Tibero Architecture

Contents

TmaxTibero

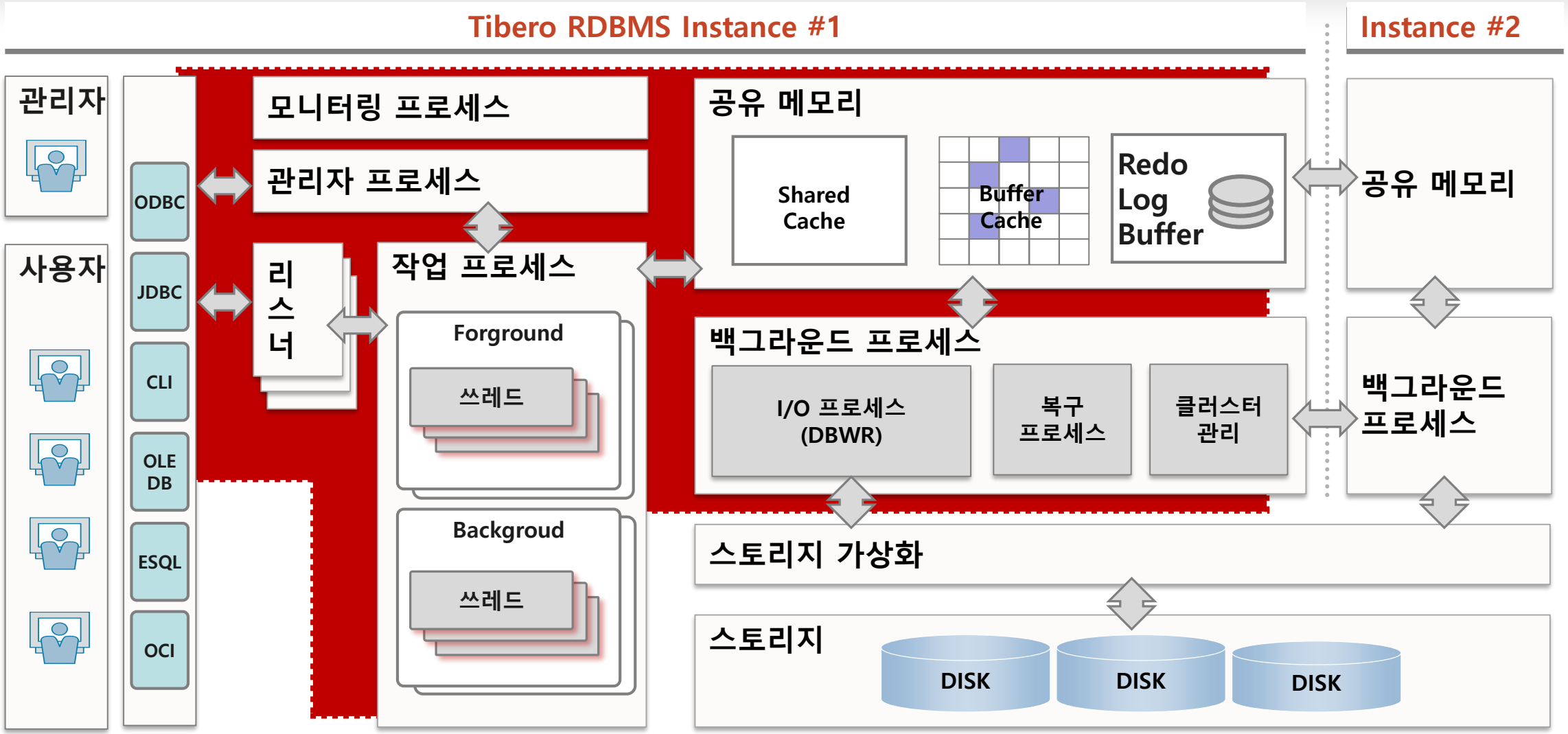
0. 교육소개

Contents

과정	내용
Tibero Architecture	**티베로 SQL 아키텍쳐와 주요 기능을 학습합니다.** • 교육기간 : 1일 • 교육대상 : Tibero 사용자

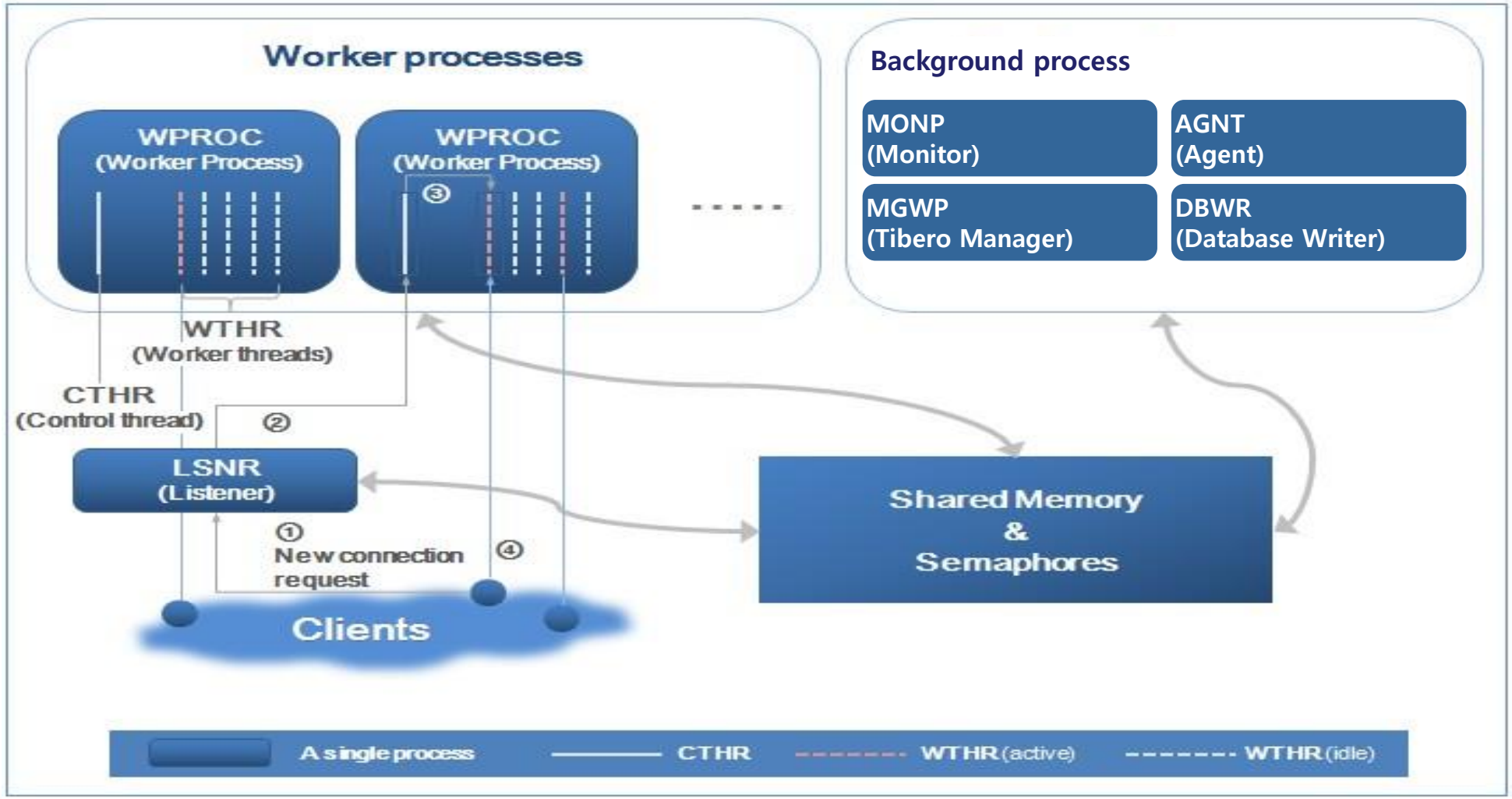
1. 티베로 인스턴스

Tibero 전체 구조



Tibero Process

- **Tibero 프로세스 구조**
  - **대규모 사용자 접속을 수용하는 다중 프로세스 및 다중 스레드 기반의 아키텍쳐 구조**
    - Listener, 워커프로세스 (Working Process 또는 Foregroud Process) , Background Process



Tibero Process

- **Listener**
  - 클라이언트의 새로운 접속 요청을 받아 이를 유휴한 워커 프로세스에 할당
  - 클라이언트와 워커 프로세스간에 중계 역할을 담당하며, 별도의 실행 파일인 tblistener를 사용하여 작업
  - 모니터링 프로세스에 의해서 생성되며 외부에서 강제 종료하더라도 자동으로 재 시작 됨
  - Listener Port

Listener port
< $TB_SID.tip > LISTENER_PORT=8629

Extra listener port	
정적	< $TB_SID.tip > LISTENER_PORT=8629 EXTRA_LISTENER_PORTS=8639;8640;
동적	alter system listener add port 8799; alter system listener delete port 8799;

  - 클라이언트의 새로운 접속 요청이 이루어지는 순서
    1) Client가 접속 요청
    2) Listener는 현재 빈 WTHR이 있는 프로세스를 찾아서 이 사용자의 접속 요청을 CTHR에게 넘겨줌.
    3) 요청을 받은 CTHR은 자기 자신의 WTHR 상태를 체크해서 일하지 않는 WTHR에게 할당
    4) WTHR은 Client와 인증 절차를 걸쳐 세션 시작

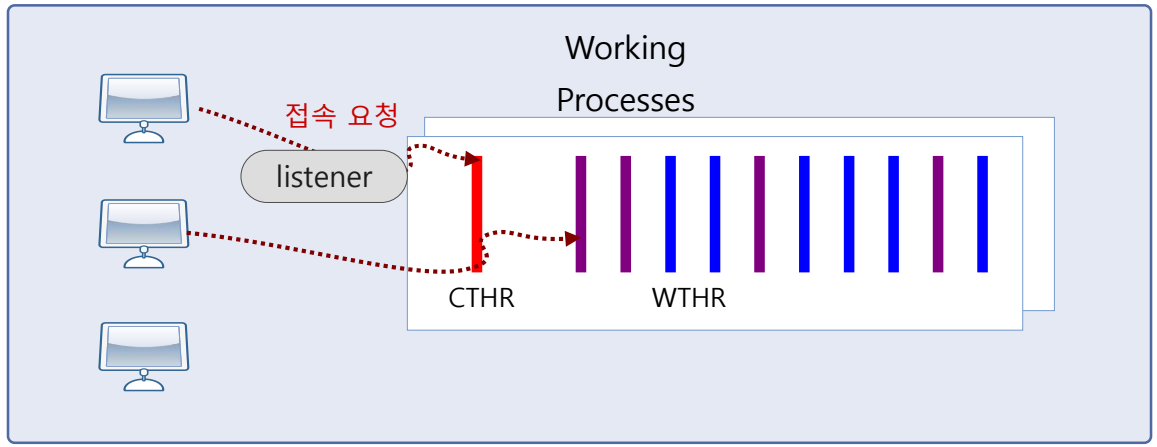
Tibero Process

- 워커프로세스 (Worker Processes 또는 Foregroud Process)

  - 클라이언트와 실제 통신을 하며, 사용자 요구 사항을 처리 하는 프로세스
  - Foregroud Worker Process는 리스너를 통해 들어온 온라인 요청 처리
  - Backgroud Worker Process는 Internal Task나 Job Scheduler에 등록된 배치 작업을 수행

  - CTHR (Control Thread)
    - 각 Working Process마다 하나씩 생성. 서버 시작 시에 지정된 개수의 Worker Thread를 생성.
    - 시그널 처리 담당.
    - I/O Multiplexing을 지원하며, 필요한 경우 워커 스레드 대신 메시지 송/수신 역할 수행.

  - WTHR (Worker Thread).
    - 각 Worker Process마다 여러 개 생성.
    - Client 가 보내는 메시지를 받아 처리하고 그 결과를 리턴.
    - SQL Parsing, 최적화, 수행 등 DBMS가 해야 하는 대부분의 일 처리.



Tibero Process

- **Background Processes**

  - 사용자의 요청을 직접 받아들이지는 않고, 워커스레드나 다른 배경 프로세스가 요청할 때, 혹은 정해진 주기에 따라 동작하며 주로 시간이 오래 걸리는 디스크 작업 담당

  - 독립된 프로세스로서, 사용자의 요청과 비동기적으로 동작.

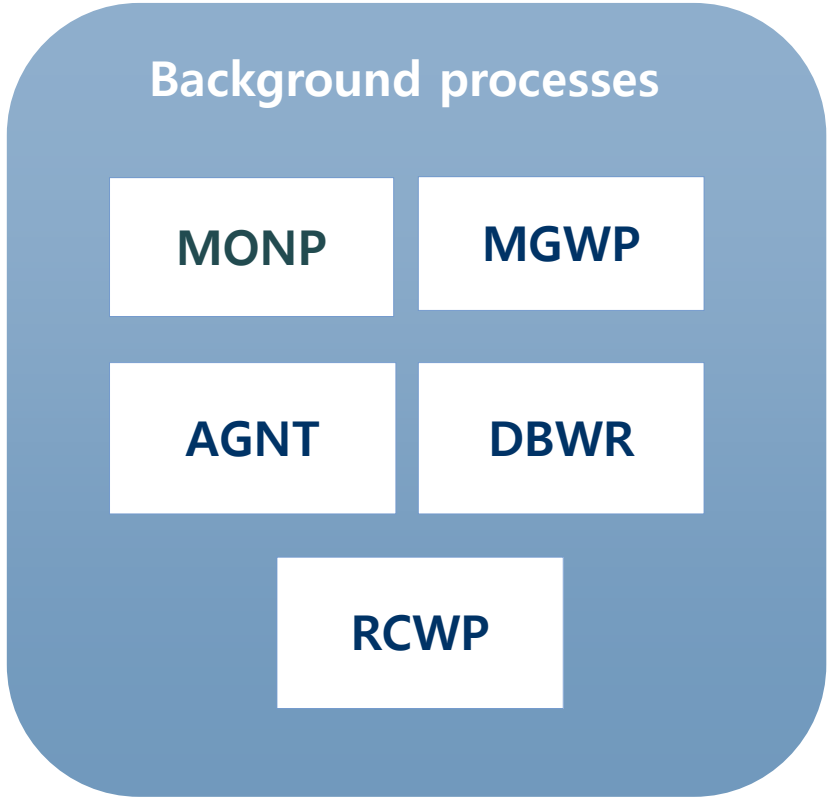
  - 감시 프로세스(MONP: monitor process)

  - 매니저 프로세스(MGWP: manager worker process)

  - 에이전트 프로세스(AGNT : agent process)

  - 데이터베이스 쓰기 프로세스(DBWR : database writer)

  - 복구 프로세스(RCWP : recover worker process)



Tibero Process

Background Processes

감시 프로세스(MONP: monitor process)

- Tibero 기동 시 최초로 생성되는 종료 시에도 마지막에 종료
- Tibero 기동 시, 리스너를 포함한 다른 프로세스를 생성, 주기적으로 각 프로세스 상태 점검
- 교착 상태 (Deadlock)도 검사

매니저 프로세스(MGWP)

- 시스템 관리 용도 프로세스
- 관리자의 접속 요청을 받아 이를 시스템 관리 용도로 예약된 워커 스레드에 접속을 할당
- 기본적으로 워커 프로세스와 동일한 역할을 수행하지만 리스너를 거치지 않고, 스페셜 포트를 통해 직접 접속
- SYS 계정만 접속이 허용, LOCAL 에서만 접속 가능

Tibero Process

Background Processes

에이전트 프로세스(AGNT : agent process)

- ➢시스템 유지를 위해 주기적으로 처리해야 하는 Tibero 내부의 작업을 담당
  - Internal Task나 Batch Job이 언제 수행되어야 하는지 판단은 AGENT 프로세스가 담당 하지만, 실제 수행은 포어그라운드 혹은 백그라운드 워커 프로세스에게 의뢰하는 구조임
- ➢다중 스레드(multi-threaded) 기반 구조로 동작하며, 서로 다른 용도의 업무를 스레드별로 나누어 수행

데이터베이스 쓰기 프로세스(DBWR)

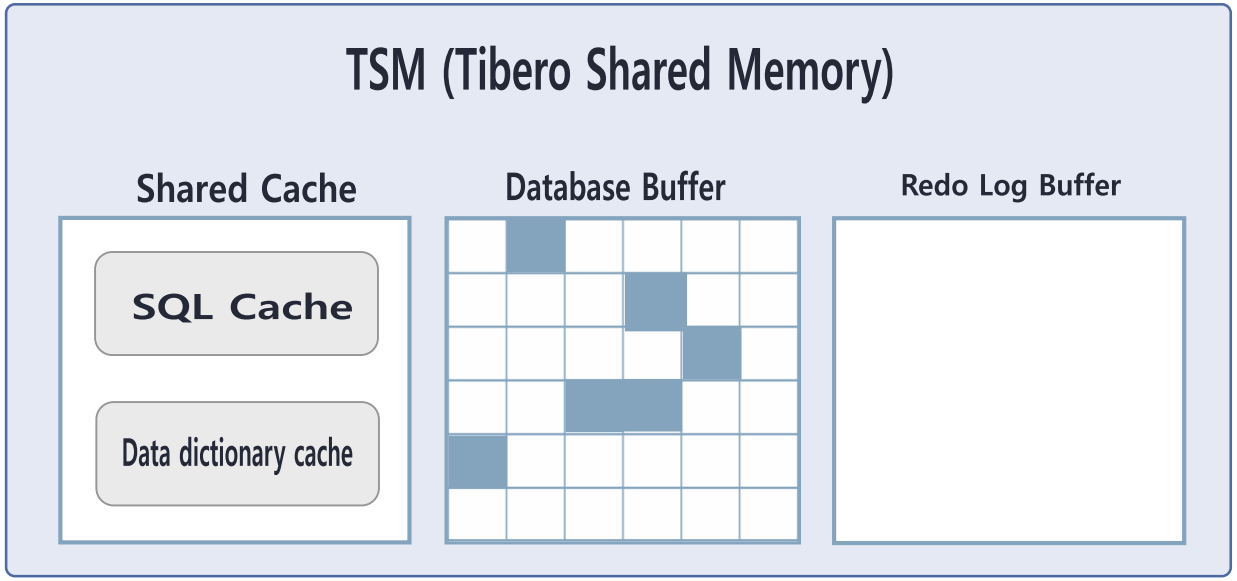
- ➢데이터베이스에서 변경된 내용을 디스크에 기록하는 일과 연관된 스레드들이 모여 있는 프로세스
- ➢사용자가 변경한 블록을 디스크에 주기적으로 기록하는 스레드
- ➢리두 로그를 디스크에 기록하는 스레드
- ➢두 스레드를 통해 데이터베이스의 체크포인트 과정을 관할하는 체크포인트 스레드

복구 프로세스(REWP)

- ➢복구 전용 프로세스
- ➢Crash / Instance Recovery 수행

Tibero Memory

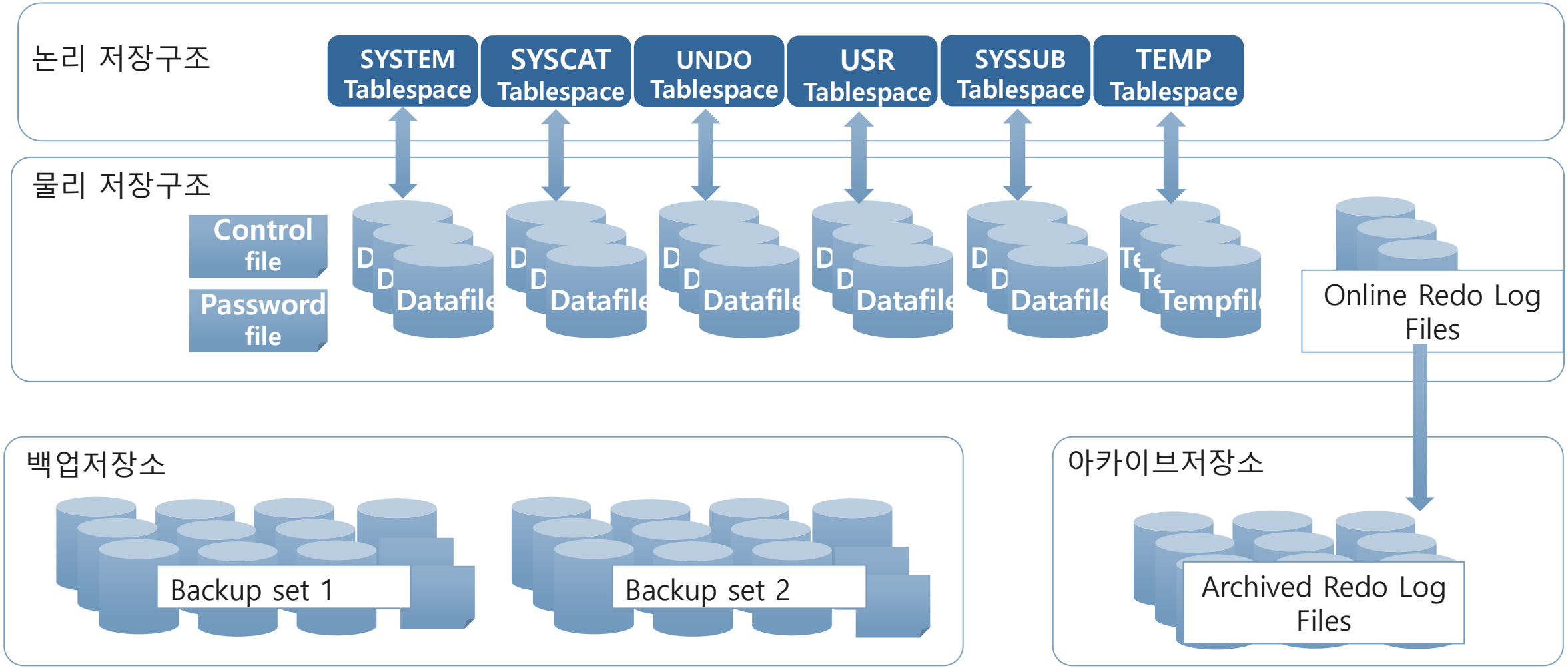
- Tibero Shared Memory (TSM)
  - 인스턴스에 대한 데이터와 제어 정보를 가지는 공유 메모리 영역
  - 사용자가 동시에 데이터를 공유
  - Database Buffer, Redo Log Buffer, SQL Cache, Data Dictionary Cache 로 구성됨.
  - Backgroud Process는 인스턴스가 시작될 때 TSM 영역을 할당하고, 인스턴스가 종료하면 할당 해제
  - TSM의 전체 크기는 인스턴스가 시작될 때 생성되어 고정.



2. 데이터베이스 저장구조

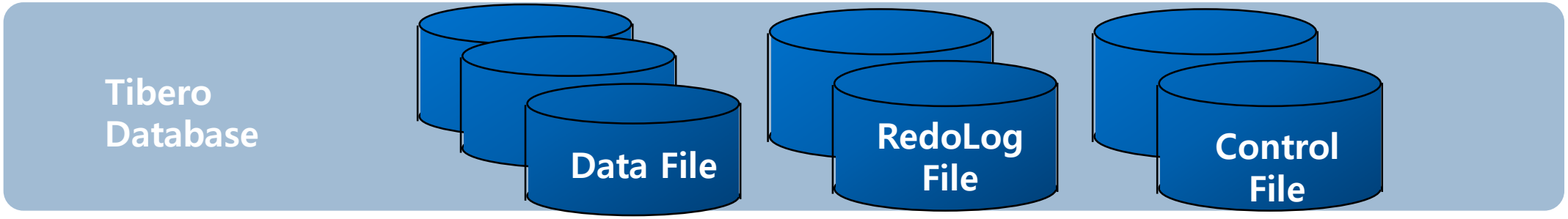
Tibero Database 구조

- **Tibero Database 저장소 구조**



Tibero Database 구조 (물리 저장구조)

Tibero Database 파일



- 하나 or 그 이상의 Control files, Datafiles 그리고 Redo log files.

- Datafiles
  - Tables & Indexes 들과 같은 Logical structure들은 물리적으로 Datafiles에 저장 되는 파일.

- Redo Log Files
  - 복구를 위하여 Database에서 변경된 모든 것을 기록 하는 파일.

- Control Files
  - Database의 물리적 구조와 상태를 기록 하는 파일.

Tibero Database 구조 (물리 저장구조)

Tibero Database 파일 저장 방식

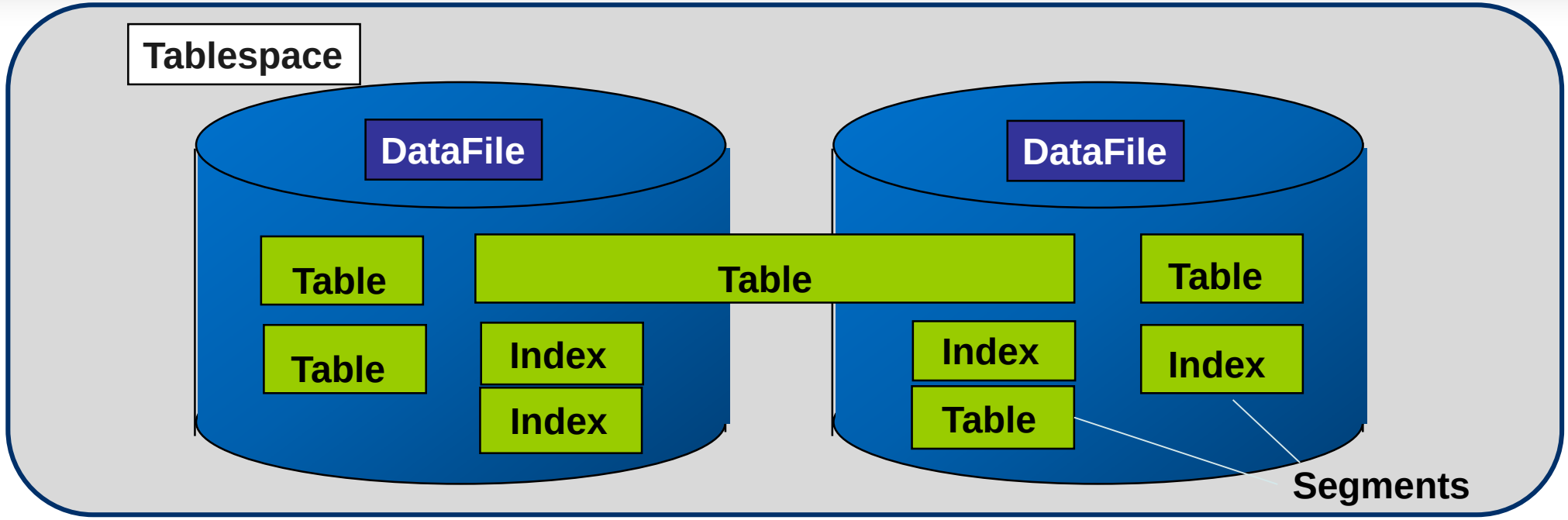
- 운영체제의 파일 시스템
  - 일반적으로 파일시스템은 논리볼륨관리자(LVM) 에 의해 생성된 논리볼륨을 기반으로 구축되어 있고, LVM 은 서로 다른 물리적 디스크 영역을 하나의 연속된 주소 공간으로 결합하여 제공함

- Tibero Active Storage (TAS)
  - Tibero Database 에서 사용하는 전용 파일시스템
  - Tibero TAC 환경에서 TAS 를 기반으로 데이터베이스를 생성하여 운영할 수 있음

- RAW 장치
  - Raw Device 는 파일시스템으로 포맷되어 있지 않은 디스크 파티션 혹은 논리 볼륨으로, 파일시스템과 달리 캐시를 거치지 않고 직접 I/O 를 수행할 수 있음
  - Tibero TAC 환경에서 TAS 를 기반으로 데이터베이스를 생성하여 운영할 수 있음

- 클러스터 파일 시스템
  - 클러스터 파일 시스템은 여러 컴퓨터에서 파일의 저장소를 공유 하는 기능을 제공함
  - Tibero TAC 환경에서 클러스터 파일시스템을 기반으로 데이터베이스를 생성하여 운영할 수 있음

Tibero Database 구조 (물리 저장구조)



- Datafiles
  - Tibero Database 는 데이터 파일 이라는 구조에 데이터베이스 데이터를 저장합니다.
  - Tablespace는 하나이상의 물리적인 datafile을 가지며, 하나의 datafile은 오직 하나의 Tablespace에 포함
  - 세그먼트는 하나 이상의 데이터 파일에 걸쳐 있을 수 있지만, 여러 테이블스페이스에 걸쳐져 있는 것은 아님

Tibero Database 구조 (물리 저장구조)

- 일반 테이블스페이스의 영속적인 데이터 파일
  - 테이블, 인덱스와 같은 영구적인 스키마 객체가 포함되며, 데이터 파일에 저장 관리됨

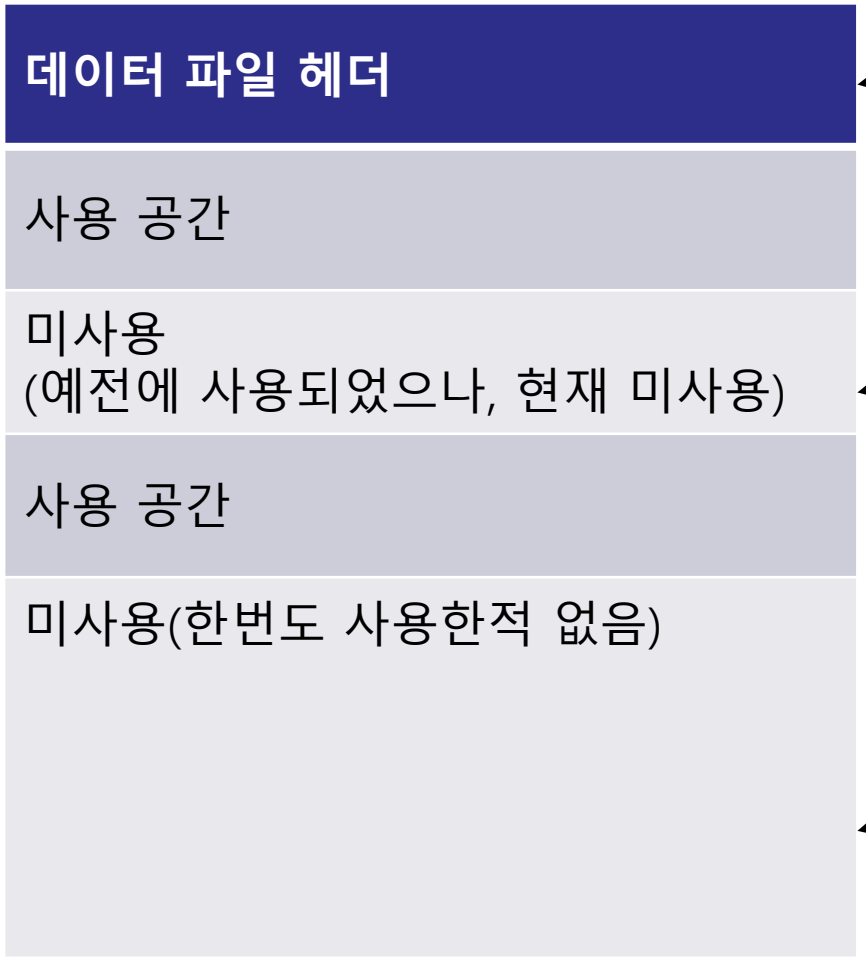
- 임시 테이블스페이스의 임시파일(Tempfile)
  - 세션의 또는 트랜잭션 지속 시간 동안 임시 테이블과 같은 스키마 객체의 데이터가 존재하며, 메모리에서의 해시 및 정렬 등의 작업 메모리 공간이 부족했을때의 저장소로 사용됨
  - 일반테이블의 영구적인 데이터는 임시파일에 저장되지 않음
  - 임시파일을 사용하는 스키마 오브젝트는 NOLOGGING 모드로 설정되며, 미디어 복구가 않됨
  - 임시 파일은 DBA_TEMP_FILES, V$TEMPFILE 에서 모니터링 할수 있고, 일반 데이터 파일을 조회하는 DBA_DATA_FILES, V$DATAFILE 뷰에서는 표시되지 않음

Tibero Database 구조 (물리 저장구조)

- 온라인 데이터 파일 및 오프라인 데이터 파일
  - 모든 데이터 파일은 온라인 (사용 가능) 또는 오프라인 (사용 불가능) 상태에 해당됨
  - MOUNT 모드에서 데이터파일을 오프라인 변경후, 테이블스페이스를 삭제할 수 있음
  - 테이블스페이스를 오프라인 변경하면 연결된 데이터파일 또한 오프라인 상태가 됨
  - 데이터파일의 이름/경로를 바꾸기 위해서는 해당 테이블 스페이스를 오프라인 상태로 설정후 변경가능

Tibero Database 구조 (물리 저장구조)

- 데이터 파일의 구조



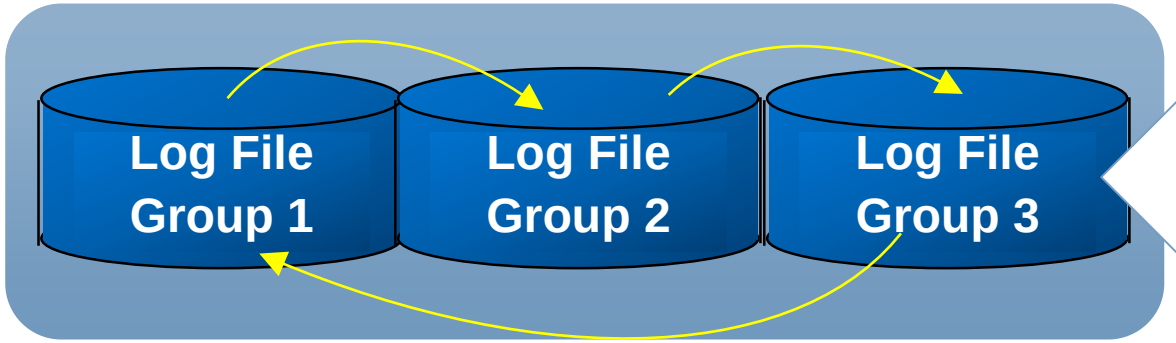
데이터 사이즈에 대한 정보, 체크 포인트 TSN, 데이터베이스에서 고유하게 식별하기 위한 파일 절대번호, 테이블스페이스에서 데이터파일을 식별하기 위한 파일 상대번호등의 메타 정보 포함

데이터의 갱신 또는 삭제가 반복되면 중간 사용하지 않는 공간이 존재하게 되며, 재사용하기에는 작은 크기일 경우, 이를 조각난 공간이라 함

데이터파일을 생성시 초기 포맷된 공간으로 테이블스페이스의 데이터 증가시 데이터파일의 여유공간을 사용하여 세그먼트에 익스텐트를 할당하게됨

Tibero Database 구조 (물리 저장구조)

- **Online Redo Log Files**



- 데이터 변경 내용
- TSN 과 타임 스탬프
- 트랜잭션 ID
- 트랜잭션이 커밋 될 때의 TSN과 타임 스탬프 (트랜잭션이 커밋 된 경우)
- 변경을 수행 한 작업의 유형
- 변경된 데이터 세그먼트의 이름 및 유형

➢ 복구를 위해 데이터베이스 변경사항을 기록하고, 리두 로그 그룹은 적어도 2개 이상으로 순환하며 사용

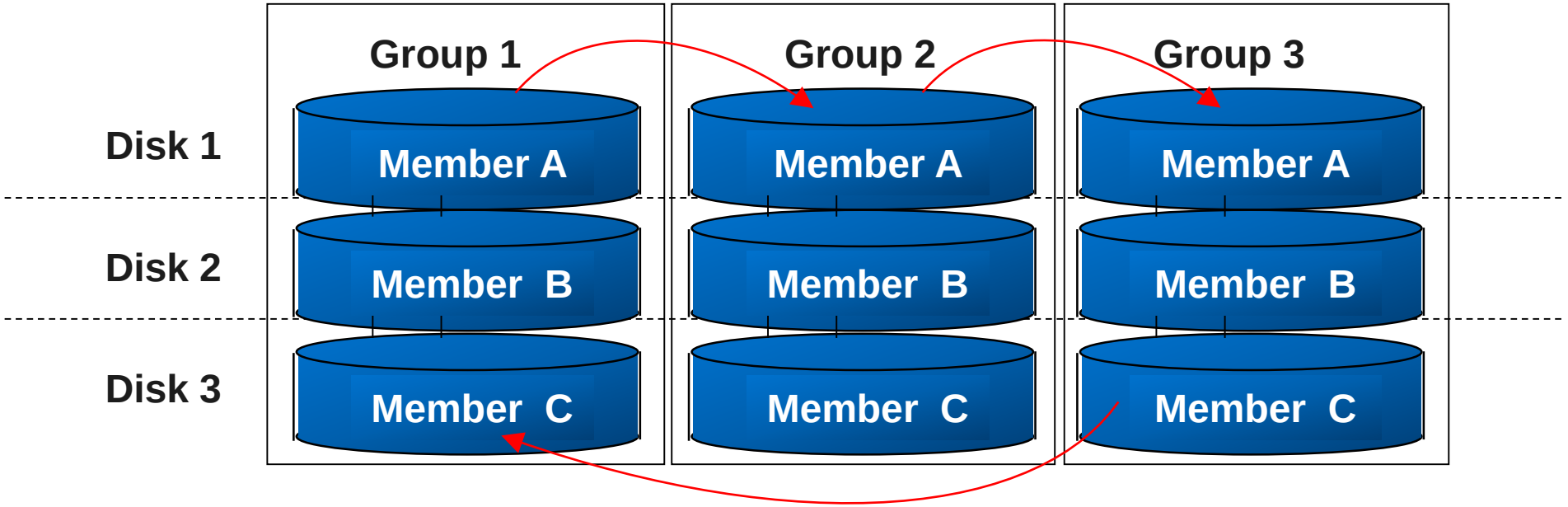
➢ 만약 Database운영 Mode가 ARCHIVELOG Mode일 경우 Log Switch가 일어날 때 log_archive_dest로 Copy되어져 향후 복구 시 사용됨

➢ (인스턴스 복구시) 온라인 리두로그의 내용을 이용하여 아직 데이터 파일에 기록되지 않은 커밋된 데이터를 복구함

➢ 리두로그 버퍼의 내용(커밋되지 않은 트랜잭션, 커밋된 트랜잭션, 언두 데이터, 스키마 객체 관리를 위해 사용한 쿼리 문 등) 이 기록됨.

Tibero Database 구조 (물리 저장구조)

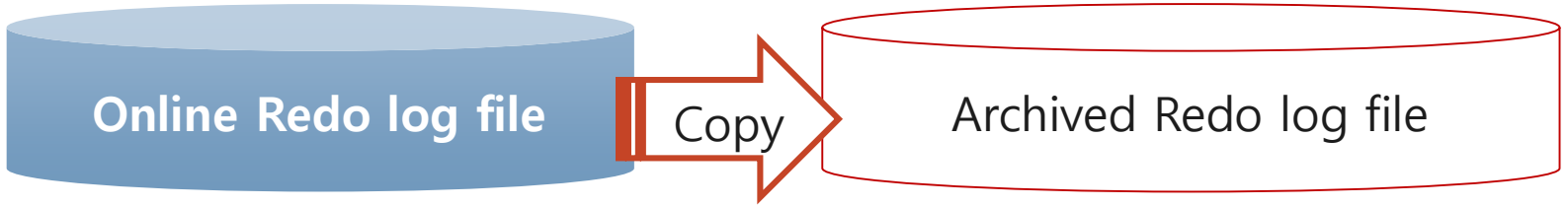
- **Multiplexed Redo Log Files**



- Redo log File을 구성하는데 1 Group에 2개 이상의 Member를 동일 Machine상에서 다른 Disk에 분리하여 구성하기 권장.
- Multiplexed Redo Log Files는 특정 그룹에서 하나의 파일을 손실 시에 특이 사항 없음.
- 그룹의 모든 Member들은 동일한 정보와 Size 를 갖음

Tibero Database 구조 (물리 저장구조)

Archived Redo Log Files



- 아카이브 리두 로그 파일은 온라인 리두로그 그룹 맴버의 복사본임
- 데이터베이스 파일에 속하지는 않지만 복구를 위해 사용되는 중요한 파일임
- 아카이브 리두 로그의 용도
  - 데이터베이스 백업 복구
  - TSC(Tibero Standby Cluster) 의 Standby DB 의 데이터 업데이트
  - 데이터복제 솔루션인 ProSync 에서 사용

Tibero Database 구조 (물리 저장구조)

- **Control Files**

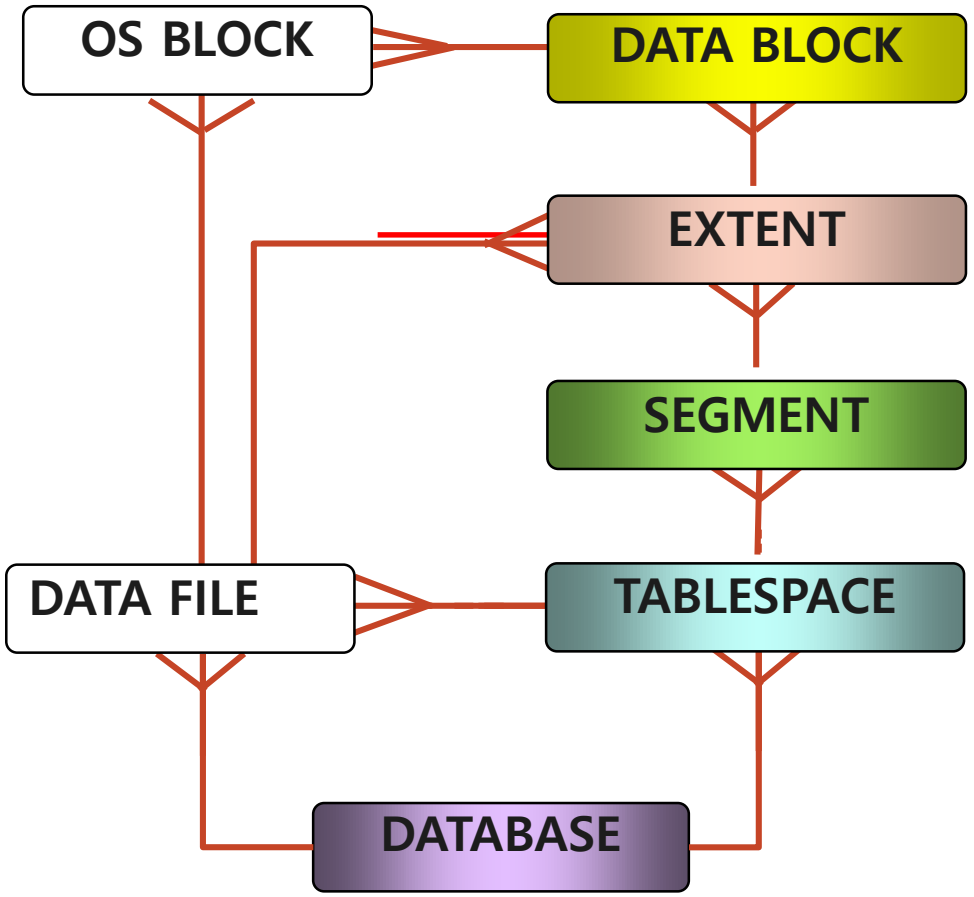
Database

Controlfile
- 데이터베이스 이름과 데이터베이스의 고유 식별자 (DBID)
- 데이터베이스 생성시각
- 데이터파일, 온라인 REDO로그, 아카이브 REDO로그 정보
- 테이블스페이스 정보
- RMGR 백업 정보

- Control File은 작은 Binary File로 Database의 구조.
- 모든 Database Files & log files들은 Control file에서 식별 가능.
- Database name은 Control file에 저장.
- Control File은 Mount, Open시에 필요.
- Recovery시 필요한 동기화된 정보를 저장.
- 최소 다른 Disk상에 2개 이상의 Control Files를 가지도록 권장.
- Control File은 Database가 Open되어 졌을 때 반드시 Tibero Server가 Writing 가능하도록 설정.

Tibero Database 구조 (논리 저장구조)

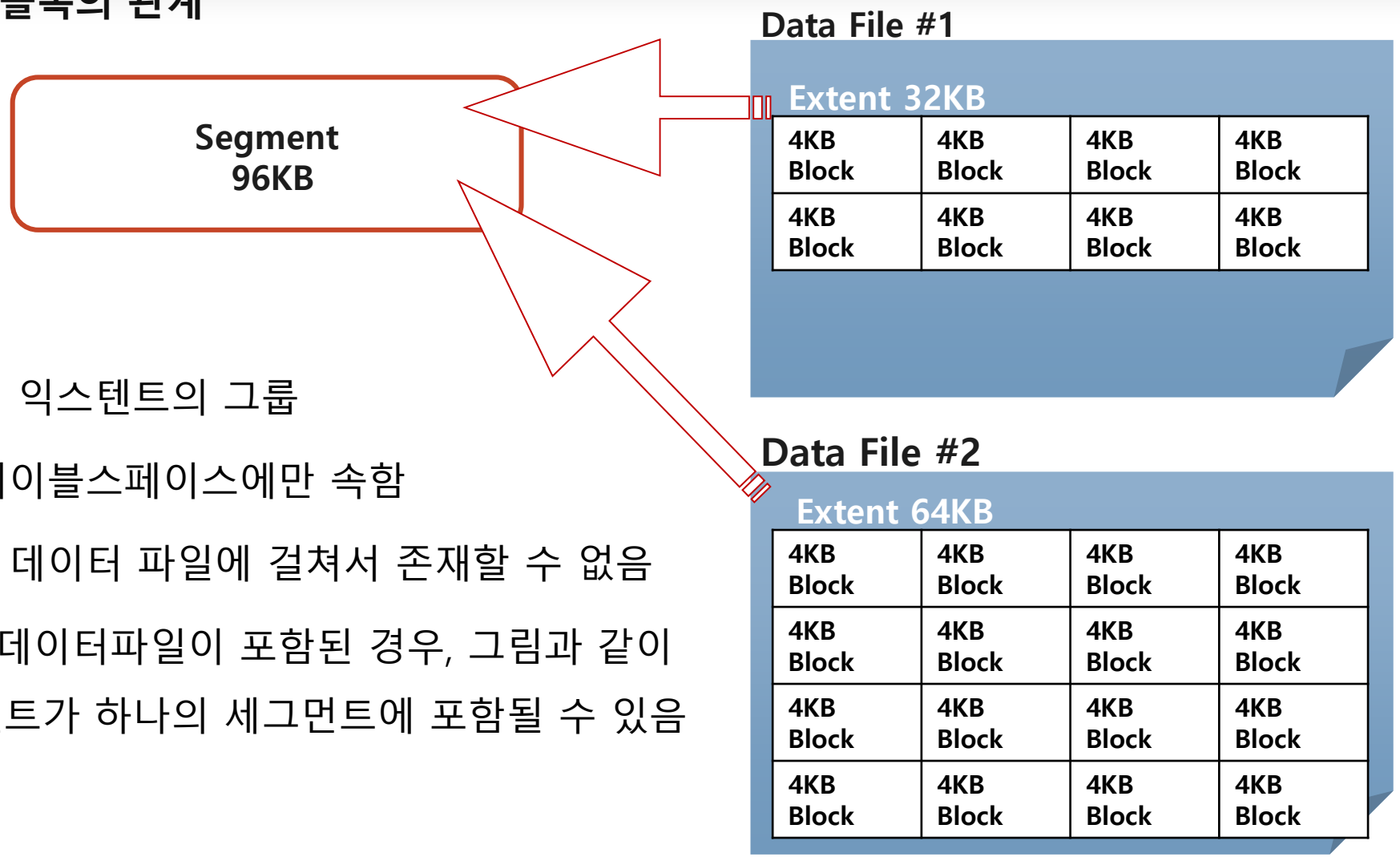
▪물리 저장 구조 ▪논리 저장 구조



- **Data Block**
  - 데이터 파일에 할당되는 물리적 파일 블럭
  - Data가 저장되는 최소의 구조단위.
  - 2k, 4k, 8k, 16k, 32k
- **Extent**
  - 연속된 데이터베이스 블럭의 집합
- **Segment**
  - tablespace내 특정 구조에 대한 모든 데이터를 갖고 있는 하나 혹은 하나 이상의 익스텐트의 집합.
  - table, index segment
- **Tablespace**
  - 물리적으로 그룹화된 데이터를 위한 논리적 저장 단위.
- **Database**
  - 테이블스페이스가 저장되어서 공유되는 논리적 집합체.

Tibero Database 구조 (논리 저장구조)

▪세그먼트와 익스텐트, 데이터 블록의 관계

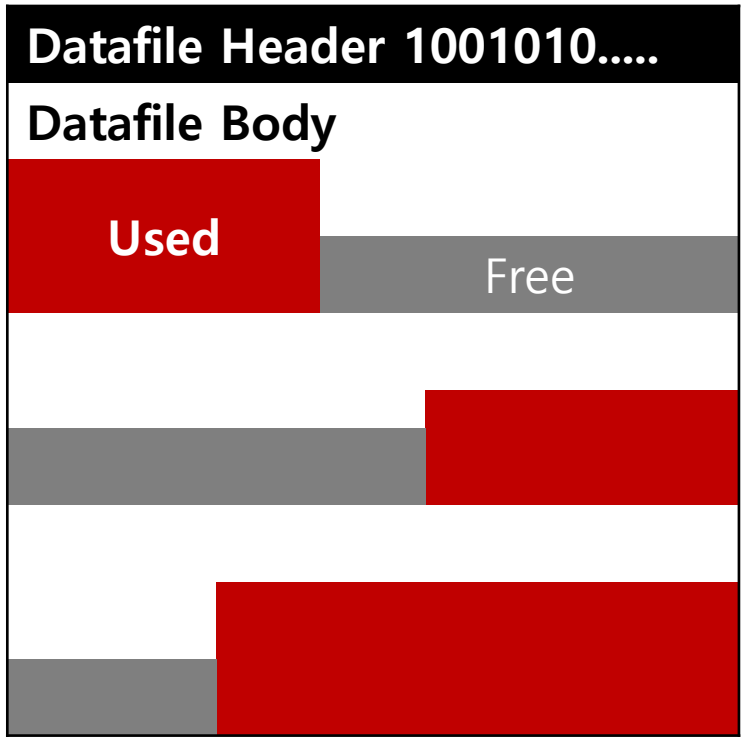


- **Segment & Extent**
  - 테이블, 인덱스에 할당된 익스텐트의 그룹
  - 각 세그먼트는 하나의 테이블스페이스에만 속함
  - 하나의 익스텐트는 여러 데이터 파일에 걸쳐서 존재할 수 없음
  - 테이블스페이스에 여러 데이터파일이 포함된 경우, 그림과 같이 여러 데이터파일의 익스텐트가 하나의 세그먼트에 포함될 수 있음

Tibero Database 구조 (논리 저장구조)

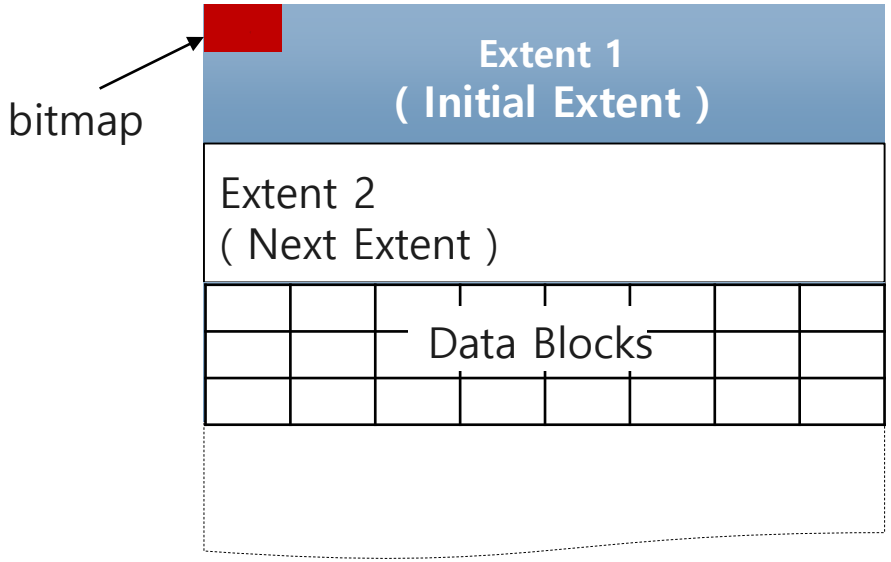
- 테이블스페이스 공간 관리

- 티베로는 로컬 관리 방식으로 데이터 파일 헤더에 데이터파일의 공간을 관리하기 위한 비트맵을 생성하여 사용함



- 세그먼트 공간 관리

- 자동 세그먼트 공간방식(ASSM: Automatic Segment Space Management)으로 초기할당 익스텐트의 비트맵을 사용하여 관리함.
- 공간관리에 사용하는 유일한 파라미터는 PCTFREE 로써 블록의 빈 공간의 비율을 설정함



Tibero Database 구조 (논리 저장구조)

▪세그먼트 생성

- 테이블, 인덱스 객체 생성시 세그먼트가 할당됩니다.
- 파티션 되어 있는 객체는 파티션마다 세그먼트가 할당됩니다.

```
CREATE TABLE MY_SEGEMENT(C1 NUMBER);
```

스키마 오브젝트	세그먼트	세그먼트 타입
MY_SEGEMENT	MY_SEGEMENT	TABLE

```
CREATE TABLE MY_SEGEMENT(C1 NUMBER PRIMARY KEY
                        ,C2 CLOB                );
```

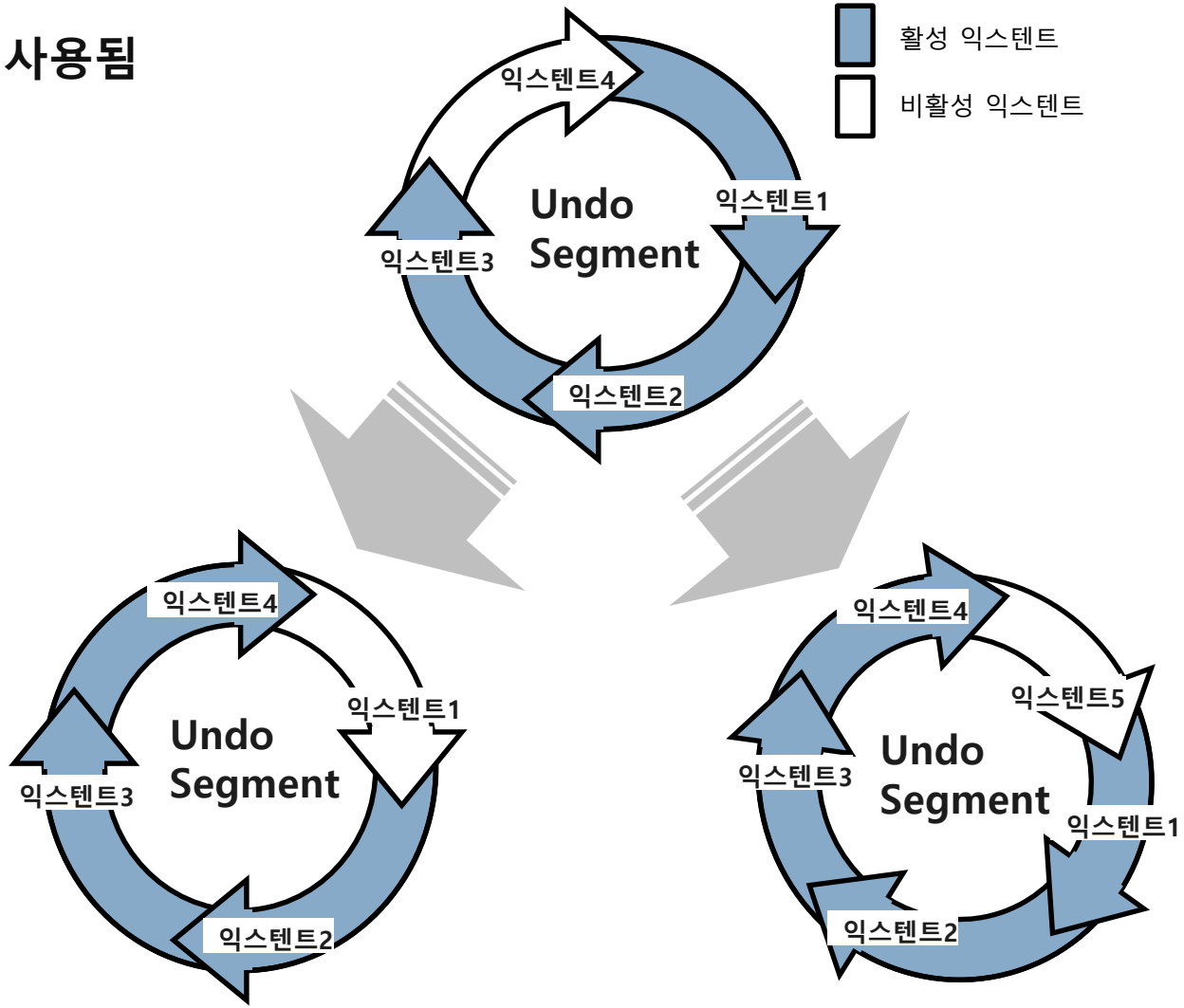
스키마 오브젝트	세그먼트	세그먼트 타입
MY_SEGEMENT	MY_SEGEMENT	TABLE
TIBERO_CON325007	TIBERO_CON32500741	INDEX
TIBERO_LOB272700	TIBERO_LOB272700	LOB
TIBERO_LIDX272800	TIBERO_LIDX272800	INDEX

Tibero Database 구조 (논리 저장구조)

▪UNDO 세그먼트

- 언두 데이터는 트랜잭션 기록을 보관하며, 다음의 경우 사용됨
  - ➢ 트랜잭션 롤백
  - ➢ 트랜잭션 복구
  - ➢ 읽기 일관성 작업
  - ➢ 플래시백 작업

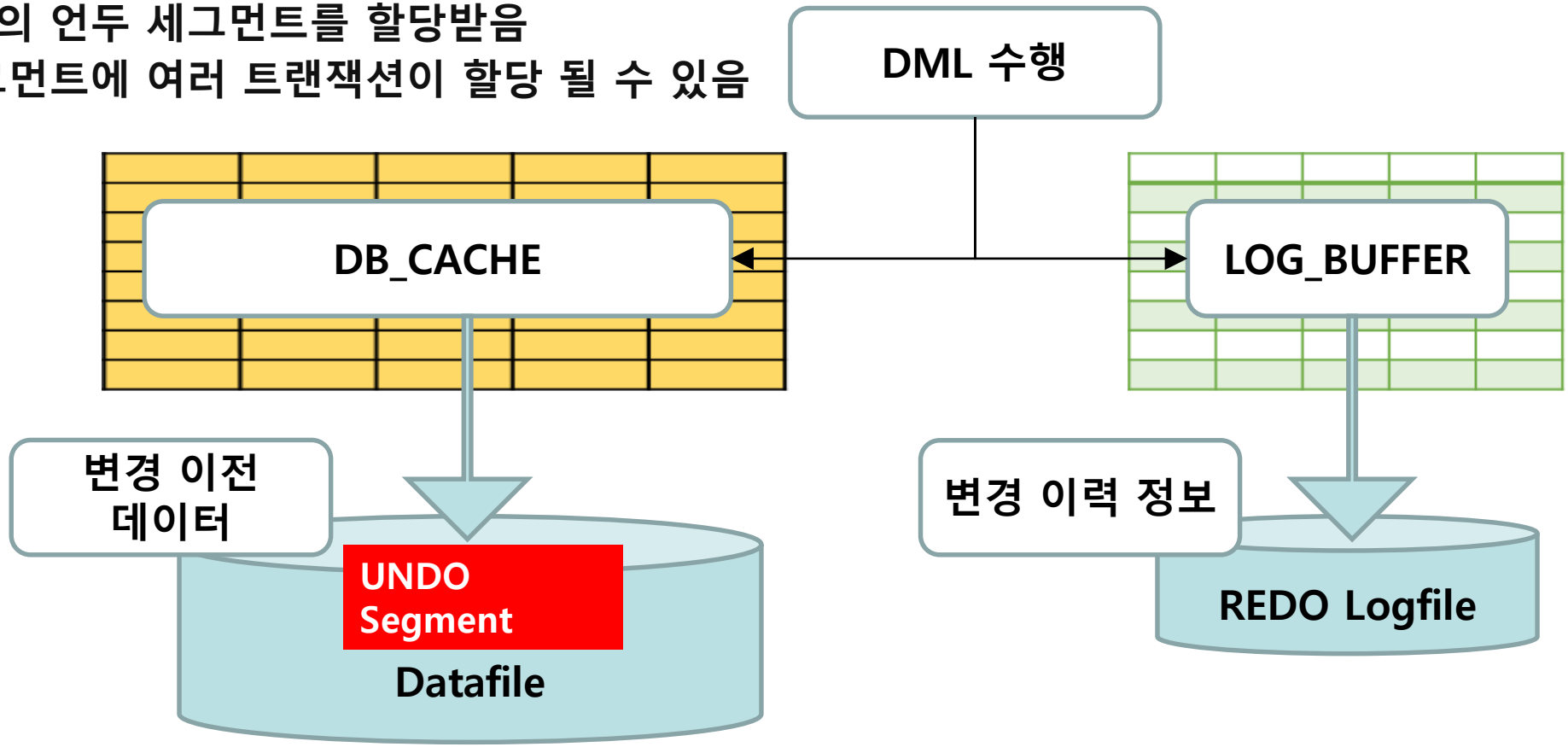
- 언두 데이터는 언두 테이블스페이스의 블록에 저장됨
- 언두 블록의 변경 발생시 REDO 로그를 생성함
- 트랜잭션 시작시 UNDO 세그먼트가 할당됨
- 언두 익스텐트는 순환하면서 재사용 및 추가 할당 됨



Tibero Database 구조 (논리 저장구조)

▪UNDO 세그먼트

- 트랜잭션은 하나의 언두 세그먼트를 할당받음
- 한개의 언두 세그먼트에 여러 트랜잭션이 할당 될 수 있음



Tibero Database 구조 (논리 저장구조)

UNDO 세그먼트

UNDO 자동 관리

- 언두 테이블스페이스에서 언두 데이터 저정하며, 공간을 자동으로 관리함
- 공간 필요시 자동으로 언두 세그먼트에 익스텐트 추가 할당됨
- 읽기 일관성 보장을 위해 UNDO_RETENTION 시간 동안 언두데이터 유지

UNDO_RETENTION

- COMMIT 언두 데이터의 보존하는 기간을 설정하는 파라미터(단위: 초)
- 기본값 900초(15분) 으로, 해당 언두 익스텐트의 데이터 보장
- UNDO_RETENTION 기간이 지난 익스텐트는 진행 중인 트랜잭션을 위해 재할당 될 수 있음
  RETENTION 기간 이내인 경우 익스텐트는 재활용 될 수 없음

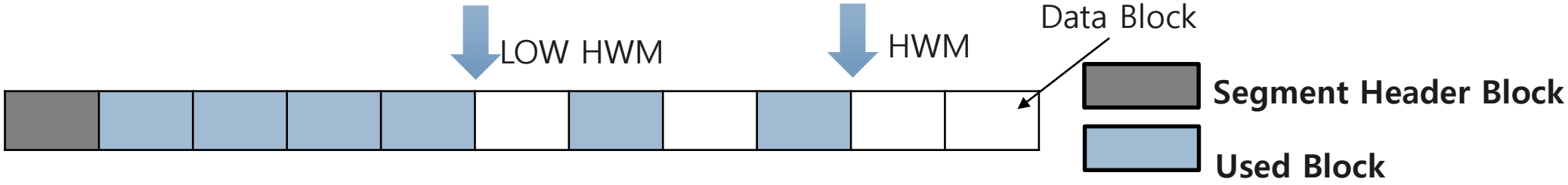
UNDO EXTENT 의 상태

- Active, Unexpired, Expired

Tibero Database 구조 (논리 저장구조)

▪세그먼트 공간관리 와 HWM(High Water Mark)

- 자동 세그먼트 공간 관리 방식으로써 BITMAP 을 이용하여 공간을 관리함
- PCTFREE 파라미터를 이용하여 블록 속성 설정
  (다른 파라미터는 허용은 되나 무시됨)
- 데이터 입력시 블록그룹을 할당하며 세그먼트 헤더블록의 비트맵에 블록 정보를 기록합니다.

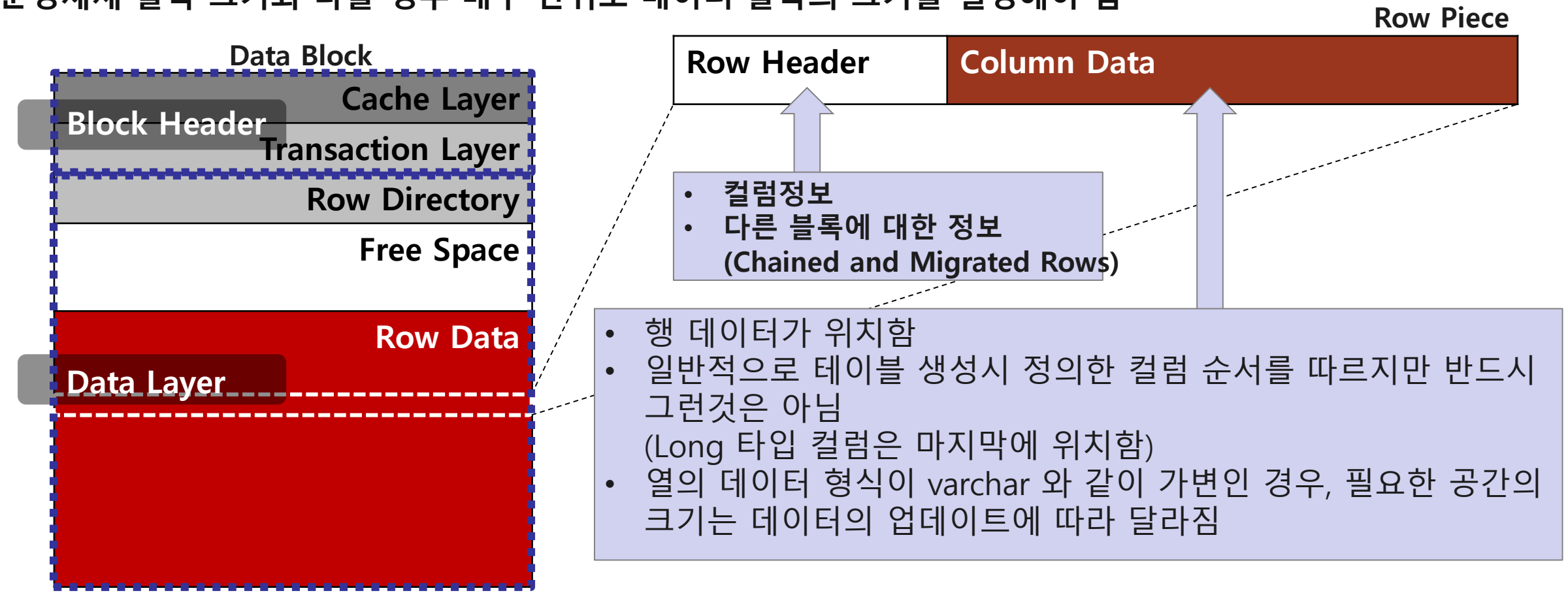


- 데이터 입력시 LOW HWM 하위의 여유공간이 있는 블록에 쓰거나, LOW HWM 과 HWM 사이의 블록을 사용함
- FULL SCAN 시에 LOW HWM 하위의 모든 블록과 LOW HWM 과 HWM 사이의 경우는 세그먼트 헤더블록의 BIT MAP 정보를 이용하여 액세스 함

Tibero Database 구조 (논리 저장구조)

▪데이터 블록

- 데이터베이스 I/O 의 단위이자 논리저장구조의 최소 단위가 데이터 블록임
- 운영체제 블록 크기와 다를 경우 배수 단위로 데이터 블록의 크기를 설정해야 함



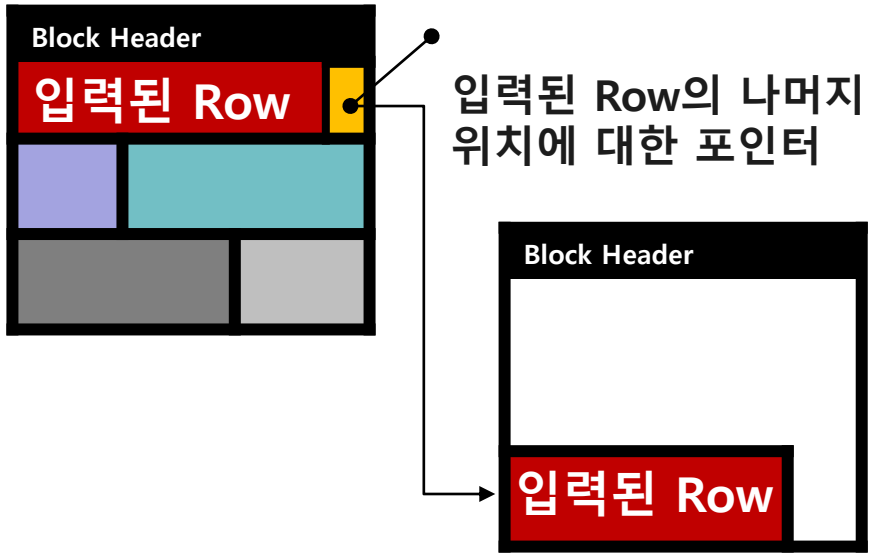
TmaxTibero

Tibero Database 구조 (논리 저장구조)

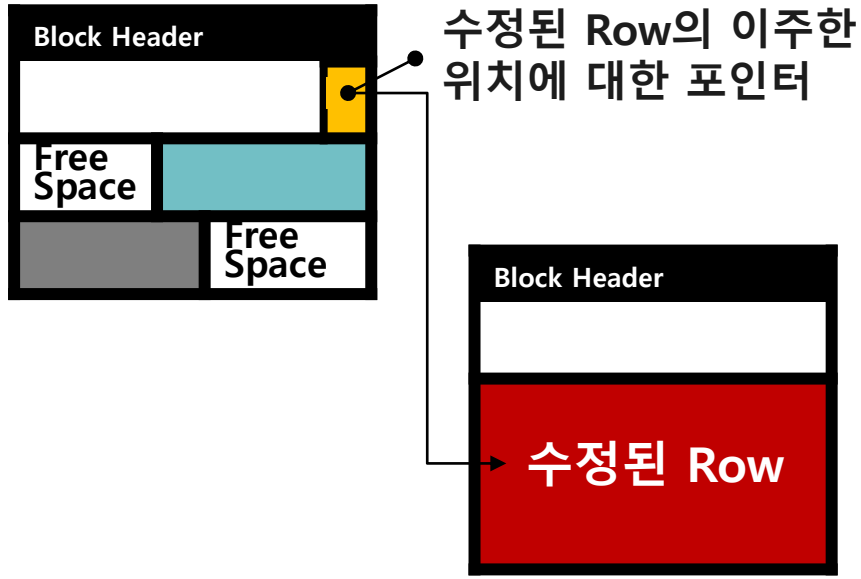
▪행 연쇄(Row Chaining) & 행 이주(Row Migration)

- INSERT 수행시 공간 부족으로 해당 블록에 모두 저장되지 않는 경우, 일부는 해당 블록에 저장되고, 다른 새로운 블록에 나머지 데이터가 이어서 저장됨
- UPDATE 수행시 큰 값으로 수정되면서 저장할 공간이 없는 경우, 값이 다른 새로운 블록으로 이동하여 저장됨
- 한개의 행 조각(row piece) 에 255 개 컬럼이 저장되므로, 컬럼 갯수가 그 이상일때 나머지 컬럼들은 다른 행 조각에 이어서 저장됨

행 연쇄(Data Block Chaining)



행 이주(Data Block Migration)

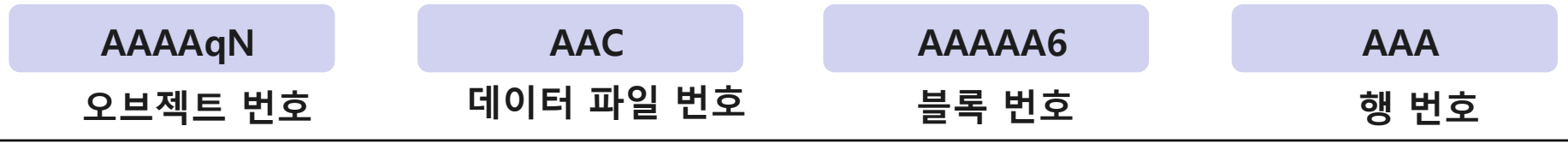


Tibero Database 구조 (논리 저장구조)

ROWID

- 행을 식별하기 위한 고유의 식별자
- 각 행의 물리적 주소를 BASE64 인코딩으로 표현한 값으로 ROWID 값 자체가 테이블에 저장되어 있는것은 아님
- 저장구조 정보(세그먼트에 대한 오브젝트 번호, 데이터파일 번호, 데이터 블록 번호, 행 번호) 를 조합하여 사용함

ROWID 형식

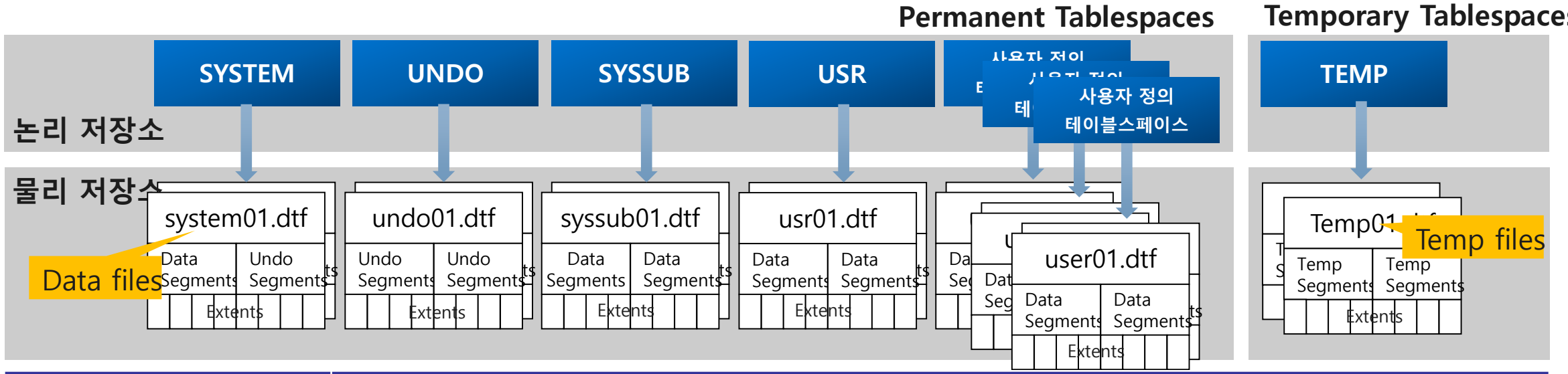


- 오브젝트 번호 : 세그먼트를 식별하는 데이터 오브젝트 번호
- 데이터 파일 번호 : 행을 포함하는 테이블스페이스의 데이터 파일의 파일 아이디
- 블록 번호 : 데이터파일 기준으로 블록의 위치를 식별하기 위한 값
  (* 따라서 Tablespace에 데이터파일이 여러 개일때 블록번호가 동일한 블록이 존재할 수 있음)
- 행 번호 : 블록내에서 행을 식별하기 위한 번호

Tibero Database 구조 (논리 저장구조)

▪테이블스페이스

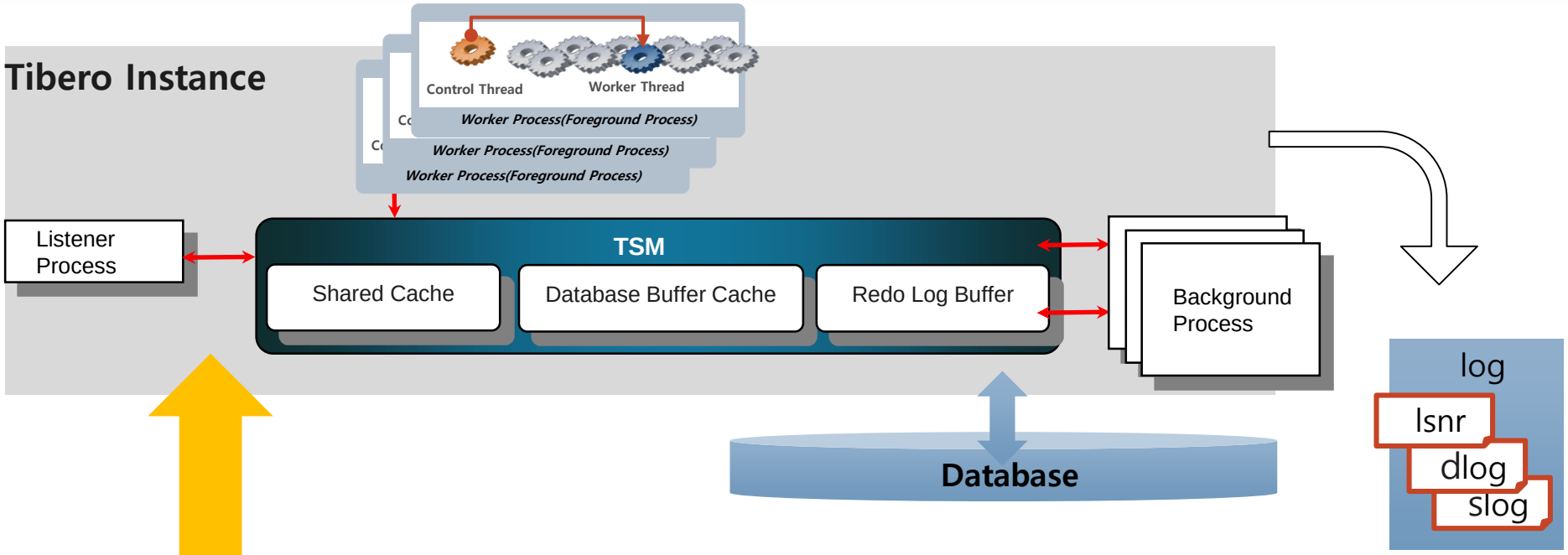
- 세그먼트를 저장하는 논리 저장소, 한개 이상의 데이터 파일이나 임시파일을 사용하여 데이터를 저장함



Tablespace	포함하고 있는 데이터
SYSTEM	- 데이터베이스 관리 정보를 포함한 테이블, 뷰(데이터 사전) - 프로시저, 패키지, 트리거 등 컴파일된 오브젝트
UNDO	언두 데이터(언두 세그먼트)
SYSSUB	TPR 의 스냅샷 데이터
USR	TIBERO, TIBERO1 유저의 default tablespace
TEMP	정렬 작업용 데이터, 세션의 유지 기간만 저장되는 임시 데이터

Tibero Database 구조 (기타)

- **The Parameter File**



- Instance 시작할때 Parameter file($TB_SID.tip)을 Read.
- 기본으로 TSM에 있는 Memory 구성을 얼마로 할 것인지를 Setting 필요.
- Control file의 위치와 이름

Tibero Database 구조 (기타)

- slog (TRACE Log)

  - Tibero Instance가 실행 중에 Error발생시 messages은 Trace or Dbms Log file에 Write.
  - 서버 성능이 저하되는 원인을 찾거나 티베로 자체 버그를 해결하는 데 사용될 수 있음
  - Log File은 $TB_HOME/instance/$TB_SID/log/slog (<--trace) Directory에 Write.
  - 다음과 같은 Error발생시 Write.
    - All internal errors
    - Block corruption errors
    - Deadlock errors
    - Etc

Tibero Database 구조 (기타)

- dlog (DBMS Log) File
  - Log File은 $TB_HOME/instance/$TB_SID/log/dlog (<--dbms) Directory에 Write.
  - 서버 기동 및 종류, DDL 문장의 수행 등이 기록되는 파일.
  - 다음과 같은 Error발생시 Write.
    - DDL, Server Manager statements (STARTUP, SHUTDOWN,ARCHIVE LOG, and RECOVER)

- Listener 로그 파일(lsnr)
  - Listener의 디버깅을 위한 파일이다. 리스너에서 일어난 중요한 일이 기록되는 파일이며, 리스너의 버그를 해결하는 데 사용될 수 있다.

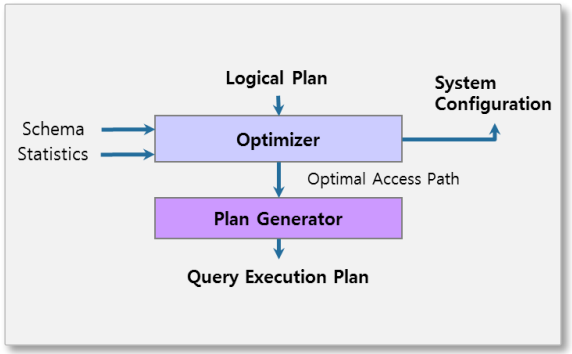
- Internal 로그 파일(ilog)
  - 스레드별로 설정된 이벤트에 대한 시스템 로그가 기록되는 파일이며, Internal 로그를 보려면 tbiv를 이용해야 한다.

3. 티베로 기능

대용량 데이터의 고성능 처리

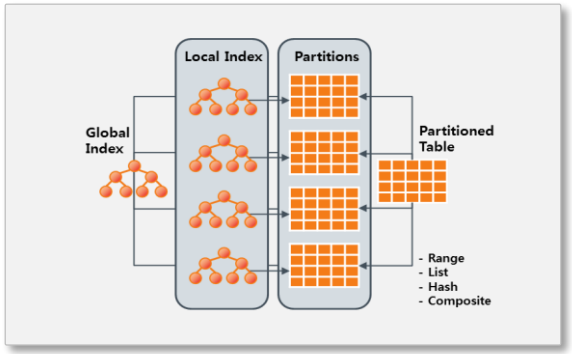
Tibero는 대용량 데이터 처리를 위한 아키텍처를 기반으로 고성능 처리를 지원합니다.

비용 기반 Optimizer



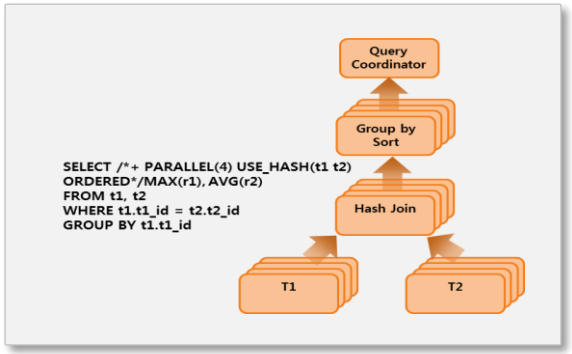
- 다양한 실행계획 중 최소 비용(Disk I/O, CPU 사용)의 실행 계획을 선택하여 빠른 SQL 실행 보장
- 실행 계획과 실행 통계를 조회하여 튜닝할 수 있는 다양한 Tool 제공
- 업무에서 많이 사용되는 약 40개의 Hint가 제공되어 실행 계획을 제어할 수 있음

다양한 Index/Partition 기술



- 다양한 Index
  - B*Tree, Reverse Key, Function-Based Index, IOT(Index-Organized Table)
- Table 및 Index 파티셔닝
  - Range, List, Hash 및 Composite Partitioning
  - 파티션에 따른 Local Index, Global Index

고성능 병렬 처리 및 압축

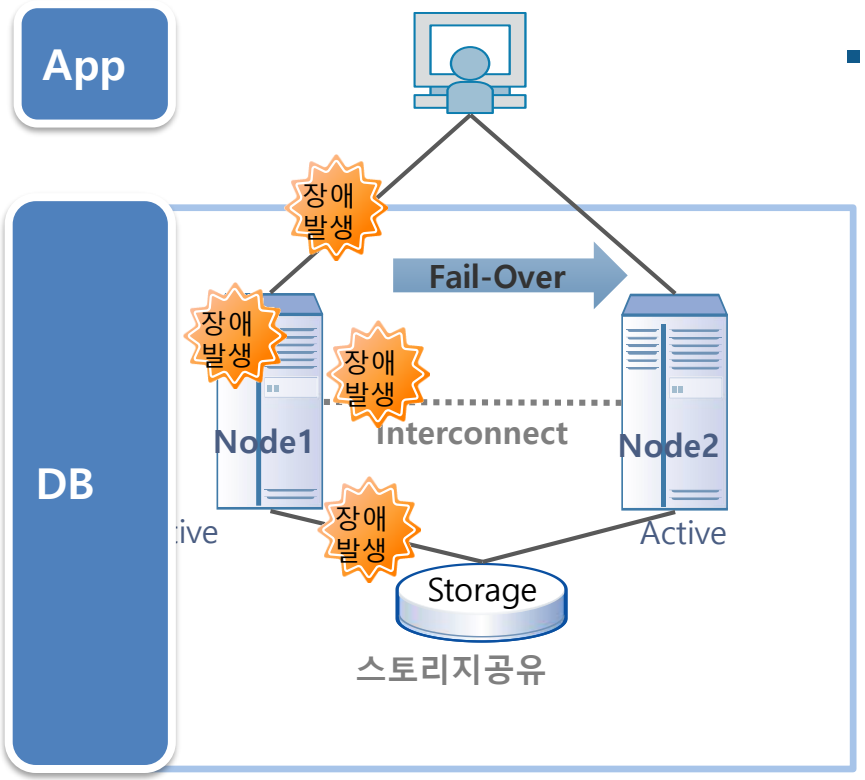


- 병렬 쿼리를 통한 성능 향상
  - 다수의 Thread를 이용하여 시스템 리소스 활용 극대화
  - 대용량 테이블 또는 파티션 인덱스 스캔, Join 쿼리
  - 대용량 테이블 인덱스 생성
  - Bulk Insert/Update/Delete
  - Aggregation 함수
- 대용량 데이터의 압축 기능
  - Block 단위 압축
  - Table 전체 압축

TAC(Tibero Active Cluster)

Tibero의 TAC 기술은 안정성과 고가용성을 위한 핵심으로서 공유디스크 기반의 Active Clustering을 통하여 다양한 유형의 장애에도 시스템 중단 없이 안정적인 서비스를 지원

공유디스크 기반 Tibero Active Cluster



- **공유디스크 기반 Active Cluster**
  - 특정 노드 장애 시 정상 노드로 자동 Fail-over
  - TBCM을 통한 HeartBeat 체크로 장애감시
  - Global Cache를 통한 인스턴스간의 버퍼캐쉬 공유
  - 데이터 정합성의 보장

장애 발생 유형

- DB엔진 장애
- 시스템 장애
- Public Network 장애
- Interconnect 장애

장애 발생 대응

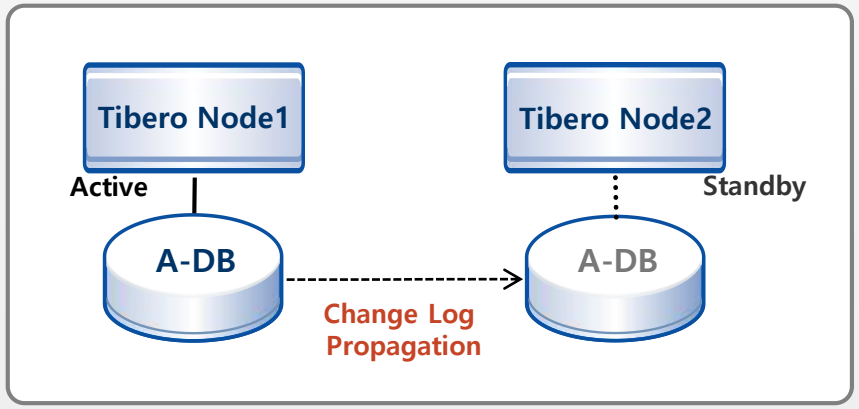
- *정상노드로 Fail-Over*

TmaxTibero

TSC(Tibero Standby Cluster)

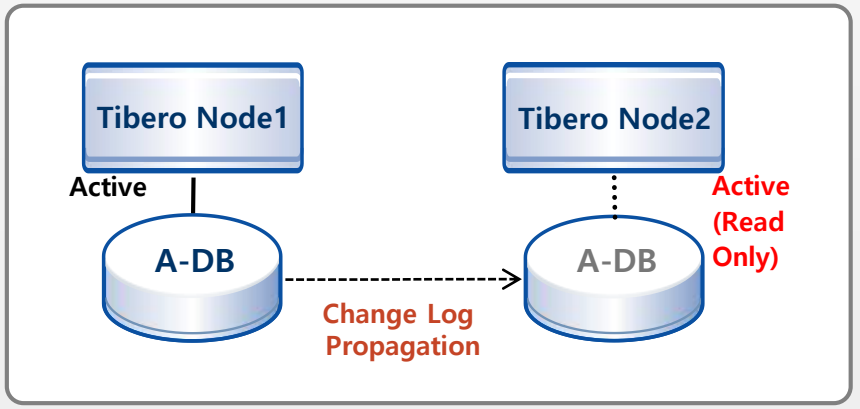
Tibero TAC 뿐만 아니라 독립 디스크 방식(Shared Nothing)의 TSC(Tibero Standby Cluster) 이중화 구성을 지원하여 운영상 고가용성과 재해복구(DR)에 적합한 아키텍처를 제공

Active-Standby 방식



- 독립디스크 기반 Active-Standby 방식으로 자료 보호, 재해 복구(Disaster Recovery) 등을 목적으로 함
- 원본 데이터베이스 또는 디스크 Fail 시 신속히 서비스를 재개하고 자료 보호
- 장애 복구 후 변경 데이터에 대한 데이터 동기화 후 기존 Active 서버로 정상적인 서비스 재개 가능

Active-Active(Read Only) 방식



- 장애 시 서비스의 중단을 최소화 복구 가능
- 복제 대상의 Active 서버(Node 2)는 Read Only로 운영되어 Select 및 테스트 용도로 사용하여 자원 효율성을 높임

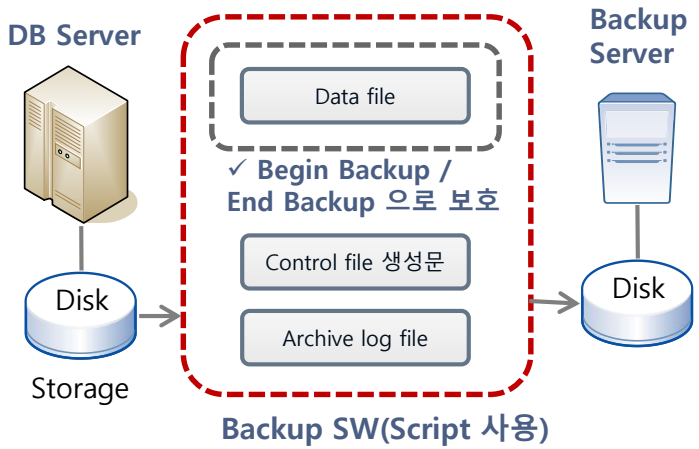
TmaxTibero

백업 & 복구

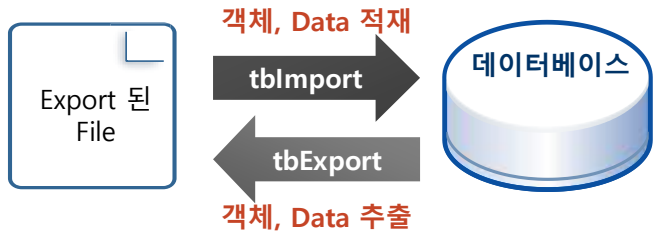
장애로부터 데이터를 안전하게 보호하기 위하여, 다양한 백업/복구 방법을 지원

백업 및 복구

- DB가 운영 중인 상태에서 주기적으로 데이터 파일 및 로그 파일을 백업 서버로 전송
  (1) 완전 복구 : DB를 가장 최신 상태로 복구
  (2) 불완전 복구
  - 특정 시간 기준
  - 특정 TSN 기준
  - 특정 Archive log 파일 기준



- Offline 백업 & 복구
- Crash Recovery
- Online Media Recovery
- Export/Import 이용한 논리적 백업/복구



- Flashback Query 지원
  - 과거 특정 시점 데이터 조회
- 복구 관리자(RMGR) 제공
  - Online Full Backup
  - Incremental Backup
  - Automatic Recovery

보안 및 암호화

Tibero는 DB 자체 보안 기능 뿐 아니라, 암호화 솔루션과의 연동 및 빠른 암호화를 지원

Tibero 내장 보안 기술

- **TDE 컬럼 암호화**
- **TDE 테이블스페이스 암호화**
- TDE 암호화 키 외부 관리
- AP에서 사용 가능한 암복호화 함수
  - DBMS_CRYPTO 패키지
- 암호화 솔루션 연동
- Profile 기능
- Audit 기능
- 특권 및 Role
- 네트워크 접근 제어
- Masking

DBMS 엔진 레벨 암호화(TDE)

기능	컬럼 암호화	테이블 스페이스 암호화
암호화 대상	• 특정 컬럼 암호화	• 전체 테이블 암호화
암호화 수준	• 데이터 파일 및 Log 암호화 • DB Buffer Cache 암호화	• 데이터 파일 및 Log 암호화 • Buffer Cache 암호화 안됨
지원 알고리즘	• DES, 3DES168, AES128, AES192, AES256, SEED	
제약 사항	• 인덱스, SALT 동시 사용 불가 • 인덱스 사용 능력 감소 • 함수기반 인덱스 불가 • Foreign Key 불가 • CLOB, BLOB 등 불가	• 컬럼 암호화에 존재했던 제약들이 테이블스페이스암호화에서 가능함 • 반면에 테이블 스페이스의 모든 데이터 블록에 암호화와 복호화가 발생하기 때문에 테이블 스페이스의 크기가 크거나 수정과 접근이 잦을수록 성능 저하가 높아질 수 있음

개발 관리 도구

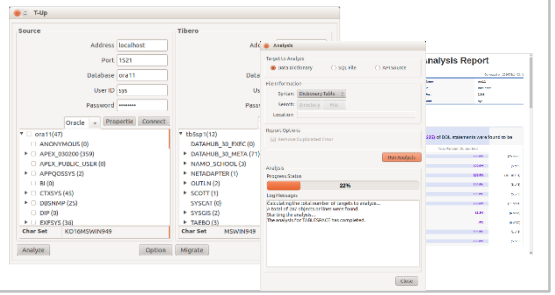
Tibero RDBMS를 이용하는 개발자와 관리자에게 필요한 Utility 를 지원

tbAdmin / tbStudio

SQL 문장의 입력, 편집, 실행
각종 모니터링 및 관리자 기능

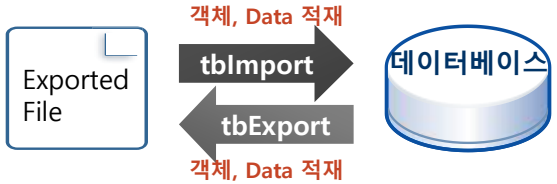
T-Up

마이그레이션 및 호환성
평가기능을 제공하는 Tool



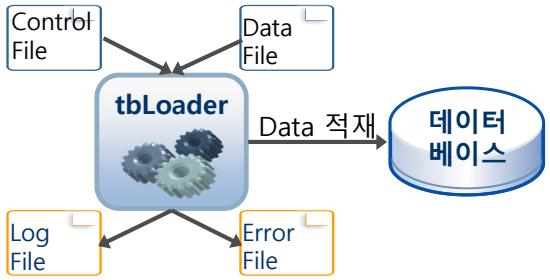
tbExport/tbImport

DB에 저장된 Schema 객체 및
데이터를 추출/적재 하는 도구



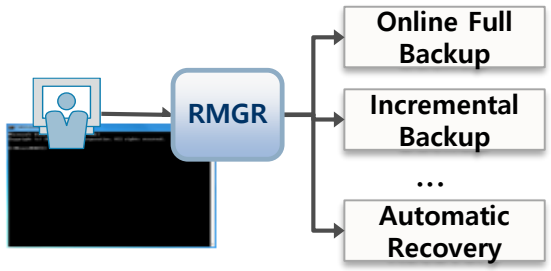
tbLoader

대용량 데이터 파일을
데이터베이스로 고속 적재



tbrmgr

데이터베이스의 온라인 백업 및
복구를 손쉽게 수행



기타 Utility

- **tbSQL**
  - ✓ 사용자가 직접 SQL 질의를 하고 그 결과를 확인하는 Client 유틸리티
- **tbpc**
  - ✓ tbESQL/C의 프리컴파일러
- **tbdv**
  - ✓ 데이터파일의 기본적인 정합성 검사 (헤더블록, 남은 공간 등)

TmaxTibero

TmaxTibero

감사합니다