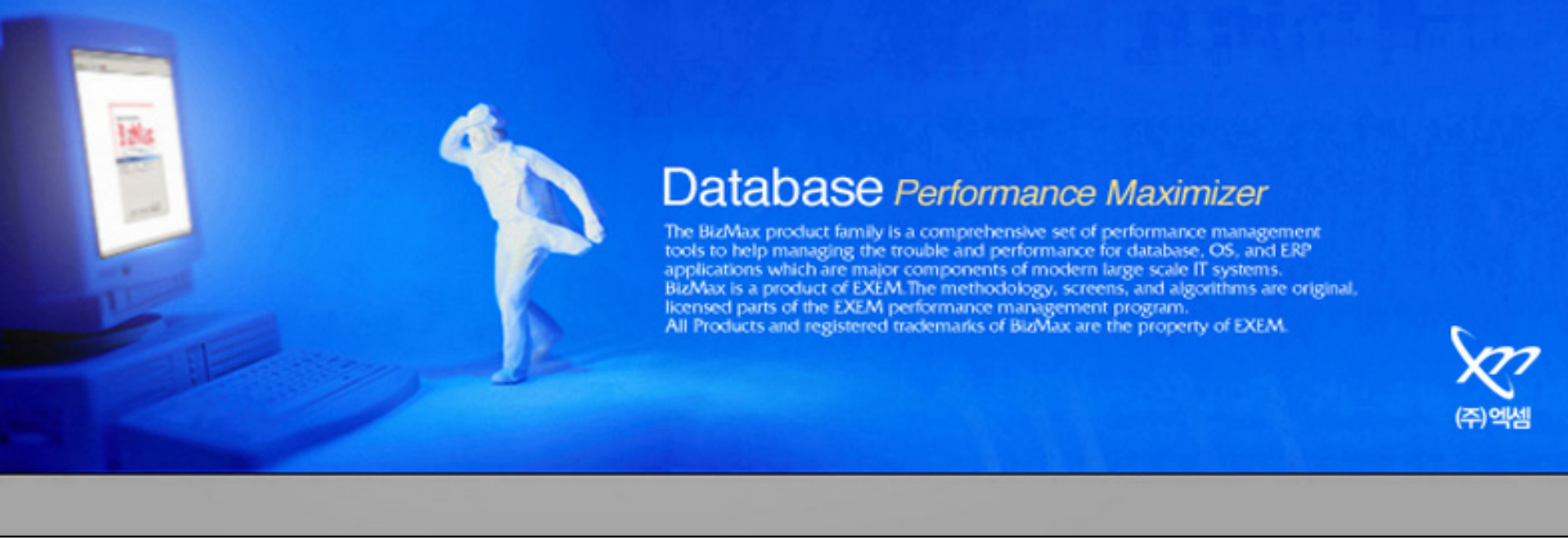


오라클 **Architecture-**기본교육

2005. 2

㈜ 엑셈

교육 개요

- 핵심 목표
  - "오라클 STAT/EVENT 분석방법" 세미나의 이해를 높이기 위한 사전 교육으로, 오라클 백그라운드/포그라운드 프로세스의 동작원리, SGA 구조, 데이터베이스 스토리지 및 Dynamic Performance View (V$) 를 설명함
- 교육 대상
  - DBA, Application 개발자
- 교육 레벨
  - 초급 ~ 중급
- 교육 내용
  - 오라클 주요 콤포넌트
  - Dynamic Performance View (V$)
  - 오라클 Built-in 패키지 2개

I. 오라클 주요 콤포넌트

1. 오라클 인스턴스
2. Shared Pool
3. Buffer Cache
4. 백그라운드 프로세스
5. Server Process
6. SELECT 의 이해
7. DML 의 이해
8. Database Object Relationships Diagram
9. Block
10. Extent
11. Data Segment
12. Temporary Segment

I-1. 오라클 인스턴스

- SGA (System Global Area) 와 백그라운드 프로세스로 구성됨

- SGA 의 구성 (show sga)
  - Fixed Size
  - Variable Size
  - Database Buffers
  - Redo Buffers

- Variable Size의 구성 내역
  - Shared Pool
  - Large Pool
  - Java Pool
  - Stream Pool (from 10g)

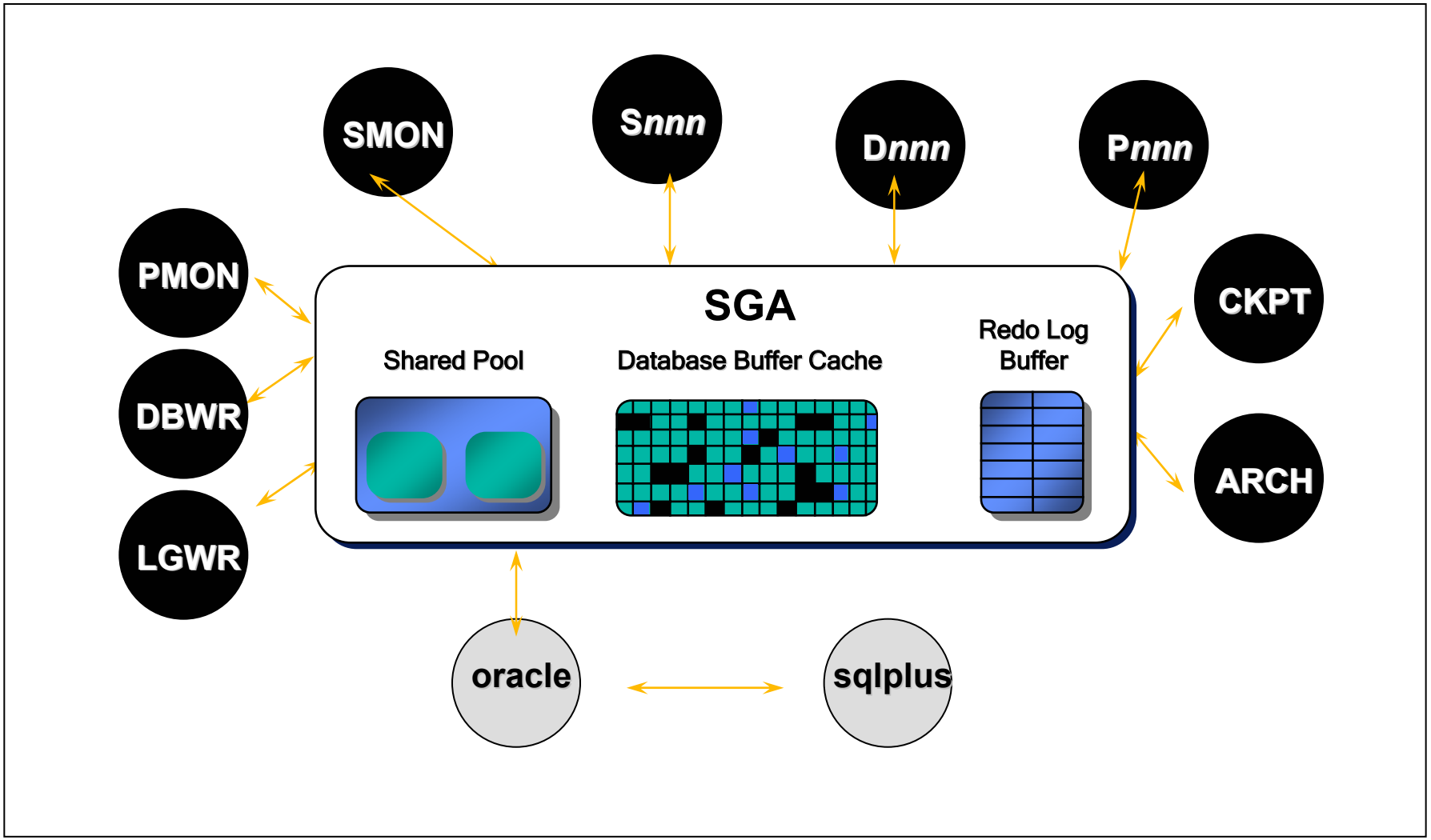
- Shared Pool 의 대표적인 구성 내역
  - Library cache
  - Data dictionary cache (row cache)

I-1. 오라클 인스턴스

- 백그라운드 프로세스의 구성

  - DBWR (DB Writer process)
  - LGWR (LOG Writer process)
  - PMON (Process monitor process)
  - SMON (System monitor process)
  - CKPT (Checkpoint process)
  - ARCH (Archive process)
  - S*nnn* (Shared Server process)
  - D*nnn* (Dispatcher process)
  - P*nnn* (Parallel Query slave process)
  - RECO (Recovery process)

I-1. 오라클 인스턴스


SMON
Snnn
Dnnn
Pnnn
PMON
DBWR
LGWR
SGA
Shared Pool
Database Buffer Cache
Redo Log Buffer
CKPT
ARCH
oracle
sqlplus

I-2. Shared Pool - 구성요소

- SHARED_POOL_SIZE 에 의해 크기 결정
- 9i 부터 CPU 개수에 따라 Shared Pool 을 최대 6개의 서브풀로 자동 구성

_kghdsidx_count 로 설정 가능 (free memory 가 많은 경우에도 ORA-4031 에러발생시 1로 변경필요)

```
SQL> SELECT x.indx+1 as num, ksppinm as name, ksppity as type, ksppstvl as value
        FROM    sys.x$ksppi x , sys.x$ksppcv y
        WHERE ( x.indx = y.indx )
        AND     ksppinm like '%&name%' ;
```

- 구성 요소

1) Library Cache

   - SQL Text, PL/SQL, Parse Tree, Execution Plan
   - LRU (Least Recently Used) Algorithm 에 의해 관리
   - Hard Parse 최소화를 목적으로 함.

2) Data Dictionary Cache

   - Data Dictionary 객체 정의 (Tables, Columns, Grant …)

3) UGA

   - MTS 사용시 세션 정보 (Large Pool 설정이 안된 경우)

I-2. Shared Pool – Library cache Tuning

- 구문분석을 최소로 유지하여 miss 를 줄여야 함.

  1) SQL Text 공유 (Bind SQL 사용 필요)

  2) SQL TEXT Age out 방지 (Literal SQL 과다사용 금지, CURSOR pin)

  3) Invalidation 최소화 (v$librarycache.invalidations, v$sql.invalidations)

     - 업무시간에 DDL 금지

+ Tip

테이블/Procedure Dependency 확인 방법

```
$ORACLE_HOME/rdbms/admin/utldtree.sql
SQL>  exec deptree_fill('TABLE','SCOTT','DEPT');

PL/SQL procedure successfully completed.

SQL> select * from ideptree;

DEPENDENCIES
--------------------------------------------------------------------------------
TABLE SCOTT.DEPT
  CURSOR <shared>."select  * from dept "
  VIEW SCOTT.DEPT_VW
    PROCEDURE SCOTT.DEPT_PROC
```

I-2. Shared Pool – Library cache Tuning

- Library Cache Fragment 방지 방법.

1) Shared Pool Reserved

- SHARED_POOL_RESERVED_SIZE parameter 사용

- _shared_pool_reserved_min_alloc parameter 확인
- v$shared_pool_reserved 확인

2) PIN

- DBMS_SHARED_POOL Package 사용

- $ORACLE_HOME/rdbms/admin/dbmspool.sql
- $ORACLE_HOME/rdbms/admin/prvtpool.plb
- SQL> exec dbms_shared_pool.keep('NAME', 'FLAG');

ex) exec dbms_shared_pool.keep('SCOTT.A_PROCEDURE', 'P');
exec dbms_shared_pool.keep('SCOTT.A_SEQUENCE', 'Q');
exec dbms_shared_pool.keep('SCOTT.A_TRIGGER', 'R');
exec dbms_shared_pool.keep('sql_address,hash_value', 'C');

- SQL> SELECT name, sharable_mem, kept FROM v$db_object_cache;

3) Large Pool 사용

- LARGE_POOL_SIZE parameter 를 설정
  - MTS 사용시
  - parallel_automatic_tuning=true 로 설정된 경우

I-2. Shared Pool – Library cache Tuning

- 용어

1) GETS : namespace 의 객체를 조회한 횟수

2) PINS : namespace 의 객체를 읽거나 실행한 횟수

3) RELOADS : Library Cache miss 횟수

4) INVALIDATIONS : Object에 DDL이 수행되면 해당 Object를 참조하는 Execution Plan 이 Invalid 되므로 다시 Parse 를 수행

+ Tip

1) PINS 의 수치는 항상 GETS보다 큼
2) GETS 는 library cache lock 리소스와 관련이 있으며
   PINS 는 library cache pin 리소스와 관련이 있음.

I-2. Shared Pool – Library cache Tuning

- Library Cache Hit Ratio 확인

```
SQL> Select namespace, gethitratio
       from   v$librarycache;
```

1) gethitratio = gethits/gets

2) SQL AREA 의 경우 90% 이상의 hit ratio 필요

I-2. Shared Pool – Library cache Tuning

- Invalidation 확인

```
SQL> Select namespace, pins, reloads, invalidations
       from   v$librarycache;
```

1) Table, Sequence, Synonym, View의 재생성 , 변경, 삭제 및 Procedure, Package에 대한 compile 시 종속된 공유 SQL Invalidation

2) 업무 시간 중 DDL금지 필요

I-2. Shared Pool – dictionary cache Tuning

- V$ROWCACHE 이용

1) 내용: Dictionary Object 정의

2) 용어: GETS : Object 에 대한 요청 횟수

GETMISSES : Cache miss 가 발생한 요청 횟수

3) 튜닝: Dictionary Cache miss 방지

I-2. Shared Pool – dictionary cache Tuning

- GETMISSES 확인

```
SQL> Select sum(gets), sum(getmisses), sum(getmisses)/sum(gets)*100
       from   v$rowcache;
```

1) gets에 대한 getmisses의 비율이 15% 이상인 경우
   SHARED_POOL_SIZE 증가 고려

+ Tip

1) dc_sequences row cache의 대기가 많이 발생할 경우 시퀀스 option 점검 필요
   NOCACHE ORDER
   NOCACHE NOORDER
   CACHE ORDER
   CACHE NOORDER

   위 4가지의 의미 및 성능이 가장 좋은 것은?

I-3. Buffer Cache - 관련 Parameter

- Buffer Cache는 Buffer pool 용 독립 서브 Cache와 다중 Block 크기용 서브캐시로 구성가능
- DB_BLOCK_SIZE에 의해 SYSTEM , 기본 Buffer Cache 의 block size 결정

- 기본 Block 크기용 Buffer Cache 관련 Parameter
  1) DB_CACHE_SIZE
  2) DB_KEEP_CACHE_SIZE
  3) DB_RECYCLE_CACHE_SIZE

- 다중 Block 크기용 Buffer Cache 관련 Parameter
  1) DB_2K_CACHE_SIZE (4K/8K/16K/32K)

I-3. Buffer Cache – Dynamic SGA

- Dynamic SGA 기능은 Instance 재기동 없이 SGA 의 구성 변경 가능
- SGA_MAX_SIZE 에 의해 제한
  - SGA_MAX_SIZE 만큼 O/S의 공유메모리를 확보함
- V$BUFFER_POOL 을 이용한 모니터링

NAME	BLOCK_SIZE	CURRENT_SIZE	TARGET_SIZE	PREV_SIZE
DEFAULT	4096	48	48	96

I-3. Buffer Cache – Cache hit ratio

- V$SYSSTAT 이용

```
SQL> Select 1 - (a.value - b.value - c.value)/d.value * 100
        from   v$sysstat a, v$sysstat b, v$sysstat c, v$sysstat d
        where  a.name='session logical reads'
        and    b.name='physical reads direct'
        and    c.name='physical reads direct (lob)'
        and    d.name='physical reads';
```

1) Hit Ratio 는 성능확인의 참고 자료일 뿐

2) Logical Reads를 과도하게 발생되는 악성 SQL 수행 시에도 99% 의 Hit Ratio 나타남.

I-3. Buffer Cache – multiple buffer cache

- Multiple Buffer Cache 유형

  1) Default buffer : DB_CACHE_SIZE

  2) keep buffer : DB_KEEP_CACHE_SIZE

  3) recycle buffer : DB_RECYCLE_CACHE_SIZE

- 동적 할당 가능
- Latch는 Oracle 에서 자동할당

  - DB_BLOCK_LRU_LATCHES (8i)

- CREATE TABLE[INDEX] … STORAGE (BUFFER_POOL KEEP);
- ALTER TABLE[INDEX] … STORAGE (BUFFER_POOL KEEP);

I-3. Buffer Cache – Free Lists

- Object 의 Free list는 Insert 시에 사용 할 수 있는 Block List를 관리
- Object 의 Free list 수는 동적 설정 가능
- 충분한 Free list 설정으로 경합 최소화
- RAC (Real Application Clusters) 인 경우 Free list groups 설정 필요
- Auto Segment space management 기능을 사용하는 경우 Free lists 필요 없음

I-3. Buffer Cache – Segment Space management

- Database Segment 내의 사용 가능 공간을 자동으로 관리
- Bitmap 을 사용하여 Free list 대신 Segment 사용 가능/사용된 공간 추적
- 테이블스페이스를 생성 할 때 지정

  SQL> CREATE TABLESPACE …..

  SEGMENT SPACE MANAGEMENT AUTO;

- 하나 이상의 LOB 열을 포함하는 테이블에는 사용 불가

I-3. Buffer Cache – multiple I/O slave

- Async I/O 를 사용할 수 없는 경우 I/O Slave를 이용하여 Async I/O Simulate
- 일반적으로 Async I/O 를 사용할 수 있는 경우 I/O Slave 사용 자제
- DBWR_IO_SLAVES 와 DB_WRITER_PROCESSES 간의 성능 비교 필요

  - 일반적으로 free_buffer_waits event가 발생하는 경우 설정 고려

- DISK_ASYNCH_IO 를 이용하여 Async I/O 기능 설정 또는 해제

I-4. 백그라운드 프로세스-DBWR

- Buffer cache의 더티블록을 디스크에 기록하는 일을 담당함
  - Checkpoint 가 발생하였을 때
  - Dirty Buffer의 수가 임계치에 도달하였을 때(_DB_BLOCK_MAX_DIRTY_TARGET)
  - Free Buffer의 수가 모자랄 때
  - 테이블 스페이스가 OFFLINE 또는 READ ONLY로 변경될 때
  - 테이블 스페이스의 BEGIN BACKUP 시
  - Drop 또는 Truncate 시

I-4. 백그라운드 프로세스-DBWR

- DBWR 성능에 문제가 발생하는 경우 어떠한 문제가 유발될 수 있는가?
- 이러한 경우에 발생되는 대표적인 EVENT는 ?

free buffer waits (USER)
write complete waits (USER)
db file parallel write (DBWR)

I-4. 백그라운드 프로세스-LGWR

- Log Buffer 의 내용을 온라인 리두로그 파일에 기록하는 일을 담당함
  - COMMIT 또는 ROLLBACK 수행 시
  - 임계치만큼 저장되었을 때
    - (Log Buffer의 1/3 또는 1M 중 작은 값을 로그블록단위로 _LOG_IO_SIZE 에 설정)
  - 매 3초마다
  - DBWR 가 기록하기 전에

I-4. 백그라운드 프로세스-LGWR

- 리두 로그 파일을 RAID-5 디스크에 위치 시켰을 때 문제점은?
- 과다한 커밋 수행이 바람직한가? 만일 아니라면 이유는?
- 과다한 커밋 수행 시에 수반되는 대기 이벤트는 ?

  log file sync

- 온라인 백업 수행중인 테이블 스페이스에 대한 DML 작업이 문제가 되는가?
- 만일 문제가 된다면 어떤 이유에서 인가?
- 이때 발생될 수 있는 대기 이벤트는 ?

  log buffer space

I-4. 백그라운드 프로세스-CKPT

- 체크포인트 정보를 Control file 과 데이터파일 헤더에 기록하는 일을 담당한다.
- 대량의 DML의 발생하는 환경에서 리두로그 파일의 아주 작다. 어떠한 문제가 발생될 수 있는가?
- 이러한 문제가 발생될 때 수반되는 대기 이벤트는 ?

  log file switch (checkpoint incomplete)

- 해결방안은?

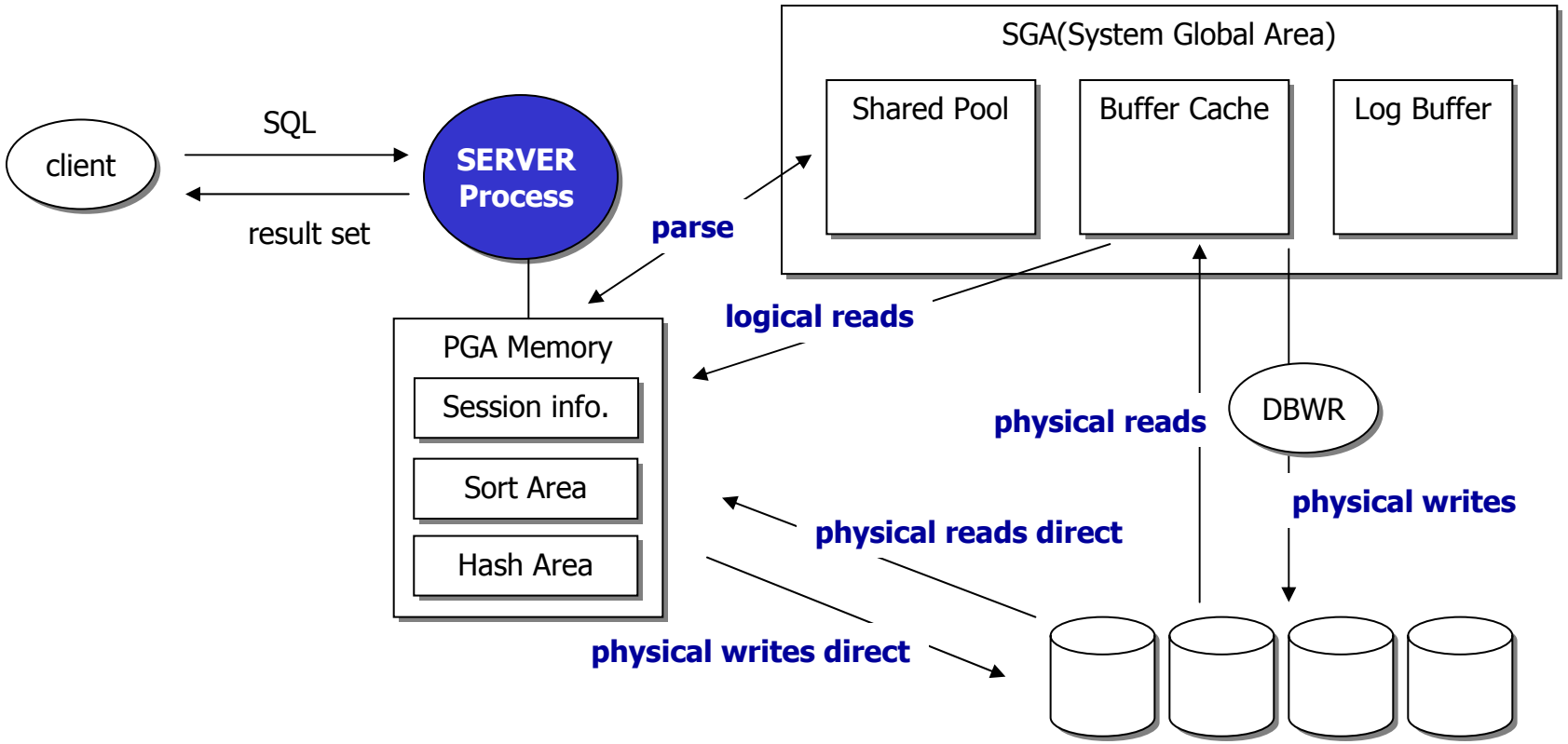
I-4. 백그라운드 프로세스-SMON

- 인스턴스 리커버리 수행
- UNDO 세그먼트 관리 (9i, UNDO_MANAGEMENT=AUTO)
- UNDO 와 ROLLBACK 의 차이점은 ?
- Transaction이 과다하게 몰리는 시점이 있는 시스템에서 UNDO 사용이 유용한가?
- ORA-1555 발생이 과다하게 발생하는 환경에서 UNDO 사용이 유용한가?
- 트랜젝션 리커버리의 코디네이터 (FAST_START_PARALLEL_ROLLBACK=LOW or HIGH)

I-4. 백그라운드 프로세스-ARCH

- 온라인 리두 로그 파일을 읽어서 아카이브 파일을 생성하는 일을 담당.
- 아카이브 모드로 운용시에 리두 로그 그룹들은 반드시 별도의 디스크에 위치시켜야 한다. 어떤 이유에서 일까?

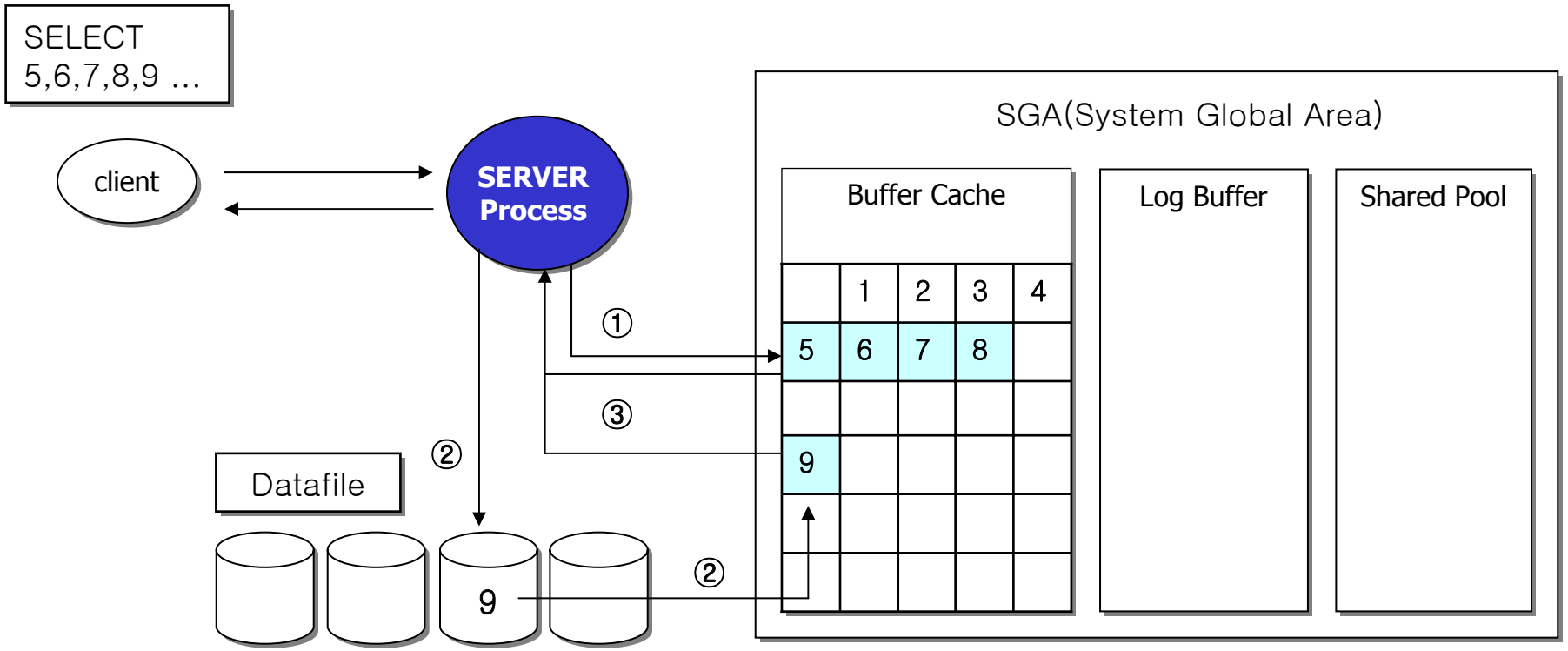
I-5. Server Process



- 서버 프로세스는 SQL을 접수 받아 파싱하고 실행하는 역할을 수행한다.
- DB로부터 블록을 읽는 경우 buffer cache를 통해서 읽게 된다.
- Sorting 작업, Hashing 작업 중에 메모리가 모자라는 경우에는 buffer cache를 거치지 않는다.
- Shared(MTS) 서버의 경우에는 session info를 SGA 영역에 보관한다.

I-6. SELECT 의 이해

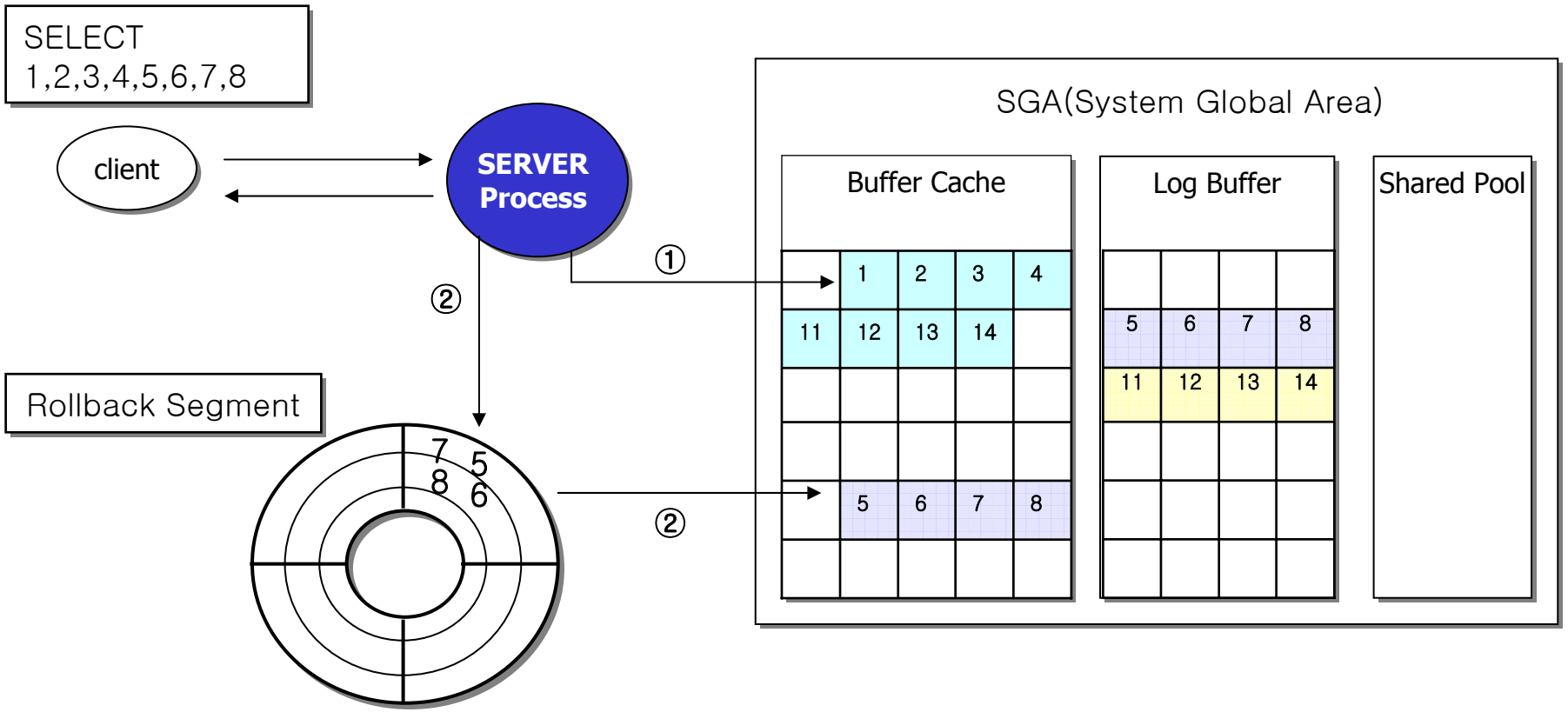
1. 다른 Session에서 DML이 없는 경우



① Buffer Cache 에서 해당 Row를 찾는다.
② 만약 Buffer Cache 에 없으면, Datafile 에서 읽은 후, Buffer Cache 에 Load.
(By LRU Algorithm)
③ Buffer Cache 로부터 읽는다.

I-6. SELECT 의 이해

2. 자신은 DML이 없고, 다른 Session에서 UPDATE후 COMMIT하지 않은 경우
   다른 Session에서 (5,6,7,8) → (11,12,13,14)

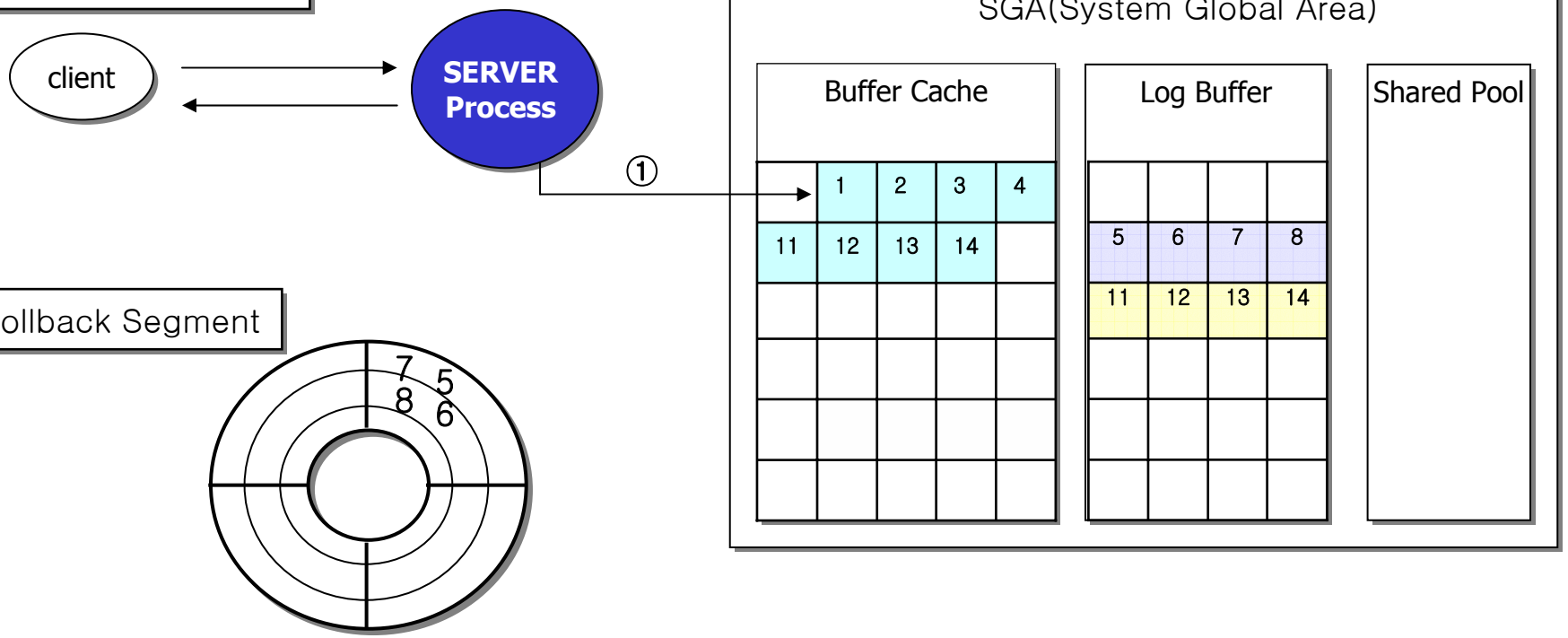


① Buffer Cache 에서 해당 Row를 찾는다.
② 다른 Session 에서 Update 후 Uncommitted 상태라면, Old Image 를 읽기 위해 CR Operation 을 수행한다. 필요하다면 Rollback Block 을 읽고, Block 을 새로운 Version 으로 Copy 한다.

I-6. SELECT 의 이해

3. 자기 Session에서 UPDATE후 COMMIT를 하지 않은 상태에서 SELECT

SELECT
1,2,3,4,11,12,13,14

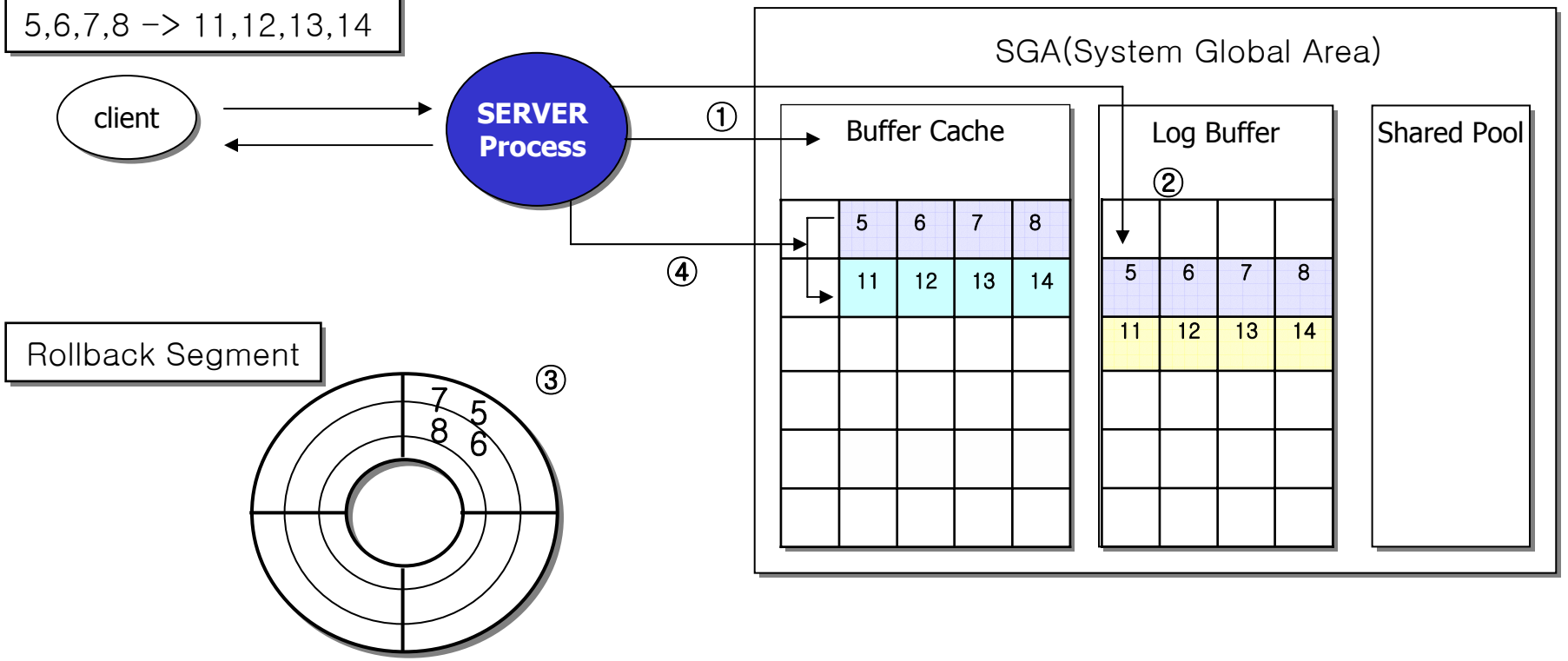


① Buffer Cache 에서 해당 Row를 찾는다.
자기 Session 에서 Update 후 Uncommitted 상태라면, 현재 Block 을 읽는다.
- Commit 이 되지 않았어도 Buffer Cache 의 Block 은 항상 마지막 DML 결과가 현재 Block 이 된다.

I-7. DML 의 이해

1. UPDATE

UPDATE
5,6,7,8 -> 11,12,13,14

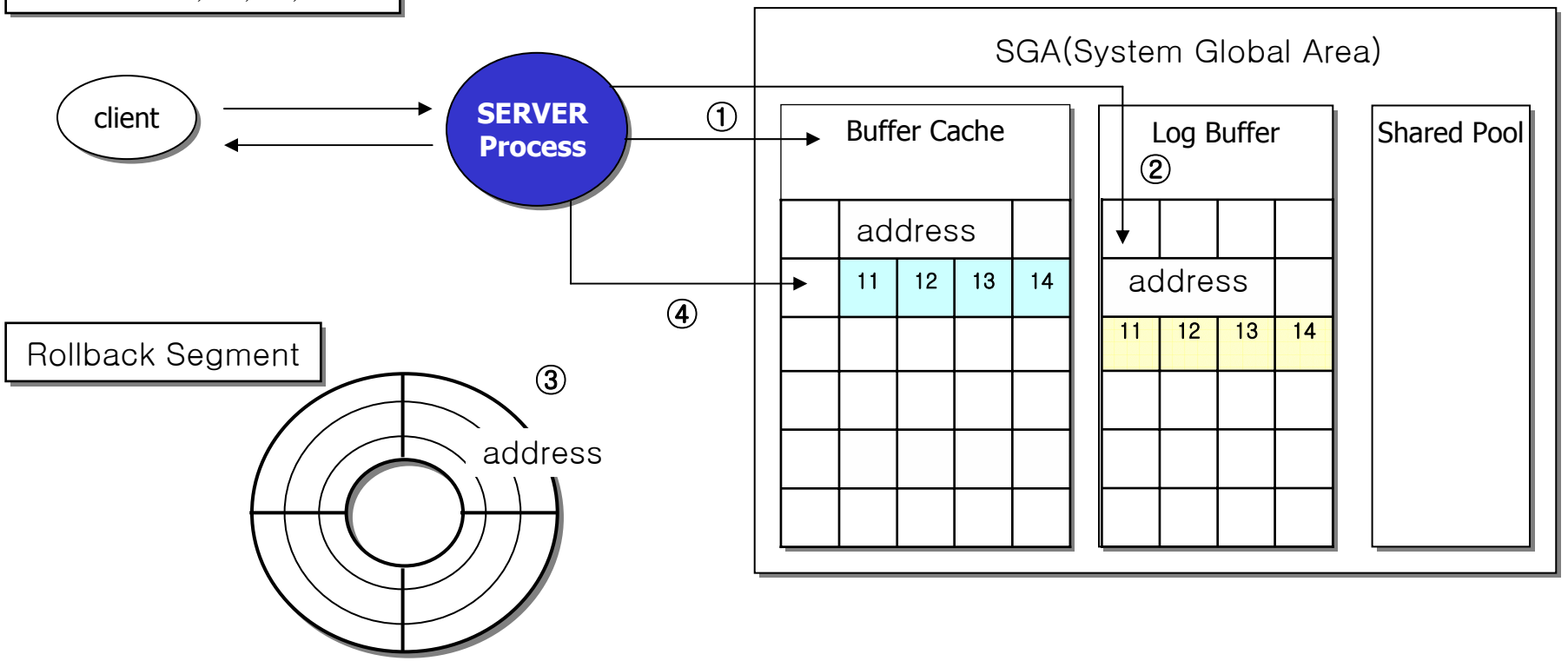


① Buffer Cache 에서 해당 Row를 찾는다. 만약, 없으면 Datafile 에서 읽어서 Buffer Cache 의 Data Block 에 Cache한다 (Cache 후, Row Lock 설정)
② Old Image 와 New Image 를 Redo Log Buffer에 기록 한다.
③ Rollback Segment 의 Undo Block 에 Data block 의 Old Image 를 기록
④ Buffer Cache 의 Data Block 에 New Image를 Update 한다.

I-7. DML 의 이해

2. INSERT

INSERT 11,12,13,14

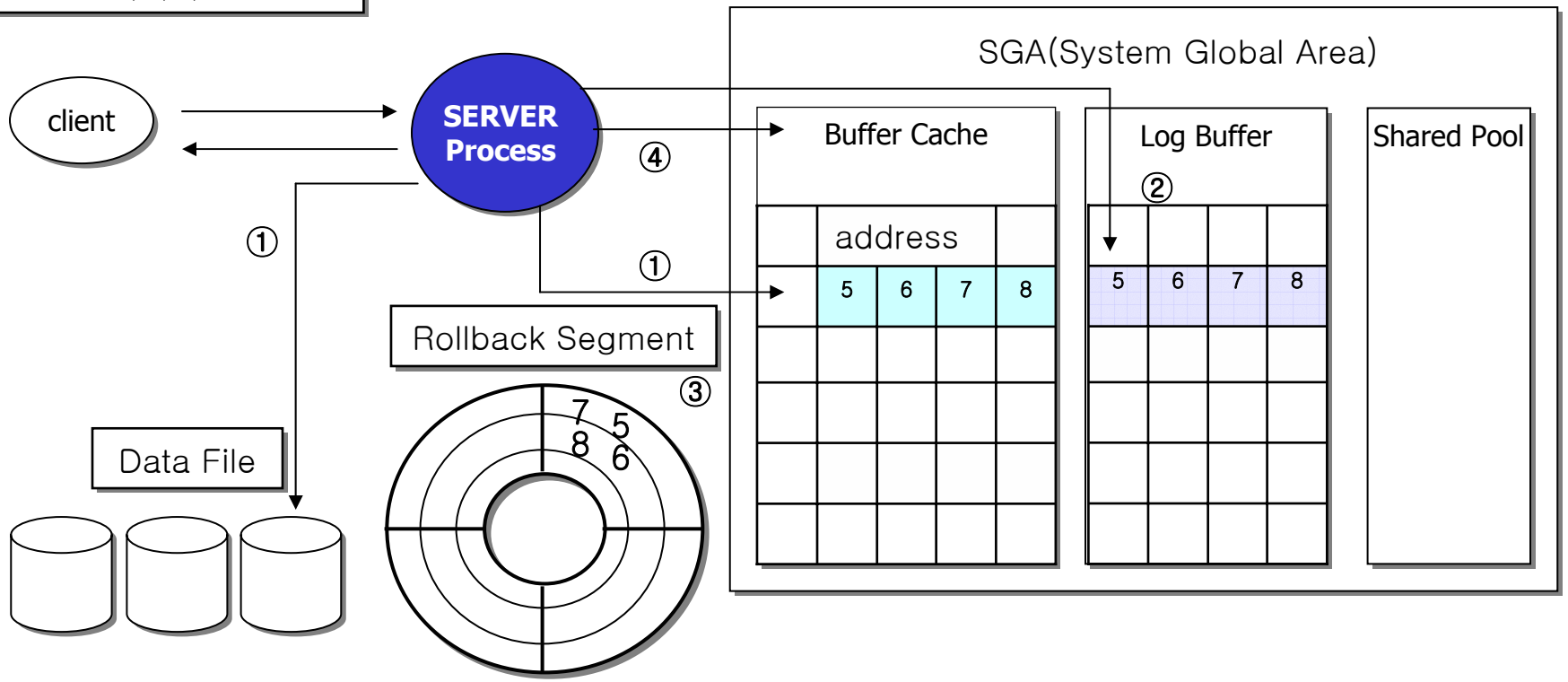


① 해당 Table 의 Free Block 을 Buffer Cache 에 Cache (Cache 후, Row Lock 설정)
② 위치정보와 New Image 를 Redo Log Buffer에 기록 한다.
③ 대상 Row 의 위치정보를 Rollback Block 에 기록
④ Buffer Cache 의 Free Block 에 New Image 기록

I-7. DML 의 이해

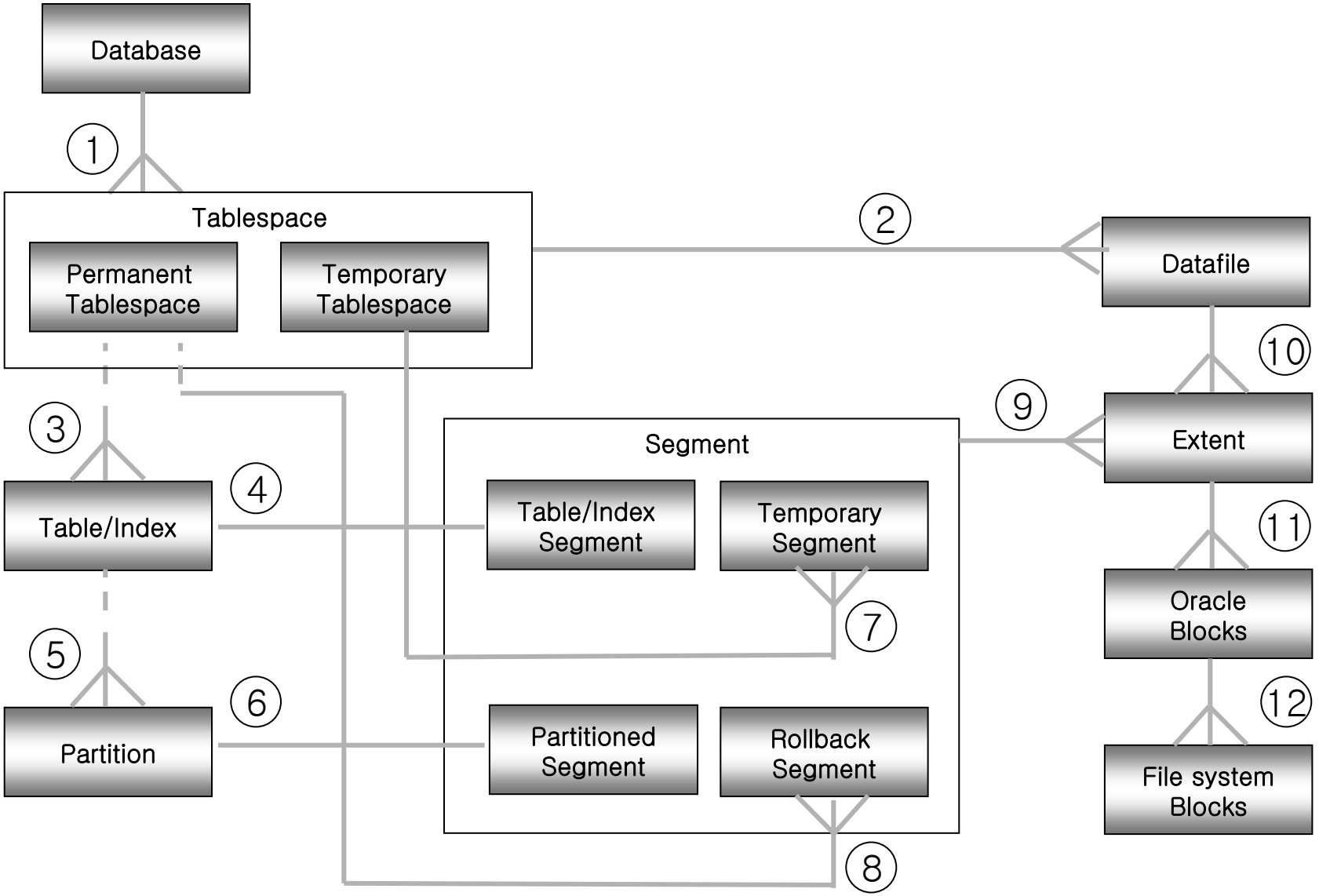
3. DELETE

DELETE 5,6,7,8



① 해당 Block 을 DB File 에서 읽어서 Delete할 대상 Row를 Buffer Cache 에 Load 시킨다 (Cache 후, Row Lock 설정)
② Deleted Row를 Redo Log Buffer 에 기록
③ 대상 Row 를 Rollback Block 에 기록
④ Buffer Cache 의 Block 에서 Delete

I-8. Database Object Relationships Diagram



I-8. Database Object Relationships Diagram

① 데이터베이스는 한 개 이상의 테이블스페이스(SYSTEM 포함)로 구성되어 있다.

② 테이블스페이스는 한 개 이상의 데이터파일로 구성되어 있다.

③ 테이블스페이스에는 0개 이상의 테이블과 인덱스를 위치시킬 수 있다.

테이블스페이스는 permanent와 temporary로 나뉘어진다.

④ 테이블과 인덱스는 물리적으로는 하나의 세그먼트로 나타내어진다. (Non-partitioned 가정)

⑤ 테이블과 인덱스는 복수 개의 파티션으로 구성될 수 있다.

⑥ 테이블과 인덱스가 파티션되는 경우 각 파티션 당 물리적인 하나의 세그먼트로 나타내어진다.

⑦ TEMP 세그먼트는 temporary tablespace에 위치시켜야 한다.

⑧ ROLLBACK 세그먼트는 permanent tablespace에 위치시킨다. (대부분 별도의 RBS 스페이스)

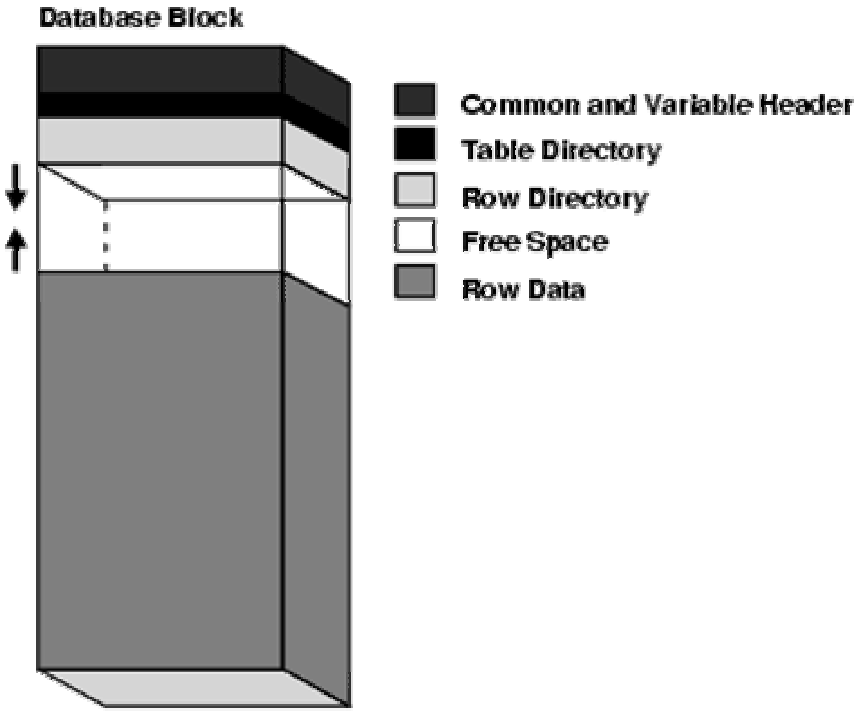
⑨ 세그먼트는 복수개의 익스텐트로 구성된다.

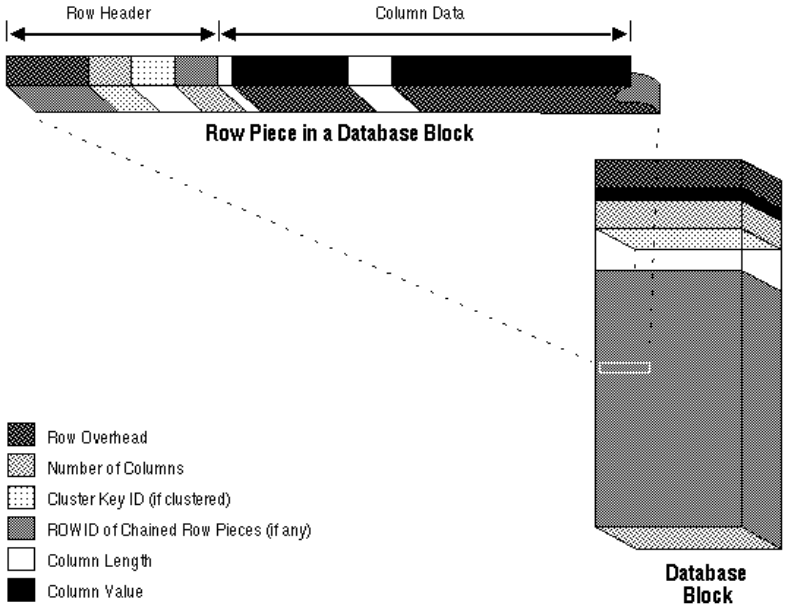
⑩ 하나의 데이터파일은 여러 개의 익스텐트를 가진다.

⑪ 하나의 익스텐트는 여러 개의 연속된 오라클 블록으로 구성된다.

⑫ 하나의 오라클 블록은 여러 개의 파일시스템의 블록으로 구성된다.

I-9. Block

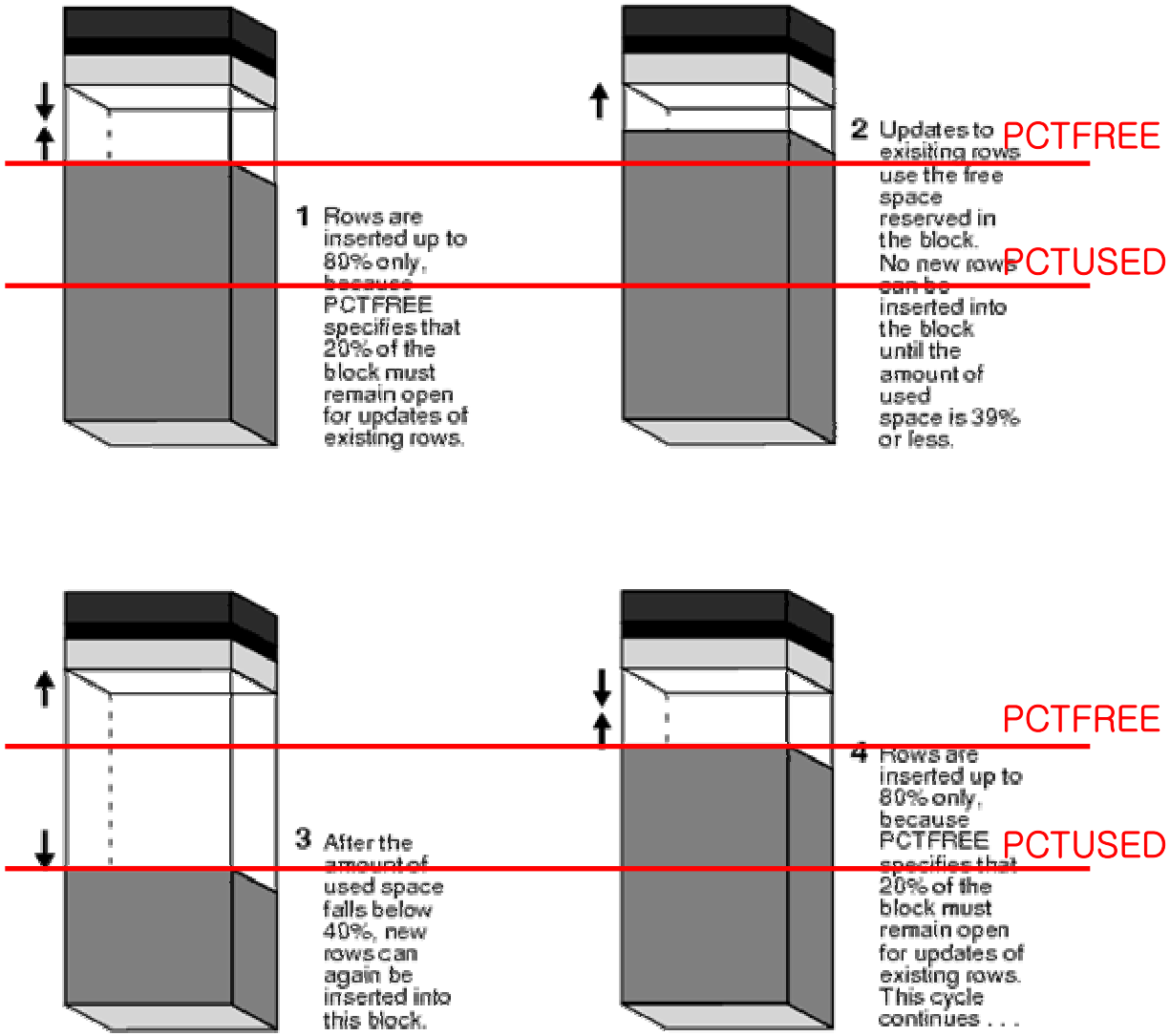




- **ITL (Interested Transaction List)**
  DML을 수행하는 트랜잭션은 하나의 slot을 차지, slot 당 24 Byte 소요, ITL이 모자라면 TX Lock 발생
- **INITRANS**
  블록에 미리 잡아 놓을 ITL slot 개수
- **MAXTRANS**
  INITRANS가 모자라는 경우 free space에 ITL영역을 만든다. (최대 MAXTRANS 개수까지)
  PCTFREE가 적게 잡힌 경우에 문제 가능성

I-9. Block

PCTFREE와 PCTUSED를 이용한 Freespace Management



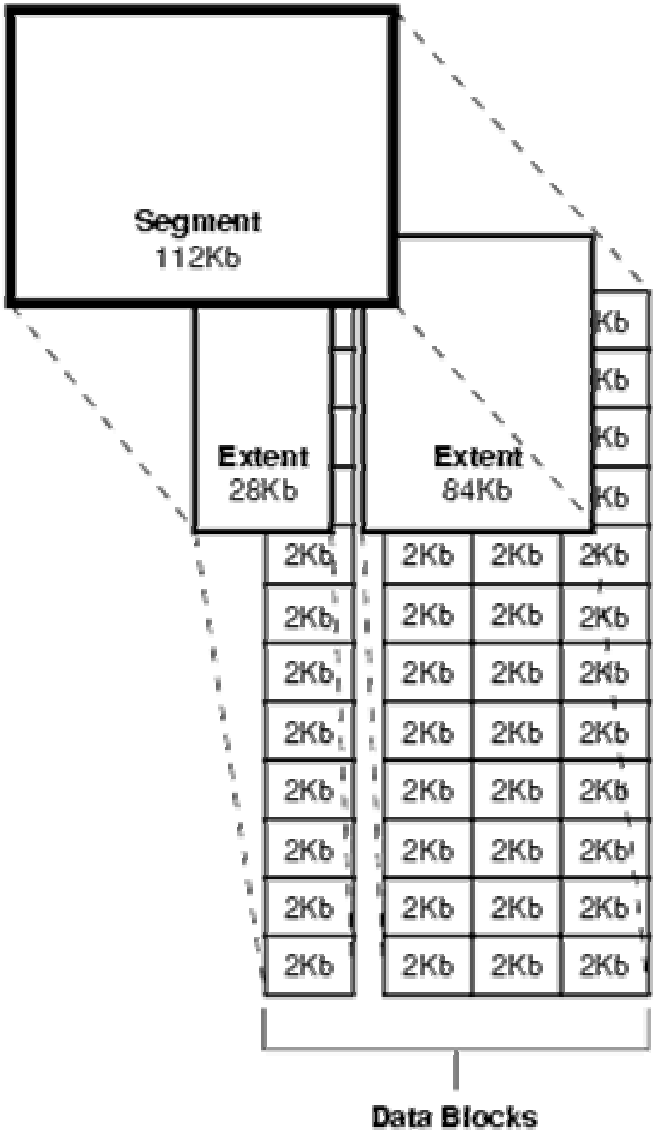
❑ PCTFREE

- o 가변길이 컬럼에 update가 빈번한 경우에는 PCTFREE 값을 크게 잡는다.
- o update가 발생하지 않는 테이블은 PCTFREE 적게 하지만 MAXTRANS 고려
- o PCTFREE를 크게 잡으면 공간의 낭비와 Block I/O가 증가할 가능성

❑ PCTUSED

- o 너무 크게 잡으면 빈번하게 freelist를 들락거림으로 인한 오버헤드
- o 너무 작게 잡으면 공간의 낭비
- o PCTFREE+PCTUSED<100

I-10. Extent



- **INITIAL**

  세그먼트 생성 초기에 할당할 익스텐트의 크기
  생성 이후에 한번에 로드 할 양이 많다면 크게 잡는다.

- **NEXT**

  두 번째로 할당할 익스텐트의 크기

- **PCTINCREASE**

  세 번째 이후에 할당할 NEXT 익스텐트의 증가율
  예, INITIAL=1M, NEXT=1M, PCTINCREASE=50 이면
  1M -> 1M -> 1.5M -> 2.25M -> ……
  INITIAL = NEXT, PCTINCREASE = 0 를 유지하면 SMON 이 coalesce할 필요가 없어진다.

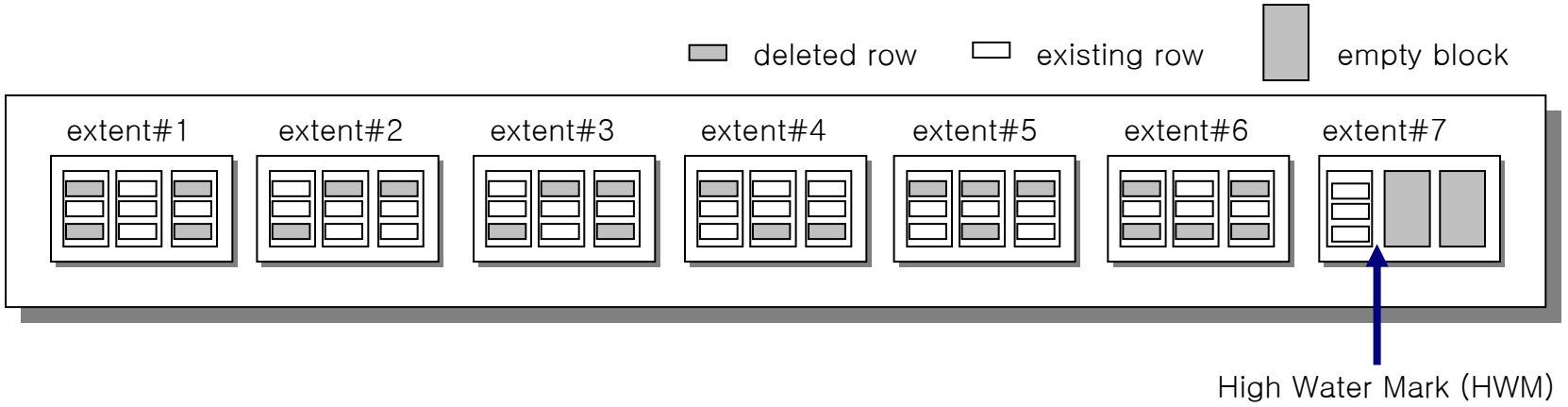
- **MINEXTENTS**

  세그먼트 생성 초기에 할당할 익스텐트의 개수

- **MAXEXTENTS**

  세그먼트에 할당할 익스텐트의 최대 개수

I-11. Data Segment



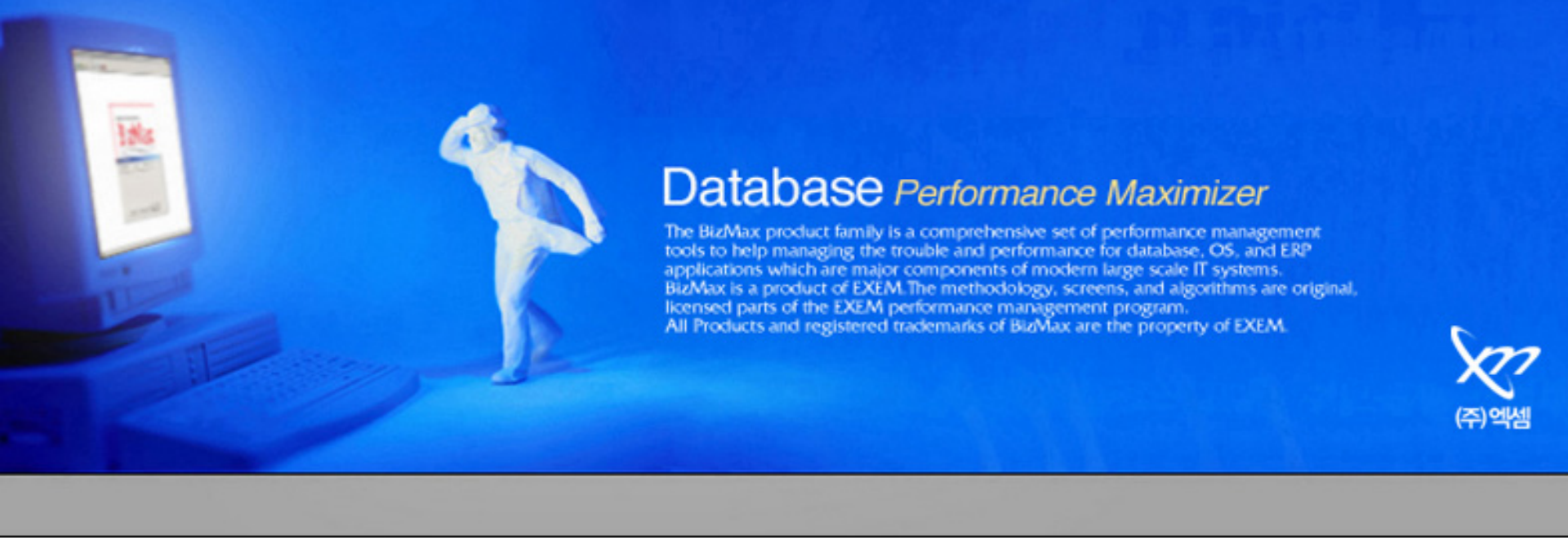
- Data Segment는 테이블, 딕셔너리 등의 데이터를 저장하기 위한 세그먼트
- 익스텐트 단위로 저장공간이 할당된다.
- 가장 마지막 extent의 마지막 블록에 High Water Mark가 찍힌다. (Full Scan 시 사용)
- Header 블록에는 freelist 정보가 있다.

I-12. Temporary Segment

Temporary Segment를 사용하는 Operation

- CREATE INDEX
- SELECT … ORDER BY
- SELECT DISTINCT …
- SELECT … GROUP BY
- SELECT … UNION
- SELECT … INTERSECT
- SELECT … MINUS
- SORT-MERGE JOIN
- HASH JOIN

- ❑ Sort Area를 늘림으로써 temporary segment를 사용할 가능성을 최소화한다. (sort_area_size, sort_area_retained_size)
- ❑ Hash join의 경우 Hash Area를 늘림으로써 temporary segment 사용 가능성을 최소화한다.
- ❑ Temporary segment를 정리하는 작업은 SMON이 수행



Dynamic Performance View (V$) & Oracle Built-in Package

2005. 2

㈜ 엑셈

목 차

I. Dynamic Performance View

1. Dynamic Performance View 란 ?
2. X$ Fixed Table vs V$ View
3. 주요 V$ Relationship
4. V$EVENT_NAME
5. V$SYSTEM_EVENT
6. V$SESSION_EVENT
7. V$SESSION_WAIT
8. V$SESSION
9. 주요 STAT
10. 주요 EVENT

I-1. Dynamic Performance View 란?

- Dynamic Performance View ?
  - 데이터베이스 유지 및 관리에 필요한 정보를 표현하며, 운영 시 계속 변경됨
  - 데이터베이스 성능관련 정보를 수집하며 V$ 접두사로 명명된 view를 통해 액세스 됨
  - 오라클의 Internal Operation 정보를 모니터링
  - 디스크와 메모리 구조에 저장됨
  - SELECT * FROM V$FIXED_TABLE 로 모든 목록 조회 가능
  - V$FIXED_VIEW_DEFINITION을 이용하여 View Query 조회 가능
- Static Data Dictionary View ?
  - Data dictionary 의 구성요서가 변경될 때 만 변경되는 뷰
  - DBA_, ALL_, USER_ 의 접두사를 가진 뷰로 오브젝트와 인스턴스의 정적인 정보를 포함
  - SELECT * FROM DICT[IONARY]로 모든 목록 조회 가능

I-2. X$ Fixed Table vs V$ View

X$ Fixed Table ?

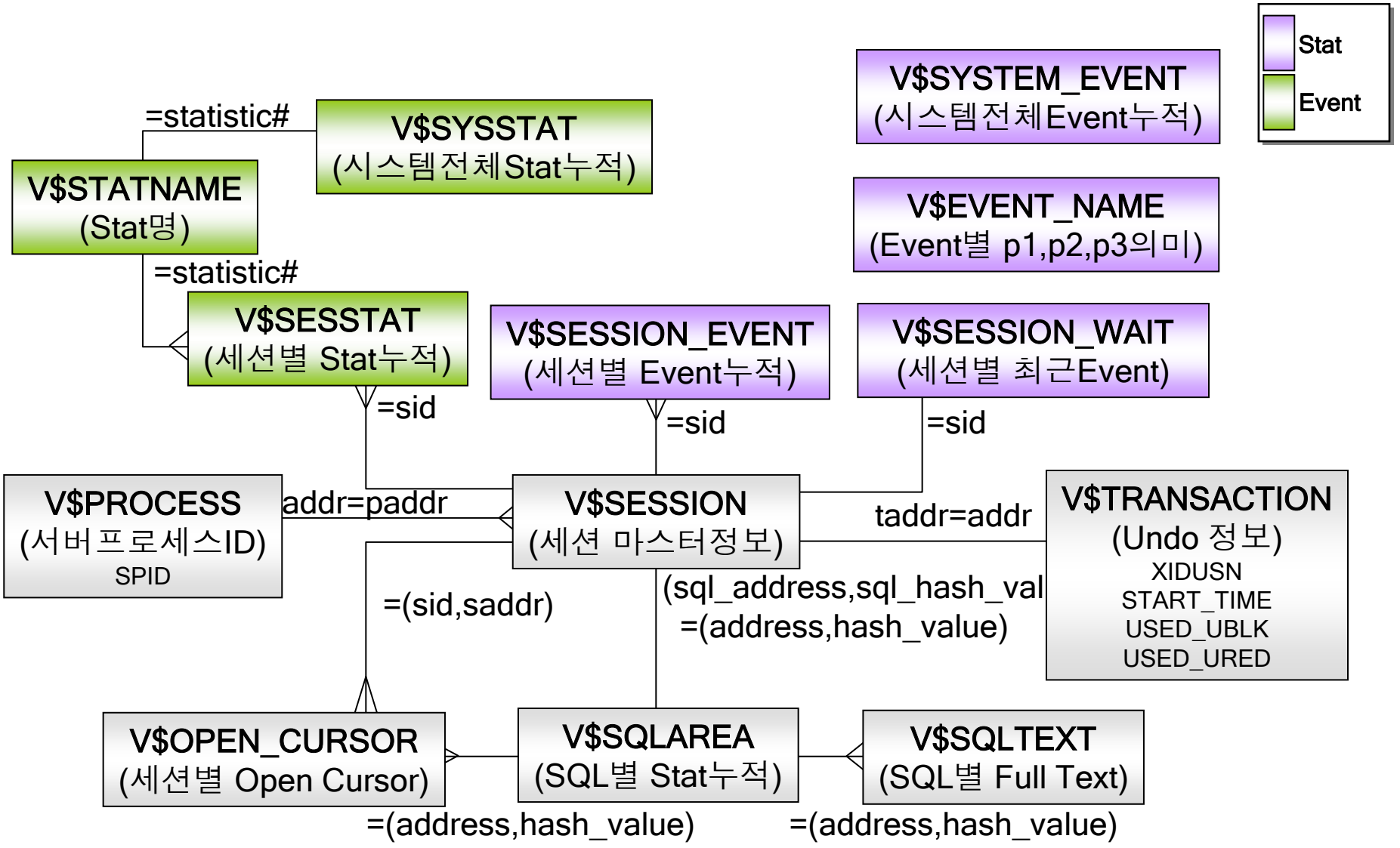
- 오라클 데이터베이스 메모리 스트럭처
- SYS 스키마에서만 접근할 수 있음
- V$ View는 X$ 에서 제공하는 모든 정보를 제공하지 않음
- V$ View를 조회할 때 발생하는 부하를 줄일 수 있음
- X$ 에 대한 조회는 메모리 액세스이므로 , 일반적인 오라클 I/O 메커니즘을 사용하지 않음 (즉 session logical reads, physical reads 발생수치 0)

X$DUAL vs DUAL

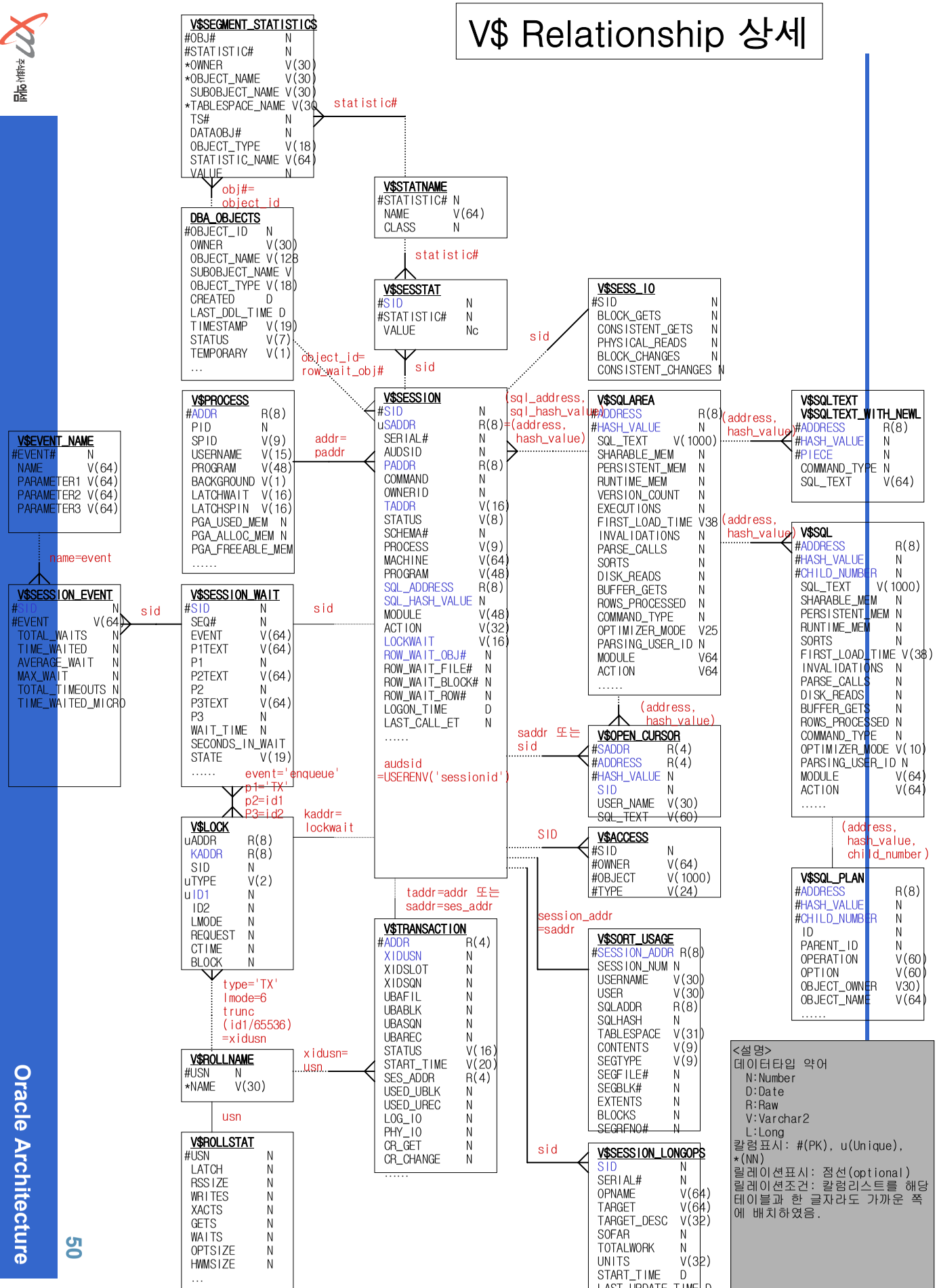
- SYS> create or replace view xm$dual as select * from sys.x$dual;
- SCOTT> SELECT dummy into :v_dummy FROM DUAL;
- SCOTT> SELECT dummy into :v_dummy FROM XM$DUAL;

- 각각의 SQL을 100,000 회 LOOP 수행
- DUAL = Response Time: 7.6 초
- XM$DUAL = Response Time: 4.3 초

I-3. 주요 V$Relationship



V$ Relationship 상세



<설명>
데이터타입 약어
N:Number
D:Date
R:Raw
V:Varchar2
L:Long
칼럼표시: #(PK), u(Unique), *(NN)
릴레이션표시: 점선(optional)
릴레이션조건: 칼럼리스트를 해당 테이블과 한 글자라도 가까운 쪽에 배치하였음.

I-4. V$EVENT_NAME

- EVENT# : Event 번호
- NAME : Event 명
- PARAMETER1 : 대기 Event P1 의미
- PARAMETER2 : 대기 Event P2 의미
- PARAMETERr3 : 대기 Event P3 의미

Check Point

1. 오라클의 모든 이벤트는 최대 3개의 속성을 가지고 있으며, 각각의 속성의 의미하는 것이 Parameter1,2,3 임

I-5. V$SYSTEM_EVENT

- EVENT : Event 명
- TOTAL_WAITS : 이벤트를 대기한 횟수
- TOTAL_TIMEOUTS : 이벤트를 획득하지 못하여 TIMEOUT이 발생한 횟수
- TIME_WAITED : 대기시간 (1/100초)
- AVERAGE_WAIT : TIME_WAITED/TOTAL_WAITS (1/100초)
- TIME_WAITED_MICRO : 대기시간 (1/1,000,000초)

Check Point

1. 인스턴스 레벨의 누적 이벤트 데이터 저장
2. 대부분의 이벤트는 wait time (timeout time) 속성을 가지고 있음
   - 세션이 wait를 재설정(renewing) 하기 전까지 대기할 수 있는 최대 시간
3. I/O 관련된 이벤트는 wait time 속성이 없음. 즉 TOTAL_TIMEOUTS=0
   - db file sequential read, db file scattered read .etc

4. I/O 관련된 이벤트는 AVERAGE_WAIT를 확인하여 1회 I/O call시에 소요되는 비용을 확인해야 함

I-6. V$SESSION_EVENT

- SID : 세션 ID
- EVENT : Event 명
- TOTAL_WAITS : 이벤트를 대기한 횟수
- TOTAL_TIMEOUTS : 이벤트를 획득하지 못하여 TIMEOUT이 발생한 횟수
- TIME_WAITED : 대기시간 (1/100초)
- AVERAGE_WAIT : TIME_WAITED/TOTAL_WAITS (1/100초)
- MAX_WAIT : 최대 대기 시간 (1/100초)
- TIME_WAITED_MICRO : 대기시간 (1/1,000,000초)

Check Point

1. 세션레벨의 누적 이벤트 데이터 저장
2. 오라클 9.2.0.3 전까지는 V$SESSION과 SID 일치하지 않는 Bug 존재
   - FROM V$SESSION_EVENT a, V$SESSION b
   - WHERE a.sid = b.sid - 1 (즉 PMON의 event정보가 없음)

I-7. V$SESSION_WAIT

- SID : 세션 ID
- SEQ# : Event SEQ Number
- EVENT : 이벤트명
- P1 : Event 속성#1
- P2 : Event 속성#2
- P3 : Event 속성#3
- SECONDS_IN_WAIT : 대기시간 (초)
- STATE : 대기상태

Check Point

1. 현재시점에 세션에 발생되는 대기 이벤트의 정보를 출력

2. SECONDS_IN_WAIT 은 대부분 3초마다 갱신됨
3. STATE
   - WAITED UNKNOWN TIME : 대기시간 측정 불가(T=False)
   - WAITED SHORT TIME : 이전의 대기가 1/100 미만
   - WAITING : 현재 대기 중
   - WAITED KNOWN TIME : 마지막대기부터 현재까지의 대기시간 측정 가능

I-8. V$SESSION

- SID : 세션 ID
- SERIAL# : 세션 Serial#
- PROGRAM : 프로그램 명
- MODULE : 모듈 명
- ACTION : 액션 명
- LAST_CALL_ET : SQL 수행시간

Check Point

1. PADDR = V$PROCESS.ADDR
2. TADDR = V$TRANSACTION.ADDR
3. (SQL_ADDRESS,SQL_HASH_VALUE) = V$SQL(ADDRESS,HASH_VALUE)

I-9. V$SGASTAT

- pool : Pool 이름
- name : SGA 콤포넌트 명
- bytes : Memory Size (byte)

```
SELECT pool, name, bytes
FROM   v$sgastat

POOL         NAME                            BYTES
------------ ------------------------- ----------
             fixed_sga                       282576
             db_block_buffers              33554432
             log_buffer                      524288
shared pool  1M buffer                      1049088
shared pool  Checkpoint queue                141152
shared pool  PL/SQL MPCODE                   175740
shared pool  PLS non-lib hp                    2068
shared pool  VIRTUAL CIRCUITS                180760
shared pool  dictionary cache                765360
shared pool  enqueue                         122808
```

I-10. 주요 STAT

- logons current
- logons cummulative
- session logical reads
- physical reads
- physical reads direct
- physical writes direct
- redo entries
- execute count
- parse count (hard)
- session cursor cache hits
- session pga memory
- Table scan blocks gotten/ table scan rows gotten
- active sessions (V$SESSION.STATUS)

I-11. 주요 EVENT

- db file scattered read
- db file sequential read
- direct path read
- direct path write
- db file parallel read
- db file parallel write
- enqueue
- latch free
- library cache pin
- library cache lock
- buffer busy waits
- free buffer waits

II. Oracle Built-in Package

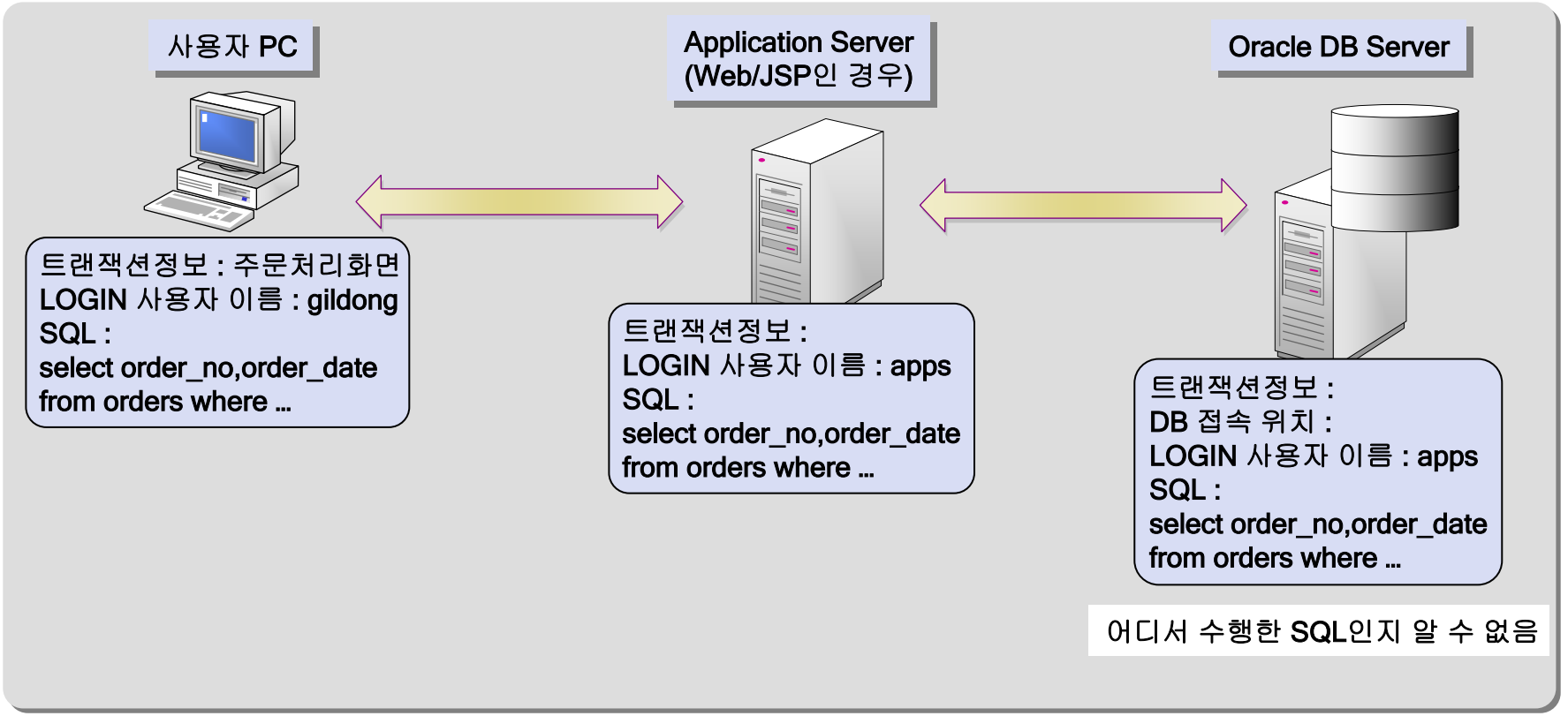
1. DBMS_APPLICATION_INFO
2. DBMS_SYSTEM
3. 기타 Trace 설정방법

II-1. DBMS_APPLICATION_INFO

- SET_MODULE ('*MODULE_NAME*', '*ACTION_NAME*');
  - 프로그램의 논리적 구분단위인 Module 명과 Action명을 저장
- SET_ACTION ('*ACTION_NAME*')
  - 프로그램의 논리적 구분단위인 Action명을 저장
- SET_CLIENT_INFO ('*CLIENT_INFO*')
  - 세션의 부가적인 Client 정보 저장

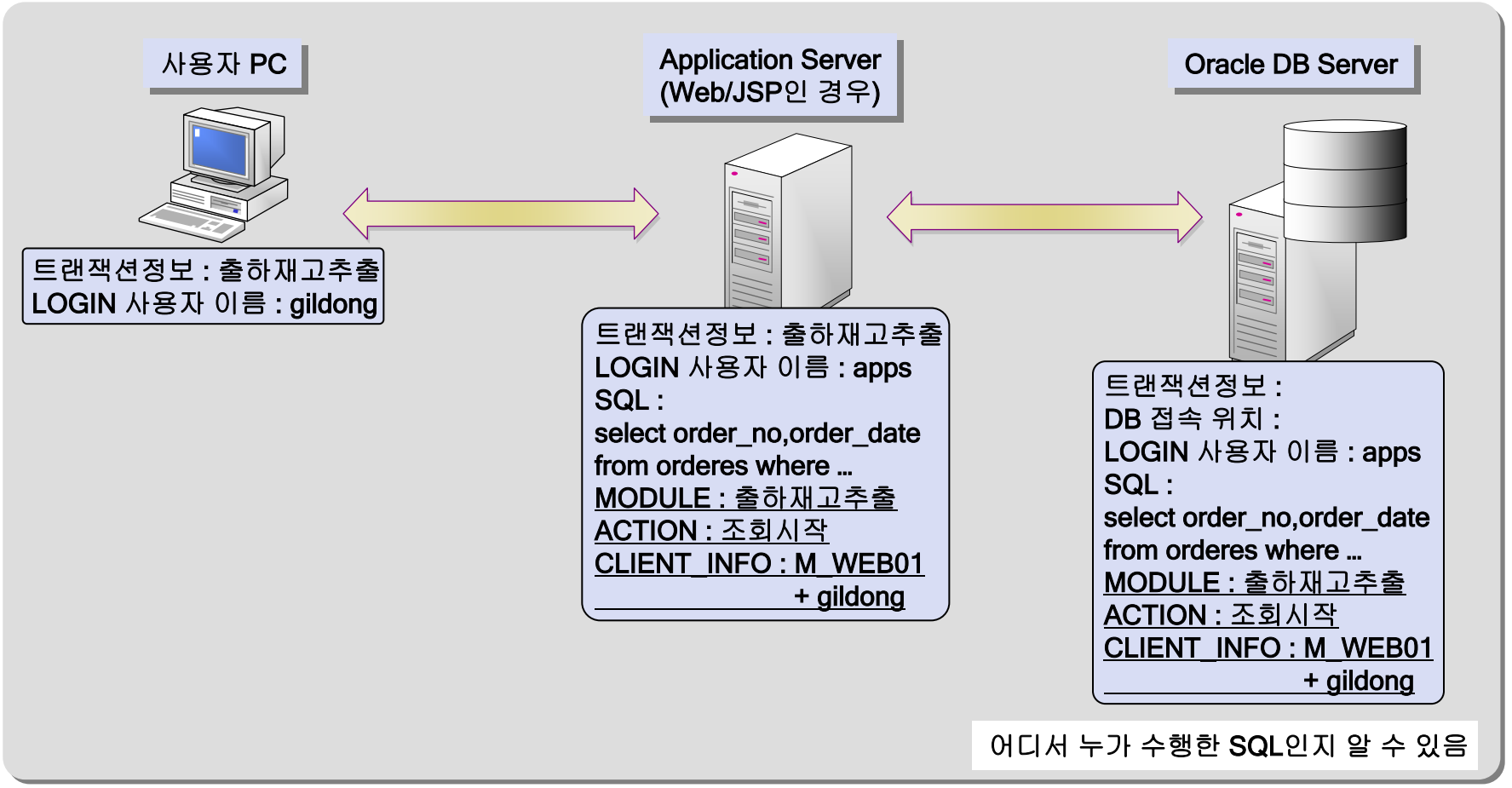
II-1. DBMS_APPLICATION_INFO

- 프로그램 식별을 위한 Oracle의 API



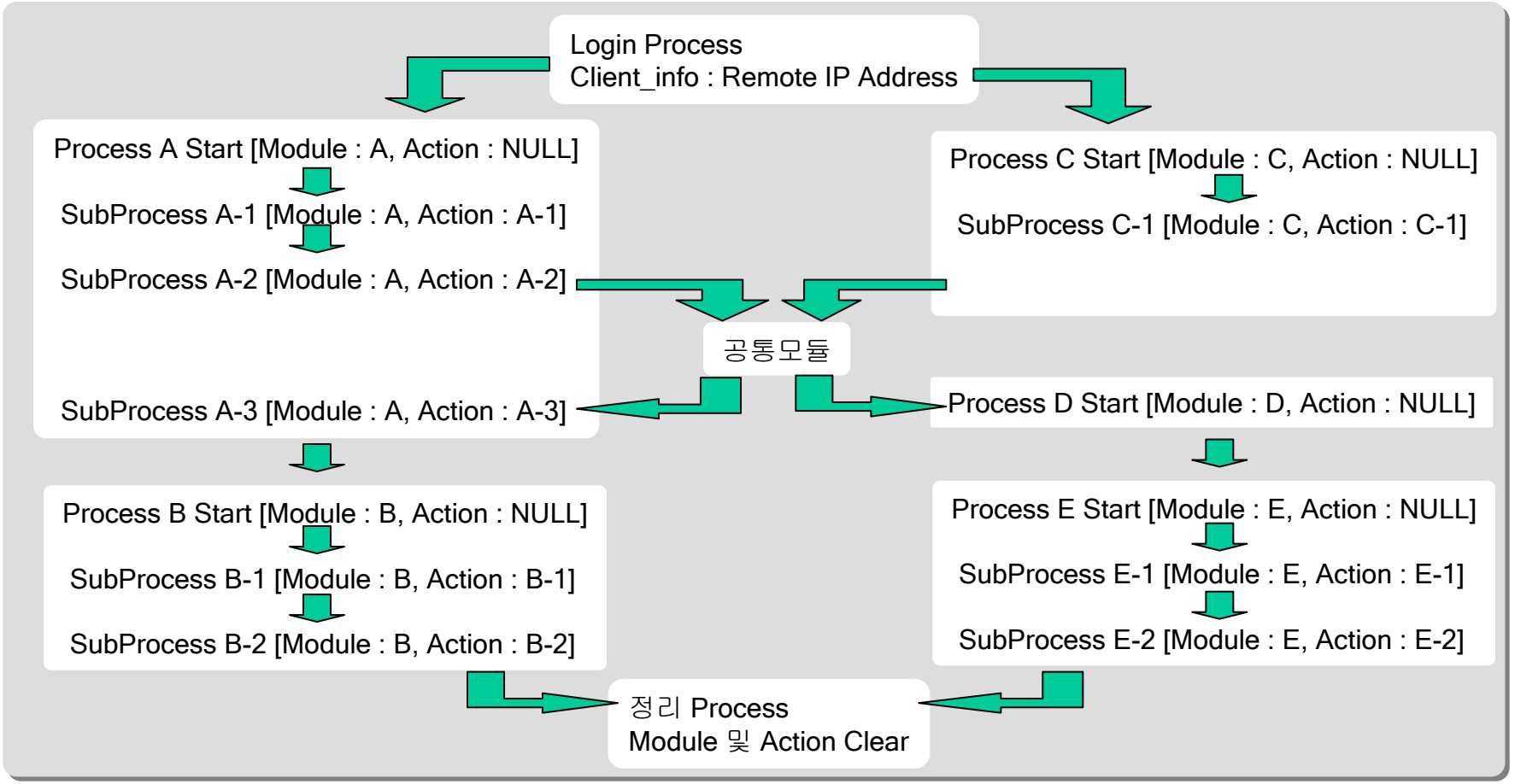
II-1. DBMS_APPLICATION_INFO

- 프로그램 식별을 위한 **Oracle**의 **API (MODULE, ACTION, CLIENT_INFO** 활용 시)



II-1. DBMS_APPLICATION_INFO

● 프로그램 식별을 위한 **Oracle**의 **API** 활용 개념도



• 공통 모듈에는 **Module** 이름이나 **Action**이름을 세팅하지 않는다. (위의 공통 모듈의 경우 **A Process**가 호출할 경우 **Module**은 **A, Action**은 **A-2**로 표시되며 **C Process**가 호출할 경우 **Module**은 **C, Action**은 **C-1**으로 표시됨)

• **ACTION** 이름은 모듈 실행 시간이 길고(**10**분이상) 프로그램이 긴 경우(**1000**라인 이상)에만 중간 중간에 다른 이름으로 설정

II-1. DBMS_APPLICATION_INFO

- 기본 프로그래밍 예 (PL/SQL)

SQL*Plus

```
SQL> EXEC dbms_application_info.set_module( '출하재고추출', '조회시작');
```

```
V$SESSION 모니터링:
 SID Logon_Time       Program      Module           ACTION     Machine      Username
---- ---------------- ------------ ---------------- ---------- ------------ --------
1360 03/29-09:48:03   sqlplusw.exe 출하재고추출     조회시작   EXEMWFORTEST APPS
```

PL/SQL

```
DECLARE
BEGIN
    dbms_application_info.set_module( '출하재고추출', '조회시작');
    ...
```

```
V$SESSION 모니터링:
 SID Logon_Time       Program      Module           ACTION     Machine      Username
---- ---------------- ------------ ---------------- ---------- ------------ --------
1360 03/29-09:48:03   sqlplusw.exe 출하재고추출     조회시작   EXEMWFORTEST APPS
```

```
    dbms_application_info.set_action( '후처리' );
    ...
END;
/
```

```
V$SESSION 모니터링:
 SID Logon_Time       Program      Module           ACTION     Machine      Username
---- ---------------- ------------ ---------------- ---------- ------------ --------
1360 03/29-09:48:03   sqlplusw.exe 출하재고추출     후처리     EXEMWFORTEST APPS
```

II-1. DBMS_APPLICATION_INFO

- 기본 프로그래밍 예 (PRO*C)

```
#include <string.h>
#include <stdlib.h>
#include <sqlda.h>
#include <sqlcpr.h>
main() {
    EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR  stmt[32768];
    VARCHAR  c[31];
    VARCHAR  username[30];
    VARCHAR  password[30];
    EXEC SQL END DECLARE SECTION;
    /****  connect to ORACLE ****/
    strcpy((char *)username.arr, "FORTEST" );
    username.len = (unsigned short) strlen((char *) username.arr);
    strcpy((char *)password.arr, "xxx" );    /* 실제 프로그램에서는 암호를 제대로 줄 것 */
    password.len = (unsigned short) strlen((char *) password.arr);
    EXEC SQL CONNECT :username IDENTIFIED BY :password;
    /****  set Module name ****/
    EXEC SQL EXECUTE
        DECLARE
        BEGIN
              SYS.DBMS_APPLICATION_INFO.SET_MODULE('출하재고추출', '조회시작');
        END;
    END-EXEC;
    /**** 테스트용
    sleep( 30 );
}
```

V$SESSION 모니터링:

SID	Logon_Time	Program	Module	ACTION	Machine	Username
1360	03/29-09:48:03	?@THSED11(TNS V1-V3)	출하재고추출	조회시작	FORTEST	APPS

II-1. DBMS_APPLICATION_INFO

- 기본 프로그래밍 예 (JAVA)

```
import java.sql.*;
...
public class test
{
  Connection         conn;
  CallableStatement  cs;

  public void test() throws SQLException{
  try {
            CallableStatement cs = conn.prepareCall("{call SYS.DBMS_APPLICATION_INFO.SET_MODULE(?, ?)}");
            cs.setString(1,"출하재고추출");
            cs.setString(2,"조회시작");

            cs.executeQuery();
            :
      } finally { }
  }
}
```

V$SESSION 모니터링:

SID	Logon_Time	Program	Module	ACTION	Machine	Username
1360	03/29-09:48:03	JDBC Thin Client	출하재고추출	조회시작	FORTEST	APPS

II-2. DBMS_SYSTEM

- SET_EV (*'SID', 'SERIAL#', EVENT#, LEVEL,EVENT_NAME*);
  - 오라클 EVENT 설정
- SET_BOOL_PARAM_IN_SESSION (*'SID','SERIAL#',PARAM,BVAL)*
  - Boolean 파라미터 설정
- SET_INT_PARAM_IN_SESSION (*'SID,'SERIAL#','PARAM',INTVAL*)
  - Integer 파라미터 설정

Example

```
SQL> exec dbms_system.set_bool_param_in_session
       (10,278,'TIMED_STATISTICS',TRUE);
SQL> exec dbms_system.set_int_param_in_session
       (10,278,'MAX_DUMP_FILE_SIZE', 2147483647);
SQL> exec dbms_system.set_ev(10,278,10046,8,'');
SQL> exec dbms_system.set_ev(10,278,10046,0,'');

Level 0:  Trace off
Level 1:  기본사항
Level 4:  기본사항+Bind 변수값
Level 8:  기본사항+Wait 정보
Level 12:기본사항+Bind변수값+Wait정보
```

II-3. 기타 Trace 설정 방법

자기 세션에 Trace 설정하는방법

```
SQL> alter session set tracefile_identifier = 'MyTrace';
SQL> alter session set timed_statistics=true;
SQL> alter session set max_dump_file_size=unlimited;
SQL> alter session set events '10046 trace name context forever, level 12';
SQL> alter session set events '10046 trace name context off';

SQL> alter session set sql_trace=true;
SQL> alter session set sql_trace=false;
```

II-3. 기타 Trace 설정 방법

Logon Trigger를 이용하여 Trace를 설정하는 방법 (from 8i)

```
CREATE OR REPLACE TRIGGER APPS.LOGIN_TRIG
 AFTER LOGON ON DATABASE
 WHEN (USER='APPS')
 begin
        execute immediate 'alter session set events '||""||'10046 trace name context
                                forever, level 12'||"";
 end;
 /
```

II-3. 기타 Trace 설정 방법

DBMS_MONITOR Package 를 이용하여 Trace 를 설정하는 방법(from 10g)

```
exec DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE (-
                service_name=>'APPS1',-
                module_name=>'GLEDGER',-
                action_name=>'DEBIT_ENTRY',-
                waits=>true,-
                binds=>true,-
                instance_name=>null);

exec DBMS_MONITOR.SERV_MOD_ACT_TRACE_DISABLE(-
                service_name=>'APPS1',-
                module_name=>'GLEDGER',-
                action_name=>'DEBIT_ENTRY');
```

II-3. 기타 Trace 설정 방법

현재 수행중인 SQL 의 Bind 변수값을 확인하는 방법

```
SELECT b.spid
FROM   v$session a , v$process b
WHERE  a.sid=24
AND    a.paddr=b.addr;

SPID
---------
14614

SQL> oradebug setospid 14614
Oracle pid: 13, Unix process pid: 14614, image: oracle@hp11i (TNS V1-V3)
SQL> oradebug dump processstate 10
SQL> oradebug tracefile_name
/home/oracle/admin/ORA920/udump/ora920_ora_14614.trc
```

trace file 안에서 bind 문자열을 검색

```
bind 0: dty=2 mxl=22(22) mal=00 scl=00 pre=00 oacflg=03 oacfl2=2063757200000000
 size=24 offset=0
   bfp=800003fb4005fb88 bln=22 avl=02 flg=05
   value=1
End of cursor dump
```

IV. 질의 응답

http://www.ex-em.com
support@ex-em.com