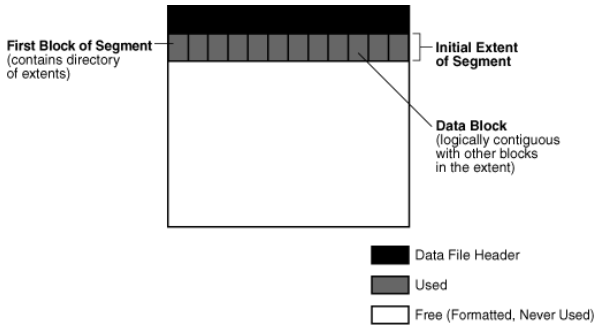
Logical Storage Structure

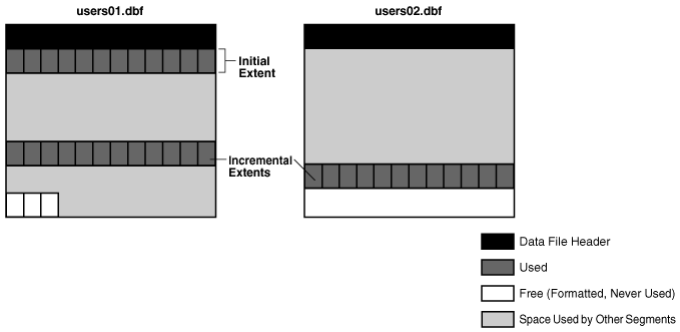
Allocation of Extent

데이터베이스는 세그먼트가 생성됐을 때 초기 extent를 할당한다.

Extent를 할당할 때 데이터파일의 비트맵에서 빈 블록을 검색한다.

세그먼트의 Extent는 다른 파일에 존재할 수 있기 때문에 Extent를 추가로 할당 받을 때 다른 파일에 할당 받을 수 있다.





Logical Storage Structure

Deallocation of Extent

일반적으로는 Extent는 할당 받은 공간을 되돌려주지 않는다.

몇몇 상황에서 수동으로 공간을 회수할 수 있다 :

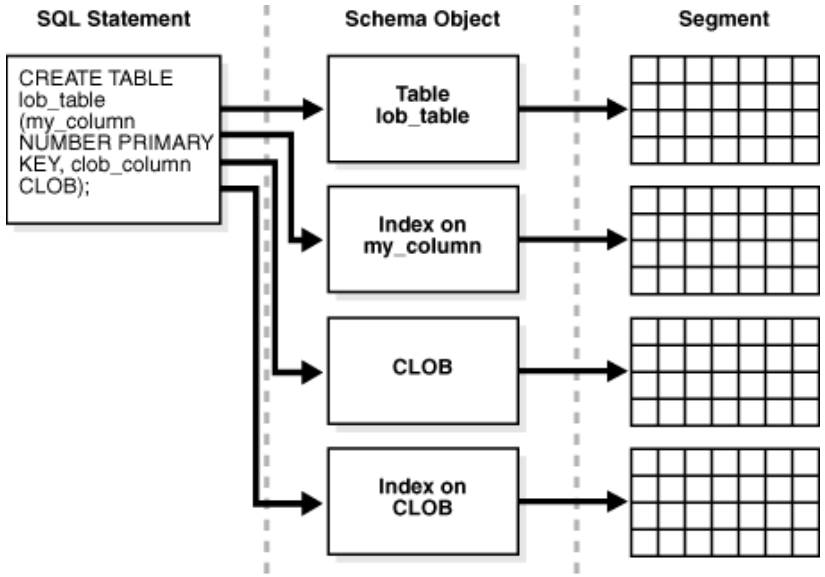
- Online segment shrink
- 테이블의 데이터를 다른 세그먼트로 옮김
- Truncate table

Logical Storage Structure

User Segments Creation

기본적으로 데이터베이스는 테이블, 인덱스 및 파티션을 작성할 때 지연된 세그먼트 생성을 사용하여 데이터베이스 메타데이터만 업데이트한다.

유저가 테이블에 데이터를 추가할 때 세그먼트를 생성한다.



Logical Storage Structure

Undo Segments

오라클은 undo data를 다음을 위해 사용한다 :

- 트랜잭션의 롤백
- 종료된 트랜잭션의 복구
- 읽기 일관성 제공

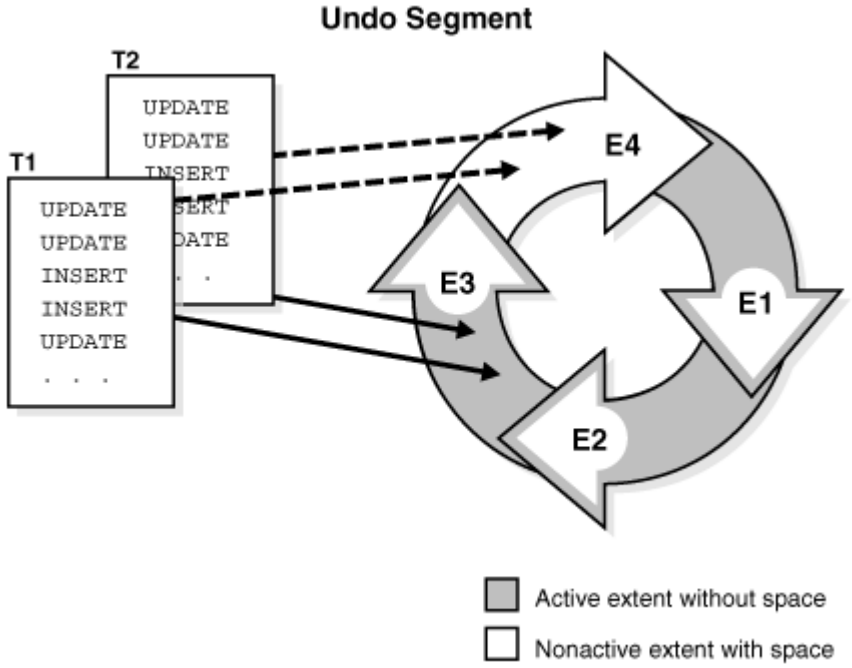
오라클은 undo 데이터를 블록 내부에 저장하여 외부 로그를 읽을 필요가 없어 효율적으로 조회가 가능하다. 또한 영구적인 오브젝트의 undo 데이터는 undo tablespace에 저장된다.

Logical Storage Structure

Undo Segments and Transactions

트랜잭션이 시작되면, 데이터베이스는 트랜잭션을 undo segment에 바인딩 한다.

인스턴스가 undo tablespace를 지정받지 못했다면, system undo segment(system tablespace)에 저장된다.



Logical Storage Structure

Temporary Undo Segments

Temporary undo segment는 temporary undo 데이터만을 위한 공간이다.

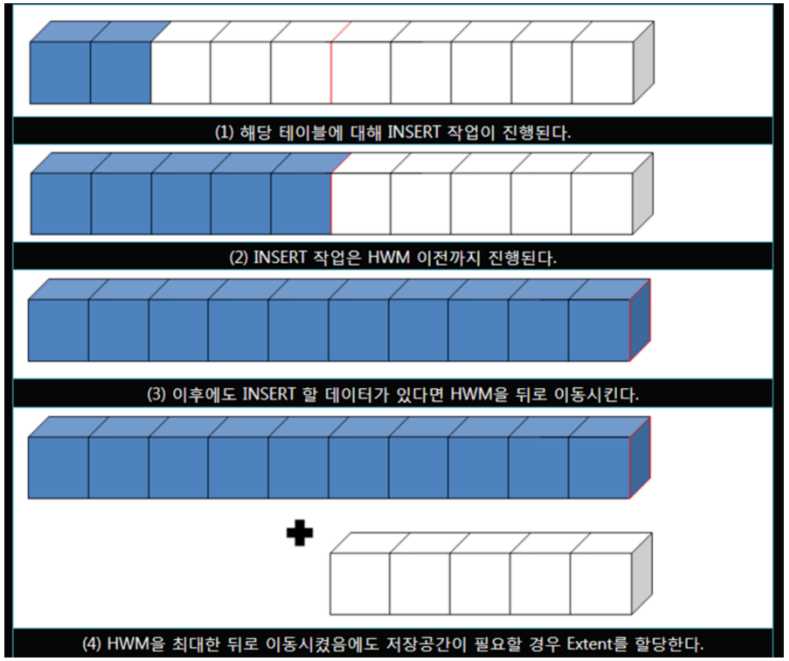
임시 테이블과 임시 테이블의 변경을 위한 undo record는 모두 특정 세션에서만 읽을 수 있고 읽기 일관성과 트랜잭션 롤백에 유용하다.

TEMP_UNDO_ENABLES 파라미터가 TRUE이면, 데이터베이스는 temporary tablespace에 temporary undo segment를 할당한다

Logical Storage Structure

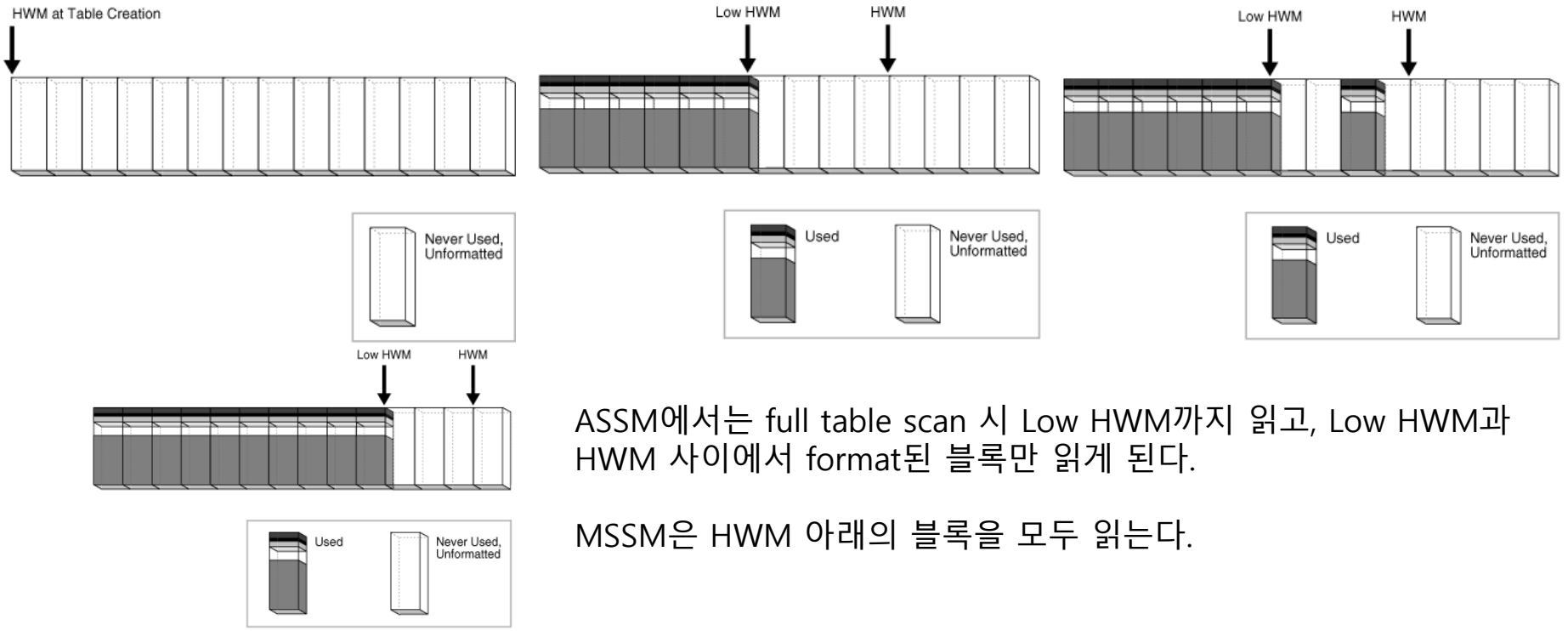
Segment Space and the High Water Mark

High water mark는 이 지점 위로는 블록이 포맷되지 않았고 사용되지 않은지 알려주는 위치 지정자이다.



Logical Storage Structure

Segment Space and the High Water Mark



ASSM에서는 full table scan 시 Low HWM까지 읽고, Low HWM과 HWM 사이에서 format된 블록만 읽게 된다.

MSSM은 HWM 아래의 블록을 모두 읽는다.

Logical Storage Structure

Permanent Tablespace

Tablespace name	설명	포함하는 정보
SYSTEM	Sys 유저에 의해 관리되는 데이터베이스 관리를 위한 테이블 스페이스	Data dictionary, Trigger, procedure, table, view 정보
SYSAUX	SYSTEM 테이블 스페이스의 부하 분산용.	
UNDO	Undo 데이터를 위한 테이블스페이스	Undo 데이터
SHADOW	Shadow lost write를 방지하기 위한 테이블 스페이스	

Logical Storage Structure

Automatic Undo Retention

Undo retention period는 old undo data를 덮어쓰기 전에 유지시켜주는 시간을 말한다.
오래 작동되는 트랜잭션의 읽기 일관성을 위해 retention period는 중요하다.

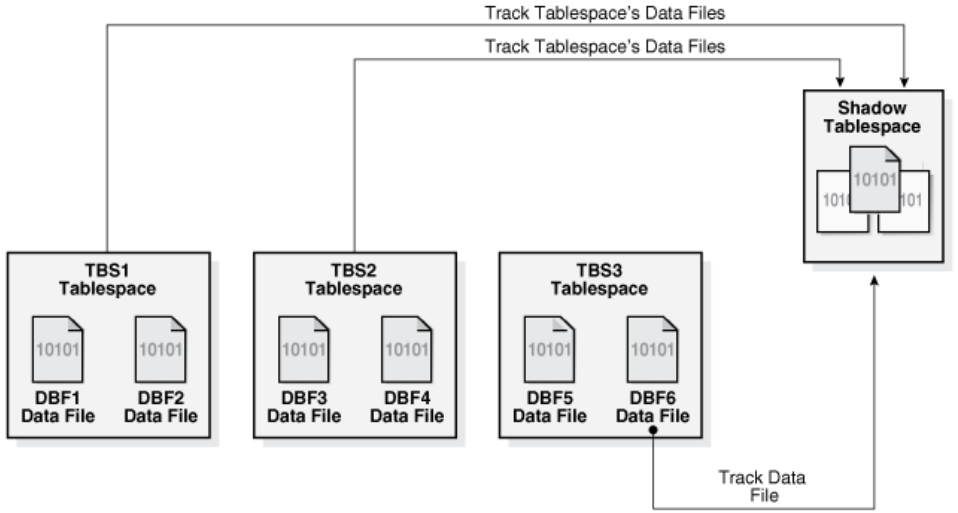
Undo tablespace의 공간이 부족해지면, AUTOEXTENT 옵션이 켜져 있을 경우 자동으로 공간을 할당하며 retention period를 좀 더 길게 잡게 되고, 일반적인 상황에서는 old undo data 를 덮어씌우게 된다

Logical Storage Structure

Shadow Tablespaces

Shadow tblespace는 shadow lost write 를 방지하기 위한 bigfile tablespace이다.

하나의 추적되는 데이터 파일은 shadow tablespace의 extent에 매핑되고, 데이터 블록 또한 shadow block의 entry에 대응된다.
각각의 entry는 추적하는 block의 가장 최근에 쓰여진 SCN을 저장한다.
만약 추적하는 block의 SCN이 더 작다면, 데이터 유실이 발생한 것이므로 에러를 발생시킨다.



Logical Storage Structure

Temporary Tablespaces

- Shared temporary tablespace는 temp file을 shared disk에 저장해서 모든 데이터베이스 인스턴스가 접근 가능하다.
- Local temporary tablespace는 각각의 데이터베이스 인스턴스가 공유되지 않는 temp file을 생성한다.
- Default Temporary Tablespace는 모든 유저 계정이 접근 가능한 Shared temporary tablespace이다.

Locally managed SYSTEM tablespace	Create with Specific temporary tablespace	데이터베이스는..
YES	YES	특정 테이블스페이스 사용
YES	NO	임시 테이블스페이스 생성
NO	YES	특정 테이블스페이스 사용
NO	NO	SYSTEM을 temporary storage로 사용한다.

Logical Storage Structure

Online and Offline Tablespaces

테이블스페이스는 open 중에는 online(접근 가능)이거나 offline(접근 불가능) 상태이다.
단, SYSTEM 테이블스페이스와 임시 테이블스페이스는 offline이 될 수 없다.

테이블 스페이스가 offline 상태이면, 데이터베이스는 다음과 같이 행동한다 :

- 데이터베이스에서 offline 테이블스페이스의 참조 개체에 대한 DML문을 허용하지 않음

- 데이터베이스는 완료된 질의에 해당하는 undo 데이터를 SYSTEM 테이블스페이스의 deferred undo segment에 저장한다. 나중에 온라인 상태가 되면 undo 데이터를 적용시킨다.

IV. Oracle Database Instance

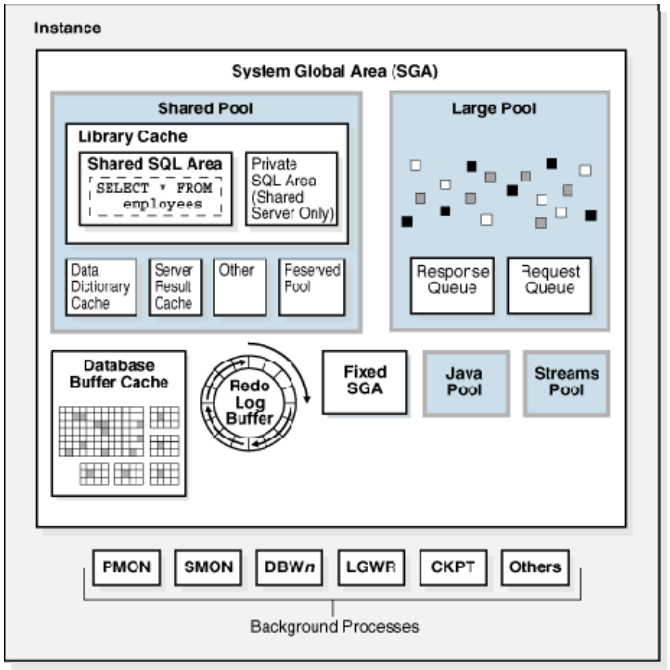
- Introduction to the Oracle Database Instance
- Overview of Chekcpoint
- Overview of Instance Recovery
- Overview of Parameter File

Oracle Instance Architecture

Introduction to the Oracle Database Instance

데이터베이스 인스턴스는 데이터베이스 파일을 관리하는 메모리 구조이다.

인스턴스가 기동되면 SGA를 할당하고 백그라운드 프로세스를 시작한다



Oracle Instance Architecture

Overview of Database Instance Startup and Shutdown

Phase	Mount State	Description
1	데이터베이스 마운팅 없이 인스턴스 시작	인스턴스가 시작됐지만, 데이터베이스와 연결되진 않음
2	데이터베이스 마운트	인스턴스가 컨트롤 파일을 읽어 데이터베이스와 연결됨
3	데이터베이스 오픈	데이터베이스가 오픈되어 유저가 접근 가능

Oracle Instance Architecture

How an Instance Is Started

1. Server parameter file을 찾고, 없으면 initialization parameter file을 사용한다
2. Initialization parameter settin을 기반으로 SGA를 할당한다
3. 백그라운드 프로세스를 시작한다
4. Alert log와 trace file을 열고 유효한 파라미터 문법으로 명시된 파라미터 세팅을 alert log에 쓴다.

이 시점에는 데이터베이스가 인스턴스와 연관되어있지 않다.

Oracle Instance Architecture

How a Database Is Mounted

데이터베이스를 마운트 하기 위해서는, 컨트롤 파일을 통해 데이터파일과 online redo log 파일의 이름을 찾는다.

마운트된 데이터베이스에서는 데이터베이스는 닫혀있고 데이터베이스 관리자만 접근할 수 있다.

Oracle Instance Architecture

How a Database Is Opened

마운트된 데이터베이스를 open하면 일반적인 데이터베이스 작업을 할 수 있다.

데이터베이스를 열면, 오라클은 다음과 같은 작업을 한다 :

- Online data file을 연다
- Undo tablespace를 요규한다
- Online redo log file을 연다

Oracle Instance Architecture

Overview of Database and Instance Shutdown

Phase	Mount State	Description
1	Database closed	데이터베이스는 마운트 되어 있지만, online redo log file과 datafile이 닫힘
2	Database unmounted	인스턴스는 가동하고있지만, 데이터베이스의 컨트롤 파일과 연관은 사라짐
3	Database instance shutdown	데이터베이스 인스턴스가 종료됨

Oracle Instance Architecture

Database Shutdown Modes

Database Behavior	ABORT	IMMEDIATE	TRANSACTIONAL	NORMAL
Permit new user connection	No	No	No	No
Waits until current sessions end	No	No	No	Yes
Waits until current transactions end	No	No	Yes	Yes
Performs a checkpoint and closes open file	No	Yes	Yes	yes

Oracle Instance Architecture

How a Database Is Closed

Normal Shutdown
- 오라클은 SGA의 데이터를 데이터 파일과 online redo log에 쓴다
- 그 후에 데이터베이스는 data file과 online redo log file을 닫는다.

Abnormal Shutdown
- 데이터베이스를 즉시 닫았기 때문에 SGA의 데이터를 data file과 redo log에 쓰지 못했다.
- 데이터베이스 오픈 시 자동적으로 수행되는 인스턴스 복구를 요구한다.

Oracle Instance Architecture

How a Database Is Unmounted and Shut Down

데이터베이스가 닫힌 이후, 오라클은 인스턴스로부터 데이터베이스를 unmount 시킨다.
데이터베이스가 unmount된 이후 오라클은 컨트롤 파일을 닫는다.

데이터베이스 종료의 마지막 단계는 인스턴스를 종료하는 것이다.
데이터베이스 인스턴스가 종료되면 SGA가 메모리 점유를 중지하고 백그라운드 프로세스가 종료된다.

Oracle Instance Architecture

When Oracle Database Initiates Checkpoints

CKPT는 데이터 파일 헤더와 컨트롤 파일을 작성하는 역할을 한다.

체크 포인트는 다음과 같은 타입을 가진다 :

- Thread checkpoints : 인스턴스 메모리에서 수정된 모든 버퍼를 디스크에 기록한다
  - Consistent database shutdown
  - Alter system checkpoint
  - Online redo log switch
  - Alter database begin backup
- Tablespace and data file checkpoints : tablespace 내의 각 datafile에 대한 datafile checkpoint의 집합
  - Alter tablespace read only
  - Alter tablespace offline
  - Shrrinking a datafile
- Incremental checkpoints : log switch에서 많은 수의 블록을 기록하지 않도록 하기 위한 checkpoint 유형. 데이터 파일에는 SCN을 기록하지 않는다.

Oracle Instance Architecture

Instance Recovery

가장 최근의 checkpoint 이후에 이뤄진 변경을 redo log file을 통해 복구하는 방법

트랜잭션이 커밋되면, LGWR은 메모리의 redo entry와 transaction SCN을 online redo log에 기록하지만, DBW 프로세스는 변경 block을 즉시 기록하지 않는다.
그렇기 때문에 커밋 되지 않은 변경사항이 데이터 파일에 남아있을 수 있다.

오라클은 다음과 같은 상황에 인스턴스 복구를 진행한다 :

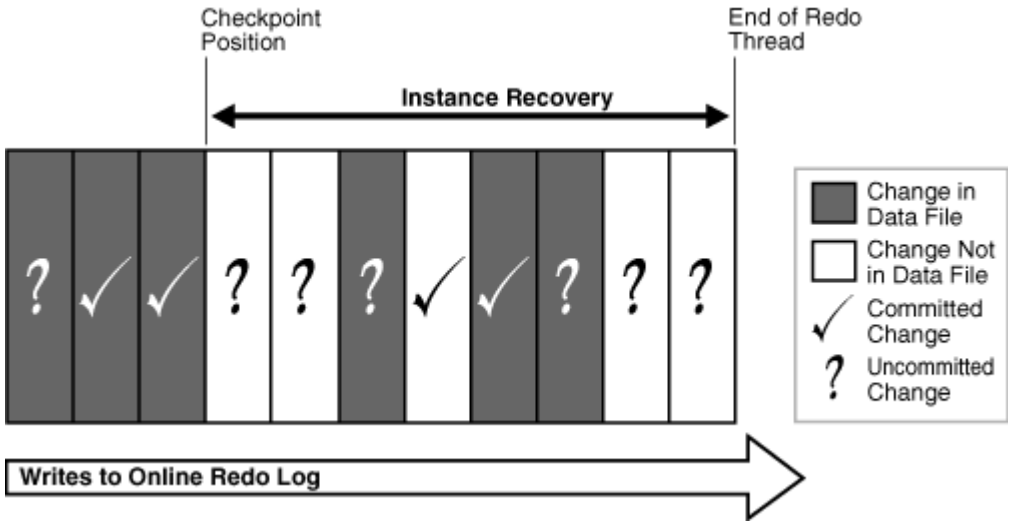
- 비정상 종료 이후 처음으로 데이터베이스를 오픈 할 때.
- Oracle RAC의 일부 인스턴스가 비정상 종료 됐을 때.

Oracle Instance Architecture

Importance of Checkpoints for Instance Recovery

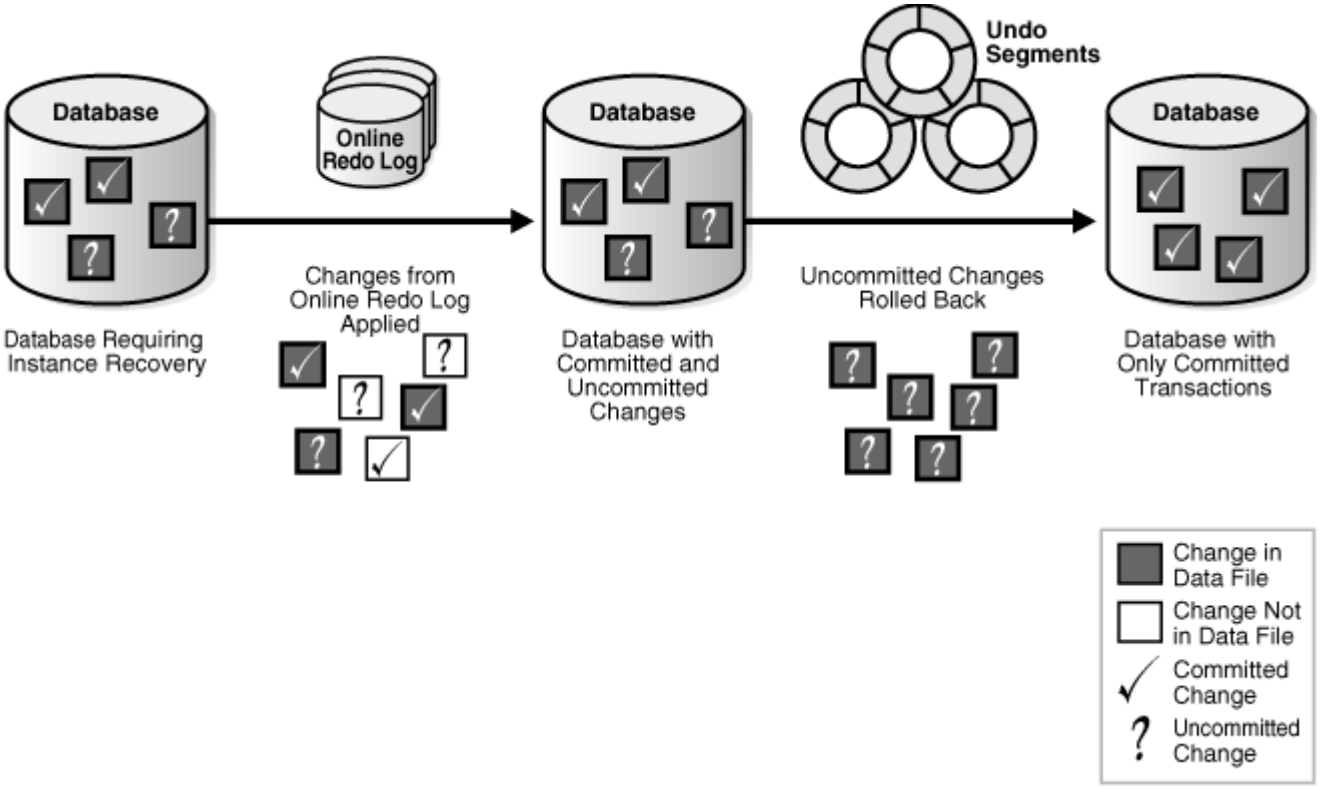
인스턴스 복구는 데이터 파일에 어떤 변경사항이 적용되어야 하는지 결정하기 위해 체크포인트를 사용한다.

체크포인트는 체크포인트 SCN보다 작은 변경에 대해서는 데이터 파일에 기록된 것을 보장한다.



Oracle Instance Architecture

Instance Recovery Phases



Oracle Instance Architecture

Overview of Parameter Files

데이터베이스 인스턴스를 시작하기 위해서는 반드시 server parameter file(추천)이나 text initialization parameter file을 읽어야 한다.

Server Parameter File(spfile)
- 오라클 데이터베이스만이 서버 파라미터 파일을 읽고 쓴다
- 오직 하나의 서버 파라미터 파일이 데이터베이스 호스트에 존재해야한다.
- 서버 파라미터 파일은 바이너리 파일이고, 텍스트 에디터로 수정될 수 없다.
- 서버 파라미터 파일의 파라미터는 지속적이기 때문에, 인스턴스 실행 중 일어난 변경도 영구적으로 적용된다.

Text Initialization Parameter File
- 이 파일은 텍스트 기반이다.
- 오라클은 이 파일을 읽을 수 있지만, 쓸 수는 없다. 파라미터를 변경하고 싶으면 수동 변경해야한다.
- ALTER SYSTEM으로 변경한 파라미터 값은 현재 인스턴스에만 반영된다.
- 이 파일은 클라이언트 어플리케이션의 호스트와 같은 곳에 저장되어 있어야 한다.

Oracle Instance Architecture

Overview of Parameter Files

파라미터 변경의 SCOPE는 변경 사항이 적용되는 시기에 따라 달라진다.

인스턴스가 spfile과 함께 시작됐을 때 ALTER SYSTEM SET 질의로 파라미터를 변경할 수 있다

- SCOPE = MEMORY
  - 데이터베이스 인스턴스에만 변경이 적용된다.
  - 데이터베이스가 종료되거나 재시작하면 사라진다

- SCOPE = SPFILE
  - Spfile에 변경을 적용시키지만 현재 인스턴스에는 영향을 끼치지 않는다

- SCOPE = BOTH
  - 기본 설정 scope로, 메모리와 spfile에 둘 다 기록한다.

Oracle Instance Architecture

Overview of Parameter Files

NAME	VALUE	ISSYS_MODIFIABLE
open_cursors	300	IMMEDIATE

변경 전

```
SQL> alter system set open_cursors=400 SCOPE=SPFILE;

System altered.

SQL> select name, value, issys_modifiable from v$parameter where name='open_cursors' ;
```

Select all rows | Save as: CSV

	NAME	VALUE	ISSYS_MODIFIABLE
☑	open_cursors	300	IMMEDIATE

CURSOR=SPFILE
변경 후

```
SQL> alter system set open_cursors=200 SCOPE=MEMORY;

System altered.

SQL> select name, value, issys_modifiable from v$parameter where name='open_cursors' ;
```

Select all rows | Save as: CSV

	NAME	VALUE	ISSYS_MODIFIABLE
☑	open_cursors	200	IMMEDIATE

CURSOR=MEMORY
변경 후

```
SQL> select name, value, issys_modifiable from v$parameter where name='open_cursors' ;
```

Select all rows | Save as: CSV

	NAME	VALUE
☑	open_cursors	400

데이터베이스 재시작

Oracle Instance Architecture

Overview of Diagnostic Files

오라클은 데이터베이스 문제를 해결, 진단, 파악, 예방하기 위해 fault diagnosability infrastructure를 포함한다. 여기서 문제는 코드 버그, 메타데이터 오염, 고객 데이터 오염이 있다.

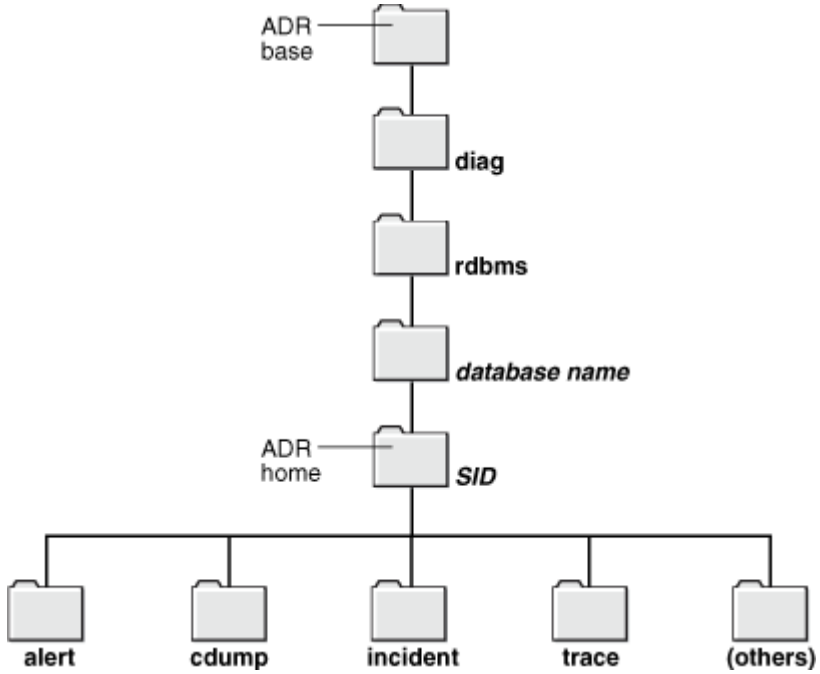
- ADR : trace file, alert log, ddl log 등을 저장하는 파일 기반 저장소다.
- Alert Log : 데이터베이스 메시지 및 오류의 시간순 로그를 포함하는 XML 파일이다
- DDL Log : alert log와 비슷한 포맷이지만 DDL 질의만 저장한다.
- Trace Files : 문제를 조사하기 위해 사용되는 진단 데이터이다.
- Diagnostic Dumps : 시간별 상세 정보를 기록하는 특별한 타입의 트레이스 파일이다.

Oracle Instance Architecture

ADR Structure

```
[ora19c@moti DB19]$ pwd
/app/ora19c/diag/rdbms/orcl/DB19
[ora19c@moti DB19]$ ls
alert  cdump  hm  incident  incpkg  ir  lck  log  metadata  metadata_dgif  metadata_pv  stage  sweep  trace
```

NAME	VALUE
Diag Enabled	TRUE
ADR Base	/app/ora19c
ADR Home	/app/ora19c/diag/rdbms/orcl/DB19
Diag Trace	/app/ora19c/diag/rdbms/orcl/DB19/trace
Diag Alert	/app/ora19c/diag/rdbms/orcl/DB19/alert
Diag Incident	/app/ora19c/diag/rdbms/orcl/DB19/incident
Diag Cdump	/app/ora19c/diag/rdbms/orcl/DB19/cdump
Health Monitor	/app/ora19c/diag/rdbms/orcl/DB19/hm
Default Trace File	/app/ora19c/diag/rdbms/orcl/DB19/trace/DB19_ora_746617.trc
Active Problem Count	0



Q & A

감사합니다