# 4주차 교육 세미나

2023. 08. 08
DQA1-3 이윤재

**I.** 지난주 피드백

**II.** 메모리 아키텍처

**III.** 프로세스 아키텍처

**IV. SQL**

# Ⅰ. 지난주 피드백

- MAX_SESSION_COUNT 실습
- Character Set 관련 지역, 국가별 설정
- 리스너 동작 방식
- 오라클 Base, Home 차이

# MAX_SESSION_COUNT 실습

## MAX_SESSION_COUNT=1

WTHR_PER_PROC로 나눠 떨어져야 한다는 에러

```
[tibero@localhost bin]$ tbdown
iparam condition check failed. name:MAX_FG_SESSION_COUNT, value: 1
*** Tibero initialization parameter (tip) file failure:
Error (-7200) occurred while processing parameter 'MAX_FG_SESSION_COUNT' and value '1'
 (MAX_FG_SESSION_COUNT must be larger than 0 and equal to or less than MAX_SESSION_COUNT
and must be divided by WTHR_PER_PROC)..
Tip file path = /home/tibero/Tibero/tibero7/config/tibero.tip
```

# MAX_SESSION_COUNT 실습

## MAX_SESSION_COUNT=10

워커 프로세스 1개 생성

```
[tibero@localhost ~]$ ps -ef | grep tbsvr_FGWP
tibero     2808   2805  0 08:52 pts/1    00:00:00 tbsvr_FGWP000  -t NORMAL
 -SVR_SID tibero
```

세션이 10개가 넘자 세션을 더 이상 열 수 없다는 에러.

```
[tibero@localhost ~]$ tbsql sys/tibero@tibero

tbSQL 7

TmaxTibero Corporation Copyright (c) 2020-. All rights reserved.

TBR-12003: Unable to open a session.

SQL>
```

tbsql bin
tbsql tibero
tbsql tibero
tbsql tibero
tbsql tibero
tbsql tibero
tbsql tibero
tbsql tibero
tbsql tibero
tbsql tibero
tbsql...

# Character set

## NLS_LANG 구조

NLS_LANG=*language_territory.characterset*

- Language : 오라클 메시지를 디스플레이에 출력할 때 사용하는 언어
- Territory : 기본 날짜, 숫자에 대한 컨벤션
- Characterset : 문자 인코딩 방식

# Character set

## NLS_LANG 구조

export NLS_LANG=AMERICAN_AMERICA.KO16MSWIN949

```
SQL> select count(*) from test;
select count(*) from test
                    *
ERROR at line 1:
ORA-00942: table or view does not exist
```

export NLS_LANG=KOREAN_KOREA.KO16MSWIN949

```
SQL> select * from test;
select * from test
              *
1행 에러:
ORA-00942: 테이블 또는 뷰가 존재하지 않습니다
```

Oracle nls_characterset이 AL32UTF8로 인코딩되어 다른 인코딩의 한글은 깨짐

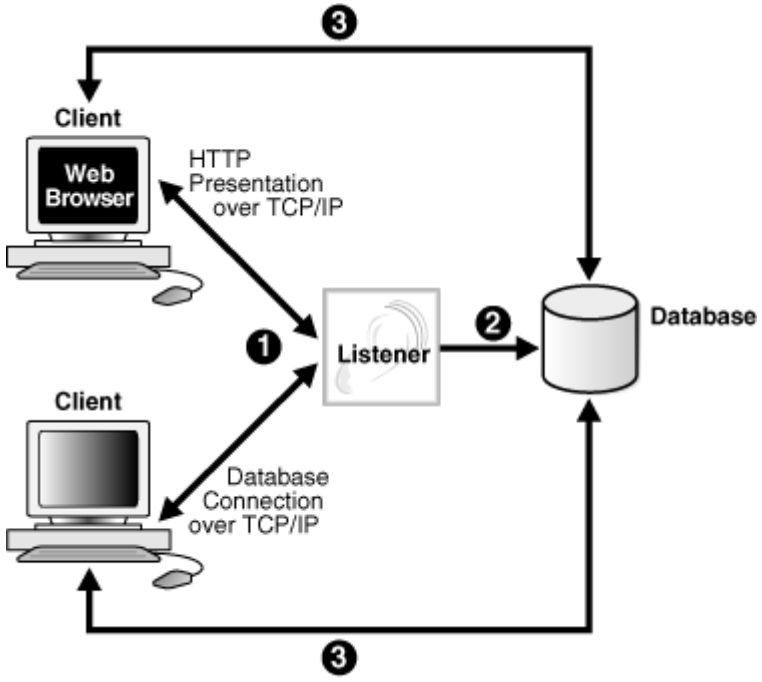
export NLS_LANG=KOREAN_KOREA.AL32UTF8

```
SQL> SELECT COUNT(*) FROM TEST;
SELECT COUNT(*) FROM TEST
                     *
1행에 오류:
ORA-00942: 테이블 또는 뷰가 존재하지 않습니다
```

# Oracle Listener

## Oracle Listener 통신

1. 클라이언트가 리스너에게 연결 요청을 함
2. 리스너가 데이터베이스에게 클라이언트의 요청을 전달한다
3. 클라이언트와 데이터베이스가 직접 통신한다.



# Oracle Listener

## Oracle Listener 통신

- 리스너는 다수의 oracle database에 접근할 수 있다.
- 서비스 이름을 통해 리스너에 접근하면, 리스너는 적절한 인스턴스에 연결해준다.
- 하나의 서비스에 다수의 데이터베이스가 할당될 수 있고, 하나의 데이터베이스는 다수의 서비스에 할당될 수 있다.

```
ORCL2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = moti)(PORT = 1522))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl2)
    )
  )

ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = moti)(PORT = 1522))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl)
    )
  )
```

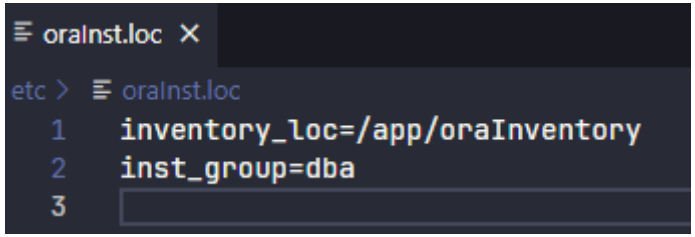
```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = moti)(PORT = 1522))
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1522))
    )
  )

SID_LIST_LISTENER=
  (SID_LIST=
    (SID_DESC=
      (GLOBAL_DBNAME=orcl)
      (ORACLE_HOME=/app/ora19c/19c)
      (SID_NAME=DB19))
      (SID_DESC=
      (GLOBAL_DBNAME=orcl2)
      (ORACLE_HOME=/app/ora19c/19c_2)
      (SID_NAME=orcl2))
```

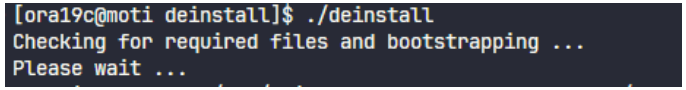
# Oracle Base, Home 차이

## OraInventory

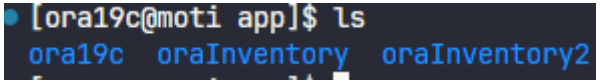
Oracle Inventory 위치를 나타내는 oraInst.loc는 루트 권한을 가지고 있는 유저만 수정할 수 있다.




oraInventory를 oracle_home에 넣어 삭제한다면, oraInventory는 다시 재생성된다. 따라서 oraInventory는 oracle_home에 종속적이다.



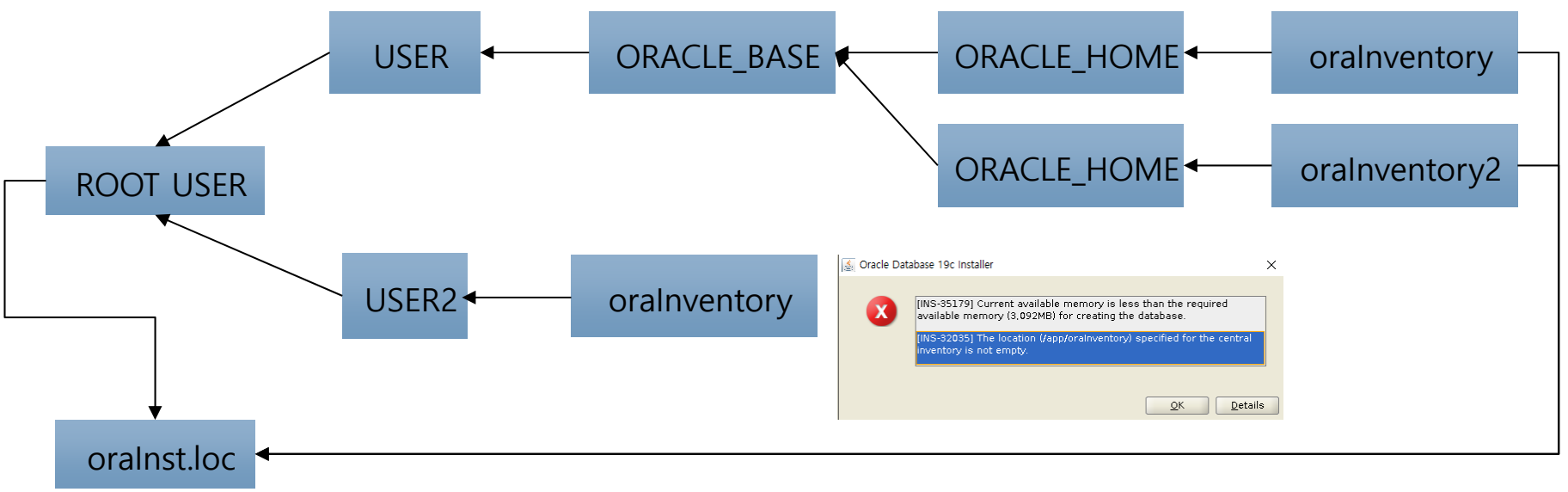

oraInventory2에 해당하는 oracle_home 삭제



# Oracle Base, Home 차이

## OraInventory

```
[ora19c@moti oradata]$ ls
ORCL  ORCL2
```




```
inventory_loc=/app/oraInventory
inst_group=dba
```

# Oracle Base, Home 차이

## Oracle Base와 Home

- Oracle Base는 다수의 Oracle Home을 가질 수 있고 각각의 데이터를 관리하는 관리 공간이다.
- Oracle Home은 하나의 오라클 프로그램의 실행 파일과 설정 파일을 담고 있는 관리 공간이다.
- Oracle Inventory는 오라클 프로그램의 패치 관리를 위한 디렉토리이고, Oracle Base에 여러 개 생길 수 있지만 다른 oracle_base끼리는 공유할 수 없다.
- Oracle Inventory는 ASM이라는 디스크 관리 기능이 여러 ORACLE_HOME에서 사용되도록 지원하는데, ORACLE_HOME을 묶어주는 역할이 필요하기 때문에 ORACLE_BASE가 필요하다.
- 티베로는 ASM이라는 기능이 없기 때문에 BASE 디렉토리가 필요 없다.

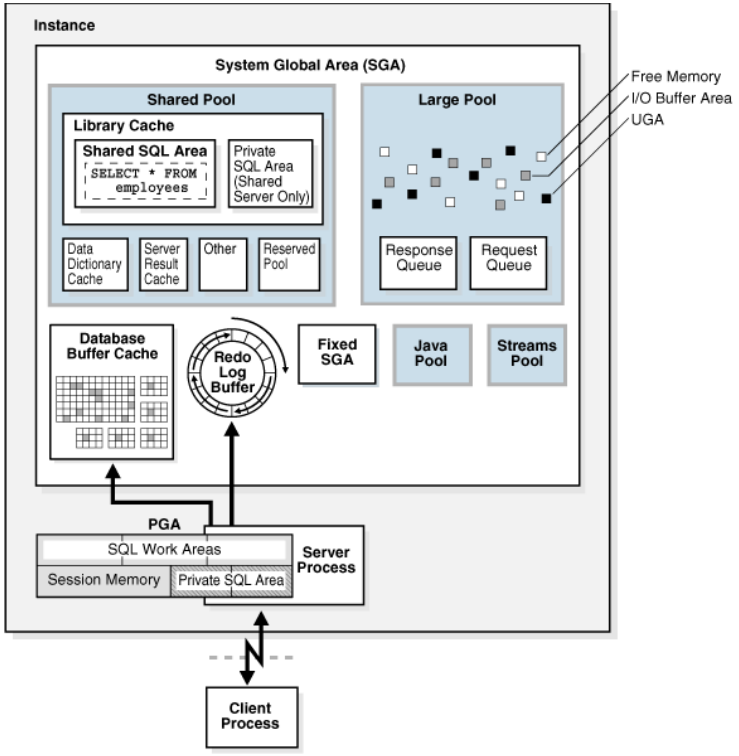
# Ⅱ. 메모리 아키텍처

- 메모리 아키텍처 개요
- UGA
- PGA
- SGA

# 메모리 아키텍처 개요

## 기본적인 메모리 구조

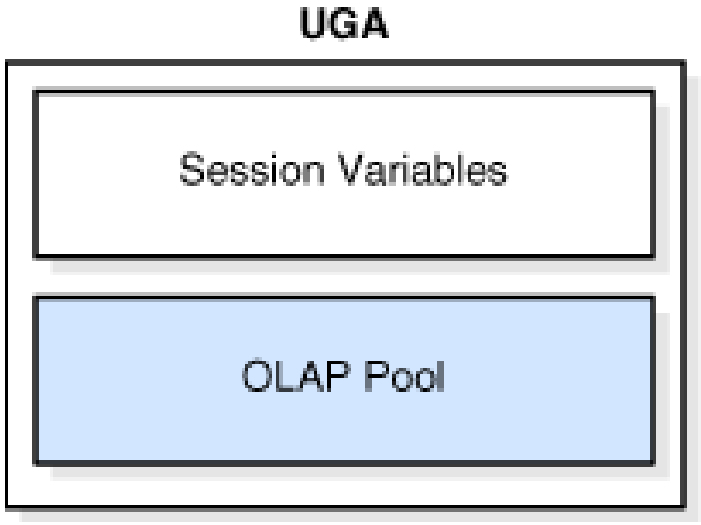
- System global area(SGA)
- Program global area(PGA)
- User global area(UGA)



# UGA

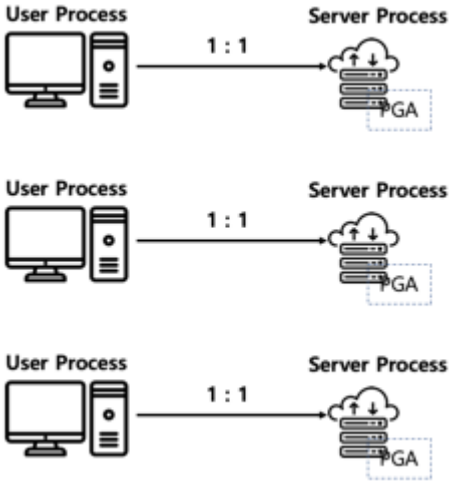
## 기본적인 메모리 구조

- Session Variables : 세션에 필요한 세션 변수들을 저장한다.
- OLAP Pool : OLAP 세션의 데이터 블록을 관리한다.

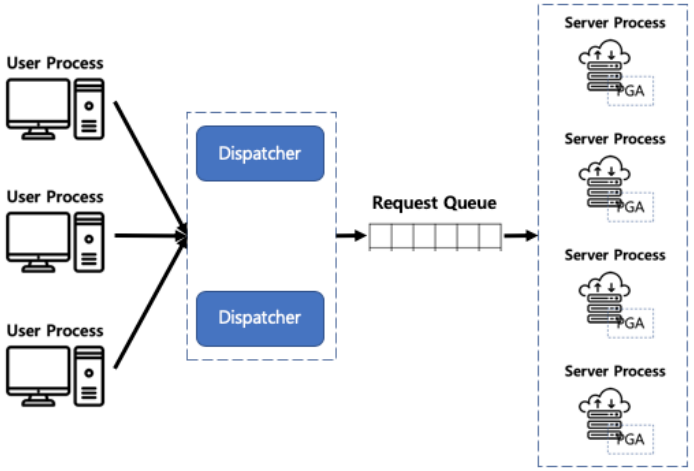


# UGA

## Shared Server와 Dedicated Server



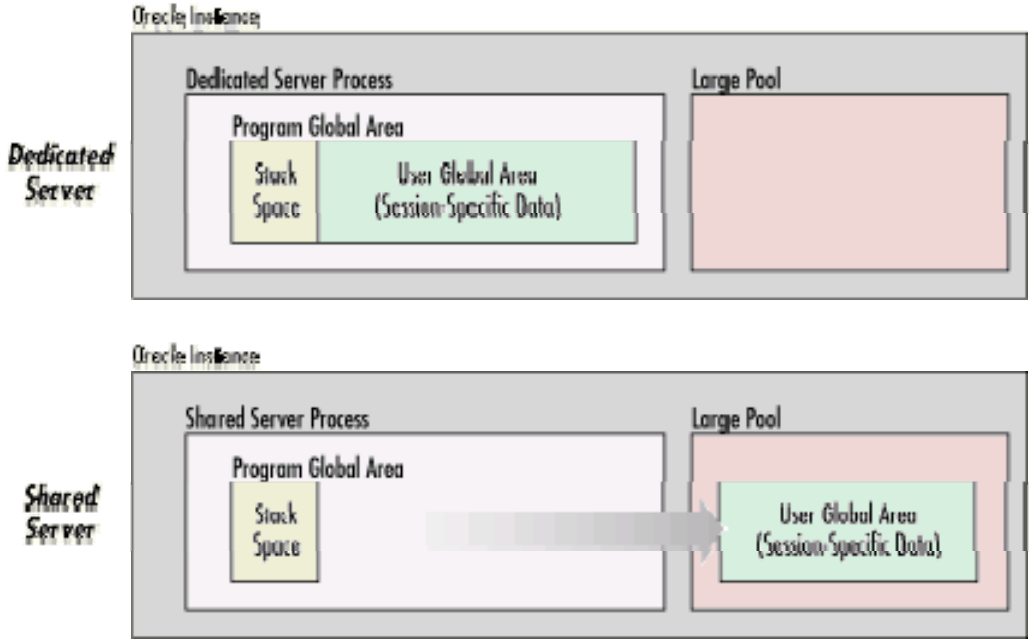

Dedicated Server




Shared Server

# UGA

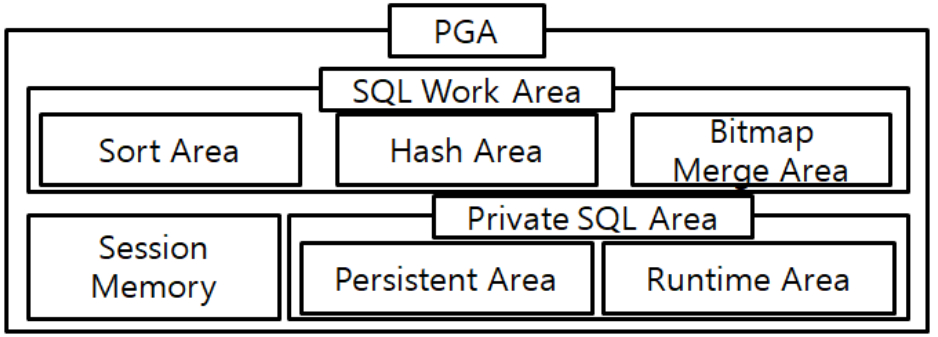
## Shared Server와 Dedicated Server에 따른 UGA 위치



# PGA

## 기본적인 메모리 구조

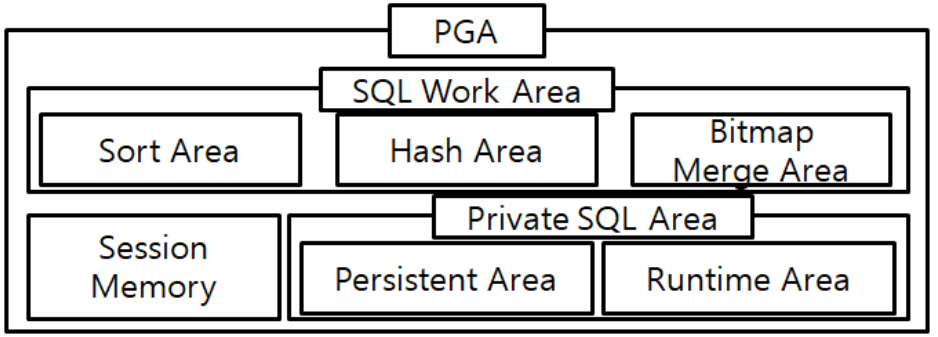
- Private SQL Area : 파싱된 SQL 질의, 바인드 변수 값 저장.
- SQL Work Area : 메모리 집약적인 작업에 사용
- Session Memory : Dedicated server에서 UGA 정보 저장



# PGA

## Private SQL Area

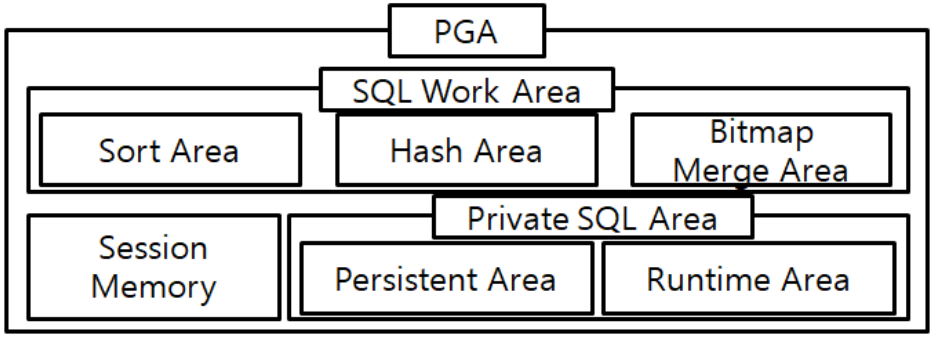
- Persistent Area : 바인드 변수를 담고 있고, cursor가 닫혔을 때 해제된다.
- Runtime Area : 쿼리 수행 상태 정보(반환된 row의 수)



# PGA

## SQL Work Area

- Sort Area : 정렬 작업 시 사용
- Hash Area : Hash Join 시 사용



# PGA

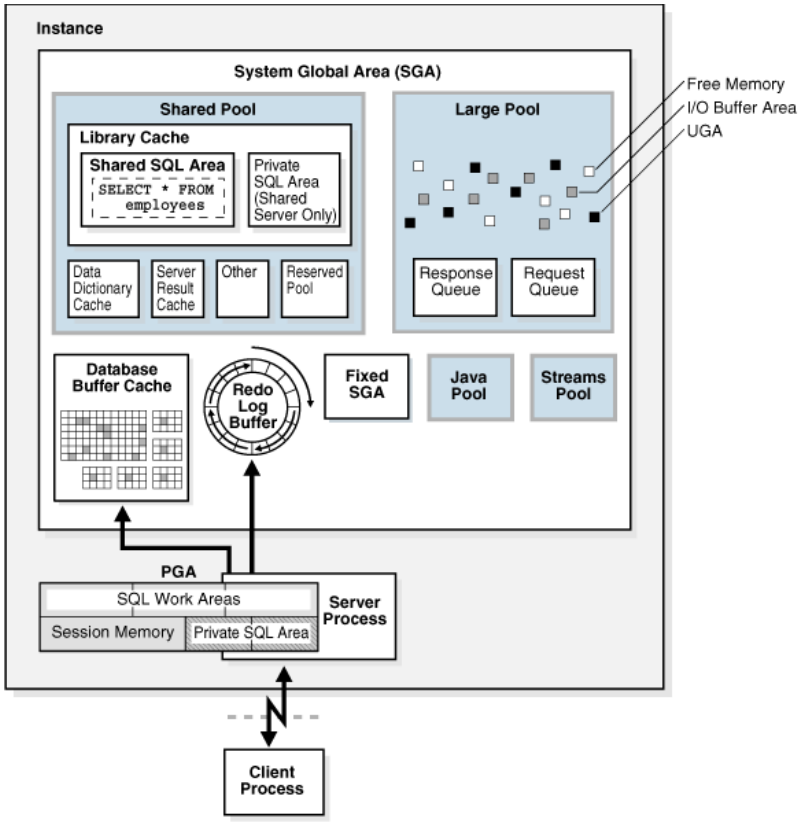
## Shared server 와 Dedicated Server 에서의 PGA 사용

| Memory Area | Dedicated Server | Shared Server |
|---|---|---|
| 세션 메모리의 특성 | Private | Shared |
| Persistent area의 위치 | PGA | SGA |
| DML과 DDL의 run-time area 위치 | PGA | PGA |

# SGA

## 기본적인 메모리 구조

- Database buffer cache
- Buffer pool
- Shared Pool
- Large Pool



# SGA

## Database buffer cache

Database Buffer Cache는 데이터파일로부터 읽어온 데이터 블록의 복사본이 저장된다

[버퍼 캐시의 목적]

- 물리적 I/O의 최적화
- 자주 액세스하는 블록을 버퍼 캐시에 보관하고 자주 액세스하지 않는 블록을 디스크에 기록

# SGA

## Database buffer cache

[버퍼 상태]
- Unused : 사용할 수 있는 버퍼
- Clean : 데이터를 포함하고 있지만 checkpoint될 필요는 없다
- Dirty : 데이터를 포함하고 있고 checkpoint가 필요하다.

[버퍼 access mode]
- Pinned : 유저 세션이 접근하고 있는 상태
- Free : 유저 세션이 접근할 수 있는 상태

# SGA

## Database buffer cache

[버퍼 모드]

- Current mode : 현재 버전의 버퍼를 검색
  - Uncommitted row를 검색할 수 있음

- Consistent mode : 블록의 읽기 일관성을 위해 undo data를 활용해 검색
  - Uncommitted row가 보이지 않음

# SGA

## Database buffer cache

[버퍼 교체 알고리즘]
- LRU-based : 최근에 접근되지 않은 버퍼를 교체하는 알고리즘

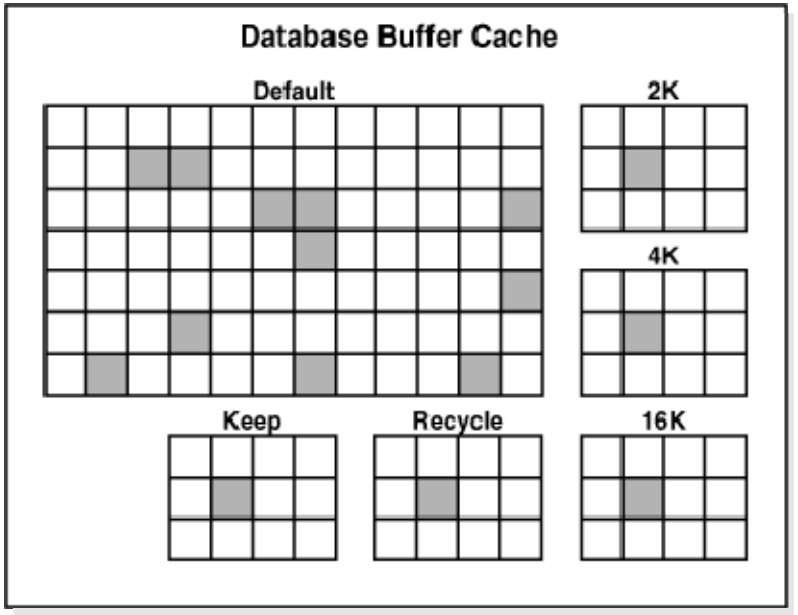
- Temperature-based : 다수의 큰 테이블을 캐싱할 때, 자주 접근되는 테이블을 버퍼에 담고, 그렇지 않은 데이터는 디스크에 저장.

# SGA

## Buffer Pool

버퍼 풀은 버퍼의 집합이다.

- Default Pool : 블록이 일반적으로 캐시 되는 곳
- Keep Pool : 자주 액세스 했지만 공간이 부족하여 밀려난 블록을 저장
- Recycle Pool : 자주 사용되지 않은 블록을 대상으로 저장.



# SGA

## Buffer와 풀 테이블 스캔

기본적으로 버퍼가 디스크로부터 읽힌다면 데이터베이스는 버퍼를 LRU list의 중간에 넣어 hot block이 캐시에 남아있도록 한다.

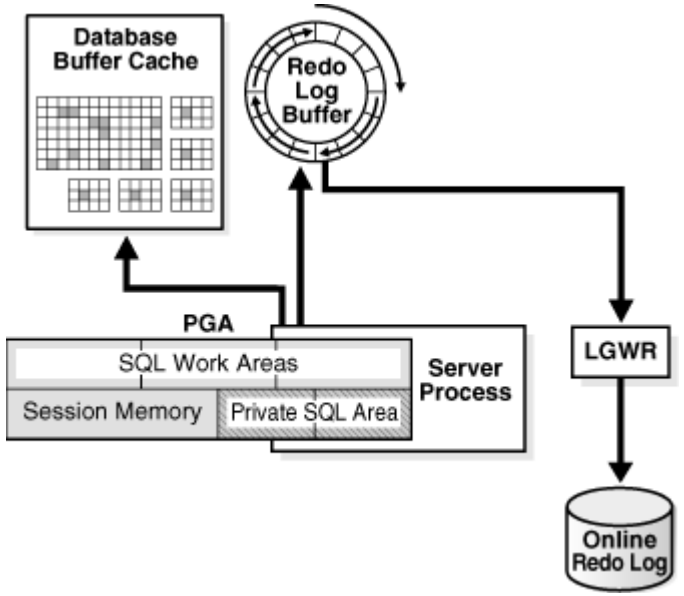
- 테이블 크기가 작을 경우 : 테이블을 메모리에 로드한다.
- 테이블 크기가 클 경우 : 데이터베이스는 형식적으로 버퍼 캐시가 채워지지 않도록 direct path read를 통해 블록을 PGA에 직접 로드하고 SGA를 완전히 우회한다.
- 중간 크기일 경우 : Direct path read를 할 수도 있고 cache read를 할 수도 있다.
  - Cache read일 경우, LRU list의 가장 마지막에 넣는다.

# SGA

## Redo Log Buffer

Redo log buffer는 변경을 표현하는 redo entry를 저장하는 SGA에 존재하는 링형 버퍼이다.

- 유저 메모리의 redo entry를 SGA의 redo log buffer로 복사한다
- Redo entry는 연속적인 버퍼 공간이다.
- 백그라운드 프로세스인 log writer process는 redo log buffer를 디스크의 redo log group에 쓴다.

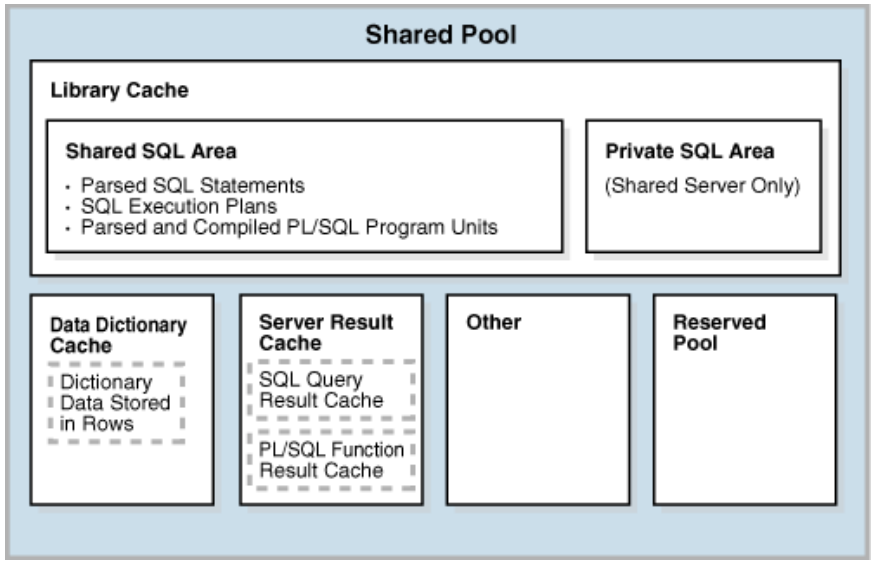


# SGA

## Shared Pool

Shared pool은 프로그램 데이터의 다양한 타입을 캐싱 한다.

- Library Cache
- Data Dictionary Cache : Data dictionary는 데이터베이스와 데이터베이스 구조와 유저에 대한 정보를 포함하는 테이블과 뷰의 집합
- Server Result Cache : Server result cache는 shared pool 내부에 있는 메모리 풀
- Reserved Pool : 연속적이고 큰 메모리 청크를 사용할 수 있는 공유 풀 안에 있는 메모리 공간



# SGA

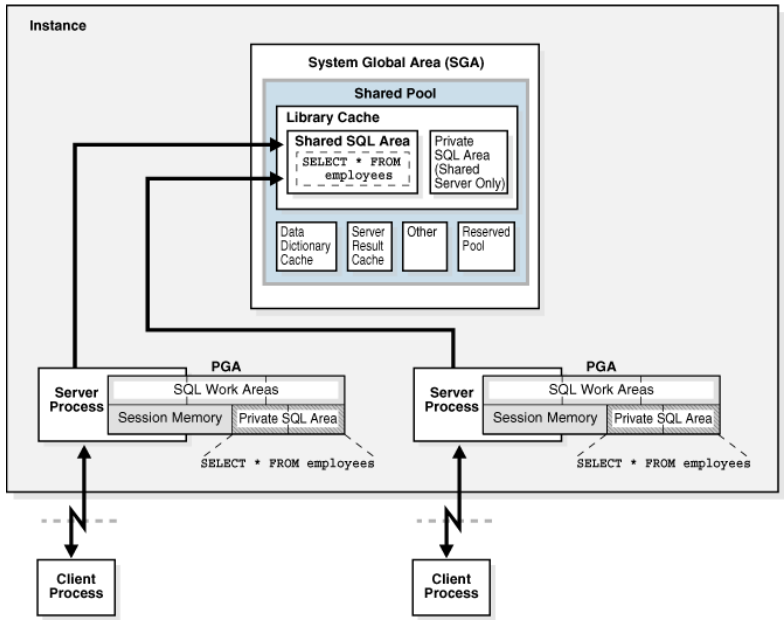
## Libarary Cache

Libaray cache는 실행 가능한 SQL과 PL/SQL 코드를 저장하는 공유 풀 메모리 구조이다.

데이터베이스는 다음과 같은 동작을 한다 :

1. Shared SQL area가 동일한 질의문을 가지는지 확인하기 위해 shared pool을 확인한다.
   - 동일한 질의문이 존재하면, 데이터베이스는 실행을 위해 shared SQL area를 사용한다
   - 만약 동일한 질의문이 존재하지 않는다면, 데이터베이스는 새로운 shared SQL area를 할당한다.

2. 세션 대신 private SQL area를 할당한다.



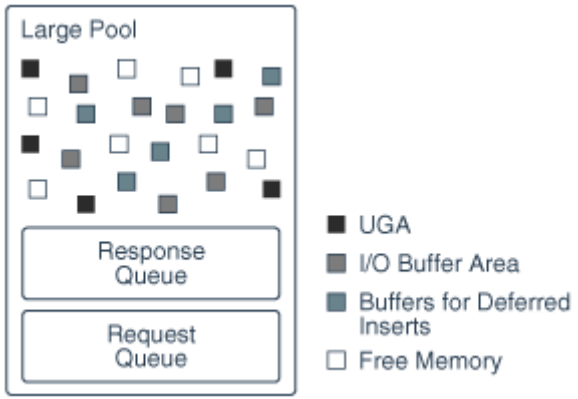
# SGA

## Large Pool

Shared pool에 적절한 크기보다 큰 메모리 할당을 위
한 것이다.

Large pool은 다음에게 큰 메모리 할당을 제공한다 :

- Shared server의 UGA
- 병렬 실행에서의 메시지 버퍼
- Recovery Manager I/O slave의 버퍼



# Ⅲ. 프로세스 아키텍처

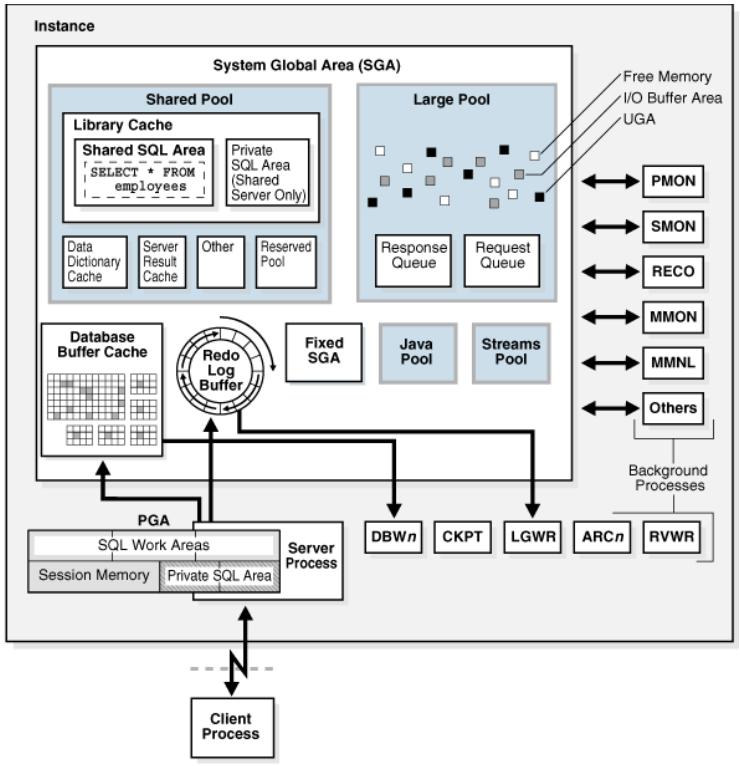
- 클라이언트 프로세스
- 서버 프로세스
- 백그라운드 프로세스

# 프로세스 개요

## 프로세스 타입

- 클라이언트 프로세스 : 어플리케이션이나 오라클 도구 코드를 실행

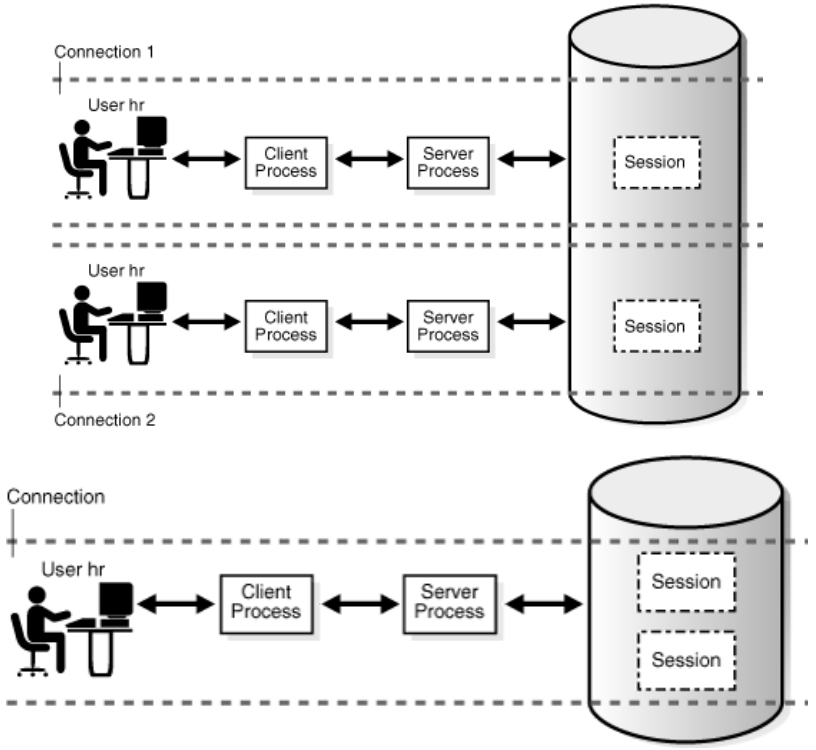
- 오라클 프로세스 : 오라클 데이터베이스 코드를 실행시키는 유닛
  - 백그라운드 프로세스 : 데이터베이스 인스턴스를 시작하고 인스턴스 복구, redo buffer 쓰기 같은 유지보수 업무를 수행
  - 서버 프로세스 : 클라이언트 요청에 따른 작업(SQL 쿼리 파싱, 쿼리 플랜 생성 등등..)



# 클라이언트 프로세스

## Connection과 Session

- Connection은 클라이언트 프로세스와 데이터베이스 인스턴스 간의 물리적 통신 경로다.
- 클라이언트 프로세스와 서버 프로세스 또는 디스패쳐 사이에서 발생함
- 세션은 현재 유저가 데이터베이스에 로그인한 상태를 대표하는 데이터베이스 인스턴스 메모리 내의 논리적인 엔티티이다
- 하나의 커넥션에서 다수의 세션이 생성될 수 있다.



# 서버 프로세스

## 서버 프로세스 개요

오라클 데이터베이스는 인스턴스에 연결되어 있는 클라이언트 프로세스의 요청을 처리하기 위해 서버 프로세스를 생성한다

서버 프로세스는 다음과 같은 작업을 한다 :

- 쿼리 플랜을 생성하고 실행시키고, SQL 질의를 파싱하고 실행시킨다
- PL/SQL 코드를 실행시킨다
- 데이터 파일로부터 읽은 데이터 블록을 버퍼 캐시에 저장

# 서버 프로세스

## Dedicated Server Process와 Shared Server Process

[Dedicated Server Process]
Dedicated server connection에서는, 클라이언트 커넥션은 하나의 서버 프로세스와 하나만 연결된다.

서버 프로세스는 프로세스 정보를 PGA 내부의 UGA에 저장한다

[Shared server Process]
Shared server connection에서는, 클라이언트 어플리케이션이 dispatcher process를 통해 연결하게 된다.

세션을 위한 UGA는 SGA에 저장된다.

# 서버 프로세스

## 서버 프로세스가 생성되는 과정

1. 클라이언트 어플리케이션이 리스너나 브로커에게 새로운 연결을 요청한다
2. 리스너나 브로커가 새로운 프로세스나 스레드의 생성을 초기화한다
3. 운영체제는 새로운 프로세스나 스레드를 생성한다
4. 오라클 데이터베이스는 다양한 컴포넌트나 알림을 초기화한다
5. 데이터베이스가 커넥션 및 커넥션별 코드를 전달한다

# 백그라운드 프로세스

## 백그라운드 프로세스 개요

백그라운드 프로세스는 데이터베이스를 운영하기 위해 필요한 작업이나 다수의 사용자를 상대로 성능을 고도화 시키기 위한 작업을 유지 관리 하기 위해 사용된다.

[필수적인 백그라운드 프로세스]

- Process Monitor Process(PMON) group
- Process Manager (PMAN)
- Listener Registration Process (LREG)
- System Monitor Process (SMON)
- Database Writer Process (DBW)
- Log Writer Process (LGWR)
- Checkpoint Process(CKPT)
- Manageability Monitor Processes (MMON and MMNL)

# 백그라운드 프로세스

## Process Monitor Process group

PMON group은 다른 프로세스의 모니터링과 프로세스 정리를 담당한다

[PMON group]

- PMON : 백그라운드 프로세스의 종료를 감지한다.

- Cleanup Main Process(CLMN) : PMON은 CLMN에 정리 작업을 위임한다. 비정상 종료를 감지하는 작업은 PMON에 남아 있다.
  - 종료된 프로세스, 종료된 세션, 트랜잭션, 네트워크 커넥션, 쉬는 상태의 세션 등등..

- Cleanup Helper Process(CLnn) : CLMN은 정리 작업을 CLnn에 위임한다. Helper 프로세스의 수는 수행할 정리 작업의 양과 정리의 현재 효율에 비례한다.

# 백그라운드 프로세스

## Process Manager(PMAN)

Process Manager(PMAN)은 shared server, pooled server, job queue process 같은 백그라운드 프로세스에 대한 감독을 수행한다

PMAN은 다음과 같은 프로세스를 모니터하고, 생성하고, 종료한다 :

- Dispatcher와 shared server process
- DRCP를 위한 Connection broke와 pooled server process
- Job queue process
- 재시작 가능한 백그라운드 프로세스

# 백그라운드 프로세스

## Listener Registration Process (LREG)

Listener registration process(LREG)는 Oracle Net Listener에 데이터베이스 인스턴스와 dispatcher process에 대한 정보를 등록한다

- 인스턴스가 시작될 때, LREG는 리스너를 polling 하여 실행중인지 여부를 확인한다.
- 만약 리스너가 실행중이면, LREG는 관련 파라미터를 제공한다.
- 만약 실행중이 아니면, LREG는 주기적으로 접근하려고 한다

# 백그라운드 프로세스

## System Monitor Process (SMON)

System monitor process(SMON)은 다양한 시스템 수준의 정리 의무를 가진다.

의무는 다음과 같다 :

- 인스턴스 복구
- 파일 읽기나 테이블 스페이스 오프라인 에러 때문에 인스턴스 복구 중 생략되어 종료된 트랜잭션의 복구
- 사용하지 않은 임시 세그먼트 정리 (extent 할당 작업 실패 시 정리)
- Dictionary-managed 테이블 스페이스 내부의 연속된 사용 가능한 extent 병합

# 백그라운드 프로세스

## Database Writer Process(DBW)

Database writer process(DBW)는 데이터베이스 버퍼의 내용을 데이터 파일로 쓰는 작업을 한다

DBW 프로세스는 다음과 같은 조건에서 더티 버퍼를 디스크로 쓴다 :

- 서버 프로세스가 임계값만큼 스캔을 수행하고 재사용 가능한 빈 버퍼를 찾지 못하면, DBW에게 쓰기 신호를 보낸다

- DBW가 주기적으로 checkpoint를 진행시키기 위해 버퍼 쓰기 작업을 한다.

# 백그라운드 프로세스

## Log Writer Process (LGWR)

Log writer process(LGWR)은 online redo log buffer를 관리한다

다음과 같은 상황에서 LGWR은 마지막으로 쓰여진 시점으로부터 버퍼로 복사된 모든 redo entry를 쓴다 :

- 트랜잭션에서 유저 커밋
- Online redo log switch 발생
- LGWR이 쓰기 작업을 한 뒤로부터 3초 지남
- Redo log buffer가 1/3이 차있거나 1MB의 버퍼 데이터가 포함될 때

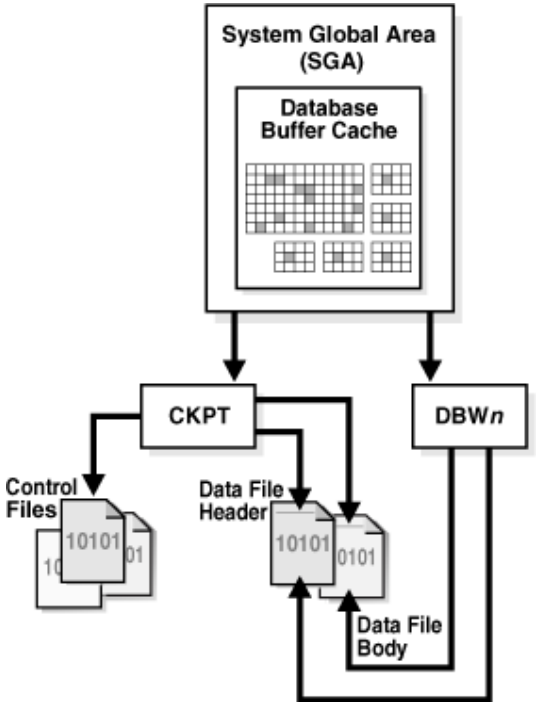
# 백그라운드 프로세스

## Checkpoint Process(CKPT)

Checkpoint process(CKPT)는 컨트롤 파일과 체크포인트 정보를 포함한 데이터 파일 헤드를 업데이트하고 DBW에게 디스크에 블록을 쓰도록 신호를 보낸다

체크포인트 발생 시점 :

- Log switch change
- Log_checkpoint_interval(체크포인트가 발생할 리두 로그 파일 블록 수 지정)
- Log_checkpoint_timeoutShutdown(체크포인트 발생 시간 간격)
- Tablespace offline



# 백그라운드 프로세스

## Manageability Monitor Processes (MMON and MMNL)

Manageability monitor process (MMON)은 Automatic Workload Repository(AWR)에 연관된 많은 작업을 수행한다.

AWR이 사용하는 스냅샷을 생성하고, 최근 수정된 SQL 개체에 대한 통계 값을 캡처할 때 MMON이 기록한다.

# IV. SQL

- 옵티마이저
- SQL 처리

# 옵티마이저

## Optimizer의 사용

오라클 데이터베이스가 SQL 질의를 어떻게 처리 하는지 이해하기 위해서는 optimizer를 이해하는 것이 필요하다.

- 옵티마이저는 가능한 실행 방법을 설명하는 실행 계획을 생성한다.
- 어떤 실행 계획이 가장 효율적인지 결정한다
- 고려 사항으로는 access path, 시스템에서 수집된 통계, 힌트 등이 있다.

# 옵티마이저

## Optimizer의 사용

오라클에 의해 처리된 SQL 질의를 위해 옵티마이저는 다음과 같은 작업을 수행한다

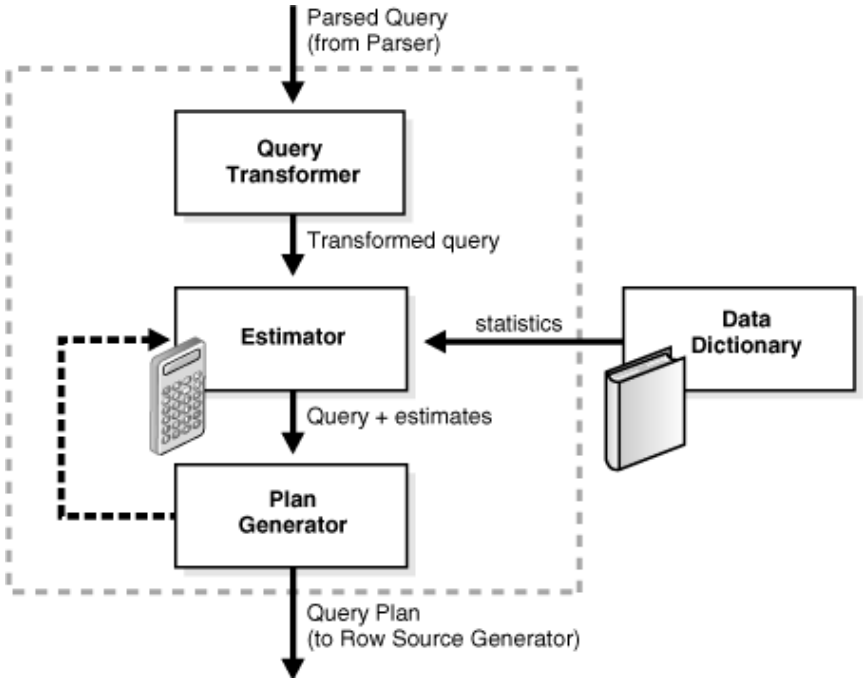
- Expression과 조건의 평가
- 무결성 제약 조건 검사
- 질의 변경
- 옵티마이저 목표 선택
- 접근 경로 선택
- 조인 순서 선택

# 옵티마이저

## Optimizer Component

옵티마이저는 파싱된 쿼리를 받아 query plan을 만들어 제공해 주는데, 세 가지 주요 컴포넌트가 있다

- Query Transformer : 옵티마이저가 더 좋은 실행 계획을 생성할 수 있게 쿼리의 form을 변경한다
- Estimator : Estimator는 주어진 실행 계획의 전체 cost를 결정한다
  - Selectivity
  - Cardinality
  - Cost
- Plan Generator : Plan generator는 제공된 쿼리를 위해 서로 다른 플랜을 생성하고 가장 적은 cost의 플랜을 선택한다



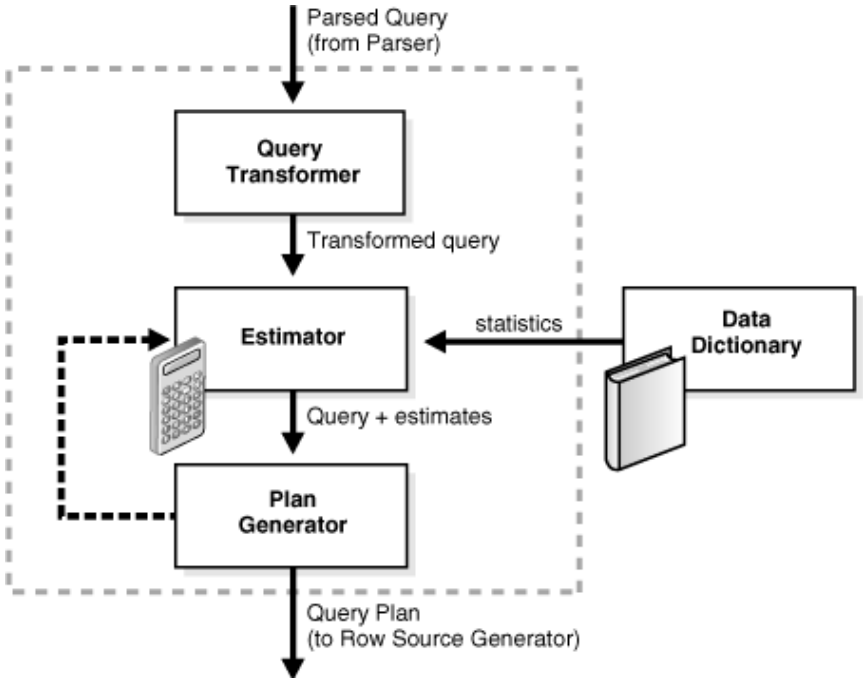
# 옵티마이저

## Access Path

Access path는 데이터베이스로부터 데이터를 가져오는 방법이다

데이터베이스는 테이블로부터 데이터를 가져오기 위한 여러 Access path가 있다:

- Full table scan : high water mark 아래의 데이터까지 스캔한다
- Rowid scans : 데이터 블록과 블록 내부의 row를 특정하여 스캔
- Index scans
- Cluster scans : Cluster index를 스캔하여 rowid를 가져오고 rowid에 기반하여 row를 찾는다
- Hash scans : 같은 hash 값을 가진 row끼리 같은 데이터 블록에 저장하고 hash 값으로 탐색.



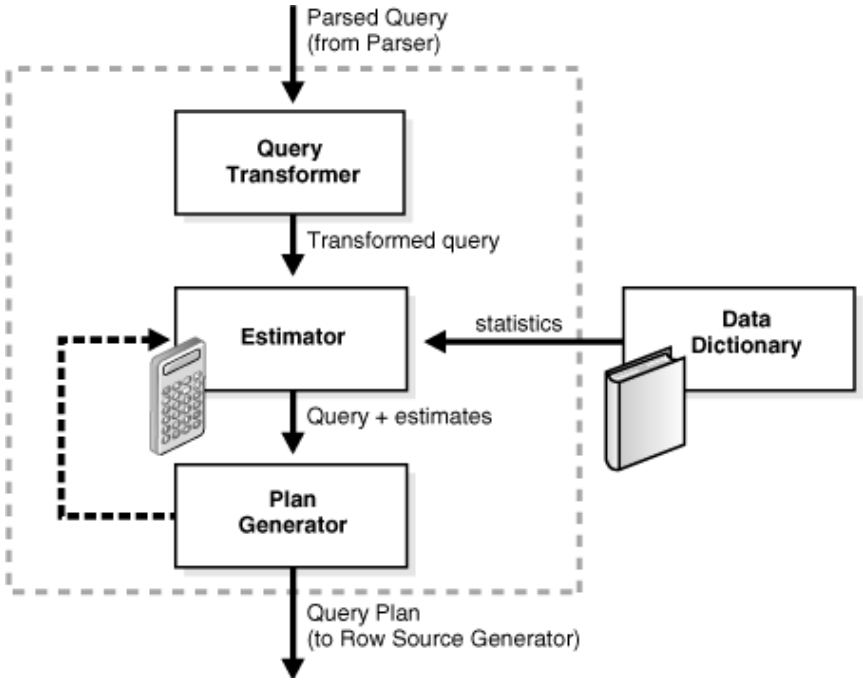
# 옵티마이저

## Optimizer Statistics

Optimizer 통계는 데이터베이스와 데이터베이스 내의 오브젝트에 대한 상세 정보를 묘사하는 데이터의 집합이다

Optimizer Statistics는 다음을 포함한다 :

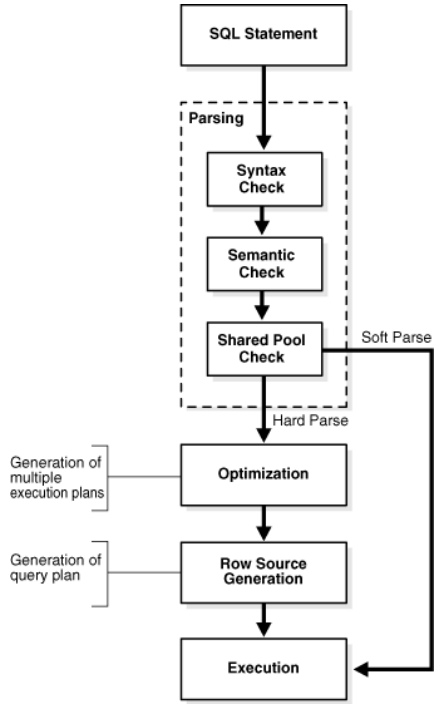
- 테이블 통계 : row의 수, block의 수, row length 평균
- 컬럼 통계 : distinct value의 수, 데이터베이스의 분포, null의 수
- 인덱스 통계 : leaf block의 수와 index level
- 시스템 통계 : CPU와 I/O 성능



# SQL 처리

## SQL 처리 단계

- SQL parsing
- Syntax Check
- Semantic Check
- Shared Pool Check



# SQL 처리

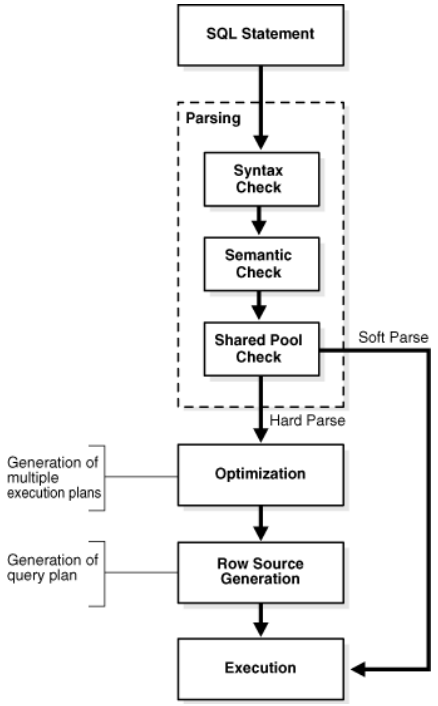
## SQL Parsing

이 단계는 다른 루틴에서 사용할 수 있도록 SQL 질의를 쪼개 데이터 구조로 만든다.

- 어플리케이션은 parse call을 데이터베이스에 보내 질의 실행을 준비한다
- Parse call은 private SQL area를 다루는 cursor를 열거나 생성한다

Parse Call 동안 데이터베이스는 다음과 같은 확인을 한다

- Syntax Check
- Semantic Check
- Shared Pool Check



# SQL 처리

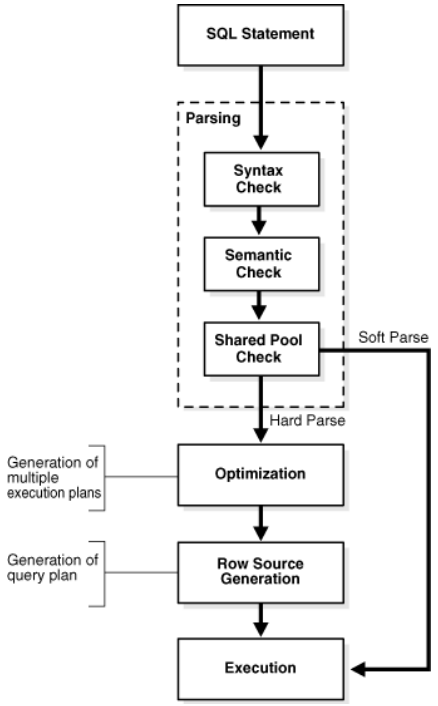
## Shared Pool Check

Parse 단계에서, Shared SQL area를 탐색하여 같은 해시 값을 가진 질의가 있는지 확인한다.

- 질의의 해시 값은 V$SQL.SQL_ID에 보여지는 SQL ID값이다.

Parse 작업은 hash check이나 질의의 타입에 따라 다음의 카테고리로 분류된다 :

- 하드 파싱
- 소프트 파싱



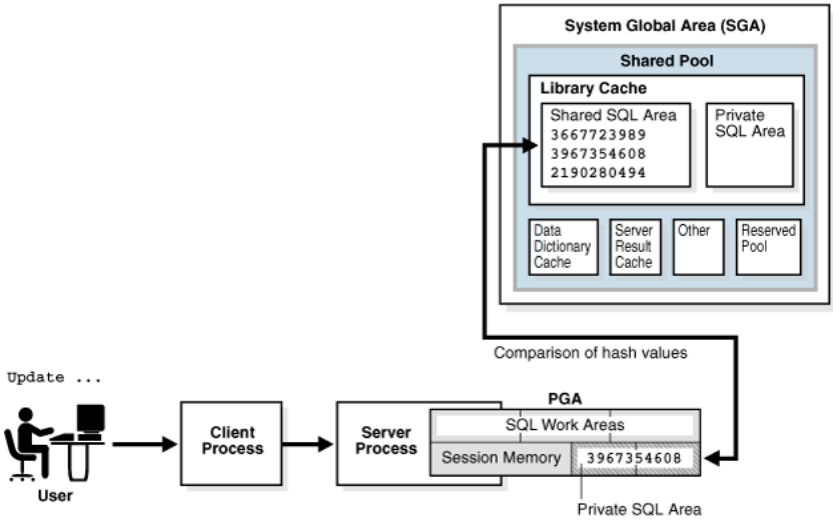
# SQL 처리

## Shared Pool Check

만약 공유 풀에 있는 질의의 해시 값이 동일하더라도
semantic 검사를 통해 의미가 동일한지 검사한다.

서로 다른 유저가 아래의 쿼리를 수행했을 때 :
CREATE TABLE my_table ( some_col INTEGER );
SELECT * FROM my_table;

옵티마이저 환경이 변했을 때 :
ALTER SESSION SET
OPTIMIZER_MODE=FIRST_ROWS;
SELECT * FROM my_table;
ALTER SESSION SET SQL_TRACE=TRUE;
SELECT * FROM my_table;



# SQL 처리

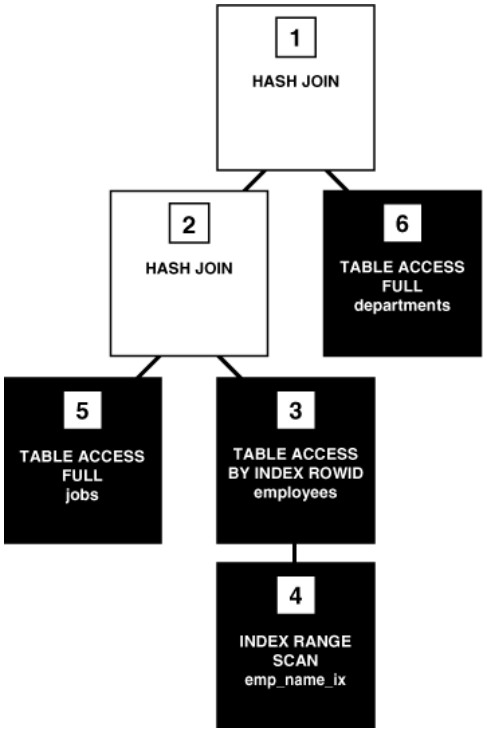
## SQL Row Source Generation

옵티마이저로부터 실행 계획을 받거나 데이터베이스의 나머지 부분에서 사용할 수 있는 query plan이라는 반복 계획을 생성하는 소프트웨어

- Row source generator는 row source의 집합인 row source tree를 생성한다
- Row source는 테이블, 뷰, 조인의 결과나 그루핑의 결과 row 집합.

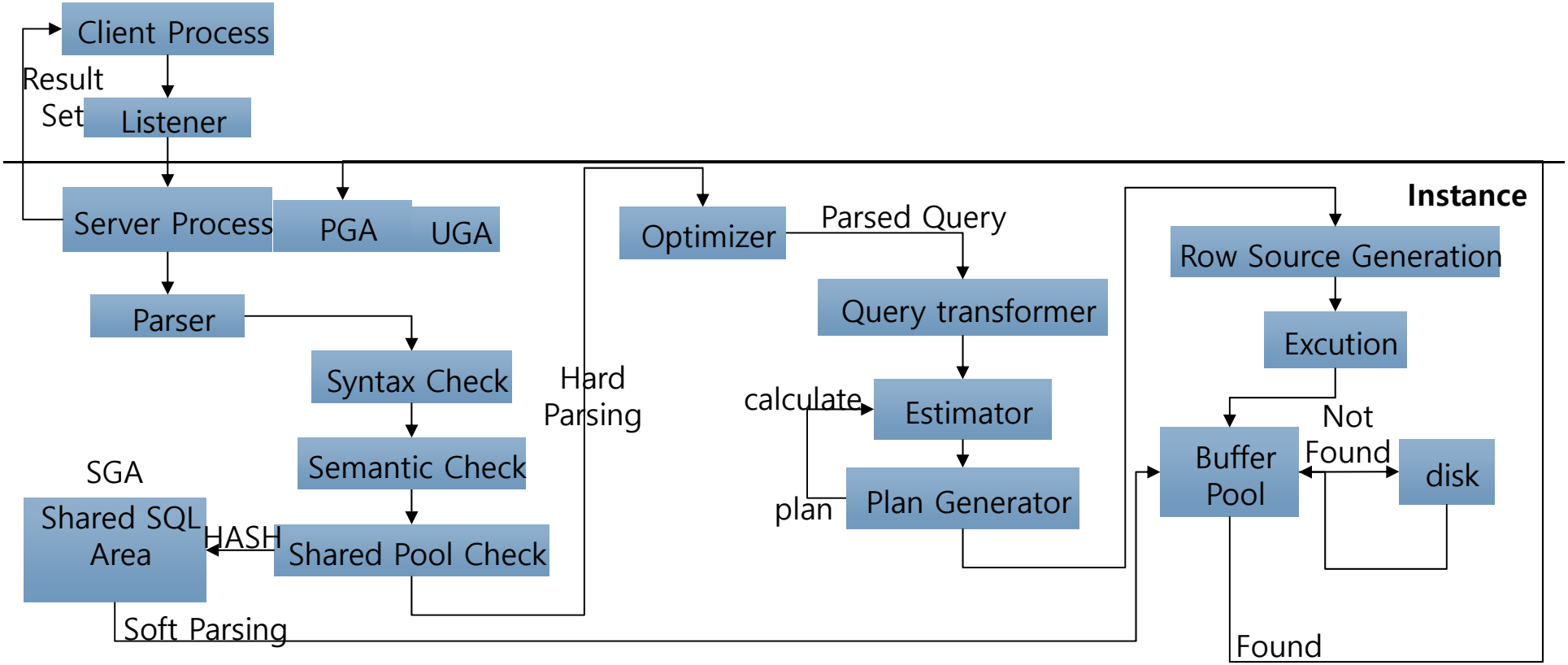
Row source tree는 다음과 같은 정보를 보여준다 :

- 질의에 의한 테이블 정렬
- 질의에 언급된 각각의 테이블을 위한 접근 방법
- 질의의 조인 작업에 의해 영향을 받은 테이블 조인 방법
- 필터, 정렬, 집계 같은 데이터 작업



# SQL 처리

## SQL(Select) 처리 과정(Dedicated Server)



# Q & A

# 감사합니다