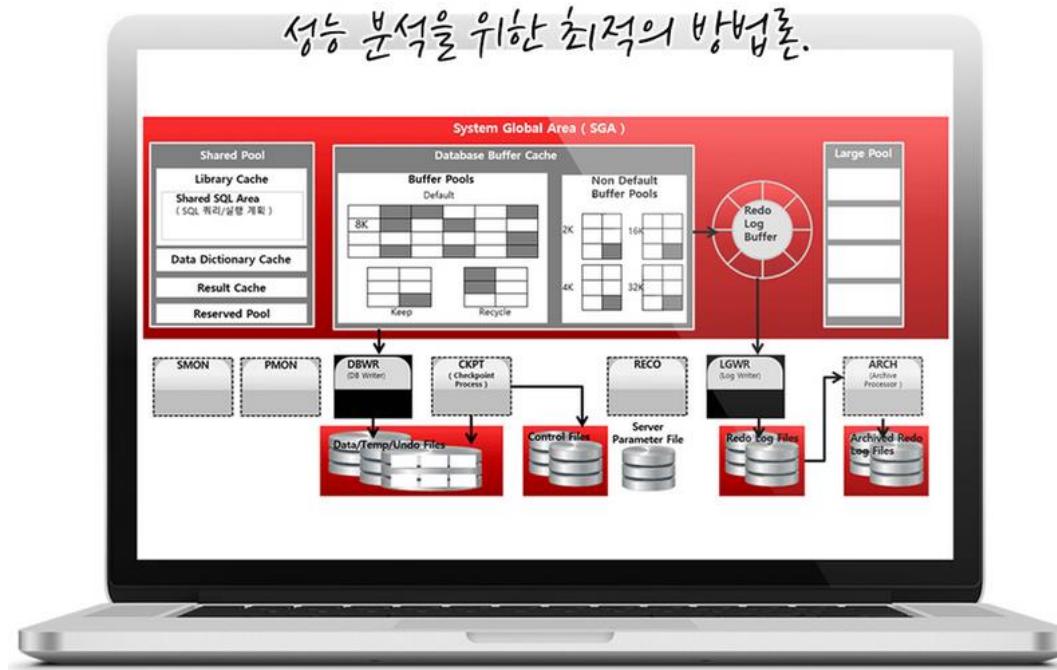


오라클 성능 분석과 튜닝 핵심 가이드

명쾌한 DBMS 기반기 +

성능 분석을 위한 최적의 방법론.



권철민

실습 환경 구축하기

실습 환경 구축하기

강의는 Oracle 18c Enterprise Edition에서 테스트 됨

- **Oracle Cloud에서 서버 구축**
 - 30일 무료 Credit를 활용. 2 Core CPU, 250GB의 Usable NVMe SSD 제공
 - Oracle Enterprise Edition 사용 가능
- **Oracle 18c XE버전을 Local PC 또는 Cloud에 구축**
 - Admin, 초기화 파일 설정 등의 실습 가능
 - Load 테스트를 강의와 동일하게 구현할 수는 없음.
 - 18c XE는 AWR이 지원되지 않음.
- **Oracle 19c Enterprise 버전을 Google Cloud에 구축**
 - Google Cloud는 최초 가입 시 300\$ 무료 Credit을 1년간 제공
 - VM을 생성하고 60GB 정도의 SSD Block storage를 구성하여 DB 구축
 - 60GB SSD Block Storage의 비용 유의

설정 시 주의 사항 – SYS Password

- SYS 패스워드는
 - 초기에 VERYWelcome123_#
 - 추후에 welcome1로 변경

Oracle Cloud에서 실습 DB 생성 방법

1. 오라클 클라우드에서 Multi Tenant DB 서비스 생성

- VCN 마법사로 VCN, Subnet, Internet Gateway 생성
- DB 생성(CPU 2 core, 250GB Usable 용량, NVMe SSD, 18c Enterprise Edition)
- 해당 VCN, Subnet의 Default Security list에서 1521번 포트 방화벽 오픈. Telnet Client에서 방화벽 오픈 확인

2. 생성된 Multi Tenant DB 삭제하고 새롭게 Non CDB 생성

- 기존 Multi Tenant DB인 OTCL을 delete_db.sh로 삭제
- 새로운 Non CDB인 ORCL을 create_noncdb.sh로 생성

3. Client PC에 SQL*Net 구성

- 로컬 PC에 Oracle Client S/W 설치 후 tnsnames.ora 설정
- DB 서버에 Listener 재 설정(Optional)

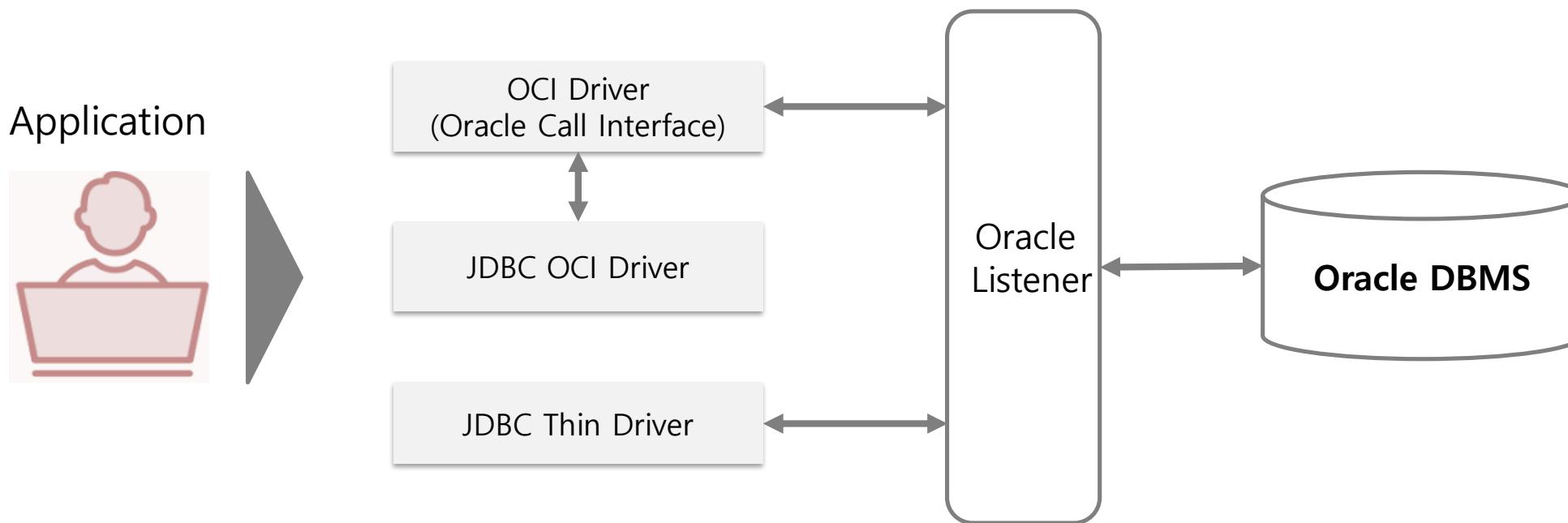
4. 기타 Post Installation 수행

Oracle 18c XE(eXpress Edition)

- 무료
- 2 CPU Thread, 2GB RAM, 12GB 의 User 공간 제약
- Partition 기능 제공, AWR 기능 제공하지 않음
- Default로 Multi Tenant DB 셋업.
- Non CDB 생성 위해서는 Default Multi Tenant DB 삭제하고 DBCA등으로 재 생성 필요.

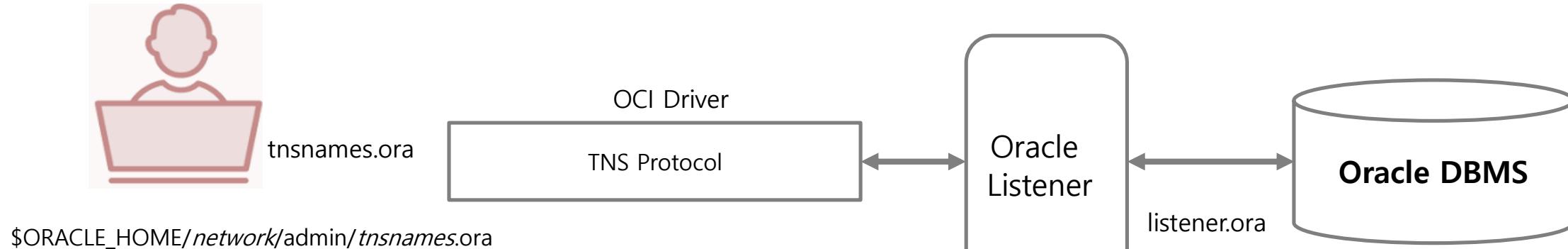
Oracle 접속 Client Driver 유형

- Oracle DB에 접속하기 위한 Client Driver는 크게 C/C++ 기반의 OCI(Oracle Call Interface)와 Java 기반의 JDBC Thin Driver가 있음
- OCI는 Application을 구동하는 클라이언트 운영 체제의 C/C++(Microsoft, Unix, Linux)로 만들어졌으며, Oracle의 SQL*Net 기반에서 구동
- ODBC, ODP for .Net, JDBC OCI Driver 모두 최종적으로 OCI Driver를 이용하여 오라클 DB에 연결
- OCI는 운영체제별로 별도의 클라이언트 툴로 설치 해야함.
- JDBC Thin Driver는 Java로 만들어졌으며 TCP 기반으로 접속되며 SQL*Net 기반의 추가적인 클라이언트 설치 불필요. OCI 보다 상대적으로 가볍고, Java가 설치되어 있는 클라이언트에서는 어디서나 빠르게 오라클 DB 접속 가능



SQL*Net 구성

- SQL*Net은 오라클 클라이언트와 오라클 DB서버가 서로 인터페이스 하는 전반적인 네트워크 구성의 의미
- SQL*Net TNS Protocol을 기반으로 하고 있으며 세개의 환경파일인 tnsnames.ora, listener.ora, sqlnet.ora로 구성
- OCI Driver를 활용하기 위해서는 반드시 tnsnames.ora가 구성되어야 함.



tnsnames.ora와 listener.ora 구성

tnsnames.ora

```
ORCL =  
(DESCRIPTION =  
  (ADDRESS_LIST =  
    (ADDRESS = (PROTOCOL = TCP)(HOST = 34.64.xxx.xxx)(PORT = 1521))  
  )  
  (CONNECT_DATA =  
    (SERVICE_NAME = ORCL) 접속 Listener가 관리하는 서비스명  
  )  
)  
SID=ORCL 도 무방
```

listener.ora

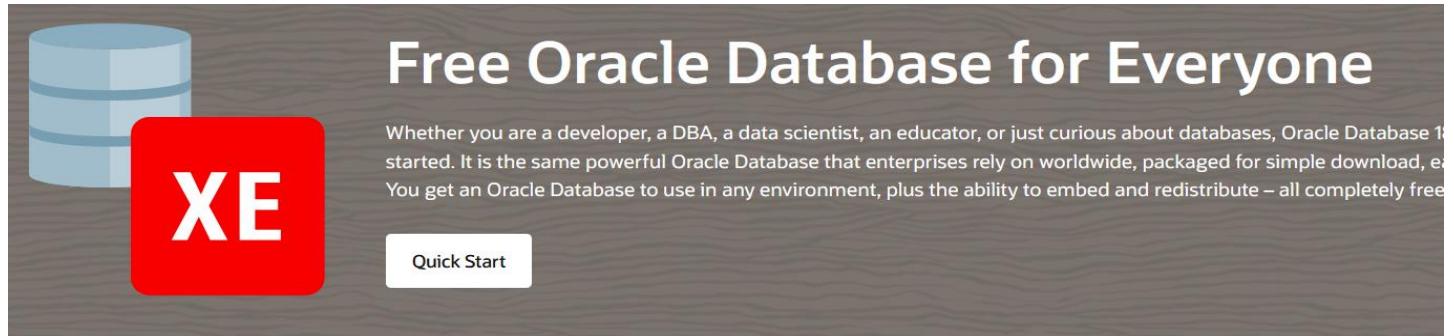
```
LISTENER =  
(ADDRESS_LIST=  
  (ADDRESS=(PROTOCOL=tcp)(HOST=orcl-03)(PORT=1521))  
  (ADDRESS=(PROTOCOL=ipc)(KEY=PNPKEY)))  
  
SID_LIST_LISTENER=  
(SID_LIST=  
  (SID_DESC=  
    (GLOBAL_DBNAME=ORCL)  
    (SID_NAME=ORCL)  
    (ORACLE_HOME=/opt/oracle/...../))  
  )  
  )  
)
```

sqlnet.ora

```
SQLNET.AUTHENTICATION_SERVICES= (NTS) 인증 서비스 방식  
NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT) DB접속 시 사용하는 문자열 조회 방식  
특정 Client IP만 접속 허용 등
```

Oracle XE 18C 개요

- **Multitenant**
- **In-Memory**
- **Partitioning**
- **Advanced Analytics**
- **Advanced Security**
- **Resources:**
 - ✓ Up to 12 GB of user data
 - ✓ Up to 2 GB of database RAM
 - ✓ Up to 2 CPU threads
 - ✓ Up to 3 Pluggable Databases



Oracle 18C XE를 RedHat Linux 에 설치하기

1. 18c XE Linux 버전을 다운로드 함

```
wget https://download.oracle.com/otn-pub/otn_software/db-express/oracle-database-xe-18c-1.0-1.x86_64.rpm
```

2. 오라클 설치 전 사전에 설치되어야 할 라이브러리 설치

2.1 먼저 preinstall 라이브러리를 다운로드

```
curl -o oracle-database-preinstall-18c-1.0-1.el7.x86_64.rpm https://yum.oracle.com/repo/OracleLinux/OL7/lates
```

2.2 다운로드 된 preinstall 라이브러리를 설치

```
yum -y localinstall oracle-database-preinstall-18c-1.0-1.el7.x86_64.rpm
```

3. 오라클 18C XE Binary 설치. Root 권한으로 수행

```
yum -y localinstall oracle-database-xe-18c-1.0-1.x86_64.rpm
```

4. Non Multi-Tenant 로 DB 생성. DB 생성 Script 수정 필요

/etc/init.d/oracle-xe-18c 파일을 열어서 CDB/PDB 관련 사항을 모두 수정

5. /etc/init.d/oracle-xe-18c configure 실행하여 DB 생성

Oracle 19C Enterprise Edition 설치 시 유의 사항

- 절대 회사 서버에는 설치해서는 안됨
- 부하 테스트를 위해서는 60GB의 SSD Block Storage가 필요한데 가격 부담이 있을 수 있음.

Oracle 19C Enterprise Edition 설치

1. 19c EE LINUX 설치 파일 rpm 버전을 개인 PC에 download 수행
2. 설치 서버에 root로 로그인 하여 preinstall package를 설치
3. 개인 PC에 있는 오라클 19C EE rpm 버전을 리눅스 서버로 올리고 yum으로 rpm 설치
4. \$ORACLE_HOME과 PATH, ORACLE_BASE를 oracle user의 .bashrc 에 설정하고 . .bashrc 수행
5. 수동으로 dbca command를 이용하여 non-cdb로 db생성.
6. Listener 수정(Optional)

추가 SSD Storage를 VM에 구성하기

- 60GB의 SSD Disk를 생성해서 VM에 할당
- SSD Disk를 Format
- 해당 Disk를 특정 영역으로 mount
- 부팅 시마다 mount할 수 있도록 /etc/fstab에 등록

오라클 DB 관리툴과 SwingBench 소개

대표적인 오라클 운영 툴

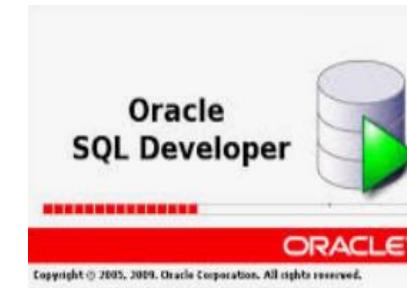
Toad for Oracle



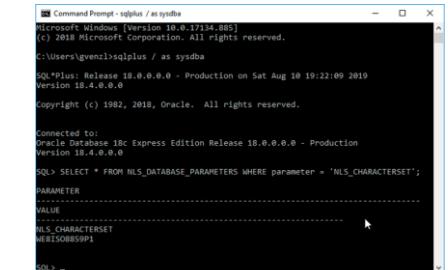
Orange for Oracle



Oracle SQL Developer



SQL*Plus



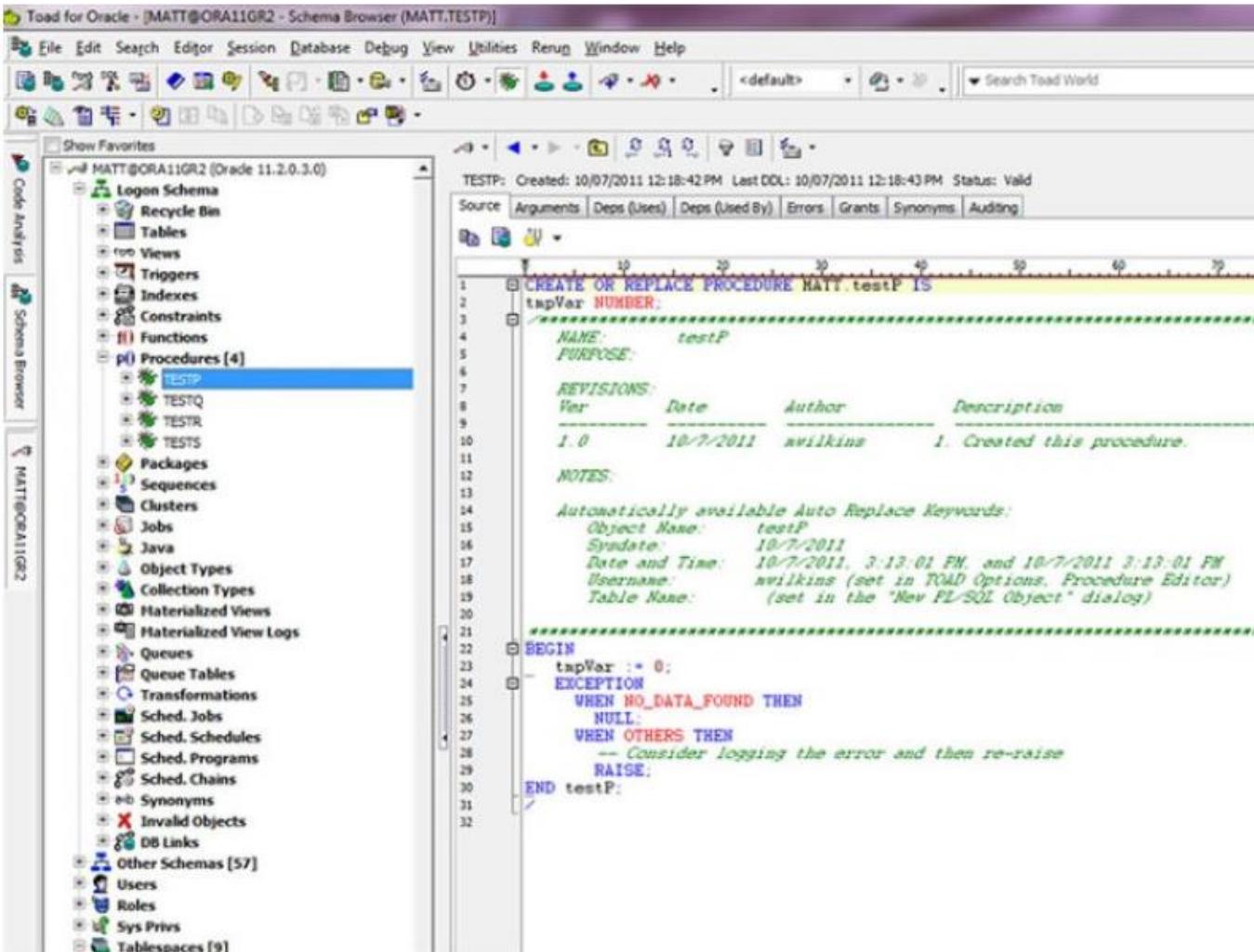
- Quest 사에서 제작
- 다양한 기능으로 전세계에서 가장 많이 사용되는 대표적인 클라이언트 툴
- 유료이며 별도의 오라클 클라이언트 설치 필요(OCI로 접속)

- 웨어밸리에서 제작
- Toad와 거의 유사한 기능 보유
- 국내에서 가장 많이 사용되는 클라이언트 툴
- 유료이며 별도의 오라클 클라이언트 설치 필요(OCI로 접속)

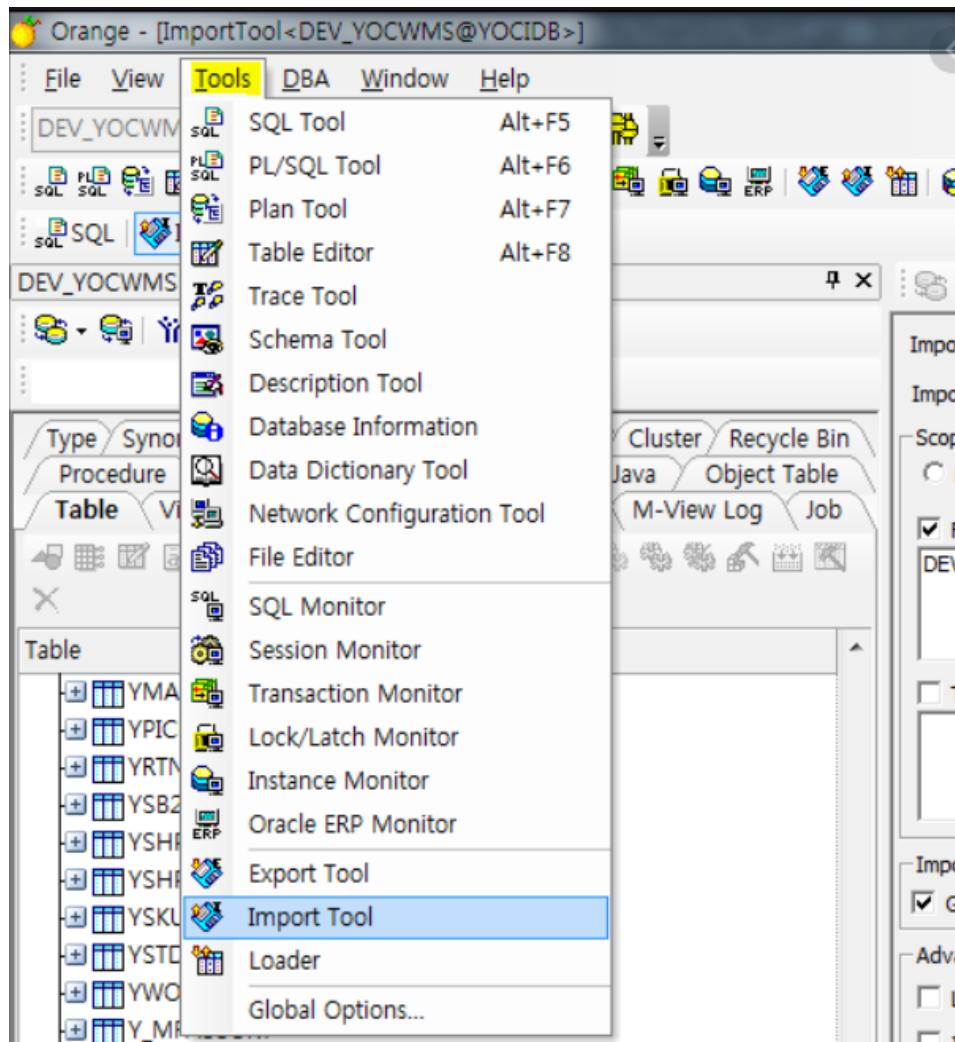
- 오라클에서 제작
- SQL과 PL/SQL 스크립팅에 뛰어난 기능 보유. 주요 모니터링 기능 있으나 Toad나 Orange에 비해 종류 및 직관성이 떨어짐.
- 무료이며 별도의 클라이언트 설치 필요 없음(JDBC thin driver로 접속)

- 콘솔상에서 구동
- 사용이 불편함. 하지만 운영을 위해서는 반드시 활용이 가능해야 함
- 웰스크립트와 결합하여 강력한 배치 프로그램 수행 가능
- 장애 시에는 오직 SQL*Plus만 가능

Toad 예시



Orange 예시



Oracle DB 부하 생성 - SwingBench

- Dominic Giles가 Java 기반으로 개발. 설치 시 JDK(또는 JRE)가 PC에 먼저 설치 되어 있어야 함.
- OLTP, DSS, Customized SQL 기반으로 다양하게 Oracle DB에 부하를 생성

SwingBench 2.6.0.1137 //34.64.156.113/orcl

File Help

Configuration Preferences Output Events

Transactions Jobs

User Details	Connection Pooling	Properties	Id	Class Name	Short Name	Load Ratio	Activate ?
Username	SOE_10G		Customer Registration	com.dom.benchmarking.swingbench.plsqltransactions.NewCusto...	NCR	15	<input checked="" type="checkbox"/>
Password	*****		Update Customer Details	com.dom.benchmarking.swingbench.plsqltransactions.UpdateCust...	UCD	10	<input checked="" type="checkbox"/>
Connect String	// /orcl		Browse Products	com.dom.benchmarking.swingbench.plsqltransactions.BrowseProd...	BP	90	<input checked="" type="checkbox"/>
Driver Type	Oracle jdbc Driver		Order Products	com.dom.benchmarking.swingbench.plsqltransactions.NewOrderP...	OP	40	<input checked="" type="checkbox"/>
			Process Orders	com.dom.benchmarking.swingbench.plsqltransactions.ProcessOrd...	PO	5	<input checked="" type="checkbox"/>
			Browse Orders	com.dom.benchmarking.swingbench.plsqltransactions.BrowseAnd...	BO	15	<input checked="" type="checkbox"/>
			Sales Rep Query	com.dom.benchmarking.swingbench.plsqltransactions.SalesRepsO...	SQ	2	<input checked="" type="checkbox"/>
			Warehouse Query	com.dom.benchmarking.swingbench.plsqltransactions.Warehouse...	WQ	1	<input checked="" type="checkbox"/>
			Warehouse Activity Query	com.dom.benchmarking.swingbench.plsqltransactions.Warehouse...	WA	1	<input type="checkbox"/>

Collect Database Statistics
 Take AWR Snapshot at Start and End

Admin Username:
Admin Password:

Load Environment Variables Distributed Controls

Number of Users:
Min. Inter Delay Between Transactions (ms):
Max. Inter Delay Between Transactions (ms):
Min. Intra Delay Within Transactions (ms):
Max. Intra Delay Within Transactions (ms):
Logon Delay (milliseconds):

Transactions Per Minute

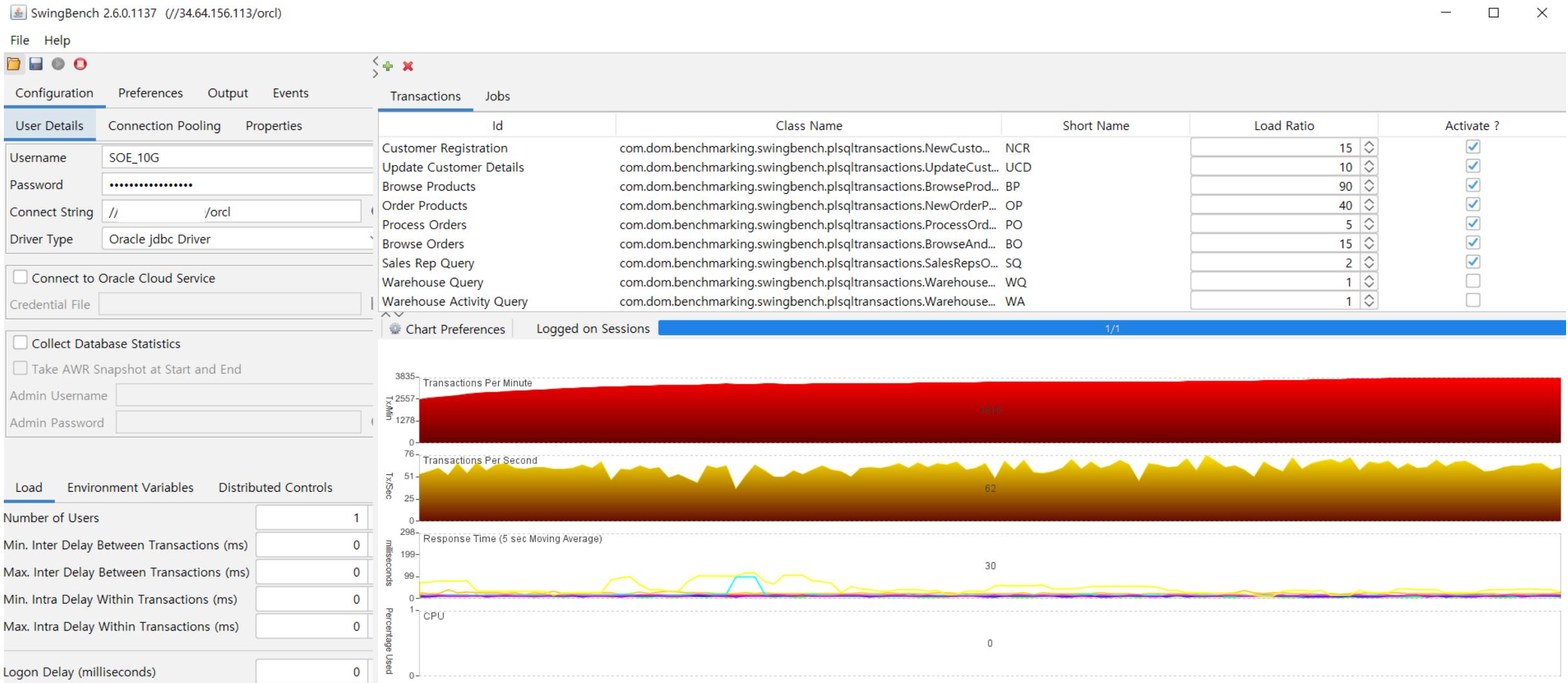
Transactions Per Second

Response Time (5 sec Moving Average)

CPU

Percentage Used

Logged on Sessions 1/1



SwingBench Load 유형

- Order Entry : OLTP
- Stored Procedure Stub: Customized SQL이나 Logic
- TPCDS : DSS

SwingBench 디렉토리 별 주요 구성 요소

- **sql**

- ✓ OrderEntry, Stored Procedure Stub, TPCDS 부하를 발생 시키기 위해서 생성해야만 하는 다양한 Object와 Schema의 DDL와 부하 실행을 위한 PL/SQL 스크립트를 가짐

- **winbin**

- ✓ Windows 환경에서 SwingBench 실행 및 데이터 생성 UI를 .bat로 가짐
- ✓ swingbench.bat: SwingBench 실행 UI
- ✓ oewizard.bat : Order Entry 데이터 생성
- ✓ tpcdswizard.bat : TPCDS 데이터 생성
- ✓ sqlbuilder: customized SQL 생성

- **source**

- ✓ Java Source 코드

로컬 디스크 (C:) > util > swingbenchlatest > swingbench		
이름	수정한 날짜	유형
bin	2020-11-09 오후 12:59	파일 폴더
configs	2020-11-09 오후 12:59	파일 폴더
launcher	2020-11-09 오후 12:59	파일 폴더
lib	2020-11-09 오후 12:59	파일 폴더
log	2020-11-09 오후 12:59	파일 폴더
source	2020-11-09 오후 12:59	파일 폴더
sql	2020-11-09 오후 12:59	파일 폴더
utils	2020-11-09 오후 12:59	파일 폴더
winbin	2020-12-03 오후 1:06	파일 폴더
wizardconfigs	2020-11-09 오후 12:59	파일 폴더
README	2019-01-07 오후 8:41	텍스트 문서

Order Entry 수행을 위한 데이터 생성

- oewizard.bat을 구동하여 생성
- 별도의 사용자/스키마 생성 필요: SOE_10G
- 추가 테이블 스페이스 필요(최소 40GB 이상, 미리 생성이 더 편리)
- Temporary Tablespace 증가 필요
- XE 일 경우에는 최소한으로 데이터 생성

오라클 아키텍처 개요와 DBMS 이해를 위한 필수 기반 지식

DBMS의 정의

DBMS란 Database Management System의 약자로, 데이터베이스를 구성하고, 이를 효율적으로 응용하기 위해 구성된 소프트웨어를 지칭함

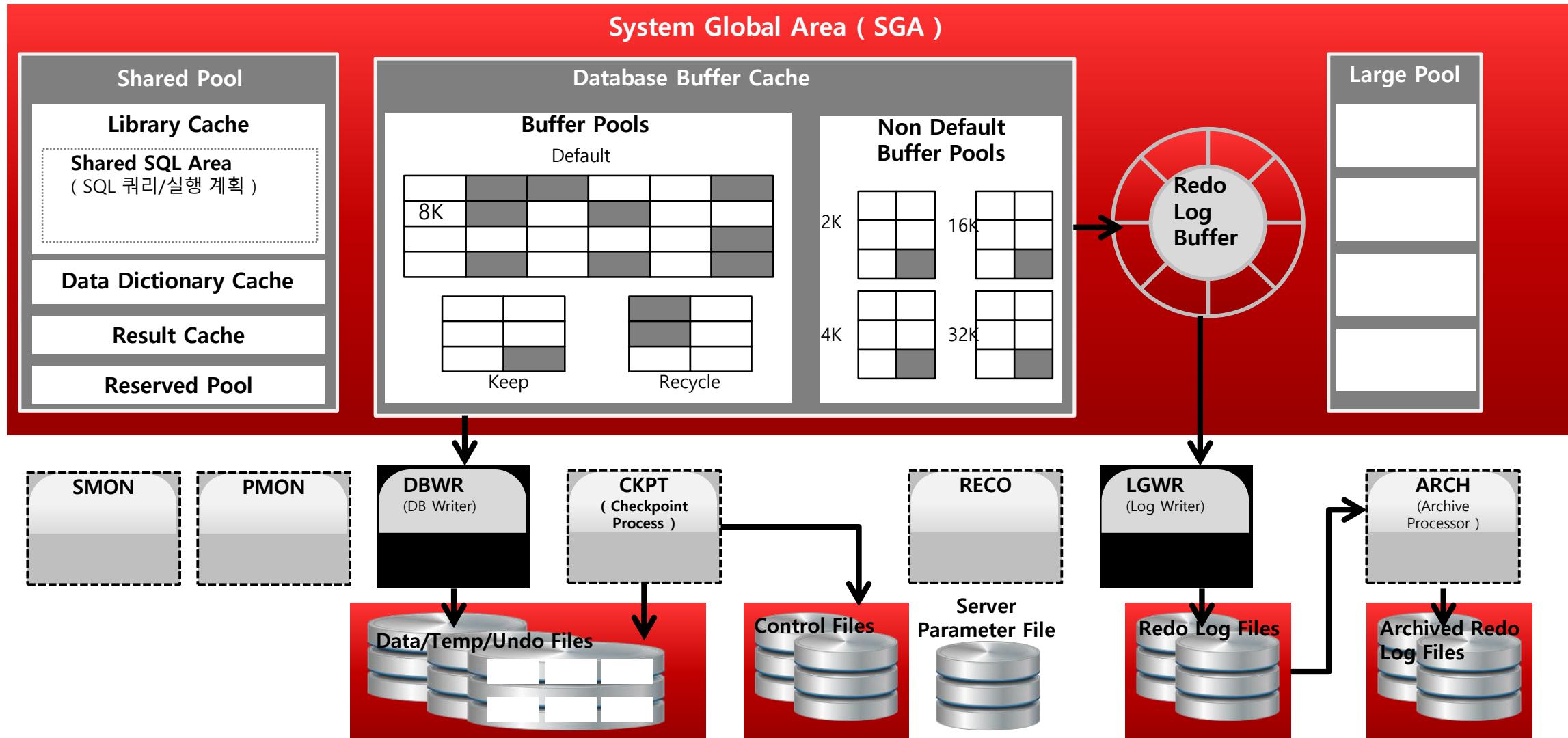
데이터와 응용 프로그램의 중간에서 응용 프로그램이 요구하는 대로 데이터를 정의하고, 읽고, 쓰고, 간신하는 등의 데이터를 조작하고 관리하는 프로그램의 집합체가 DBMS임



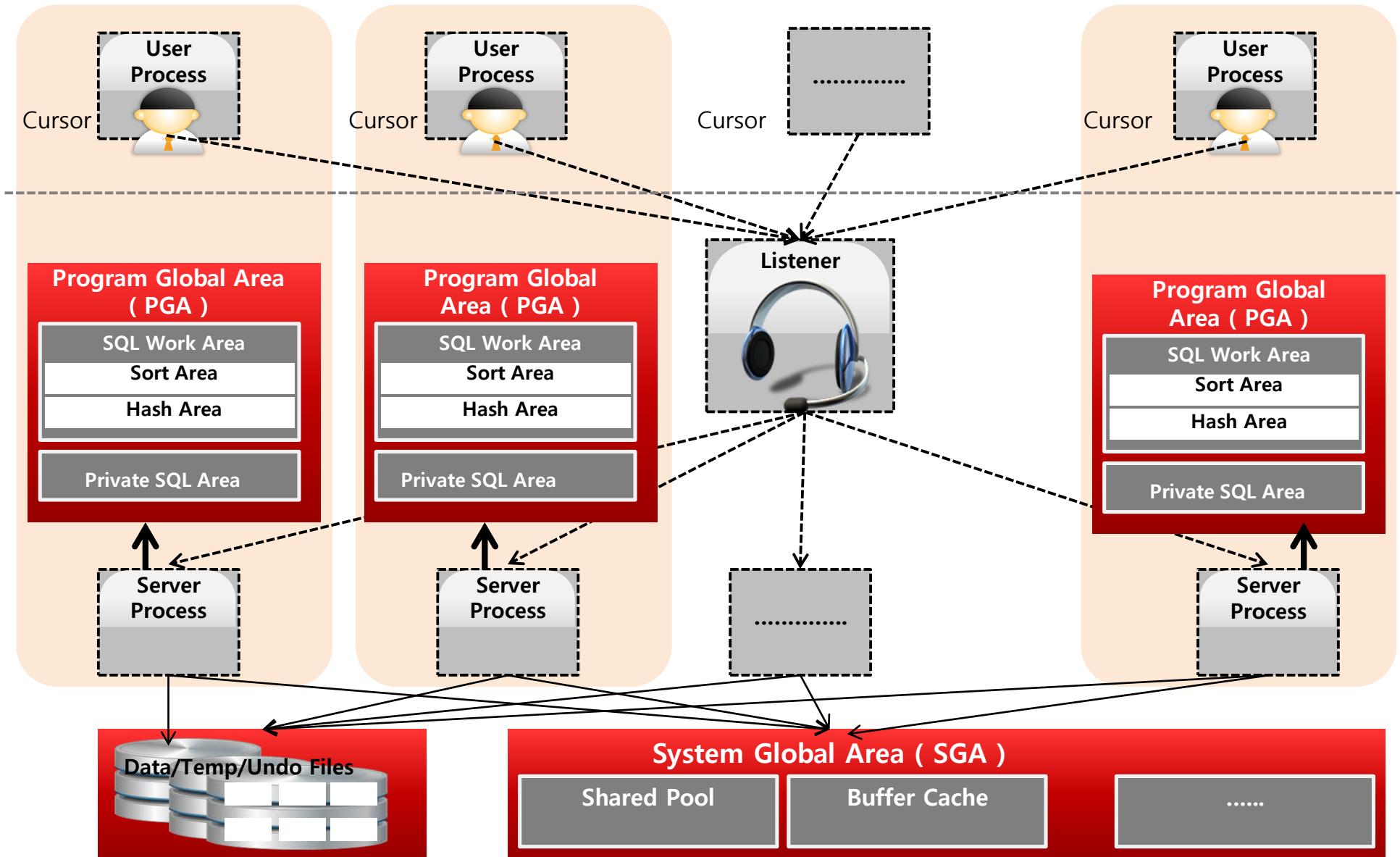
즉, 데이터를 효율적으로 **저장, 간신, 삭제, 읽기(검색)** 등을 할 수 있게 관리하는 소프트웨어가 바로 DBMS

오라클 데이터베이스 아키텍처 개요

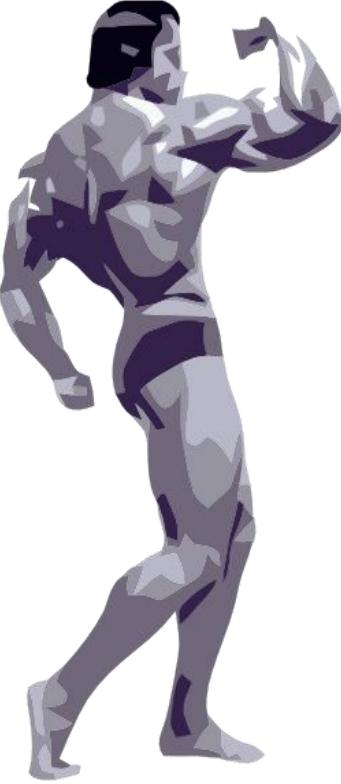
Oracle DBMS는 많은 사용자 프로그램이 사용하는 공유 메모리 영역인 SGA , Server Process 단독으로 사용하는 PGA 그리고 Datafile , Control file, redo Log file을 Background Process , Server Process 가 Access 하면서 데이터 처리를 수행함



Oracle Connection & Server Process



오라클 아키텍처 구성 요소



우리 몸을 건강하게 유지하는 3대 필수 영양소는

단백질 , 지방 , 탄수화물

오라클 RDBMS 이해를 위한 3대 필수 사항은

프로세스 , 메모리 , 데이터베이스 파일

오라클 아키텍처 구성 요소 설명



프로세스

- 사용자 프로세스와 Oracle 프로세스로 구분하며 Oracle 프로세스는 다시 Server Process 와 Background Process로 분류
- Server Process 와 Background Process 의 결합으로 대부분의 Database Work을 수행



메모리

- 많은 Server Process들이 동시에 사용하는 공유 메모리 구조인 SGA 와 Server Process 개별로 할당되는 PGA 영역으로 구분됨



데이터베이스 파일

- 오라클 DB를 구성하는 주요 파일로서 주로 사용자 데이터가 저장되는 Datafile 과 DML 과 같은 데이터의 변경사항을 실시간 기록하는 Redo Log File , 오라클 Structure 주요 변경사항을 기록하는 Control File로 구성

DBMS 성능에 가장 큰 영향을 미치는 것은 ?

1. CPU
2. Memory
3. I/O 

DBMS 성능에 가장 큰 영향을 미치는 것은 ?

1. CPU
2. Memory
3. I/O 

최적 저장 구조를 위한 고려 사항

데이터를 임의의 공간에 보관



VS

데이터를 약속된 공간에 정확히 보관



- 데이터를 검색 할 시 **시간이 더 걸림**
- 데이터를 저장 할 시 **빠르게 완료됨**

- 데이터를 검색 할 시 **빠르게 완료됨**
- 데이터를 저장 할 시 **시간이 더 걸림**

데이터의 저장 방식과 검색은 매우 밀접한 관계에 있음. 다양한 데이터 저장 방식을 활용하여 상황에 맞게 검색 속도를 향상 시킬 필요가 있음

인덱싱을 통한 검색과 Full Scan을 통한 검색

기말고사에 나올 페이지를 표기하기 위해 목차를 참고하여 해당 페이지를 표기하는 것이 빠른가 ? 책 페이지를 Scan 하면서 표기하는 것이 나은가 ?

목차를 이용한 페이지 표기

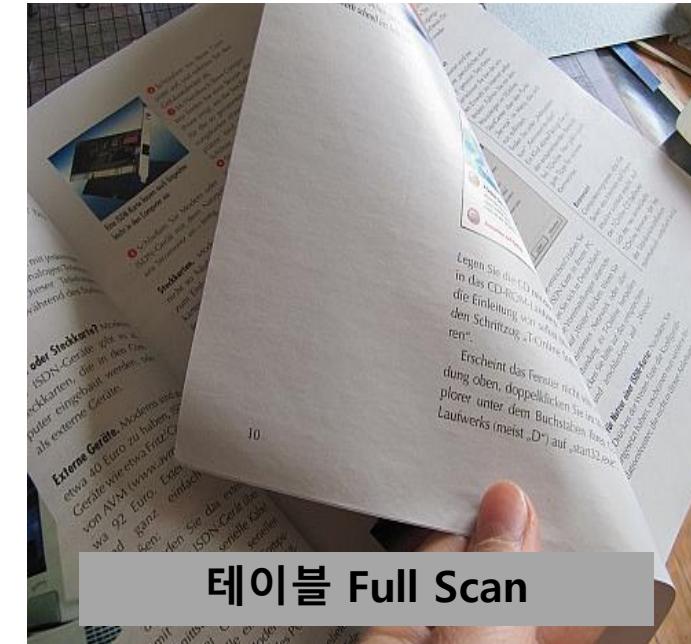
Contents

The Australian Science Archives Project	ix
The University of Melbourne Archives	xi
How to Use this Guide	xiii
Glossary of Archival Terms	xiv
Inventory, Series Descriptions and Detailed Listings	17
Biographical Note, John Stewart Turner	19
Related Records Creators / Sources	23
Records Summary	25
Series 01 Pre-retirement Subject Files	27
Series 02 Alphabetical Post-retirement Subject Files	179
Series 03 Personal Items - Biographical Manuscripts, Memorabilia and Ephemera	251
Series 04 Personal - Paintings, Photography and Manuscripts	265
Series 05 Reports and Related Publications by J. S. Turner	277
Series 06 Reprints and Publications	283
Series 07 Correspondence Filings, Unlisted	299
Series 08 Indexes to Files	321
Series 09 Australian Academy of Science Journals	323

인덱스를 통한 테이블 Access

VS

페이지 Scan을 통한 표기



테이블 Full Scan

명확한 목차 항목의 여부 , 표기해야 할 페이지 수 , 각 표기 페이지들의 인접성 등의 차이에 따라 어떠한 방법이 최적이 될지 결정됨

Database Block의 정의

DB Block

- DB 데이터 검색과 저장의 가장 기본 단위 (8K, 16K, 32K, 64K)
- 모든 DB I/O 는 DB Block 단위로 수행
- 보통 Block 당 평균 수십 개의 레코드가 들어갈 수 있는 크기로 구성
- **단 하나의 레코드를 읽을 지라도 최소한 1 Block은 Access 해야 함.**

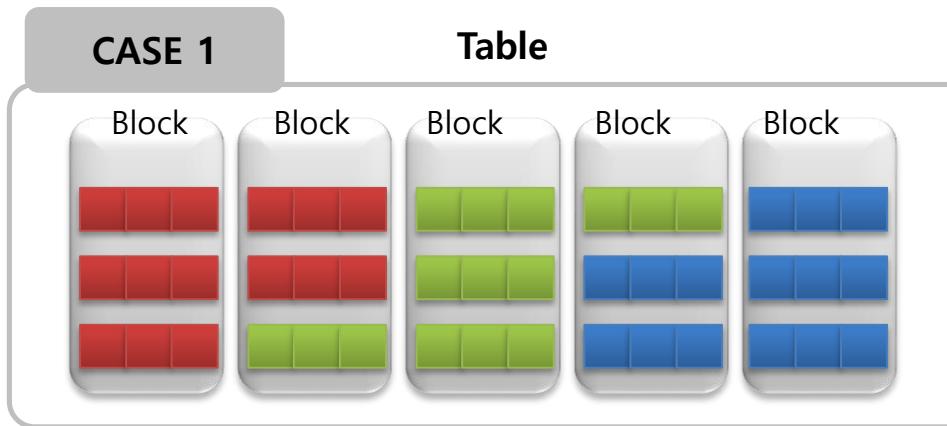


- 백만개의 Record를 가지는 두개의 테이블이 있다. 검색 속도는 서로 동일한가?
- 백만개의 Record 를 가지고 컬럼 크기와 값도 서로 동일한 두개의 테이블이 있다. 검색 속도는 서로 동일한가?

클러스터링 팩터

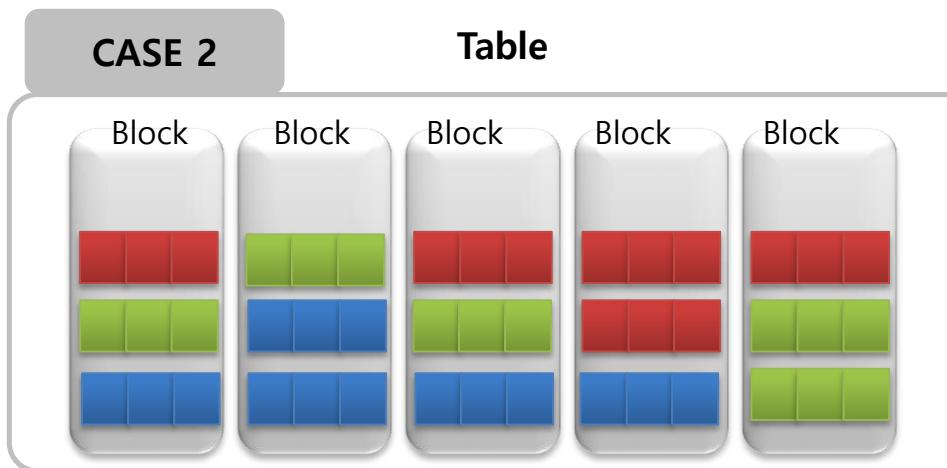
클러스터링
팩터

사용자가 자주 검색하는 **비슷한 값들이 얼마나 서로 모여 있느냐**에 따라서 액세스 하는 Block 건수가 달라짐. 즉 I/O 횟수가 차이가 나게 됨.



2 Block Access

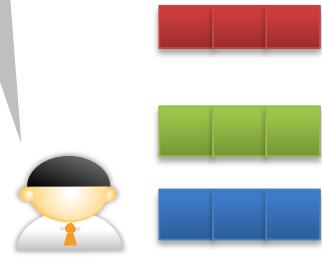
Select * from Tab
where Color = '■'



4 Block Access

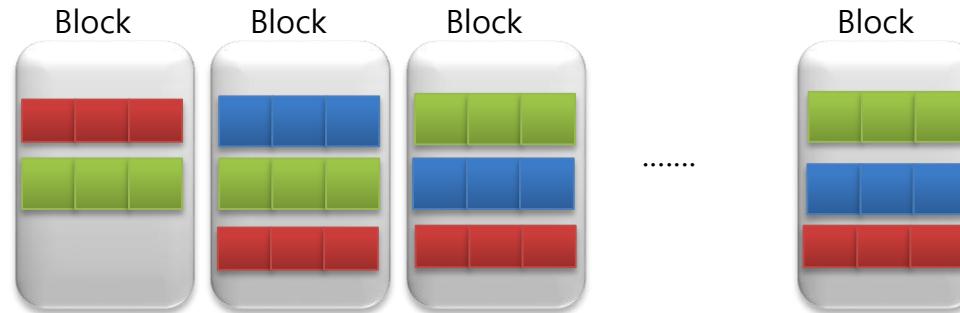
데이터 저장구조 최적화의 이슈

사용자가 입력하는 데이터의 순서는 사용자가 조회하고자 하는 데이터의 순서대로 입력되지 않음



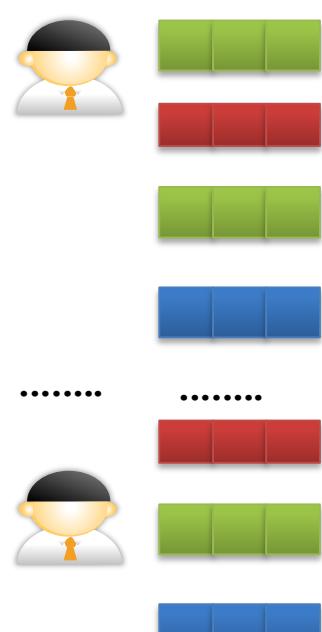
사용자가 입력하는 데이터를 주어진 블록의 여유공간에 순차적으로 위치

빠른 데이터 입력 속도, 상대적으로 나쁜 클러스터링 팩터



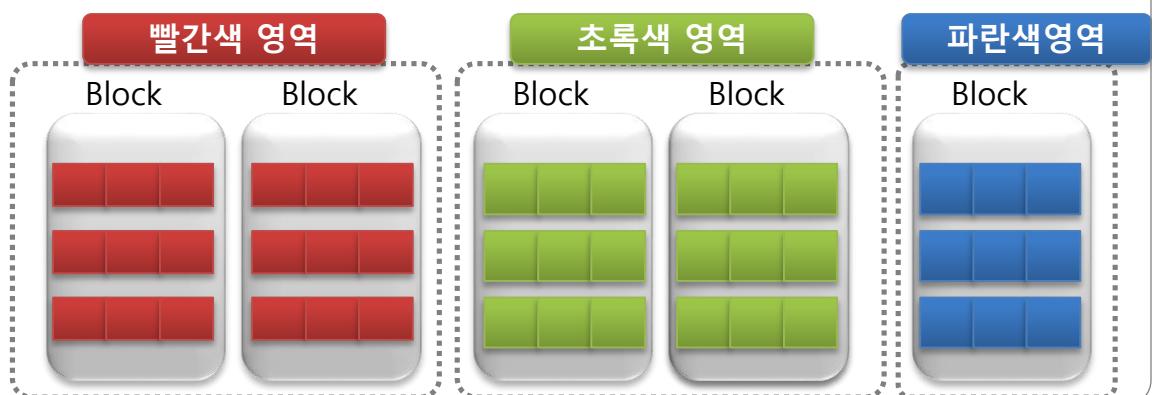
VS

상호간에 분명한 장단점이 존재함. 어떻게 최적화 시킬 것인가?



사용자가 입력하는 데이터를 그 값에 따라 지정된 블록 영역에만 입력

느린 데이터 입력 속도, 상대적으로 좋은 클러스터링 팩터



데이터 저장구조 최적화의 이슈

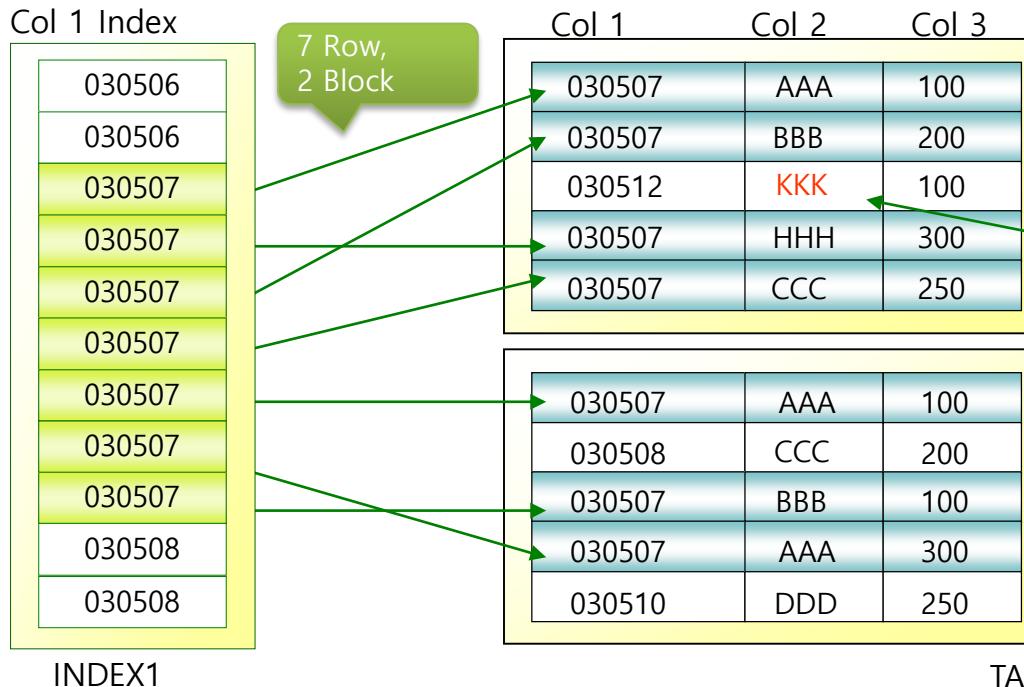
- 특정 조건의 검색 최적화를 위해 데이터를 저장하였지만 해당 조건이 아닌 다른 조건으로 검색할 경우에는 오히려 성능저하가 발생할 수 있음.
- 데이터를 바라보는 View는 단일 조건일 수 만은 없으며 다양한 조건 검색을 염두에 두고 데이터 저장 구조가 설계 되어야 함.



Index를 통한 테이블 Access의 클러스터링 팩터

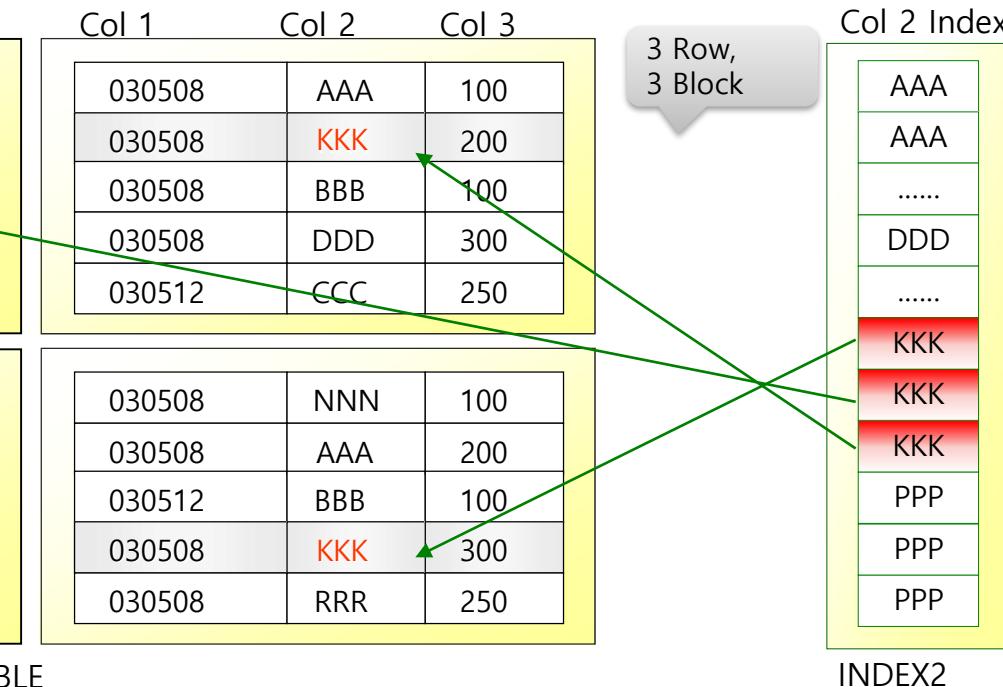
인덱스 로우의 순서와 테이블에 저장되어 있는 데이터 로우의 위치가 얼마나 비슷한 순서로 저장 되어 있는가?

SELECT * FROM TABLE WHERE COL1 = '030507'



클러스터링 팩터가 좋다

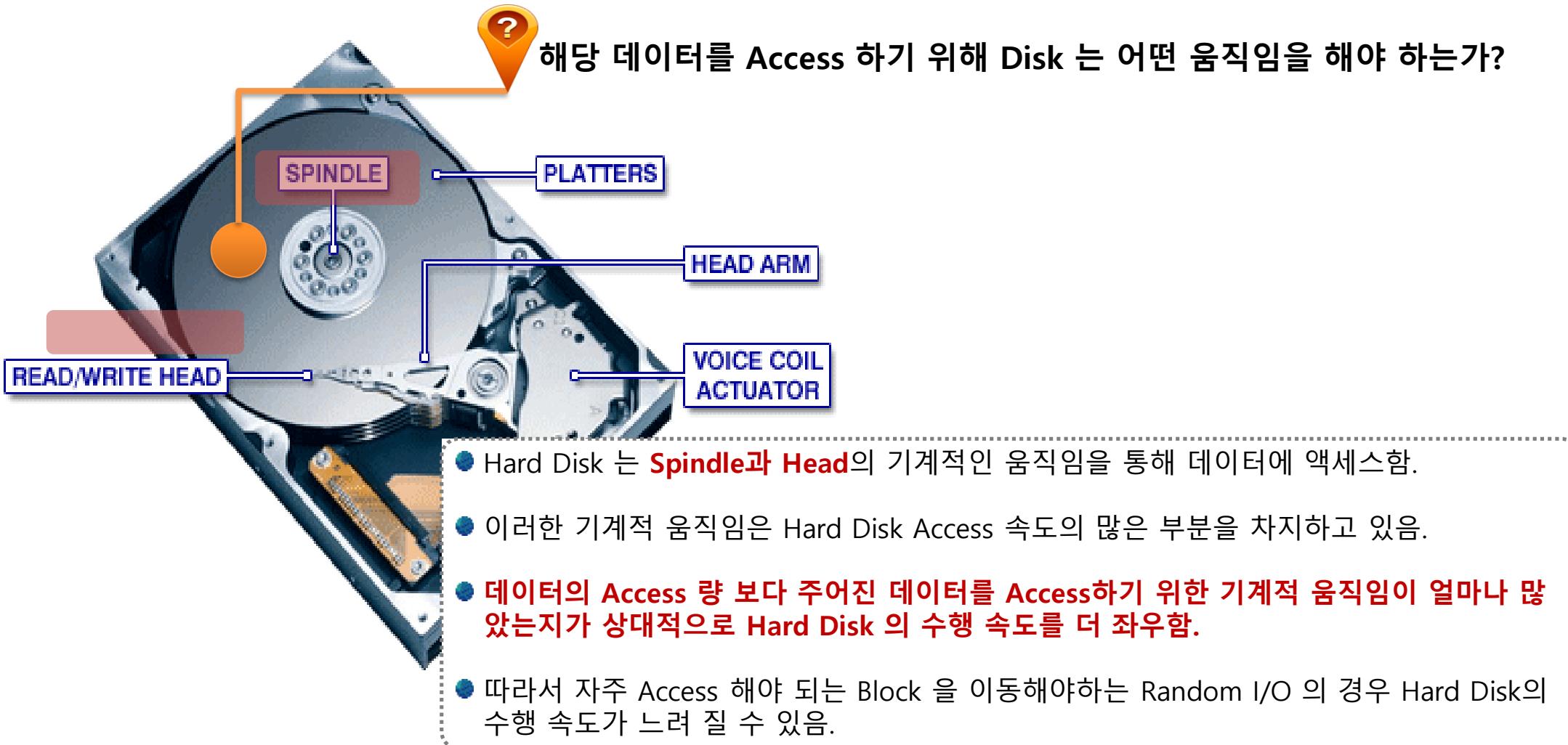
SELECT * FROM TABLE WHERE COL2 = 'KKK'



클러스터링 팩터가 나쁘다.

동일한 테이블이더라도 검색조건에 따라 클러스터링 팩터가 달라짐.

Hard Disk 의 데이터 Access 원리



I/O Access 유형

Sequential Access (순차 Scan)

- 보조기억장치에 저장된 파일로부터 정해진 순서대로 데이터를 순차적으로 검색함. 순차 액세스에서는 데이터들이 차례차례로 읽혀짐. 스캔 방식으로 동작함

Table Full Scan

Multi 블록을 I/O 단위로 사용 가능,
DB_FILE_MULTIBLOCK_READ_COUNT
로 설정

Random Access (임의 Access)

- 어떤 파일 내에 있는 특정한 레코드를 찾을 때 다른 레코드를 순차적으로 읽지 않고 원하는 레코드만을 직접 액세스. 데이터를 빨리 검색할 수 있는 액세스 방식
- 검색될 레코드를 명시하기 위해서는 임으로 액세스 하려는 레코드들은 그들과 관련된 key를 가져야 함

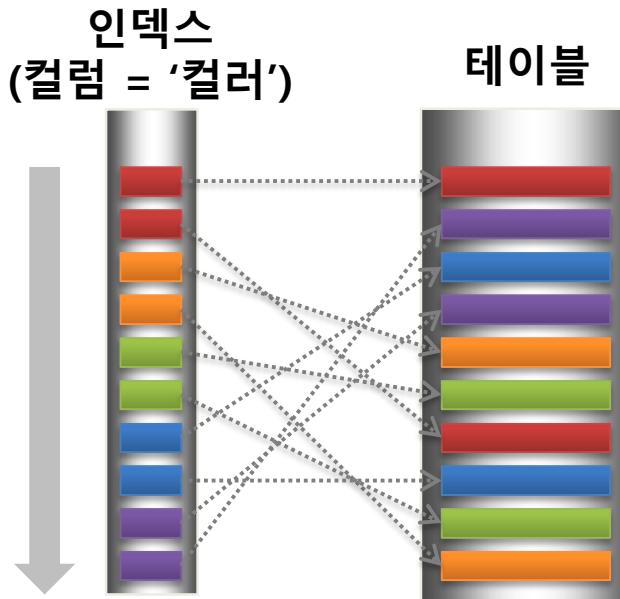
Index 를 경유한 Table Random Acess

1 블록이 I/O 단위

디스크 Access Time (Seek Time)

- 데이터를 읽어 들이거나 기록하기 위해 디스크 드라이브의 판독/기록 헤드의 위치를 표면 위의 특정 트랙으로 이동시키는데 소요되는 시간.

테이블 Access I/O 유형



인덱스를 경유한 Table Random Access

- 인덱스 스캔을 통해 알게된 Rowid 값을 가지고 Table을 Access함(Row id는 테이블의 위치를 알수 있는 고유 값)
- 인덱스를 통해 테이블을 Access할 경우 테이블 Block Access 순서가 Random 이 되는 Random I/O 발생
- 대부분의 OLTP 성 Application은 Random I/O Access 유형

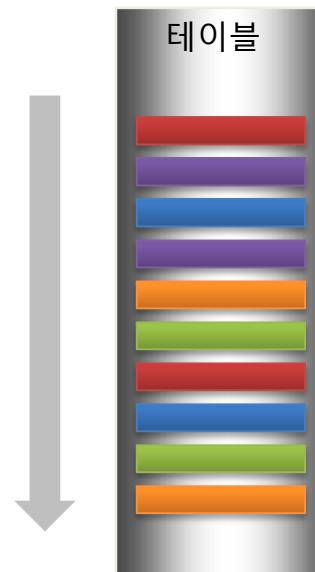


Table Full Scan

- 테이블은 인덱스를 경우하지 않을 경우 Segment의 처음 부터 끝까지 순차적으로 Full Scan 함
- 데이터를 읽는 량은 Random I/O 보다 많을 수 있으나 대용량의 데이터를 읽을 경우는 Random I/O 보다 훨씬 효율적

Random I/O 증가량에 따른 성능 변화

Random I/O , Full Scan 모두 Disk I/O 로 가정

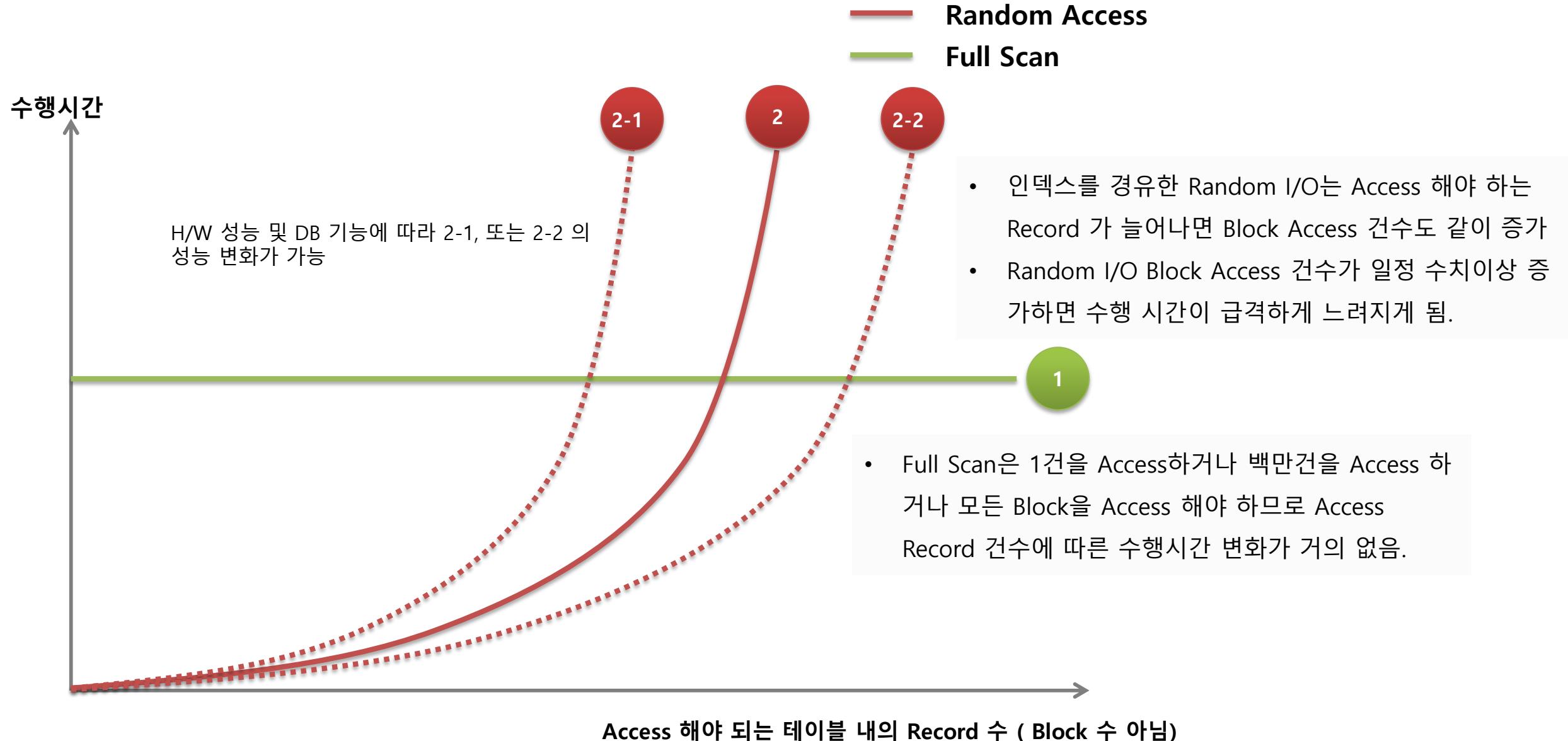
- 1만건의 Record를 Access 하기 위해서 **Random I/O 시 1,000 Block** 이 필요하고 **Full Scan 시 십만 Block** 필요할 경우 어떤 Access 가 더 빠른가 ?
- 백만건의 Record를 Access 하기 위해서 **Random I/O 시 십만 Block** 이 필요하고 **Full Scan 시 천만 Block** 필요할 경우 어떤 Access 가 더 빠른가 ?



Random I/O 는 그 절대량이 늘어날 수록 H/W 적인 제약으로 인하여 수행 성능이 급격하게 저하됨

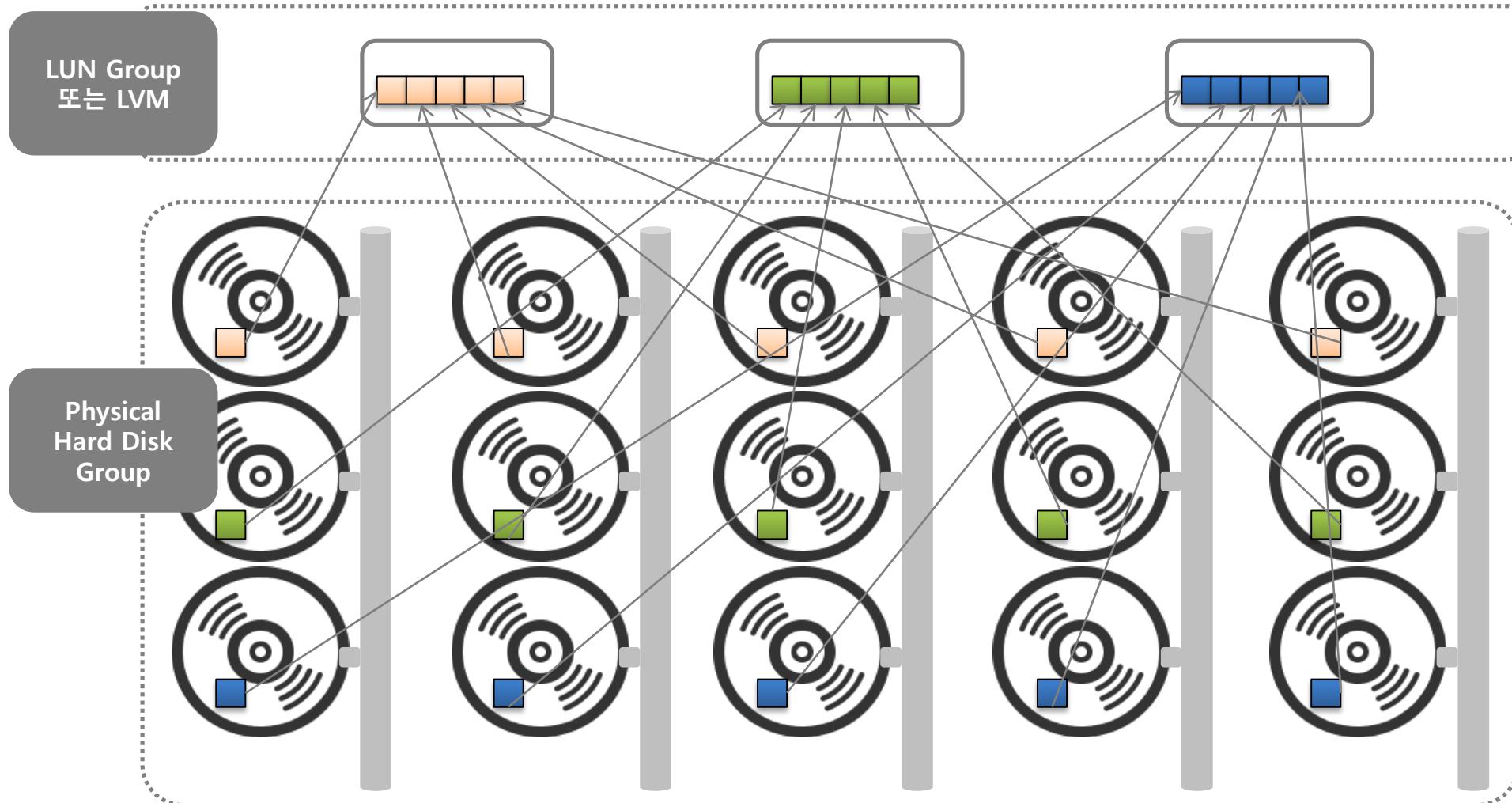
- 특정 TABLE의 컬럼값 분포도가 균일하게 10% 일 경우 , 가령 컬럼값 A,B,C,D,E,F,G,H,I,J 열 종류가 모두 균일하게 건수가 같다고 가정하자
- 1억건이 넘는 대용량 테이블에 해당 컬럼으로 단독 인덱스를 생성할 것인가 ?
- 만일 동일 테이블에 해당 컬럼값 분포도가 균일하게 1%라면 , 가령 컬럼값이 100종류이면 모두 균일하게 건수가 같다면 해당 컬럼으로 단독 인덱스를 생성할 것인가?

Table Full Scan 과 Random I/O 비교



Disk Storage의 Stripping

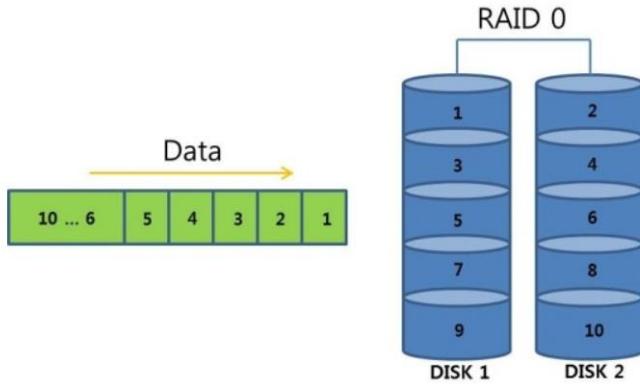
Storage 에 RAID 0 Stripping을 한 경우 I/O 가 균일하게 모든 Disk Drive에 동시에 수행되어 빠른 시간안에 I/O 처리



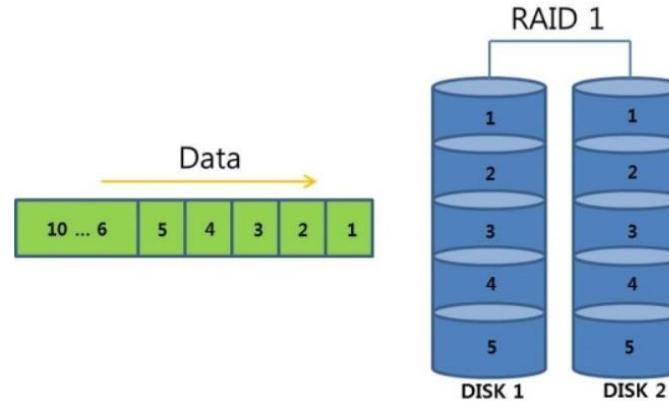
RAID 예시

<https://m.blog.naver.com/leekh8412/100175594400>

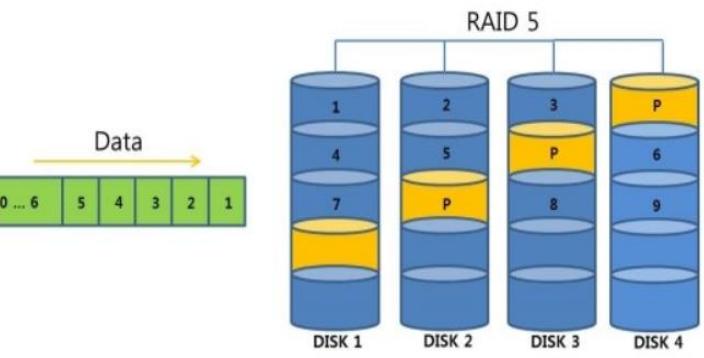
RAID 0



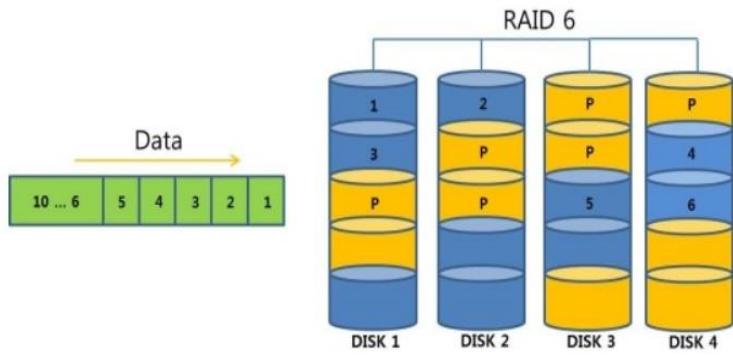
RAID 1



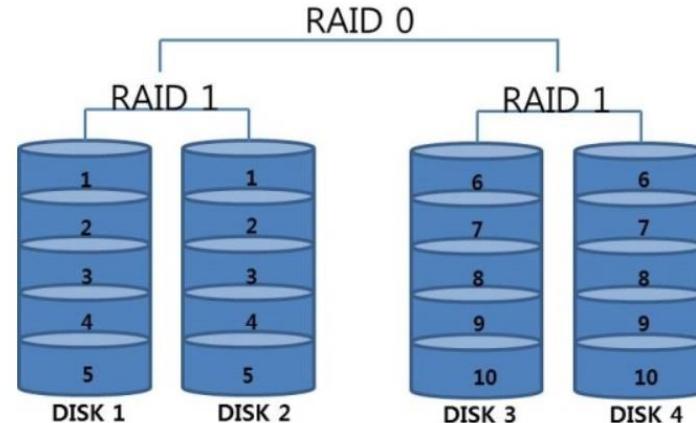
RAID 5



RAID 6

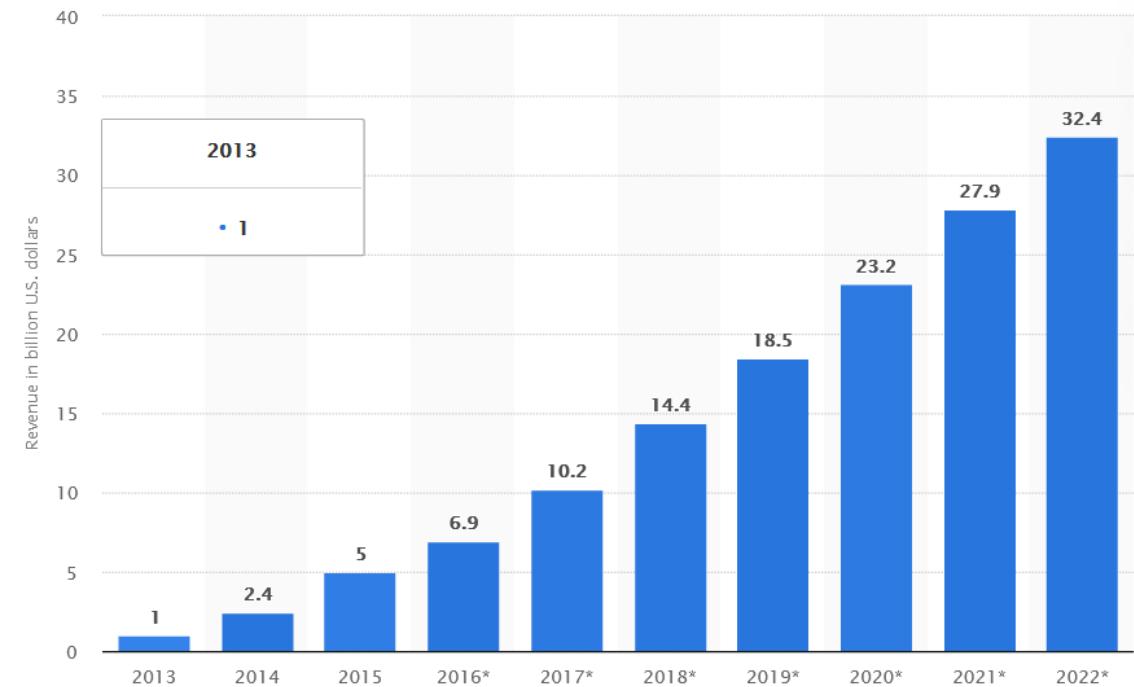


RAID 1+0



Flash Storage의 폭발적인 성장

SSD 기반 Flash Storage는 물리적인 디스크의 움직임없이 전기 신호를 기반으로 하므로 Random I/O를 기준 HDD 기반 몇배 이상 향상
Flash Storage의 저장 용량이 증가하고, 가격이 하락하면서 기존 HDD 기반 Storage 시장을 급속하게 대체함



2022년까지 전 세계 Flash Storage 업계의 예상 성장

OLTP 와 Batch 그리고 I/O Latency와 Throughput

OLTP
Real time
processing



OLTP(On-Line Transaction Processing)는 **실시간으로** DB 데이터를 조회/갱신하는 시스템

- 대부분의 업무 시스템(금융/통신/ERP등 조회업무, 쇼핑몰 주문, 콘서트 예매 등)
- 건당 매우 빠른 수행 시간이 생명(보통 0.01 ~ 1초 이하)
- 빠른 수행 시간을 위해 I/O Latency가 우선시 됨

Batch
Processing



Batch Processing은 작업을 몰아 두었다가 **한번에** 처리하는 시스템

- 주로 야간, 주말, 월말 등 정산/마감/통계 생성 등 대량의 데이터를 일괄처리하는 프로세스
- 대량의 데이터를 처리 성능을 위해서 I/O Throughput이 우선시 됨

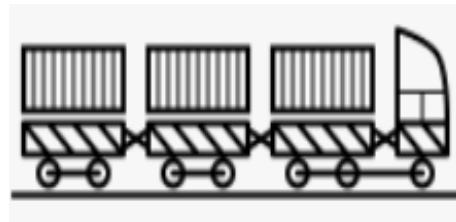
I/O Latency와 Throughput 그리고 IOPS



Latency

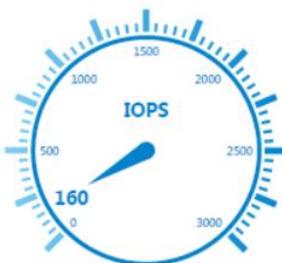
시스템에서 I/O가 요청 되었을 때 이를 수행하는 데 걸리는 시간. 보통은 I/O 시 Delay 시간을 의미

Disk 기반 스토리지에서는 보통 milliseconds 단위로 측정.



Throughput

정해진 시간 동안(보통은 1초) 얼마나 많은 데이터를 처리할 수 있는지를 나타내는 지표.



IOPS (I/O Per Seconds)

초당 I/O 횟수. 보통 IOPS는 Sequential, Random으로 분리하여 측정

- 일반적으로 IOPS가 높을 수록 Latency가 빠름.
- Throughput 은 IOPS * I/O Size

SQL 의 이해

SQL = Program ?

사용자 SQL은 Oracle 내부에서 상세 실행 계획으로 변경



PLAN_TABLE_OUTPUT											
Plan hash value: 3841029040											
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	IN-OUT	PQ Distrib	
0	SELECT STATEMENT										
1	PX COORDINATOR										
2	PX SEND QC (ORDER)	:TQ10003	24	3014	35 (0)	00:00:43			P->S	QC (ORDER)	
3	VIEW								PCWP		
4	SORT GROUP BY								PCWP		
5	PX RECEIVE								PCWP		
6	PX SEND RANGE	:TQ10002	24	3014	35 (0)	00:00:01			PCWP		
*	HASH JOIN		24	3014	11 (0)	00:00:01			PCWP		
7	PX RECEIVE		24	3014	9 (0)	00:00:01			PCWP		
9	PX SEND BROADCAST	:TQ10001	480	64K	8 (0)	00:00:01			P->P		
10	PX SEND BROADCAST		480	64K	8 (0)	00:00:01			PCWP		
11	PX BLOCK ITERATOR		480	64K	8 (0)	00:00:01			PCWP		
12	PX BLOCK ITERATOR		480	64K	8 (0)	00:00:01			PCWP		
*	13	TABLE ACCESS FULL	EMP	480	64K	8 (0)	00:00:21		PCWP		
14	PX BLOCK ITERATOR		1	248	4 (0)	00:00:01	1		PCWP		
*	15	TABLE ACCESS FULL	DEPT	8	248	4 (0)	00:00:11	8	14	PCWP	
16	PX PARTITION RANGE SINGLE		18	960	18 (0)	00:00:01	4		PCWP		
17	PX PARTITION HASH JOIN-FILTER		18	960	18 (0)	00:00:01	:BF0000	:BF0000	PCWP		
*	18	TABLE ACCESS FULL	SALGRADE	18	960	18 (0)	00:00:18	KEY	KEY	PCWP	

*Note: Query/Explain Plan adjusted per NDA

SQL 활용 능력의 중요성

SQL

SQL은 DB에게 일을 시키는 주체



Smart하게 일을 시키자

최소 Block Access
또는
최소 시간으로 결과 도출

- 오라클 아키텍처의 이해
- 인덱스 활용, 인덱스 적용 전략
- 조인의 원리, 다양한 데이터 연결 원리 체득
- 부분 범위 처리
- 집합 기반의 SQL 처리
- 대용량 데이터 처리 방안(파티셔닝 및 병렬 처리)

오라클 DB 기초 이해

데이터 딕셔너리(Data Dictionary)

<u>DB Creation</u>	<u>TBS Management</u>	<u>Storage Parameters</u>	<u>Rollback Segments</u>	<u>Undo Management</u>
V\$SGA	DBA_TABLESPACES	DBA_SEGMENTS	DBA_SEGMENTS	DBA_UNDO_EXTENTS
V\$INSTANCE	DBA_TABLESPACE_GROUPS	DBA_EXTENTS	USER_SEGMENTS	DBA_SEGMENTS
V\$DATABASE	DBA_DATA_FILES	DBA_TABLES	DBA_ROLLBACK_SEGS	USER_SEGMENTS
V\$PROCESS	DBA_FREE_SPACE	DBA_INDEXES	V\$ROLLSTAT	V\$UNDOSTAT
V\$SYSAUX_OCCUPANTS	V\$TABLESPACE	DBA_TABLESPACES	V\$ROLLNAME	U\$TRANSACTION
<u>Users & Resources</u>				
DBA_USERS	DATABASE_PROPERTIES	DBA_DATA_FILES	Data PUMP	Tuning
ALL_USERS	Roles & Privileges	DBA_FREE_SPACE	DBA_DATAPUMP_JOBS	V\$PX_PROCESS
USER_USERS	ALL_COL_PRIVS	Auditing	USER_DATAPUMP_JOBS	V\$PX_SESSION
DBA_TS_QUOTAS	USER_COL_PRIVS	STMT_AUDIT_OPTION_MAP	DBA_DIRECTORIES	V\$PX_PROCESS_SYSSTAT
USER_TS_QUOTAS	ALL_TAB_PRIVS	AUDIT_ACTIONS	<u>Dispatchers</u>	
USER_PASSWORD_LIMITS	USER_TAB_PRIVS	ALL_DEF_AUDIT_OPTS	V\$DISPATCHER_CONFIG	
USER_RESOURCE_LIMITS	ALL_TAB_PRIVS_MADE	DBA_STMT_AUDIT_OPTS	V\$MTS	
DBA_PROFILES	USER_TAB_PRIVS_MADE	USER_OBJ_AUDIT_OPTS	V\$DISPATCHER	
RESOURCE_COST	ALL_TAB_PRIVS_REC'D	DBA_OBJ_AUDIT_OPTS	Redo Log Files	
V\$SESSION	USER_TAB_PRIVS_REC'D	USER_AUDIT_TRAIL	V\$LOG	
V\$SESSTAT	DBA_ROLES	DBA_AUDIT_TRAIL	VSLOGFILE	
V\$STATNAME	DBA_COL_PRIVS	USER_AUDIT_SESSION	V\$LOG_HISTORY	
<u>RMAN Recovery Catalog</u>				
RC_ARCHIVED_LOG	USER_ROLE_PRIVS	DBA_AUDIT_STATEMENT	V\$RECOVERY_LOG	
V\$ARCHIVED_LOG	DBA_ROLE_PRIVS	USER_AUDIT_OBJECT	V\$ARCHIVED_LOG	
RC_BACKUP_CONTROLFILE	USER_SYS_PRIVS	DBA_AUDIT_OBJECT	<u>Archived Redo Log Files</u>	
V\$BACKUP_DATAFILE	DBA_SYS_PRIVS	DBA_AUDIT_EXISTS	V\$ARCHIVED_LOG	
RC_BACKUP_DATAFILE	COLUMN_PRIVILEGES	USER_AUDIT_SESSIONS	V\$ARCHIVE_DEST	
V\$BACKUP_DATAFILE	DBA_TAB_PRIVS	DBA_AUDIT_SESSION	V\$ARCHIVE PROCESSES	
RC_BACKUP_PIECE	ROLE_ROLE_PRIVS	USER_TAB_AUDIT_OPTS	<u>Security</u>	
V\$BACKUP_PIECE	ROLE_SYS_PRIVS	Security	DBA_USERS	
RC_BACKUP_REDOLOG	SESSION_PRIVS		DBA_USERS_WITH_DEFPWD	
V\$BACKUP_REDOLOG	SESSION_ROLES			
<u>Control Files</u>				
RC_BACKUP_SET	VSCONTROLFILE			
V\$BACKUP_SET	VSCONTROLFILE_RECORD_SECTION			
RC_DATABASE				
V\$DATABASE				
RC_DATAFILE				
V\$DATAFILE				
RC_RMAN_CONFIGURATION				
V\$RMAN_CONFIGURATION				
RC_LOG_HISTORY				
V\$LOG_HISTORY				

- 오라클 DB를 구성하는 모든 요소에 대한 정보를 가짐.
- (Object, Schema, Segment, SGA, Parameter, Log, Data File, Backup, Recovery 등)
- 다양한 성능 정보와 Wait 정보 제공
- DB Admin, Monitoring, Tuning을 위한 가장 중요한 정보

오라클 Data Dictionary 유형

- **USER_XXX Views**
- **All_XXX Views**
- **DBA_XXX Views**
- **V\$ XXX Views**
- **X\$ XXX Tables**

특정 사용자/전체 사용자/DBA 권한
사용자에게만 보여지는 Object 정보

DB 전반에 걸쳐 여러 요소들의 정보 제공

DB Internal 핵심 정보를 가지며 많은
Data Dictionary View들의 백 정보를
제공

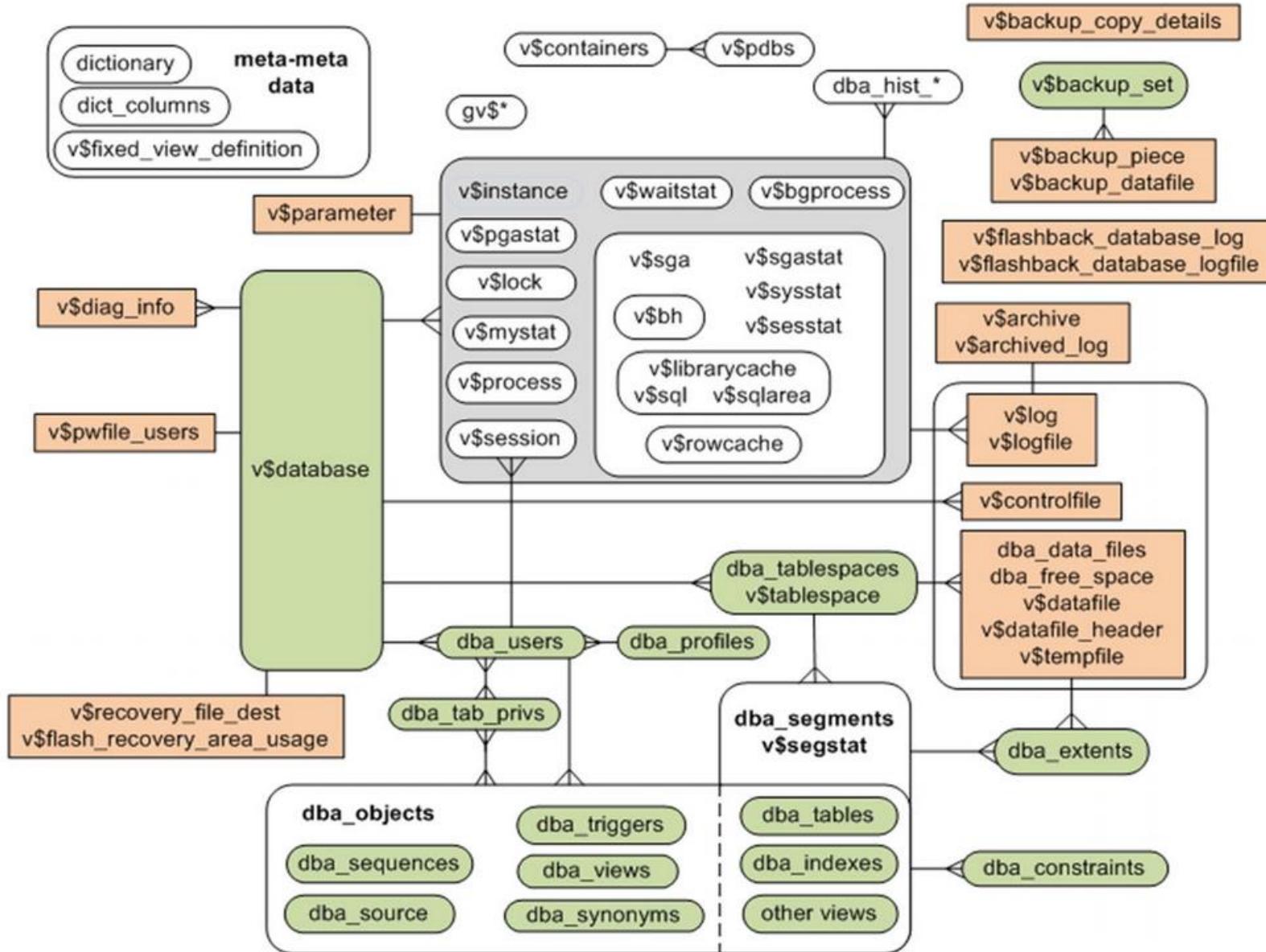
사용자는 Data Dictionary에 오직 Select 만 가능

오라클 Data Dictionary의 시대별 변화

오라클 DB의 다양한 기능이 추가되면서 Data Dictionary 개수도 지속적으로 증가

Version	V\$ Views	X\$ Tables
6	23	(35?)
7.1	72 (+213% vs. V6)	126
8.0	132	200
8.1	185 (+157% vs. V7)	271 (+115% vs. V7)
9.0	227	352
9.2	259 (+29% vs. 8i)	394 (+45% vs. 8i)
10.1	340	543
10.2	372 (+44% vs. 9i)	613 (+56% vs. 9.2)
11.1	484	798
11.2	525 (+41% vs. 10.2)	945 (+54% vs. 10.2)
12.1	606	1062
12.2	746 (+42% vs. 11.2)	1312 (+39% vs. 11.2)

오라클 주요 Data Dictionary



V_\$ View와 V\$ Synonym 생성

V\$ Synonym

GV\$ Synonym

V_\$ Views

GV_\$ View

- V\$ View들을 사실은 Synonym이며 V_\$ View를 가리킴
- V_\$ View는 GV_\$ View로 부터 만들어짐. GV_\$ View는 1개의 DB의 2개 이상의 Instance를 가지는 RAC구조 Data dictionary(Global View)의 View 정보임. RAC가 아닌 경우는 Instance가 1개 (instance_id=1)인 경우이므로 GV_\$ View에서 해당 instance_id의 값만 View 재 생성

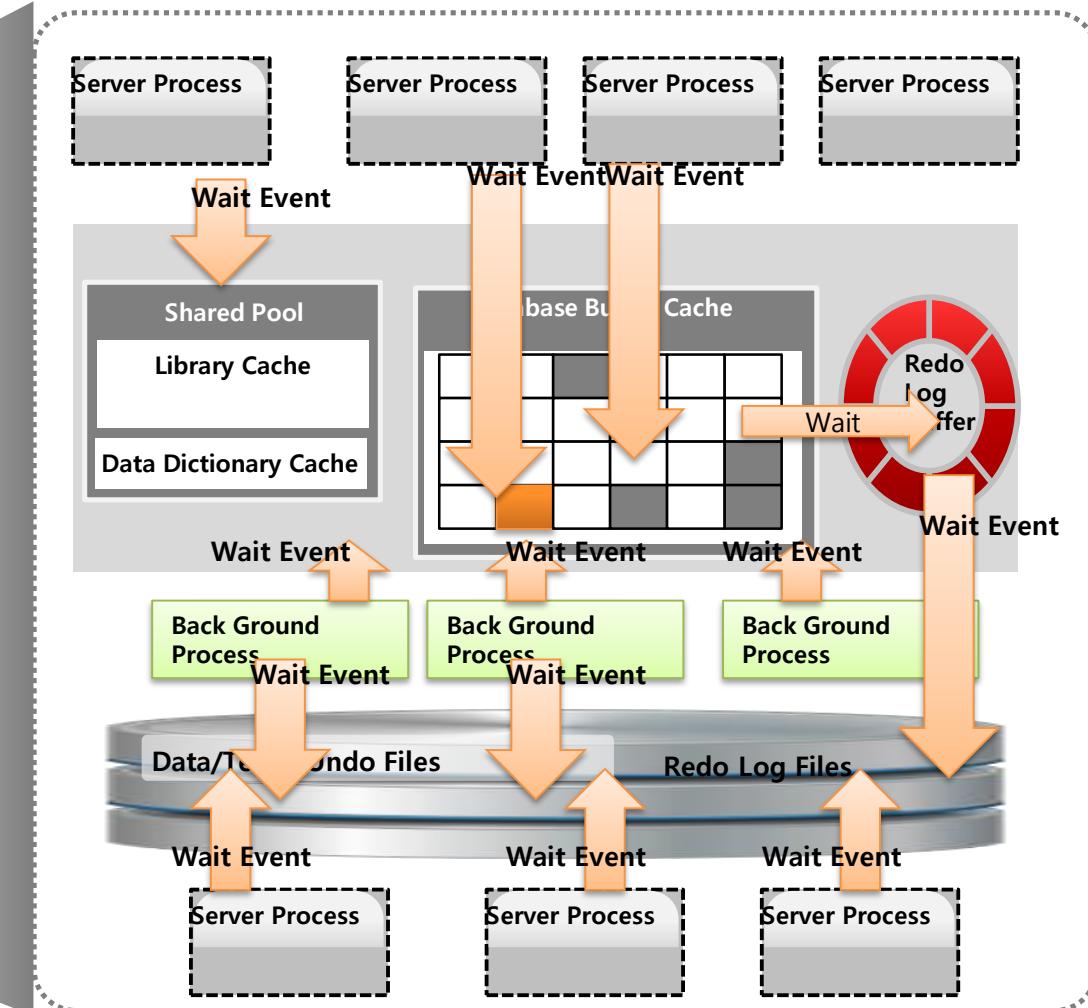
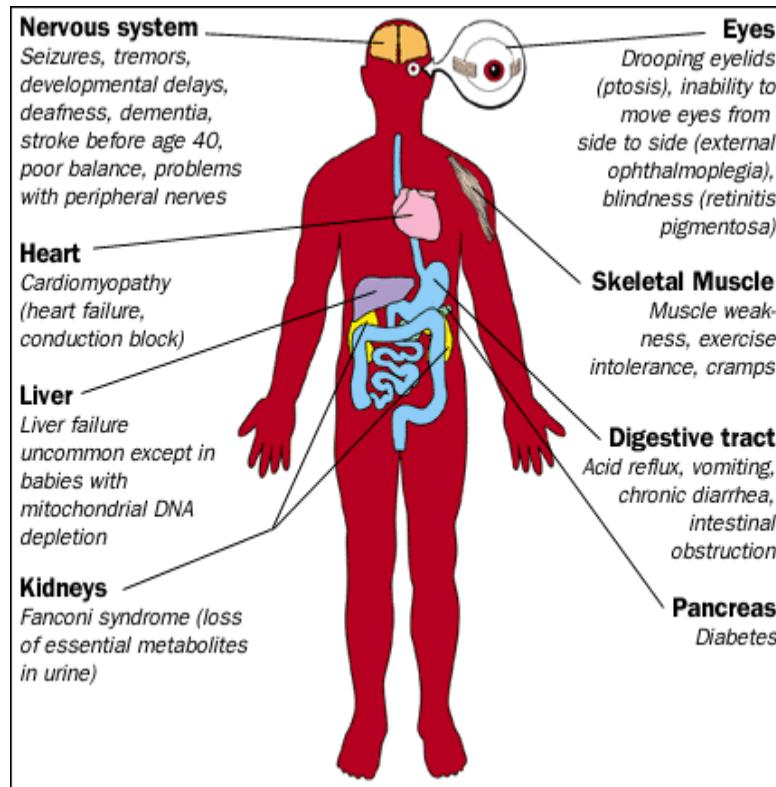
Oracle Wait Event

Wait Event

- 성능 및 운영 관리를 위해 알아야 할 가장 중요한 요소 중 하나
- 오라클은 서버프로세스 또는 백그라운드 프로세스가 DB의 CPU , Memory , Database File, I/O 등의 모든 자원을 액세스하고 이용하는 세부 단계별로 상세한 Wait Event (대기 이벤트) 를 발생
- 오라클의 모든 성능 요소 및 자원 사용현황은 Wait Event 의 발생정도에 따라 모니터링 가능함

DB 내 문제점 진단과 분석을 위한 Wait Event

인체내의 각 기관별로 진단을 위한 다양한 증상 및 이에 대한 해결책을 발전시켜 왔듯이, 오라클은 DB내의 거의 모든 세부 프로세스 절차에서 **Wait Event**를 의무적으로 발생시키는 성능 분석 Framework 을 발전시켜 옴.



Wait Event는 오라클의 DB 성능 개선 노력의 산물

성능, 문제점 들에 대한 완벽한 진단 및 개선을 위한 오라클의 철저한 노력의 산물



오라클의 Wait Event 종류는 모두 **1100** 가지가 넘음.



- 오라클 내부에서 처리되는 거의 세부 프로세스들은 각 단계별로 고유의 Wait Event 를 생성
- 이는 Oracle이 성능 및 시스템 문제 분석과 진단, 개선을 위해 얼마나 많은 공을 들이고 있는지를 알 수 있는 반증.

주요 Wait Event들

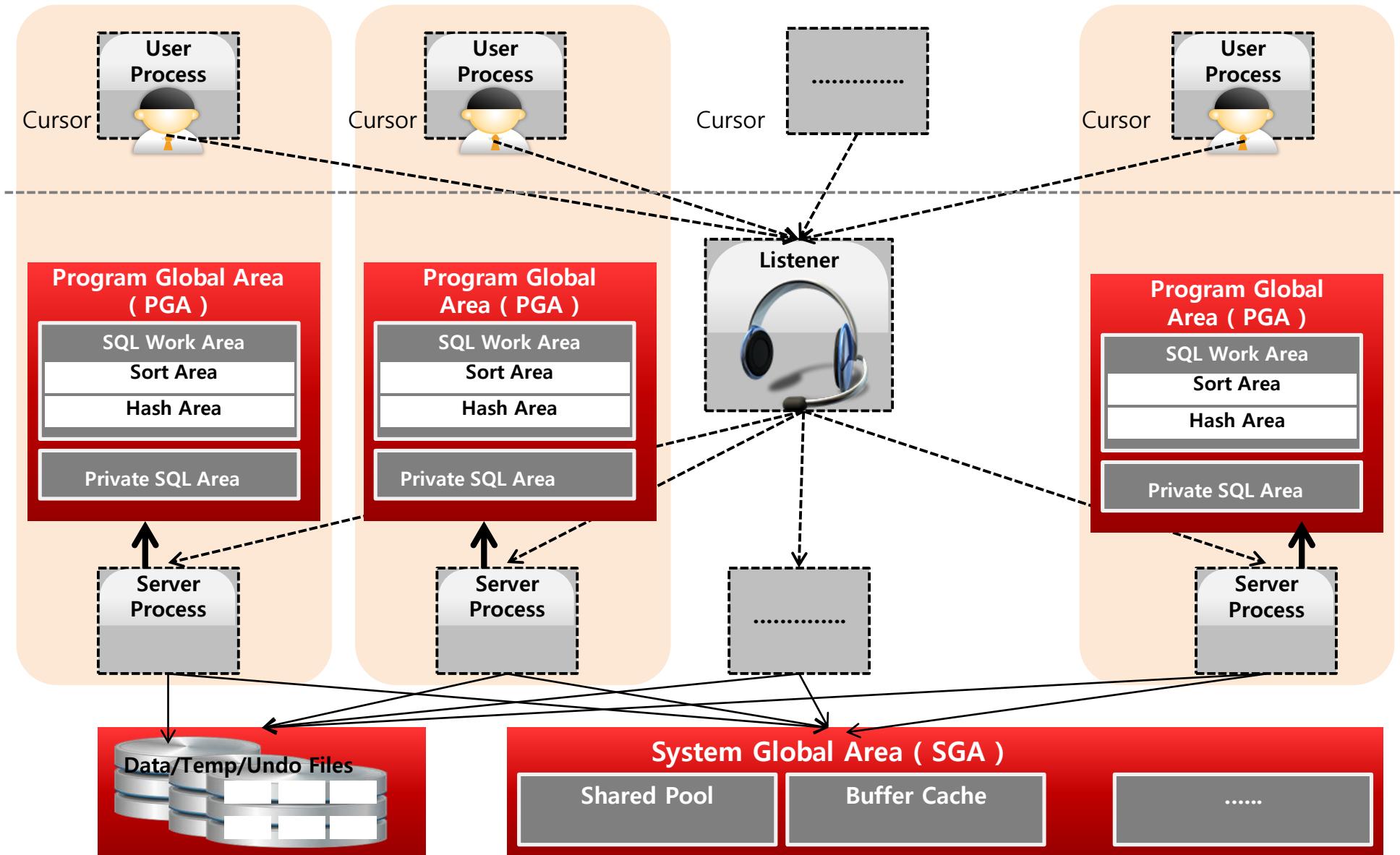
- Wait Event를 알기 위해서는 반드시 먼저 오라클의 Process, 메모리, 데이터베이스 파일들로 구성되는 내부처리 아키텍처를 이해해야 함
- 모든 Wait Event를 아는 것은 불가능하며, 주요 Wait Event들을 이해하고 왜 이러한 Event들이 발생할 수 밖에 없는지를 알아내는 것이 보다 중요.

유형	Event 명	Event 설명
Disk I/O	db file Sequential Read	Single block Access시에 발생(디스크에 Random I/O 수행할 경우 발생) 일반적으로 OLTP 시스템에서 가장 많이 발생
	db file Scattered Read	디스크로 부터 Full Scan을 수행할 경우 발생
자원 경합	library Cache Latch	Library cache 메모리에 접근하기 위해 Latch를 얻을 경우 발생
	library Cache Lock	Library cache 내의 특정 Object에 Lock 접근할 때 발생
	log file sync	변경 log buffer 를 log file에 반영하는 동안 발생
	enqueue	주로 Table 자체, 또는 Table Data의 Lock 수행 시 발생
	buffer busy wait	Buffer Cache내 동일 블록에 대한 동시 액세스 경합
	latch free	다양한 Shared Memory 영역에 대한 Latch

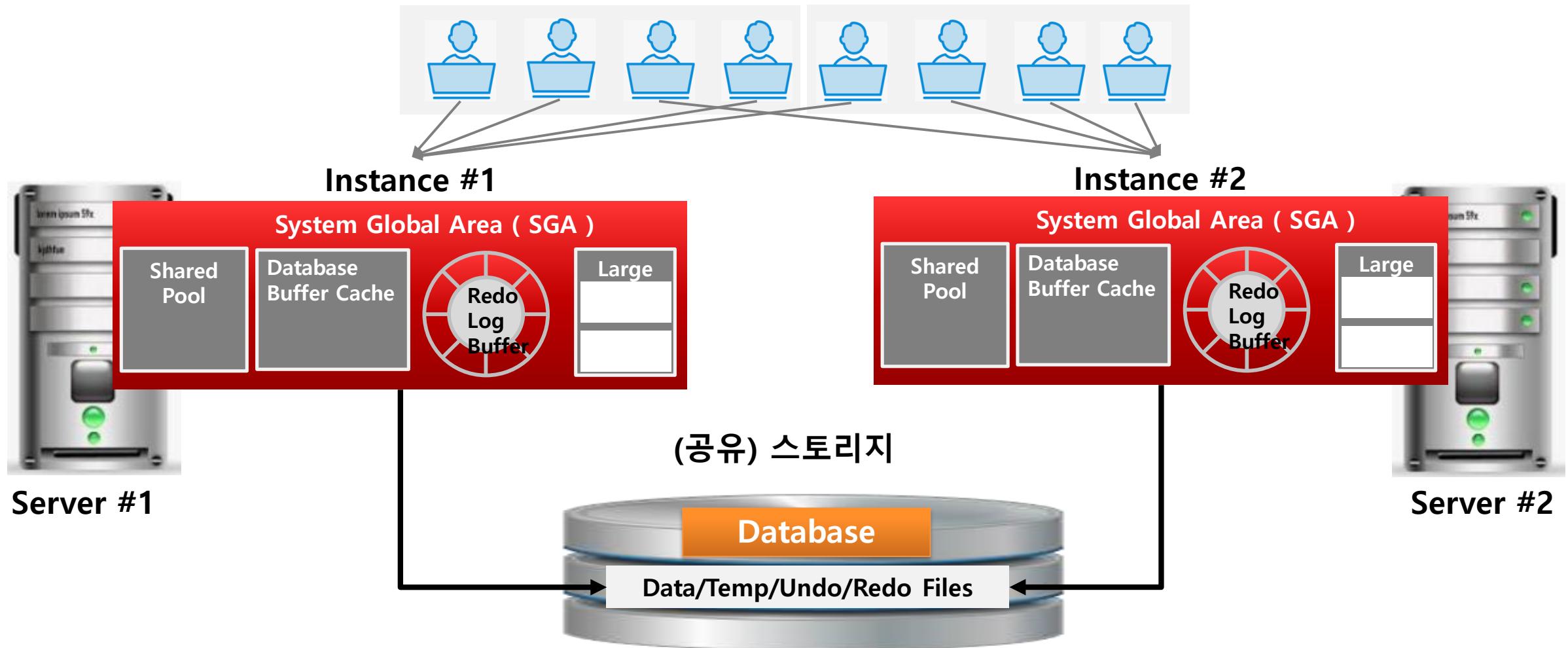
Wait Event Class

Class 명	Class 설명	Wait Event 예
USER I/O	User I/O 관련된 다양한 Wait Event	Db file sequential read Db file scattered read
System I/O	Background process I/O 와 관련된 Wait Event	Db file parallel write
Administrative	Index Rebuild와 같은 DBA Command 로 인하여 발생하는 Wait Event	Alter rbs offline
Concurrency	Latch와 같은 Internal Database resource와 관련된 Wait Event	Buffer busy wait Library cache lock
Cluster	RAC 의 Resource와 관련된 Wait Event	Gc cr block busy
Application	사용자 Application에서 사용된 row level Locking 또는 명확한 Lock Command 로 인한 Wait Event	Enq:TX – row lock Contention Wait for Table Lock
Commit	Commit 시 발생하는 Redo log write에 관련된 Wait Event	Log file sync
Idle	세션이 Inactive 상태에서 Work을 수행 준비하는 단계에서 발생하는 Wait Event	SQL *Net Message form client
Network	네트워크 Messaging시에 발생하는 Wait Event	SQL *Net more data to dblink
Other	정상적인 상황에서는 잘 발생하지 않는 Wait Event	Wait for EMON to spawn
Configuration	Database 또는 Instance 자원의 부적절한 환경 설정 예를 들어 log file size , shared pool size 를 적절하게 설정하지 못한 경우 발생하는 Wait Event	Write complete waits

Oracle DB 접속 개요



Instance와 Database의 구분 및 RAC 개념



SID, SERVICE_NAMES, DB_NAME, GLOBAL_DBNAME 구분

- SID는 Instance의 고유명을 나타냄.
- SERVICE_NAMES은 개념적으로 DB에서 서비스되는 Database 서비스명을 의미, 일반적으로 1개 또는 여러 개의 instance가 결합되어 서비스 가능.
- DB_NAME은 DB의 NAME, DB_UNIQUE_NAME은 DR을 고려한 DB의 고유 NAME, DB_DOMAIN은 집합적인 DB 도메인
- GLOBAL_DBNAME 역시 DB 이름 이지만 DB_NAME + DB_DOMAIN으로 구성. 기본 SERVICE_NAME은 GLOBAL_DBNAME으로 설정함.

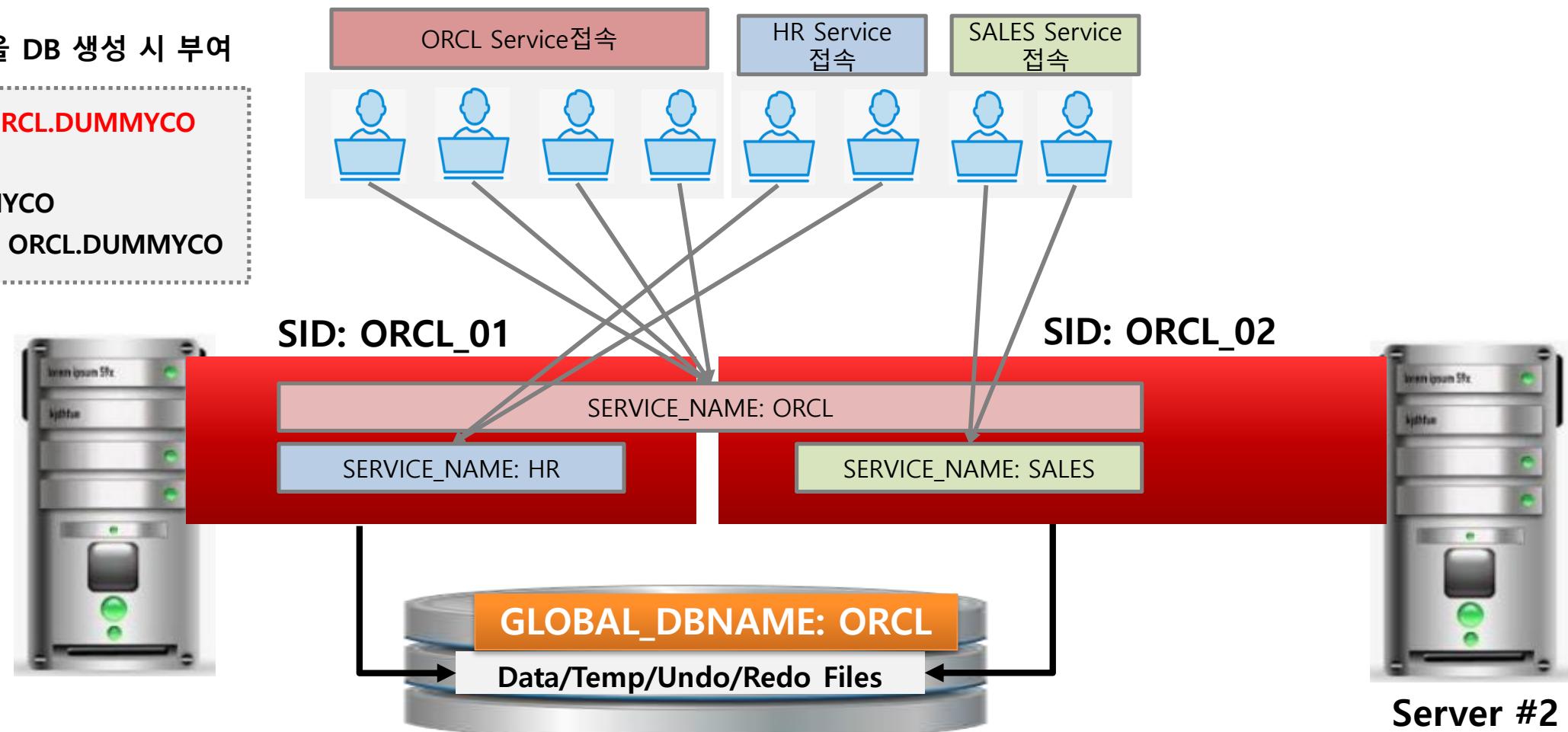
GLOBAL_DBNAME을 DB 생성 시 부여

GLOBAL_DBNAME: ORCL.DUMMYCO

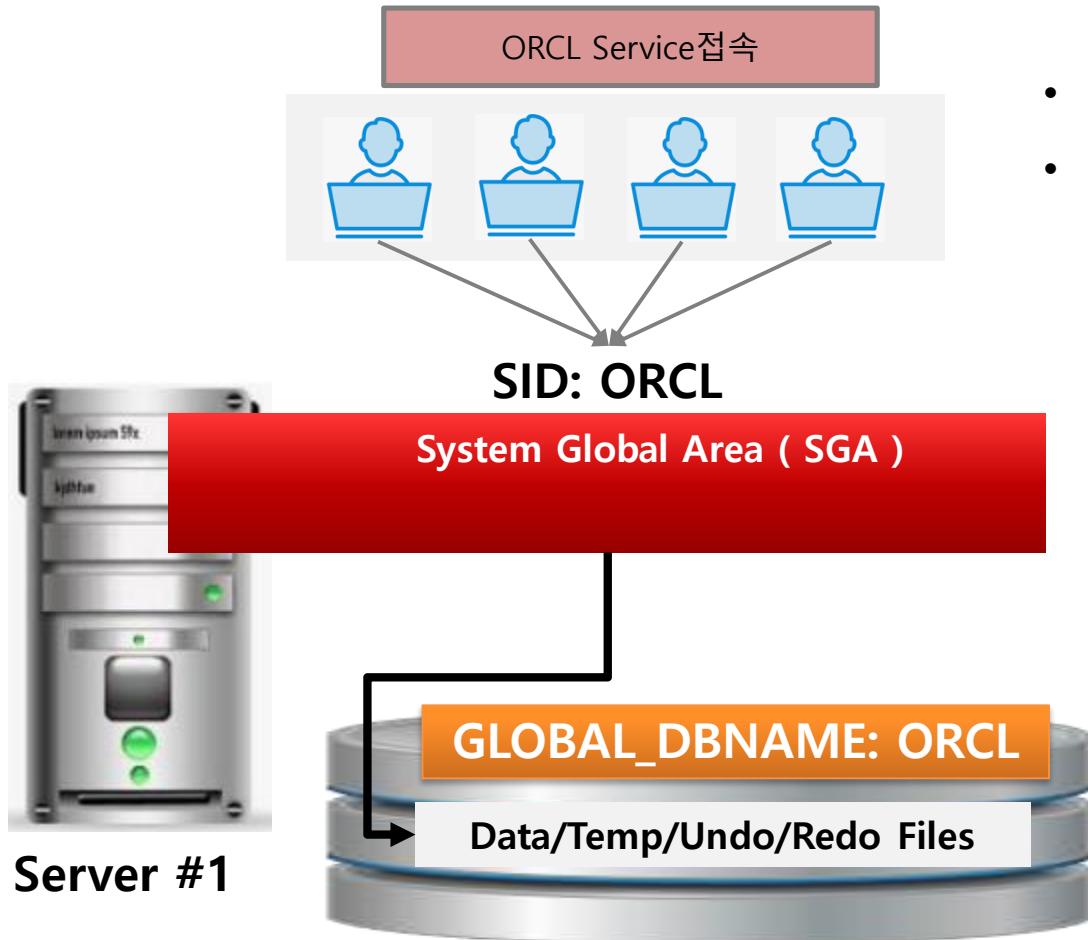
DB_NAME : ORCL

DB_DOMAIN: DUMMYCO

기본 SERVICE_NAME: ORCL.DUMMYCO



SID, SERVICE_NAME, DB_NAME, GLOBAL_DBNAME 구분



- 특별한 요건이 없는 이상 이들 4개는 보통은 모두 동일하게 설정
- GLOBAL_DBNAME의 경우 큰 조직은 경우 DB_NAME+DB_DOMAIN으로 부여

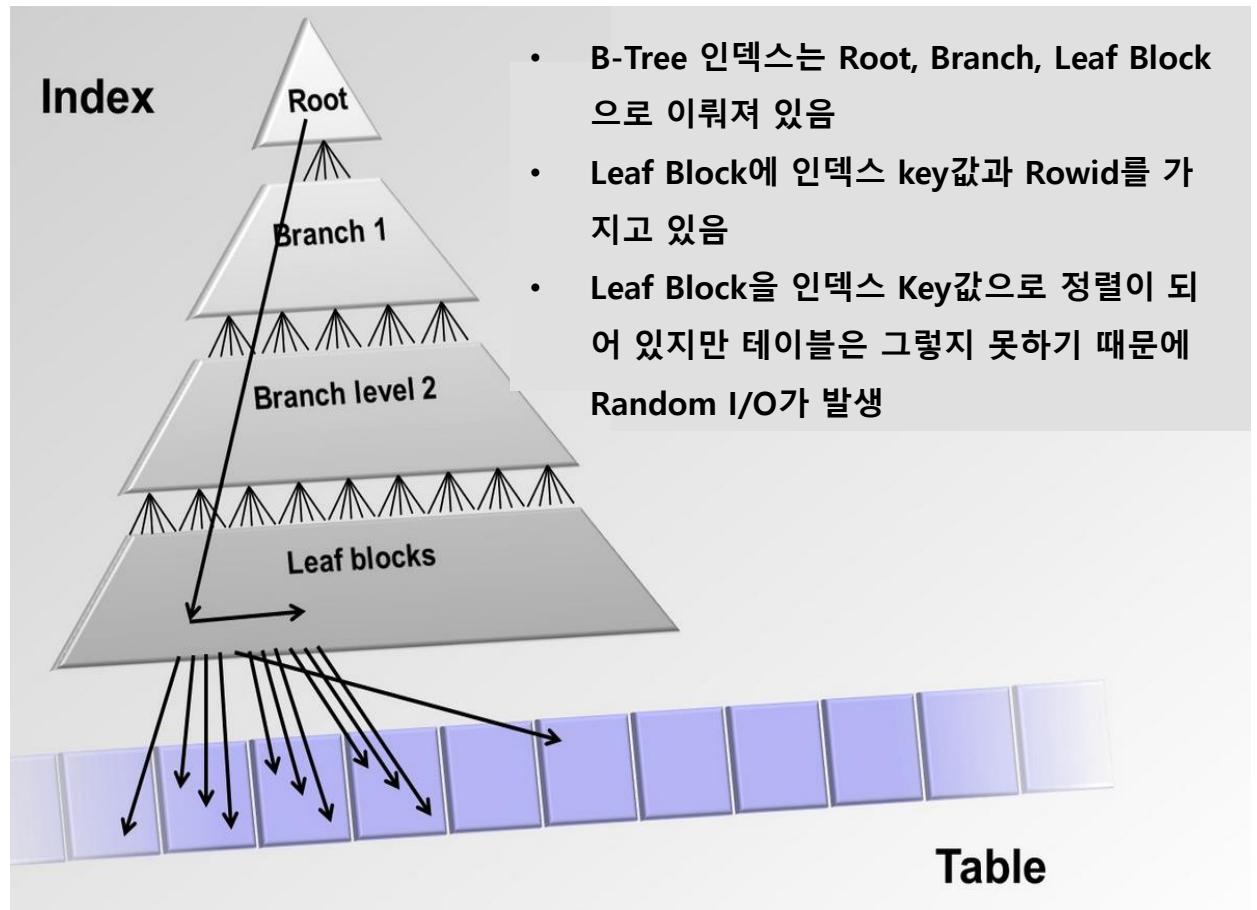
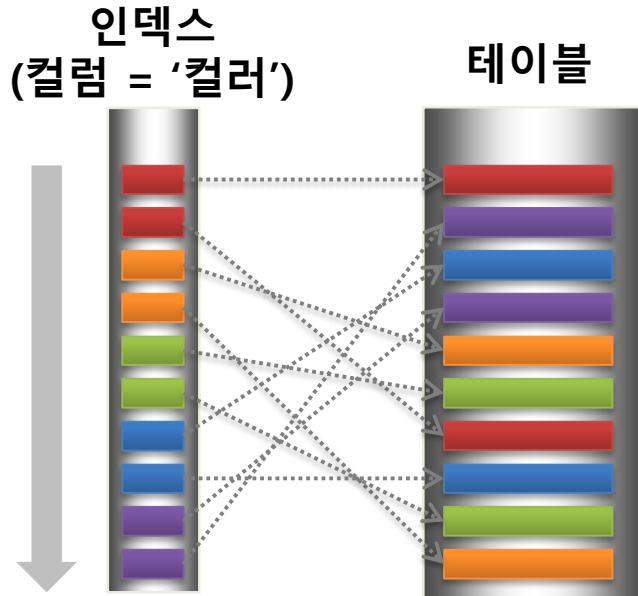
SID=DB_NAME=SERVICE_NAME=GLOBAL_DB_NAME

B-Tree 인덱스

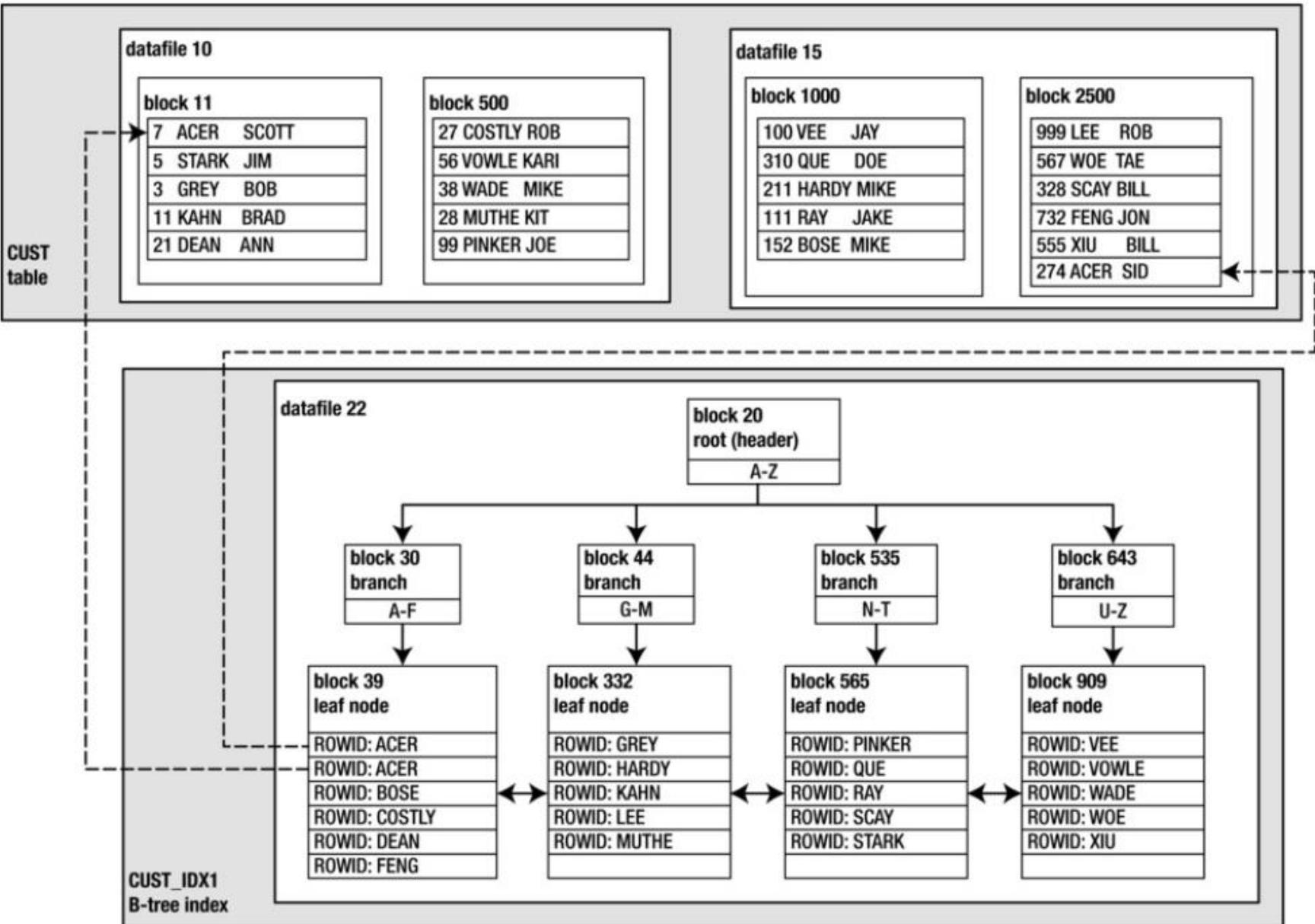
- 컬럼값과 해당 컬럼값의 테이블 Row가 위치해 있는 Rowid 값으로 구성되어 있음.
- 인덱스에서 해당 컬럼값을 조사한 뒤 Rowid를 이용하여 Table에 Access 함.

Index		Table	
Indexed Value	ROWID	ID	Value
1	AAAS5SAAEAAAAC1AAA	1	'Guatemala'
2	AAAS5SAAEAAAAC1AAB	2	'Brazil'
3	AAAS5SAAEAAAAC1AAC	3	'Argentina'
4	AAAS5SAAEAAAAC1AAD	4	'Colombia'
5	AAAS5SAAEAAAAC1AAE	5	'Uruguay'

B-Tree 인덱스를 통한 테이블 Access

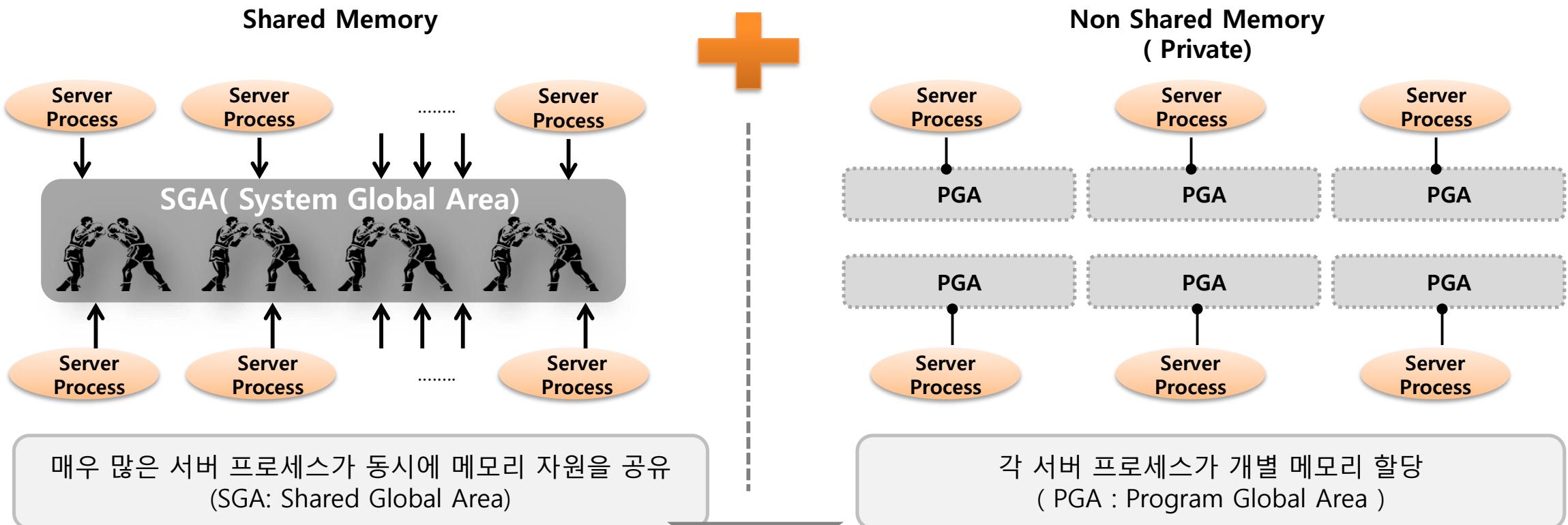


B-Tree 인덱스 세그먼트 구조



SGA(Buffer Cache, Shared Pool, Redo Log)와
ASMM 이해

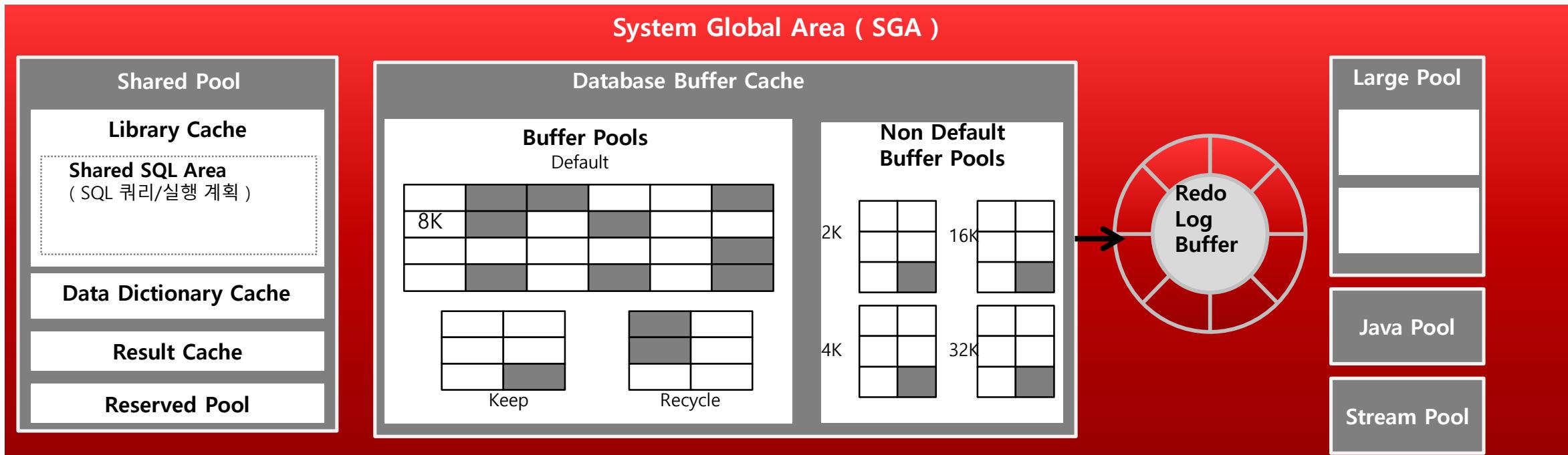
Oracle Memory Structure



오라클의 메모리 활용 전략

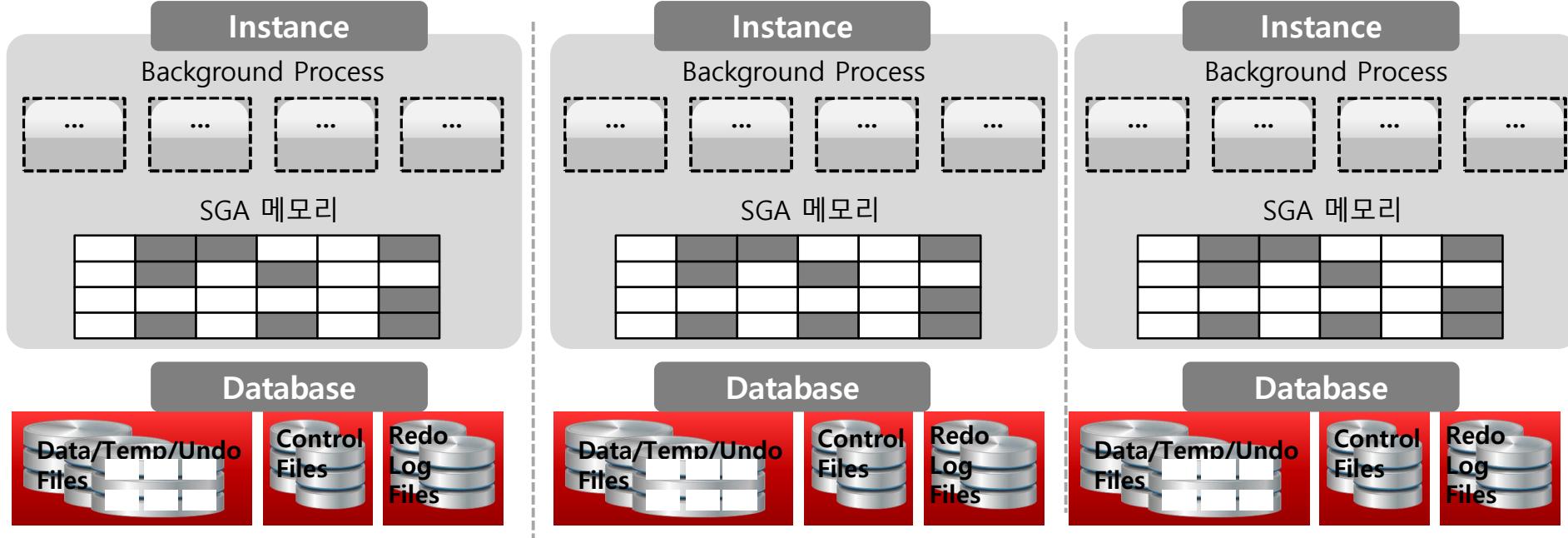
- Disk I/O 를 최대한으로 줄일 수 있는가?
- 동시에 많은 프로세스에서 최대한 효율적으로 자원할당을 할 것인가?
- 시스템 안정성을 어떻게 유지할 것인가?

Oracle SGA 의 개요



- **Database Buffer Cache :** Disk I/O의 영향도를 줄이기 위해 데이터 파일의 데이터 블록을 메모리로 Copy
- **Shared Pool :** 사용자들간에 SQL 및 SQL 실행 계획을 공유하기 위한 메모리 공간
- **Redo Log Buffer :** 데이터 변경사항 (DML 발생시) 정보를 가지며 Redo log file에 Write 되기 이전에 Memory에 먼저 Write 되는 영역
- **Large Pool :** Parallel Query 메시징 또는 대용량의 메모리 할당이 필요한 경우 사용됨.
- **Java Pool :** Java Object에 대한 메모리 영역
- **Stream Pool:** Data 복제 등을 위한 Stream 사용 영역

Instance 와 Database 에 대한 용어 정의



● Instance란?

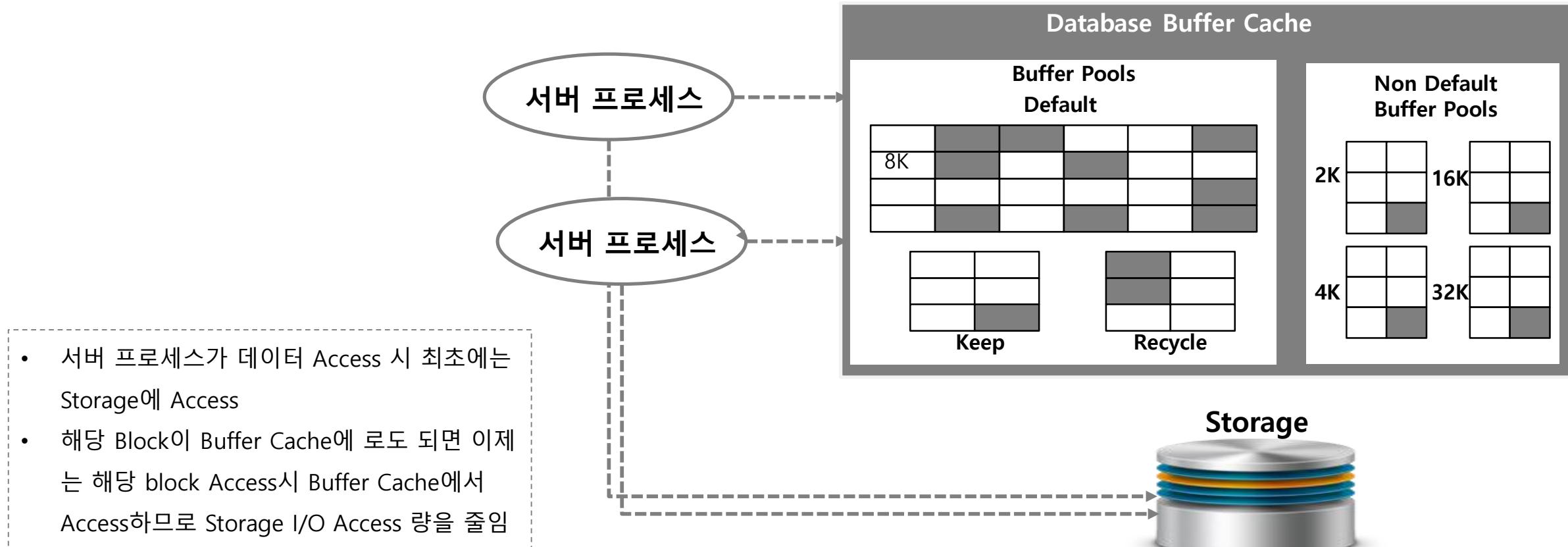
- Background Process와 SGA 메모리 영역을 일컫는다. 따라서 Instance를 시작하게 되면 SGA영역을 할당하고 Background Process를 구동시키게 된다.

● Database란?

- 데이터베이스를 저장하는 물리적 영역이라고 할 수 있다. dbf 파일, control 파일, log 파일 등 데이터를 저장되어 있는 물리적 공간이라고 생각하면 된다.

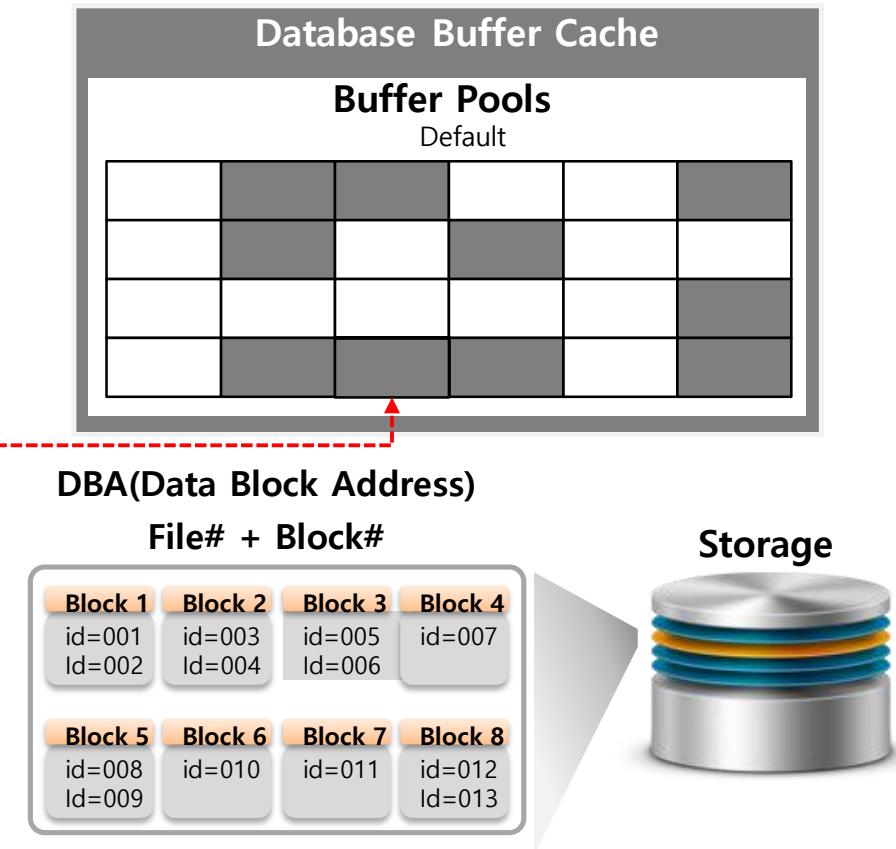
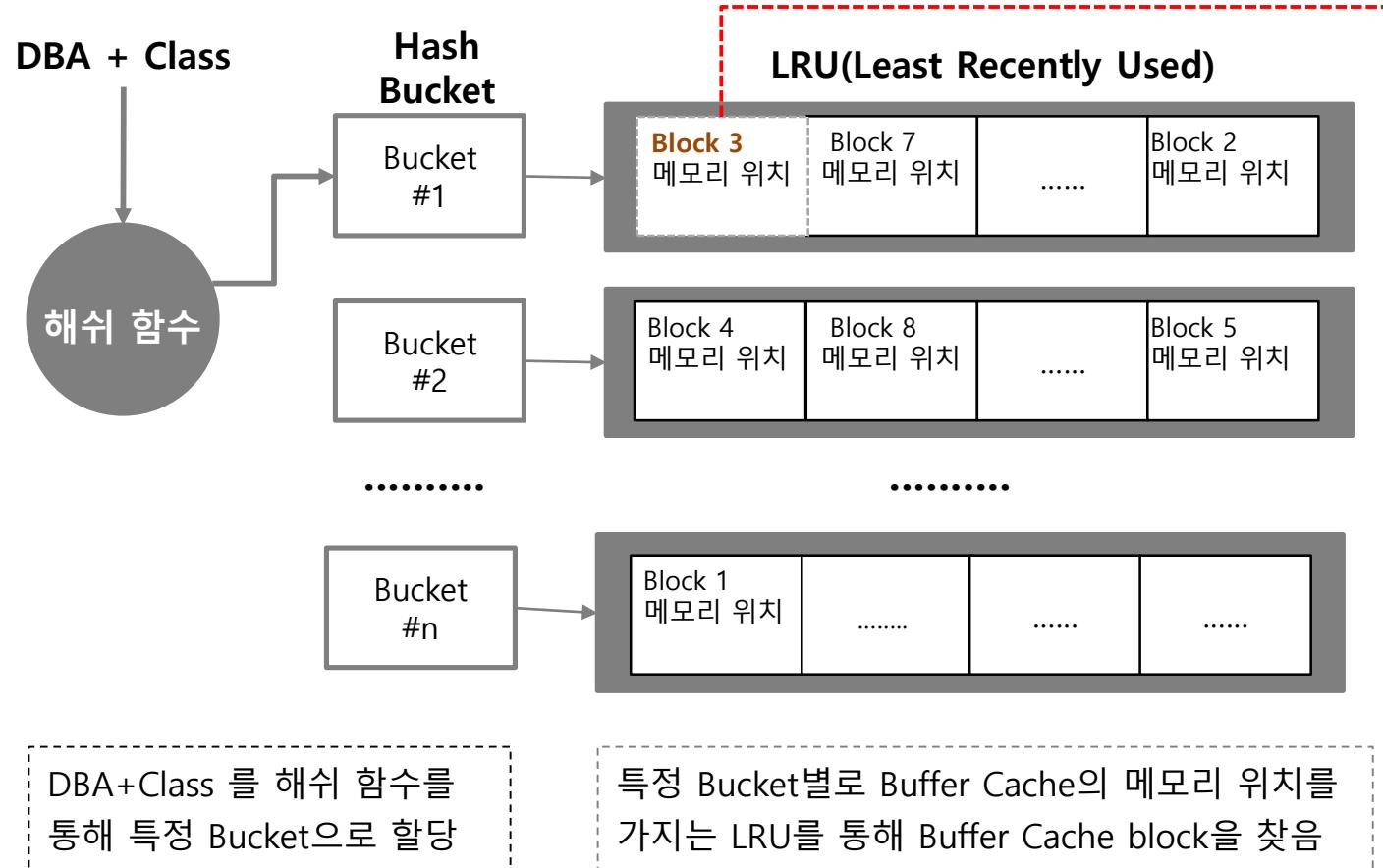
Buffer Cache 개요

- Storage I/O Access를 최소화 하기 위해서 한번 Access 한 Block은 RAM 영역에 저장하기 위한 공유 메모리 영역
- 특히 Random I/O 성능 영향을 최소화하는 중요 역할



Buffer Cache를 통한 데이터 Access

- Buffer Cache에 어떤 Block이 존재하는지 찾기 위해서 Hash Bucket과 LRU 제공
- 개별 데이터 Block은 File#와 Block#로 구성되는 DBA(Data Block Address)로 표현 될 수 있으며, DBA는 Hash Bucket으로 쪼개져서 LRU 내에서 Buffer cache의 위치 address를 가지게 됨.



Buffer Cache를 통한 데이터 Access



Select * from customer where id='005'

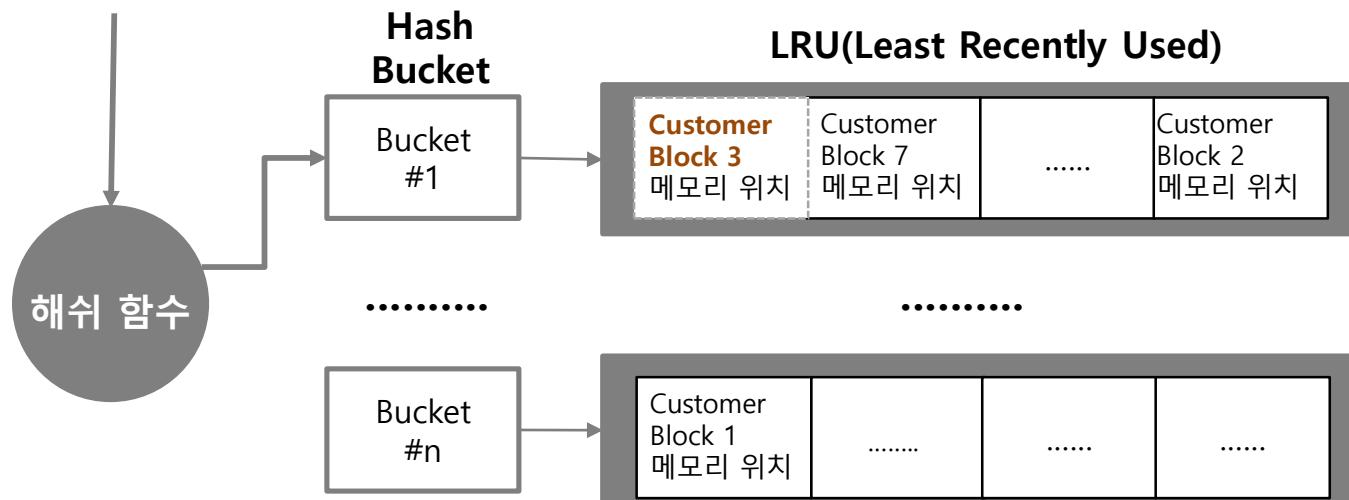
DBA(Data Block Address)

File# + Block#

DBA + Class

1

Data Dictionary 등의 내부 정보를 통해서 Customer 테이블의 id='005' 데이터가 있는 Block의 DBA 정보를 알아냄



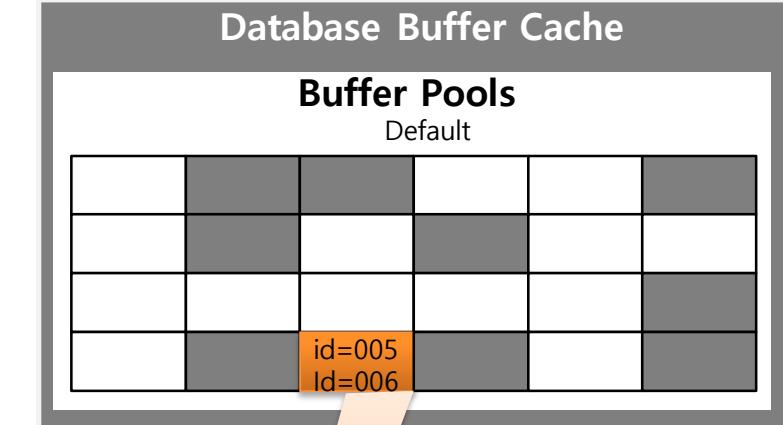
2

해당 Block의 DBA 파악 후 Buffer Cache에 이미 올라가 있는 Block 인지 확인하기 위해 Hash Bucket을 거쳐서 해당 LRU 검색

LRU는 특정 Block의 Buffer Cache내 위치가 어디인지 정보를 가지고 있음.

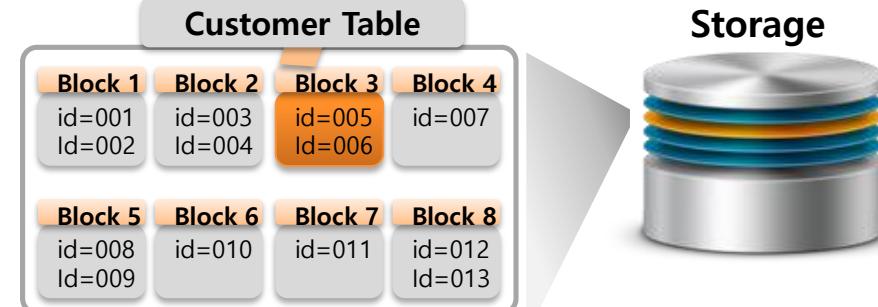
3

LRU를 Scan 하였지만 대상 Block이 없다면 이는 Buffer Cache에 존재하지 않는 것이므로 1에서 얻어낸 위치정보로 Disk에 Access

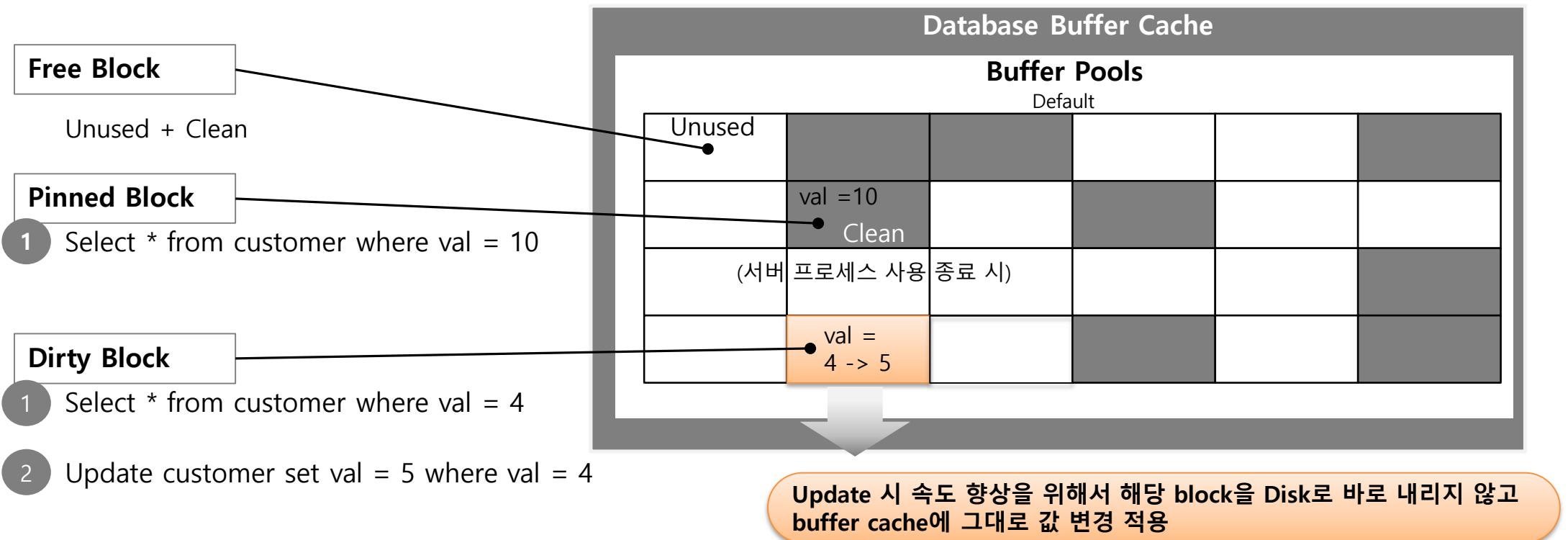


4

디스크의 해당 Block을 Buffer Cache에 올림.
LRU의 맨앞에 해당 Block에 대한 정보를 등록



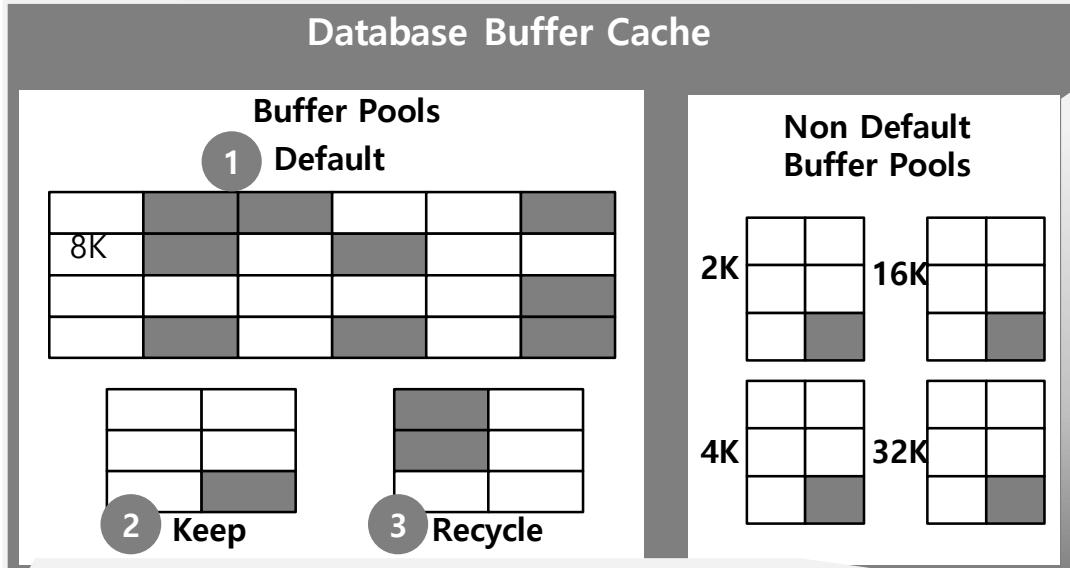
Buffer Cache 내 블록 유형 (Free/Dirty/Pinned block)



- **Free Buffer:** 아직 데이터가 할당되지 않은 buffer block(Unused)이거나 Pinned buffer 였다가 서버 프로세스의 사용이 종료되고 update되지 않은 buffer(Clean). 디스크에서 읽은 신규 데이터를 할당 가능함.
- **Pinned Buffer:** 데이터가 할당된 후 수정되지 않고 사용자 세션에 의해 사용되고 있는 Buffer block.
- **Dirty Buffer:** 데이터가 이미 할당된 이후에 다시 데이터가 수정되었으나 아직 disk 상의 data file에 Write 되지 못한 Buffer block.

Database Buffer Pools 의 유형

DB_BLOCK_SIZE = 8K 일 경우



- DW성, 이력 통계 데이터와 같은 대용량의 데이터를 Access해야 할 경우 큰 Block Size가 상대적으로 유리
- 이를 위해 DB_BLOCK_SIZE에 기술된 SIZE와 다른 BLOCK 크기를 가진 테이블 스페이스 생성 가능. (2K, 4K, 16K, 32K 등)
- 이와 같은 용도로 생성된 테이블스페이스 상의 Object에 Access 할 경우 Non Default Buffer pool 을 이용

- 1 **Default Buffer Cache** : DB_BLOCK_SIZE에서 정의된 Block Size로 Block이 할당된 Buffer Cache로서 별도의 Storage Option이 없으면 Table/Index등의 Object는 Default Buffer Cache에 등록
- 2 **Keep Buffer Cache** : Table/Index 등의 Object를 Keep Buffer에 등록하면 Buffer Cache에서 내려오지 않고 지속적으로 유지됨. 자주 쓰이는 조회성 데이터의 메모리 Hit Ratio를 높이기 위해 사용
- 3 **Recycle buffer cache** : 비교적 Access 빈도가 적은 Object가 메모리에 올라오게 되면 해당 Transaction이 종료된 후에는 다시 메모리에서 제거됨. 거의 사용되지 않음.

Database Buffer Pools 의 유형

Buffer Pool 유형	Buffer Cache 유형	크기 설정 파라미터
Default Buffer pool	Default Buffer Cache	DB_CACHE_SIZE
	Keep Buffer	DB_KEEP_CACHE_SIZE
	Recycle Buffer	DB_RECYCLE_CACHE_SIZE
Non Default Buffer Pool	Non Default Buffer Cache	DB_nK_CACHE_SIZE (n 은 Kbyte block size. 2,4,8,16,32 중 하나)

명령어

Keep Buffer에 Object 등록: ALTER TABLE CUSTOMER **STORAGE (BUFFER_POOL KEEP)**

Recycle Buffer에 Object 등록: ALTER TABLE CUSTOMER **STORAGE (BUFFER_POOL RECYCLE)**

Default Buffer 활용 :ALTER TABLE CUSTOMER **STORAGE (BUFFER_POOL DEFAULT)**

DB_KEEP_CACHE_SIZE = 48MB

DB_RECYCLE_CACHE_SIZE=24MB

DB_CACHE_SIZE=128M /* Standard DB_BLOCK_SIZE 8K */

DB_4K_CACHE_SIZE=48MB /* Non Standard DB_BLOCK_SIZE 4K */

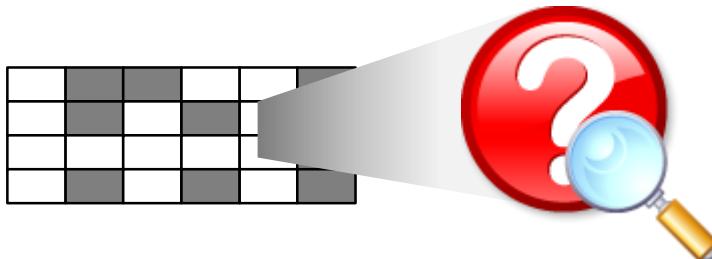
DB_16K_CACHE_SIZE=192MB /* Non Standard DB_BLOCK_SIZE 16K */

DB_32K_CACHE_SIZE=384MB /* Non Standard DB_BLOCK_SIZE 32K */



총 Buffer Cache 크기는 824MB

Buffer Cache Hit Ratio



시스템이 얼마나 Disk I/O 를 줄이고 Buffer Cache를 잘 활용하고 있는지를 나타내는 지표

$$\text{Hit Ratio} = (1 - (\text{Physical reads}/\text{Logical Reads})) * 100$$

- Logical Reads : Buffer Cache Block 액세스 수
- Physical Reads : Disk Block 액세스 수

일반적으로 90% 이상의 수치 보장 필요

90% 이하 수치는 나쁜 것인가?

VS

90% 이상 수치는 항상 좋은 것인가?

Buffer Cache Hit Ratio 통계 함정 - 1

Buffer Cache Hit Ratio 99%이면 전반적인 SQL 수행속도가 양호한 것일까?

100,000의 1%

VS

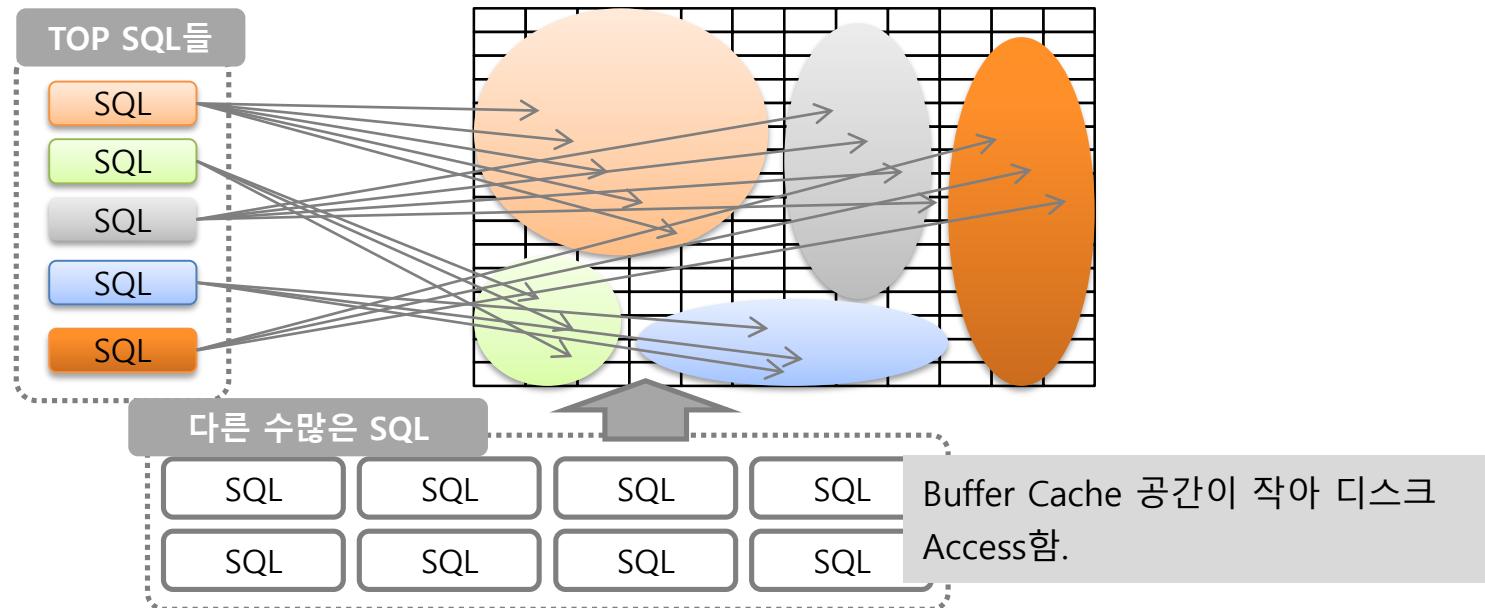
1,000,000,000의 1%

십만의 1%는 천, 그러나 십억의 1%는 천만

수백, 수천개의 SQL 중 단 1%의 SQL들의 호출빈도가 전체 SQL 호출 빈도에 대부분을 차지하게 된다면?

만일 Application 전체적으로 SQL들이 Block을 많이 Access해야 되는 잘 튜닝 되어 있는 않은 SQL이라면?

- Access block수는 작지만 호출 빈도수가 매우 높은 경우 전체 Logical Reads 는 호출 횟수 x access block 수가 되어 크게 증가하게 됨.
- 호출 빈도수가 높으면서 Access Block 수는 많은 SQL 들이 Buffer Cache를 지속적으로 점유하게 되면 Logical Reads 는 크게 증가하게 됨.
- 이들을 제외한 다른 많은 SQL들은 Buffer Cache 공간을 차지할 기회가 상대적으로 적어 디스크 Access 를 많이 하게 되나 워낙 호출 빈도수가 높은 SQL들의 Logical Reads 수치가 높아 Buffer Cache Hit Ratio가 높게 나오는 현상 발생



Buffer Cache Hit Ratio 통계 함정 – 2

- 저녁/새벽에 수행되는 Batch 프로그램은 Buffer cache hit ratio를 떨어뜨리지만 batch 수행 시 저하되는 hit ratio는 워낙 대용량 데이터를 처리하면서 발생하므로 자연스러운 현상일 수 있음.

Buffer Cache Hit Ratio 통계 함정 -3



Hit Ratio와 같은 성능 통계 정보는 불균일한 Load가 발생할 경우 정확한 문제 인식을 제공하지 못할 가능성이 높음



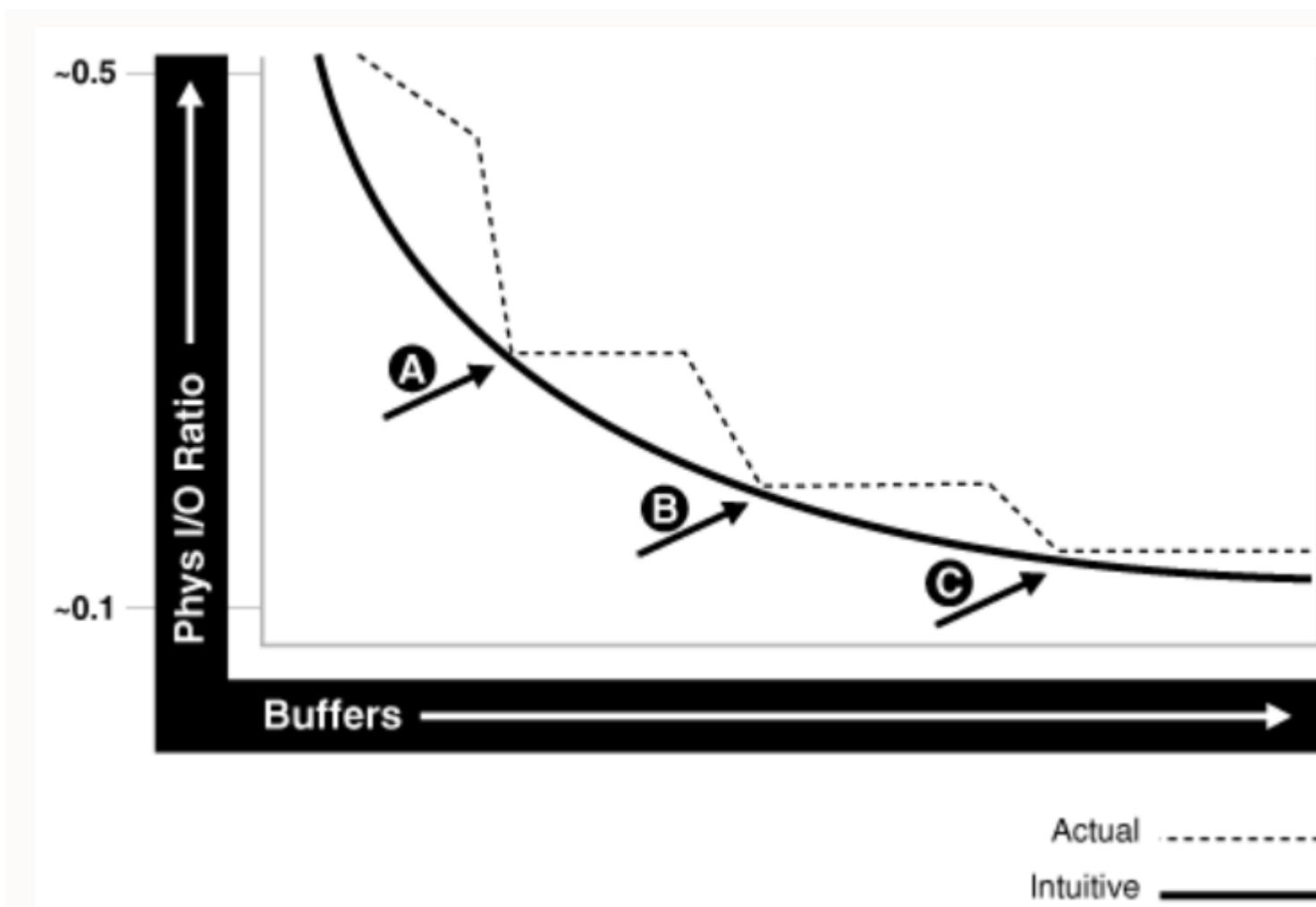
퍼센트가 아닌 절대 일량 수치 (메모리 Access량 , 디스크 Access 량)을 기준으로 성능 평가 필요



Buffer Cache Hit Ratio 외에 Load Profile + Wait Event 를 결합한 분석 필요

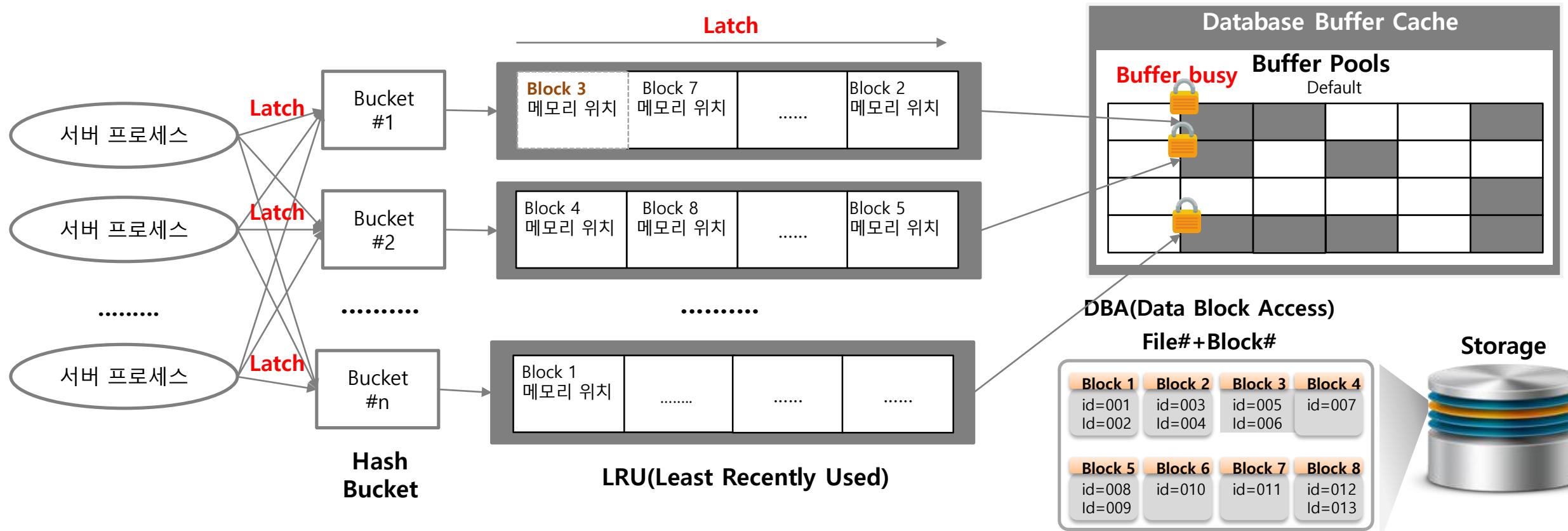
Buffer Cache를 증가하는 것은 언제나 성능향상을 시키는가?

Buffer Cache 증설 대비 Physical I/O Ratio 감소 효과



Buffer Cache를 Access하는 작업은 높은(?) 비용을 소모

- Buffer Cache를 활용하면 Random Access를 획기적으로 줄여 SQL 수행 성능을 크게 향상 시킬 수 있으나, Buffer Cache를 Access하기 위해서는 메모리의 Latch나 Lock과 같은 비용이 소모됨
- 만일 대량의 Latch와 Buffer busy가 소모되어 이는 공유 메모리를 사용하기 위한 자원을 감당하기 어려운 임계점이 되면 전체 SQL 성능을 떨어뜨릴 수 있을 뿐만 아니라 시스템 안정성을 위협할 수 있음



악성 SQL들로 인해 Buffer Cache 증설 효과 미비

1시간 동안의 AWR

Cache Sizes

	Begin	End		
Buffer Cache:	10,384M	10,384M	Std Block Size:	8K
Shared Pool Size:	7,056M	7,056M	Log Buffer:	30,700K

Buffer Cache가 현재 10G, 더 증가 시켜야 하는가?

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	99.72	Redo NoWait %:	100.00
Buffer Hit %:	97.94	In-memory Sort %:	100.00
Library Hit %:	97.27	Soft Parse %:	97.12
Execute to Parse %:	63.81	Latch Hit %:	99.61
Parse CPU to Parse Elapsd %:	85.98	% Non-Parse CPU:	97.47

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
db file sequential read	2,799,626	11,216	4	41.0	User I/O
CPU time		9,562		34.9	
gc buffer busy	1,095,344	2,177	2	8.0	Cluster
gc cr grant 2-way	1,357,518	1,144	1	4.2	Cluster
db file scattered read	488,660	961	2	3.5	User I/O

악성 SQL들로 인해 Buffer Cache 증설 효과 미비

1시간 동안의 AWR

SQL ordered by Gets

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- Total Buffer Gets: 385,423,694
- Captured SQL account for 49.0% of Total

Buffer Gets	Executions	Gets per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
48,802,744	2,002	24,377.00	12.66	1232.80	1210.51	3cckptg80jp0r	JDBC Thin Client	Query 1
39,510,347	1,620	24,389.10	10.25	984.51	966.90	152h031qafmsb	JDBC Thin Client	Query 2
13,316,796	207	64,332.35	3.46	348.81	563.31	3c7cdsarqcb1g	JDBC Thin Client	Query 3
10,282,777	124	82,925.62	2.67	57.61	101.14	aa5kg8y8km37vm	JDBC Thin Client	Query 4
8,883,054	82	108,329.93	2.30	45.89	90.40	22k8b99gfd8bh	JDBC Thin Client	Query 5
8,388,060	90	93,200.67	2.18	56.85	55.76	7xyvf367s1hqp	JDBC Thin Client	Query 6
7,882,506	84	93,839.36	2.05	67.26	66.06	9kh1htsk6wgjv	JDBC Thin Client	Query 7

SQL ordered by Reads

- Total Disk Reads: 4,303,014
- Captured SQL account for 54.4% of Total

Physical Reads	Executions	Reads per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
1,443,555	207	6,973.70	33.55	348.81	563.31	3c7cdsarqcb1g	JDBC Thin Client	Query 3
147,615	74	1,994.80	3.43	58.91	1052.78	3t012kz0cq67u	JDBC Thin Client	Query 8
132,243	33	4,007.36	3.07	54.78	548.83	90tgjk30g94wr	JDBC Thin Client	Query 9
97,187	3	32,395.67	2.26	13.81	222.11	8j9x0brshcvm6	JDBC Thin Client	Query 10
74,506	43	1,732.70	1.73	14.32	425.69	0xt49thxbu7az	JDBC Thin Client	Query 11
67,626	60	1,127.10	1.57	14.46	233.52	00jtqjrc3vhrr	JDBC Thin Client	Query 12

Buffer Cache 크기 설정

크기 설정 유형	DB_CACHE_SIZE	SGA_TARGET	SGA_MAX_SIZE	MEMORY_TARGET	MEMORY_MAX_TARGET
수동으로 크기 설정	Non Zero(예: 10G)	0	0	0	0
ASMM 으로 크기 설정	0	Non Zero (예: 12G)	Non Zero (예 SGA_TARGET 보 다는 큐)	0	0
AMM으로 크기 설정	0	0	0	Non Zero (예: 20G)	Non Zero (예: MEMORY_TARGET 보 다는 큐)

V\$DB_CACHE_ADVICE 설명

V\$DB_CACHE_ADVICE는 BUFFER CACHE의 크기 설정에 대한 Advice 정보 제공. DB_CACHE_ADVICE 초기화 파라미터가 ON되어야 함.

Column	Description
BLOCK_SIZE	해당 Buffer Pool의 Block Size
ADVICE_STATUS	ADVICE 기능 ON/OFF 여부. DB_CACHE_ADVICE 초기화 파라미터 ON/OFF
SIZE_FOR_ESTIMATE	성능 예측 대상 Buffer Cache 크기
SIZE_FACTOR	현재 Buffer Cache 크기 대비 예측 Buffer Cache 크기 비율
BUFFERS_FOR_ESTIMATE	성능 예측 대상 Buffer Cache의 buffer block수
ESTD_PHYSICAL_READ_FACTOR	현재 Physical Read 대비 예측 Physical Read I/O 의 비율
ESTD_PHYSICAL_READS	예측 Physical Reads block 수
ESTD_PHYSICAL_READ_TIME	예측 Physical Reads 시간
ESTD_PCT_OF_DB_TIME_FOR_READS	전체 수행 시간 대비 Physical Reads 시간

AWR의 Buffer Pool Advisory

Buffer Pool Advisory

- Only rows with estimated physical reads >0 are displayed
- ordered by Block Size, Buffers For Estimate

P	Size for Est (M)	Size Factor	Buffers (thousands)	Est Phys Read Factor	Estimated Phys Reads (thousands)	Est Phys Read Time	Est %DBtime for Rds
D	40	0.09	5	1.12	7,835	1	4187.00
D	80	0.19	10	1.07	7,474	1	3992.00
D	120	0.28	15	1.06	7,375	1	3939.00
D	160	0.37	20	1.05	7,314	1	3906.00
D	200	0.46	24	1.04	7,281	1	3888.00
D	240	0.56	29	1.04	7,251	1	3872.00
D	280	0.65	34	1.03	7,223	1	3857.00
D	320	0.74	39	1.02	7,160	1	3823.00
D	360	0.83	44	1.01	7,057	1	3768.00
D	400	0.93	49	1.00	7,017	1	3746.00
D	432	1.00	53	1.00	6,987	1	3730.00
D	440	1.02	54	1.00	6,979	1	3726.00
D	480	1.11	59	0.99	6,940	1	3705.00
D	520	1.20	63	0.99	6,901	1	3683.00
D	560	1.30	68	0.98	6,862	1	3663.00
D	600	1.39	73	0.98	6,835	1	3648.00
D	640	1.48	78	0.97	6,810	1	3635.00
D	680	1.57	83	0.97	6,784	1	3620.00
D	720	1.67	88	0.97	6,760	1	3607.00
D	760	1.76	93	0.96	6,733	1	3593.00
D	800	1.85	98	0.96	6,699	1	3574.00

Buffer Cache 내에 적재된 Object 들의 정보 확인

V\$BH는 실시간으로 Buffer Cache에 적재되어 있는 Block들에 대해 Object 정보 및 Block 상태 정보 제공 (DBA_OBJECTS와 조인하여 Object명 조회).

Column	Description
FILE#	Datafile# (DBA_DATA_FILES에서 datafile 명 조회)
BLOCK#	Buffer block number
CLASS#	Class number
STATUS	Buffer의 상태: <ul style="list-style-type: none">free - Not currently in usexcur - Exclusivescur - Shared currentcr - Consistent readread - Being read from diskmrec - In media recovery modeirec - In instance recovery mode
DIRTY	Y - block이 수정된 Dirty block 여부
OBJD	해당 buffer block의 Object ID
TS#	해당 buffer block의 테이블 스페이스 고유 번호

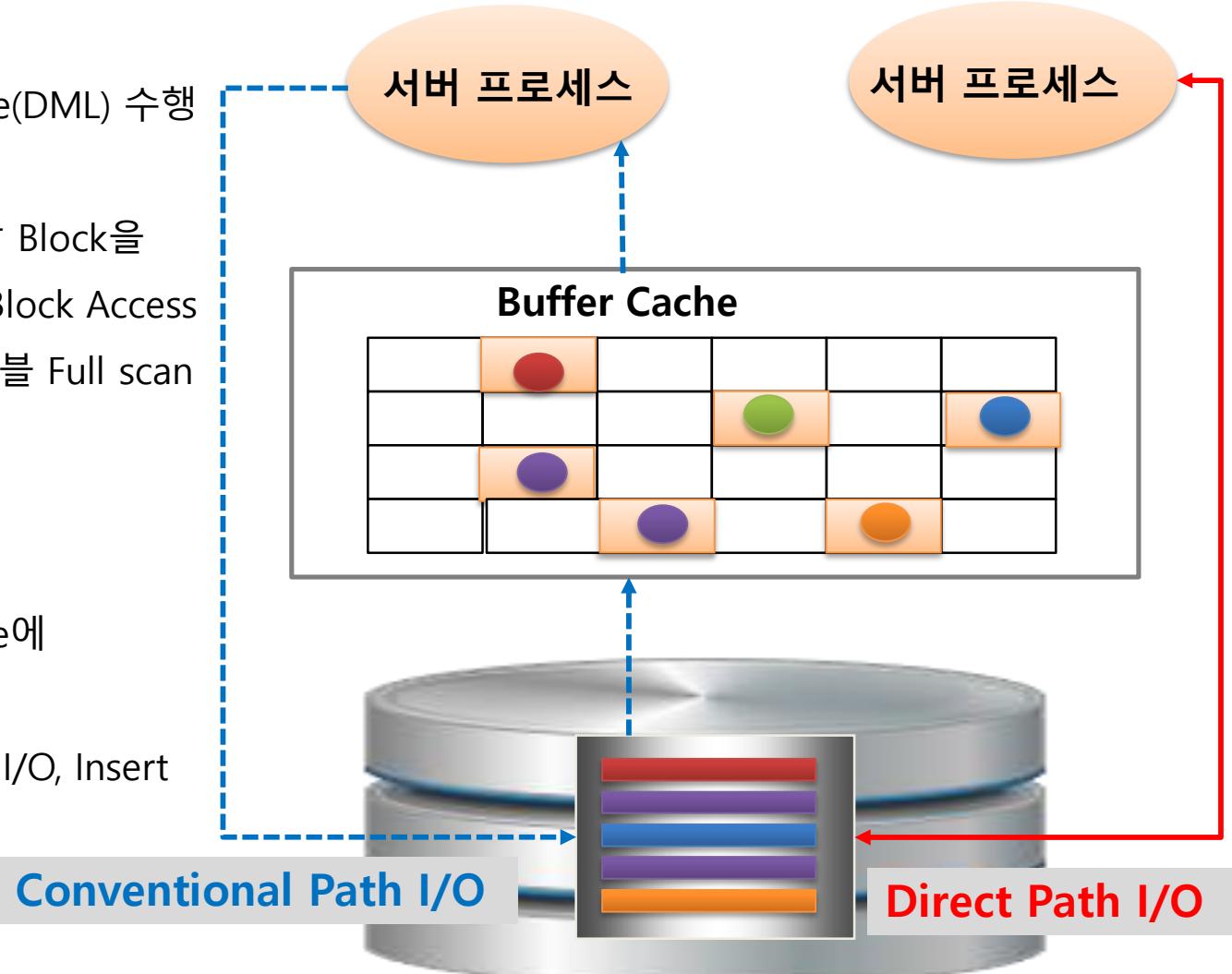
Conventional Path I/O vs Direct Path I/O

Conventional Path I/O

- 서버 프로세스가 Buffer Cache를 거쳐서 Read(Select)/Write(DML) 수행하는 I/O
- Buffer Cache에 Access할 Block이 없으면 Storage에서 해당 Block을 Access하여 Buffer Cache에 Load 한 뒤 Buffer Cache에서 Block Access
- Index를 경유한 테이블 Access(Random Access), 작은 테이블 Full scan

Direct Path I/O

- 서버 프로세스가 Buffer Cache를 거치지 않고 직접 Storage에 Read/Write 수행하는 I/O
- 대용량 테이블의 Full Scan, Parallel Query, Temp segment I/O, Insert 시 SQL Hint /*+ append */ 를 적용한 SQL
- DML 시에는 테이블 전체에 대해서 Exclusive Lock 필요



SQL 의 이해

SQL = Program ?

```
SELECT B.DEPT_ID, C.SAL_GRADE_ID, COUNT(A.EMP_ID) FROM EMP A, DEPT B, SALGRADE C WHERE A.DEPT_NO = B.DEPTNO  
AND B.SAL_GRADE_ID = C.SAL_GRADE_ID GROUP BY C.SAL_GRADE_ID GROUP BY B.DEPT_ID, C.SAL_GRADE_ID
```

사용자 SQL은 Oracle 내부에서 상세 실행 계획으로 변경



PLAN_TABLE_OUTPUT											
Plan hash value: 3841029040											
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	IN-OUT	PQ Distrib	
0 SELECT STATEMENT			24	3014	35 (0)	00:00:43			P->S	QC (ORDER)	
1 PX COORDINATOR		:TQ10003	24	3014	35 (0)	00:00:31			PCWP		
2 PX SEND QC (ORDER)									PCWP		
3 VIEW									PCWP		
4 SORT GROUP BY									PCWP		
5 PX RECEIVE			24	3014	35 (0)	00:00:01			PCWP		
6 PX SEND RANGE		:TQ10002	24	3014	35 (0)	00:00:01			PCWP		
* 7 HASH JOIN			24	3014	11 (0)	00:00:01			PCWP		
9 PX RECEIVE			24	3014	9 (0)	00:00:01			PCWP		
10 PX SEND BROADCAST		:TQ10001	480	64K	8 (0)	00:00:01			P->P		
11 PX BLOCK ITERATOR			480	64K	8 (0)	00:00:01			PCWP		
12 PX BLOCK ITERATOR			480	64K	8 (0)	00:00:01			PCWP		
* 13 TABLE ACCESS FULL		EMP	480	64K	8 (0)	00:00:21			PCWP		
14 PX BLOCK ITERATOR			1	248	4 (0)	00:00:01	1		PCWP		
* 15 TABLE ACCESS FULL		DEPT	8	248	4 (0)	00:00:11	8	14	PCWP		
16 PX PARTITION RANGE SINGLE			18	960	18 (0)	00:00:01	4	4	PCWP		
17 PX PARTITION HASH JOIN-FILTER			18	960	18 (0)	00:00:01	:BF0000	:BF0000	PCWP		
* 18 TABLE ACCESS FULL		SALGRADE	18	960	18 (0)	00:00:18	KEY	KEY	PCWP		

*Note: Query/Explain Plan adjusted per NDA

SQL 실행계획 영향 요소



```
SELECT count(*) CNT  
FROM kx045dt a, kx051dt b, ke020dk c WHERE a.doc_serial_no = b.srl_no AND  
a.proc_status > '00' AND b.docke_code = c.docke_code  
AND b.rec_dt BETWEEN TO_DATE (?) || '000000', 'YYYYMMDDHH24MISS')  
AND TO_DATE (?) || '235959', 'YYYYMMDDHH24MISS')  
AND a.owner_no = ? AND INS_CODE_SE=?
```

Optimizer 가 최적 실행 계획 판단

어느 테이블을 선정 집합
으로 드라이빙 할 것인
가 ?

Nested Loop Join? 또는
Hash Join ?

Full Table Scan ?
Index Range Scan ?

.....

kx045dt 테이블에
proc_status 인덱스가 있는가?

kx051dt 테이블의 크기는 얼
마인가?

kx051dt 테이블에 doc_code
인덱스가 있는가?

ke020dk테이블의 크기는 얼
마인가

.....

kx045dt 테이블에 owner_no
인덱스가 있는가?

ke020dk 테이블에
INS_CODE_SE 인덱스가 있는
가?

kx051dt 테이블에 rec_dt 인
덱스가 있는가?

kx051dt테이블의 크기는 얼마
인가

.....

Optimizer에 영향을 미치는 요소



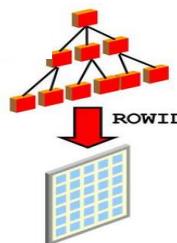
사용 컬럼,
연산자 형태

=, Like, in
Is not null,
NVL()

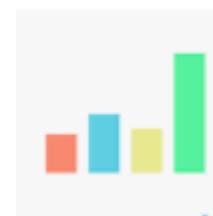
SQL 유형

Select * from
Tab where

인덱스 및
Segment 구조



통계 정보



옵티마이저 설정

ALL_ROWS,
FIRST_ROWS

DBMS 버전

11G, 12C,
18C, 19C

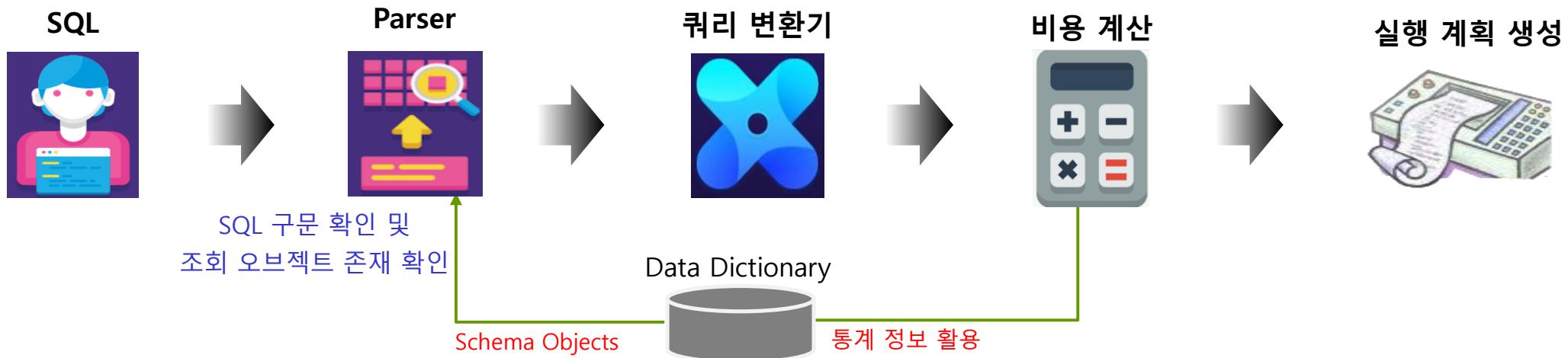
실행계획 생성을 위한 고려 요소



- 최종 결정된 실행 계획은 수많은 선택의 요소를 상호 Cost 비교하여 가장 최적의 결과 값으로 선택된 것임.
- 이를 위해 많은 CPU 및 기타 자원 소모.
- 특히 OLTP와 같은 초당 수십~수백개의 SQL이 동시에 들어올 경우 반드시 SQL 및 실행계획의 공유가 필요함.

Rows	Row Source Operation
1	SORT AGGREGATE
13	TABLE ACCESS BY INDEX ROWID JK040DT
4165876	NESTED
7875	NESTED LOOPS
12686	TABLE ACCESS BY INDEX ROWID JW051DT
12686	INDEX RANGE SCAN S2_JZ051DT (object id 53093)
7875	TABLE ACCESS BY INDEX ROWID JB020DM
12686	INDEX UNIQUE SCAN PK_JB020DM (object id 53058)
4158000	INDEX RANGE SCAN S10_JK040DT (object id 53084)

Optimizer SQL 실행계획 수행 절차



1

- 호출된 SQL은 데이터 딕셔너리를 참조하여 Parsing후 옵티マイ저가 좀 더 편하게 이해할 수 있는 행태로 쿼리가 일차 변환 됨.
- 옵티마이저는 변환된 쿼리를 기반으로 잠재적인 일차 실행계획들을 생성

2

- 데이터 딕셔너리에서 여러 통계정보(데이터의 분포도, 테이블 저장구조, 인덱스 구조, 파티션 형태, 비교연산자) 등을 감안하여 각 실행계획의 비용을 계산

3

- 실행계획들의 비용을 기반하여 빠르게 **가장 최소의 비용을 가진 실행계획을 선택**

Hard Parsing 과 Soft Parsing

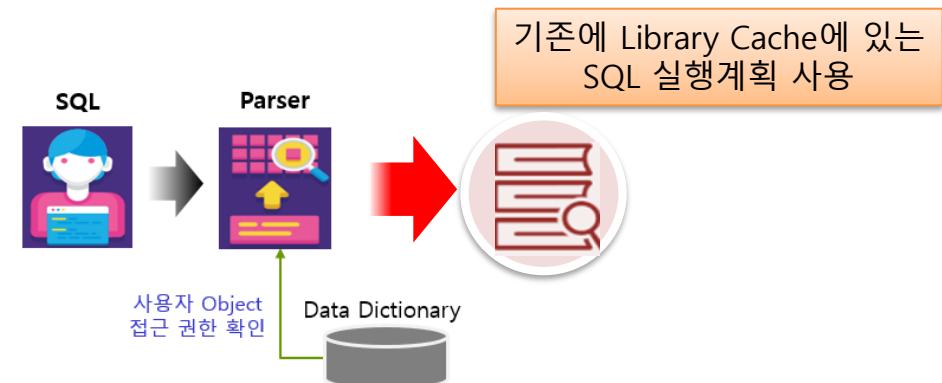
Hard Parsing

SQL 및 실행계획이 Library Cache에 존재하지 않아 SQL의 실행계획을 처음부터 파싱, 쿼리 변환, 비용 계산, 실행 계획 생성 등의 복잡한 단계를 거쳐서 만들어 지는 Parsing



Soft Parsing

SQL 및 실행계획이 Library Cache에 존재하므로 해당 Object에 대한 사용자 접근 권한정도만 확인한 뒤에 Library Cache에 있는 SQL 실행 계획을 그대로 사용하는 Parsing



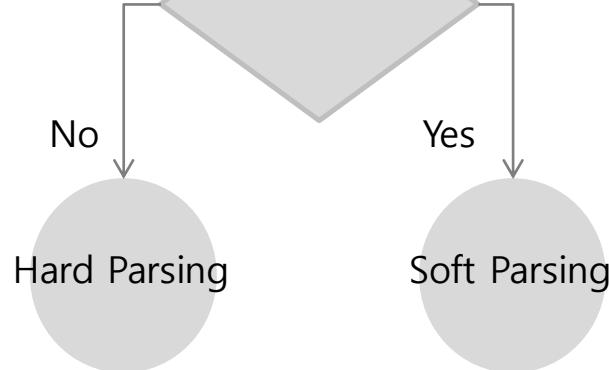
- Hard Parsing 이 높아 질수록 CPU 부하 , 특히 Library Cache의 Latch 관련 자원 사용률이 매우 높아져서 시스템 장애를 일으킬 정도의 부하를 유발할 수도 있음.
- SQL의 첫번째 사용시에는 Hard Parsing이 필요하나 이후에는 반드시 Soft Parsing이 될 수 있어야 함.
- SQL Bind 변수의 필수 사용 등으로 거의 대부분의 SQL은 Soft Parsing으로 유도 할 것

Shared Pool 구조

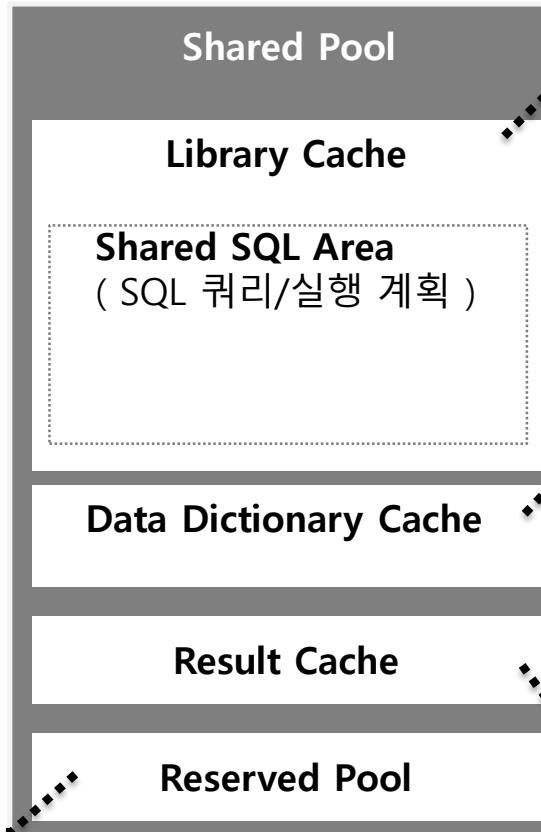


Select * from customer where id='005'

해당 SQL이 Library Cache에 존재하는가 ?



매우 큰 PL/SQL 패키지와 같은 대용
량 SQL관련 오브젝트들을 저장하기
위한 별도 공간



SQL 또는 PL/SQL 코드를 모두 보관하는 장소

사용된 SQL 및 실행계획이 파싱 되어 보관됨.

테이블/인덱스 등의 Object 정의, 사용자명, Role ,
권한 등의 정보를 보관하는 장소

주로 사용자가 해당 Object에 접근 가능 권한이 있
는지 확인하는데 사용

사용자 SQL의 결과 값을 지속적으로 보관

동일한 SQL이 요청될 경우 Buffer 나 Disk I/O
Access 작업을 수행하지 않고 Result Cache에 있
는 값을 그대로 반환

SQL Bind 변수 사용의 필요성

Literal SQL

```
Select * from Customer where cust_id = '100000'  
Select * from Customer where cust_id = '200001'  
Select * from Customer where cust_id = '200002'
```

- Optimizer는 SQL 의 문자 하나만 차이가 생겨도 서로 다른 SQL 로 인식
- 왼쪽의 3개 SQL 은 모두 서로 다른 SQL로 인식되며 첫 번째 호출된 SQL에서 생성된 실행계획은 다른 SQL과 서로 공유할 수 없으므로 모두 Hard Parsing 수행
- 위와 같은 Literal SQL 형식으로 Application 이 작성되고 많은 동시 접속 사용자에 의해 사용된다면 **Hard Parsing으로 인한 시스템 부하가 급증 가능성 존재**

VS

Bind 변수 SQL

```
Select * from Customer where cust_id = 1  
Select * from Customer where cust_id = 1  
Select * from Customer where cust_id = 1
```

Bind
변수

- SQL에 Bind 변수를 사용하면 Optimizer에서 실행 계획 생성이 완료된 후에 변수값을 후에 호출하여 SQL을 수행하겠다는 것임.
- 따라서 왼쪽 3개 SQL 은 모두 같은 SQL 로 인식하고 첫번째 호출된 SQL만 Hard Parsing 되고 이후에 호출되는 모든 SQL은 모두 Soft Parsing 됨.
- **OLTP에서 사용되는 거의 모든 SQL들은 반드시 Bind 변수 형식으로 작성되어야 함.**
- JDBC 를 사용하는 경우 PreparedStatement 객체로 구현 가능

SQL 공유 강화를 위한 초기화 파라미터 설정



이미 Bind Variable을 사용하지 않은 수많은 SQL들을 Bind 변수를 사용하도록 SQL을 수정하는 것은 많은 노력과 테스트가 필요함.



비슷한 SQL들을 Library Cache에 공유하기 위해서 Optimizer가 동일한 SQL로 인지할 수 있는 기능 필요

ALTER SYSTEM SET CURSOR_SHARING = 'FORCE';

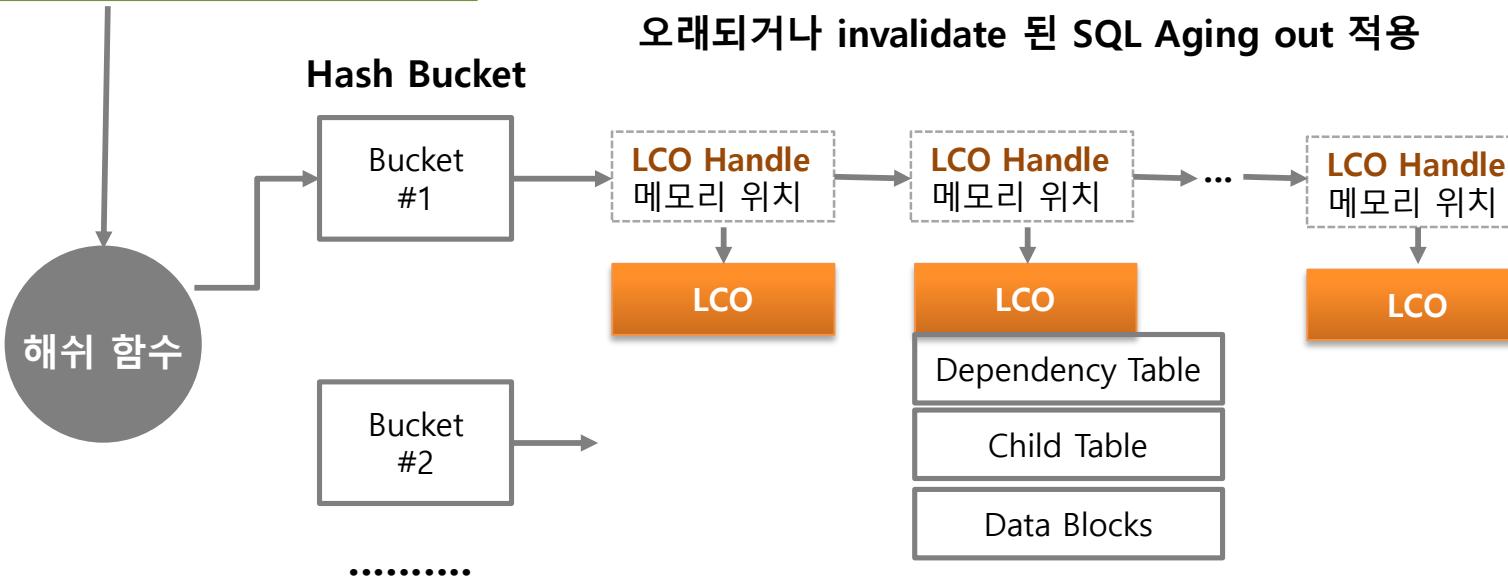
```
Select * from Customer where cust_id = '100000'  
Select * from Customer where cust_id = '200001'  
Select * from Customer where cust_id = '200001'
```

- CURSOR_SHARING='EXACT' 는 SQL이 완벽하게 동일해야 SQL 공유
- **CURSOR_SHARING**을 **FORCE**로 설정함으로 왼쪽의 3개 SQL은 Bind 변수를 사용한 것과 유사하게 SQL을 공유하게 되고 동일한 실행계획을 가짐.
- Oracle 9iR2 이전 버전에서는 버그가 발생할 수 있으므로 이후부터 적용 필요

Library cache 상세 구조

SELECT * FROM CUSTOMER WHERE ID = 'XXXX'

고유 해쉬번호: 1a300xxxxx



LCO(Library Cache Object)는 SQL을 수행하기 위한 여러 오브젝트에 대한 정보 보유
SQL, Package, Procedure, 테이블, 인덱스 등

- SQL에서 참조하는 오브젝트들(테이블, 뷰 등)의 구성
- 참조 권한
- SQL 및 실행 계획 관련 메모리 영역 포인터

만일 ALTER TABLE DROP이나 ALTER INDEX와 같은 DDL 수행되면 LCO에 이에 대한 변경이 반영되어야 하므로 Library Cache내의 해당 LCO들에 대한 Latch/Lock이 수행되어야 함

Library cache의 SQL 다중 버전 관리

스키마 명: TOM

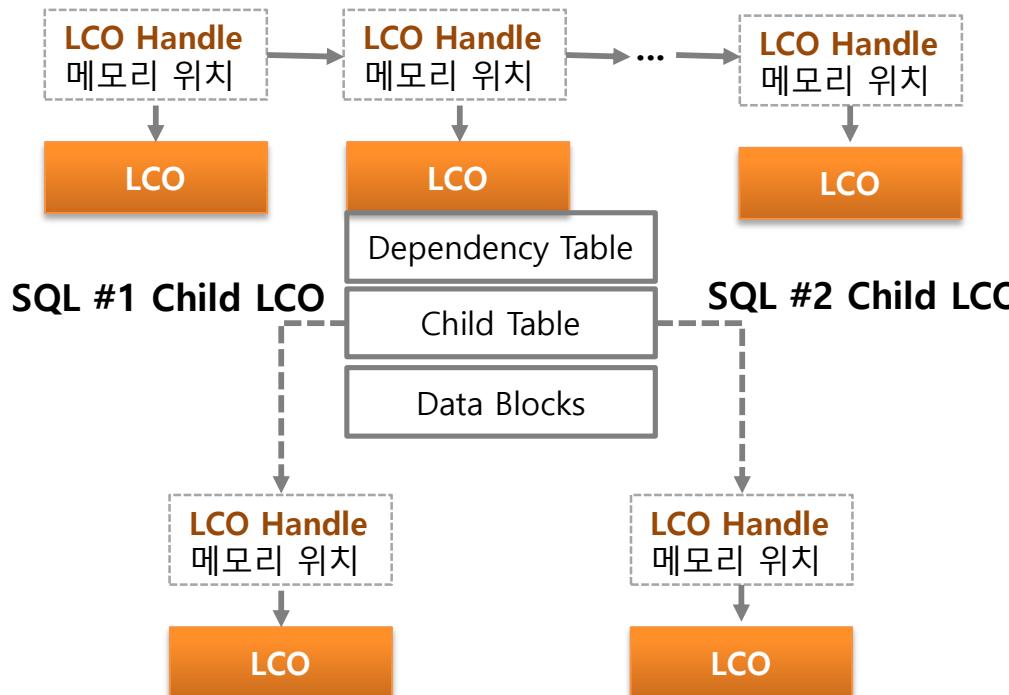
SQL #1: Select * from customer where id=:a1

id가 Number 형

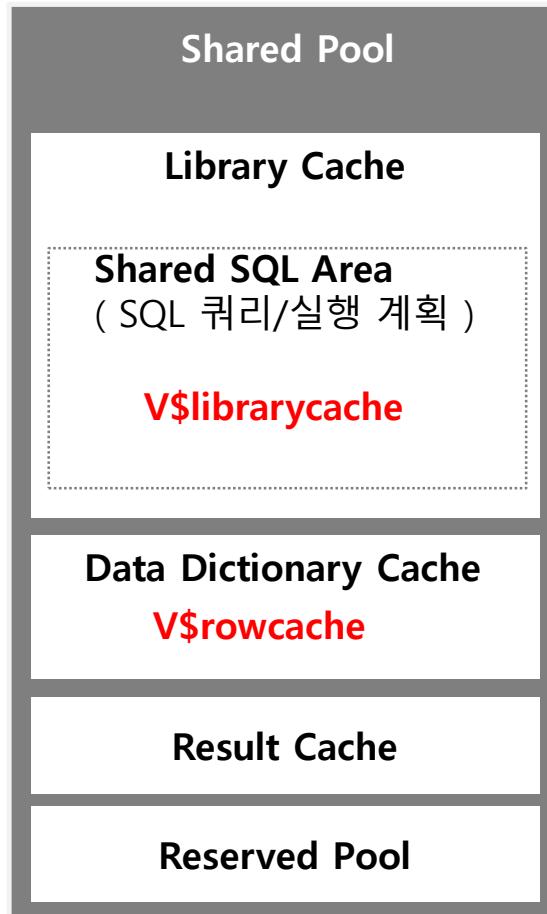
스키마 명: BRAD

SQL #2: Select * from customer where id=:a1

id가 Varchar2 형



Shared Pool 주요 성능 이슈 및 해결 방안



Shared Pool 성능 영향 주요 원인

- Literal SQL
- 매우 큰 PL/SQL 패키지

- DB 연동 Framework 사용 시 반드시 Static SQL 사용 유도.(Dynamic SQL 적용 예외 규정)
- DB 연동 Framework 사용 시 필수 사항 준수(Close() 수행 전 ResultSet 등 먼저 Close할 것 등)
- Shared Pool Size를 충분히 키울 것
- Library Cache, Data Dictionary Cache의 Hit Ratio는 95%~99% 이상을 유지할 수 있도록 노력
- Library Cache, Data Dictionary Cache 크기는 Shared Pool 크기에 따라 자동으로 할당
- 크기가 큰 PL/SQL 패키지는 Age out 되지 않게 pinning 고려
 - ✓ execute dbms_shared_pool.keep("패키지명")

V\$LIBRARYCACHE, V\$ROWCACHE, V\$SHARED_POOL_ADVICE 등의 테이블들을 참조하여 Shared Pool의 Health Check 수행.

오라클의 메모리 크기 자동 관리 기법

9i

- SGA_MAX_SIZE를 통해 DB_CACHE_SIZE, SHARED_POOL등의 SGA Component 크기를 동적으로 변경 가능

10g

ASMM(Automatic Shared Memory Management)

- SGA_TARGET 파라미터를 통해 DB_CACHE_SIZE, SHARED_POOL등의 SGA Component 크기를 동적으로, 그리고 사용량에 따라 자동 변경

11g

AMM(Automatic Memory Management)

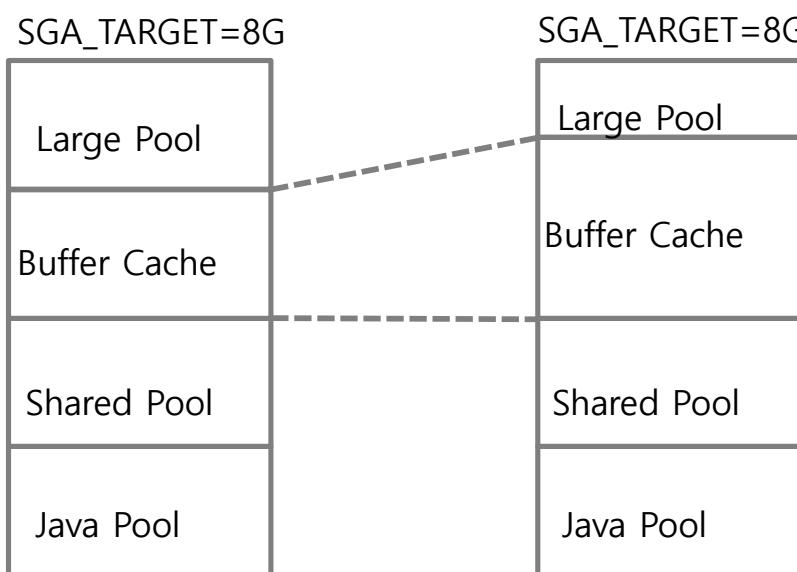
- MEMORY_TARGET, MEMORY_MAX_TARGET 파라미터를 통해 SGA 내 Component들의 메모리 크기 뿐만 아니라 PGA 메모리 크기도 동적/자동 최적 변경

오라클 메모리 관련 주요 초기화 파라미터

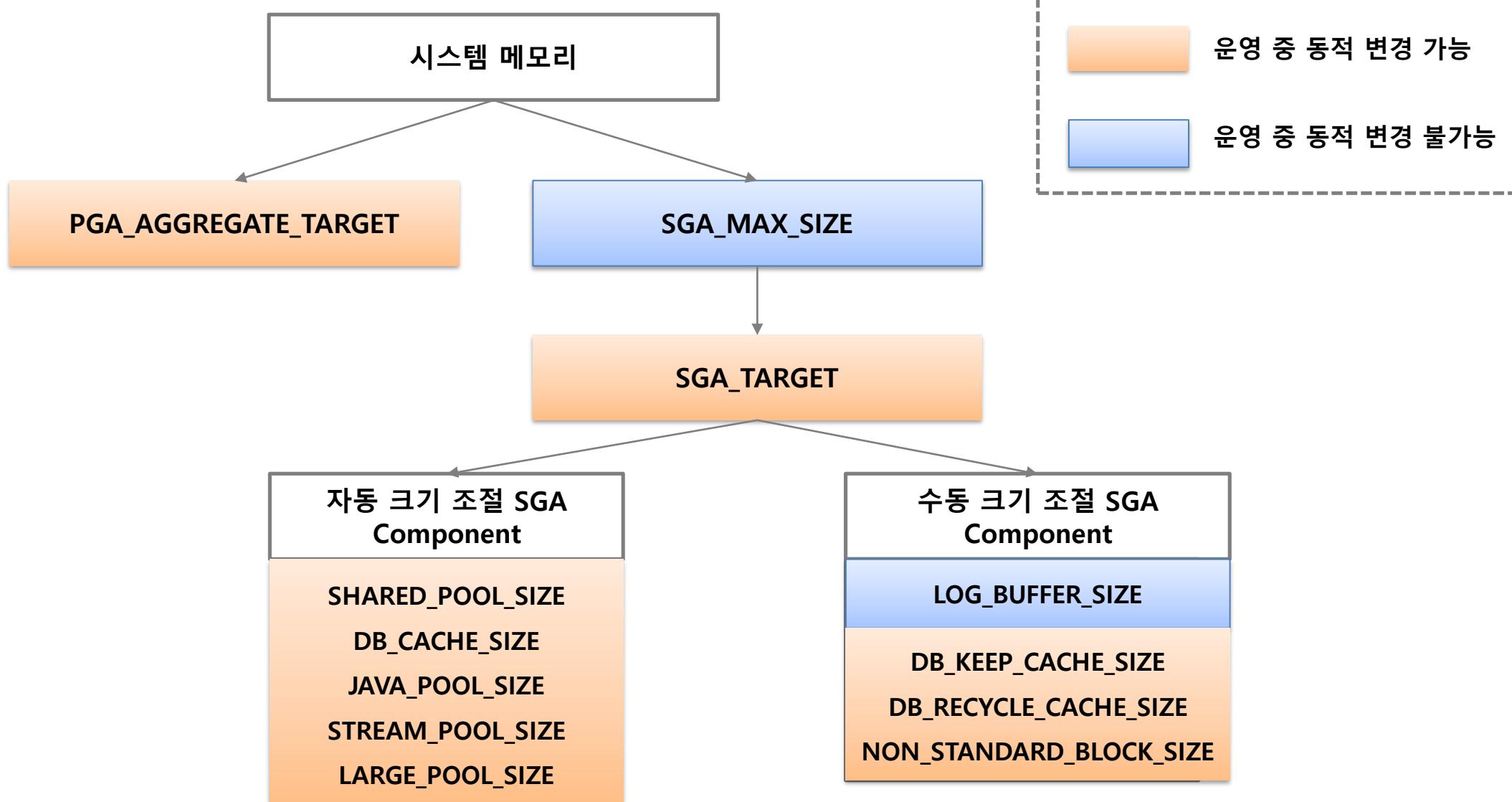
- MEMORY_TARGET
- MEMORY_MAX_TARGET
- SGA_TARGET
- SGA_MAX_SIZE
- PGA_AGGREGATE_TARGET
- PGA_AGGREGATE_LIMIT
- DB_CACHE_SIZE
- SHARED_POOL_SIZE

ASMM(Automatic Shared Memory Management) 개요

- SGA_TARGET과 SGA_MAX_SIZE를 기반으로 ASMM 작동
- SGA_TARGET을 기반으로 SGA 내의 Buffer Cache, Shared Pool, Java Pool, Stream Pool, Large Pool등의 메모리 크기를 운영 중에 변경하고 자동으로 최적 메모리 할당
- SGA_MAX_SIZE를 SGA의 최대 메모리 크기. SGA_TARGET은 SGA_MAX_SIZE내에서 동적으로 변경 가능한 SGA 메모리 크기
- SGA_MAX_SIZE 변경을 반영하려면 **인스턴스를 재 기동해야 함**(정적 초기화 파라미터). SGA_TARGET은 **운영 중에 변경 가능**
- SGA_MAX_SIZE에 설정된 메모리 크기로 인스턴스 최초 기동 시 메모리를 할당함.



ASMM의 주요 파라미터 특성



ASMM 파라미터 설정 방법

AMM Disable 후

MEMORY_MAX_TARGET = 0

MEMORY_TARGET=0

- SGA_TARGET 값을 0 보다 크게 설정하면 자동으로 SHARED_POOL_SIZE , DB_CACHE_SIZE 등이 결정
- SGA_TARGET 값을 0으로 설정하면 ASMM을 사용하지 않으므로 SHARED_POOL_SIZE , DB_CACHE_SIZE등을 수동으로 설정함
- SGA_TARGET 값은 SGA_MAX_SIZE 보다 클 수 없음.
- 만일 SGA_TARGET=8G , SGA_MAX_SIZE=10G 라면 SGA는 최소 8G부터 할당되어 최대 10G 까지 설정 가능함.
- 메모리 크기를 증가시킬 시 Granule 단위로 증가함. SGA 크기가 1GB 이상이면 Granule 단위 크기는 16MB
- ASMM 상에서 SHARED_POOL_SIZE=2G 와 같이 명확하게 값을 설정 할 수 있지만, 이 경우 지정된 값은 최소한으로 할당되는 값이며 증가 후 다시 줄여도 시간을 가지고 값이 설정 됨.

가상 메모리 관리 – 페이징(Paging)

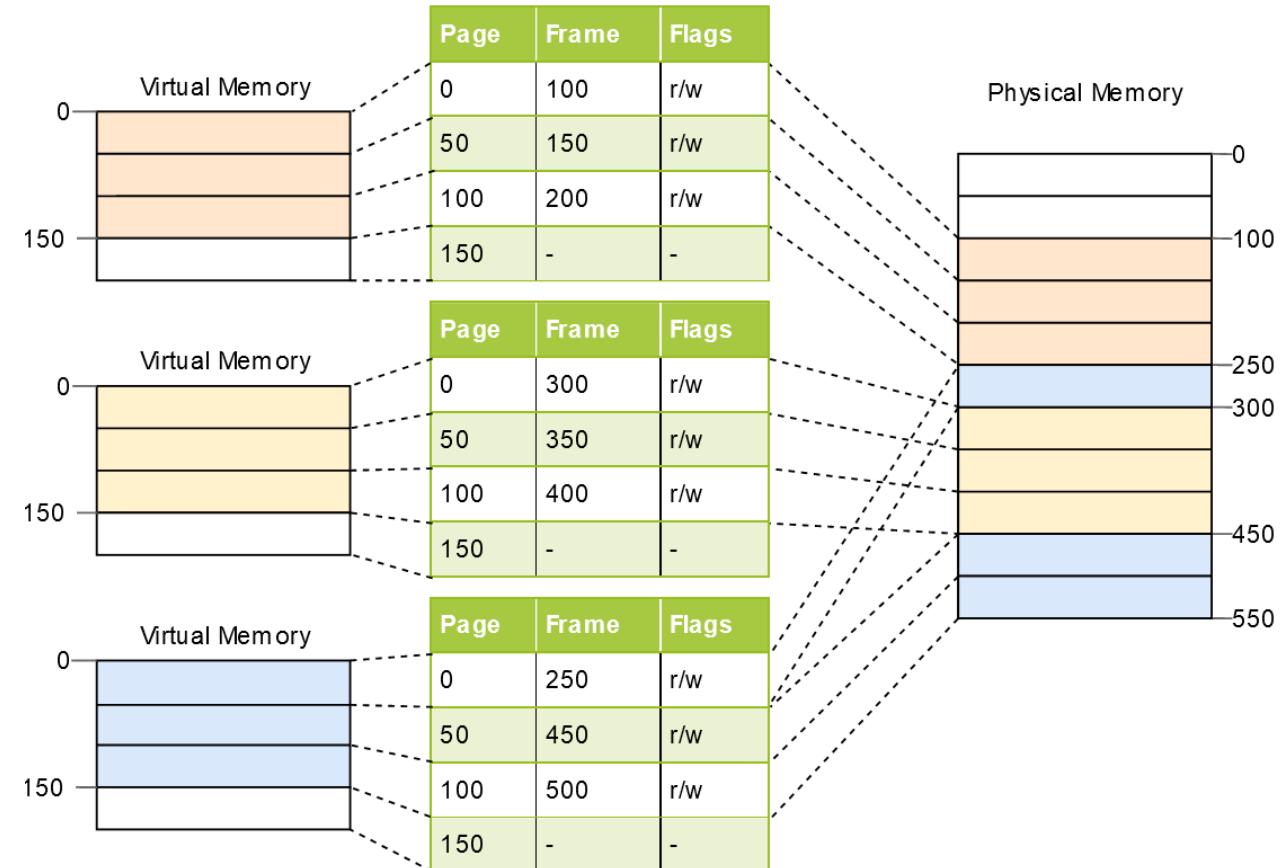
가상 메모리 관리:

- 물리 메모리 크기의 한계를 극복하기 위함. 물리 메모리 보다 더 큰 프로세스를 수행 하려면 가상 메모리가 필요. 프로세스가 수행할 때 필요한 부분만 메모리에 적재.
- 가상 메모리 주소를 물리적인 메모리 주소가 추상화 되었고 이를 변환하여 물리 주소를 접근.
- 메모리 단편화 등의 문제 발생

페이징(Paging)

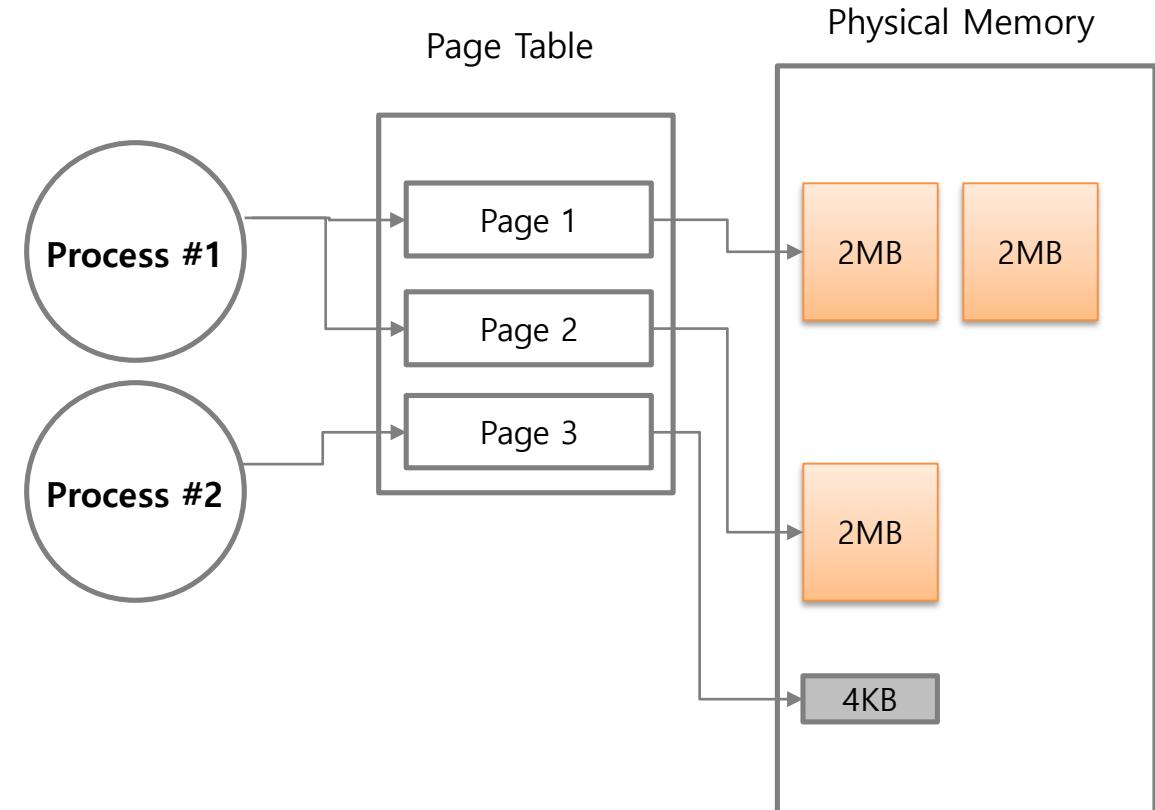
- 메모리 단편화 등의 문제를 해결하기 위해서 가상 메모리와 물리 메모리 공간을 연속된 작은 블록으로 나눔. 가상 메모리 공간의 블록은 페이지(Page)라고 함.
- Page Table을 통해 페이지와 Physical 메모리를 매핑
- 리눅스는 기본 Page 크기가 4KB임.
- 물리적인 메모리의 크기가 수십 GB ~ 수 TB 이상 커지면서 Page Table 관리를 위한 리소스 비용이 커짐

Page Table



리눅스 메모리 관리 - Huge Page

- 리눅스의 기본 메모리 페이지(page) 크기는 4KB. 하지만 이를 2MB(또는 1GB) 단위로 사용하는 것을 Huge Page라고 함.
- 메모리의 크기가 커질 수록 메모리 페이지 관리를 위한 Overhead가 증가되고 자원 사용률이 늘어남
- Huge Page는 상대적으로 적은 수의 관리 정보를 가지고 메모리 페이지를 관리. Page table 크기가 줄어서 좀 더 많은 물리적 메모리를 사용할 수 있음
- Huge Page는 일반 Page와는 다르게 메모리 Swap 되지 않음. (Oracle SGA는 최대한 메모리 Swap 되지 않아야 함)



Oracle DB에 Huge page 설정하고 SGA 활용하기

1. 현재 Huge page 정보 조회
 - Huge page 개수 조회
2. hugepages_settings.sh 수행하여 적합한 huge page 개수 추정
 - sysctl -w vm.nr_hugepages= 값 설정
 - /etc/sysctl.conf 파일에서 vm.nr_hugepages값을 위 스크립트 실행 결과 값으로 설정
 - Huge page 정보 재 조회
3. memlock (고정 메모리) 크기 수정. root로 수정
 - Oracle user로 로그인 하여 ulimit -l 명령어로 memlock 설정 확인
 - 최소 Hugepages 개수 * 2048KB 만큼 설정 이상 (또는 전체 RAM 메모리의 70% 이상 설정.)
 - /etc/security/limits.conf 파일 수정 후 Reboot
4. HugePages는 반드시 SGA보다 커야 함. 오라클 11.2.0.2 이하에서는 HugePages가 SGA보다 작을 경우 아예 HugePages를 SGA가 사용하지 못했음. 11.2.0.3 이상 부터 HugePages가 SGA보다 작을 경우 모자라는 영역은 Normal Pages를 사용함.
5. 반드시 AMM이 아님을 확인하고 DB 초기화 파라미터 중 USE_LARGE_PAGES=only로 설정 후 재 기동.

Redo Log Buffer와 Online Redo Log 파일

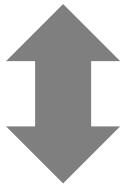
DML

Insert

Update

Delete

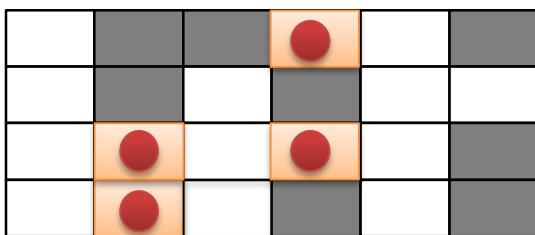
Commit 된 모든 Transaction은 완벽하게 저장되어야 하며 어떠한 상황에서도 완벽하게 복구 되어야 한다.



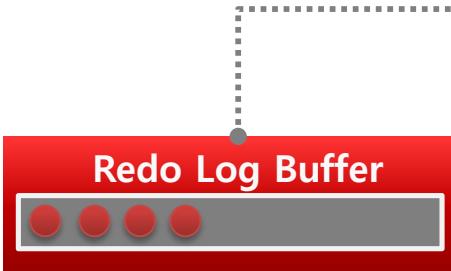
대량의 데이터 변경 작업도 빠르게 수행되어야 한다.

DML Transaction의 Commit 과 Rollback

Buffer Cache



Redo Log Buffer의 역할



LOG_BUFFER 파라미터를 사용하여 Buffer 크기를 조절

수십MB 정도로 설정

Redo Log File I/O 성능 향상을 위해 사용

- 모든 데이터의 변경정보를 메모리에 임시 보관
- DML 처리 시 Redo Log File I/O 성능 향상을 위해서 Redo Log Buffer 가 사용되며 Redo Log Buffer에 있는 데이터는 바로 Redo Log File 에 기록되지 않고 특정한 조건을 만족할 경우에만 일시에 Write 됨

Redo Log Buffer에서 Online Log File write 조건

- 사용자가 Commit 할 경우
- Redo log buffer 가 1/3 정도 Full 된 경우
- Buffer cache의 free 공간이 없어서 변경 데이터를 Flush 해야 할 경우
Log Writer 가 redo log buffer 도 함께 redo log 에 Write 함.

Redo Log Buffer 의 활용

UPDATE CUSTOMER SET SIDO = '서울 특별시' WHERE SIDO = '서울'

- 1 Data에 변경사항 DML 발생
(Insert , Update , Delete)
- 2 Redo Log Buffer 에 변경 전/후
값 전달
- 3 LGWR 을 통해 Redo Log File에 Write



DML 등으로 데이터의 변경이 발생
할 경우 Buffer cache의 해당 Block
내 값을 변경한 뒤에 바로 Redo
Log Buffer에 변경전/후 값을 전달

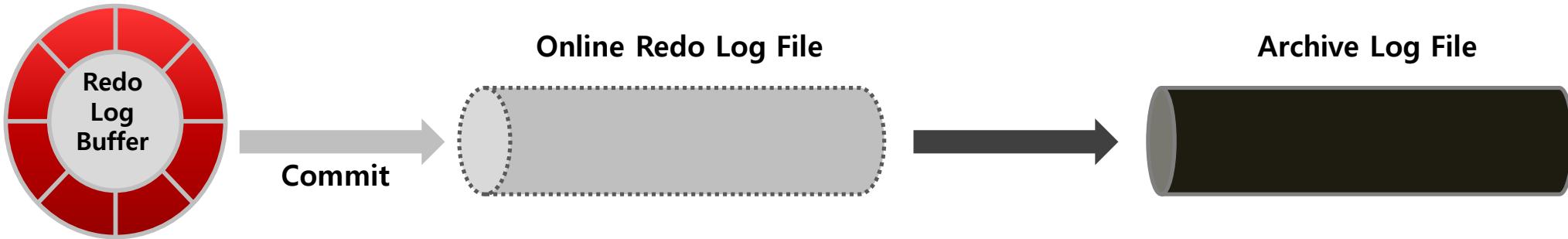
변경 전/후 값 : 서울->서울 특별시

- 사용자가 Commit 할 경우
- Redo log buffer 가 1/3 정도 Full 된 경우
- Buffer cache의 free 공간이 없어서 변경 데이터
를 Flush 해야 할 경우 Log Writer 가 redo log
buffer 도 함께 redo log 에 Write 함.

온라인 Redo Log File의 필요성

Redo Log File의 필요성

Commit 된 모든 Transaction은 완벽하게 저장되어야 하며 어떠한 상황에서도 완벽하게 복구 되어야 한다



- Commit 되지 않은 Transaction 은 수행 성능 향상을 위해 Redo Log File에 바로 Write되지 않고 Redo Log Buffer 에 먼저 기록됨

- 테이블 데이터의 모든 변경사항이 기록됨
- 많은 트랙잭션들이 Commit 시 바로 Datafile에 변경을 기록하게 되면 Write I/O 가 과다 발생하여 수행 성능 저하되므로 Online Redo Log 에 일차 기록함.
- Commit 발생할 경우 기록되는 일차 파일이므로 데이터 복구를 위해 매우 중요 함.** 반드시 그룹으로 다중화 되어야 함.

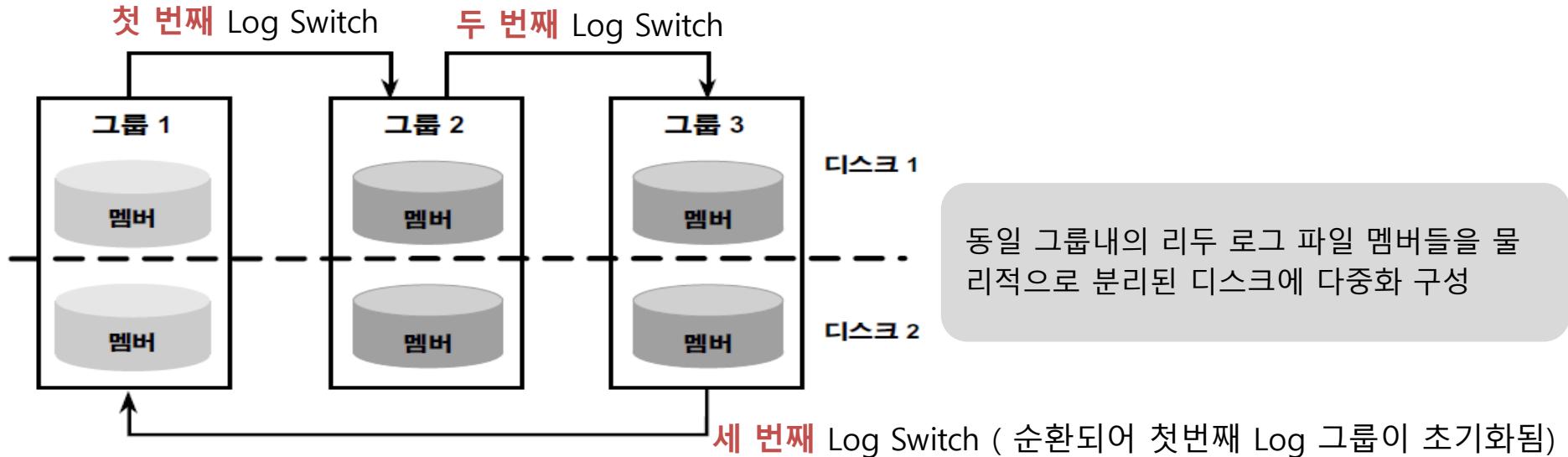
- Redo Log File이 다 차게 되면 현 Redo Log File을 별도의 Archive Log File로 생성하고 Log switch
- DB 복구 시 매우 중요한 파일이므로 별도의 Tape 등의 Storage로 백업 수행

Redo Log File 내용

Redo Log File 내용

- 트랜잭션의 시작되었음을 알리는 구분자
- 트랜잭션 고유명
- DML 되고 있는 Object명 (테이블 명)
- 트랜잭션 Before Image (변경 전 데이터)
- 트랜잭션 After Image (변경 후 데이터)
- Commit 이 되었음을 알리는 구분자

온라인 리두 로그 파일 구조



온라인 리두 로그 파일 그룹

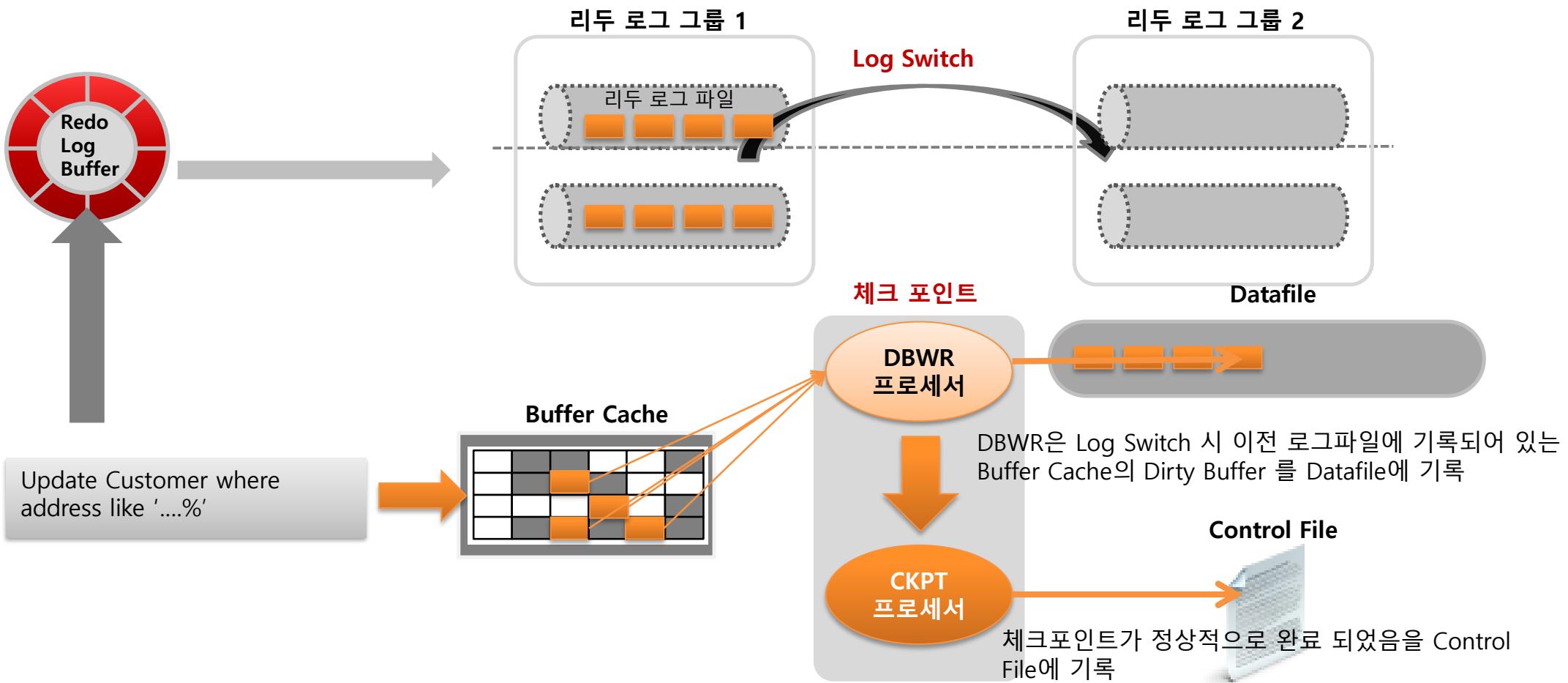
- 온라인 리두 로그 파일의 동일한 복사본 모음
- LGWR 프로세스는 그룹에 있는 모든 온라인 리두 로그 파일에 동일한 정보를 동시에 기록
- 데이터베이스의 정상적인 운영을 위해 **최소 두 개의 온라인 리두 로그 파일 그룹이 필요**

온라인 리두 로그 파일 멤버

- 그룹에 있는 각 온라인 리두 로그 파일
- 그룹에 있는 각 멤버는 동일한 로그 시퀀스 번호 및 크기를 가짐
- 로그 시퀀스 번호는 오라클이 온라인 리두 로그 파일을 고유하게 식별하기 위함. 로그 시퀀스 번호는 Control File과 데이터파일 헤더에 기록

온라인 리두 로그 파일 작동 방법

- 온라인 리두 로그 파일은 순환 방식으로 사용
- 온라인 리두 로그 파일이 가득 차면 LGWR을 다음 로그 그룹으로 이동
 - 로그 스위치를 호출 하며 체크포인트 작업도 발생
 - Control File에 정보를 기록

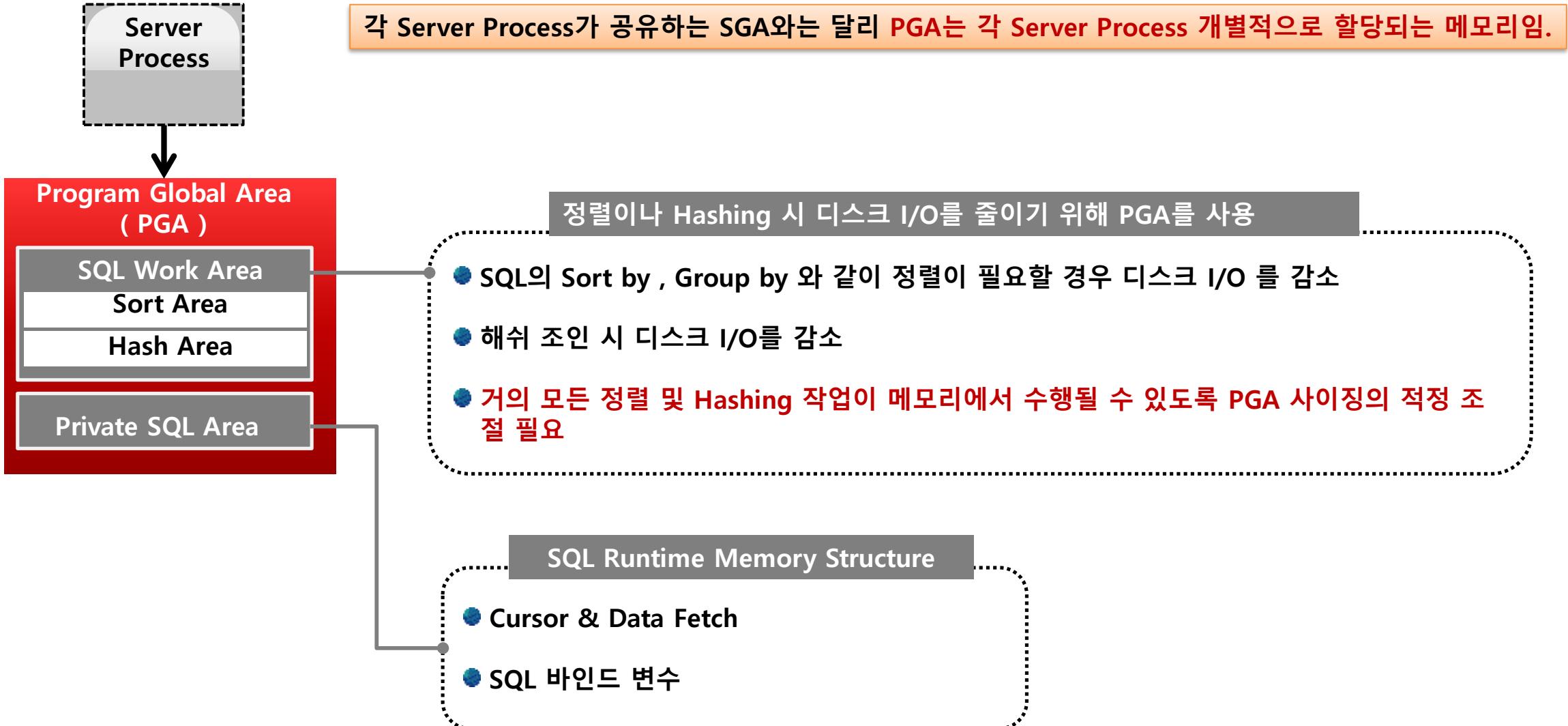


온라인 리두 로그 그룹과 파일에 대한 주요 Data Dictionary

Data Dictionary	설명
V\$THREAD	온라인 리두 로그 그룹의 수, 현재 로그 그룹, Sequence Number 등을 조회
V\$LOG	리두 그룹과 멤버에 대한 정보
V\$LOGFILE	온라인 리두 로그 파일에 대한 상태와 위치

PGA와 AMM 이해

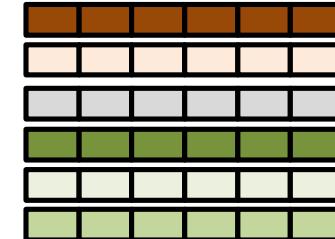
PGA (Program Global Area)



PGA를 활용한 SQL 정렬 메커니즘

SELECT * FROM CUSTOMER ORDER BY CUSTOMER_NAME DESC

정렬 대상 데이터



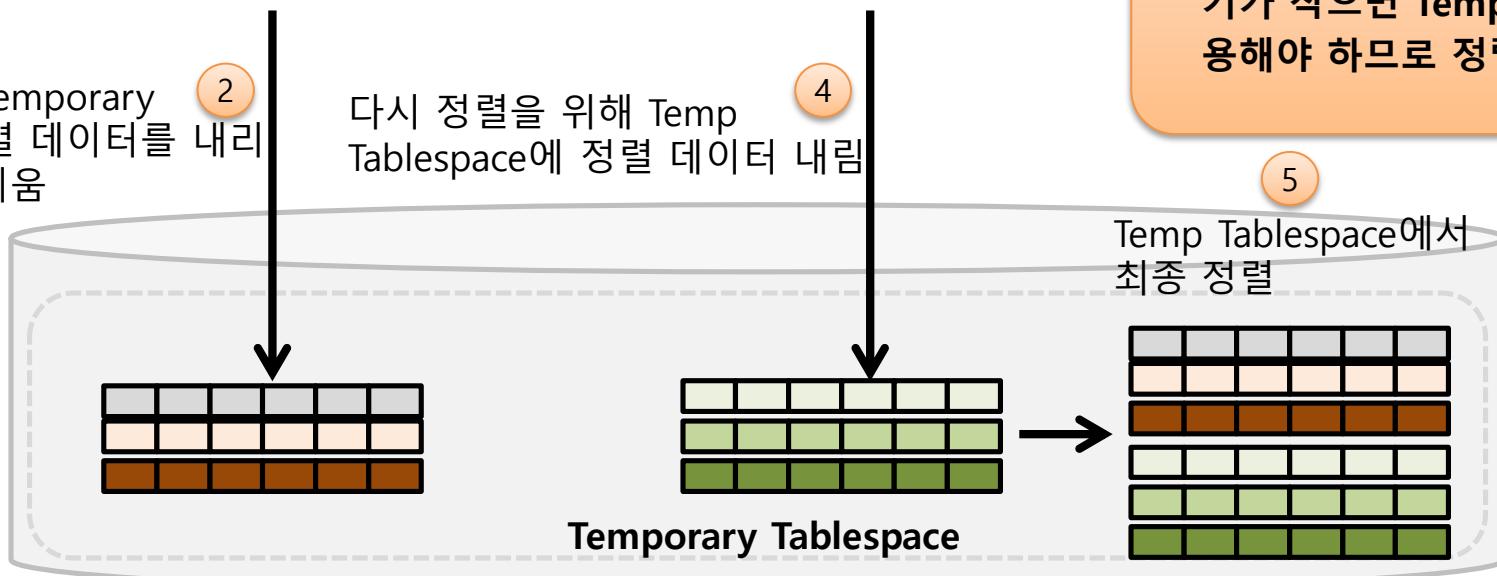
- 1 PGA 메모리에서 정렬 가능한 데이터량만 큼만 1차로 정렬(PGA 사이즈 점차 증가)
3 추가 데이터를 다시 PGA에서 2차 정렬



추가 정렬을 위해 Temporary Tablespace 1차 정렬 데이터를 내리고 PGA 메모리를 비움

다시 정렬을 위해 Temp Tablespace에 정렬 데이터 내림

- 정렬해야 할 대상이 작으면 PGA 메모리 내에서 정렬이 가능하므로 정렬 속도가 빠름
- 정렬해야 할 대상이 크고 PGA의 크기가 작으면 Temp Tablespace 를 활용해야 하므로 정렬 속도가 느려짐



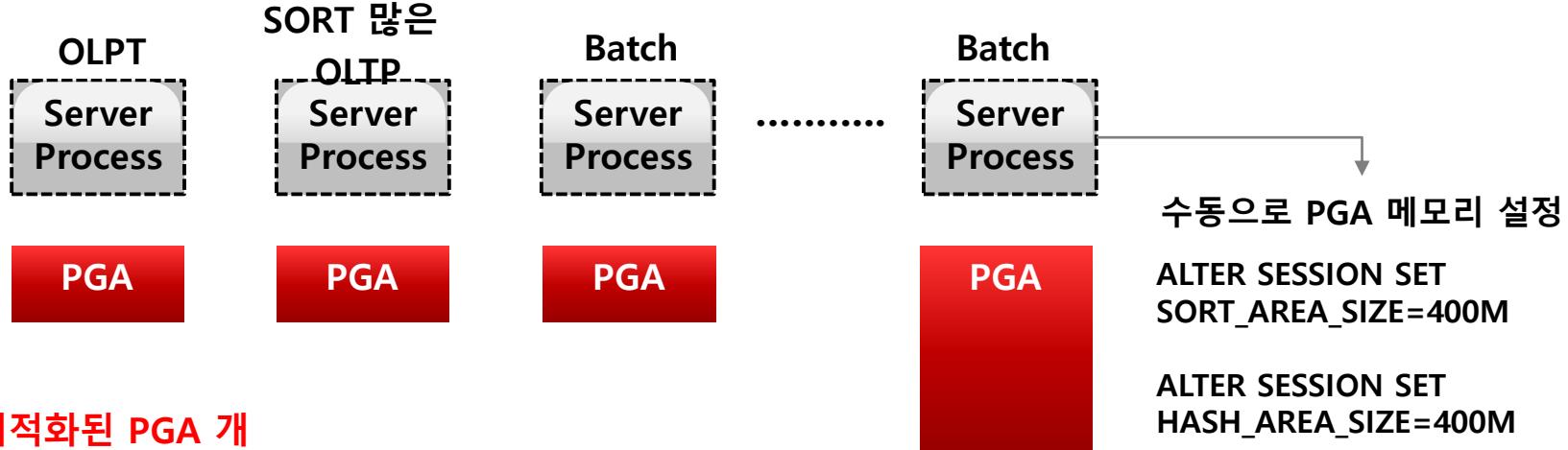
PGA 메모리 할당 방법

DBA가 Workload를 감안하여 PGA의 개별 메모리 구성 요소를 직접 설정

SORT_AREA_SIZE=10M

SORT_AREA_RETAINED_SIZE=5M

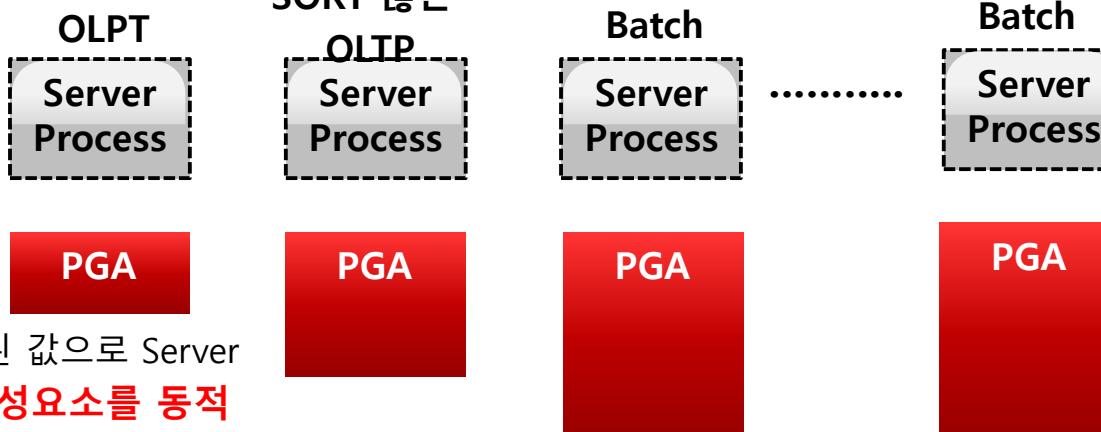
HASH_AREA_SIZE=10M



다양한 SQL/업무 유형에 따라 **최적화된 PGA 개별 메모리 구성 요소를 설정 어려움**

오라클이 동적으로 PGA 개별 메모리 구성 요소를 최적 동적 할당

WORKAREA_SIZE_POLICY = AUTO
PGA_AGGREGATE_TARGET=4G



PGA_AGGREGATE_TARGET에 설정된 값으로 Server Process들이 **PGA 개별 메모리 구성요소를 동적으로 할당 받을 수 있도록 최적 조정**

현재 가용 가능한 메모리 양, 세션 별로 사용중인 SORT 부하 등을 감안하여 자동 할당

PGA 할당 정책 및 크기 설정을 위한 파라미터

9i 이전에는 PGA크기 조절을 위한 세부 파라미터들을 모두 사용자가 정해야 하므로 메모리 최적화에 어려움이 있었으나 9i 부터는 자동으로 PGA를 최적으로 할당하기 위한 파라미터가 적용됨

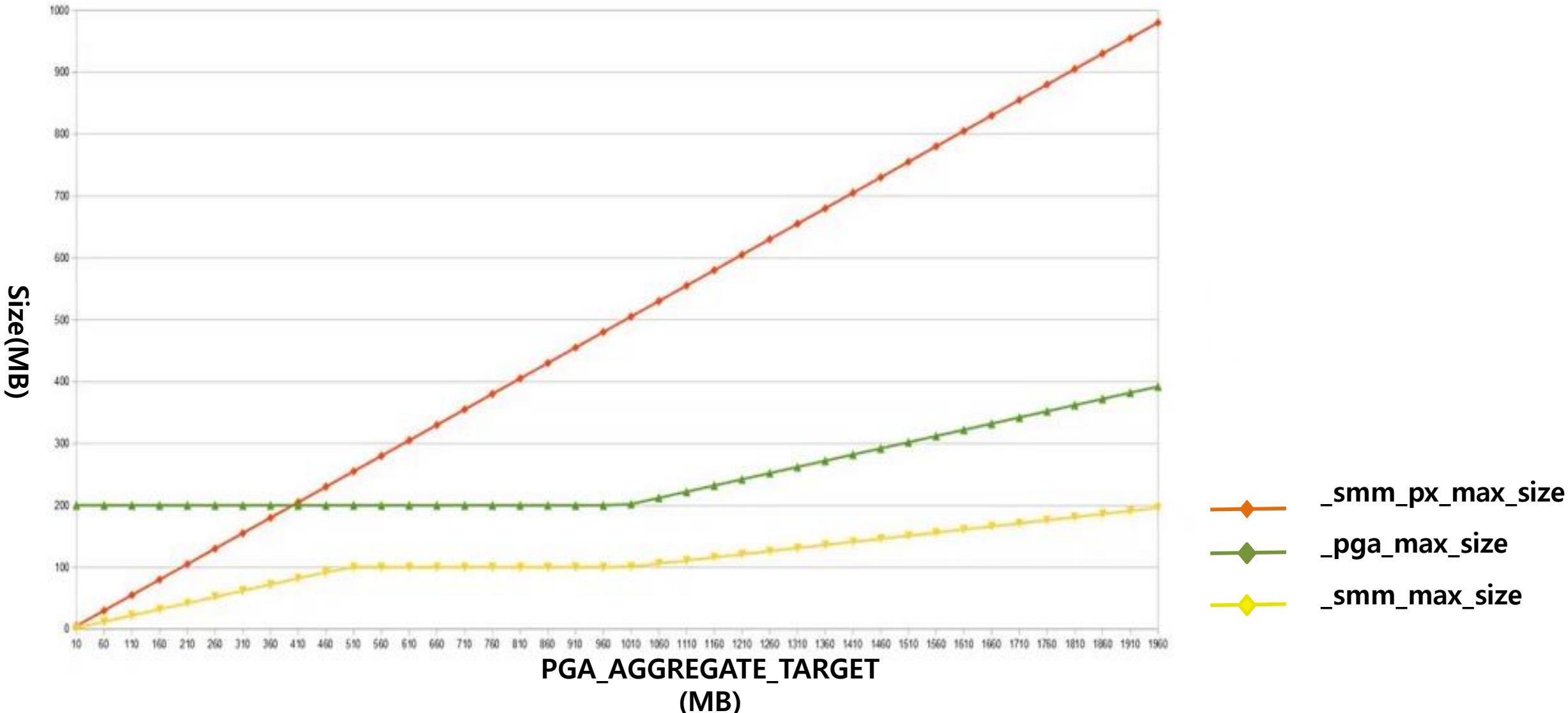
DB Version	파라미터 명	파라미터 설명
9i 이전	SORT_AREA_SIZE	Sort 및 Group by 를 위해 PGA에서 최대 할당 가능한 메모리
	SORT_AREA_RETAINED_SIZE	PGA에서 정렬작업이 완료된 후에도 Server Process가 계속 가지고 있어야 되는 기본 정렬 메모리 공간
	HASH_AREA_SIZE	Hash Join을 위해 PGA에서 최대 할당 가능한 메모리
9i 이후	WORKAREA_SIZE_POLICY	AUTO로 설정할 경우 오라클이 PGA_AGGREGATE_TARGET 에 기술된 메모리 내에서 최적화된 정렬 및 Hashing등을 위한 공간을 동적으로 자동 할당. (가장 바람직한 방식) MANUAL로 설정할 경우 9i 이전과 마찬가지로 정렬과 Hashing을 위한 별도 파라미터 지정 값으로 메모리 할당 (SORT_AREA_SIZE 등을 활용)
	PGA_AGGREGATE_TARGET	WORKAREA_SIZE_POLICY가 AUTO로 지정될 경우 자동으로 확장하는 PGA 영역 (Soft Limit)
12c	PGA_AGGREGATE_LIMIT	PGA 영역이 최대로 확장 가능한 메모리 크기(Hard Limit)

PGA 크기를 결정하는 Hidden Parameter

파라미터 명	설명
_pga_max_size	한 개의 서버 프로세스가 최대 사용 가능한 PGA Workarea 메모리
_smm_max_size	한 개의 서버 프로세스가 단일 Operation에서 최대 사용 가능한 Workarea 메모리. Parallel operation에서는 하나의 PQ가 단일 Operation에서 사용 가능한 메모리. _pga_max_size의 50%
_smm_px_max_size	(DOP 5 이상) PQ 사용 시 한개의 세션이 최대 사용 가능한 Workarea메모리. PGA_AGGREGATE_TARGET의 50%

PGA 크기를 결정하는 Hidden Parameter

<https://makeoracle.wordpress.com/2016/06/09/pga-how-oracle-allocate-memory-to-the-process/>



대용량 데이터 Parallel Query의 수동 PGA 설정 시 유의점

- 대용량 데이터를 정렬, Group by, Hashing 할 경우 수행 속도를 향상 시키기 위해 Session 레벨에서 PGA 관련 메모리 파라미터를 수동으로 설정
- 이 경우 Parallel Query 사용에 극도로 유의하여야 함.

ALTER SESSION SET WORKAREA_SIZE_POLICY=MANUAL;



Session 레벨에서 PGA 메모리 파라미터를 수동 조정

ALTER SESSION SET SORT_AREA_SIZE=800M; /*+ 정렬용 800M */

ALTER SESSION SET SORT_AREA_RETAINED_SIZE=400M; /*+ 정렬이 완료된 후에도 400M 그대로 유지 */

ALTER SESSION SET HASH_AREA_SIZE=800M; /*+ HASHING용 800M */

```
SELECT /*+ NO_PARALLEL(A) */ SVC_CNTR_NO, PRDC_CD, EFCT_STRT_DT, EFCT_END_DT, PRDC_HIST_SRI_NO  
FROM MIG_SVC_CNTR_PRDC_HIST A , PRDC_INFO B WHERE A.PRDC_CD = B.PRDC_CD  
GROUP BY SVC_CNTR_NO, PRDC_CD, EFCT_STRT_DT, EFCT_END_DT, PRDC_HIST_SRI_NO  
HAVING COUNT( * ) > 1 ;
```

만일 해당 SQL을 Parallel Query로 수행한다면 ?

ALTER SESSION SET WORKAREA_SIZE_POLICY=AUTO;



다시 PGA 메모리 자동할당으로 설정

- SQL 수행을 위해 해당 Session에 할당된 PGA 메모리는 정렬용 800M + Hashing용 800M로 최대 1.6GB까지 가능
- 만일 Parallel SQL 을 /*+ Parallel (A 4) Parallel (B 4) */ Hash Join Group by와 같이 사용할 경우 각 Parallel Server가 최대 1.6G 까지 할당 가능하고 8개의 parallel child server가 사용될 수 있으므로 $1.6 * 8 = 12.8G$ 까지 메모리가 사용될 수 있음. 이 경우 시스템 가용 메모리 상황에 따라서 심각한 장애를 유발할 수도 있으므로 Parallel Query 사용시에는 반드시 유의해야 함.

PGA_AGGREGATE_TARGET 크기 설정

시스템 유형	크기 설정(Baseline) by Oracle Tuning guide
OLTP	<p>PGA_AGGREGATE_TARGET: 가용 메모리(전체 Physical Memory * 0.8) * 0.2</p> <p>SGA 메모리: 가용 메모리(전체 Physical Memory * 0.8) * 0.8</p>
DSS	<p>PGA_AGGREGATE_TARGET: 가용 메모리(전체 Physical Memory * 0.8) * 0.5</p> <p>SGA 메모리: 가용 메모리(전체 Physical Memory * 0.8) * 0.5</p>



- 전체 Physical Memory의 크기
 - 세부적인 추가 Workload 사항
 - V\$PGA_TARGET_ADVICE,
- DBA_HIST_PGA_TARGET_ADVICE(AWR) 권고 사항

V\$PGA_TARGET_ADVICE

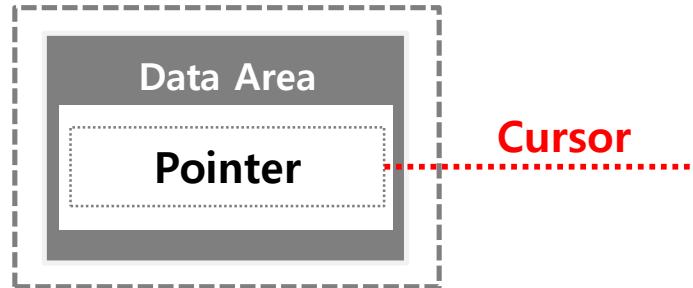
컬럼명	설명
PGA_TARGET_FOR_ESTIMATE	예측 PGA_AGGREGATE_TARGET 크기
PGA_TARGET_FACTOR	현재 설정된 PGA_AGGREGATE_TARGET 대비 예측 PGA_AGGREGATE_TARGET 크기 비율
ADVICE_STATUS	ADVICE 설정 셋팅 여부
BYTES_PROCESSED	WORK AREA 내에서 처리된 총 BYTES수
ESTD_TIME	총 BYTES 처리 시간
ESTD_EXTRA_BYTES_RW	추가적으로 처리되어야 할 총 BYTES 수(One-pass, Multi-pass 시)
ESTD_PGA_CACHE_HIT_PERCENTAGE	PGA Cache Hit percent. BYTES_PROCESSED/(BYTES_PROCESSED+ESTD_EXTRA_BYTES_RW)
ESTD_OVERALLOC_COUNT	PGA Memory over-allocation 횟수

Cursor의 이해

Cursor란 서버 프로세스내의 Private SQL area를 가리키는 일종의 Handle(Pointer)를 지칭. Cursor는 클라이언트 프로세스가 서버의 데이터를 가져오기 위해 가지는 일종의 Pointer임

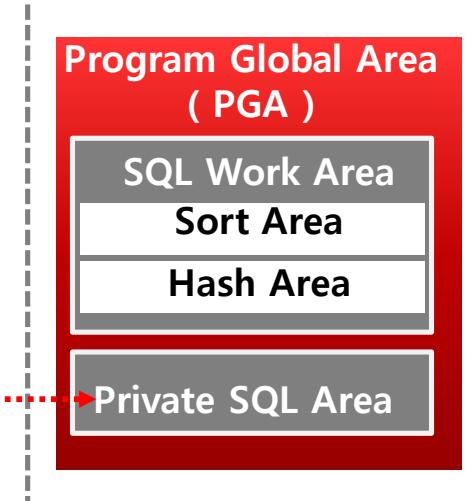
SELECT * FROM CUSTOMERS WHERE ADDRESS='서울'

클라이언트 프로세스



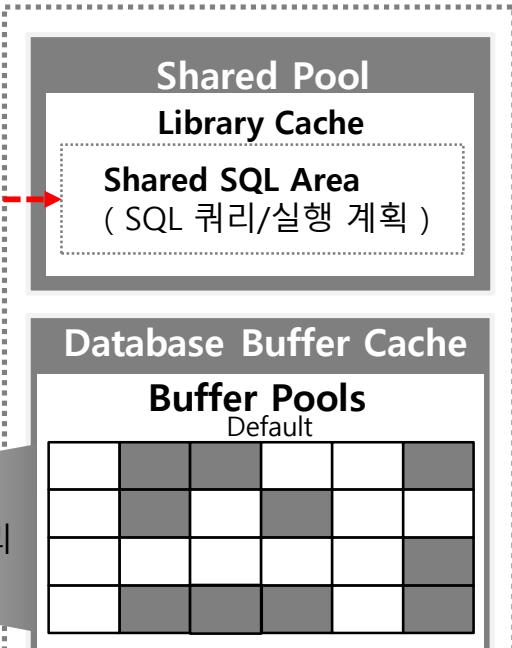
OCI, JDBC Driver 기반 Application

서버 프로세스



데이터 조회/처리

Database

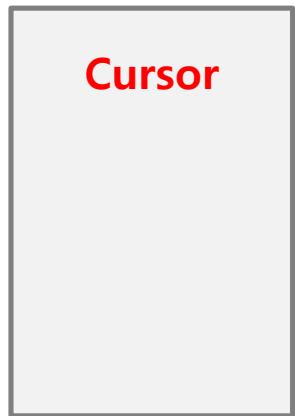


1. 클라이언트는 Cursor를 통해 서버 프로세스에게 데이터 조회/처리를 요청
2. 서버프로세스는 Buffer Cache나 Storage에서 조회/처리된 데이터를 Private SQL Area에 저장 후에 해당 데이터를 클라이언트에게 반환
3. 클라이언트는 요청한 SQL의 데이터가 완전히 반환 될 때까지 계속해서 Fetch를 요구

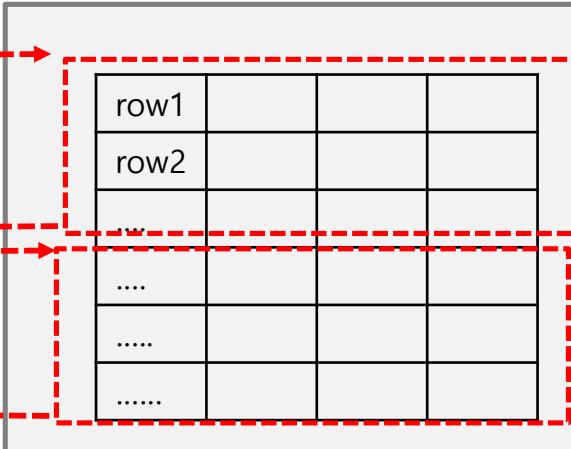
Cursor 동작 메커니즘

SELECT * FROM CUSTOMERS WHERE ADDRESS='서울'

클라이언트 프로세스



서버 프로세스



Fetch
Fetch

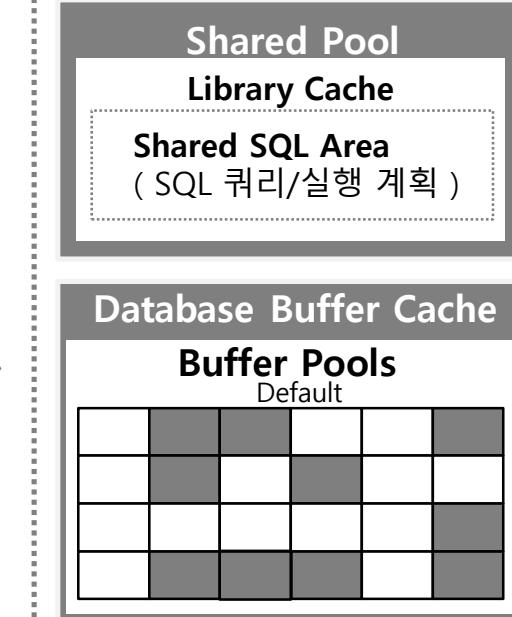
PL/SQL

1. Declare Cursor
2. Open Cursor
3. Fetch Cursor
4. Close Cursor

JDBC

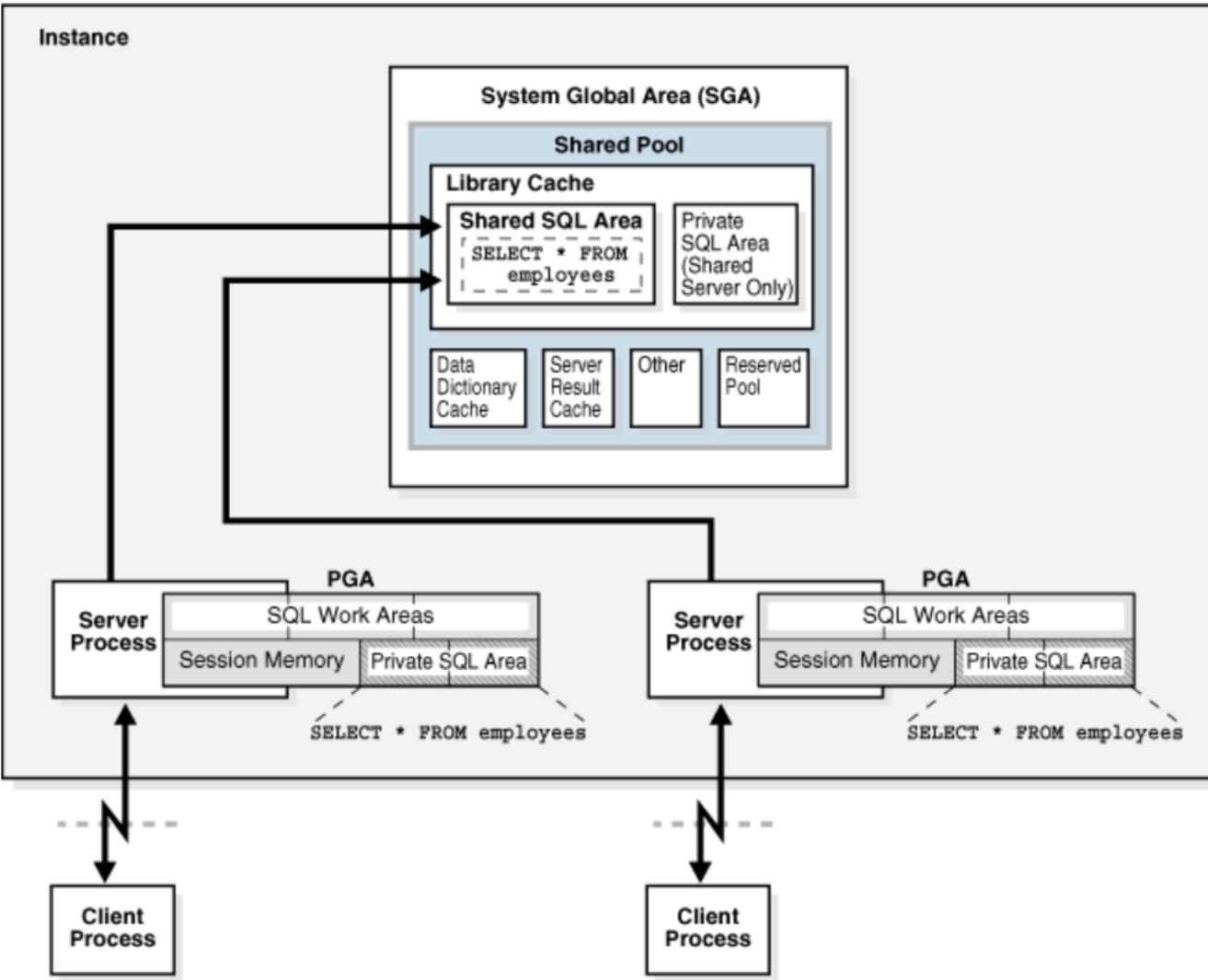
1. Create PreparedStatement
2. ResultSet = Execute Query
3. ResultSet next()
4. ResultSet close()

Database



Private SQL Area와 Shared Pool의 Shared SQL Area

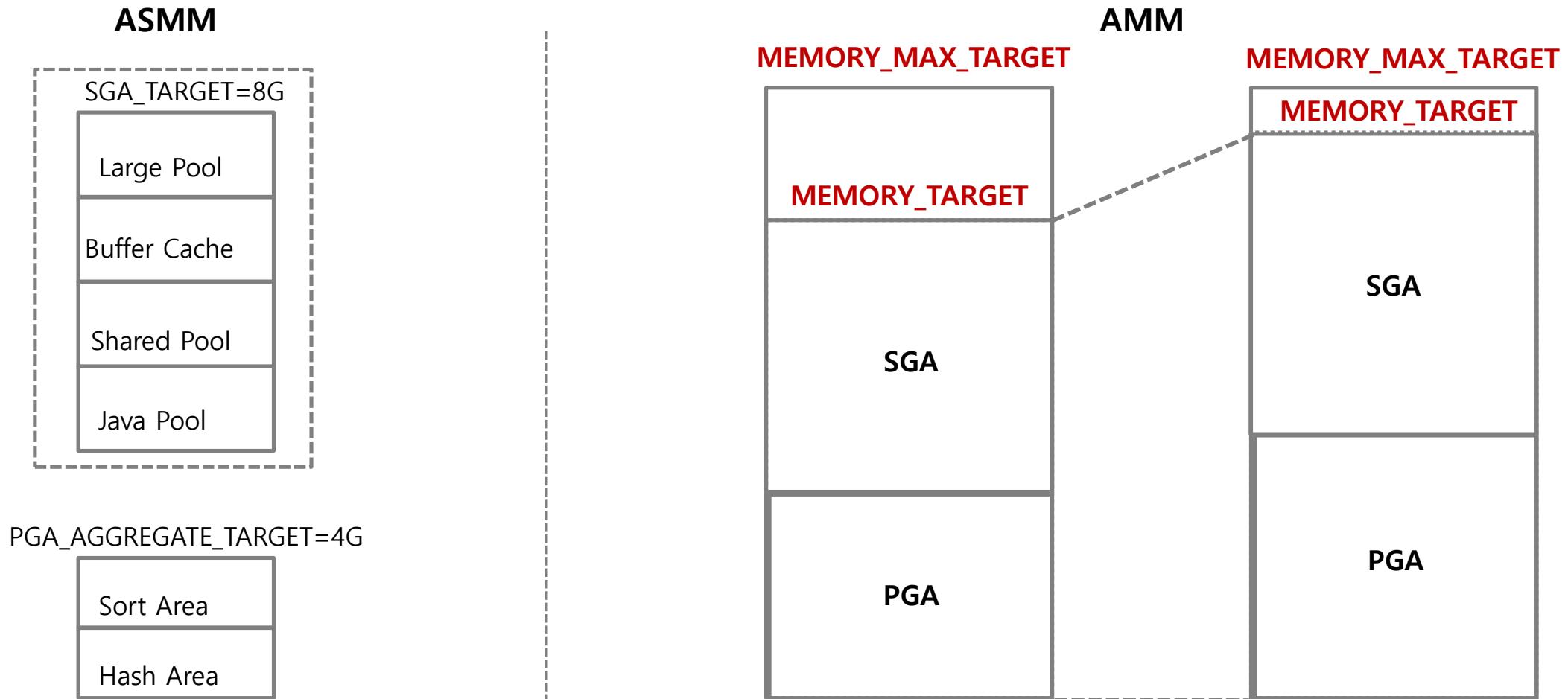
Figure 14-10 Private SQL Areas and Shared SQL Area



오라클 DB 아키텍처 문서에서

AMM(Automatic Memory Management) 개요

10g에서 소개된 ASMM(Automatic Shared Memory Management)는 SGA내의 Component들을 자동으로 동적 최적 크기 조정을 가능하게 하였고, 11g부터 소개된 AMM은 SGA뿐만 아니라 PGA 크기도 자동으로 동적 최적 크기 조정을 가능하게 함.



AMM 설정 방법 및 이슈

MEMORY_TARGET

SGA와 PGA를 동적으로 조정하여 할당할 수 있도록 지정된 메모리 영역.
MEMORY_MAX_TARGET 값을 넘을 수 없다.

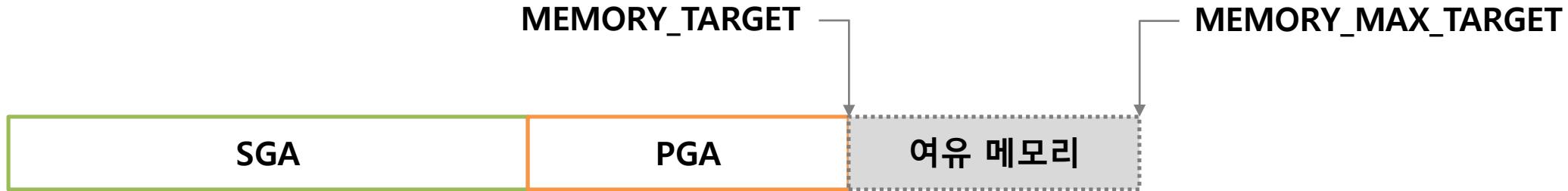
MEMORY_MAX_TARGET

MEMORY_TARGET값이 최대로 할당 될 수 있는 범위

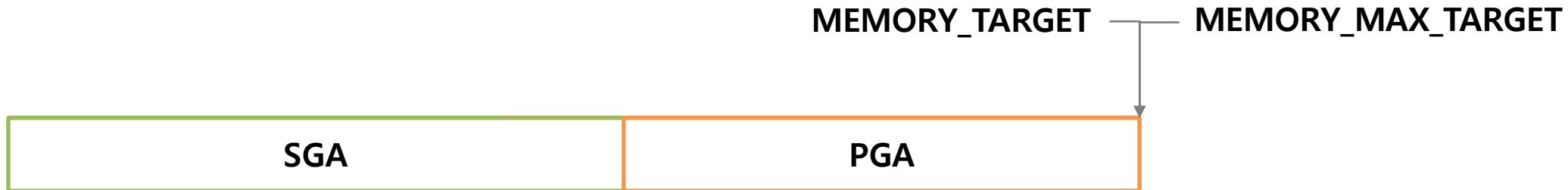
- MEMORY_TARGET과 MEMORY_MAX_TARGET을 >0로 설정하여 운용
 - SGA_TARGET, PGA_AGGREGATE_TARGET은 0으로 설정 권장
- 적정한 MEMORY_TARGET값은 V\$MEMORY_TARGET 등을 조회하여 결정
- **대용량 DB 또는 중요 DB에는 사용하지 않도록 주의 필요**
 - SGA와 PGA를 서로 메모리 크기를 조정하면서 동적으로 할당이 가능하므로 시스템에 악영향을 미칠 가능성 존재
 - Linux의 경우에는 os의 HugePages를 지원하지 않음.

AMM 메모리 할당 메커니즘

- 1 전체 여유 메모리가 있는 경우 이를 SGA와 PGA가 활용하여 각각 메모리를 추가적으로 활용



- 2 SGA와 PGA가 MAX까지 확장하여 전체 여유 메모리가 없는 경우 **각각 상대방의 여유 메모리**를 자기 것으로 확장하여 활용

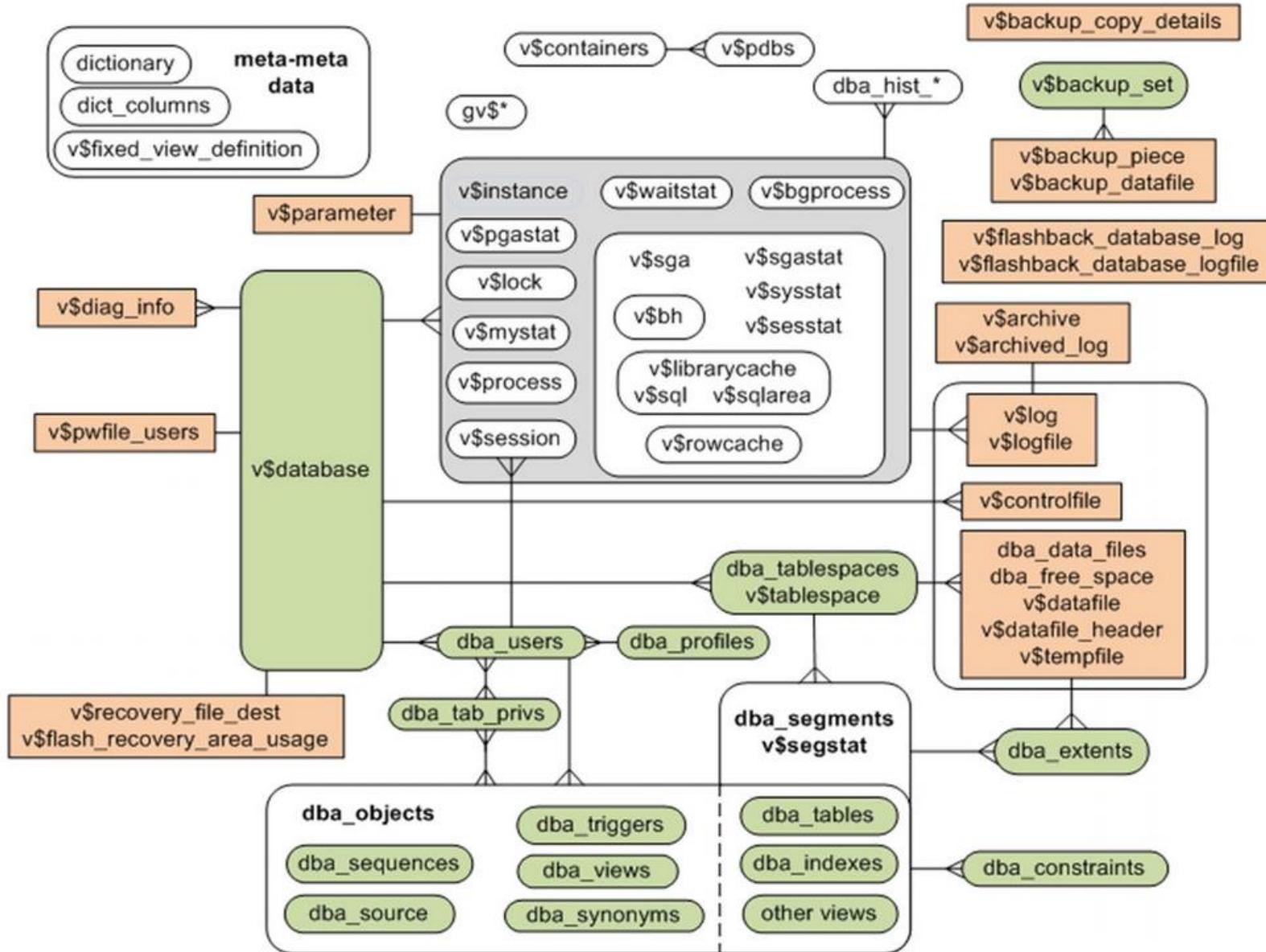


AMM 파라미터 설정 및 Case별 최초 할당 값

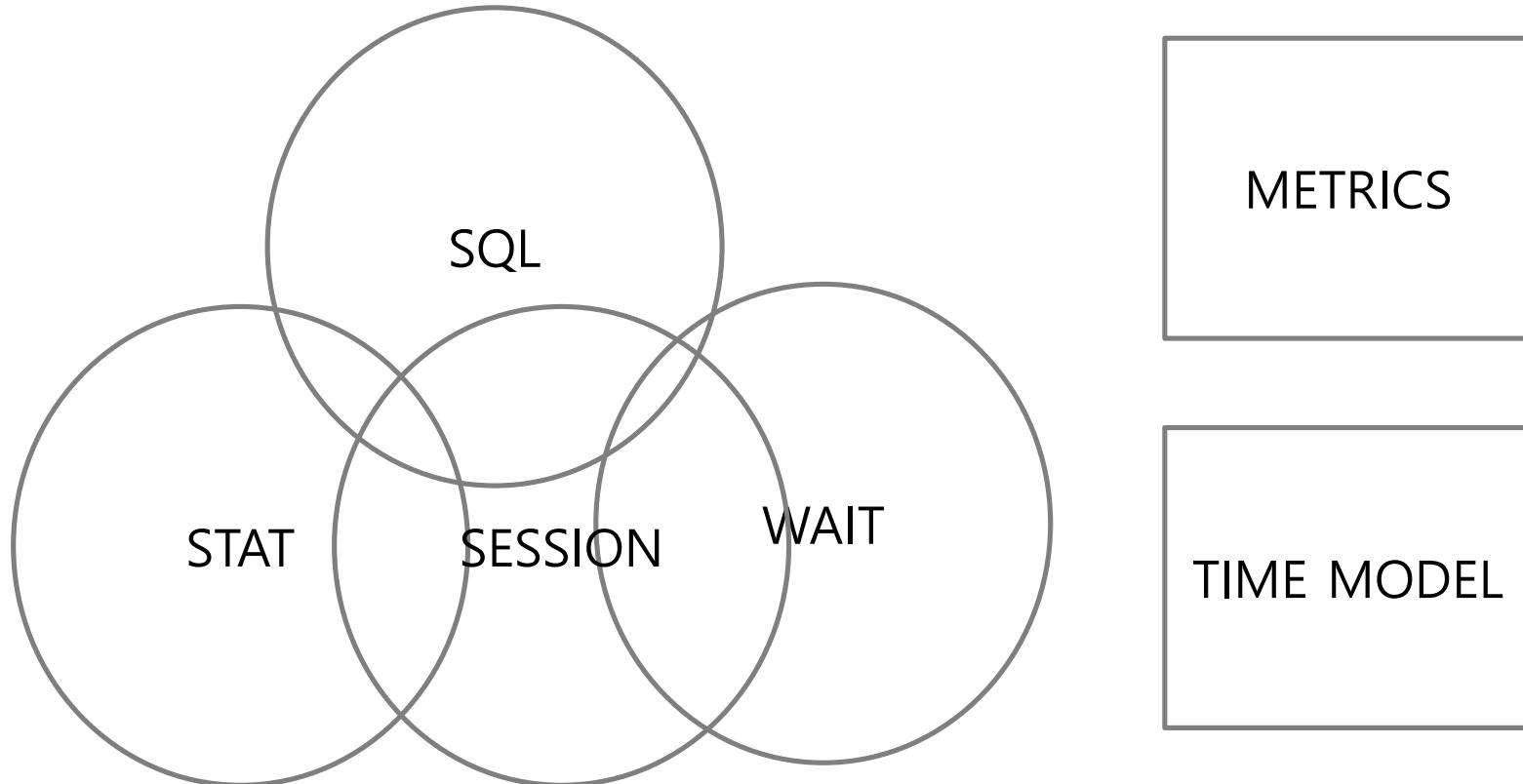
MEMORY_MAX_TARGET	MEMORY_TARGET	SGA_TARGET	SGA_MAX_SIZE	PGA_AGGREGATE_TARGET	설정
26G	20G	0	0	0	SGA와 PGA를 자동으로 설정(SGA는 MEMORY_TARGET의 약 60%, PGA는 40%로 설정. SGA_MAX_SIZE가 자동으로 설정됨)
26G	20G	10	12	4	설정 시 SGA_TARGET, PGA_AGGREGATE_TARGET의 최소값으로 사용

주요 성능 분석 Data Dictionary

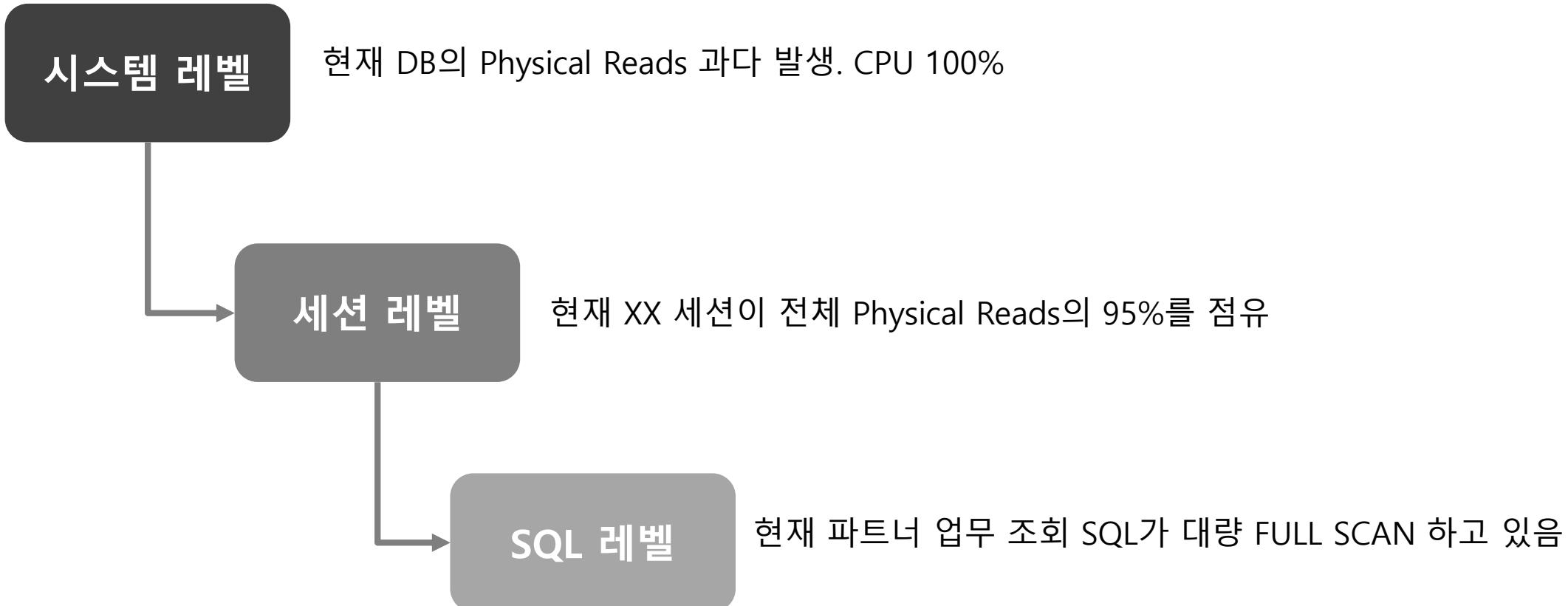
오라클 주요 Data Dictionary



DB 성능 모델 주요 구성 요소



일반적인 성능 분석 프로세스

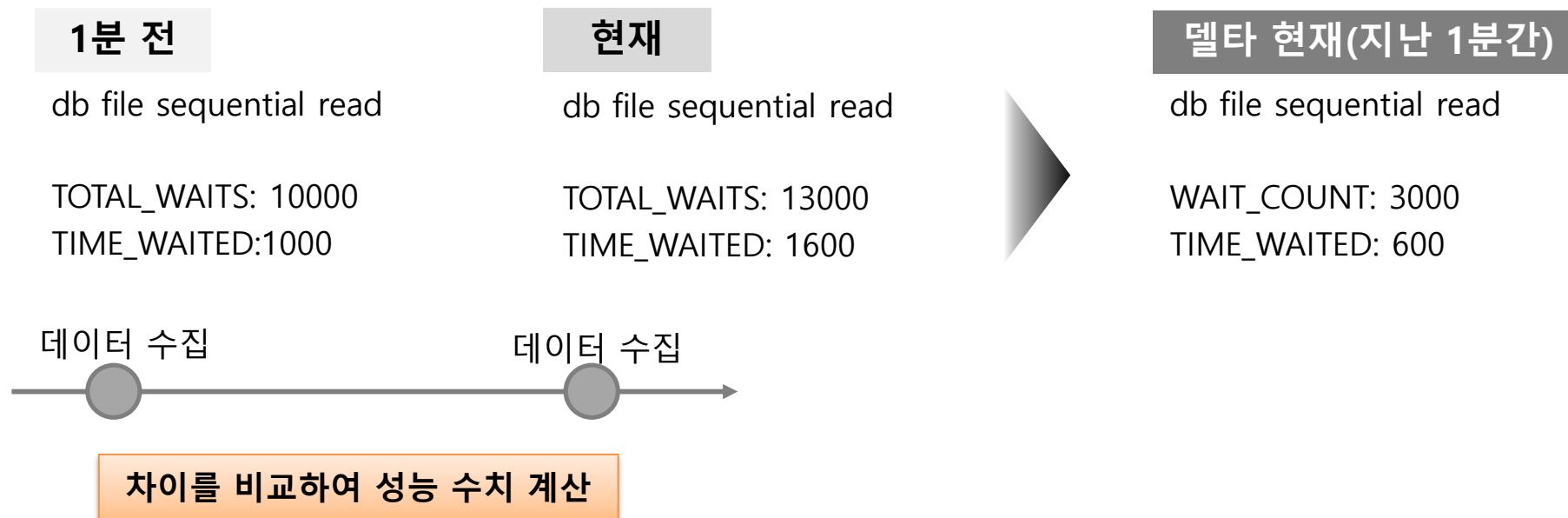


DB Performance 분석을 위한 주요 Data Dictionary

	시스템	세션	SQL
성능 수치(Stat)	V\$SYSSTAT	V\$SESSTAT	V\$SQLSTATS
Wait 수치 (Wait Event)	V\$SYSTEM_EVENT V\$SYSTEM_WAIT_CLASS	V\$SESSION_EVENT V\$SESSION_WAIT V\$SESSION_WAIT_CLASS	

주요 Performance Dictionary의 과거 개선 방향성

- 지표별 누적(Cumulative) 최종 값에서 델타 값으로
- 현재 시점 데이터에서 과거 특정 시점(Recent)부터 현재까지 데이터로
- 복합적인 지표로 직관적인 판단을 줄 수 있도록



시스템 Wait 누적 정보에서 Delta 최근

누적 현재

V\$SYSTEM_EVENT

Delta 현재

V\$EVENTMETRIC

Delta 최근

V\$SYSTEM_WAIT_CLASS

V\$WAITCLASSMETRIC

V\$WAITCLASSMETRIC_HISTORY

주요 Stat 메트릭

특정 기간 동안

- Buffer Cache Hit Ratio
- Hard Parse Ratio
- Library Cache Hit Ratio
- Shared Pool Free %
- Memory Sorts Ratio
- PGA Cache Hit %
-

초당(Per Second) 또는 트랜잭션당(Per Transaction)

- Physical Reads
- Physical Writes
- Physical Reads Direct
- Logical Reads
- Consistent Read Gets
- Consistent Read Changes
- DB Block Changes
- Redo Writes
- Executions
-

시스템 Stat 누적 정보에서 Delta 최근

누적 현재

V\$SYSSTAT

Delta 현재

V\$SYSMETRIC

Delta 최근

V\$SYSMETRIC_HISTORY

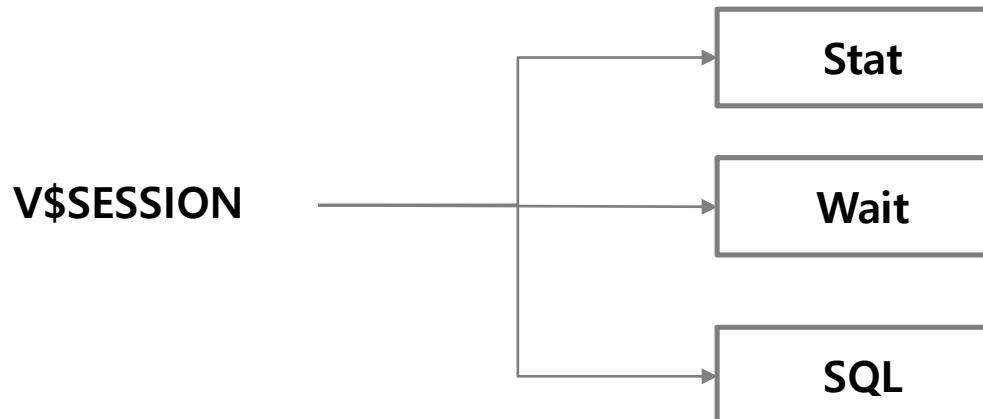
V\$SYSMETRIC_SUMMARY

Session 관련 주요 Performance Dictionary

V\$SESSION_WAIT : 세션의 현재 또는 마지막 Wait Event 정보

V\$SESSION_EVENT : 세션의 과거 부터 현재까지의 모든 Wait Event 정보

V\$SESSTAT : 세션의 Stat 수치를 누적 현재 값으로 표현



세션의 현재 또는 마지막 Stat, Wait, SQL 정보를 가짐

V\$SESSION과 V\$ACTIVE_SESSION_HISTORY

- V\$SESSION은 접속 세션의 현재 정보만 나타냄. V\$ACTIVE_SESSION_HISTORY는 세션의 30분 동안의 정보를 나타냄
- V\$ACTIVE_SESSION_HISTORY 은 V\$SESSION과 마찬가지로 세션의 Stat, Wait, SQL 레벨의 정보 및 보다 다양한 정보를 제공.
- 수집 단위가 1초이며, 당시 Active 인 Session에 대한 정보만 수집.
- V\$SESSION에 모니터링이 집중되는 현상을 피하고 보다 다양한 성능 정보를 제공하기 위해 만들어졌으나 수집 단위가 1초이므로 이보다 짧은 Operation이 세션에서 이뤄졌을 때 데이터 수집을 못할 수 있음.

세션 Wait/Stat 정보에서 Delta 최근

모든 Wait와 Stat

현재(마지막) Wait

과거 10개 Sample

과거 30분 델타

V\$SESSION_EVENT
V\$SESSTAT

V\$SESSION_WAIT

V\$SESSION_WAIT_HISTORY

V\$ACTIVE_SESSION_HISTORY

V\$SESSION

V\$SQL과 V\$SQLSTATS

V\$SQL : 한번 수행된 SQL에 대한 광범위한 정보를 가지고 있음. SQL Parsing, Optimizer에 대한 정보 뿐만 아니라, 수행 횟수, 수행 모듈, I/O 및 메모리 사용량 등의 다양한 성능 지표 등 SQL에 대한 거의 대부분의 데이터를 가지고 있음

V\$SQLSTATS : V\$SQL 보다 성능 지표 측면에서 보다 편리한 정보를 제공. 모니터링 툴에서 V\$SQL을 많이 조회함에 따라서 이에 따른 영향도를 줄이기 위해 편리하게 SQL 성능 지표를 제공하기 위해서 만들어짐.

AWR을 구성하는 DBA_HIST_XXX

Wait

DBA_HIST_WAITSTAT
DBA_HIST_SYSTEM_EVENT
DBA_HIST_BG_EVENT_SUMMARY

Snapshot 기준

DBA_HIST_SNAPSHOT
DBA_HIST_DATABASE_INSTANCE

Stat

System

DBA_HIST_SYSSTAT
DBA_HIST_SYS_TIME_MODEL

SGA

DBA_HIST_SGA
DBA_HIST_SGASTAT
DBA_HIST_PGASTAT
DBA_HIST_LIBRARY_CACHE
DBA_HIST_BUFFER_POOL_STAT

Advice

DBA_HIST_DB_CACHE_ADVICE
DBA_HIST_PGA_TARGET_ADVICE
DBA_HIST_SHARED_POOL_ADVICE
DBA_HIST_JAVA_POOL_ADVICE

SQL

DBA_HIST_SQLSTAT
DBA_HIST_SQLTEXT
DBA_HIST_SQL_SUMMARY

Latch & Enqueue

DBA_HIST_LATCH
DBA_HIST_ENQUEUE_STAT

Segment, Tablespace, Datafile I/O

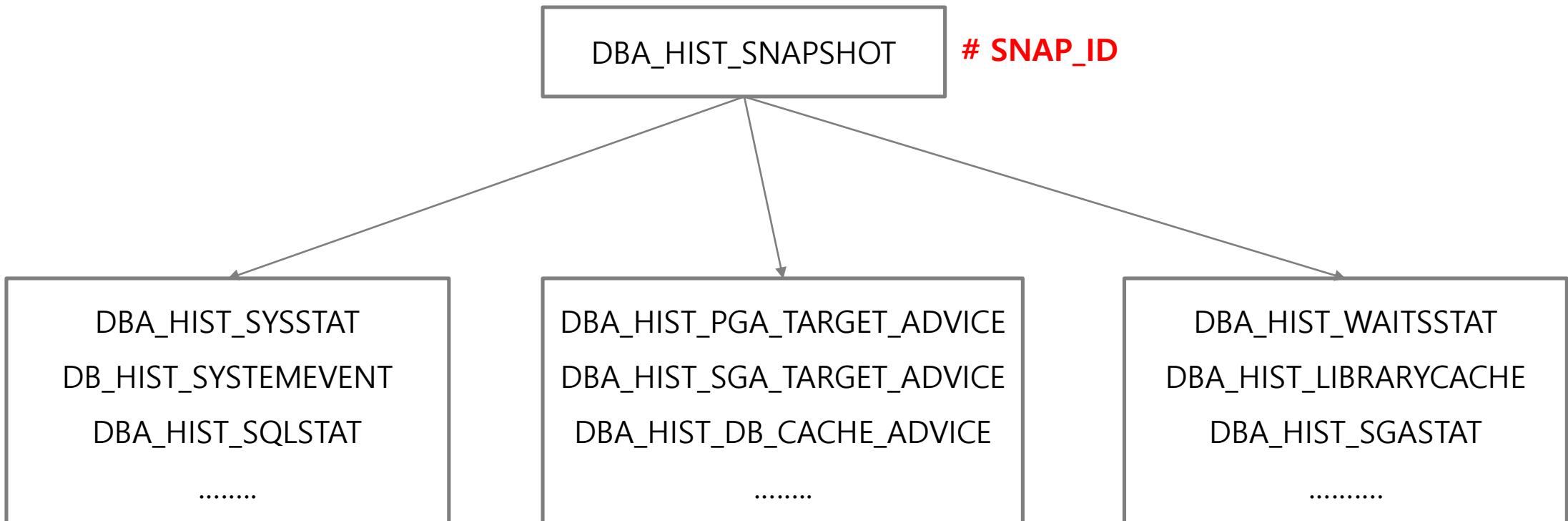
DBA_HIST_SEG_STAT
DBA_HIST_TABLESPACE
DBA_HIST_UNDOSTAT
DBA_HIST_ROLLSTAT
DBA_HIST_DATAFILE
DBA_HIST_TEMPFILE

DBA_HIST_XXX에 직접 SQL 적용하여 AWR 데이터 추출하기

```
select sy.stat_name, sy.snap_id, sy.instance_number inst, to_char(sn.begin_interval_time,'yyyy-mm-dd hh24:mi:ss') b_snap_time,  
to_char(sn.end_interval_time,'yyyy-mm-dd hh24:mi:ss') e_snap_time,      n  
nvl(decode(greatest(sy.value,nvl(lag(sy.value) over (partition by sy.dbid,sy.instance_number, sy.stat_name order by sy.snap_id),0))  
,sy.value ,sy.value - lag(sy.value) over (partition by sy.dbid,sy.instance_number,sy.stat_name order by sy.snap_id) ,sy.value)  
, 0) delta_val,  
.....  
from dba_hist_sysstat sy, dba_hist_snapshot sn  
where sy.snap_id = sn.snap_id  
and sy.instance_number = sn.instance_number  
and sy.dbid = sn.dbid  
and sy.stat_name in (  
'redo size' , 'session logical reads', 'db block changes',  
'physical reads', 'physical reads direct', 'physical writes',  
'physical writes direct', 'user calls', 'logons cumulative',  
'CPU used by this session'  
)  
and sn.begin_interval_time >= to_date('20200514','yyyymmdd') and sn.end_interval_time < to_date('20200515','yyyymmdd')
```

physical reads	41233	1	2013-05-14 00:00:56	2013-05-14 01:00:32	0	0
physical reads	41234	1	2013-05-14 01:00:32	2013-05-14 02:00:09	98417	27.5171309
physical reads	41235	1	2013-05-14 02:00:09	2013-05-14 03:00:45	7133	1.96141784
physical reads	41236	1	2013-05-14 03:00:45	2013-05-14 04:00:19	7175	2.00771241
CPU used by this session	41233	1	2013-05-14 00:00:56	2013-05-14 01:00:32	0	0
CPU used by this session	41234	1	2013-05-14 01:00:32	2013-05-14 02:00:09	14005	3.91576068
CPU used by this session	41235	1	2013-05-14 02:00:09	2013-05-14 03:00:45	11719	3.2224668
CPU used by this session	41236	1	2013-05-14 03:00:45	2013-05-14 04:00:19	11467	3.2087022
CPU used by this session	41237	1	2013-05-14 04:00:19	2013-05-14 05:00:47	11611	3.20073966
CPU used by this session	41238	1	2013-05-14 05:00:47	2013-05-14 06:00:20	11794	3.30034296

DBA_HIST_XXX View 구조 개요



Wait Event

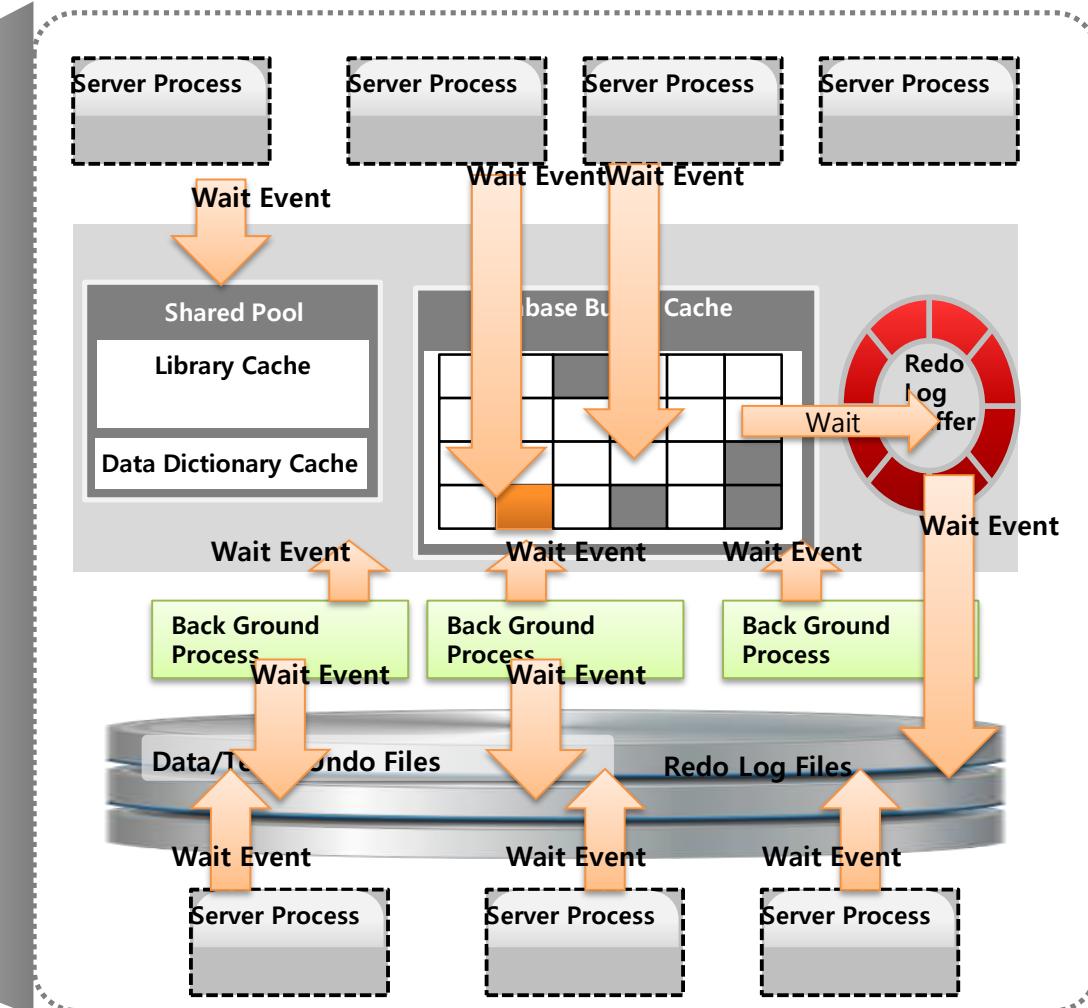
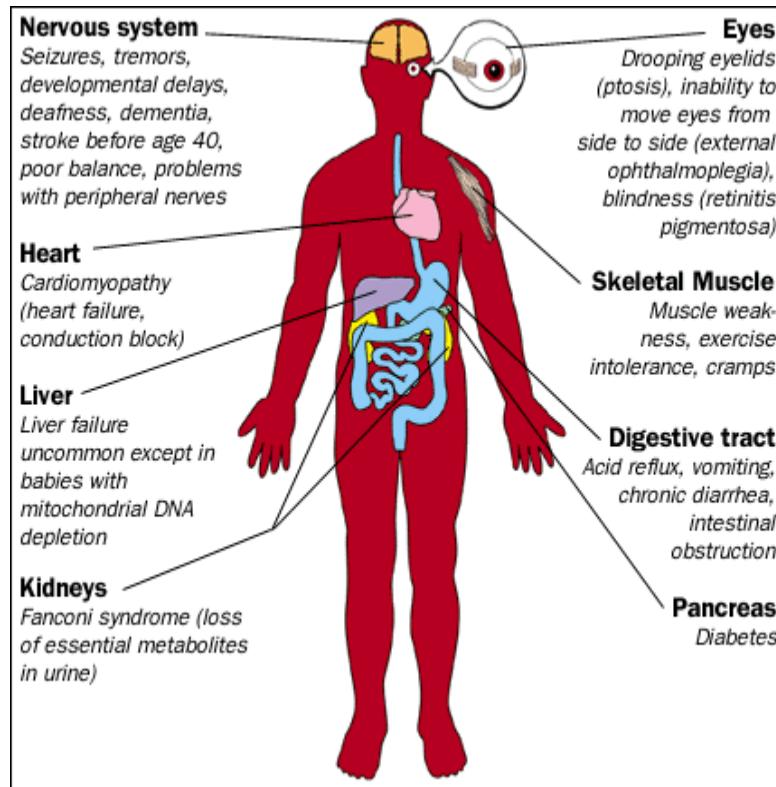
Oracle Wait Event

Wait Event

- 성능 및 운영 관리를 위해 알아야 할 가장 중요한 요소
- 오라클은 서버프로세스 또는 백그라운드 프로세스가 DB의 CPU , Memory , Database File, I/O 등의 모든 자원을 액세스하고 이용하는 세부 단계별로 모든 Wait Event (대기 이벤트) 를 발생
- 오라클의 모든 성능 요소 및 자원 사용현황은 Wait Event 의 발생정도에 따라 모니터링 가능함
- Load Profile (메모리/디스크/Transaction 수 등의 사용 부하)와 결합되어 더욱 세밀한 분석이 가능함.

DB 내 문제점 진단과 분석을 위한 Wait Event

인체내의 각 기관별로 진단을 위한 다양한 증상 및 이에 대한 해결책을 발전시켜 왔듯이, 오라클은 DB내의 거의 모든 세부 프로세스 절차에서 **Wait Event**를 의무적으로 발생시키는 성능 분석 Framework 을 발전시켜 옴.



Idle Wait Event & Non Idle Wait Event

Idle Wait Event

Timer, Message Sending, Client data waiting 등 리소스를 사용하지 않거나 클라이언트로 부터 답변을 기다려야 하는 등의 Wait Event

DB 성능에 큰 영향을 미치지 않는 Wait Event(그렇다고 application 영향이 없는 것은 아님)



Non Idle Wait Event

DB내에서 Active Work 처리 과정 중에 발생하는 Wait Event
DB Time에 포함되어 중점적인 성능 모니터링 요소인 Wait Event

DB Time



Foreground 세션이 DB Call을 하여 자원을 사용한 총 시간

CPU 시간 , I/O 시간 , Non Idle Wait Event 시간 등으로 구성

오라클 성능 측정을 위한 기본 수치

Database Time은 User process 가 실제 일을 하면서 , 또는 db에서 waiting 되면서 사용된 총 시간을 말함

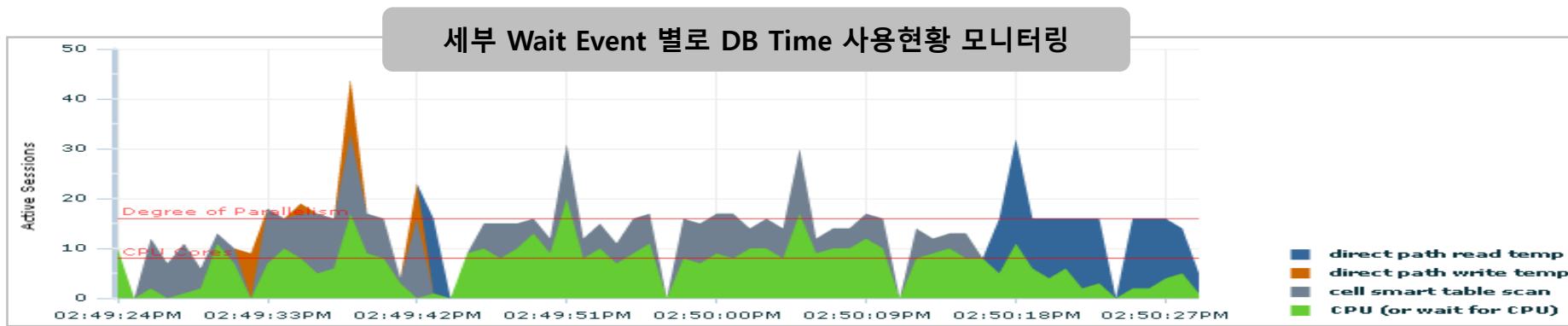
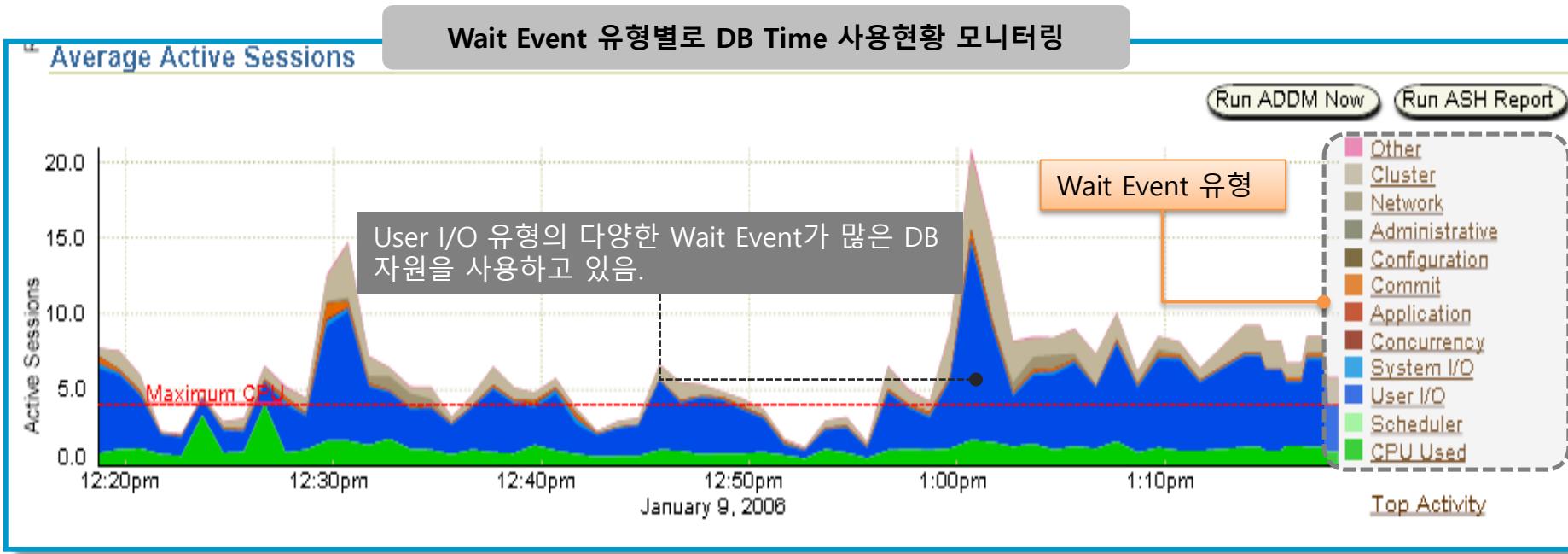
$$\text{DB Time} = \text{CPU Time} + \text{Non Idle Wait Time}$$

전체 DB Time 중 어떤 유형의 Wait Event 가 많은 비중을 차지하고 있는가 ?

DB 시스템의 현재 어떤 자원을 이용하는데 가장 많은 DB Time을 사용하고 있는가?

DB가 가장 많이 사용하는 시스템 자원은 무엇이며 , 성능 상의 병목은 무엇인가?

Wait Event로 DB Time 사용현황 모니터링 예제



Wait Event Class

Class 명	Class 설명	Wait Event 예
USER I/O	User I/O 관련된 다양한 Wait Event	Db file sequential read Db file scattered read
System I/O	Background process I/O 와 관련된 Wait Event	Db file parallel write
Administrative	Index Rebuild와 같은 DBA Command 로 인하여 발생하는 Wait Event	Alter rbs offline
Concurrency	Latch와 같은 Internal Database resource와 관련된 Wait Event	Buffer busy wait Library cache lock
Cluster	RAC 의 Resource와 관련된 Wait Event	Gc cr block busy
Application	사용자 Application에서 사용된 row level Locking 또는 명확한 Lock Command 로 인한 Wait Event	Enq:TX – row lock Contention Wait for Table Lock
Commit	Commit 시 발생하는 Redo log write에 관련된 Wait Event	Log file sync
Idle	세션이 Inactive 상태에서 Work을 수행 준비하는 단계에서 발생하는 Wait Event	SQL *Net Message form client
Network	네트워크 Messaging시에 발생하는 Wait Event	SQL *Net more data to dblink
Other	정상적인 상황에서는 잘 발생하지 않는 Wait Event	Wait for EMON to spawn
Configuration	Database 또는 Instance 자원의 부적절한 환경 설정 예를 들어 log file size , shared pool size 를 적절하게 설정하지 못한 경우 발생하는 Wait Event	Write complete waits

오라클 주요 구성 요소별 I/O 관련 Wait Event

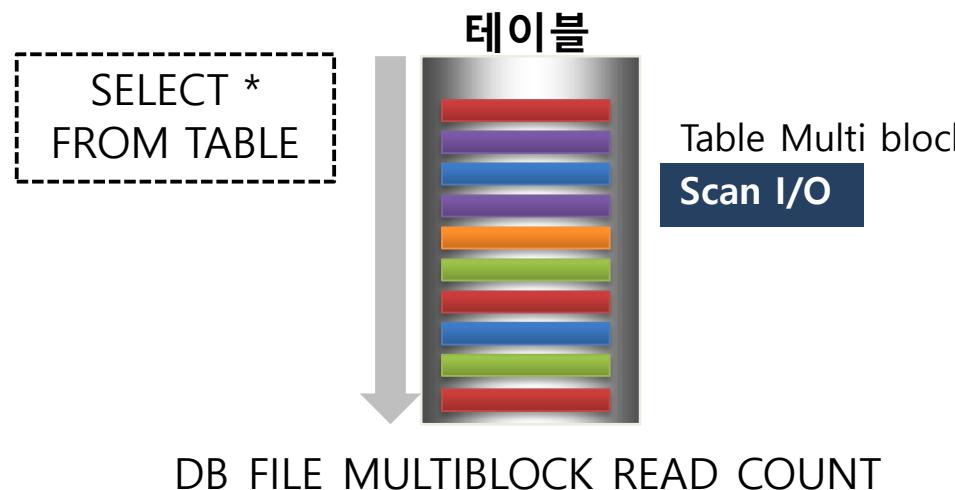
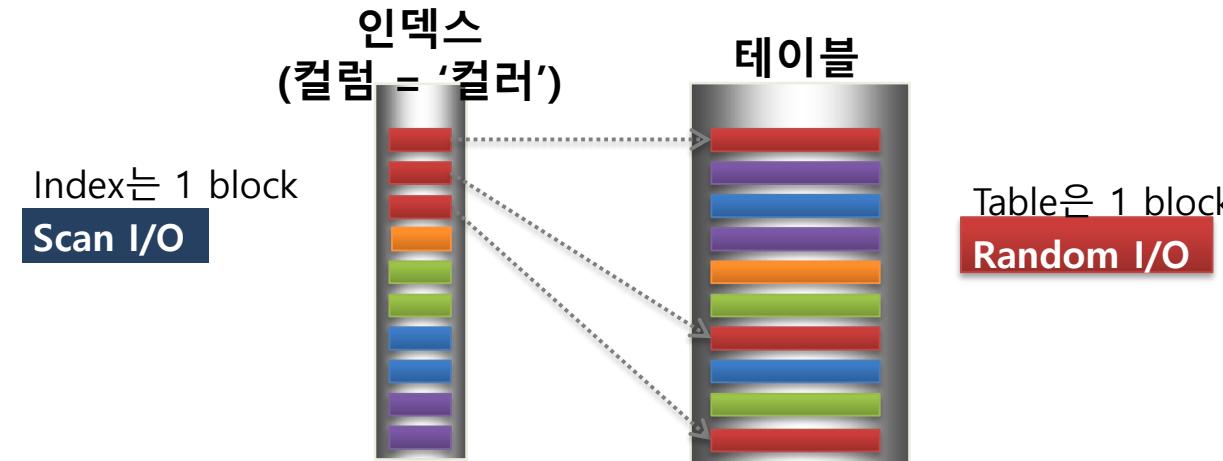
주요 I/O 대상 구분	Wait Event	Wait Class
Datafile I/O	db file sequential read	User I/O
	db file scattered read	User I/O
	direct path reads and writes	User I/O
Temporary Datafile I/O	direct path write temp	User I/O
	direct path read temp	User I/O
Buffer Cache (Latch 포함)	latch: cache buffer chains	Concurrency
	latch: cache buffer lru chains	Concurrency
	buffer busy wait	Concurrency
	db file parallel write	System I/O
	free buffer waits	Configuration
	write complete waits	Configuration

오라클 주요 구성 요소별 I/O 관련 Wait Event

주요 I/O 대상 구분	Wait Event	Wait Class
Redo Logging I/O	Log file sync	Commit
	Log file parallel write	System I/O
	Log buffer space	Configuration
	Log file switch	Configuration
	Log file switch completion	Configuration
Shared Pool	Library cache: mutex x	Concurrency
	Library cache lock	Concurrency
	Cursor: pin s wait on x	Concurrency
	Latch: shared pool	Concurrency
	Kksfbc child completion	Concurrency

db file sequential read와 db file scattered read

SELECT * FROM TABLE WHERE 컬러='RED'



db file sequential read

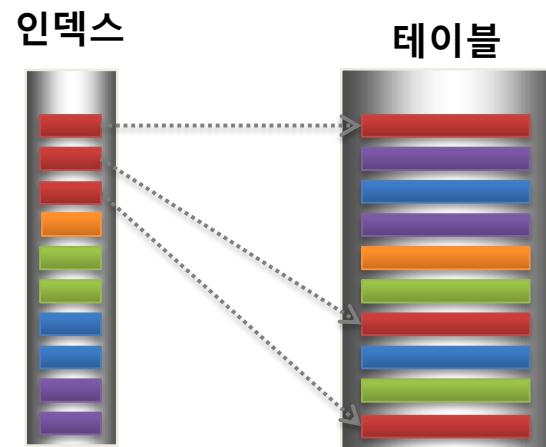
- Single block Storage I/O 를 수행할 때 발생
- 주로 Index를 경유하여 Table을 random access 할 때 발생하며 많은 Wait time 소모 (Index unique/range scan, Table random I/O Access 형태)
- 대량 발생 시 주로 OLTP 시스템에서 성능을 저하시키는 주 원인

db file scattered read

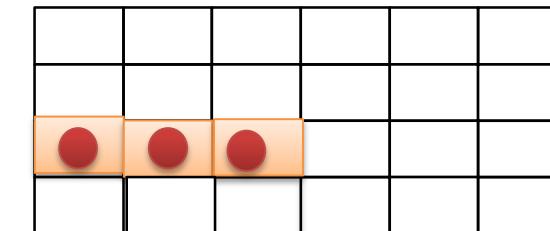
- Multi block Storage I/O를 수행할 때 발생
- 주로 테이블을 Full Scan할 때 발생(Index Full 시에도 발생)
- db_file_multiblock_read_count에 지정된 수 만큼 multi block을 한번에 Access

db file sequential read와 db file scattered read

db file sequential read

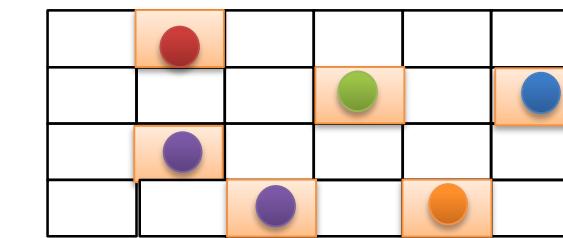
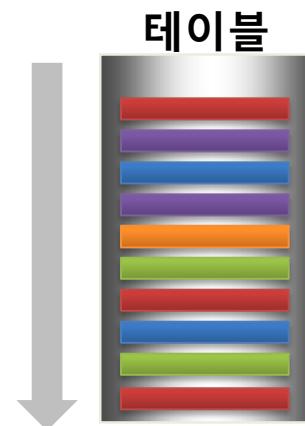


Buffer Cache



Random I/O Access 된 Block이 Buffer Cache에는
Sequential하게 보관됨

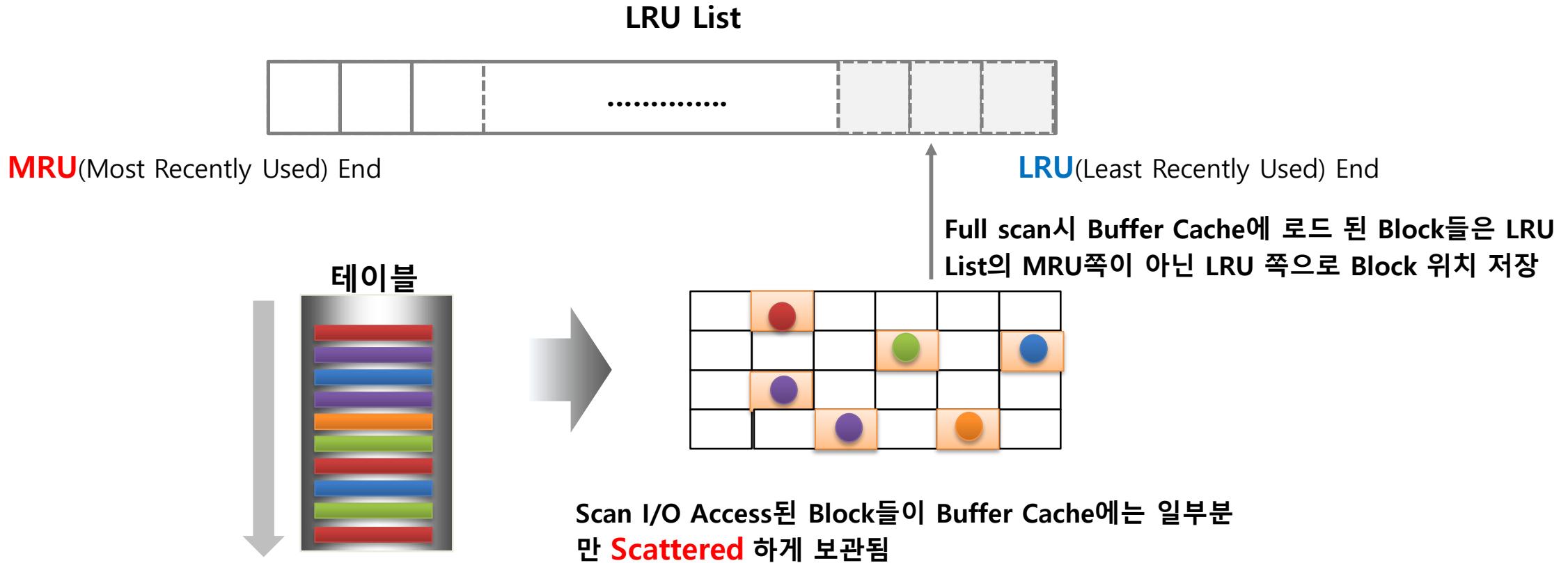
db file scattered read



Scan I/O Access된 Block들이 Buffer Cache에는 일부분만
Scattered하게 보관됨

Full Scan 시 Buffer Cache 활용 메커니즘

대량의 데이터가 Buffer Cache에 로드되면서 기존의 Pinned 된 Block들이 Buffer Cache에서 Out을 최대한 억제하는 방향으로 관리



11g 이후부터는 비교적 작은 테이블만 full scan시 db file scattered read를 발생하면서 Buffer cache에 로드 됨. 일정 용량 이상의 테이블을 full scan시 direct path read 를 발생하며 Buffer cache에 로드 하지 않음.

db_file_sequential_read와 db_file_scattered_read 개선 방안

db_file_sequential_read 개선 방안

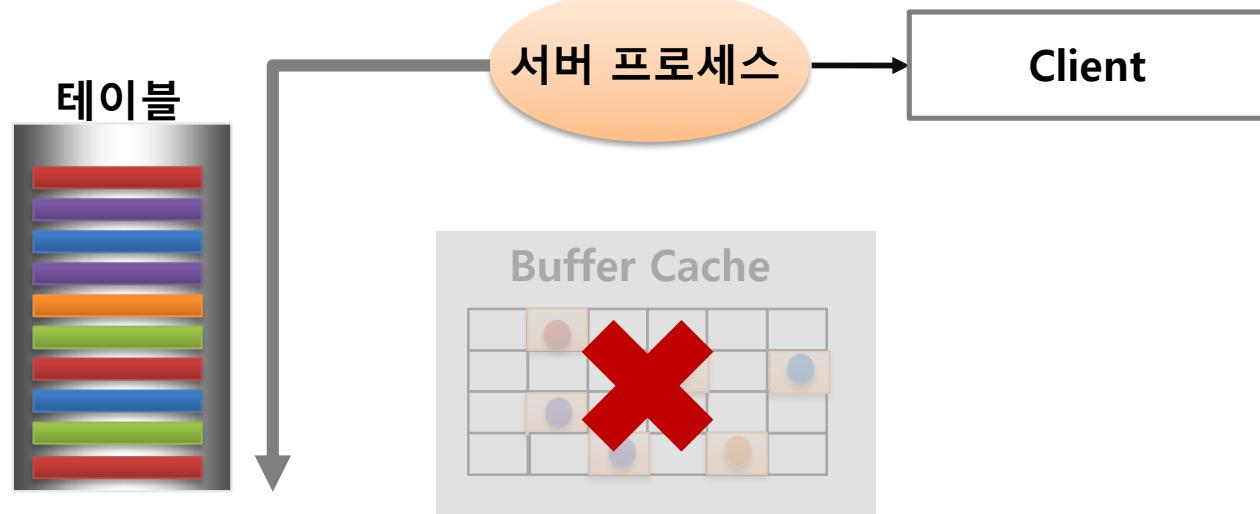
- Random I/O 를 많이 소모하는 SQL 튜닝 (Physical I/O 뿐만 아니라 Buffer Cache 많이 소모하는 SQL)
- Random I/O 성능이 더 뛰어난 스토리지 기술 적용(Striping, SSD를 포함한 스토리지 고려)
- ? Buffer Cache Size 늘리기(단 V\$DB_CACHE_ADVICE 에서 증가 효과가 검증 된 경우)
- ? Table(Access하려는 컬럼 순으로 재 정렬 생성)과 Index Rebuild는 해당 문제 해결에 도움이 되지 않는 경우가 대부분
- db_file_sequential_read는 OLTP에서 자연스러운 Wait Event임. CPU 대비 발생 비율, 1회 발생시 Wait time에 주목할 것

db_file_scattered_read 개선 방안

- Throughput이 뛰어난 스토리지 기술 적용(Striping, SSD를 포함한 스토리지 고려)
- DB_FILE_MULTIBLOCK_READ_COUNT를 늘리는 것은 큰 도움이 되지 못함
- 11g 부터는 큰 테이블의 경우 주로 direct I/O가 발생. Hidden parameter인 _small_table_threshold, _serial_direct_read에 따라 결정

direct path read, direct path write

서버 프로세스가 Buffer Cache를 거치지 않고 바로 Storage에 direct read, write I/O를 수행 시에 발생



한번 Access 시 `_db_file_direct_io_count` 히든 파라
미터 설정값(기본 1M bytes) 만큼 Scan

direct path read

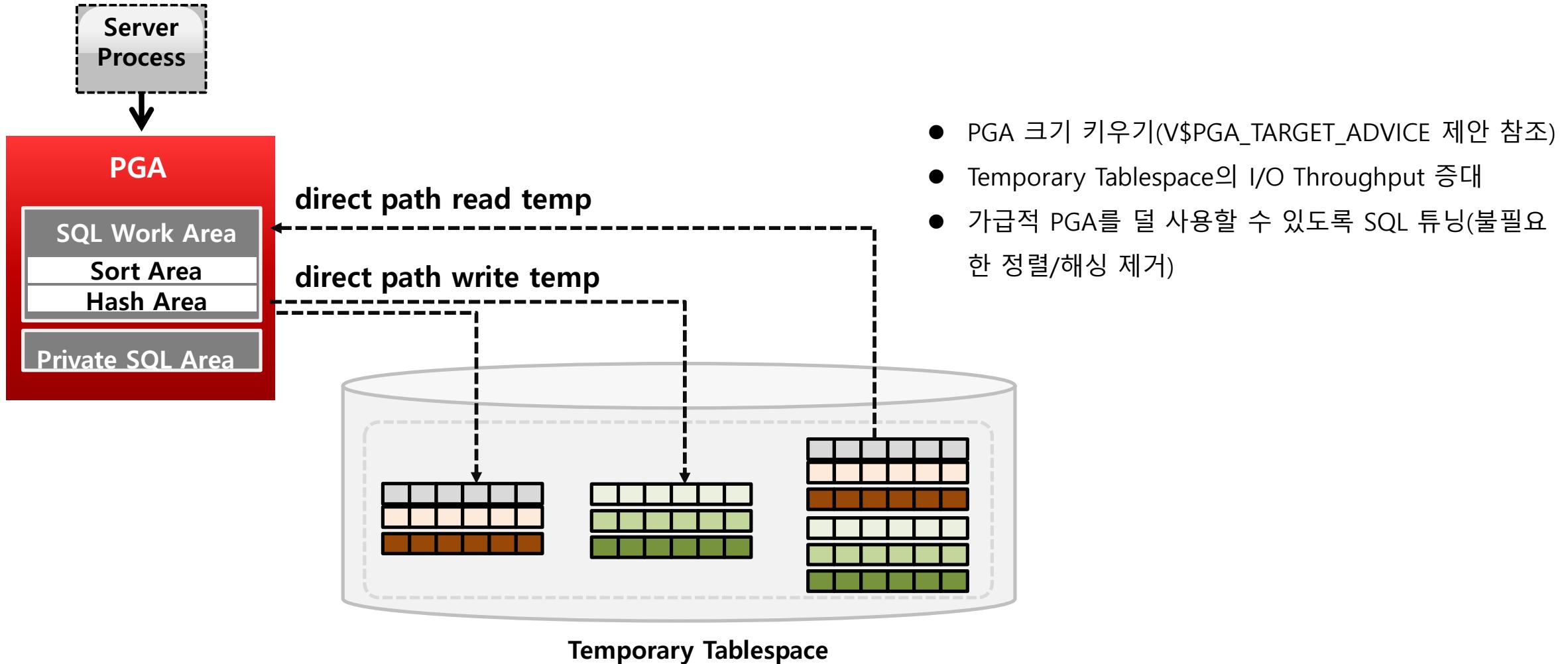
- 일정 크기 이상의 테이블 Full Scan시
- Create table AS Select 구문 시(select 시)
- Parallel Query 를 이용하여 Full Scan 시

direct path write

- Insert /*+ append */ direct I/O
- Create table AS Select 구문 시(Create 시)
- Parallel DML

direct path read temp와 direct path write temp

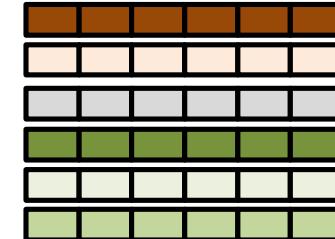
서버 프로세스가 정렬 또는 해싱 작업 등으로 PGA를 활용 시 메모리가 부족할 경우 Temporary Tablespace에 read/write를 수행하면서 발생하는 Wait Event



PGA를 활용한 SQL 정렬 메커니즘

SELECT * FROM CUSTOMER ORDER BY CUSTOMER_NAME DESC

정렬 대상 데이터



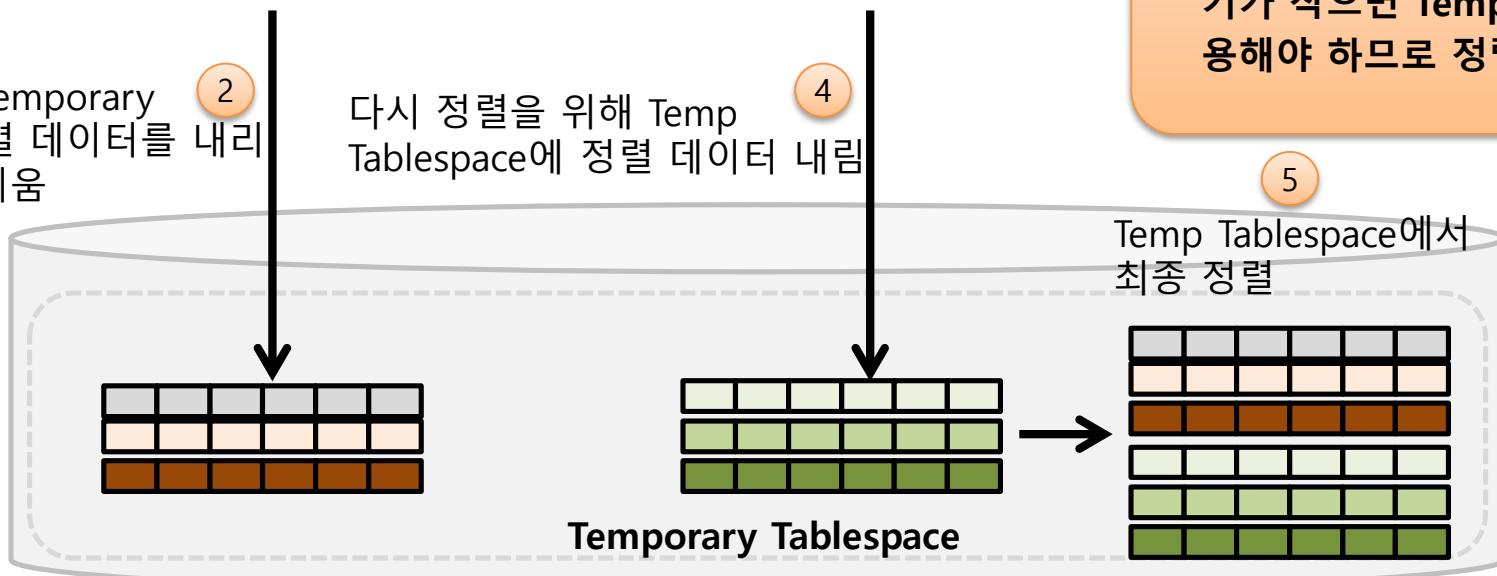
- 1 PGA 메모리에서 정렬 가능한 데이터량만
큼만 1차로 정렬(PGA 사이즈 점차 증가) 3 추가 데이터를 다시 PGA에서
2차 정렬



추가 정렬을 위해 Temporary
Tablespace 1차 정렬 데이터를 내리
고 PGA 메모리를 비움

다시 정렬을 위해 Temp
Tablespace에 정렬 데이터 내림

- 정렬해야 할 대상이 작으면 PGA 메모리 내에서 정렬이 가능하므로 정렬 속도가 빠름
- 정렬해야 할 대상이 크고 PGA의 크기가 작으면 Temp Tablespace 를 활용해야 하므로 정렬 속도가 느려짐



Oracle Latch와 Lock(Enqueue) 개요

Lock(Enqueue)



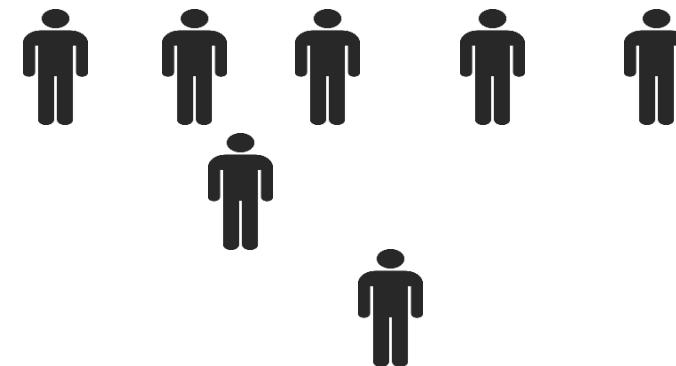
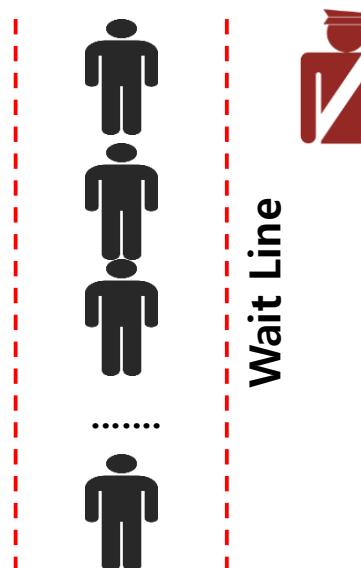
요청 순서를 보장. 단일 처리 시간이 상대적으로 오래 걸림

Latch



요청 순서를 보장하지 않음. 단일 처리 시간이 매우 빠름

아주 빠른 속도로 동작하는 경량화된 Lock

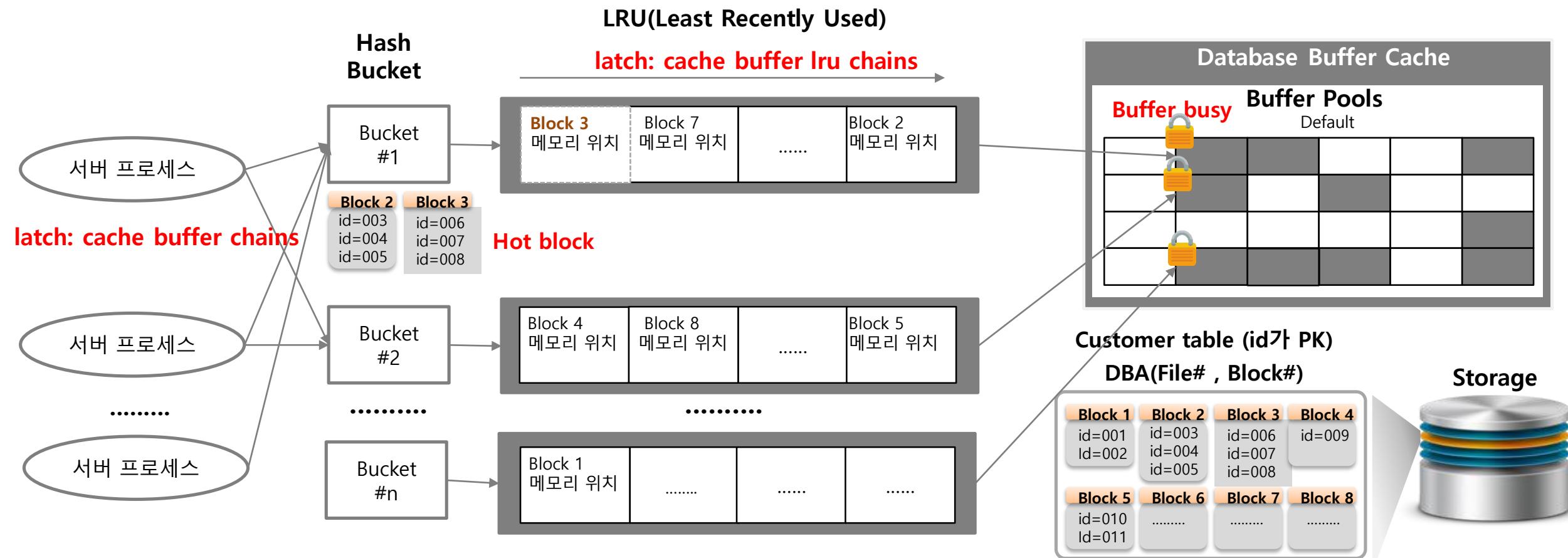


Latch, Enqueue, Mutex 비교

특성	Latch	Lock(Enqueue)	Mutex
사용 범위	SGA 내부의 데이터 구조의 접근 제어를 위해 사용.	테이블, DB Block 등 오브젝트에 접근 제어를 위해 사용함 대부분 트랜잭션 단위	주로 SGA의 Library Cache 영역에 적용 (Oracle 10g부터 적용됨)
실행 속도	CPU Register에 직접 Access하는 CPU 로직 구현으로 매우 빠름	OS FIFO를 이용. 구현이 복잡하고 상대적으로 느림	OS Mutex를 사용하지만 매우 빠름
지속 시간	짧은 순간만 지속됨 (microsecond 단위) = 100만분의 1초	일정시간동안 지속(트랜잭션 동안)	대부분 짧은 순간만 지속됨
요청 순서 보장 여부	큐(queue)로 관리되지 않으며, 요청한 순서대로 서비스 되지 않음 (일부 Latch 예외)	프로세스가 락 획득을 실패한 후, 해당 요청은 큐(queue)로 관리되며, 요청한 순서대로 서비스됨(nowait 모드는 예외)	요청한 순서대로 서비스 되지 않음
범위	SGA 내부에 정보가 존재하며, 로컬 인스턴스에만 볼 수 있음.	데이터베이스 내부에 정보가 존재하며, 모든 인스턴스에서 볼 수 있음.	SGA 내부에 정보가 존재하며, 로컬 인스턴스에만 볼 수 있음.
획득방식	willing-to-wait 또는 no-wait	6가지 모드로 요청 가능 : null, row share, row exclusive, share, share row exclusive, 또는 exclusive	

cache buffer chains와 cache buffer lru chains 개요

- 서버 프로세스가 Buffer Cache내에 있는 block을 찾기 위해서는 Hash Bucket과 LRU를 거쳐야 하는데 이들의 Latch를 획득해야 함.
- latch: cache buffer chains는 Hash Bucket의 Latch획득 시 발생하는 wait event이며, latch: cache buffer lru chains는 LRU의 Latch획득 시 발생하는 Wait Event임.
- 주로 대량 범위를 Access하는 악성 SQL들이 빈번하게 Cache Buffer를 사용하면서 발생하는 Wait Event



다량의 I/O를 빈번하게 수행하는 SQL

1시간 동안의 AWR

SQL ordered by Gets

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- Total Buffer Gets: 385,423,694
- Captured SQL account for 49.0% of Total

Buffer Gets	Executions	Gets per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
48,802,744	2,002	24,377.00	12.66	1232.80	1210.51	3cckptg80jp0r	JDBC Thin Client	Query 1
39,510,347	1,620	24,389.10	10.25	984.51	966.90	152h031qafmsb	JDBC Thin Client	Query 2
13,316,796	207	64,332.35	3.46	348.81	563.31	3c7cdsarqcb1g	JDBC Thin Client	Query 3
10,282,777	124	82,925.62	2.67	57.61	101.14	aa5kggy8km37vm	JDBC Thin Client	Query 4
8,883,054	82	108,329.93	2.30	45.89	90.40	22k8b99gfd8bh	JDBC Thin Client	Query 5
8,388,060	90	93,200.67	2.18	56.85	55.76	7xyvf367s1hqp	JDBC Thin Client	Query 6
7,882,506	84	93,839.36	2.05	67.26	66.06	9kh1htsk6wgjv	JDBC Thin Client	Query 7

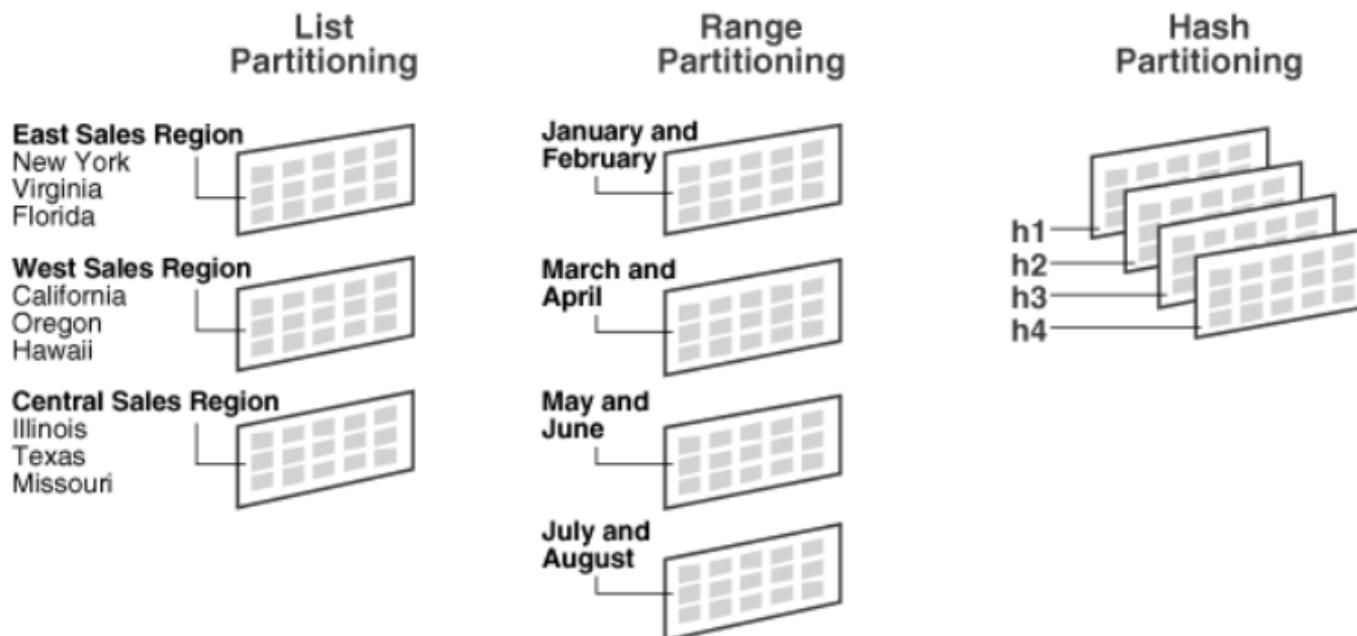
SQL ordered by Reads

- Total Disk Reads: 4,303,014
- Captured SQL account for 54.4% of Total

Physical Reads	Executions	Reads per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
1,443,555	207	6,973.70	33.55	348.81	563.31	3c7cdsarqcb1g	JDBC Thin Client	Query 3
147,615	74	1,994.80	3.43	58.91	1052.78	3t012kz0cq67u	JDBC Thin Client	Query 8
132,243	33	4,007.36	3.07	54.78	548.83	90tgjk30g94wr	JDBC Thin Client	Query 9
97,187	3	32,395.67	2.26	13.81	222.11	8j9x0brshcvm6	JDBC Thin Client	Query 10
74,506	43	1,732.70	1.73	14.32	425.69	0xt49thxbu7az	JDBC Thin Client	Query 11
67,626	60	1,127.10	1.57	14.46	233.52	00jtqjrc3vhrr	JDBC Thin Client	Query 12

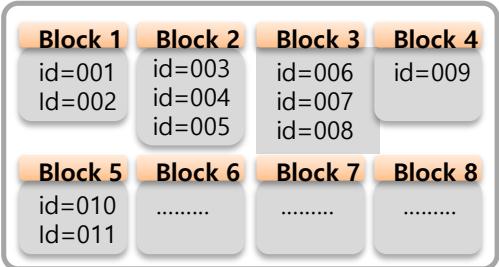
cache buffer chains와 cache buffer Iru chains 해결 방안

- 대량 범위의 I/O를 수행하는 SQL 튜닝
- Hash Partition등으로 Block 내부의 데이터를 hot block을 회피 할 수 있도록 재 정렬

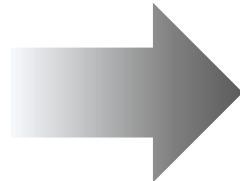


Hash Partition을 통한 Hot block 개선

기존 테이블

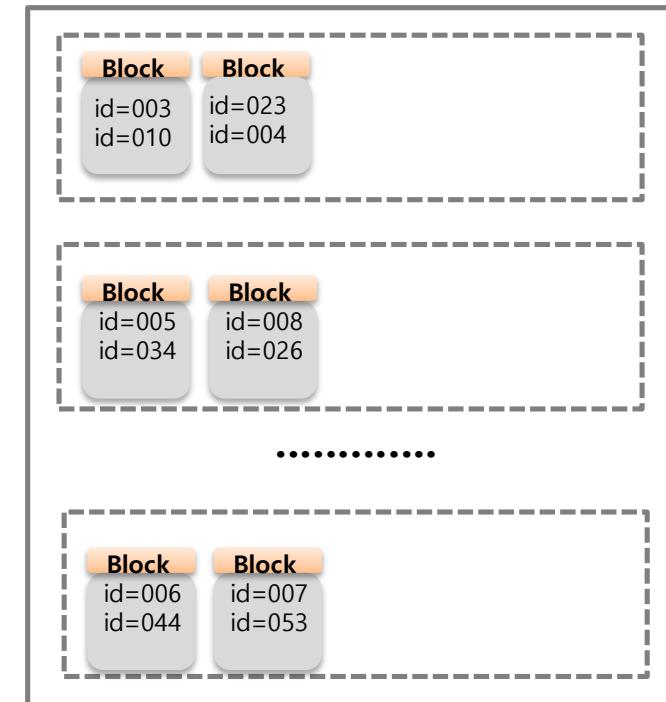


Customer table
(id가 PK)



- Hash Partition으로 Block 내부의 데이터들을 id 별로 다시 hash key에 따라 재 분배하여 Block을 재 생성하므로 Hot Block을 회피할 수 있음.
- 하지만 Hash partition 생성 시 Global Index 생성 등 관리 필요성 증대 및 Clustering factor 저하 등의 영향도 감안 필요

Hash Partition된 테이블



CREATE TABLE CUSTOMER

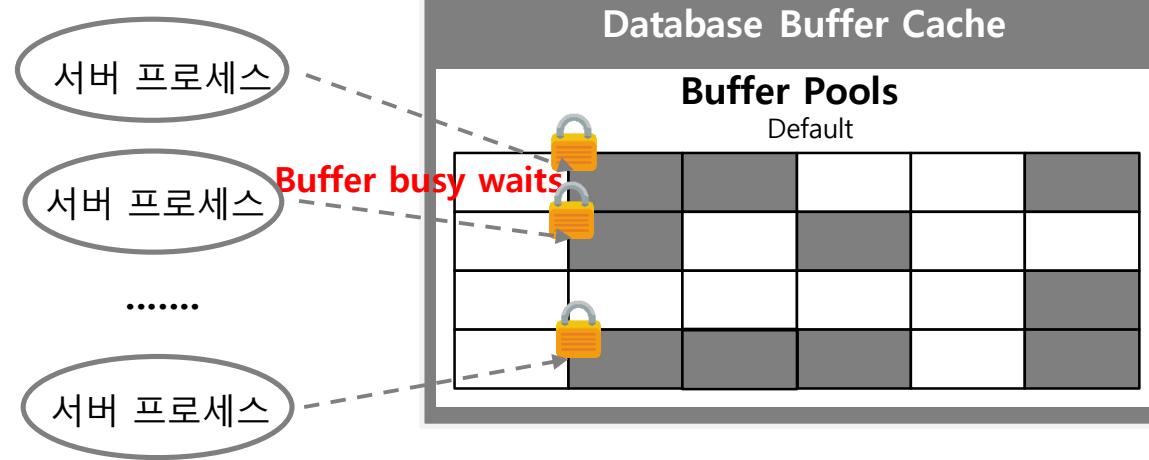
(id NUMBER, name VARCHAR2 (60))

PARTITION BY HASH (id)

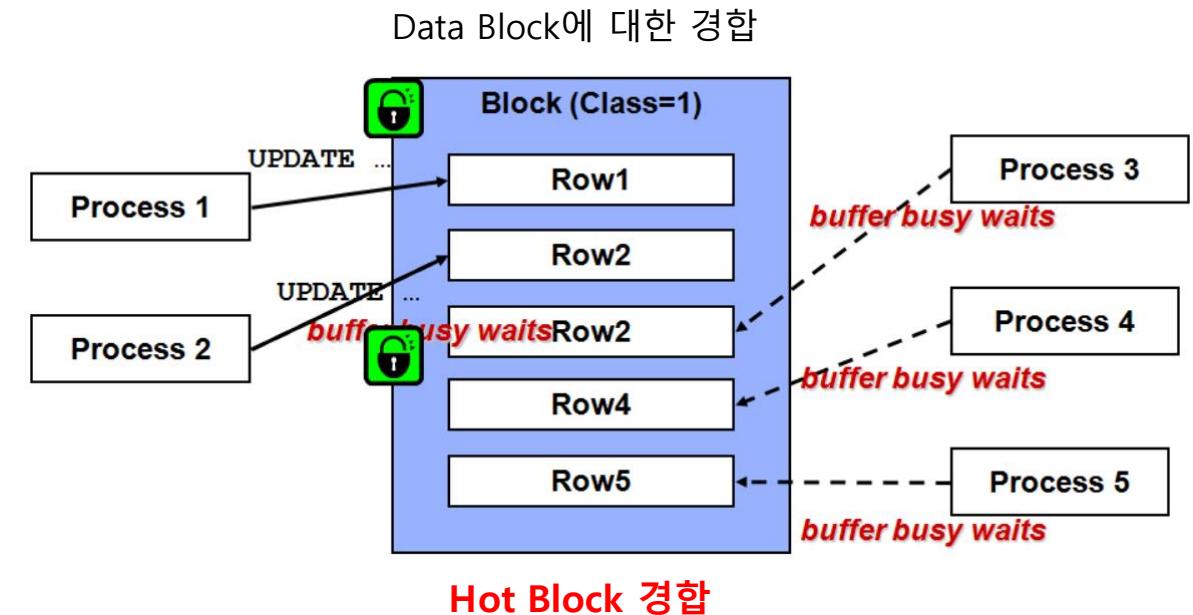
PARTITIONS 32

Buffer Busy Waits 개요

- 서버 프로세스가 Buffer cache내 block을 Access하기 위해서는 해당 Block에 대한 Lock을 획득 필요
- 동시에 여러 서버 프로세스들이 동일한 buffer block에 Access시 경합 발생.



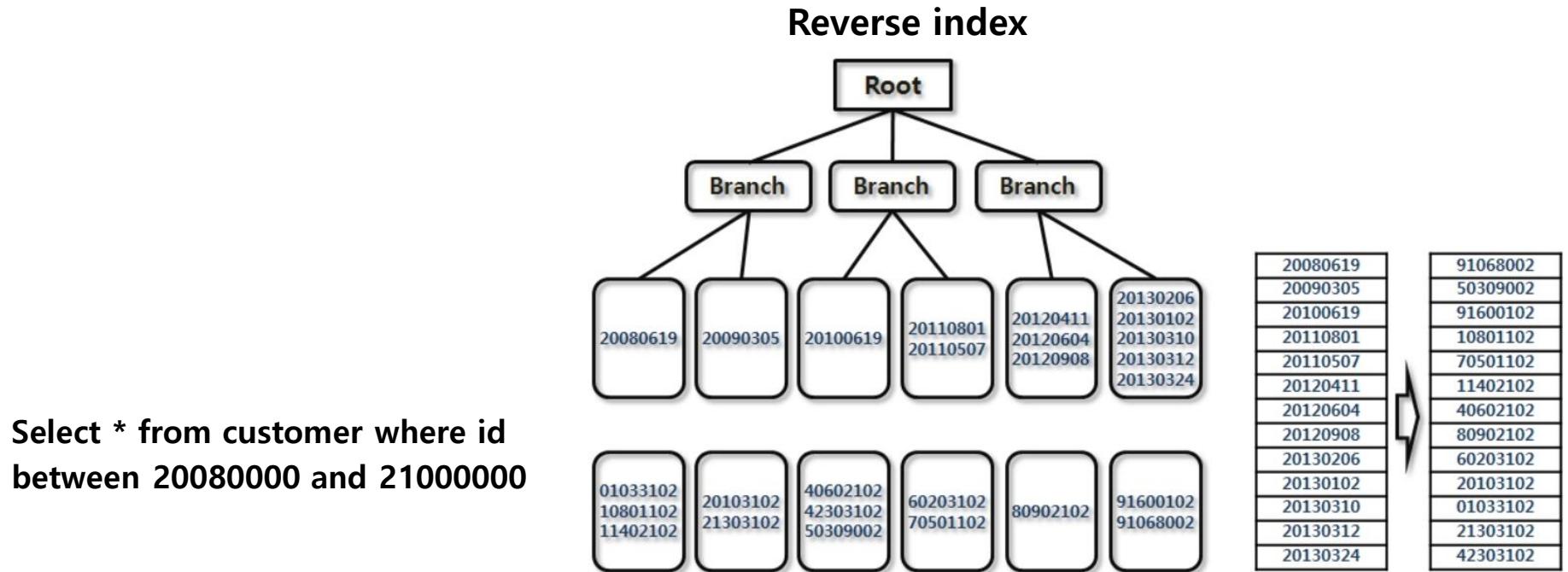
Select * from customer where name like '김%'



<http://wiki.gurubee.net/display/CORE/buffer+busy+waits>

Buffer busy waits 해결 방안

- 대량 범위의 I/O를 수행하는 SQL 튜닝
- Hash Partition등으로 Block 내부의 데이터를 hot block을 회피 할 수 있도록 재 정렬
- Reverse Index는 Index Block의 Buffer Busy Waits 자체는 해결할 수 있지만 range scan등이 기존과는 다른 결과를 초래하므로 **적용 금지**



free buffer waits, db file parallel write, write complete waits 개요

free buffer waits

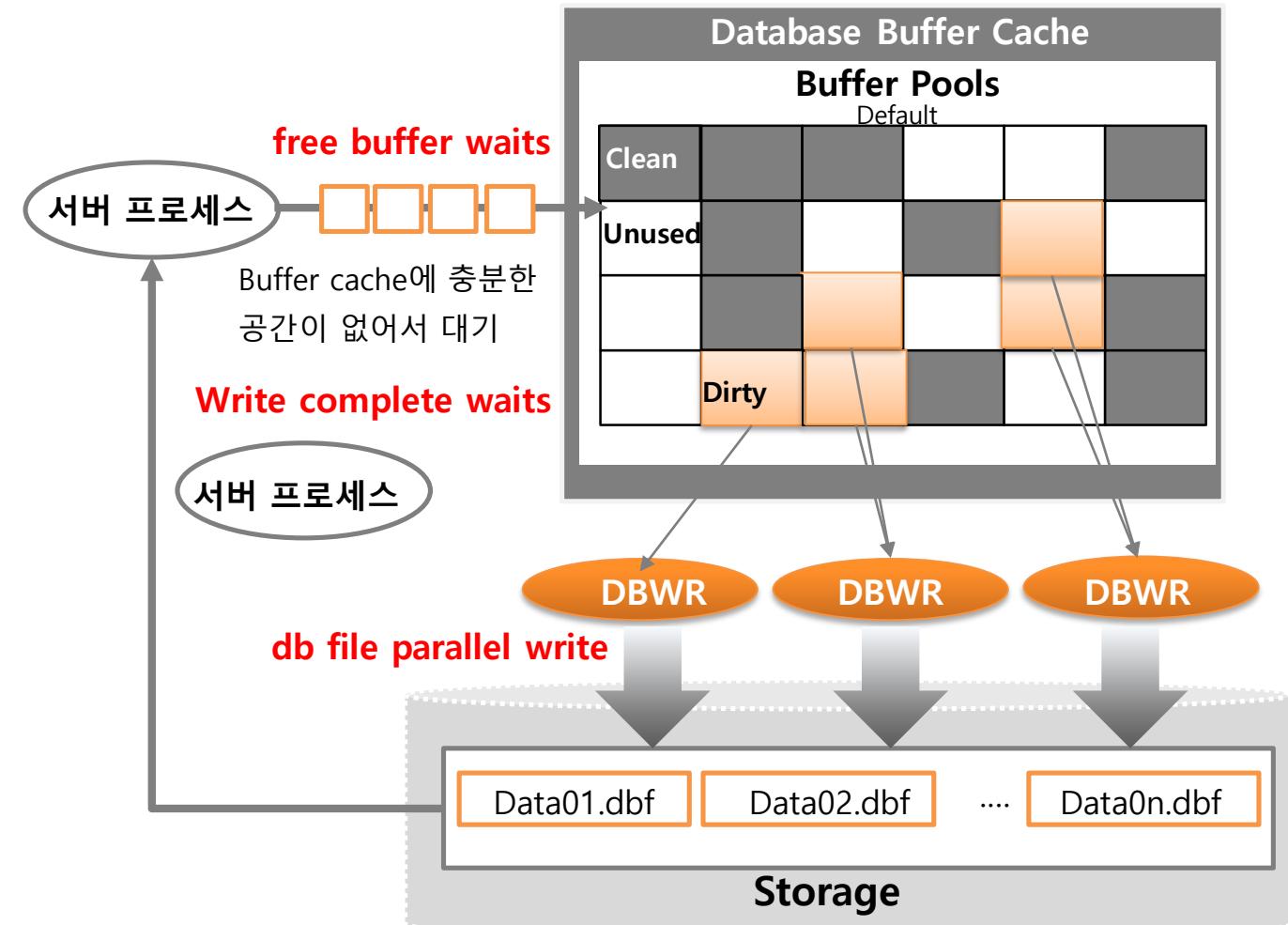
서버 프로세스가 buffer cache에 데이터를 로드하려 했지만 free buffer가 부족하여 DBWR에게 Dirty Buffer 를 디스크에 Write하여 비워 줄 것을 요청하면서 대기하는 Wait

db file parallel write

DBWR이 Dirty block을 Storage의 데이터 파일에 Write 수행 시 대기하는 Wait

write complete waits

DBWR이 write하고 있는 버퍼 블록을 서버 프로세스가 읽기 위해서 대기하는 Wait



free buffer waits, db file parallel write, write complete waits 개선 방안

발생 원인 및 개선 방안

1. 대량 범위를 Access하는 악성 SQL

- ✓ SQL 튜닝 및 Direct I/O로 유도 가능한지 검토

2. 느린 Storage I/O Write 성능

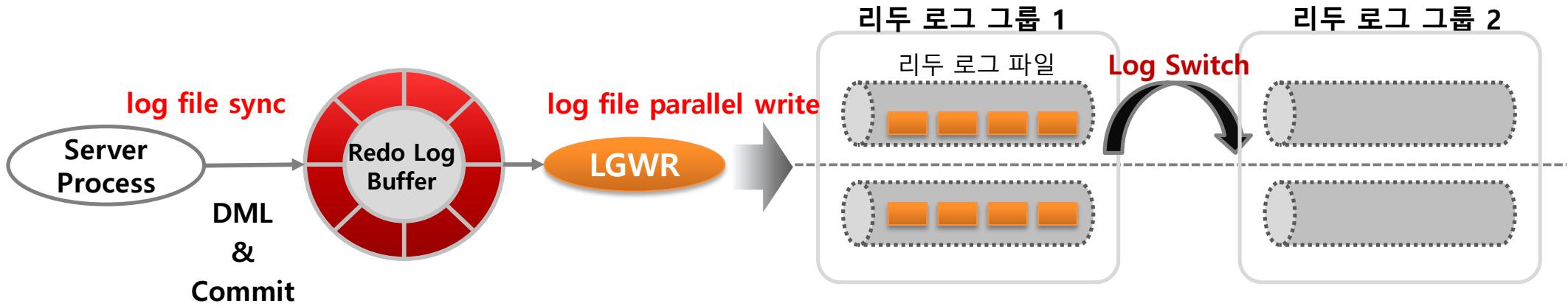
- ✓ db file parallel write 의 wait 시간 검토 후 Storage 성능 점검
- ✓ DBWR 숫자 증가(보통 Storage I/O가 Async I/O 지원되면 한 개로도 충분, db_writer_processes=CPU_COUNT/8 권장)

3. DBWR의 많은 작업량

- ✓ 잦은 Checkpoint가 발생하는지 확인(FAST_START_MTTR_TARGET이 지나치게 작을 경우 발생 가능)

4. Write complete waits, db file parallel write가 일반적으로 나타난다면 보통은 Storage I/O 성능 이슈일 가능성이 높음

log file sync와 log file parallel write 개요



Log file sync

- Server process가 DML을 수행 후 Commit을 적용하게 되면 Redo Log Buffer 적용된 변경 사항을 LGWR를 통해 리두 로그 파일에 Write 요청할 때 발생하는 Wait
- 주로 매우 빈번한 commit 이 발생하거나 redo log file의 I/O 시스템 성능이 좋지 않을 때 발생.

Log file parallel write

- Server Process의 요청에 따라 LGWR이 Redo Log file을 Write 할 때 발생하는 Wait Event(LGWR이 발생시키는 Wait). Log file sync와 마찬가지 이유로 주로 발생함.

log file sync와 log file parallel write 개선 방안

발생 원인 및 개선 방안

1. 너무 빈번한 Commit으로 발생. Commit 단위를 줄이자

너무 상세하게 업무적으로 분리된 Transaction별로 commit을 수행한다면 이들을 통합하여 하나의 Transaction으로 통합 고려

Transaction 1:

```
Update customer set name='Kwon' where id='aaa'; commit
```

Transaction 2:

```
Update customer_address set addr='Kyounggi' where id='aaa'; commit;
```



Transaction :

```
Update customer set name='Kwon' where id='aaa';
```

```
Update customer_address set addr='Kyounggi' where id='aaa'; commit;
```

PL/SQL For loop내에서 1건당 Commit을 한다면 여러건을 한번에 Commit으로 수정

```
Cursor c1 IS select id, name from customer ;
```

```
commit_interval number := 0
```

```
Begin
```

```
For cus_rec in c1:
```

```
Loop
```

```
    update customer_address set addr='....' where id = cur_rec.id;
```

```
    commit_interval := commit_interval + 1;
```

```
    if MOD(commit_interval, 100) = 0 then commit;
```

```
    end if
```

```
End Loop
```

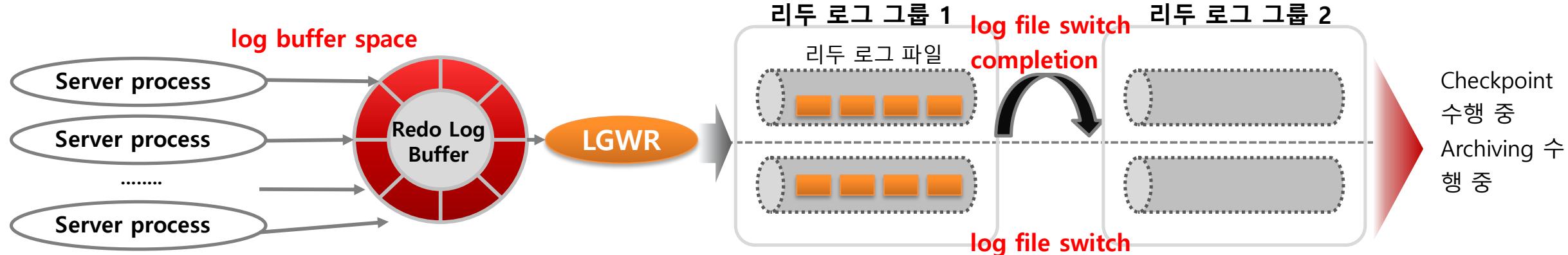
log file sync와 log file parallel write 개선 방안

발생 원인 및 개선 방안

2. I/O Write 성능 이슈

- RAID 5 일 경우 DML이 많은 시스템은 log file sync이 빈번하게 발생하여 성능 영향을 끼칠 가능성이 높음. DML 이 많은 시스템은 RAID 5 보다는 RAID 1 + 0 구성 권고
- Storage DR을 Sync 레벨로 구성한 경우 DR 시스템의 거리에 따라 log file sync가 빈번하게 발생하여 성능 영향 가능성
- 리두 버퍼 크기가 너무 클 경우 기록해야 할 데이터가 늘어나 log file sync 대기시간 증가 가능.

log buffer space, log file switch completion, log file switch 개요



log buffer space

- 서버 프로세스들이 매우 많은 DML Transaction을 발생 시키면 서 빠르게 Redo log buffer를 점유하는 상황에서 주로 발생.
- LGWR이 리두 로그 파일에 write를 충분히 완료하지 못한 상태에서도 계속적으로 Redo log buffer에 데이터가 쌓이면서 서버 프로세스가 필요한 Redo log buffer 공간을 할당 받지 못하면서 이를 얻기 위해 대기 하는 Wait

Log file switch completion과 log file switch

- log file switch completion은 서버프로세스가 redo를 redo log buffer를 거쳐서 리두 로그 파일에 write할 경우 해당 리두 로그 파일이 가득차게 되면 log switch를 하게 되면서 대기하는 Wait
- log file switch는 log file switch completion으로 대기하는 와중에 다음으로 write될 리두 로그 파일이 이전에 수행했던 다른 작업으로 여전히 busy하여 리두를 write하지 못하고 대기하는 wait

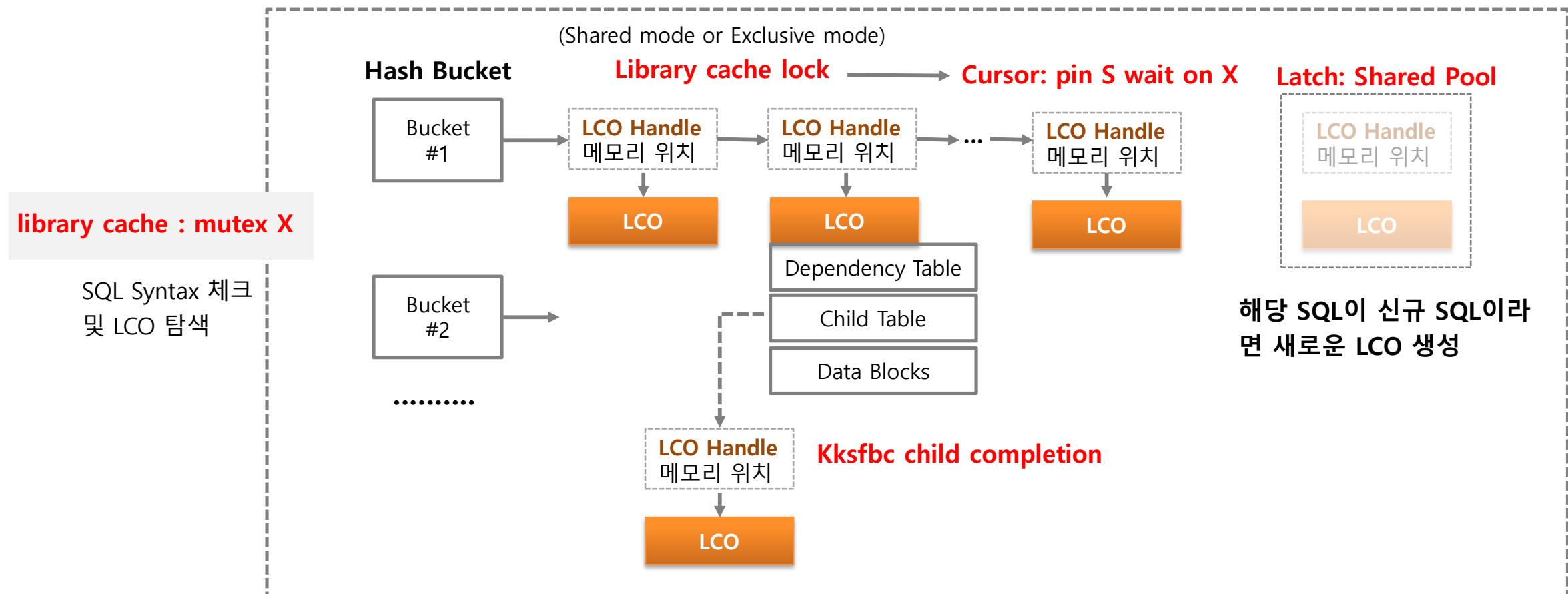
log buffer space, log file switch completion, log file switch 개선 방안

발생 원인 및 개선 방안

1. Log buffer space의 경우 redo log buffer 크기가 너무 작으면 심각하게 발생할 수 있음. redo log buffer 사이즈를 10M 이상 증가
2. I/O 성능을 개선
3. Log file switch completion/log file switch의 경우 redo log file 크기를 증가 시킴

Library cache 주요 Wait Event 개요

SELECT * FROM CUSTOMER WHERE ID = 'XXXX'



Library cache 주요 Wait Event 발생 원인 및 개선 방안

발생 원인 및 개선 방안

1. Dynamic/Literal SQL 빈도가 높은지 확인

- ✓ 반드시 Static SQL 적용(변환이 어려울 경우 CURSOR_SHARING=FORCE 적용)

2. Shared Pool 크기 증가가 필요한지 확인

- ✓ Library Cache, Data Dictionary Cache의 Hit Ratio는 95%~99%이상을 유지할 수 있도록 노력
- ✓ V\$SHARED_POOL_ADVICE 등을 참조하여 적정 Shared Pool 크기 산정

3. 동시 접속 부하가 많은 업무 시간 중에 DDL 수행 금지

- ✓ ALTER TABLE, 특히 INDEX 생성, Rebuild 등의 작업은 동시 접속 부하가 많을 시에는 금지

4. 크기가 큰 PL/SQL 패키지는 Age out 되지 않게 pinning 고려

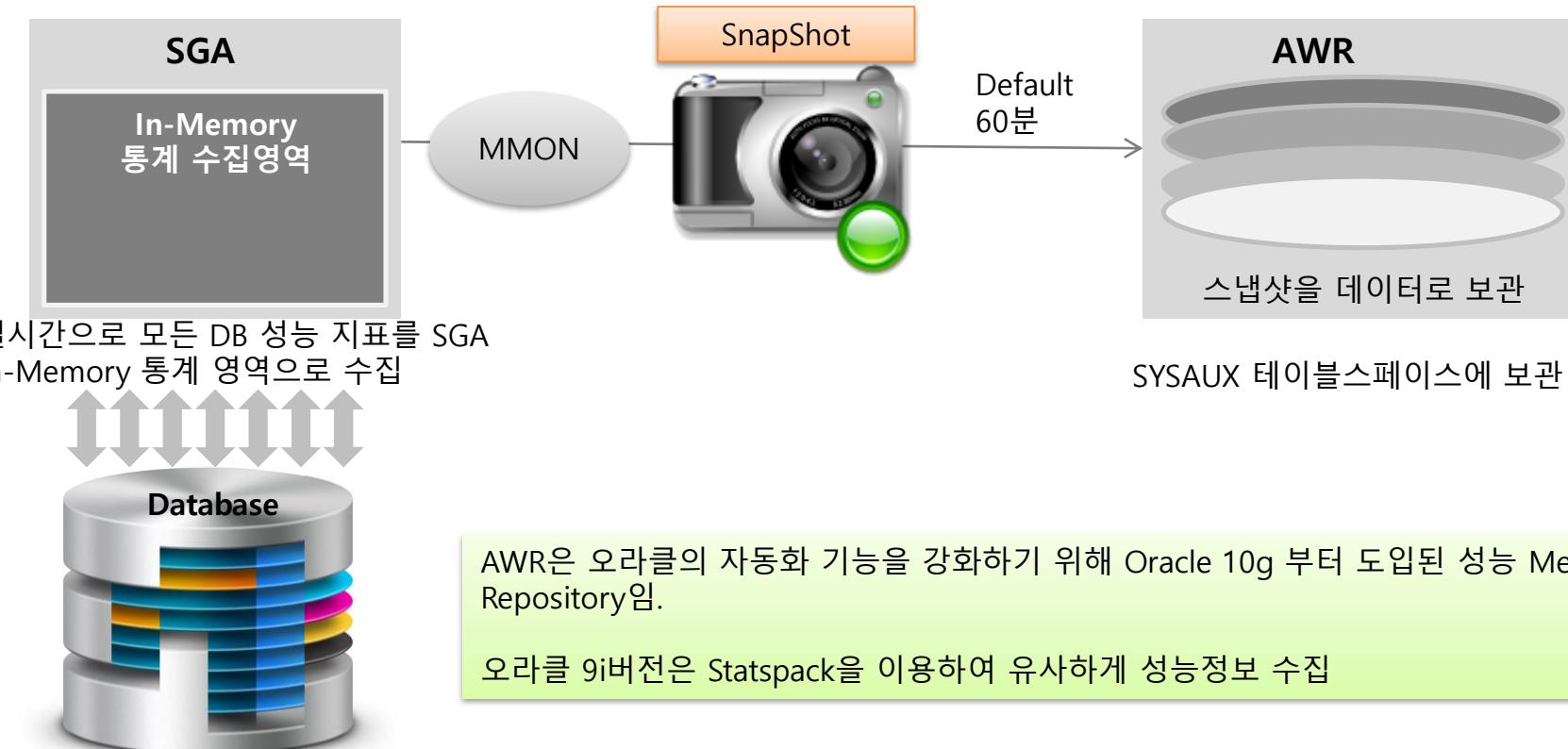
- ✓ execute dbms_shared_pool.keep('패키지명')

5. 버그 등을 의심

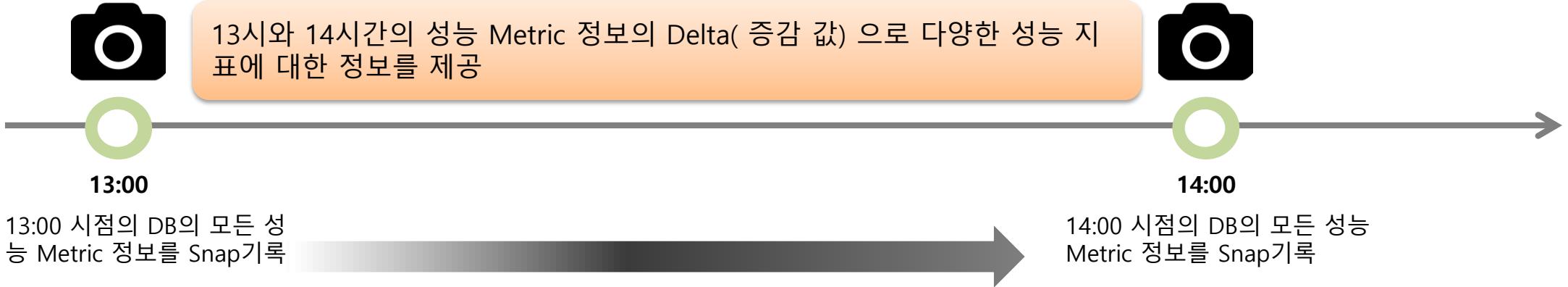
AWR, ADDM과 Statspack의 이해와 활용

AWR(Automatic Workload Repository)

- 성능 정보에 대한 내장 Repository
- Default 60 분마다 데이터베이스 Metrics의 Snapshot을 생성하여 Default 8일간 보관
- 모든 Oracle 자체 관리기능 (Self Management) 의 기본 자료가 됨.**
(ADDM , SQL Tuning Advisor , Undo Adviser , Segment Adviser)



AWR 의 작동

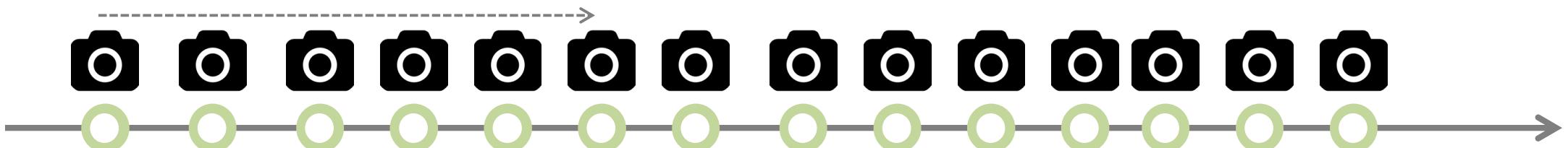


Physical reads : 10,000,000

Physical reads : 15,000,000

13시에 Physical reads가 천만 , 14시에 Physical Reads가 천오백만 이면 한 시간동안에 약 5백만 블록을 physical read 했다는 것이며 초당 1,388 블록 (5백만/3600초) physical reads 수행

생성된 AWR Snapshot을 이용하여 어느 시간 구간에서도 성능 분석이 가능함(시간,일,주,월)



AWR을 구성하는 DBA_HIST_XXX

Wait

DBA_HIST_WAITSTAT
DBA_HIST_SYSTEM_EVENT
DBA_HIST_BG_EVENT_SUMMARY

Snapshot 기준

DBA_HIST_SNAPSHOT
DBA_HIST_DATABASE_INSTANCE

Stat

System

DBA_HIST_SYSSTAT
DBA_HIST_SYS_TIME_MODEL

SGA

DBA_HIST_SGA
DBA_HIST_SGASTAT
DBA_HIST_PGASTAT
DBA_HIST_LIBRARY_CACHE
DBA_HIST_BUFFER_POOL_STAT

Advice

DBA_HIST_DB_CACHE_ADVICE
DBA_HIST_PGA_TARGET_ADVICE
DBA_HIST_SHARED_POOL_ADVICE
DBA_HIST_JAVA_POOL_ADVICE

SQL

DBA_HIST_SQLSTAT
DBA_HIST_SQLTEXT
DBA_HIST_SQL_SUMMARY

Latch & Enqueue

DBA_HIST_LATCH
DBA_HIST_ENQUEUE_STAT

Segment, Tablespace, Datafile I/O

DBA_HIST_SEG_STAT
DBA_HIST_TABLESPACE
DBA_HIST_UNDOSTAT
DBA_HIST_ROLLSTAT
DBA_HIST_DATAFILE
DBA_HIST_TEMPFILE

AWR Report 항목

AWR Report 의 다양한 항목

Report Summary

Wait Event Statistics

SQL Statistics

Instance Activity Statistics

IO Stats

Buffer Pools Statistics

Advisory Statistics

Wait Statistics

Undo Statistics

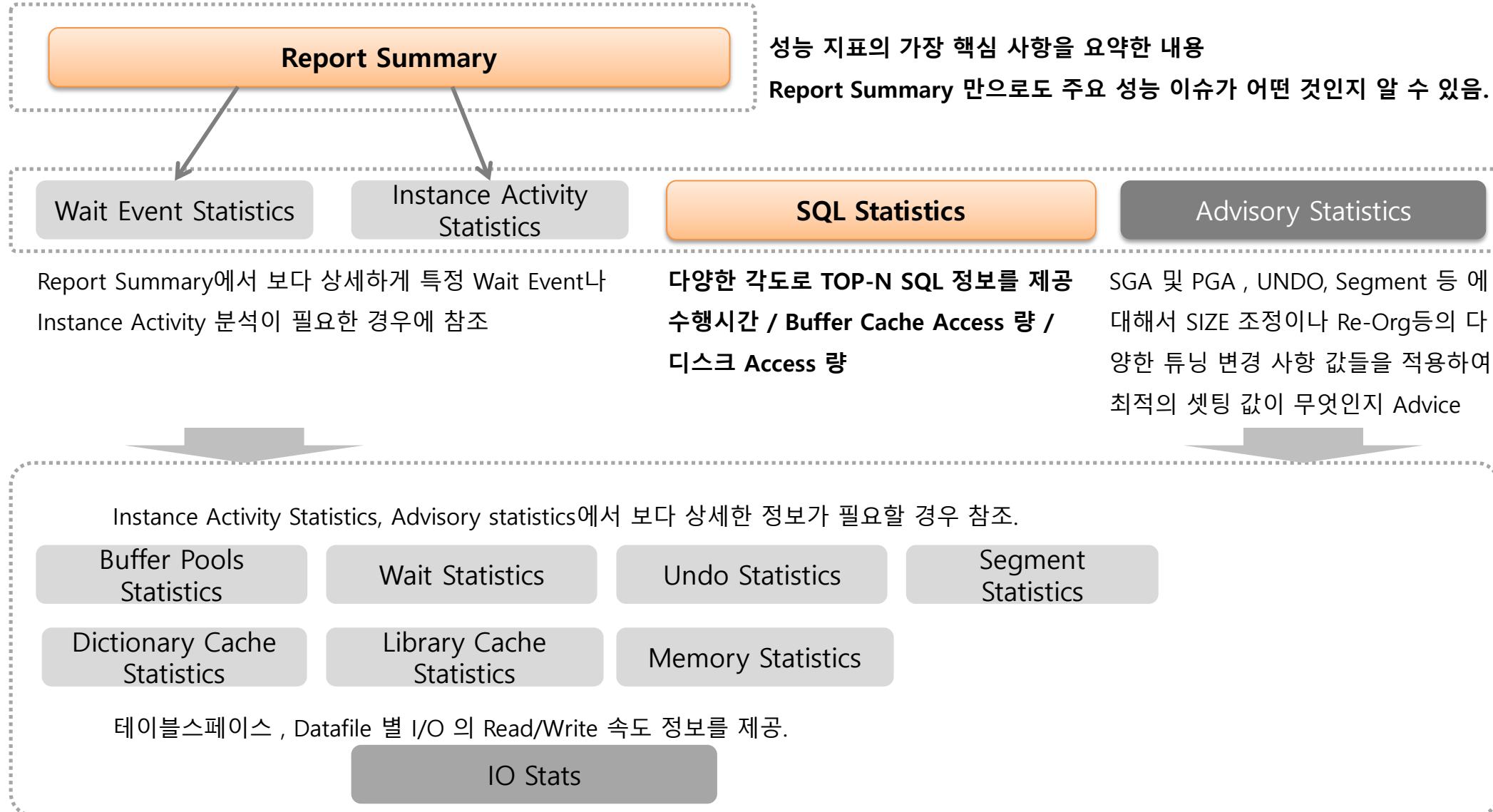
Segment Statistics

Dictionary Cache Statistics

Library Cache Statistics

Memory Statistics

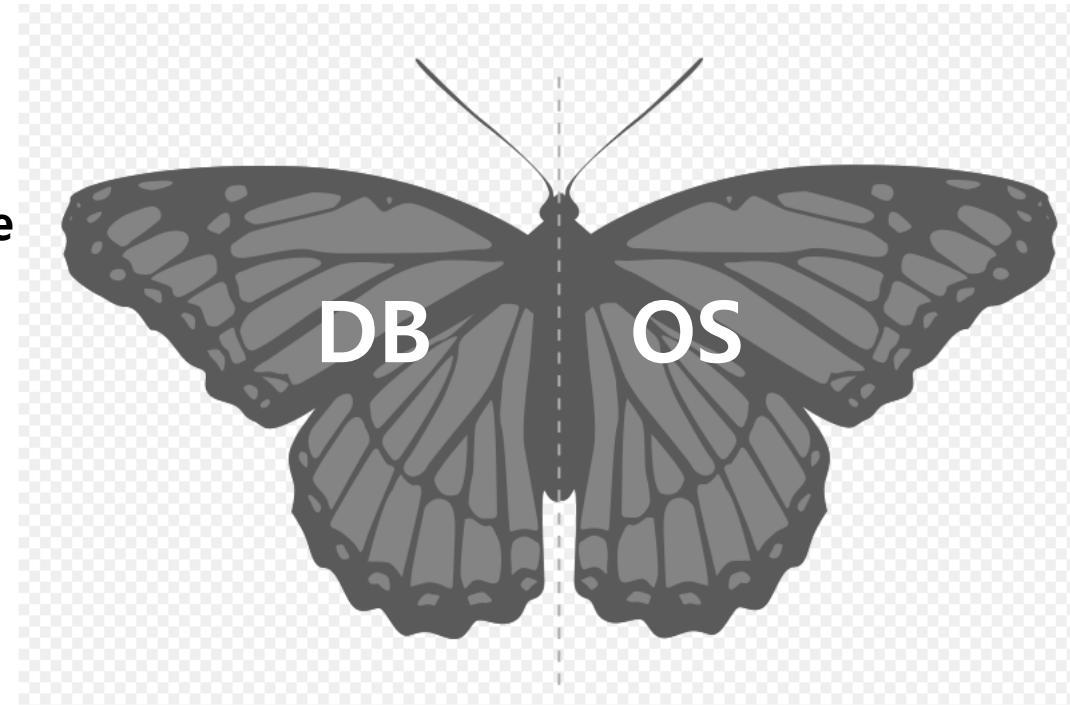
AWR Report 항목 계층별 구성



DB의 성능지표와 OS 성능지표 이해

DB의 성능 지표인 Load Profile 과 Wait Event가 어떻게 시스템의 CPU Usage 와 Wait로 연계되어 있는지 직관적인 감을 유지하는 것이 필요.

Load Profile
+
Wait Event



CPU 부하(sar, vmstat)
사용량(Usr/Sys) , Wait

Wait Event 와 Load Profile의 결합 분석

- **Wait Event의 경우 CPU Time 대비 다른 Wait Event가 차지하는 비중이 높으면 문제가 있는 시스템으로 간주하기 쉽다.** 그러나 시스템 전체적인 부하가 적은 상태에서의 이와 같은 Wait Event 분포도는 극소수의 Application/SQL이 상대적으로 자원을 많이 사용하면서 발생할 수 있으며 시스템 전체적으로는 대부분 큰 문제가 아니다.
- 따라서 Wait Event의 분포도만으로 시스템에 문제가 있다고 판단해서는 안되며, 반드시 Load Profile의 메모리/디스크 Read/Write, Transaction/수행 횟수 등 지표 분석을 함께 수행해야 한다.
- 또한 DB Wait Event와 Load Profile의 분석을 통해 판단된 결과와 현 시스템의 CPU 사용% (Usr , Sys)과 Wait % 수치에 대해서 결합하여 전체 시스템의 성능 이슈에 대한 직관적인 판단 능력을 갖추는 것이 필요하다.

Top 5 Timed Events

Event	Wait 횟수	시간(s)	시간 %
CPU time		1657	22%
db file sequential read	1536882	2347	32%
read by other session	1.80%

Load Profile

	초당 수치	트랜잭션당 수치
Redo size
Logical reads	10786.6
.....
Physical reads	564.3
Transactions

Wait Event에서 db file sequential read 가 32%로 CPU 보다 많은 Elapsed time을 소모하였음. 그러나 Load Profile의 Logical Read, Physical Read가 그다지 높은 수준이 아니기 때문에 시스템 전체적으로 큰 성능 문제가 없는 것으로 판단가능하며 CPU 사용량과 비교하면 더욱 분명하다.

Wait Event 와 Load Profile의 결합 분석

- Load Profile에서 중요한 지표는 Logical Reads(초당 Buffer Cache Access 블록 개수) 와 Physical Reads (초당 Disk Access 블록 개수) I/O 수치이다.
- 이들 I/O 수치는 Access 블럭수만 나타내므로 Scan Access 냐 Random Access 냐에 따라 수치가 부여하는 가중치가 매우 달라진다.
- 즉 Physical Reads 가 초당 10,000 블록을 Access 하더라도 I/O Access 유형에 따라 시스템 성능에 큰 이슈가 될만한 수치인지 아닌지 달라진다.
- 일반적으로 High-End 급 서버 구성에서 Physical reads 가 초당 4,500 블록 이상이 거의 Random Access라면 시스템은 정상적인 운영이 힘든 수준의 고 부하를 감당해야만 한다. (예측컨데 CPU 부하는 90% 이상 , 이 중 30~40%는 Wait 일 것임. 단 Storage를 SSD로 도배하지 않은 경우에 한함.). 그러나 만일 초당 4,500 블록 중 Scan Access 가 어느 정도 있다면 해당 시스템은 별 문제 없이 안정성 있게 운영중일것이다.
- Physical reads의 수치중에서 Scan과 Random의 비율(Big I/O vs Small I/O)을 정확히 알수 있는 방법은 없다. (Exadata에서는 Storage 서버에 질의하여 가능). 그러나 Wait Event와 연계하여 직관적으로 판단은 가능하다.
- **이들 수치는 DB 시스템을 구성하는 H/W 용량이나 구성에 따라 시스템이 포용 가능한 안정성 있는 수치의 범위가 달라지기 때문에 절대치 기준은 없다. 다양한 시스템을 분석한 뒤 자신만의 경험 포토폴리오를 구축하여 해당 수치를 판단하는 것이 중요.**

Physical Reads : 10,000 block

Db file Sequential read : 30%
Db file scattered read : 5%

AWR Report 분석 예제 – Load Profile & Wait

Load Profile

	Per Second	Per Transaction
Redo size:	213,917.17	6,110.08
Logical reads:	124,930.95	3,568.38
Block changes:	1,079.78	30.84
Physical reads:	3,052.05	87.18
Physical writes:	87.42	2.50
User calls:	4,923.28	140.62
Parses:	1,784.56	50.97
Hard parses:	122.70	3.50
Sorts:	303.56	8.67
Logons:	17.36	0.50
Executes:	2,103.00	60.07
Transactions:	35.01	

현 시스템은 Logical I/O, Physical I/O의 사용량이 많은 시스템 인가 ?

시스템의 H/W 등을 감안할 때 Physical read의 량은 상당수준 높은 편
이며 특히 Logical Reads가 매우 많음. 이를 감안할 때 자주 수행되는
SQL이 많은 Block을 Access하여 사용됨을 알 수 있음.

Top 5 Timed Events

Event	Waits	Time (s)	Avg wait (ms)	%Total	Call Time	Wait Class
CPU time		36,075		27.3		
db file sequential read	12,876,955	21,583	2	16.3	User I/O	
log file sync	441,853	14,898	34	11.3	Commit	
gc cr grant 2-way	6,042,849	11,147	2	8.4	Cluster	
gc cr block busy	100,835	5,866	58	4.4	Cluster	

DB Wait Event 분석 결과 성능상의 이슈는 무엇인가?

Random I/O 시 발생하는 DB file sequential read가 일정 수준 이상 존재함.
Random I/O에 대한 보강 필요.
Log file sync 가 상당수준 이상으로 SQL 수행 성능에 영향을 미치고 있음.

성능 분석을 위한 AWR Report 활용 방법론



Load Profile
DB에 얼마나 많은 User들이 DB Call 을 하여 얼마나 많은 Logical (Memory) Reads/Write , Physical(Disk) Reads/Write, Transaction 등을 알 수 있는 수치 제공

Top Wait Events
DB의 Top Wait Event 정보를 제공 각 Wait Event의 사용 % , 총 시간 , 평균 시간 등의 수치 제공

SQL Statistics
어떠한 SQL 이 어느 정도의 I/O 자원을 사용하는지 수행시간은 얼마나 걸리는지에 대한 정보 제공

수행시간 / Buffer Cache Access 량 / 디스크 Access 량에 따른 TOP-N 제공



부하가 Heavy 한 시스템인가 ? 그렇지 않은 시스템인가? 판단의 근거 제공



어떠한 Wait Event가 시스템 성능에 가장 큰 영향을 주고 있는지 파악하여 성능 향상을 위해서 어떠한 요소를 개선해야 할지 파악 가능



Wait Event가 I/O 관련일 경우 Buffer Cache 또는 Disk I/O 를 과다 사용하는 SQL 대한 추적 분석 및 튜닝 수행

OS 성능 지표와 결합되면 더욱 효율적 분석 가능

AWR Report 분석 – Load Profile & Wait Event

Load Profile은 DB에 어떤 유형의 부하가 얼마만큼 가해지고 있는지 정보를 제공하며 이를 통해 어떠한 유형의 Application이 주로 구동되어 있는지 또한 알 수 있음

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	1.7	0.0	0.00	0.00
DB CPU(s):	1.0	0.0	0.00	0.00
Background CPU(s):	0.1	0.0	0.00	0.00
Redo size (bytes):	645,878.0	4,226.4		
Logical read (blocks):	144,775.8	947.4		
Block changes:	4,001.8	26.2		
Physical read (blocks):	773.8	5.1		
Physical write (blocks):	360.1	2.4		
Read IO requests:	769.1	5.0		
Write IO requests:	186.8	1.2		
Read IO (MB):	6.1	0.0		
Write IO (MB):	2.8	0.0		
IM scan rows:	0.0	0.0		
Session Logical Read IM:	0.0	0.0		
User calls:	791.7	5.2		
Parses (SQL):	289.1	1.9		
Hard parses (SQL):	0.3	0.0		
SQL Work Area (MB):	3.1	0.0		
Logons:	0.1	0.0		
User logons:	0.1	0.0		
Executes (SQL):	4,150.1	27.2		
Rollbacks:	0.0	0.0		
Transactions:	152.8			

Per Second: 1초당 지표

Per Transaction: Transaction(Commit, Rollback)당 평균 지표

- Redo size(bytes): redo log 의 증가 크기(bytes)
- Logical reads(blocks): Buffer cache read block 수
- Block changes: (DML로) 변경된 block수
- Physical read(blocks): Storage I/O read block 수
- Physical write(blocks): Storage I/O write block 수
- Read IO Requests: Read I/O 요청 횟수
- Write IO Request: Write I/O 요청 횟수
- Read IO(MB): Read I/O 크기
- Write IO(MB): Write I/O 크기
- User Calls: 클라이언트가 요청한 call 수(Parse, Execute 포함)
- Parses(SQL): Soft Parse + Hard Parse
- Executes(SQL): SQL 실행 횟수
- Rollbacks: Rollback 횟수
- Transactions: Rollback + Commits

Load Profile & Wait Event 사례

Case 1

오전 09 ~ 10 시

Load Profile

	Per Second	Per Transaction
Redo size:	24,613.52	3,973.88
Logical reads:	145,422.59	23,478.63
Block changes:	140.39	22.67
Physical reads:	127.63	20.61
Physical writes:	13.48	2.18
User calls:	18.83	3.04
Parses:	13.41	2.16
Hard parses:	0.07	0.01
Sorts:	3.25	0.53
Logons:	0.02	0.00
Executes:	53.19	8.59
Transactions:	6.19	

Case 2

오전 09 ~ 10 시

Load Profile

	Per Second	Per Transaction
Redo size:	23,326.98	1,376.91
Logical reads:	540,874.37	31,926.02
Block changes:	139.67	8.24
Physical reads:	16,026.30	945.98
Physical writes:	9.29	0.55
User calls:	679.19	40.09
Parses:	199.96	11.80
Hard parses:	14.50	0.86
Sorts:	2,317.07	136.77
Logons:	13.26	0.78
Executes:	473.84	27.97
Transactions:	16.94	

Load Profile & Wait Event 사례

Case 3

오전 09 ~ 10 시

Load Profile

	Per Second	Per Transaction
Redo size:	504,916.64	110,039.79
Logical reads:	126,779.55	27,629.90
Block changes:	1,437.36	313.25
Physical reads:	2,944.38	641.69
Physical writes:	85.84	18.71
User calls:	1,675.93	365.25
Parses:	301.46	65.70
Hard parses:	8.68	1.89
Sorts:	125.36	27.32
Logons:	0.24	0.05
Executes:	833.01	181.54
Transactions:	4.59	

Load Profile & Wait Event 사례

SQL ordered by Reads

- Total Disk Reads: 10,529,464
- Captured SQL account for 73.4% of Total

Physical Reads	Executions	Reads per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
2,440,742	65	37,549.88	23.18	196.90	576.86	4yrt8qrc761bm	JDBC Thin Client	Com.batch1
1,210,861	65	18,628.63	11.50	98.94	307.96	a315bs2jn8z2f	JDBC Thin Client	Com.batch2
1,195,983	65	18,399.74	11.36	97.89	306.27	dtz66xntrv1un	JDBC Thin Client	Com.batch3
522,765	1	522,765.00	4.96	60.80	222.43	fp5dh75qn5x3m		
419,441	34	12,336.50	3.98	66.32	145.94	3c7cdsarqcb1g	JDBC Thin Client	
396,144	3	132,048.00	3.76	125.53	1161.32	gjx81wpwyj5gm		
210,856	5	42,171.20	2.00	50.75	227.99	b1282827nnkvt	JDBC Thin Client	
203,504	5	40,700.80	1.93	49.84	221.28	5pkcx0dkvyq5z	JDBC Thin Client	
158,203	25	6,328.12	1.50	49.27	417.38	90tgjk30g94wr	JDBC Thin Client	
138,327	32	4,322.72	1.31	33.20	1029.77	2jc6ca8jaab6	JDBC Thin Client	
115,300	11	10,481.82	1.10	58.46	85.34	g01dxv9xu96xq		

Select 유형 SQL

AWR Report 분석 – SQL Statistics

- [SQL ordered by Elapsed Time](#) (수행 시간)
- [SQL ordered by CPU Time](#) (CPU 시간)
- [SQL ordered by User I/O Wait Time](#) (User I/O Wait 시간)
- [SQL ordered by Gets](#) (Buffer Cache access block 수)
- [SQL ordered by Reads](#) (Physical(Storage) access block 수)
- [SQL ordered by Physical Reads \(UnOptimized\)](#)
- [SQL ordered by Executions](#) (호출/수행 횟수)
- [SQL ordered by Parse Calls](#) (Parsing 횟수)
- [SQL ordered by Sharable Memory](#) (shared pool 점유)
- [SQL ordered by Version Count](#)
- [Complete List of SQL Text](#)

AWR Report 분석 – SQL Statistics

Buffer Cache Access TOP-N SQL

SQL Order by Gets (Gets는 성능지표에서 Buffer Cache Access Block 수를 의미)

Buffer Gets	수행 횟수	Gets Per Exec	% Total	CPU Time (s)	Elapsed Time(s)	SQL
132,136,479	14,535	9,090.9	14.7	2329.05	3830.50	SELECT /*+ index (.....) */ brk_cd, FROM xxxx WHERE imp_dcl_xxx= 'B' AND imp_dcl_date <= TO_CHAR(SYSDATE,'YYYYMMDD') AND (length(.....o) <> 10 OR like 'X%') AND rownum = 1
57,126,766	20,417	2,798.0	6.4	334.53	419.82	SELECT FROM xxxx WHERE rownum = 1
39,544,033	30	1,318,134.4	4.4	168.63	237.33	select /*+ index(a xxxx) */ nvl(max('Y'), 'N') as gubun from xxxxxx a where a.prn_no in (select prn_reg_no from yyyy where brk_cd = :1)

자주 사용되는 SQL 들이 매우 많은 Buffer Cache를 차지하고 있으므로 상대적으로 다른 SQL들이 Buffer Cache를 사용할 확률이 적어지며 이로 인해 전반적인 SQL의 수행속도가 저하됨.

자주 호출되지 않는 특정 SQL도 매우 많은 Buffer Cache를 사용하고 있으므로 이들 SQL에 대한 튜닝 필요.

Instance Efficiency Percentages

- Buffer Cache, Library Cache, PGA(Sort), Redo, Parsing, Latch 등의 요소에 대한 주요 Ratio 지표를 축약하여 나타냄

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.08	In-memory Sort %:	100.00
Library Hit %:	99.94	Soft Parse %:	99.08
Execute to Parse %:	93.38	Latch Hit %:	99.99
Parse CPU to Parse Elapsd %:	71.43	% Non-Parse CPU:	99.53
Flash Cache Hit %:	0.00		

지표	설명
Buffer Nowait %	서버 프로세스가 Buffer cache 에 접근하기 위해 Wait하지 않은 비율. 99% 이상의 수치 유지. 만약 낮다면 Hot block 경합에 따른 Wait 의심(주로 Buffer busy)
Redo NoWait %	Redo (buffer+redo log) 접근 시 Wait 하지 않은 비율. 99% 이상의 수치 유지. 만약 낮다면 Redo 관련 Wait 경합 의심
Buffer Hit %	Buffer Cache Hit Ratio. OLTP에서 95% 이상 유지. 이슈 시 SQL 튜닝 또는 Buffer 크기 증가
In-memory Sort %	Storage가 아닌 PGA내의 SORT AREA 메모리 내에서 Sorting된 비율. OLTP에서 95% 이상 유지. 이슈 시 PGA 크기 증대
Library Hit %	Library Cache에서 밀려나지 않고 최초 로딩이 아닌 경우)이 아닌 비율. OLTP에서 95% 이상 유지.

Instance Efficiency Percentages

- Buffer Cache, Library Cache, PGA(Sort), Redo, Parsing, Latch 등의 요소에 대한 주요 Ratio 지표를 축약하여 나타냄

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.08	In-memory Sort %:	100.00
Library Hit %:	99.94	Soft Parse %:	99.08
Execute to Parse %:	93.38	Latch Hit %:	99.99
Parse CPU to Parse Elapsd %:	71.43	% Non-Parse CPU:	99.53
Flash Cache Hit %:	0.00		

지표	설명
Soft Parse %	Soft Parsing 비율. 95% 이상 유지 노력. 이슈 시 Shared Pool 크기를 키우거나, Soft Parsing 유도
Execute to Parse %	1 - Parsing 대비 Execute 비율
Latch Hit %	Latch 획득 시 대기(Latch Free)하지 않은 비율. 99% 이상 유지
Parse CPU to Parse Elapsed	SQL 파싱 시 전체 수행 시간 대비 CPU 사용 시간 비율. 100 - Parsing 시 I/O Wait 등으로 Wait 비율. 이슈시 Shared Pool 크기 증가
% Non-Parse CPU	Parsing하는데 사용되지 않은 CPU 비율. 즉 100 - CPU가 SQL Parsing에 소모한 비율 이슈 시 Shared Pool 크기를 키우거나, Soft Parsing 유도

Advisory Statistics

- [Buffer Pool Advisory](#)
 - [PGA Aggr Summary](#)
 - [PGA Aggr Target Stats](#)
 - [PGA Aggr Target Histogram](#)
 - [PGA Memory Advisory](#)
 - [Shared Pool Advisory](#)
 - [SGA Target Advisory](#)
 - [Streams Pool Advisory](#)
 - [Java Pool Advisory](#)
- 주로 SGA와(상세 컴포넌트 별) PGA 메모리 크기 Advise
 - V\$XXX_ADVICE 와 유사

AWR Report 분석 예제 - Advisory Statistics

PGA Memory Advisory

PGA Memory Advisory DB/Inst: KCSDB/KCSDB1 Snap: 448
-> When using Auto Memory Mgmt, minimally choose a pga_aggregate_target value where Estd PGA Overalloc Count is 0

PGA Target Est (MB)	Size Factr	W/A MB Processed	Estd Extra W/A MB Read/ Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count
500	0.1	13,502,241.6	10,531,614.1	56.0	202,540
1,000	0.3	13,502,241.6	10,126,785.1	57.0	186,586
2,000	0.5	13,502,241.6	372,147.2	97.0	14
3,000	0.8	13,502,241.6	91,833.5	99.0	0
4,000	1.0	13,502,241.6	84,562.1	99.0	0
4,800	1.2	13,502,241.6	74,372.1	99.0	0
5,600	1.4	13,502,241.6	74,372.1	99.0	0
6,400	1.6	13,502,241.6	74,372.1	99.0	0
7,200	1.8	13,502,241.6	74,372.1	99.0	0
8,000	2.0	13,502,241.6	74,372.1	99.0	0
12,000	3.0	13,502,241.6	74,372.1	99.0	0
16,000	4.0	13,502,241.6	74,372.1	99.0	0
24,000	6.0	13,502,241.6	74,372.1	99.0	0
32,000	8.0	13,502,241.6	74,372.1	99.0	0

현 PGA 메모리 크기를 기준으로 Size를 늘리거나 줄였을 때의 메모리 Hit 영향도를 비교하여 나타냄

PGA Aggregate Target

항목	의미	관련컬럼
PGA Target Est (MB)	PGA 예상 크기(단위:MB)	PGA_TARGET_FOR_ESTIMATE / 1024 / 1024
Size Factor	현재 설정된 PGA 크기 대비 예상 PGA 크기 비율	PGA_TARGET_FACTOR
W/A MB Processed	PGA 사용 효율 예측 시 고려된 전체 SQL Work Area	BYTES_PROCESSED / 1024 / 1024
Estd Extra W/A MB Read/Written to Disk	PGA 크기 변경 시 예측되는 Temp tablespace I/O 발생 크기(단위:MB)	ESTD_EXTRA_BYTES_RW / 1024 / 1024
Estd PGA Cache Hit %	PGA 크기 변경 시 예측되는 메모리 Hit Ratio	ESTD_PGA_CACHE_HIT_PERCENTAGE
Estd PGA Overalloc Count	PGA 크기 변경 시 예상되는 추가 메모리 할당 수	ESTD_OVERALLOC_COUNT

SGA Advisory

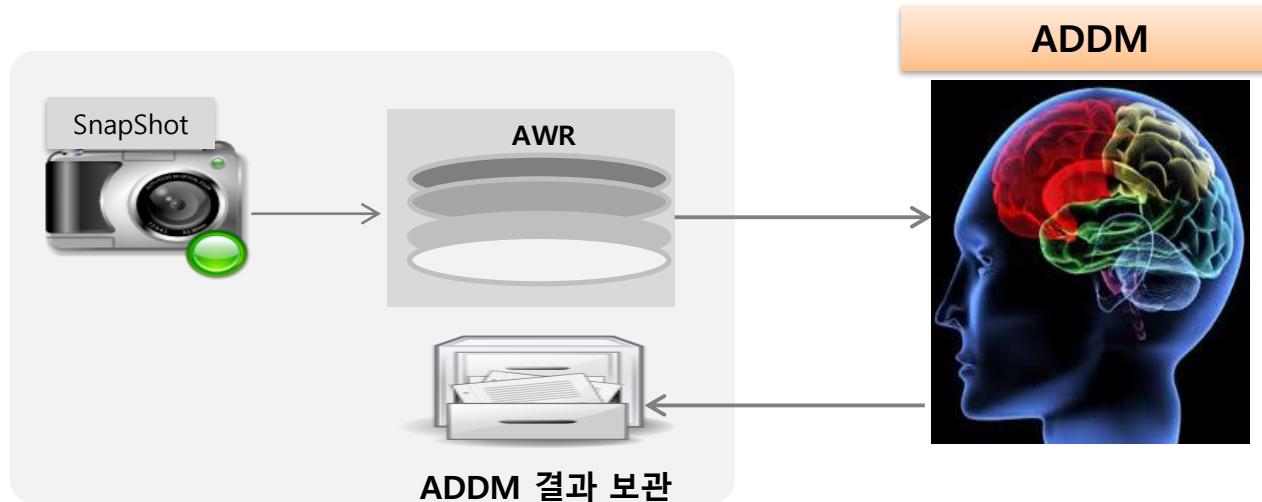
항목	의미
SGA Target Size(M)	SGA 예상 크기(단위:MB)
SGA Size Factor	현재 설정된 SGA 대비 예상 SGA 크기 비율
Est DB Time(s)	예상 DB Time(단위:초)
Est Physical Reads	예상 디스크 I/O 횟수

Shared Pool Advisory

항목	의미
Shared Pool Size(M)	공유 풀 예상 크기(단위:MB)
SP Size Factr	현재 설정된 공유 풀 대비 예상 공유 풀 크기 비율
Est LC Size(M)	라이브러리 캐시 예상 크기(단위:MB)
Est LC Mem Obj	라이브러리 캐시 오브젝트 예상 개수
Est LC Time Saved(s)	예상 파스 단축 시간(단위:초)
Est LC Time Saved Factr	현재 파스 시간 대비 예상 파스 단축 시간 비율
Est LC Load Time(s)	예상 공유 풀 적재 시간(단위:초)
Est LC Load Time Factr	현재 공유 풀 적재 시간 대비 예상 공유 풀 적재 시간 비율
Est LC Mem Obj hits	예상 라이브러리 캐시 오브젝트 적중 횟수

ADDM(Automatic Database Diagnostic Monitor)

- 사용자가 AWR Report 또는 다양한 모니터링 툴을 통해 시스템의 문제점을 분석하지 않아도 오라클이 자동으로 문제점을 분석하고 해결 방안을 제시
- 어떠한 문제점이 얼마만큼의 성능 영향을 미치고 있으며, 이를 해결하면 어느 정도의 성능 향상이 있는지 수치 제공



- Resource 병목
- I/O 영향
- SGA, PGA 메모리 권장
- 로드량이 많은 SQL
- 부실한 Oracle Net 연결 관리
- Lock 경합
- 기타 다수

- 각 AWR 스냅샷 생성 후마다 실행
- Instance를 모니터하여 병목 지점을 감지
- AWR에 결과를 저장

ADDM 활용 예제

FINDING 2: 12% impact (376753 seconds)

개선이 된다면 약 12% 성능향상 임팩트

SQL statements were not shared due to the usage of literals. This resulted in additional hard parses which were consuming significant database time.

RECOMMENDATION 1: Application Analysis, 12% benefit (376753 seconds)

ACTION: Investigate application logic for possible use of bind variables instead of literals.

ACTION: Alternatively, you may set the parameter "cursor_sharing" to "force".

RATIONALE: At least 7862 SQL statements with PLAN_HASH_VALUE 1350883730 were found to be using literals. An example is SQL statement with SQL_ID "0ksay60kt62w".

```
SELECT * FROM (SELECT  
person_id  
,grade_name  
,start_date StartDate  
,end_date EndDate
```

SYMPTOMS THAT LED TO THE FINDING:

SYMPTOM: Hard parsing of SQL statements was consuming significant database time. (12% impact [397808 seconds])

SYMPTOM: Contention for latches related to the shared pool was consuming significant database time. (7.8% impact [251772 seconds])

INFO: Waits for "library cache lock" amounted to 3% of database time.

Waits for "library cache pin" amounted to 3% of database time.

SYMPTOM: Wait class "Concurrency" was consuming significant database time. (7.9% impact [253733 seconds])

자주 사용되는 SQL의 Hard Parsing으로 인해 많은 DB Time을 소모

Application에서 Bind 변수를 사용하거나 cursor_sharing 파라미터를 force로 설정할 것을 Advice

상기 SQL이 7862번 (8시간 동안) Hard Parsing 됨

해당 원인을 찾아내게 된 증상에 대해서 기술
Library cache Lock과 Library cache Pin Wait Event 가 DB Time을 많이 차지함.

Statspack 개요



- 오라클의 체계적인 성능 모니터링 기법이 **Statspack**에서 **AWR**로 이관됨
- AWR은 유료로 구입해야 하는 별도 Option이지만 Stackspack은 여전히 무료임
- 성능 분석 **주요** 항목은 Statspack과 AWR이 대동 소이함.

기능	Statspack Report	AWR Report
Load Profile	YES	YES
Top 5 Wait Event	YES	YES
SGA 및 PGA 지표	YES	YES
SQL by Class	YES	YES
Latch/Enqueue 지표	YES	YES

Statspack 생성 및 Report 보기

STATSPACK Object 생성

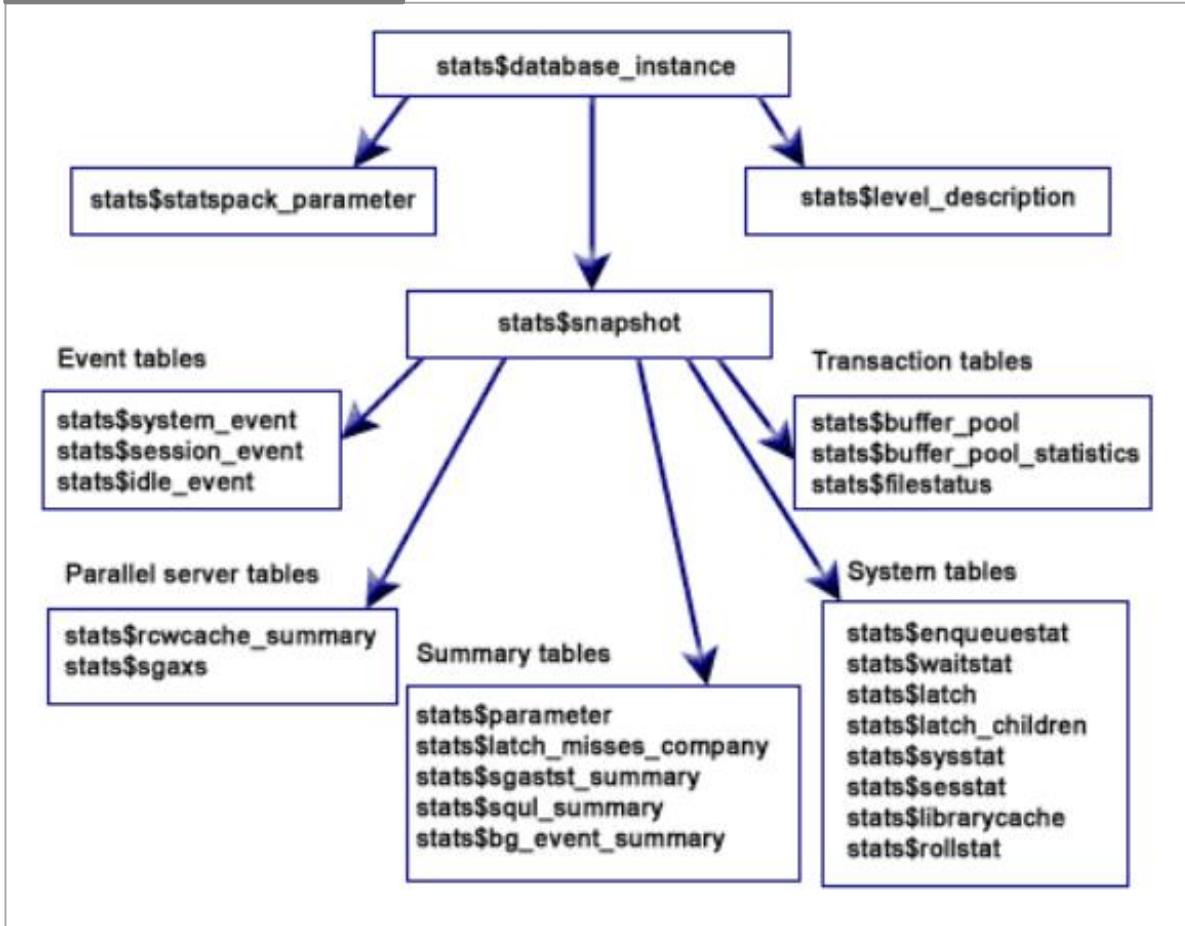
1. sysdba로 로그인 후 **@\$ORACLE_HOME/rdbms/admin/spcreate.sql** 을 이용하여 Statspack과 관련한 USER와 Object를 생성
2. Statspack 생성을 위한 PERFSTAT USER 생성
3. Statspack과 관련한 모든 Object들은 모두 PERFSTAT 스키마내에서 생성 됨

STATSPACK Snap 생성 및 Report보기

1. **SQL Plus로 PERFSTAT 으로 로그인 한 뒤**
2. **특정 기간별로 execute statspack.snap 을 수행하여 snapshot을 생성**
3. **@\$ORACLE_HOME/rdbms/admin/spreport.sql** 을 이용하여 snapshot 구간 별 Report 생성 후 vi로 보기

Statspack의 Object 개요

PERFSTAT



RAM
v\$ Structures

v\$sysstat
v\$gastat
v\$parameter
v\$librarycache

Disk
perfstat Tables

stats\$sysstat
stats\$gastat
stats\$parameter
stats\$librarycache

Statspack report 생성 시 유의 사항

- Statspack이 출시된 시점 보다 Idle event의 대상이 현재는 더 많이 추가됨.
- Top-5 Wait Event에서 Idle event를 제대로 걸러주지 못하면서 부족한 정보 표출
- Idle event 재 구성 필요.