ALTIBASE ADMINISTRATION I

ALTIBASE ADMINISTRATION I

CONTENTS

- INTRODUCTION TO ALTIBASE
- ARCHITECTURES
- REPLICATION

ALTIBASE ADMINISTRATION I

INTRODUCTION to ALTIBASE

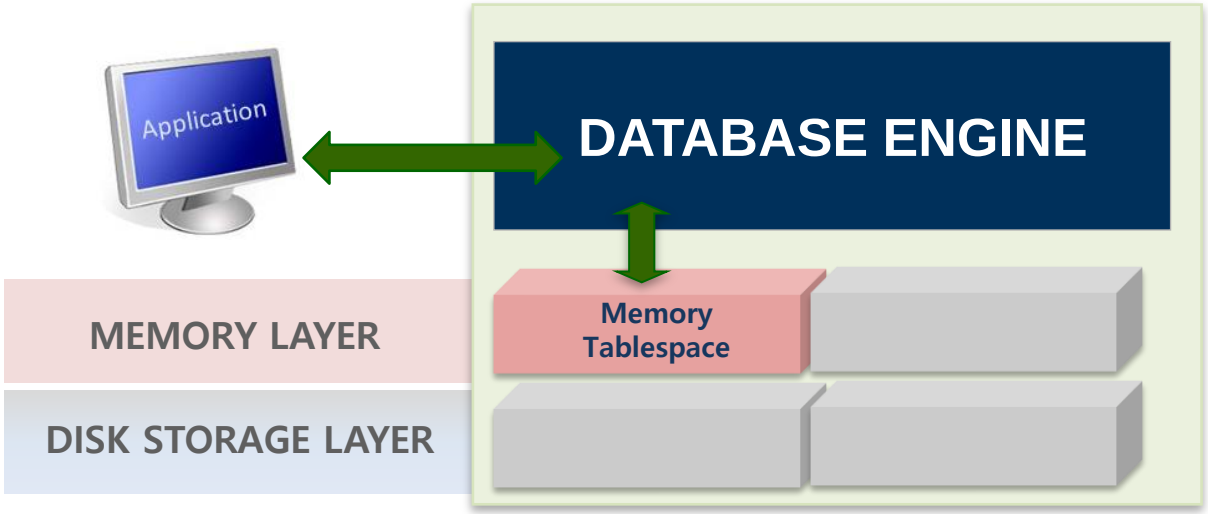
CONTENTS

INTRODUCTION TO ALTIBASE

ALTIBASE CONCEPT

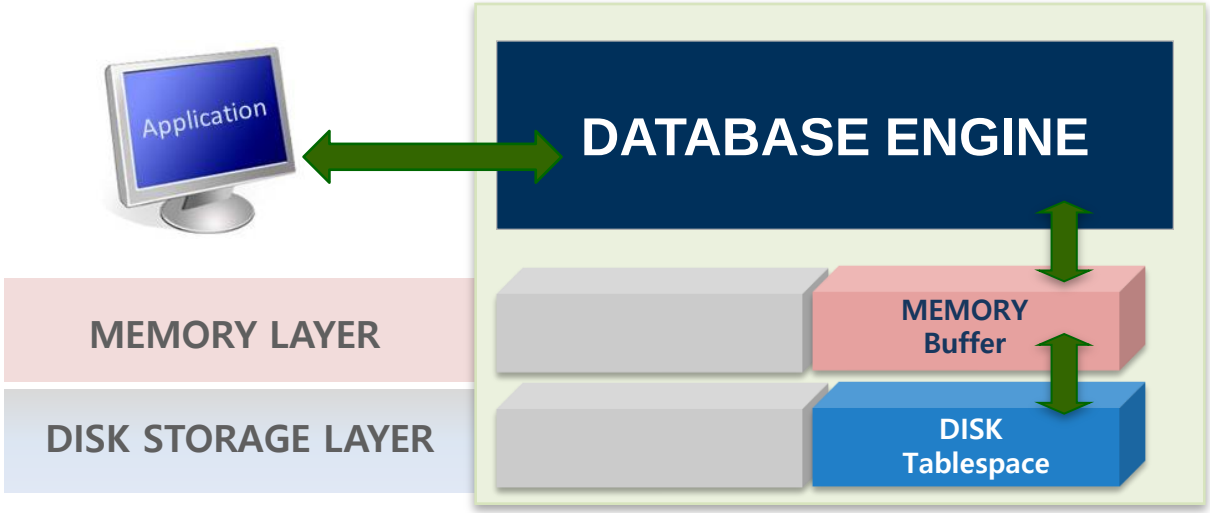
❖ 메모리 DBMS의 특징

- 빠른 데이터 처리 성능
  - 데이터와 인덱스를 모두 메모리에 저장하여 처리하므로 빠른 데이터 처리 가능
  - 메모리 인덱스는 RowID가 아닌 Physical한 포인터로 관리하기 때문에 빠른 접근이 가능
  - 디스크 I/O로 인한 성능 저하가 거의 발생하지 않음
  - OLTP성 업무에 사용하기 적합
- 물리적 메모리의 크기만큼만 데이터를 적재할 수 있는 제약사항 존재



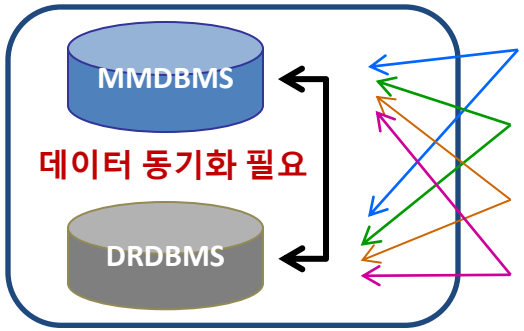
❖ 디스크 DBMS의 특징

- 대용량 데이터 처리
  - 메모리 DBMS에 비해 데이터의 저장 공간에 대한 제약이 거의 없음
  - History성 데이터, DW 용으로 사용 적합
- 처리 성능 제한
  - 디스크I/O로 인한 성능 저하 발생
  - Buffer에 데이터 상주시켜도 인덱스는 디스크에 생성



❖ HYBRID DBMS를 통한 효율성 증대

일반적 MMDBMS의 적용(혼용 구조)



DBA

APPLICATION

Resource

유지보수

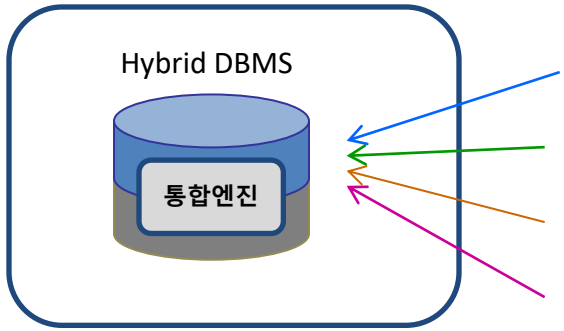
운영, 개발 및 구성적인 모든 비용과 노력들이 각각의 DBMS별로 발생(비용 및 시간 증가)

관리적 비용 : DRDBMS + MMDBMS 비용 추가

운영적 비용 : DRDBMS + MMDBMS 비용 추가

구성적 비용 : DRDBMS + MMDBMS 비용 추가

Hybrid DBMS의 적용



DBA

APPLICATION

Resource

유지보수

메모리와 디스크 저장장치를 하나의 엔진으로 제어함으로 효율성 극대화

관리적 비용 : Hybrid DBMS만을 관리 [비용 ½ 절감]

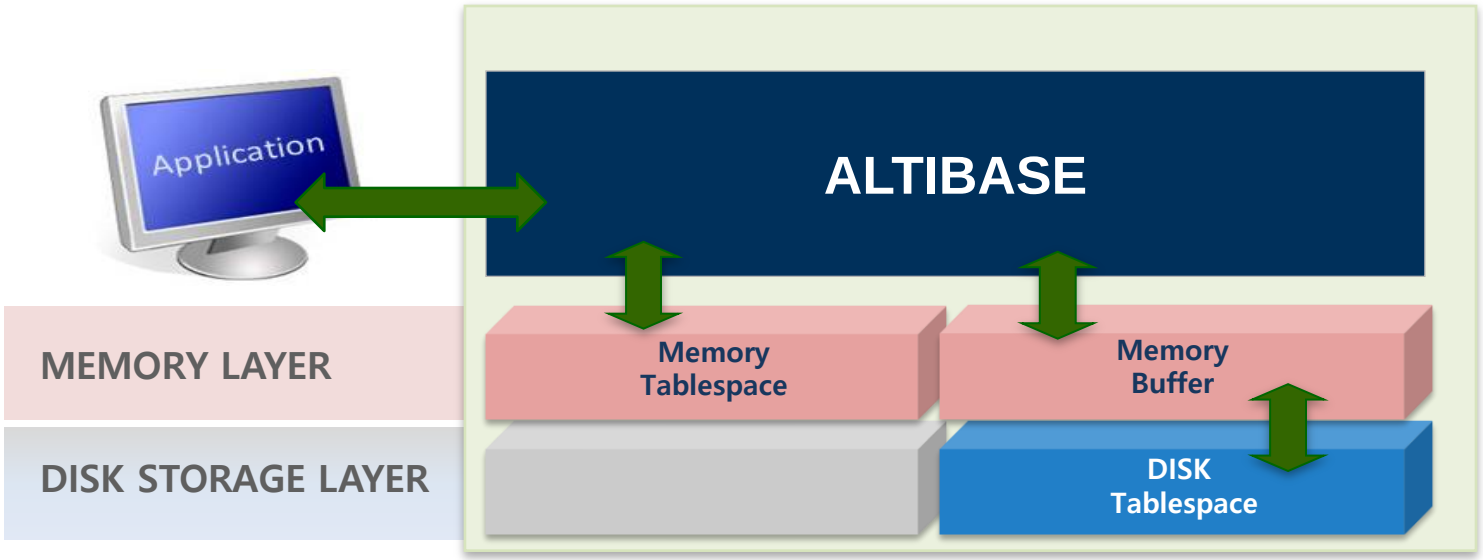
운영적 비용 : Hybrid DBMS만을 관리 [비용 ½ 절감]

구성적 비용 : Hybrid DBMS만을 관리 [비용 ½ 절감]

ALTIBASE CONCEPT

❖ HYBRID DBMS 개념

➢ 사용자는 메모리 DBMS, 디스크 DBMS 구분없이 하나의 DBMS만 접근



INTRODUCTION TO ALTIBASE

COMPARISON WITH OTHER DBMS

COMPARISON WITH OTHER DBMS

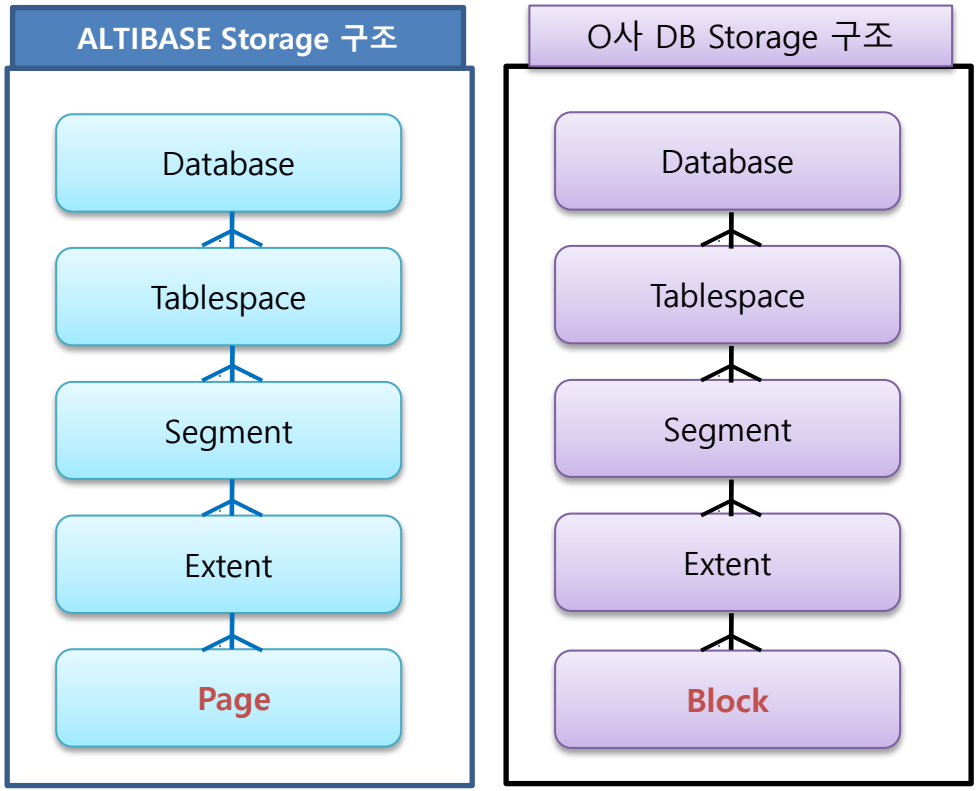
기능	ALTIBASE	O사 DB	비고
DBMS Process구조	Multi Thread 구조	Multi Process 구조	
모 델	Relational DBMS 구조	Relational DBMS 구조	
ARCHITECTURE	CLIENT-SERVER 구조	CLIENT-SERVER구조	
High Availability 방식	이중화 (Replication)	데이터베이스 클러스터링 (RAC)	Replication 은 테이블의 데이터만 복제
	별개 Instance	별개 Instance	
	Storage 별도	Storage 공유	
	스키마 별도	스키마 공유	
	데이터 복제	데이터 공유	
64bit 모드 지원	지원됨	지원됨	
Locking Mode	Row-Level Locking	Row-Level Locking	MVCC 지원
DB Recovery	Datafile & Redo logfile 이용	Datafile & Redo logfile 이용	
DeadLock Detection	Auto Deadlock Detect & Recovery	Auto Deadlock Detect & Recovery	

COMPARISON WITH OTHER DBMS

기능	ALTIBASE	O사 DB	비고
DB정보 파일	loganchor 파일	Control 파일	
Online 로그 파일	Redo log 파일(Sequential)	Redo log 파일 (Recycle)	
Archive 로그 파일	logfile0 ~	%t_%s_%r.arc	
Undo TBS	SYS_TBS_DISK_UNDO	UNDO 사용자 지정	
System TBS	SYS_TBS_MEM_DIC SYS_TBS_MEM_DATA SYS_TBS_DISK_DATA	SYSTEM SYSAUX	
Temp TBS	SYS_TBS_DISK_TEMP 사용자 지정	TEMP 사용자 지정	
Memory TBS	사용자 지정	없음	
Volatile TBS	사용자 지정	없음	
Disk TBS	사용자 지정	사용자 지정	

COMPARISON WITH OTHER DBMS

❖ STORAGE 구조 비교



COMPARISON WITH OTHER DBMS

❖ 일반 기능 비교

	ALTIBASE	O사 DB	비고
Table	지원됨	지원됨	
Multi Key-Index	지원됨	지원됨	
Stored Procedure	지원됨	지원됨	
Stored Function	지원됨	지원됨	
Package	지원됨	지원됨	
Trigger	지원됨	지원됨	
View	지원됨	지원됨	
Sequence	지원됨	지원됨	
Queue	지원됨	지원됨	
Monitoring View	지원됨	지원됨	
권한관리	지원됨	지원됨	
Role	지원됨	지원됨	V6.5 부터 지원
Snapshot	지원 됨	지원됨	V7.1 부터 지원
DB Link	지원됨	지원됨	

COMPARISON WITH OTHER DBMS

❖ 일반 기능 비교

	ALTIBASE	O사 DB	비고
Synonym	지원됨	지원됨	
Table partitioning	지원됨	지원됨	
User Defined Type	지원됨	지원됨	Procedure 에서만 지원
Cluster Object	지원 안됨	지원됨	
On-Line Backup	지원됨	지원됨	
XML 지원	지원 안됨	지원됨	
DB공간 자동확장	지원됨	지원됨	

COMPARISON WITH OTHER DBMS

❖ SQL 비교

	ALTIBASE	O사 DB	비고
SQL	표준 SQL (ANSI-SQL92지원)	표준 SQL , 변형 SQL (ANSI-SQL92, ANSI-SQL1999 지원)	ANSI-SQL1999의 객체 지향 기능은 지원하지 않음
Sub-query(In-Line View)	지원됨	지원됨	
Sub-query(Scalar)	지원됨	지원됨	
Sub-query(=,IN,EXISTS)	지원됨	지원됨	
Equi Join	지원됨	지원됨	
Inner Join	지원됨	지원됨	
Outer Join	지원됨	지원됨	O사 Style Outer Join(+) 지원
Self Join	지원됨	지원됨	
계층적 SQL CONNECT BY ~ WITH	지원됨	지원됨	
Array Processing	지원됨	지원됨	
Move 구문	지원됨	지원 안됨	

COMPARISON WITH OTHER DBMS

❖ SQL 비교

	ALTIBASE	O사 DB	비고
Queue	Enqueue/Dequeue	Advanced Queue	사용 구문/방법의 차이
SELECT ~ FOR UPDATE	지원됨	지원됨	Join 사용은 지원하지 않음
SELECT DISTINCT ~	지원됨	지원됨	
UNION	지원됨	지원됨	
UNION ALL	지원됨	지원됨	
INTERSECT	지원됨	지원됨	
MINUS	지원됨	지원됨	
CERATE TABLE AS SELECT ~	지원됨	지원됨	
Literal/Bind SQL	지원됨	지원됨	
VIEW를 통한 DML	지원됨	지원됨	
Hint 기능	지원됨	지원됨	
Cost Optimizer	지원됨	지원됨	
Parallel Select	지원됨	지원됨	
Parallel Insert	지원 안됨	지원됨	
Parallel Index Build	지원됨	지원됨	

INTRODUCTION TO ALTIBASE

INSTALLATION

INSTALLATION

❖ 지원 OS 현황

OS	CPU	OS 버전	GLIBC	JDK	참고
AIX	PowerPC	AIX 6.1 TL3, AIX 6.1 TL9, AIX 7.1	-	1.5 이상	-
HP	IA64	HP-UX.IA64 11.31	-	1.5 이상	-
Linux(x86)	x86_64	Redhat 6.x, 7.x CentOS 6.8,7 Oracle Linux 6.5,6.6,7.1,7.2,7.4	2.12~2.20	1.5 이상	gcc 4.6.3 이상
Linux(PPC)	Power7, Power8	Redhat 6.5 Redhat 7.x	2.12~2.20	1.5 이상	gcc 4.6.3 이상
Linux(PPC-LE)	Power8(LE)	Redhat 7.2	2.17	1.7 이상	gcc 4.8.5 이상

INSTALLATION

❖ 지원하지 않는 OS (V7.1.0 기준)

- 32bit SERVER , CLIENT 패키지 미 지원
- HP-UX : PA-RISC 미지원 (V6.5.1 이하 버전 지원)
- Linux(x86) : ubuntu, Fedora, openSUSE 미지원
- Linux(ia64) : 미지원
- Sun OS : V6.3.1 이하 버전 지원
- Windows : V6.5.1 이하 버전 지원

INTRODUCTION TO ALTIBASE

PROPERTY

ALTIBASE PROPERTY

❖ ALTIBASE MAJOR PROPERTY

PROPERTY NAME	설 명	기본값
AUTO_COMMIT	세션에서 SQL 문을 수행할 때 하나의 SQL 문을 하나의 TRANSACTION으로 처리하여 COMMIT 여부 결정 (1 : AUTO COMMIT / 0 : NON-AUTOCOMMIT)	1
BUFFER_AREA_SIZE	BUFFER POOL 사용하는 총 메모리 크기 (가용 가능한 메모리 크기를 고려하여 설정 권장)	128(MB)
DB_NAME	데이터베이스 이름(데이터베이스 생성 이후 변경 불가)	mydb
MAX_CLIENT	데이터베이스에 접속할 수 있는 CLIENT 최대 개수	1000
MEM_MAX_DB_SIZE	서비스 과정 중 동적으로 늘어날 수 있는 메모리 데이터베이스 최대 크기 설정 된 값을 초과할 경우, TRANSACTION을 오류 처리되고 이후 수행되는 SQL 문들도 오류 처리 (단, SELECT 문 제외)	4G
MULTIPLEXING_THREAD_COUNT	ALTIBASE가 유지하는 서비스 쓰레드 최소 개수 기본값은 장비의 CPU 개수	CPU 개수
PORT_NO	TCP/IP로 CLIENT와 SERVER가 통신할 때 사용하는 포트 번호 사용자는 루트 영역을 제외한 나머지 영역(최대 65535) 중 임의 지정 가능	20300
PREPARE_LOGFILE_WAIT_COUNT	ALTIBASE는 계속해서 새로운 로그 파일을 생성해서 사용 로그파일 생성으로 인해 성능 지연을 방지하기 위해 여분의 로그 파일을 미리 생성	5

ALTIBASE PROPERTY

❖ ALTIBASE MAJOR PROPERTY

PROPERTY NAME	설 명	기본값
REMOTE_SYSDBA_ENABLE	sys 사용자가 원격에서 SYSDBA 모드로 접속할 수 있는 여부 설정 (0 : 원격에서 접속 불가 / 1 : 원격에서 접속 가능)	1
SORT_AREA_SIZE	디스크 인덱스 생성 시 데이터에서 추출한 키들을 정렬할 때 사용 될 메모리 크기	1048576 (byte)
SQL_PLAN_CACHE_SIZE	SQL 플랜 캐시의 최대 크기	64(MB)
TIMED_STATISTICS	WAIT EVENT 대기 시간과 SQL 연산의 소요 시간 측정 여부 설정 (0 : 측정하지 않음 / 1 : 측정)	0
TRANSACTION_TABLE_SIZE	ALTIBASE 서비스 과정 중 동시에 생성될 수 있는 TRANSACTION 개수 이에 대해 메모리가 미리 할당	1024
TRX_UPDATE_MAX_LOGSIZE	하나의 DML에 의해 생서되는 로그 크기가 설정된 값보다 커지면 해당 TRANSACTION을 중단하고 오류 반환 사용자 실수로 인한 대용량 배치 작업으로 인한 부하 발생으로 성능 저하 방지하기 위해 사용	10485760 (byte)

INTRODUCTION TO ALTIBASE

ALTIBASE DIRECTORY

ALTIBASE DIRECTORY

❖ ALTIBASE 디렉토리 구조

디렉토리 명	설 명
/admin	ALTIBASE 시스템 정보와 관련된 view를 생성하는 adminview.sql 파일과 프로시져, 테이블 정보를 볼 수 있는 프로시져를 생성하는 SQL 파일
/arch_logs	복구를 위해 로그 파일을 백업하는 백업 디렉토리
/audit	이중화 동작 시 발생한 불일치를 해결하는 ALTIBASE 유틸리티인 audit의 예제 스크립트 파일이 들어있는 디렉토리
/bin	ALTIBASE를 포함한 ALTIBASE 관리도구와 ALTIBASE를 사용하는데 필요한 지원 도구들의 실행 파일이 존재하는 디렉토리
/conf	ALTIBASE 환경설정파일(altibase.properties) 과 라이선스(license) 파일 저장
/dbs	기본 프로퍼티를 이용할 경우 데이터베이스의 파일들이 생성되는 디렉토리
/include	ALTIBASE 응용프로그램 작성에 필요한 makefile을 위한 매크로 설정 등이 포함된 altibase_env.mk 파일과 README 파일
/install	sqlplus 와 같은 유틸리티 사용 시 화면에 출력되는 message file 들이 저장
/lib	응용 프로그램 작성에 필요한 라이브러리를 수록한 디렉토리
/logs	로그앵커 파일들과 로그 파일들이 존재하는 디렉토리
/msg	오류 메시지를 수록한 파일들을 포함하는 디렉토리
/sample	ALTIBASE의 응용 프로그램을 샘플로 제공한 디렉토리 JDBC, ODBC, C/C++ 전처리기(precompiler) 라이브러리를 이용하여 작성된 프로그램과 Makefile 수록
/trc	ALTIBASE 운영 상태를 기록한 파일들이 존재

INTRODUCTION TO ALTIBASE

STARTUP & SHUTDOWN

STARTUP & SHUTDOWN

❖ STARTUP 과정

- 구동 단계는 총 4단계로 구분
- 각 단계마다 "STARTUP" 이라는 구문을 이용하여 다음 단계로 전이
  - PROCESS 단계
    - 내부 모듈의 초기화 및 iSQL이 접속 가능하도록 프로세스 구동
    - 프로퍼티 및 라이센스 파일의 유무 및 정합성 체크
    - 데이터베이스작업도 일체 수행할 수 없음 (단, CREATE/DROP DATABASE만 수행 가능)
  - CONTROL 단계
    - 데이터베이스복구가 가능한 수준까지 내부 모듈 준비
    - 복구 수행 및 데이터베이스 운영모드(아카이브모드)를 변경 가능
    - 데이터베이스 복구에 필요한 메타 및 성능 뷰 조회 가능
  - META 단계
    - 데이터베이스 구동을 위해 메모리/디스크 데이터 파일(체크포인트 이미지 파일) 정합성 체크
    - Restart recovery 수행
    - 내부 주요 쓰레드 활성화
  - SERVICE 단계
    - 사용자가 데이터베이스를 사용할 수 있도록 모든 준비 완료

STARTUP & SHUTDOWN

❖ DBMS STARTUP

```
Shell::/home/alti1> isql -sysdba
------------------------------------------------------------------
     Altibase Client Query utility.
     Release Version 7.1.0.1.0
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
------------------------------------------------------------------
Write UserID : sys
Write Password : manager
ISQL_CONNECTION = UNIX, SERVER = 127.0.0.1, PORT_NO = 20301
[ERR-910FB : Connected to idle instance]
iSQL(sysdba)> STARTUP SERVICE;
Connecting to the DB server... Connected.…..
….
 [RP] Initialization : [PASS]
--- STARTUP Process SUCCESS ---
```

❖ 구동 이후 프로세스 확인

➢ OS에서 제공하는 "ps" 명령을 통해 프로세스ID 확인 가능

(alti1 사용자인 경우)

```
Shell::/home/alti1> ps -ef ¦ grep "altibase -p boot from" ¦ grep alti1 ¦ grep -v grep
alti1 26804     1  0 13:05:09 ?         0:51
/home/alti1/altibase_home/bin/altibase -p boot from admin
```

➢ OS 명령어를 이용한 CPU 및 메모리 사용량

```
Shell::/home/alti1> export UNIX95=1  (일부 OS에서 아래 ps -o 옵션이 안될 경우 사용)
Shell::/home/alti1> ps -o pcpu,vsz -p <altibase_process_id>
 %CPU       VSZ
 0.71       825652   (Kbyte단위로 표시됨)
```

STARTUP & SHUTDOWN

❖ DBMS SHUTDOWN 과정

- SHUTDOWN 과정은 STARTUP의 역순 진행
- STARTUP과 달리 단계의 전이는 존재하지 않음
- "SHUTDOWN" 구문을 통해 진행 가능
- 아래의 3가지 옵션에 따라 다른 동작 수행
  - NORMAL 옵션
    - 접속한 세션이 정상 종료될 때까지 대기 후 STOP 과정 진행
  - IMMEDIATE 옵션
    - 접속한 세션을 강제 종료시킨 후 STOP 과정 진행
    - 강제 종료되는 세션의 TRANSACTION은 모두 롤백 처리
  - ABORT 옵션
    - "kill -9" 와 같이 프로세스 강제 종료
    - 이 경우 정상 종료와 달리 재 구동 시 restart recovery라는 자동 복구 과정 진행

STARTUP & SHUTDOWN

❖ DBMS STOP

```
Shell::/home/alti1> isql  -sysdba
--------------------------------------------------------------------
     Altibase Client Query utility.
     Release Version 7.1.0.1.0
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
--------------------------------------------------------------------
Write UserID : sys
Write Password : manager
ISQL_CONNECTION = UNIX, SERVER = 127.0.0.1, PORT_NO = 20301
[ERR-910FB : Connected to idle instance]
iSQL(sysdba)> SHUTDOWN IMMEDIATE;
Ok..Shutdown Proceeding....
TRANSITION TO PHASE : Shutdown Altibase
  [RP] Finalization : PASS
shutdown immediate success.
```

STARTUP & SHUTDOWN

❖ server 스크립트

- $ALTIBASE_HOME/bin/ 에 위치
- 데이터베이스의 구동, 종료 및 생성 등을 손쉽게 사용하기 위한 스크립트

명령 행 인자 구분	설명
create	데이터베이스를 생성하는 기능 ex) shell> server create MS949 UTF8
start	ALTIBASE를 구동 시키는 기능 ex) shell> server start
stop	ALTIBASE를 정상적인 과정으로 종료하는 기능 ex) shell> server stop
restart	ALTIBASE를 재구동하는 기능 ex) shell> server restart
kill	ALTIBASE를 강제로 kill 시키는 기능 ex) shell> server kill

STARTUP & SHUTDOWN

❖ server 스크립트의 start 부분

```
#!/bin/sh
ADMIN="${ALTIBASE_HOME}/bin/isql -u sys -p manager  - sysdba  -noprompt"
ISQL="${ALTIBASE_HOME}/bin/isql -s localhost -u sys -p manager -silent"
if [ "$1" = "status" ]; then
        MODE=`echo $* | cut -f 2-4 -d ' '`
        if [ "$MODE" = "status" ]; then
                MODE=
        fi
fi
case "$1" in                        # server 실행 시 첫번째 인자값이 "start" 인 경우
'start')
# for WINDOWS natc
    if [ -f ${ALTIBASE_HOME}/bin/chkFileLock ]; then    # 이미 구동된 상태인지 체크
        chkFileLock >> ${ALTIBASE_HOME}/flock.log
    fi
        ${ADMIN} << EOF     # startup SQL구문을 iSQL을 통해 실행하는 형태
startup
quit
EOF
```

STARTUP & SHUTDOWN

❖ server 스크립트의 stop 부분

```
#!/bin/sh
ADMIN="${ALTIBASEHOME}/bin/isql -u sys -p manager  -sysdba  -noprompt"
ISQL="${ALTIBASE_HOME}/bin/isql -s localhost -u sys -p manager -silent"
…..
…..
case "$1" in              # server 실행 시 첫번째 인자값이 "stop" 인 경우
'stop')
        ${ADMIN} << EOF > /dev/null
ALTER SYSTEM SET CHECKPOINT_BULK_WRITE_PAGE_COUNT = 0;
ALTER SYSTEM SET CHECKPOINT_BULK_WRITE_SLEEP_SEC  = 0;
ALTER SYSTEM SET CHECKPOINT_BULK_WRITE_SLEEP_USEC = 0;
quit;
EOF
        killCheckServer > ${ALTIBASE_HOME}/trc/killCheckServer.log 2>&1
        ${ADMIN} << EOF
shutdown immediate;       # shutdown SQL구문을 iSQL을 통해 실행하는 형태
quit;
EOF
```

STARTUP & SHUTDOWN

❖ server 스크립트를 이용하는 구동

```
Shell::/home/alti1> server start
------------------------------------------------------------------
     Altibase Client Query utility.
     Release Version 7.1.0.1.0
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
------------------------------------------------------------------
ISQL_CONNECTION = UNIX, SERVER = 127.0.0.1, PORT_NO = 20301
[ERR-910FB : Connected to idle instance]
Connecting to the DB server... Connected.
…..
…..
 [RP] Initialization : [PASS]

--- STARTUP Process SUCCESS ---
Command execute success.
```

STARTUP & SHUTDOWN

❖ server 스크립트를 이용하여 종료

```
Shell::/home/alti1> server stop
------------------------------------------------------------------
     Altibase Client Query utility.
     Release Version 7.1.0.1.0
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
------------------------------------------------------------------
ISQL_CONNECTION = UNIX, SERVER = 127.0.0.1, PORT_NO = 20301
Ok..Shutdown Proceeding....


TRANSITION TO PHASE : Shutdown Altibase
  [RP] Finalization : PASS
shutdown immediate success.
```

INTRODUCTION TO ALTIBASE

ISQL

ISQL

iSQL

- $ALTIBASE_HOME/bin 에 위치
- 데이터베이스에 접속하여 SQL 수행 및 결과를 조회하는 유틸리티
- DBA권한으로 데이터베이스 구동 및 종료, 백업 및 복구 수행
- 세션의 강제 종료 수행

```
Shell::/home/alti1> isql -u sys -p manager -port 20301 -nls_use MS949 -s 127.0.0.1
--------------------------------------------------------------------
     Altibase Client Query utility.
     Release Version 7.1.0.1.0
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
--------------------------------------------------------------------
ISQL_CONNECTION = TCP, SERVER = 127.0.0.1, PORT_NO = 20301
iSQL>
```

ISQL

❖ is 스크립트

- $ALTIBASE_HOME/bin 에 위치
- iSQL의 입력 시 옵션들을 생략하고 접속하기 위한 스크립트

```
Shell::/home/alti1> cat $ALTIBASE_HOME/bin/is
#!/bin/sh

trap "" TSTP
${ALTIBASE_HOME}/bin/isql -s 127.0.0.1 -u sys -p manager $*
```

ISQL

❖ iSQL 실행 시 입력 옵션

입력 옵션	설명
-s	ALTIBASE SERVER가 위치한 IP 지정
-u	ALTIBASE 사용자 명 지정
-p	ALTIBASE 사용자 패스워드 지정
-port	ALTIBASE Listen Port번호 지정
-nls_use	ALTIBASE 생성 시 입력한 문자셋 지정
-o	iSQL에서 실행한 결과를 저장할 파일명 지정
-f	iSQL에서 수행할 SQL 및 명령을 저장한 입력 파일명 지정
-h	입력 옵션에 대한 도움말 출력
-silent	iSQL 배너를 화면에 출력하지 않게 할 경우 지정

ISQL

❖ iSQL 실행 후 옵션 (1/2)

옵션	설명
desc	테이블 구성 정보 확인
@	지정된 파일명 실행
!	OS 명령을 수행하고자 할 경우
h	수행된 SQL 목록 확인
/	직전에 수행한 SQL 재수행
ed	직전에 수행한 SQL 편집하고자 할 경우
autocommit	현재 세션의 autocommit 모드를 변경할 경우
spool [fileName]	spool 명령에 입력된 파일에 현재 수행 결과 기록
show all	iSQL의 현재 설정 상태 및 사용자 명 출력
show user	iSQL에 접속한 사용자 명 출력

ISQL

❖ iSQL 실행 후 옵션 (2/2)

옵션	설명
colsize	컬럼 사이즈 길이 지정
linesize	하나의 레코드의 출력라인 길이 지정
pagesize	지정된 개수만큼 레코드 출력 후 컬럼 타이틀 출력
heading	출력 결과에서 컬럼 타이틀을 보이거나 감추도록 설정
timing	실행한 SQL 수행 시간을 1/100 초 단위 출력
vertical	컬럼들을 세로로 출력 한 라인에 (컬럼 명 : Value) 형태로 결과 출력
foreignkeys	desc 명령으로 테이블 조회 시 참조키 정보 출력
querylogging	수행한 변경 SQL를 $ALTIBASE_HOME/trc/isql_query.log파일에 수행한 SQL를 시간 순으로 저장

ISQL

❖ glogin.sql

- iSQL 접속할 때 자동으로 수행시키려는 설정 값을 전역으로 설정할 경우
- $ALTIBASE_HOME/conf/glogin.sql 로 저장

❖ login.sql

- glogin.sql과 동일하며 iSQL을 실행하는 유저에게만 적용하고자 할 경우
- iSQL을 실행하는 디렉토리에 login.sql을 저장
- glogin.sql , login.sql 이 동시에 존재 할 경우 glogin.sql→login.sql 순으로 적용

❖ iSQL에서 주석처리

- Single Line 주석 : -- comment
- Multi Line 주석 : /* comment */

```
iSQL> -- test comment
iSQL> SELECT /* comments */ sysdate FROM DUAL;
```

ISQL

❖ iSQL 사용 예 : 일반 조회 시와 VERTICAL 옵션의 사용 차이

```
iSQL> SELECT * FROM t1 LIMIT 3;
A
--------------
1
2
3
3 rows selected.

iSQL>  SET VERTICAL ON;

iSQL> SELECT * FROM t1 LIMIT 3;
A : 1
A : 2
A : 3
3 rows selected.
iSQL>
```

ISQL

❖ iSQL 사용 예 : 실행 계획 조회할 경우

```
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
Alter success.
iSQL> SELECT * FROM t1 WHERE a = 2;
A
--------------
2
1 row selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 1, TUPLE_SIZE: 4 )
 SCAN ( TABLE: T1, INDEX: __SYS_IDX_ID_102, ACCESS: 1, SELF_ID: 2 )
------------------------------------------------------------
```

- PLAN 옵션 = [ON / ONLY / OFF]

❖ iSQL 사용 예 : 세션의 강제 종료

➢ 대상이 되는 세션의 고유번호 확인 필요

```
Shell> is
iSQL> ALTER DATABASE db_name SESSION CLOSE session_id;

< 정상 처리의 결과 >
iSQL> ALTER DATABASE mydb SESSION CLOSE 14;
Alter success.

< 존재하지 않는 세션을 단절하려고 할 경우 >
iSQL> ALTER DATABASE mydb SESSION CLOSE 14;
[ERR-41080: Invalid Session ID 14]
```

ALTIBASE ADMINISTRATION I

ARCHITECTURES

ARCHITECTURES

CONTENTS

- FEATURES
- ARCHITECTURE
- TABLESPACE MANAGEMENT
- TRANSACTION MANAGEMENT
- BUFER MANAGEMENT

ARCHITECTURES

FEATURES

FEATURES

❖ CLIENT-SERVER 엔진 구조

- CLIENT-SERVER 프로토콜을 선택(TCP/IP, IPC, Unix Domain Socket)

❖ 인터페이스

- ANSI SQL 92/SQL99 Full Spec, ODBC 3.0, JDBC 2.0, C/C++, ADO.NET, C/C++ Precompiler 제공

❖ 다중버전 동시성 제어 기법(MVCC)

- TRANSACTION 동시성 제어를 위해 다중 버전 동시성 제어
- MVCC(Multi Version Concurrency Control) 기법 사용

FEATURES

❖ 고립화 수준

- TRANSACTION 고립화 수준을 Commited Read(default), Repeatable Read, No Phantom Read(=Serializable) 지정

❖ FUZZY / PING-PONG CHECKPOINT 제공

- 체크포인트 수행 중 발생한 TRANSACTION을 처리할 수 있도록 Fuzzy Checkpoint를 제공하며 체크포인트 중에도 TRANSACTION 처리를 빠르게 하는 Ping-Pong Checkpoint 제공

❖ DEADLOCK 감지

- 비정상적인 TRANSACTION 감지를 위해 프로세스를 따로 두지 않고 DEADLOCK 발생 순간 에러 발생하여 신속 조치

FEATURES

❖ STORED PROCEDURE / FUNCTION 제공

- ➢ 업무 절차를 하나의 SERVER 모듈로 만든 후, DBMS에 저장하여 모듈 이름만 호출로 업무 프로세스를 DBMS에 수행 가능

❖ 데이터베이스 이중화

- ➢ TCP 기반 네트워크를 통해 변경 TRANSACTION 로그를 전송하고 원격 SERVER에서 이를 반영하여 데이터 복제 수행

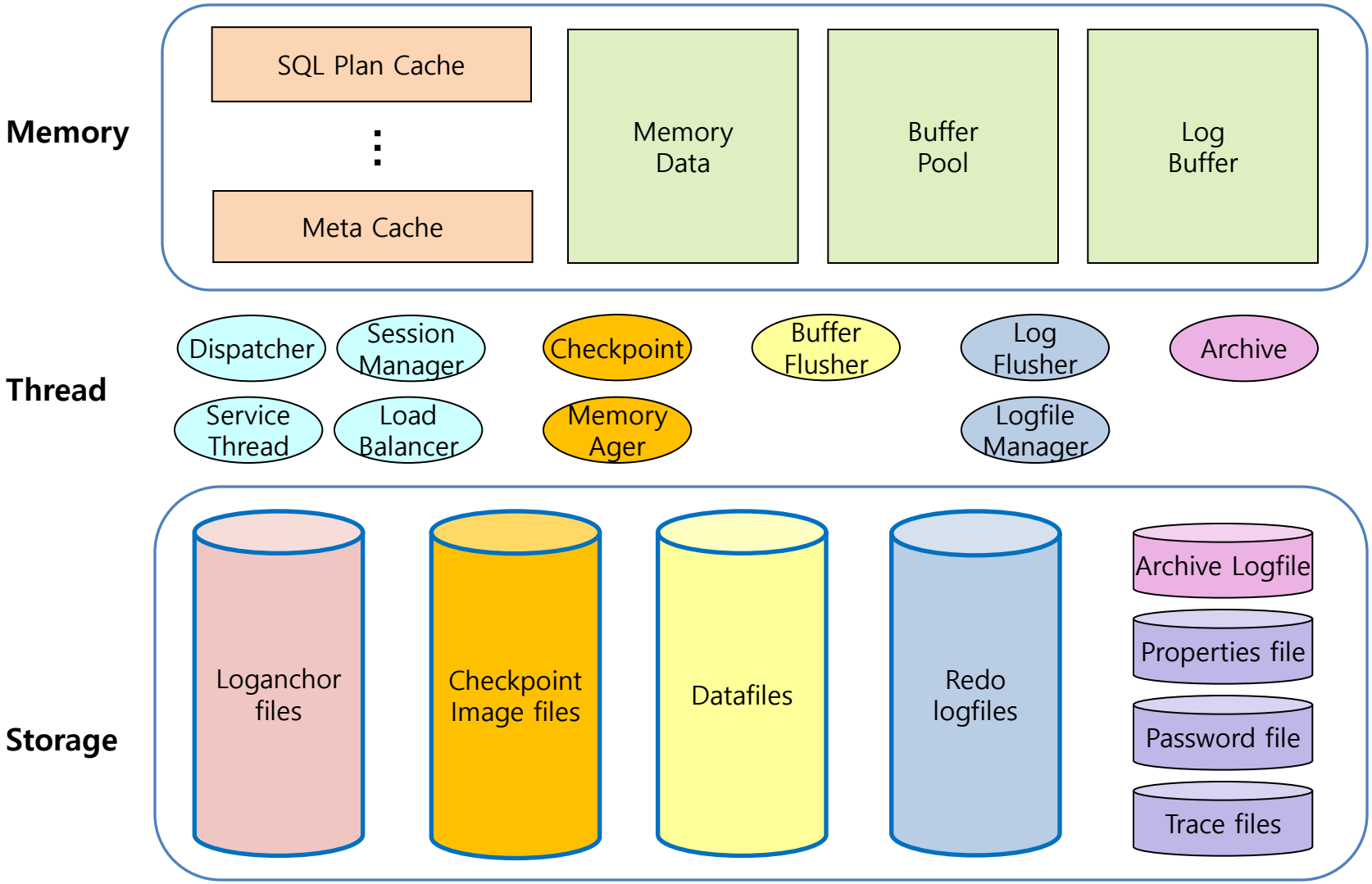
❖ 데이터베이스 공간

- ➢ ALTIBASE는 하나 이상의 테이블스페이스로 구성되고, 테이블스페이스는 메모리와 디스크로 나뉨

ARCHITECTURES

ARCHITECTURE

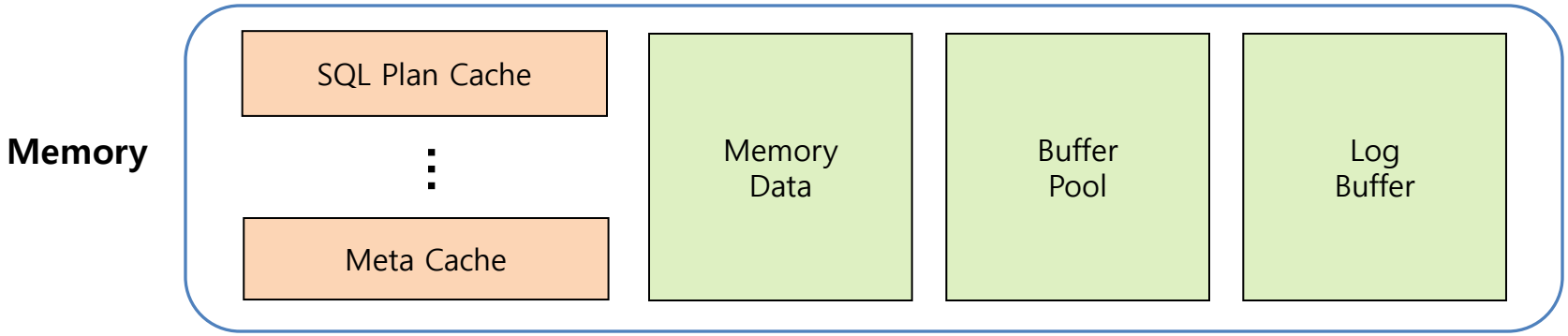
ARCHITECTURE



ARCHITECTURE

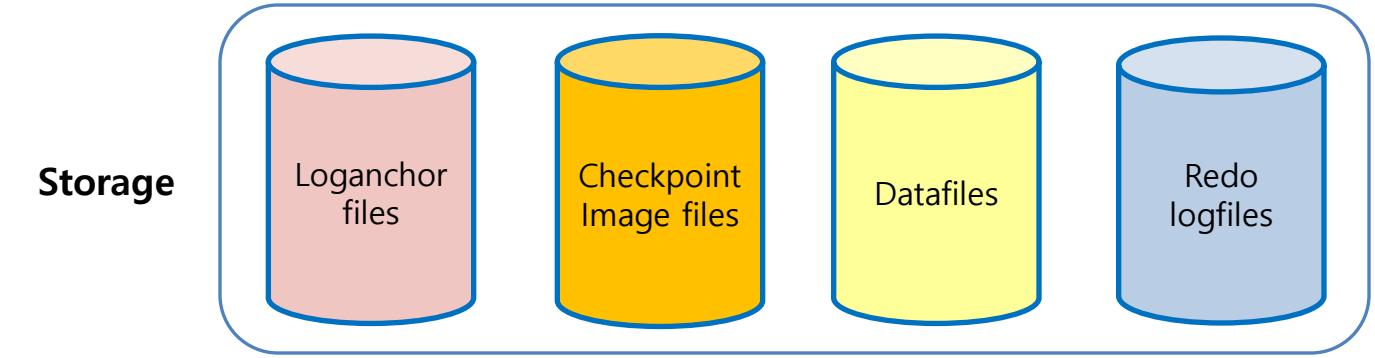
❖ ARCHITECTURE(MEMORY)

- Memory Data
  - 메모리 데이터 공간이며, 모든 TRANSACTION 처리가 메모리에서 처리
- Buffer Pool
  - 디스크 I/O를 줄이고 성능 향상을 위해 디스크 데이터 일부를 메모리에 적재하여 사용하는 공간
- Log Buffer
  - 변경 TRANSACTION 처리 시 생성되는 redo log를 저장하는 공간
- Cache
  - 모든 세션이 공유하는 영역
  - SQL Plan Cache, Stored Procedure Cache, Meta Cache로 구성



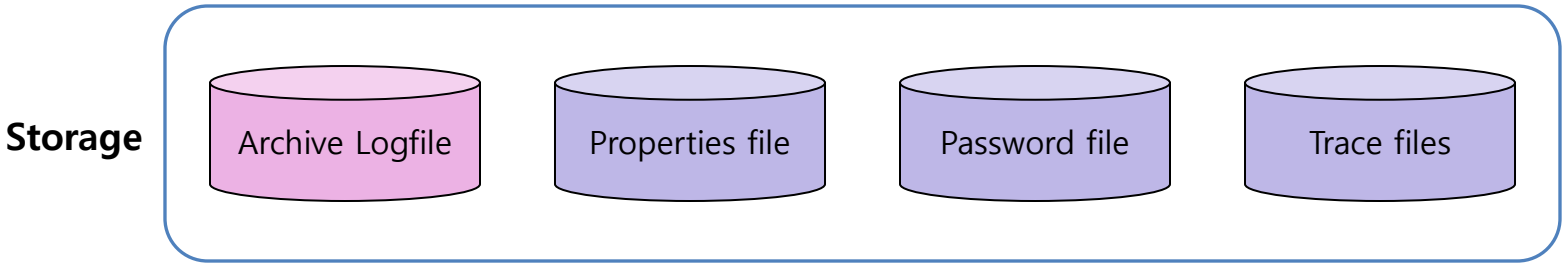
ARCHITECTURE(STORAGE)

- Checkpoint image files
  - Memory는 휘발성 매체이기 때문에 메모리 데이터를 디스크에 안정적으로 백업하는 물리적인 파일(장애 발생 시, 복구 시간 단축)
- Datafiles
  - 디스크 데이터를 저장하는 물리적인 파일
- Redo logfiles
  - Log Buffer에 기록된 변경정보가 저장되는 물리적인 파일
- Loganchor files
  - 데이터베이스의 구동정보, 데이터파일 경로, 복구 시점 등에 대한 전반적인 데이터베이스 정보가 기록되는 파일



ARCHITECTURE(STORAGE)

- Archive logfiles
  - 로그 파일의 백업 파일로, 데이터베이스가 아카이브 모드로 운영 중일 때만 생성
- Properties File
  - 데이터베이스 Property 가 저장된 파일, $ALTIBASE_HOME/conf/altibase.properties
- Password File
  - DBA 유저인 SYS 유저에 대한 패스워드가 암호화되어 저장된 파일
- Trace Files
  - 데이터베이스 운영 상태 및 발생하는 에러, 장애 상황 등이 기록되는 파일

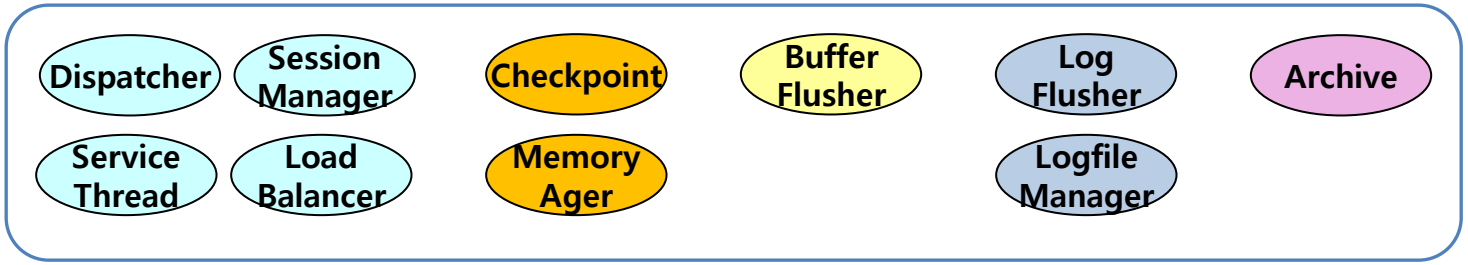


ARCHITECTURE

❖ ARCHITECTURE(THREAD)

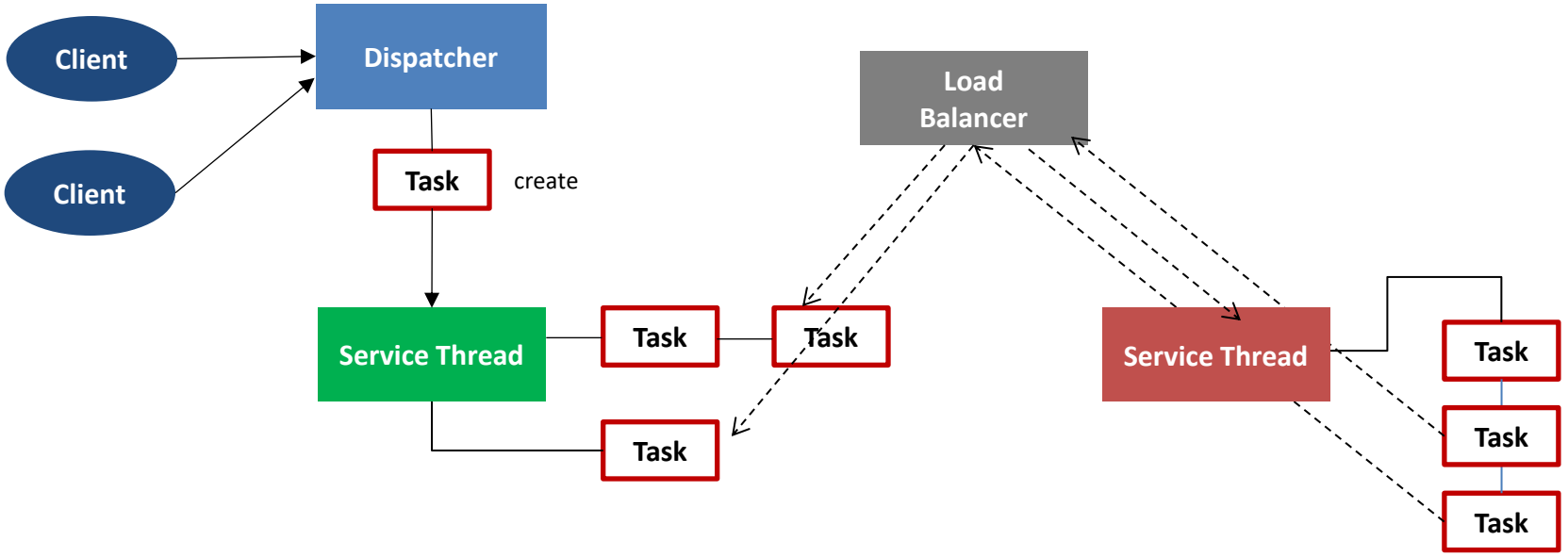
Thread	설명
Dispatcher	CLIENT의 접속 요청을 감지하여 Service Thread 에게 Task 전달
Service Thread	CLIENT가 요청한 SQL를 처리하고 완료된 내용 전달
Load Balancer	Service Thread의 부하를 감지하여 Service Thread를 추가/삭제하고 Task 분배
Session Manager	세션들의 Timeout 상태를 감지하고 처리
Checkpoint	메모리 데이터를 물리적인 파일에 기록
Memory Ager	메모리 테이블 변경 작업 후, 불필요한 이미지 삭제
Buffer Flusher	Buffer Pool에 있는 Dirty 페이지를 데이터 파일에 Sync
Logfile Manager	현재 사용 중인 로그 파일 외에 미리 여유 로그 파일 생성
Log Flusher	로그 버퍼에 기록된 변경정보를 물리적인 Redo logfile로 기록
Archive	모드에서 리두 로그파일이 스위칭 될 때마다 archiving 수행

Thread



❖ 서비스 쓰레드(SERVICE THREAD)

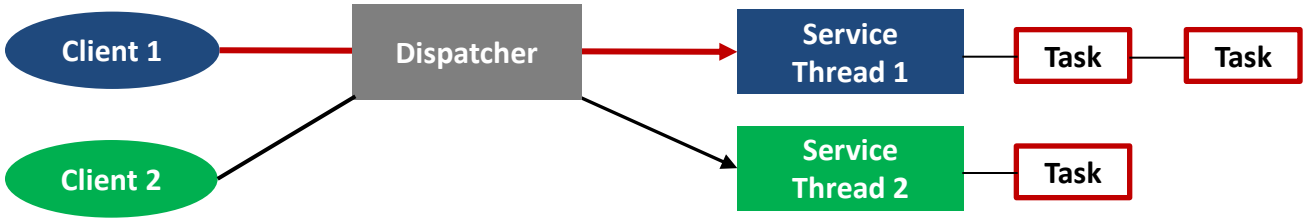
- Multiplexing Thread Mode
  - 여러 개의 세션을 하나의 쓰레드가 관리하는 방식
  - 동시 접속 사용자가 많은 경우에 유리하며, SERVER의 자원 사용량이 적음
  - 태스크 분배 시 오버헤드가 존재하여 사용자가 적은 경우 자원이 낭비될 수 있음



❖ 서비스 쓰레드(SERVICE THREAD)

➢ Dedicated Thread Mode

- 하나의 세션을 하나의 쓰레드가 관리하는 방식
- load balancing 을 통한 태스크 재분배 과정을 거치지 않고 각각의 세션이 통신을 수행하여 속도가 빠르고 관리에 용이
- 세션 연결 시마다 쓰레드를 늘려야 하기 때문에 사용자 증가에 따른 CPU 사용량도 증가

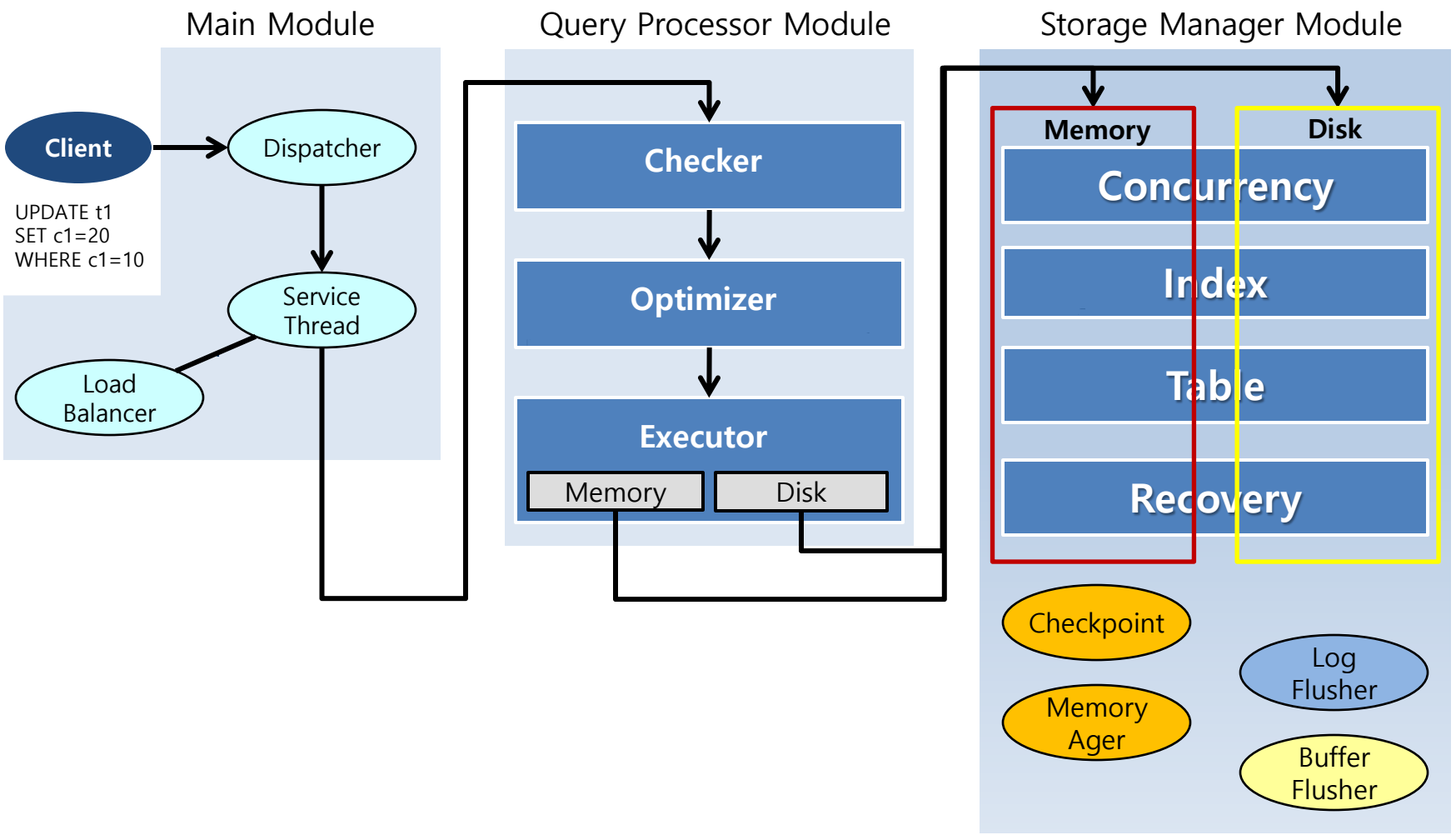


➢ Dedicated thread mode 설정 방법

- DEDICATED_THREAD_MODE 값을 1로 설정(기본값은 0)
- CPU affinity 를 적용 시, multiplexing mode에 비해 성능이 크게 향상
  - CPU affinity : CPU 연계 기능을 활성화하면 하나의 서비스 쓰레드는 동일한 CPU 코어에만 할당되어 다른 CPU 코어로 쓰레드 정보를 이동하는 비용 감소

ARCHITECTURE

❖ 내부프로세스



ARCHITECTURES

TABLESPACE MANAGEMENT

TABLESPACE MANAGEMENT

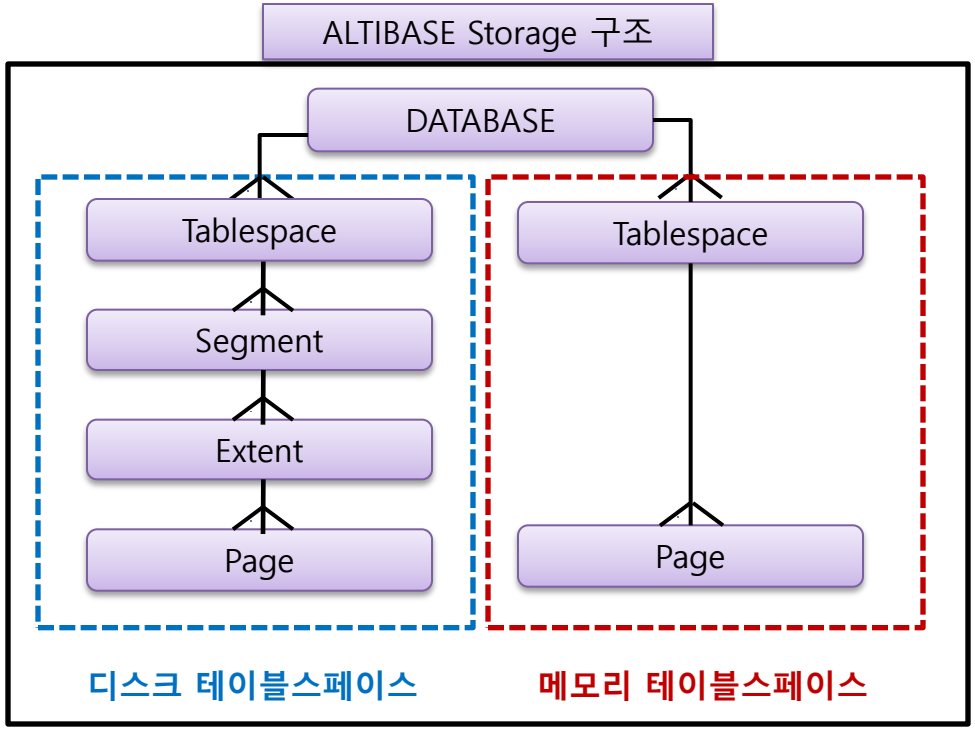
❖ 테이블스페이스(TABLESPACE / TBS)

- 데이터베이스를 구성하는 최상위 논리적인 구조
- 테이블, 인덱스 등의 데이터베이스 객체들이 저장되는 논리적인 저장소
- 데이터베이스 운영을 위해 기본적으로 하나 이상의 테이블스페이스 필요

TABLESPACE MANAGEMENT

❖ STORAGE 구조

- 하나의 데이터베이스는 한 개 이상의 테이블스페이스로 구성되며, 하나의 테이블스페이스는 다수의 세그먼트 또는, 다수의 페이지로 구성



- 메모리 테이블스페이스
  - 32K 크기의 페이지들로 구성
- 디스크 테이블스페이스
  - 다수의 세그먼트로 구성
  - 세그먼트는 다수의 익스텐트로 구성
  - 익스텐트는 8K 크기의 페이지 64개로 구성(512K)

TABLESPACE MANAGEMENT

❖ ALTIBASE에서 제공하는 테이블스페이스 종류

- 데이터 속성에 따른 분류
  - 메모리 테이블스페이스(Memory Tablespace)
  - 디스크 테이블스페이스(Disk Tablespace)
- 생성시점에 따른 분류
  - 시스템 테이블스페이스(System Tablespace)

사용자	테이블스페이스 종류
시스템	SYSTEM DICTIONARY TABLESPACE SYSTEM UNDO TABLESPACE
일반 사용자, SYS	SYSTEM MEMORY DEFAULT TABLESPACE SYSTEM DISK DEFAULT TABLESPACE SYSTEM DISK TEMPORARY TABLESPACE

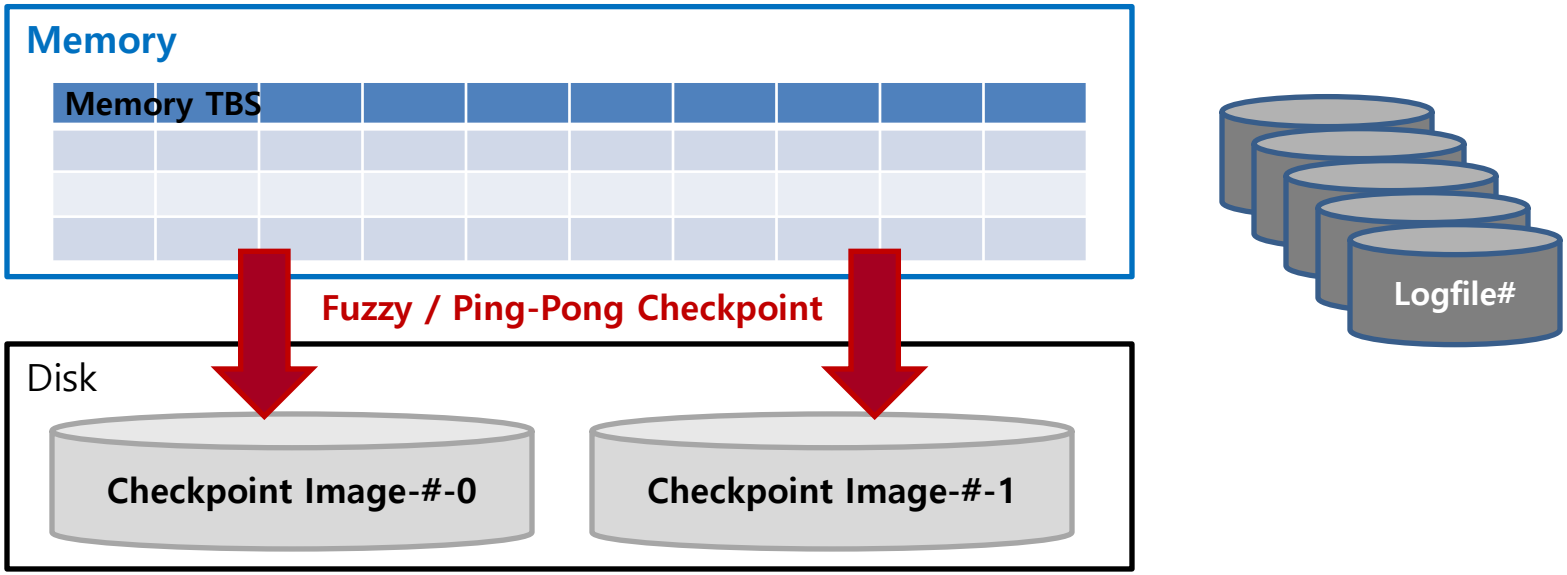
  - 사용자 테이블스페이스(User Tablespace)
    - 사용자의 필요에 따라 선택적으로 생성
    - 임시 TBS, 데이터 TBS(메모리 TBS, 휘발성 TBS, 디스크 TBS)

TABLESPACE MANAGEMENT

❖ 메모리 테이블스페이스(MEMORY TABLESPACE)

- ➢ 모든 데이터가 메모리 공간에 저장되는 테이블스페이스를 의미
- ➢ ALTIBASE v4까지는 하나만 존재(추가 불가)
- ➢ ALTIBASE v5부터는 사용자/업무별로 확장하여 추가 가능

❖ 구조 (메모리 TBS + 체크포인트 이미지 파일)



TABLESPACE MANAGEMENT

❖ 메모리 테이블스페이스의 공간 할당

- ➢ 32K 페이지 단위로 테이블에 공간 할당

❖ PAGE 상태

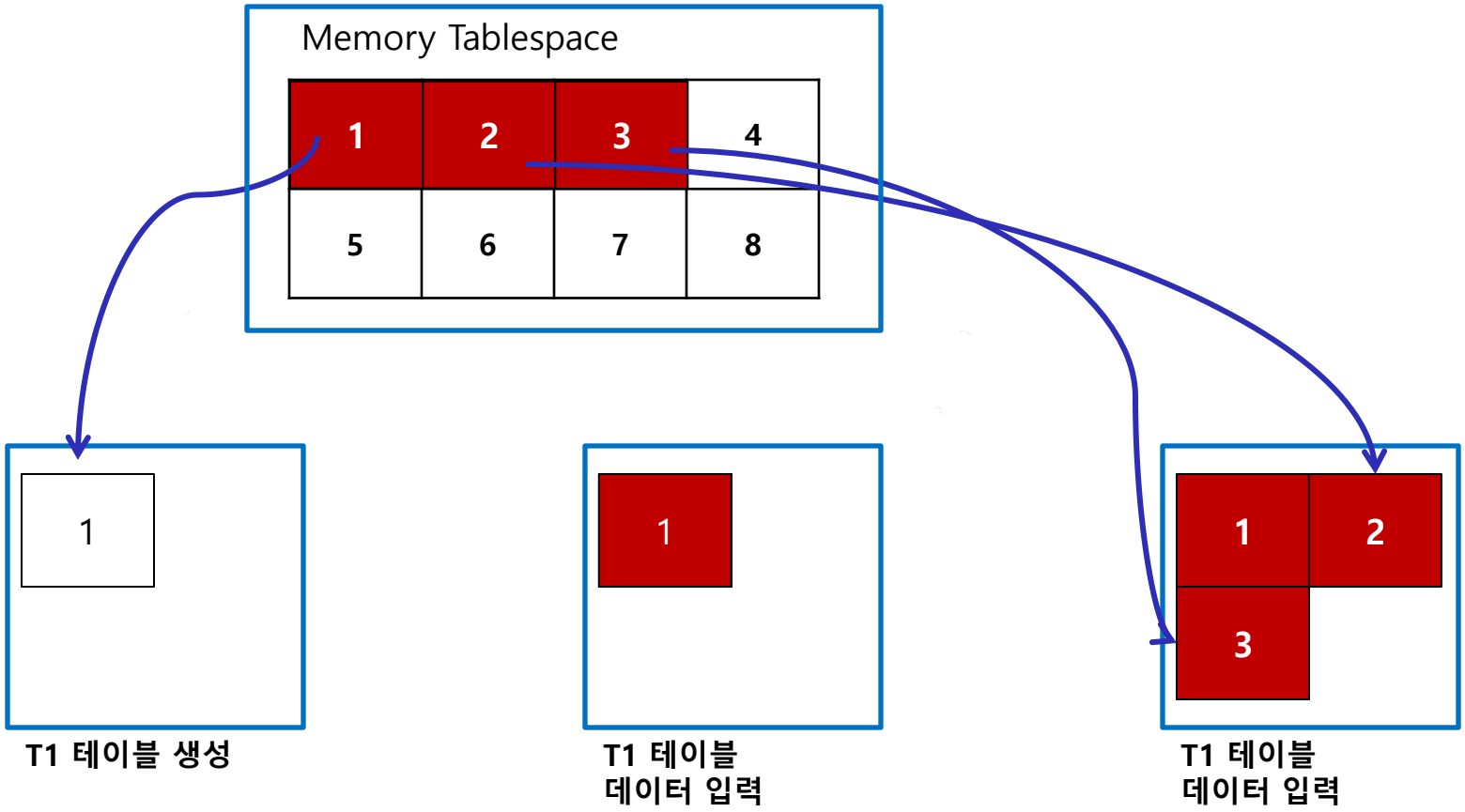
객체	Free	Used
테이블스페이스	특정 테이블에 할당되지 않은 공간 1 Page 씩 특정 테이블에 할당 가능	테이블에 할당한 공간 테이블에서 반납하기 전까지 다른 테이블에서 사용 불가
테이블	테이블이 할당 받은 공간 중 데이터가 들어 있지 않은 공간 해당 테이블 내에서 재사용 가능	테이블이 할당 받은 공간 중 데이터가 들어 있는 공간 데이터를 삭제하지 않으면 재사용 불가

❖ 메모리 테이블 PAGE 상태변화

- 메모리 테이블에 Delete 수행
  - 테이블 안에서 Page 상태가 Used → Free 로 전환(해당 테이블 내에서 재사용 가능)
  - 테이블스페이스로 Page를 반환하지 않음
  - Delete 수행 후 Compaction을 수행하면 해당 Page를 테이블스페이스에 반환(다른 테이블에서 사용 가능)하고, 테이블스페이스의 Page 상태는 Used → Free로 변경
- 메모리 테이블에 Truncate 수행
  - 테이블에 할당 되었던 Page를 테이블스페이스에 반환
  - 테이블에 할당되었던 테이블스페이스의 Page 상태는 Used → Free로 전환(다른 테이블에서 할당 받아 사용 가능)
- 메모리 테이블에 Move 수행
  - Move 구문을 통해서 데이터를 다른 테이블로 이동시켜도 Delete 한 것과 동일하게 해당 테이블 안에서만 재사용 가능
  - Move 수행 후, Compaction을 수행하면 해당 Page를 테이블스페이스에 반환

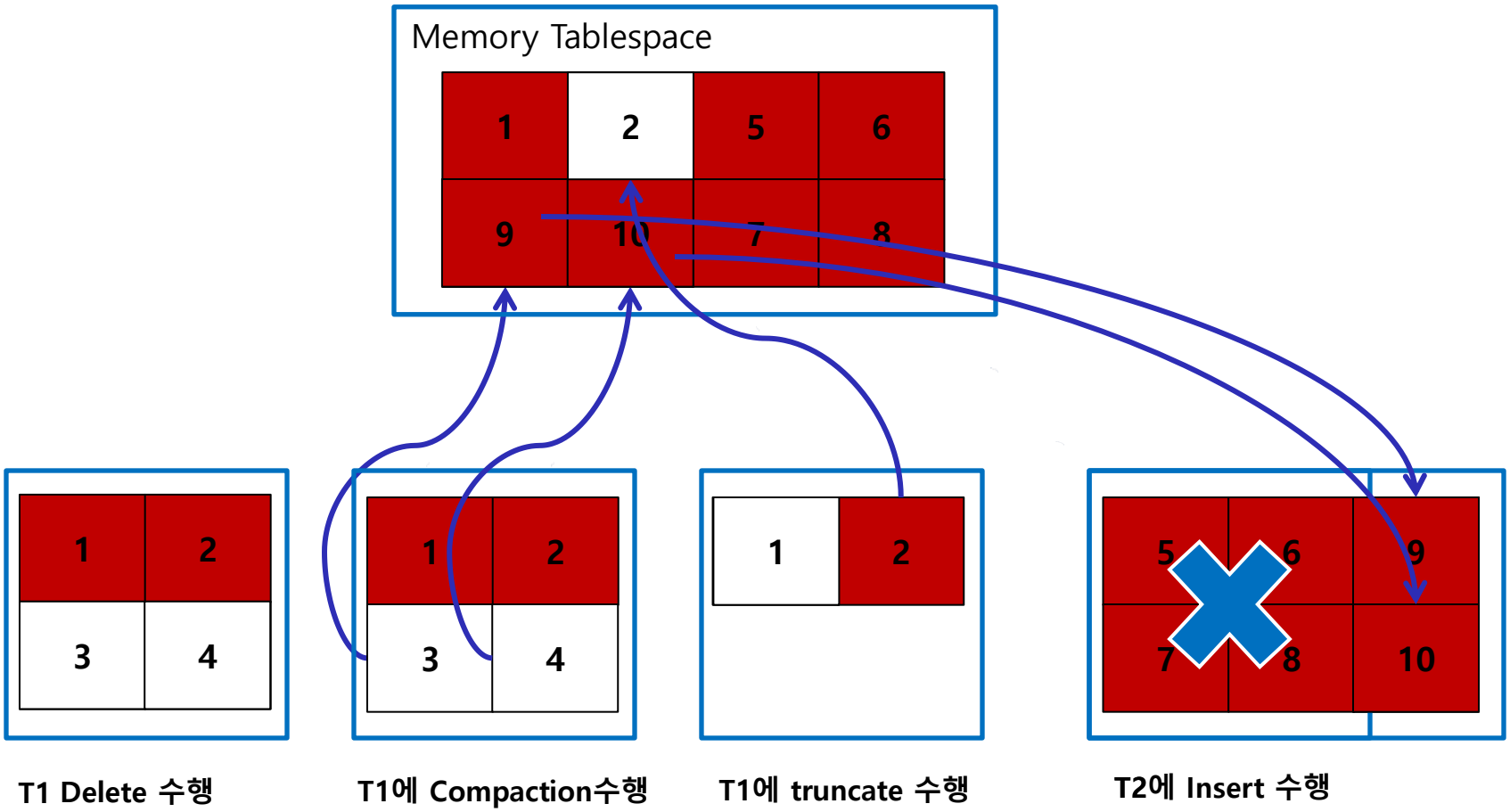
TABLESPACE MANAGEMENT

❖ 메모리 테이블스페이스 공간 할당 구조



메모리 테이블스페이스의 공간 할당

❖ 메모리 테이블스페이스 공간 반납 구조



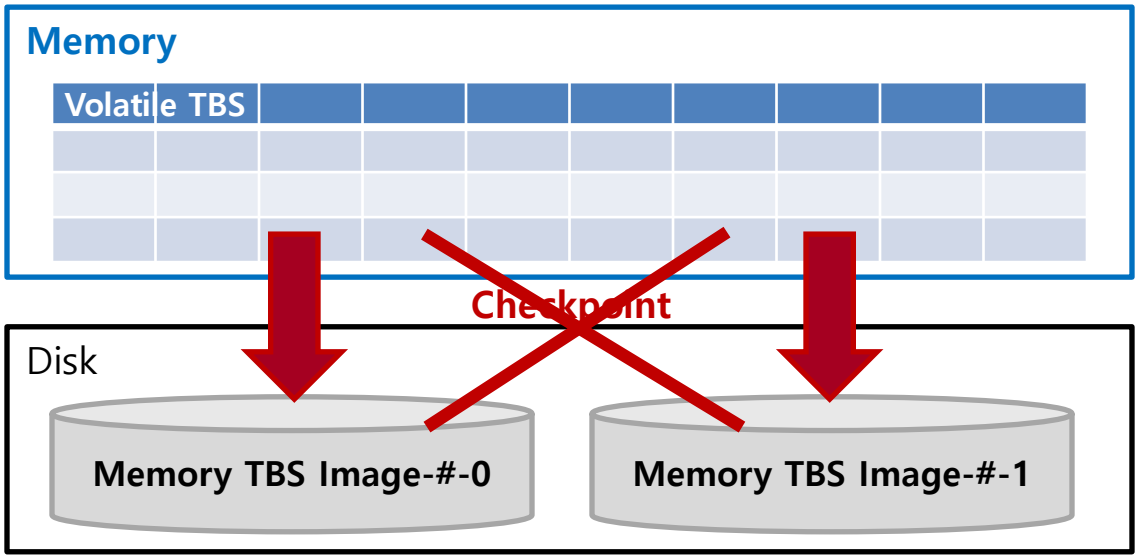
TABLESPACE MANAGEMENT

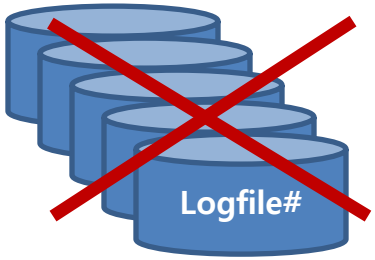
❖ 휘발성 테이블스페이스(VOLATILE TABLESPACE)

- ➢ 데이터가 메모리에만 상주
- ➢ 디스크에 체크포인트 이미지 파일을 가지지 않음

❖ 특징

- ➢ 디스크 로깅(logging)을 수행하지 않고 체크포인트 대상에서 제외
- ➢ 디스크 I/O가 없음
- ➢ 메모리 테이블스페이스와 비교하여 상대적으로 빠른 갱신 성능을 보장





TABLESPACE MANAGEMENT

※ 6.3.1 버전부터 제공

❖ 임시 테이블(TEMPORARY TABLE)

- 데이터를 일시적으로 보관하기 위해 사용하는 테이블
- 응용 프로그램에서 여러 개의 DML 작업을 실행할 때 생기는 결과 집합을 일시적으로 저장할 때 유용
- 임시 테이블의 데이터는 테이블에 데이터를 입력한 세션에서만 확인 가능
- 세션이나 TRANSACTION이 종료되면 임시 테이블은 자동으로 truncate
- 임시 테이블내의 데이터는 백업이나 시스템 장애 시 복구가 불가능

❖ 제약사항

- 파티셔닝 불가
- 외래키 지정 불가
- 휘발성 테이블스페이스에서만 저장 가능
- 분산 TRANSACTION 지원 불가

TABLESPACE MANAGEMENT

❖ 임시 테이블 생성 구문

```
CREATE [GLOBAL] TEMPORARY TABLE table_name
ON COMMIT {DELETE ¦ PRESERVE} ROWS ;
```

❖ 예제

➢ TRANSACTION에 한정되는 임시 테이블 생성 예제

```
iSQL> CREATE VOLATILE TABLESPACE my_vol_tbs
        2 SIZE 12M AUTOEXTEND ON MAXSIZE 1G;
Create success.
iSQL> CREATE TEMPORARY TABLE temp1 (c1 INTEGER, c2 VARCHAR(10))
         2 ON COMMIT DELETE ROWS
         3 TABLESPACE my_vol_tbs;
Create success.
```

❖ 임시 테이블 생성 구문 예제

➢ 세션에 한정되는 임시 테이블 생성 예제

```
iSQL> CREATE VOLATILE TABLESPACE my_vol_tbs
        2 SIZE 12M AUTOEXTEND ON MAXSIZE 1G;
Create success.
iSQL> CREATE TEMPORARY TABLE temp2 (c1 INTEGER, c2 VARCHAR(10) )
         2 ON COMMIT PRESERVE ROWS
         3 TABLESPACE my_vol_tbs;
Create success.
```

❖ 예제(ON COMMIT DELETE ROWS)

➢ 임시 테이블을 생성하고 데이터를 입력한 후, TRANSACTION이 종료되면 데이터가 삭제되는 예제

```
iSQL> AUTOCOMMIT OFF;
iSQL> CREATE VOLATILE TABLESPACE my_vol_tbs
        2 SIZE 12M AUTOEXTEND ON MAXSIZE 1G;
Create success.
iSQL> CREATE TEMPORARY TABLE t1 (c1 INTEGER, c2
VARCHAR(10) )
         2 ON COMMIT DELETE ROWS
         3 TABLESPACE my_vol_tbs;
Create success.
iSQL> INSERT INTO t1 VALUES (1, '20141201');
1 row inserted.
iSQL> INSERT INTO t1 VALUES (2, '20141202');
1 row inserted.
iSQL> INSERT INTO t1 VALUES (3, '20141203');
1 row inserted.
iSQL> SELECT * FROM t1;
C1          C2
------------------------------------
1           20141201
2           20141202
3           20141203
3 rows selected.
```

TRANSACTION에 한정되는 임시테이블(t1)

```
iSQL> COMMIT;
Commit success.

iSQL> SELECT * FROM t1;
C1          C2
------------------------
No rows selected.
```

Commit, Rollback 시 데이터 지워짐

❖ 예제(ON COMMIT PRESERVE ROWS)

➢ 한 세션에서 임시 테이블을 생성하고 데이터를 삽입한 후, 해당 세션에서만 데이터가 조회되고 다른 세션에서는 조회되지 않는 예제

```
iSQL> CREATE VOLATILE TABLESPACE my_vol_tbs
        2 SIZE 12M AUTOEXTEND ON MAXSIZE 1G;
Create success.
iSQL> CREATE TEMPORARY TABLE t2 (c1 INTEGER, c2 VARCHAR(10) )
         2 ON COMMIT PRESERVE ROWS
         3 TABLESPACE my_vol_tbs;
Create success.
iSQL> DESC t2;
[ TABLESPACE : MY_VOL_TBS ]
[ ATTRIBUTE ]
--------------------------------------------------------------------
NAME                                TYPE                    IS NULL
--------------------------------------------------------------------
C1                                  INTEGER                 FIXED
C2                                  VARCHAR(10)             FIXED
T2 has no index
T2 has no primary key
```

세션에 한정되는 임시테이블(t2)

TABLESPACE MANAGEMENT

```
iSQL> AUTOCOMMIT OFF;
iSQL> ALTER TABLE t2 ADD CONSTRAINTS  t2_pk  PRIMARY KEY (c1) ;
Alter success.
iSQL> INSERT  INTO  t2  VALUES (1, 'abc') ;
1 row inserted.
iSQL> INSERT  INTO  t2  VALUES (2, 'def') ;
1 row inserted.
iSQL> COMMIT ;
iSQL> SELECT * FROM t2;
C1          C2
---------------------------
1           abc
2           def
2 rows selected.
iSQL> CONNECT sys/manager;
Connect success.
iSQL> SELECT * FROM t2;
C1          C2
---------------------------
No rows selected.
```

해당 세션에서 조회 결과

다른 세션에서 조회 결과

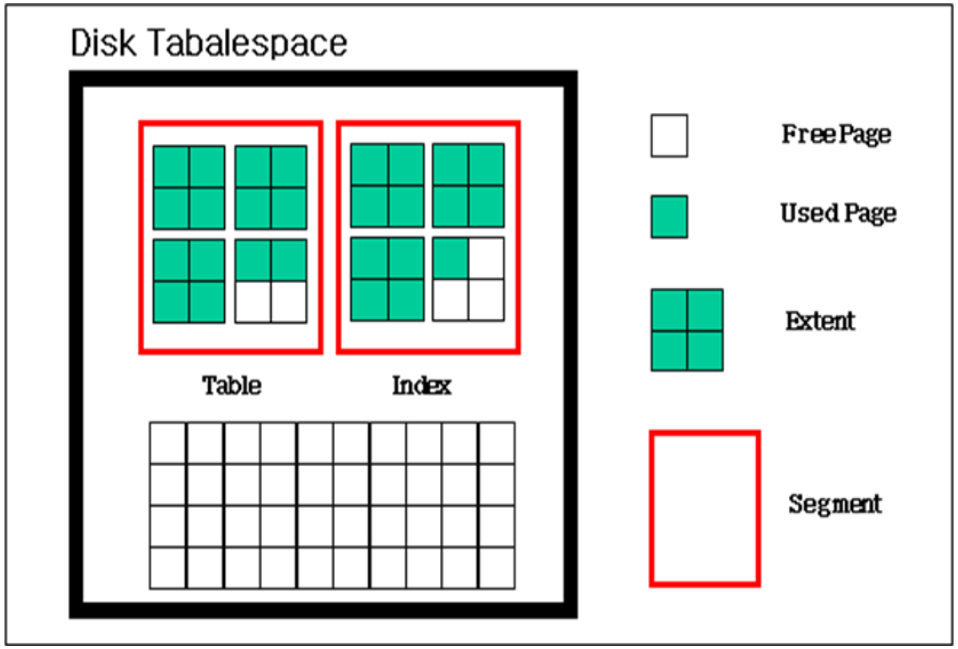
TABLESPACE MANAGEMENT

❖ 디스크 테이블스페이스(DISK TABLESPACE)

- ➢ 모든 데이터가 디스크 공간에 저장되는 테이블스페이스를 의미

❖ 구조

- ➢ 페이지(PAGE)
  - ▪ 테이블과 인덱스의 레코드를 저장하는 최소 단위
    - ◆ 디스크 페이지 : 기본 8KB
- ➢ 익스텐트(EXTENT)
  - ▪ 페이지 할당 단위
  - ▪ 데이터 저장 시 FREE 페이지가 부족하면 테이블스페이스로부터 할당 받음
  - ▪ 기본 값
    - ◆ 64개의 페이지(512KB)로 구성



TABLESPACE MANAGEMENT

❖ 디스크 테이블스페이스의 공간 할당

- ➢ 테이블에 512K Extent 단위로 공간 할당

❖ PAGE 상태

객체	Free	Used
테이블스페이스	특정 테이블에 할당되지 않은 공간 Extent 단위로 특정 테이블에 할당 가능	테이블에 할당한 공간 테이블에서 반납하기 전까지 다른 테이블에서 사용 불가
테이블	테이블이 할당 받은 공간 중 데이터가 들어 있지 않은 공간 해당 테이블 내에서 재사용 가능	테이블이 할당 받은 공간 중 데이터가 들어 있는 공간 데이터를 삭제하지 않으면 재사용 불가

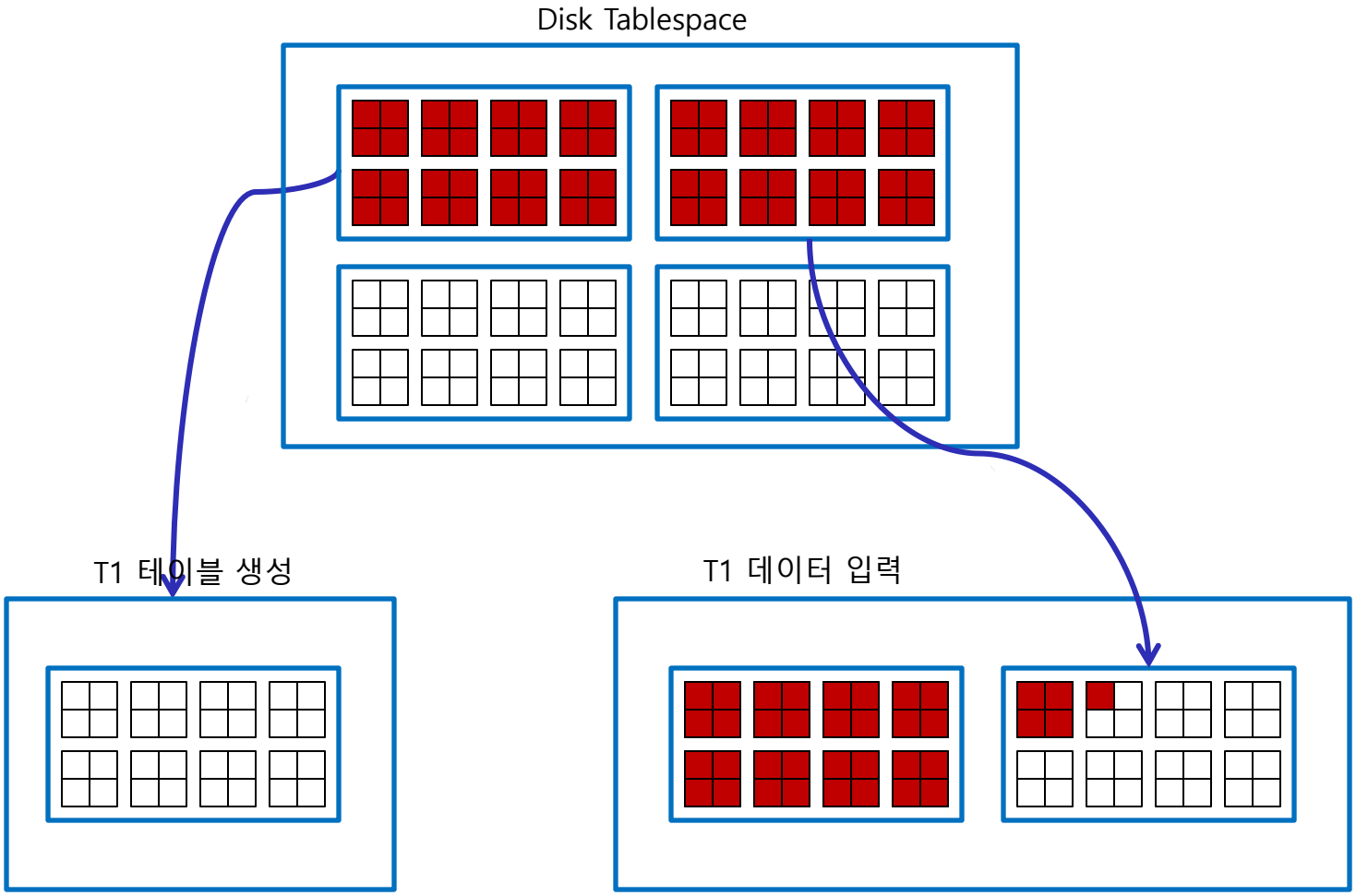
❖ 디스크 테이블 PAGE 상태 변화

- 디스크 테이블에 Delete 수행
  - 테이블 안에서 Page 상태가 Used → Free 로 전환(해당 테이블 내에서 재사용 가능)
  - 테이블스페이스로 Page를 반환하지 않음
- 디스크 테이블에 Truncate 수행
  - 테이블에 할당 되었던 Page를 테이블스페이스에 반환
  - 테이블에 할당되었던 테이블스페이스의 Page 상태는 Used → Free로 전환(다른 테이블에서 할당 받아 사용 가능)
- 디스크 테이블에 Move 수행
  - Move 구문을 통해서 데이터를 다른 테이블로 이동시켜도 Delete 한 것과 동일하게 해당 테이블 안에서만 재사용 가능

※디스크 테이블에는 Compaction 기능 없음

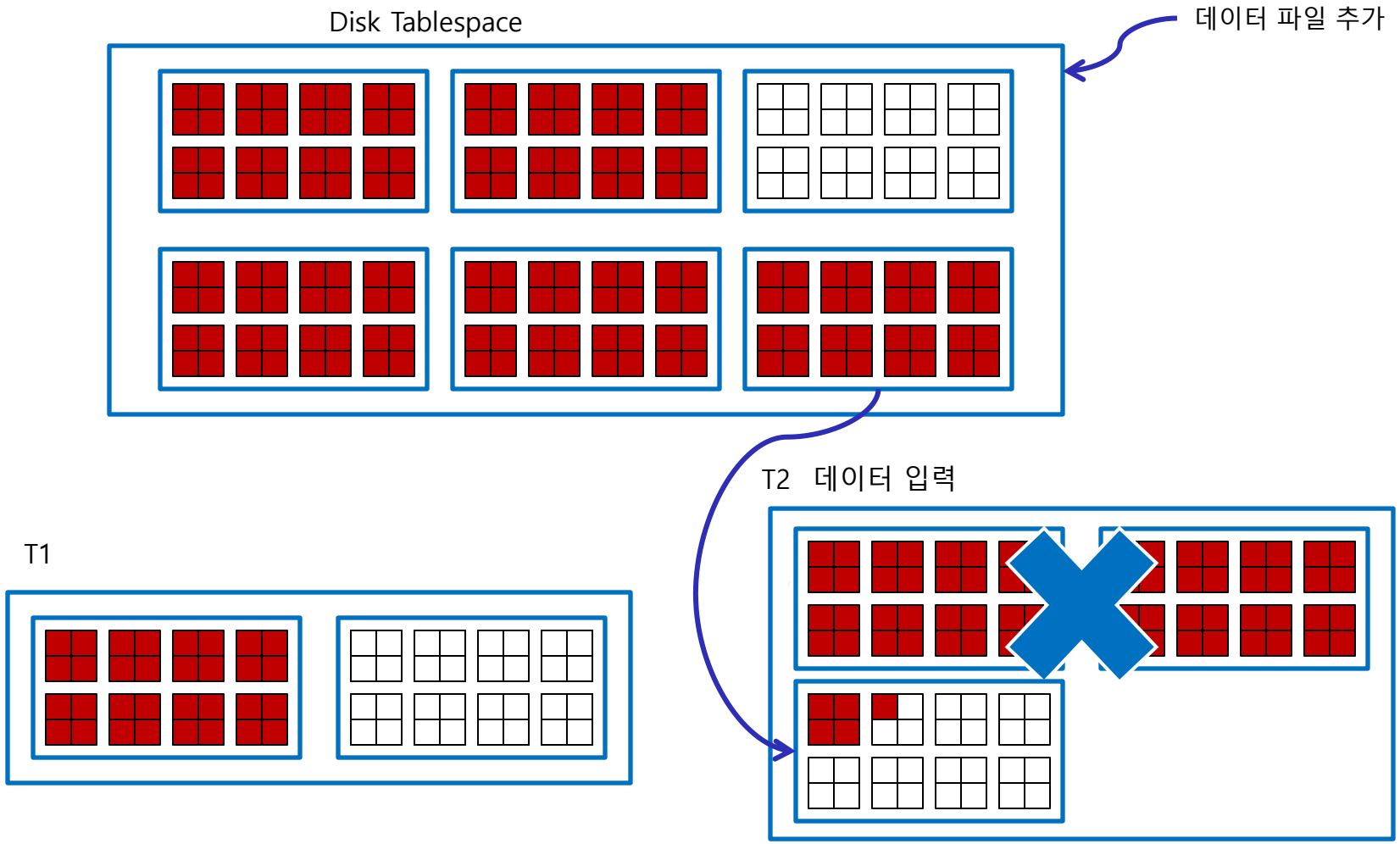
TABLESPACE MANAGEMENT

❖ 디스크 테이블스페이스 공간 할당 구조



TABLESPACE MANAGEMENT

❖ 디스크 테이블스페이스 공간 반납 구조



TABLESPACE MANAGEMENT

❖ 테이블스페이스 관련 성능 뷰

이름	내용
V$TABLESPACES	전체 테이블스페이스에 대한 정보
V$DATAFILES	디스크 테이블스페이스의 데이터파일에 대한 정보 데이터파일의 사용량을 확인하고 싶을 때 사용
V$MEM_TABLESPACES	메모리 테이블스페이스의 사용량을 확인하고 싶을 때 사용
V$VOL_TABLESPACES	휘발성 테이블스페이스의 사용량을 확인하고 싶을 때 사용
V$MEMTBL_INFO	어느 메모리 테이블이 메모리를 많이 사용하는지 확인하고 싶을 때 사용
V$DISKTBL_INFO	어느 디스크 테이블이 디스크를 많이 사용하는지 확인하고 싶을 때 사용

TABLESPACE MANAGEMENT

❖ 메모리 테이블스페이스

- 데이터를 메모리에 저장하여, 모든 TRANSACTION를 메모리 상에서 처리
- 체크포인트 시에 물리적인 파일(checkpoint image file)에 저장
- 데이터베이스 구동 시에 모든 데이터를 하드디스크에 저장된 물리적인 파일로부터 읽어서 메모리로 업로드 하여 구동

❖ 메모리 테이블스페이스 생성 구문(기본)

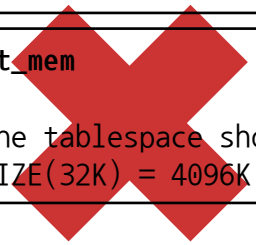
```
CREATE MEMORY [DATA] TABLESPACE tablespace_name
SIZE size (K | M | G) ;
```

❖ 메모리 테이블스페이스 생성 예제(기본)

- 초기 사이즈가 512M인 메모리 테이블스페이스 생성 예제

```
iSQL> CREATE MEMORY TABLESPACE test_mem
        2 SIZE 512M ;
Create success.
```

```
iSQL> CREATE MEMORY TABLESPACE test_mem
        2 SIZE 510M ;
[ERR-110EE : The initial size of the tablespace should be multiple of expand chunk size
( EXPAND_CHUNK_PAGE_COUNT * PAGE_SIZE(32K) = 4096K )]
```



메모리 테이블스페이스는 기본적으로 4M 단위로 생성 및 확장 가능함

TABLESPACE MANAGEMENT

❖ 메모리 테이블스페이스 생성 구문(자동확장 추가)

```
CREATE MEMORY [DATA] TABLESPACE tablespace_name
SIZE size (K ¦ M ¦ G)
[AUTOEXTEND [ON [NEXT size] [MAXSIZE size] ¦ OFF) ] ;
```

❖ 메모리 테이블스페이스 생성 예제(자동확장 추가)

➢ 초기 사이즈가 512M이고, 128M 단위로 자동 확장 가능한 최대 크기가 2G 인 메모리 테이블스페이스 생성 예제

```
iSQL> CREATE MEMORY TABLESPACE test_mem
        2 SIZE 512M
        3 AUTOEXTEND ON NEXT 128M MAXSIZE 2G;
Create success.
```

➢ 초기 사이즈가 512M 이고, 자동확장을 하지 않는 메모리 테이블스페이스 생성 예제

```
iSQL> CREATE MEMORY TABLESPACE test_mem
        2 SIZE 512M
        3 AUTOEXTEND OFF;
Create success.
```

TABLESPACE MANAGEMENT

❖ 메모리 테이블스페이스 생성 구문(체크포인트 경로 추가)

```
CREATE MEMORY [DATA] TABLESPACE tablespace_name
SIZE size (K ¦ M ¦ G)
[AUTOEXTEND [ON [NEXT size] [MAXSIZE size] ¦ OFF) ]
[CHECKPOINT PATH 'path' [SPLIT EACH size]] ;
```

❖ 메모리 테이블스페이스 생성 예제 (체크포인트 경로 추가)

➢ 초기 사이즈가 512M이고, 최대 1G까지 128M 단위로 자동 확장 가능한 메모리 테이블스페이스 생성 (체크포인트 이미지 파일은 다중화를 위해 3개의 디렉토리에 나누어 저장) 예제

```
iSQL> CREATE MEMORY TABLESPACE test_mem
         2 SIZE 512M
         3 AUTOEXTEND ON NEXT 128M MAXSIZE 1G
         4 CHECKPOINT PATH ' /dbs/path1 ' , ' /dbs/path2 ' , ' /dbs/path3 ' ;
Create success.
```

```
iSQL> CREATE MEMORY TABLESPACE test_mem
         2 SIZE 512M
         3 AUTOEXTEND ON NEXT 128M MAXSIZE 1G
         4 CHECKPOINT PATH ' /dbs/path1 ' , ' /dbs/path2 ' , ' /dbs/path3 '
         5 SPLIT EACH 256M ;
Create success.
```

TABLESPACE MANAGEMENT

❖ 휘발성 테이블스페이스

- 메모리 테이블스페이스와 동일한 구조의 테이블스페이스
- 체크포인트를 하지 않고, 리두 로그 기록하지 않음

❖ 휘발성 테이블스페이스 생성 구문

```
CREATE VOLATILE [DATA] TABLESPACE tablespace_name
SIZE size (K | M | G)
[AUTOEXTEND [ON [NEXT size][MAXSIZE size] | OFF) ] ;
```

❖ 휘발성 테이블스페이스 생성 예제

- 초기 사이즈가 512M이고, 최대 1G까지 128M 단위로 자동 확장 가능한 휘발성 테이블스페이스 생성 예제

```
iSQL> CREATE VOLATILE DATA  TABLESPACE test_mem
        2 SIZE 512M
        3 AUTOEXTEND ON NEXT 128M MAXSIZE 1G ;
Create success.
```

TABLESPACE MANAGEMENT

❖ 디스크 테이블스페이스

- ➢ 모든 데이터가 디스크에 저장되는 테이블스페이스
- ➢ 물리적으로 데이터 파일로 구성되고, 논리적으로 세그먼트, 익스텐트, 페이지로 구성

❖ 디스크 테이블스페이스 생성 구문(기본)

```
CREATE [DISK] [DATA] TABLESPACE tablespace_name
DATAFILE ' datafile_name ' ;
```

❖ 디스크 테이블스페이스 생성 예제(기본)

- ➢ 기본 경로에 데이터 파일 test01.dbf 를 생성하는 test_disk 테이블스페이스 생성 예제

```
iSQL> CREATE TABLESPACE test
         2 DATAFILE 'test01.dbf';
Create success.
```

TABLESPACE MANAGEMENT

❖ 디스크 테이블스페이스 생성 구문(자동확장 추가)

```
CREATE [DISK] [DATA] TABLESPACE tablespace_name
DATAFILE ' datafile_name '
[SIZE size (K | M | G) ] [REUSE]
[AUTOEXTEND [ON [NEXT size][MAXSIZE size] | OFF)];
```

❖ 디스크 테이블스페이스 생성 예제(자동확장 추가)

➢ 데이터 파일 test01.dbf, test02.dbf, test03.dbf 로 구성된 100MB의 test_disk 테이블스페이스 생성(자동확장 하지 않음) 예제

```
iSQL> CREATE TABLESPACE test_disk
      2 DATAFILE ' test01.dbf ' , ' test02.dbf ' ,  ' test03.dbf '
      3 SIZE 100M AUTOEXTEND OFF;
Create success.
```

➢ 데이터 파일 test01.dbf, test02.dbf, test03.dbf 로 구성되고, 초기크기가 100MB, 2G까지 자동 확장하는 test_disk 테이블스페이스 생성 예제

```
iSQL> CREATE TABLESPACE test_disk
      2 DATAFILE ' test01.dbf ' , ' test02.dbf ' ,  ' test03.dbf '
      3 SIZE 100M AUTOEXTEND ON NEXT 10M MAXSIZE 2G ;
Create success.
```

TABLESPACE MANAGEMENT

❖ 임시 테이블스페이스

- ➢ 디스크 데이터에 대한 SQL 수행 중 생성되는 임시 결과를 저장하기 위한 테이블스페이스
- ➢ TRANSACTION이 종료하는 시점에 해당 SQL이 남긴 모든 데이터들은 삭제

❖ 임시 테이블스페이스 생성 구문

```
CREATE TEMPORARY TABLESPACE tablespace_name
TEMPFILE ' tempfile_name '
[SIZE size (K | M | G) ] [REUSE]
[AUTOEXTEND [ON [NEXT size][MAXSIZE size] | OFF) ] ;
```

❖ 임시 테이블스페이스 생성 예제

- ➢ tbs.temp로 구성된 test_temp 임시 테이블스페이스 생성(임시 파일의 크기는 10M 이고, 5M 크기로 자동 확장) 예제

```
iSQL> CREATE TEMPORARY TABLESPACE test_temp
        2 TEMPFILE 'tbs.temp '
        3 SIZE 10M AUTOEXTENED ON NEXT 5M ;
Create success.
```

TABLESPACE MANAGEMENT

❖ 테이블스페이스 변경(디스크)

- ALTER TABLESPACE 구문으로 테이블스페이스에 데이터파일 추가/삭제, 데이터 파일 크기, 데이터 파일 이름 등에 대해서 변경 가능

❖ 테이블스페이스 변경 구문(디스크)

```
ALTER TABLESPACE tablespace_name
{  [ ADD ¦ DROP ] [ DATAFILE ¦ TEMPFILE ] …
   [ ALTER [ DATAFILE ¦ TEMPFILE ] file_name SIZE ]
   [ AUTOEXTEND [ON [NEXT size][MAXSIZE size] ¦ OFF) ]
   [RENAME DATAFILE ‘기존 데이터 파일 경로 및 이름’ TO ‘새로운 데이터 파일 경로 및 이름’]
};
```

❖ 테이블스페이스 변경 예제(디스크)

- test_disk 테이블스페이스에 64 MB의 데이터 파일 test04.dbf 추가(공간이 더 필요할 때는 512K 씩 파일이 자동확장) 예제

```
iSQL> ALTER TABLESPACE test_disk
         2 ADD DATAFILE ‘test04.dbf‘ SIZE 64M
         3 ALTER AUTOEXTEND ON NEXT 512K;
Alter success.
```

TABLESPACE MANAGEMENT

❖ 테이블스페이스 변경(메모리)

- ALTER TABLESPACE 구문으로 테이블스페이스에 체크포인트 경로 추가/삭제/변경, 테이블스페이스 크기 등에 대해서 변경 가능

❖ 테이블스페이스 변경 구문(메모리)

```
ALTER TABLESPACE tablespace_name
{  [ ADD ¦ DROP ] [ CHECKPOINT PATH  ‘ path ’ ] …
   [ RENAME CHECKPOINT PATH  ‘ 기존디렉토리 경로 ’  TO  ‘ 새로운 디렉토리 경로 ’ ]…
   [ ALTER  AUTOEXTEND ( ON [NEXT size][MAXSIZE size] ¦ OFF) ]
};
```

❖ 테이블스페이스 변경 예제(메모리)

- test_mem 테이블스페이스를 8M 단위로 최대 1GB 자동 확장 변경 예제

```
iSQL> ALTER TABLESPACE test_mem
         2 AUTOEXTEND ON NEXT 8M MAXSIZE 1G;
Alter success.
```

TABLESPACE MANAGEMENT

❖ 테이블스페이스 삭제

- 데이터베이스에서 테이블스페이스 제거
- 시스템 테이블스페이스는 삭제 불가능

❖ 테이블스페이스 삭제 구문

```
DROP TABLESPACE tablespace_name
[INCLUDING CONTENTS]
[ AND DATAFILES ¦ CASCADE CONSTRAINTS] ;
```

❖ 테이블스페이스 삭제 예제

- 메모리 테이블스페이스 test_mem 삭제 예제

```
iSQL> DROP TABLESPACE test_mem;
Drop success.
```

- 디스크 테이블스페이스 test_disk의 모든 객체, 데이터 파일들과 함께 테이블스페이스 삭제 예제

```
iSQL> DROP TABLESPACE test_disk
         2  INCLUDING CONTENTS AND DATAFILES;
Drop success.
```

TABLESPACE MANAGEMENT

❖ 테이블스페이스 상태

- 테이블스페이스는 서비스 상태에 따라 온라인(online), 오프라인(offline), 또는 폐기(discard) 상태로 구분

❖ 테이블스페이스 상태 변경 구문

```
ALTER TABLESPACE tablespace_name
[ONLINE / OFFLINE / DISCARD];
```

- **ONLINE**
  - 테이블스페이스에 속한 모든 객체에 사용자가 접근할 수 있는 일반적인 상태
- **OFFLINE**
  - 테이블스페이스와 관련된 모든 자원이 해제된 상태이며, 데이터베이스에서 테이블스페이스를 일시적으로 사용할 수 없게 설정된 상태
- **DISCARD**
  - 특정 테이블스페이스의 데이터에 오류가 발생하여 정상적인 데이터베이스구동이 불가능할 시 특정 테이블스페이스를 폐기할 수 있도록 설정된 상태

ARCHITECTURES

TRANSACTION MANAGEMENT

TRANSACTION MANGEMENT

❖ TRANSACTION 정의

- TRANSACTION이란 하나 이상의 SQL문장들로 이루어진 논리적인 작업 단위

❖ TRANSACTION 관리

- 사용자 TRANSACTION의 동시성을 제어하고 데이터 일관성을 유지 하도록 하는 데이터베이스의 가장 기본적인 기능 중 하나

❖ 정상적인 TRANSACTION 무결성 조건

- 원자성(Atomicity)
- 일관성(Consistency)
- 고립성(Isolation)
- 영속성(Durability)

TRANSACTION MANGEMENT

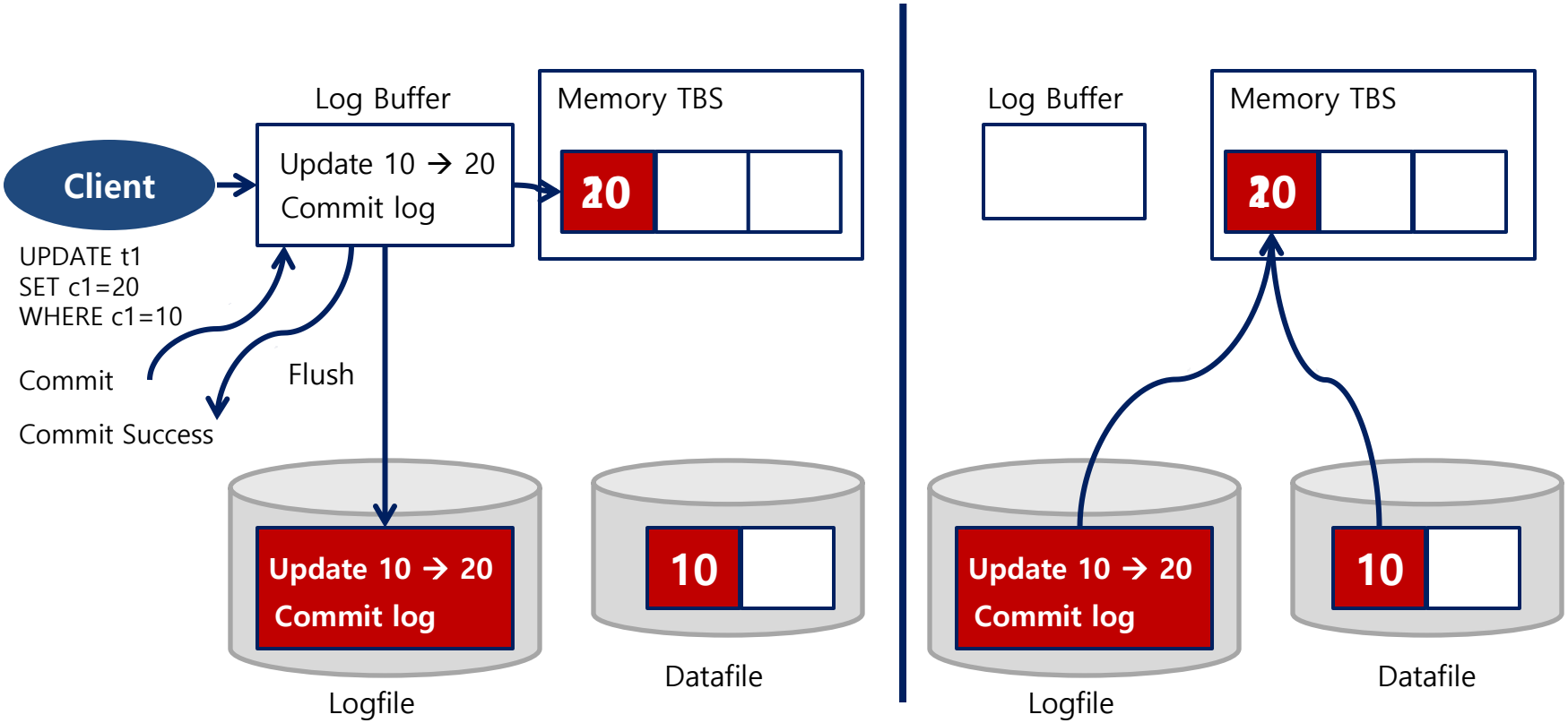
❖ TRANSACTION 영속성

- DBMS가 사용자에게 TRANSACTION 커밋(commit) 응답 했을 경우,

  데이터베이스 객체에 대한 해당 변경 사항이 디스크에 반영(flush) 되기 전에 시스템 장애가 발생하였더라도 해당 TRANSACTION의 커밋은 보장 되어야 한다는 속성
- 데이터베이스 관리 시스템은 TRANSACTION의 durability를 제공하기 위해 로그(log) 기록 관리
- 로그 기록으로 인한 디스크 I/O는 TRANSACTION 처리의 병목(bottleneck)으로 작용하게 되어 TRANSACTION 처리 성능 저하 원인
- 완벽한 TRANSACTION durability과 TRANSACTION 처리 성능 관계는 안정성과 성능이라는 각기 다른 목표를 가지는 상충(tradeoff) 관계

TRANSACTION MANGEMENT

❖ TRANSACTION 영속성과 복구



비정상 종료 발생 → SERVER Restart → Restart Recovery

TRANSACTION MANGEMENT

❖ DURABILITY LEVEL 범위

- Durability Level은 0~5 Level로 구분
- ALTIBASE는 0과 1 Level을 제외한 2에서 5 Level까지 지원
- 기본 Durability Level은 **3 Level**

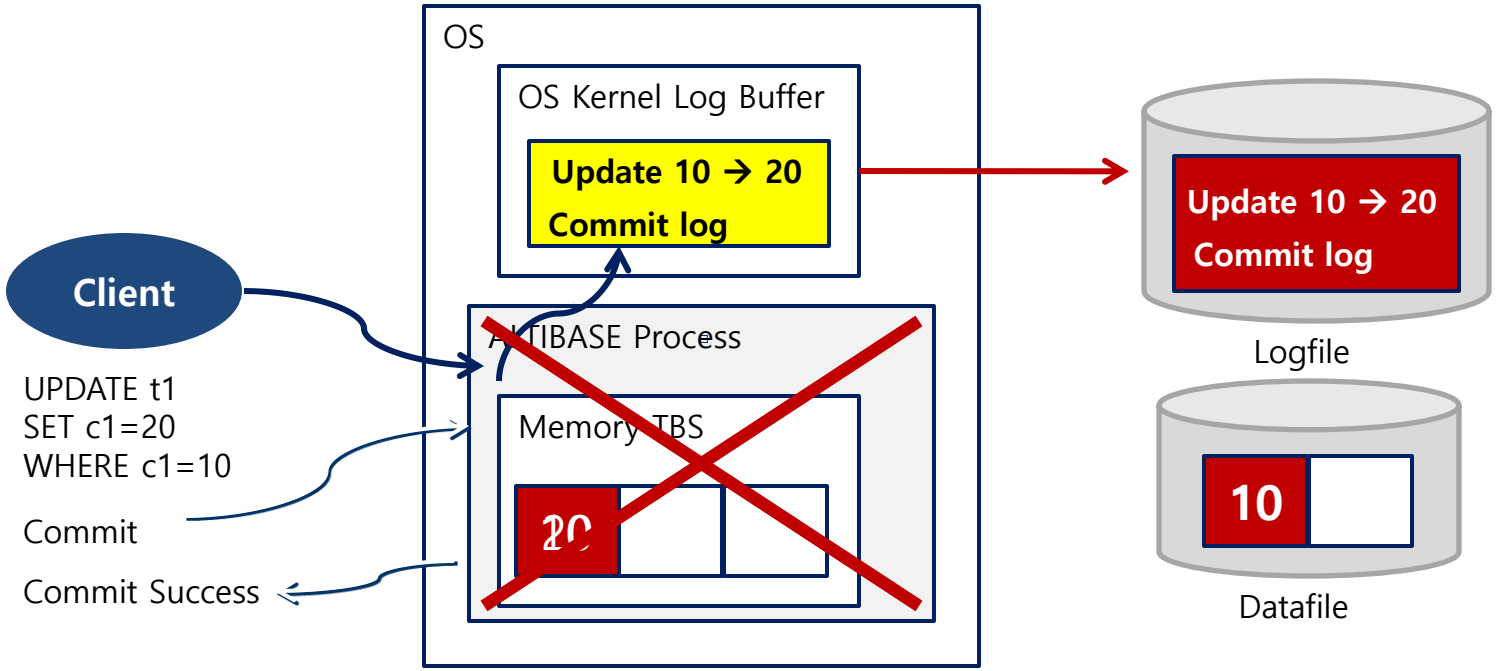
❖ DURABILITY LEVEL 설정 방법

- altibase.properties 파일에서 관련 프로퍼티의 변경 설정
  - COMMIT_WRITE_WAIT_MODE, LOG_BUFFER_TYPE 프로퍼티 설정
- 데이터베이스 구동 시 설정된 Durability Level 구동
- 데이터베이스 운영 중에 실시간으로 변경 불가능

TRANSACTION MANGEMENT

❖ DURABILITY LEVEL 3

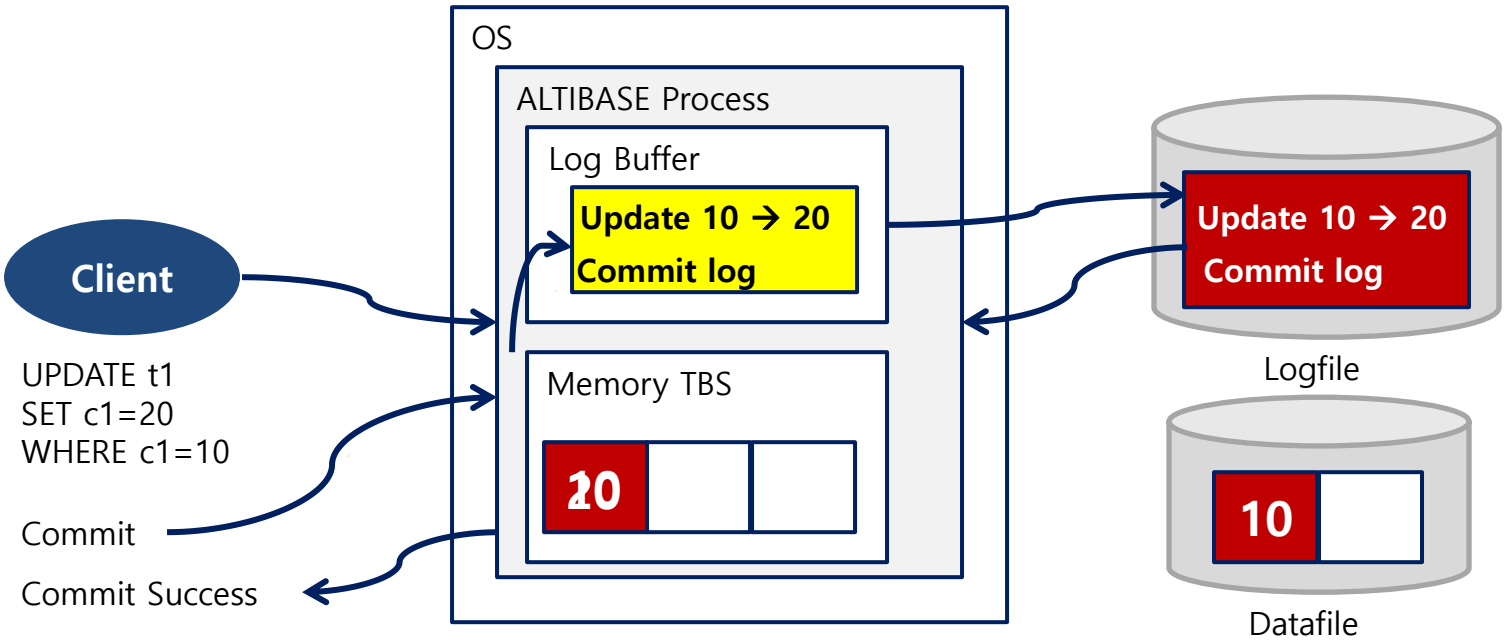
- OS Kernel 영역의 로그버퍼를 사용하기 때문에 ALTIBASE 프로세스가 비정상 종료를 하더라도 TRANSACTION이 commit한 로그는 운영체제의 의해 로그파일 반영
- ALTIBASE의 기본 durability level(성능지향 설정 방법)
- OS의 crash 상황만 아니라면 TRANSACTION durability 완벽 지원



TRANSACTION MANGEMENT

❖ DURABILITY LEVEL 5

- 로그를 프로세스 영역의 로그버퍼에 기록하고, 물리적인 로그파일에 기록하는 것을 보장하기 때문에 ALTIBASE의 장애 시에도 durability 보장
- 모든 로그가 로그파일에 반영됨을 보장하기 때문에 어떠한 시스템 장애 상황에서도 완벽하게 TRANSACTION durability를 보장하나 durability level중 성능이 가장 느림



TRANSACTION MANGEMENT

❖ 버전 관점의 동시성 제어기법 분류

- ➢ 동일 레코드에 수행되는 Read/Modify 연산에 대한 동시성 제어기법

❖ SVCC(SINGLE VERSION CONCURRENCY CONTROL)

- ➢ 레코드에 버전(version 또는 image)이 한 개만 존재하는 기법으로 검색 수행만으로도 다른 TRANSACTION에게 영향

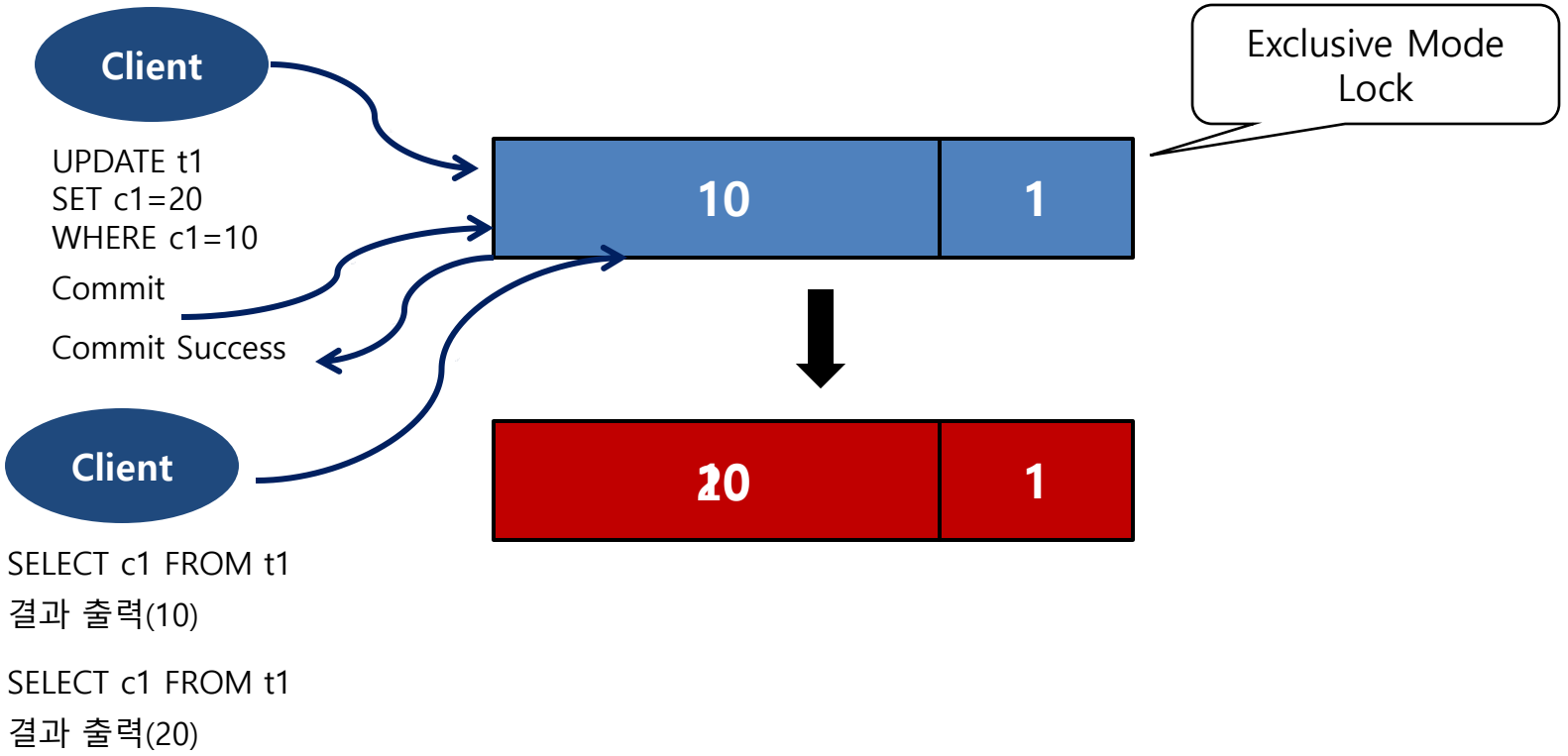
❖ MVCC(MULTI VERSION CONCURRENCY CONTROL)

- ➢ 하나의 레코드에 대해 변경이 발생할 경우 그 레코드의 원래 버전은 그대로 유지한 채로 새로운 버전을 만들어 그 새로운 버전에 대해 변경을 수행함으로써 비록 한 TRANSACTION이 동일 레코드에 대해 연산을 수행하고 있더라도 그 레코드를 검색하는 다른 TRANSACTION에게는 영향을 미치지 않도록 하는 동시성 제어 기법

TRANSACTION MANGEMENT

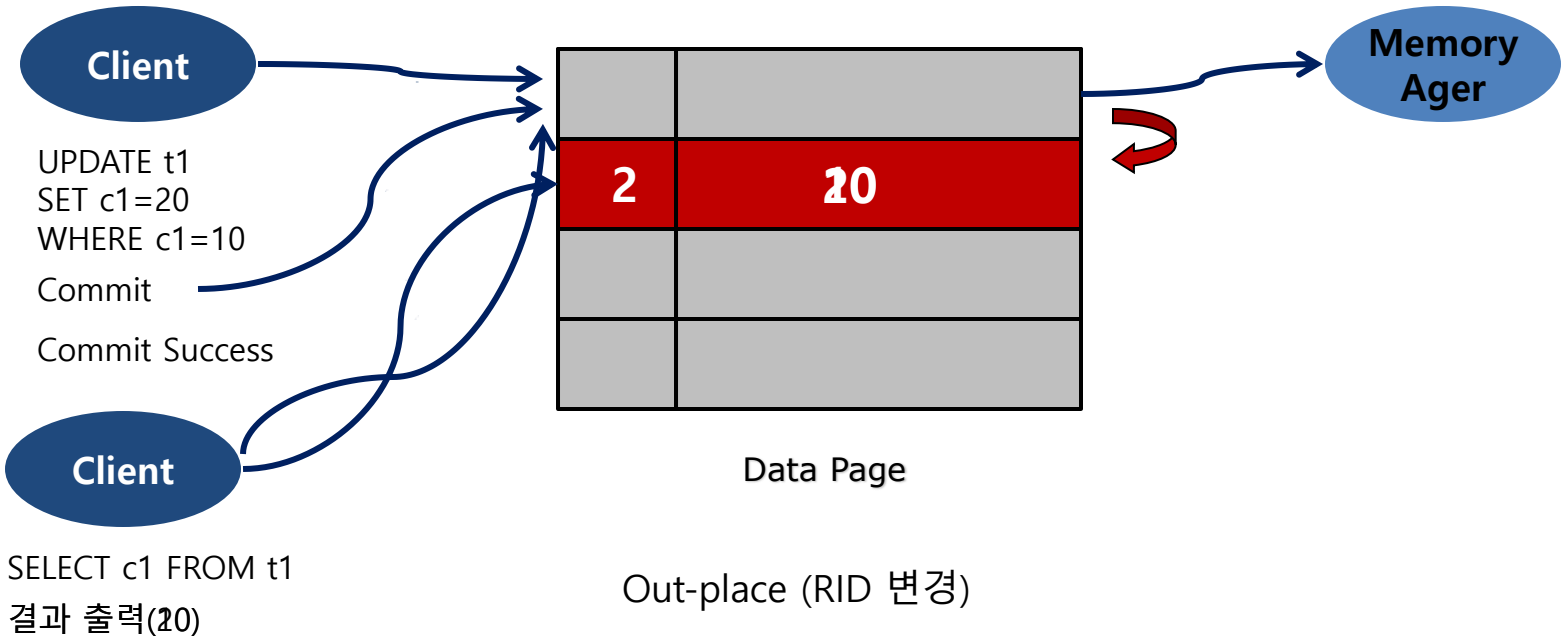
MVCC

- 변경 연산 시 Lock을 획득하고 새로운 버전을 생성하여 변경하기 때문에 Read/Modify 연산간에 충돌이 없고 복잡한 TRANSACTION 환경에서 높은 성능 보장



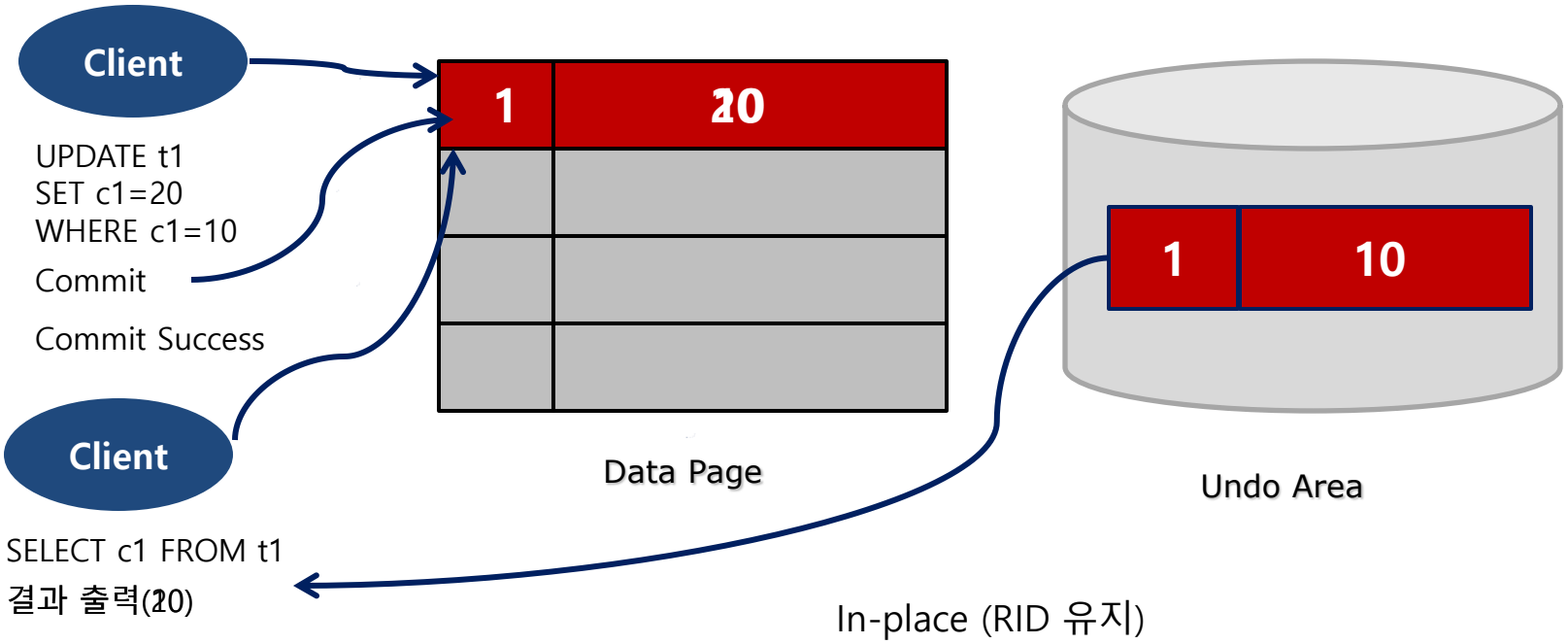
❖ 메모리 테이블 : OUT-PLACE

- 새로운 레코드 버전을 데이터 페이지의 별도 RID로 저장하는 Out-place 구조로 빠른 성능을 보장하나 데이터 페이지의 사용 효율성이 저하될 가능성이 있으므로 특정 레코드의 버전이 프로퍼티에 지정된 사이즈보다 커질 경우에는 In-place Update로 전환되도록 설계



❖ 디스크 테이블 : IN-PLACE

- 기존의 레코드에서 변경되는 컬럼들을 Undo 테이블스페이스에 Undo 로그 레코드로 기록하고 변경 이후 값들을 기존 레코드의 해당 위치에 복사하는 In-place 구조



ARCHITECTURES

BUFFER MANAGEMENT

BUFFER MANAGEMENT

❖ BUFFER 개요

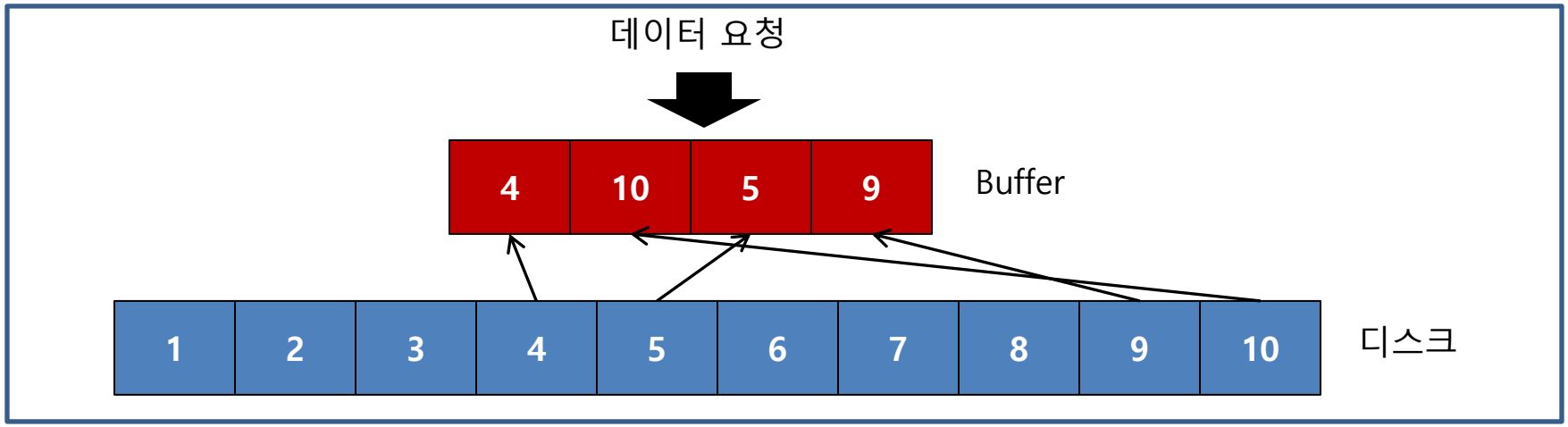
- ➢ 디스크 I/O를 줄이고 성능 향상을 위해서 디스크에 있는 페이지를 임시로 메모리에 적재하여 사용하는 메모리 영역

❖ BUFFER REPLACE

- ➢ 디스크의 모든 페이지를 버퍼에 적재할 수 없으므로 새로운 디스크의 페이지를 적재할 때, 버퍼의 일부 페이지를 새로운 페이지로 교체

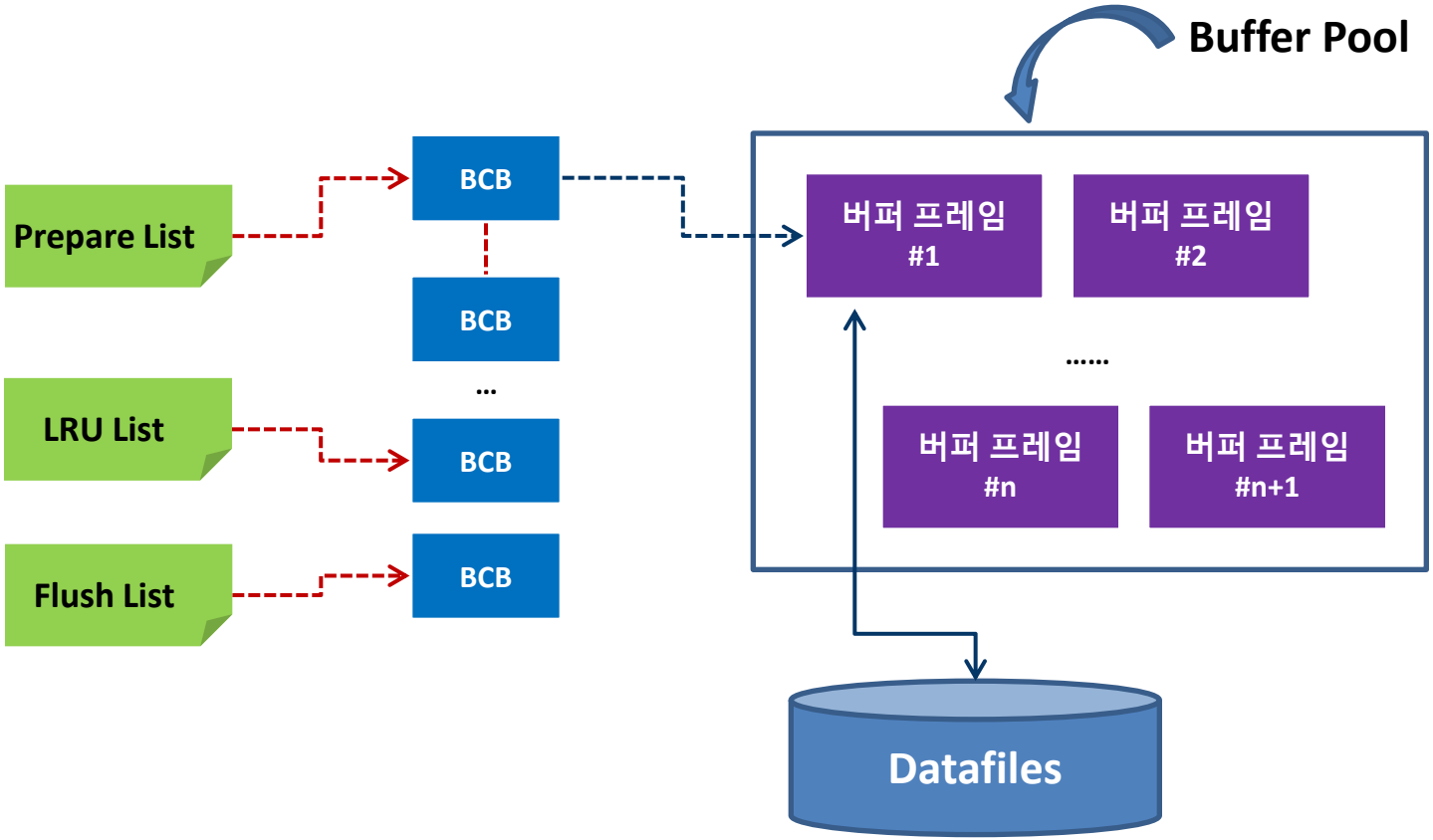
❖ VICTIM

- ➢ Buffer Replace를 위해 선정된 버퍼 프레임과 해당 BCB를 아울러 지칭



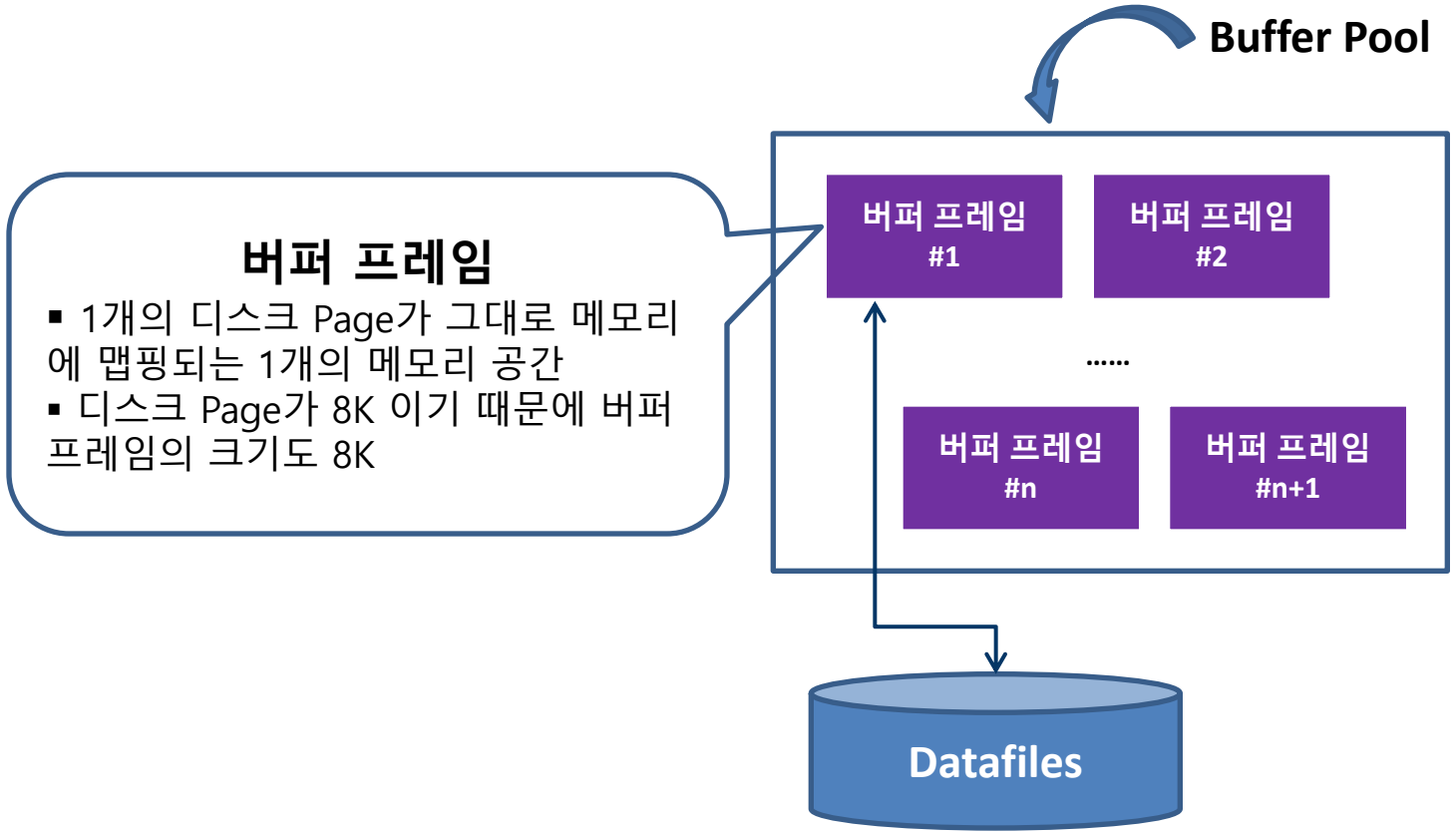
BUFFER MANAGEMENT

❖ BUFFER 구성요소



BUFFER MANAGEMENT

❖ BUFFER 구성요소



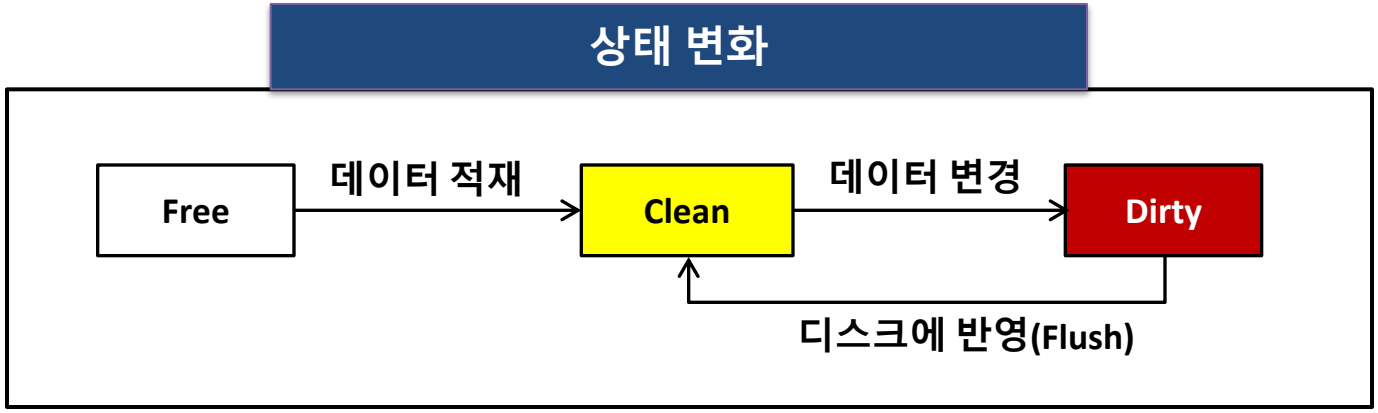
버퍼 프레임

- 1개의 디스크 Page가 그대로 메모리에 맵핑되는 1개의 메모리 공간
- 디스크 Page가 8K 이기 때문에 버퍼 프레임의 크기도 8K

BUFFER MANAGEMENT

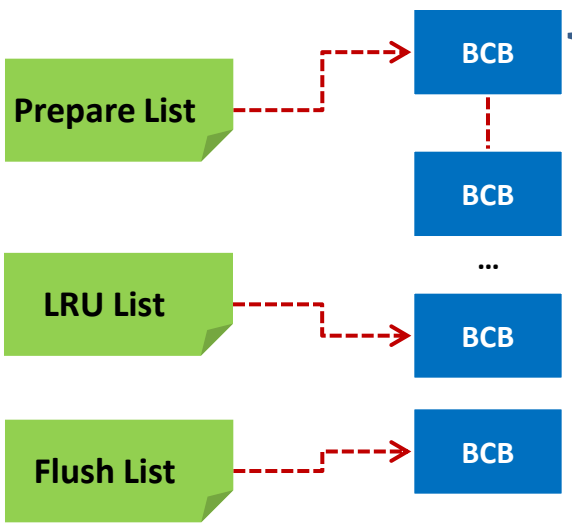
❖ BUFFER 구성요소

항목	설명
Free	최초 상태이며, 버퍼 프레임에 페이지가 적재되지 않은 상태
Clean	버퍼 프레임에 Page가 적재된 상태이며 변경이 발생하지 않은 상태
Dirty	버퍼 프레임에 적재된 Page에 변경이 발생한 상태



BUFFER MANAGEMENT

❖ BUFFER 구성요소



BCB(Buffer Control Block)

- 1개의 버퍼프레임에 대한 정보를 관리하는 하나의 구조체정보를 의미.
- 버퍼 프레임과 BCB는 1:1 관계임.
- BCB가 관리하는 항목은 다음과 같음

항목	설명
ID	BCB의 고유식별번호
Frame PTR	Buffer Frame의 주소값
Space ID	디스크 Page가 속한 테이블스페이스 ID
Page ID	디스크 데이터베이스의 Page 식별번호
State	버퍼 프레임의 상태정보
ListItem	BCB가 어떤 리스트에 위치해 있는지 표시
ListNO	병렬화된 리스트 중 BCB가 위치한 리스트 번호

BUFFER MANAGEMENT

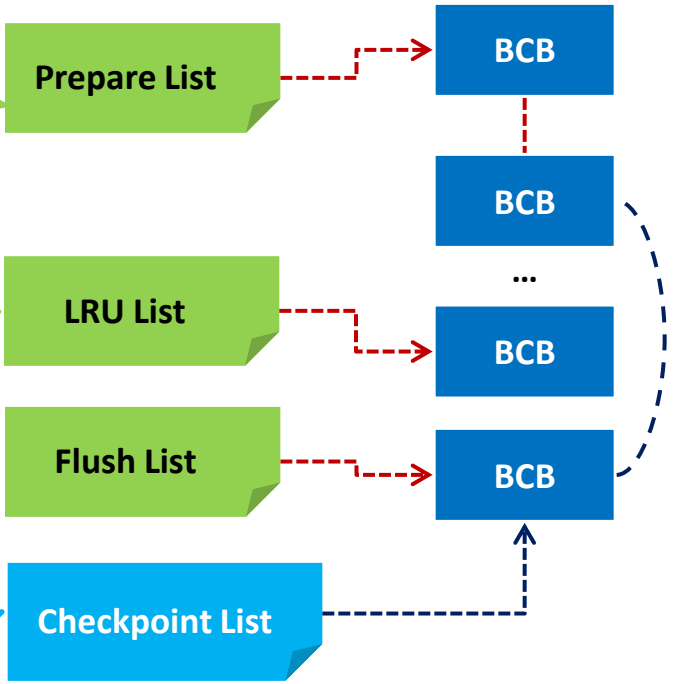
❖ BUFFER 구성요소

Prepare List
- 데이터베이스 구동 시에 Free 상태의 BCB가 위치하고 있는 리스트
- Victim 이 필요하면 가장 먼저 탐색함
- Flush가 발생하면, Buffer Replace를 위해 Flush list에서 Prepare List 로 이동함

LRU List
- LRU 알고리즘으로 BCB 들을 관리함
- 버퍼 프레임에 적재된 순서대로 BCB를 관리하여 오래된 BCB들부터 Victim 으로 선정하여 Replace 함

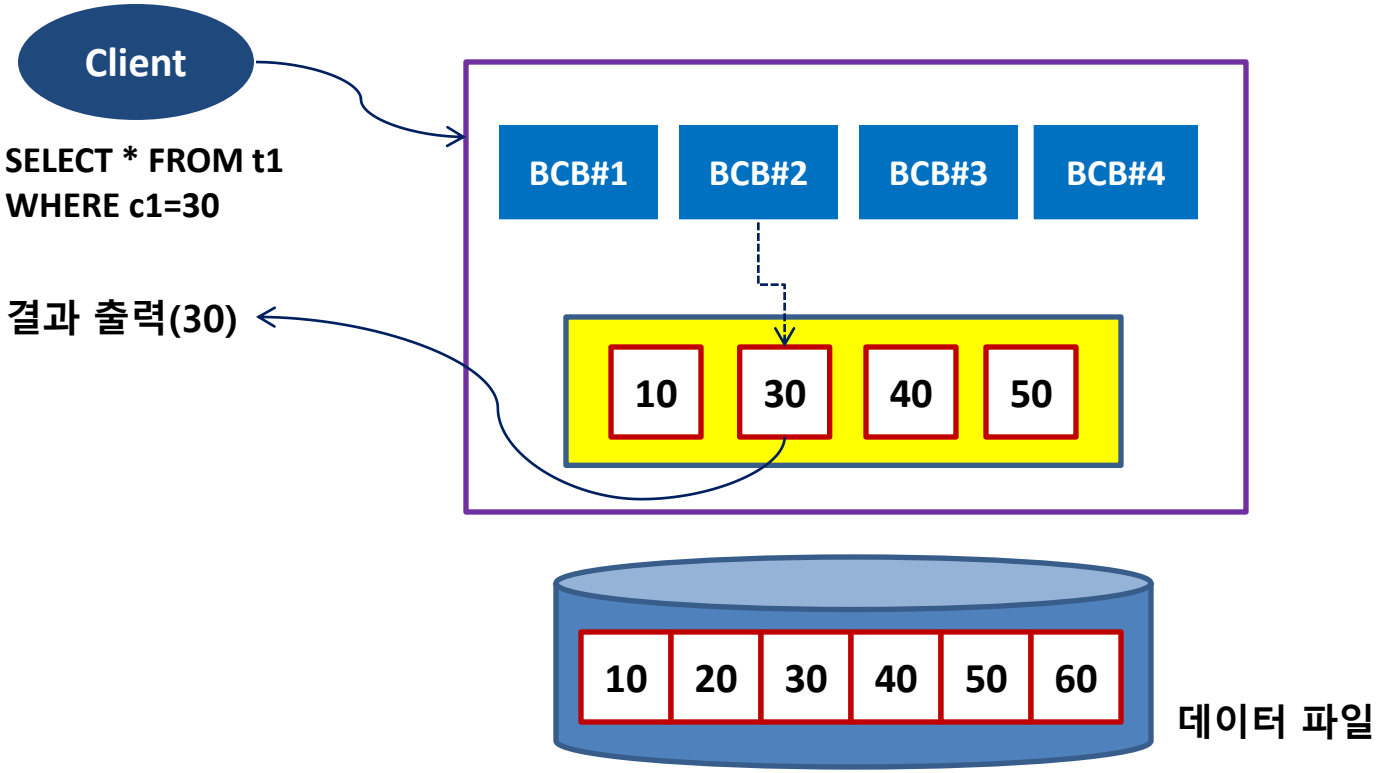
Flush List
- Dirty 상태의 BCB들을 관리하며, Buffer Flusher에 의해서 디스크 Page로 flush됨
- Flush 가 되면, BCB의 상태는 Clean으로 변경되고 Replace를 위해 Prepare List 로 이동됨

Checkpoint List
- 3개의 리스트에 위치한 Dirty 상태의 BCB 들을 관리하는 리스트
- 3개의 리스트와 독립적으로 BCB들을 관리함
- Buffer Flush 와는 별개의 동작으로 Dirty BCB 들을 데이터 파일에 저장함



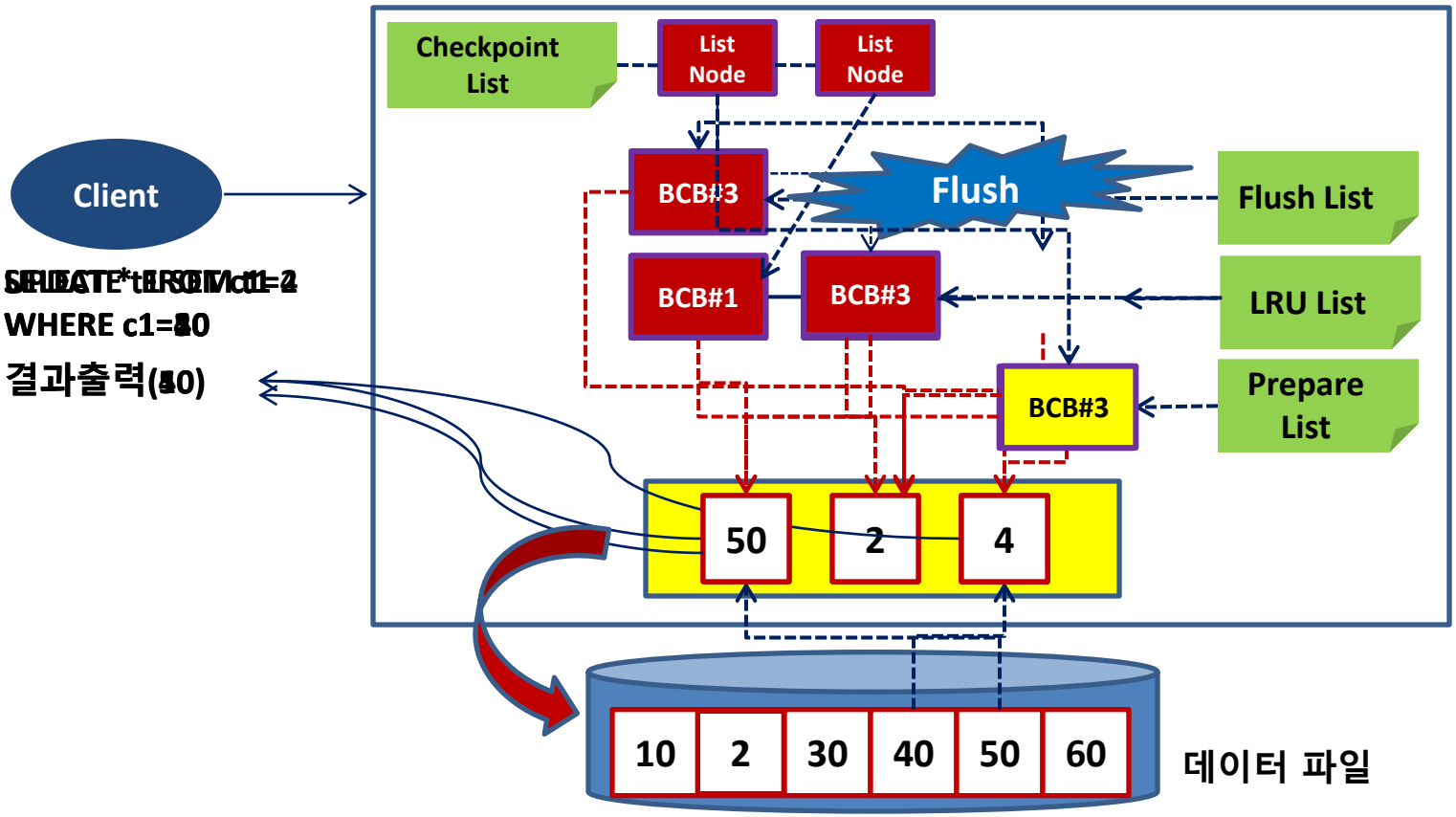
BUFFER MANAGEMENT

❖ BUFFER 구성요소



BUFFER MANAGEMENT

❖ BUFFER 구성요소



ALTIBASE ADMINISTRATION I

REPLICATION

❖ CONTENTS

REPLICATION OVERVIEW

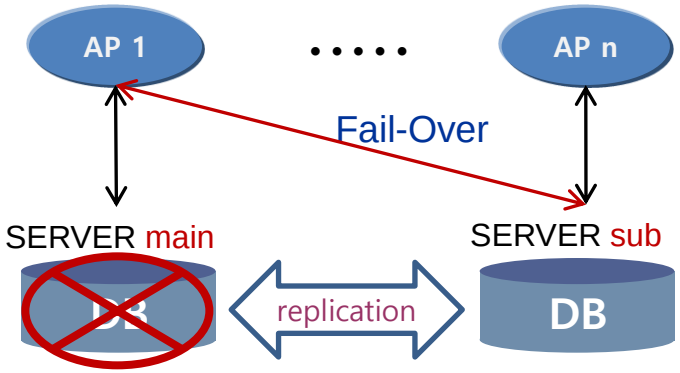
REPLICATION OVERVIEW

❖ 이중화 정의

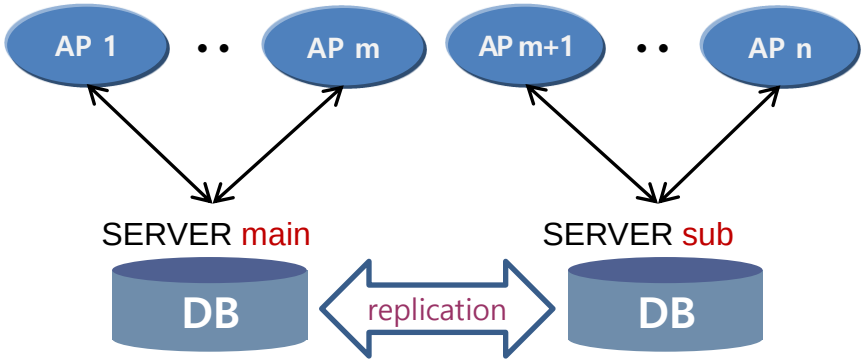
- 데이터베이스의 변경된 내용을 Network를 통하여 다른 데이터베이스로 복제하는 기술

❖ 이중화 목적

- 고가용성(High Availability) 확보
- 부하분산(Load-Balancing)을 통한 성능개선 및 확장성(scalability) 향상
- 물리적인 장애, 재해 시 데이터손실 최소화



[그림1. 2-way 이중화 시스템에서의 고가용성 확보]



[그림2. 2-way 이중화 시스템에서의 확장성 향상]

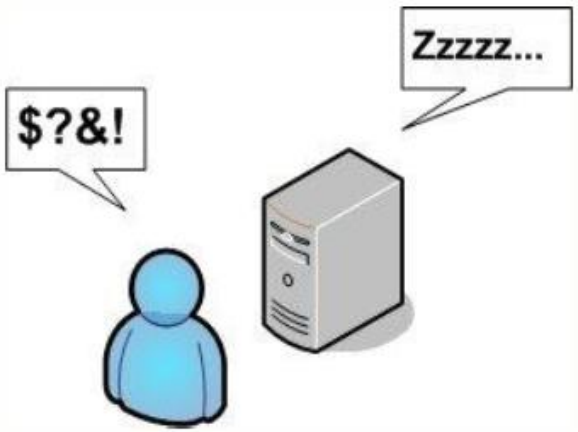
REPLICATION OVERVIEW

❖ 무정지 서비스 시스템

- Downtime을 최소화, 시스템 가용성을 100%로 유지
  - Planed Downtime - 정기점검, 시스템 upgrade/patch
    - Switch-Over 수행: 정상적인 서비스 주체 변경
  - Unplanned Downtime - 시스템 일부의 failure
    - Fail-Over 수행: 긴급 서비스 주체 변경

❖ 고가용성(HIGH AVAILABILITY / HA)

- 무정지 시스템 또는 무정지 시스템 구성요소의 가용성
- five 9 (99.999%)
- S/W, H/W 차원의 다양한 실현 기법 존재



❖ DBMS의 HA

- 노드 간 데이터베이스를 동기화함으로 실현
- 병렬 데이터베이스 ARCHITECTURE에 따라 기법 상이

REPLICATION OVERVIEW

❖ SHARED NOTHING VS SHARED DISK

구분	Shared Nothing ARCHITECTURE	Shared Disk ARCHITECTURE
공유자원	공유 자원이 없음	디스크
데이터동기화 방안	Network를 통한 복제	디스크를 공유
성능*	공유 자원이 없으므로 빠른 성능	공유자원에 대한 복잡한 처리(2PC/3PC)로 성능저하
시스템 구축비용*	저비용 (로컬디스크, Network)	고비용 (공유스토리지 설비)
거리적용*	일반적인 TCP 기반의 Network를 사용하여도 원거리 적용 가능	디스크 공유를 위한 고비용의 전용망이 요구되므로 실질적으로는 거리 제한 존재
데이터정합성*	Network 복제 특성상 노드 간 데이터불일치 현상을 억제하기 위한 별도의 고려 필요	디스크를 공유하므로 노드 간 데이터 정합성이 보장
치명적인 Failure 요소	Network Failure시 관련 노드 데이터동기화 불가	디스크 Failure시 시스템 전체 서비스 불가능
적합 시스템	**완벽한 데이터정합성보다는 빠른 성능 요구**	**성능보다는 완벽한 데이터 정합성 요구**
관련 대표 DBMS 기술	**이중화 (replication)**	**RAC (Real Application Cluster)**
관련 DBMS 벤더	ALTIBASE, DB2, MS-SQL, ORACLE, SYBASE	ORACLE

※ "성능, 시스템 구축비용, 거리적용"과 "데이터정합성" 측면에서의 trade-off

REPLICATION

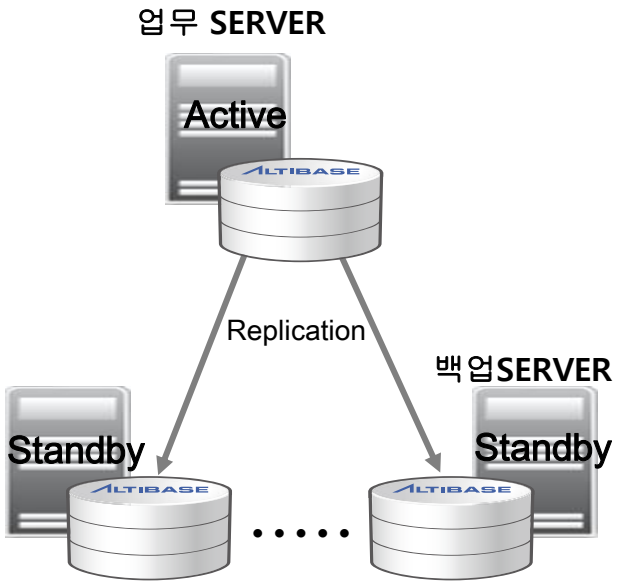
ALTIBASE REPLICATION

ALTIBASE REPLICATION

❖ ALTIBASE 이중화 주요 특징

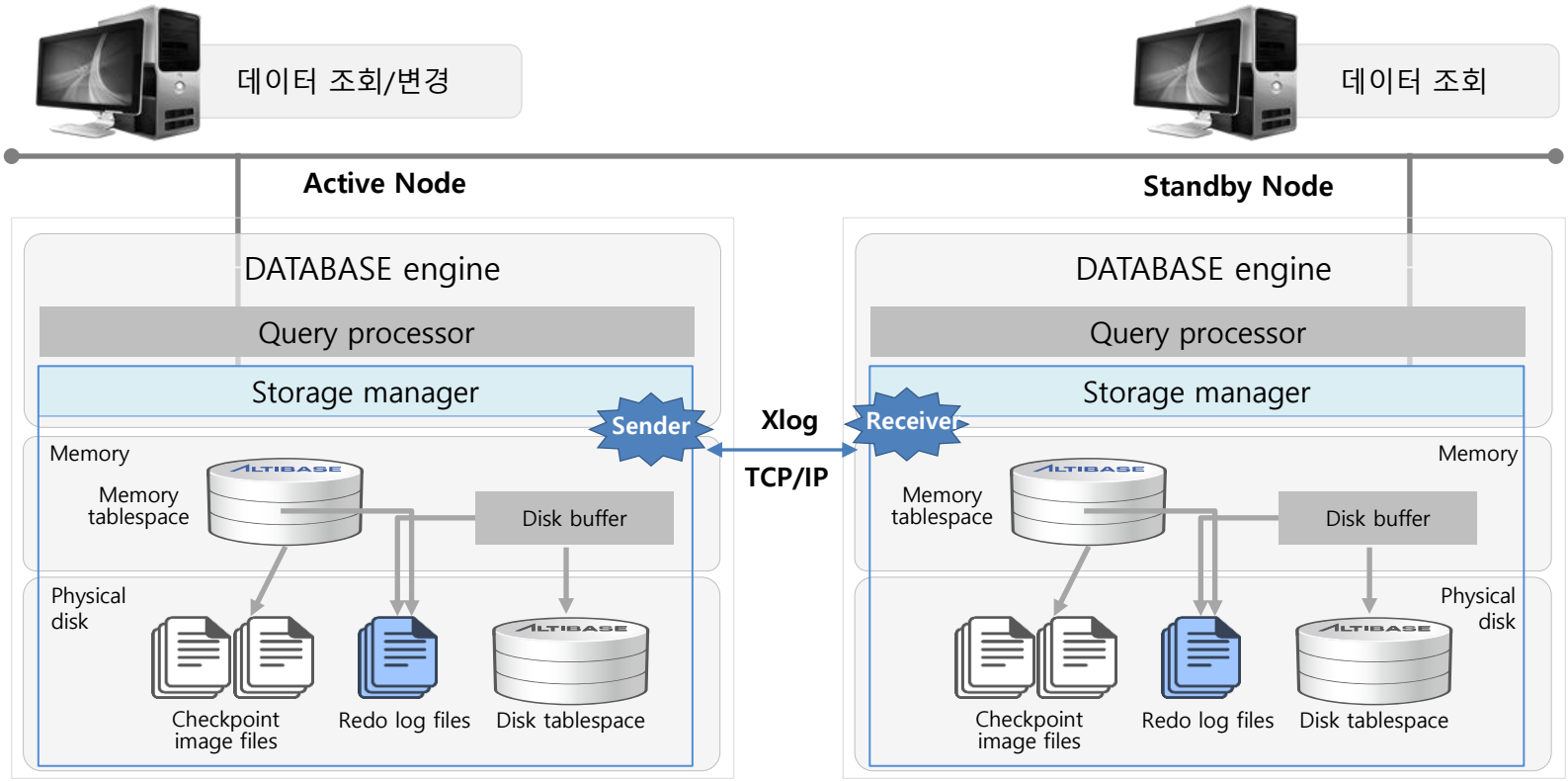
➢ ALTIBASE Replication은 백업 SERVER에 최신 데이터를 유지하며, SERVER 장애 시 백업 SERVER를 이용하여 서비스를 재개할 수 있는 무정지 운영 환경을 제공하는 것이 목적

- 안정성
  - 동일한 데이터 보유한 백업 SERVER로 무정지 시간 최소화
  - SERVER 장애 후 복구(Failback)에 대한 데이터 동기화 지원
  - Xlog 기반의 빠른 이중화 성능
- 확장성
  - 이중화 확장 가능(최대 10240)
  - 원거리 node에 대한 이중화 구성
  - 백업 SERVER를 "읽기 전용" 업무에 적용으로 부하 분산 효과
- 경제성
  - DBMS 내 Replication 모듈화로 이중화 구성 추가 비용 없음
  - 테이블 단위 이중화 수행으로 불필요한 자원 낭비 제거
  - Network 기반 이중화로 시스템 구축 비용 절감



ALTIBASE REPLICATION

❖ SHARED NOTHING 기반 고가용성



안정성

- 동일한 데이터 보유한 백업SERVER로 무정지시간 최소화
- SERVER 장애 후 복구(Failback)에 대한 데이터 동기화 지원
- XLog 기반의 빠른 이중화 성능

확장성

- 원거리 멀티 노드에 대한 이중화 구성
- 백업SERVER를 "읽기전용" 업무에 적용함으로써 부하 분산 효과

경제성

- DBMS 내 Replication 모듈화로 이중화 구성 추가 비용 없음
- 테이블 단위 이중화 수행에 따른 불필요한 자원 낭비 제거
- Network 기반의 이중화로 시스템 구축 비용 절감

ALTIBASE REPLICATION

❖ ALTIBASE 이중화 주요 특징

주요특징	상세설명
TCP/IP Network 기반	Network 설비만 되어 있다면 이중화가 가능하므로 이중화를 위한 별도 비용 없음 Network 성능에 따라 장거리 복제 가능 (Giga Bit LAN 필수)
이 기종 OS간 이중화 지원	OS bit, CPU endian 관계없이 이중화 가능
모듈화	이중화를 모듈화, DBMS와 유기적으로 결합된 형태이므로 이중화 성능 향상 이중화를 위한 별도의 ALTIBASE 패키지 불필요 사용자에 목적에 따라 ALTIBASE 를 가변적으로 사용 가능
리두로그 기반	리두로그를 실시간 전송하는 레코드 단위 이중화
테이블 단위 관리	이중화를 수행할 레코드를 테이블 단위로 관리 운영 중 이중화 대상 테이블을 실시간으로 추가, 삭제 가능
lazy, eager 복제방식 지원	복제 방식으로 lazy(비동기), eager (동기)모두 지원
고속복제	lazy 복제방식의 경우 마스터 TRANSACTION 수행속도에 영향을 주지 않으면서 마스터 TRANSACTION의(standalone) 95% 이상에 달하는 빠른 속도로 복제 가능 (Giga bit LAN 환경의 UNIX 시스템에서의 성능 테스트 측정치)
병렬 쓰레드 지원	Eager 모드 사용 시, 송신 쓰레드를 지정한 개수만큼 병렬로 사용하여 이중화 성능 향상
최대 10240-way 이중화 지원	하나의 ALTIBASE는 REPLICATION_MAX_COUNT에 설정 된 값 만큼 다중화 가능 업무에 따라 부하 분산 및 다양한 시스템 구성 가능
Point-To-Point 방식의 복제	이중화 노드간 1:1로만 복제, 다른 이중화 노드로 전이되지 않음

ALTIBASE REPLICATION

❖ ALTIBASE 이중화 주요 특징

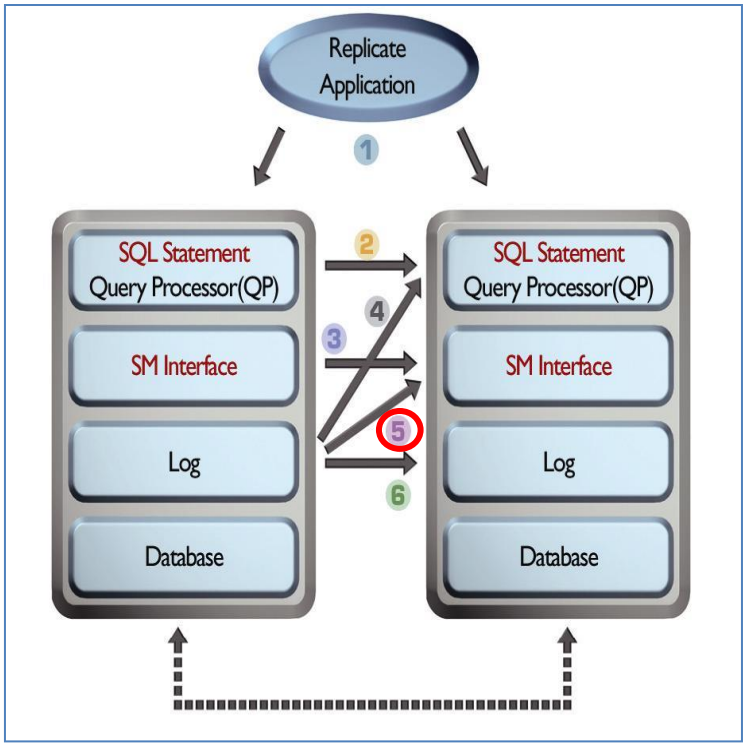
주요특징	상세설명
Network 장애 감지	물리적인 Network 손상을 감지하기 위한 별도 쓰레드 운영
자동회복	마지막으로 이중화를 수행하던 시점을 기록 및 유지 이중화 장애 시 이중화 재 수행만으로도 자동회복 가능
다중 IP 지원	하나의 이중화 대상 SERVER에 대해 두 개 이상의 물리적인 IP 주소를 부여 Network 장애 시 자동으로 전환함으로써 이중화 자체의 가용성 증대
SQL 인터페이스의 명령어	이중화 사용을 위한 모든 명령어를 SQL 인터페이스화하여 편의성 증대(alter ~)
4개 Conflict Resolution 제공	Conflict Resolution을 위한 3개의 Scheme 과 1개의 유틸리티 제공

ALTIBASE REPLICATION

❖ ALTIBASE 이중화 방식

- 이중화를 구현할 수 있는 다양한 방법
- ALTIBASE 이중화는 성능과 유연성을 고려하여 **"5. (리두) 로그를 직접 실행 가능한 논리적 구조로 변환"**하는 방법 사용

1. 응용프로그램 처리
   - 응용프로그램 작성 및 데이터정합성 보장 곤란
2. SQL 전송
   - QP 부하 가중 및 이중화 충돌 감지 곤란
3. SQL에 대한 실행계획 전송
   - 전송량 증가로 인한 통신 부하 가중
4. 리두로그 전송 후 SQL로 변환
   - SQL 변환 비용 및 QP 부하 가중
5. **리두로그를 SM에서 직접 실행이 가능한 논리적 구조로 치환하여 전송**
   - 치환 비용은 발생하나 이중화 성능 빠름
6. 리두로그 전송 후 회복 방법으로 반영
   - 속도는 빠르나, Active-Active 불가

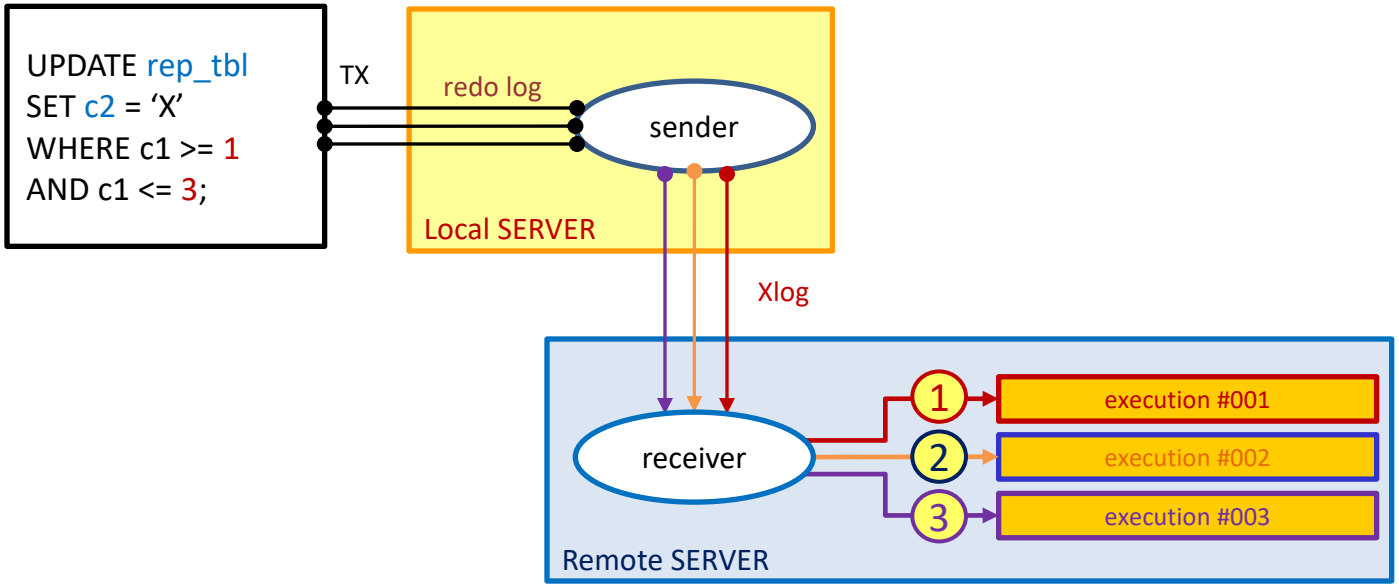


ALTIBASE 이중화 ARCHITECTURE

- Xlog
  - SM에서 직접 실행(execution) 가능한 가능한 논리적인 구조
    - 리두로그에서 이중화 수행에 필요한 부분만 추출한 플랫폼 중립의 이중화 로그
  - 기본적으로 하나의 레코드에 대응하는 리두로그는 하나의 Xlog로 치환되어 실시간 전송
    - 레코드가 클 경우에는 여러 개로 분할하여 치환
- 이중화 송/수신 쓰레드
  - 이중화 송신 쓰레드(sender)
    - 리두로그를 Xlog로 치환 후 이중화 대상 SERVER로 전송
  - 이중화 수신 쓰레드(receiver)
    - 전송 받은 Xlog를 SM에 통하여 실행
- XSN (Xlog Sequence Number)
  - sender가 receiver에게 최종 전송한 리두로그 위치
  - 이중화가 중지되었다가 다시 시작할 때 이중화 시작 위치가 됨

SQL 구문 관점의 상세 이중화 과정

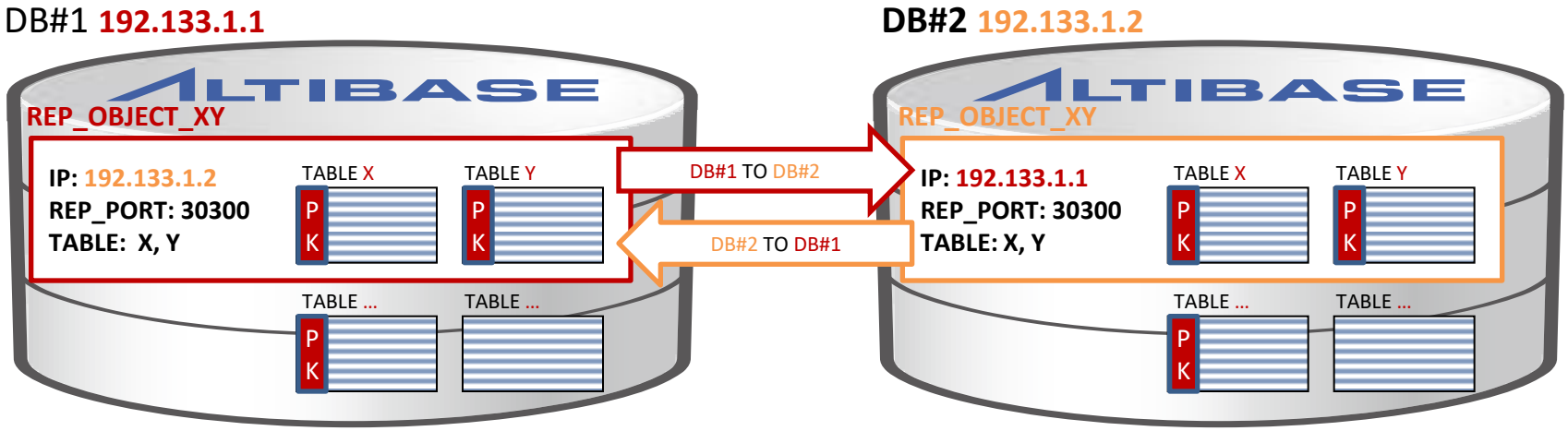
- 3개의 레코드를 변경하는 하나의 UPDATE 구문 수행
- 관련 리두로그가 최소 3개의 Xlog로 치환되어 전송
  - 치환될 때마다 Xlog 단위로 실시간 전송
- Xlog에 대응하는 레코드 변경연산 실행이 총 3회 발생



ALTIBASE REPLICATION

❖ 이중화 객체(REPLICATION OBJECT)

- 이중화 수행을 위한 객체로 지역SERVER와 원격SERVER에 모두 존재해야만 이중화 가능
  - 동일한 이름으로 대응되는 이중화 객체간 1:1로만 이중화 수행, 전이되지 않음
- IP, 이중화 포트(port)번호로 식별
  - 하나의 데이터베이스에 REPLICATION_MAX_COUNT 설정 값 만큼 이중화 생성 가능
- 객체 별 다중 IP 지원
- 객체 별 lazy, eager 복제방식 지정 가능
- 객체 별 이중화 대상 테이블의 컬럼 및 상세조건과 기타 정보 유지



❖ 이중화 대상 테이블의 요건

- PK 필수
- 이중화 대상 테이블의 컬럼은 이름, 데이터 타입 및 길이가 일치
  - 레코드 중 복제될 컬럼을 이름으로 식별
  - 컬럼 개수 및 순서는 지역SERVER와 원격SERVER가 각각 다르더라도 무방
  - 지역SERVER의 테이블에는 존재하고 원격SERVER에 없는 컬럼은 NULL로 채워짐
  - 컬럼 이름은 동일한데 스펙이 다르면 이중화 구동(START)시점에 에러 발생
  - 데이터타입, 길이, 제약조건 등
  - FK 불가

❖ 이중화 객체 생성 구문

```
CREATE [LAZY¦EAGER] REPLICATION replication_name [AS MASTER¦AS SLAVE]
[OPTIONS options … [options … ] ]
WITH { ‘ remote_host_ip ‘ , remote_replication_port_no }
FROM user_name.table_name TO user_name.table_name
[, FROM user_name.table_name TO user_name.table_name ] ;
```

- LAZY, EAGER : 이중화 복제방식, 생략 시 lazy로 지정
- MASTER, SLAVE : Conflict Resolution을 위한 SERVER의 역할 지정, 생략 시 미지정으로 설정
- option : 오프라인 이중화 같은 이중화 객체에 대한 부가 기능
- replication_name : 이중화 객체 명, 이중화를 수행하려는 SERVER간 이름이 동일해야 함
- remote_host_ip : 원격SERVER IP 주소
- remote_replication_port_no : 원격SERVER 이중화 수신 포트번호
- FROM - TO : 이중화 대상 테이블을 지역SERVER, 원격SERVER 순으로 명시

❖ 이중화 대상 SERVER IP 추가/삭제/설정 구문 (다중 IP 설정)

```
ALTER REPLICATION replication_name {ADD¦DROP¦SET} HOST
‘ remote_host_ip ‘ , remote_replication_port_no ;
```

❖ 이중화 대상 테이블 추가/삭제 구문

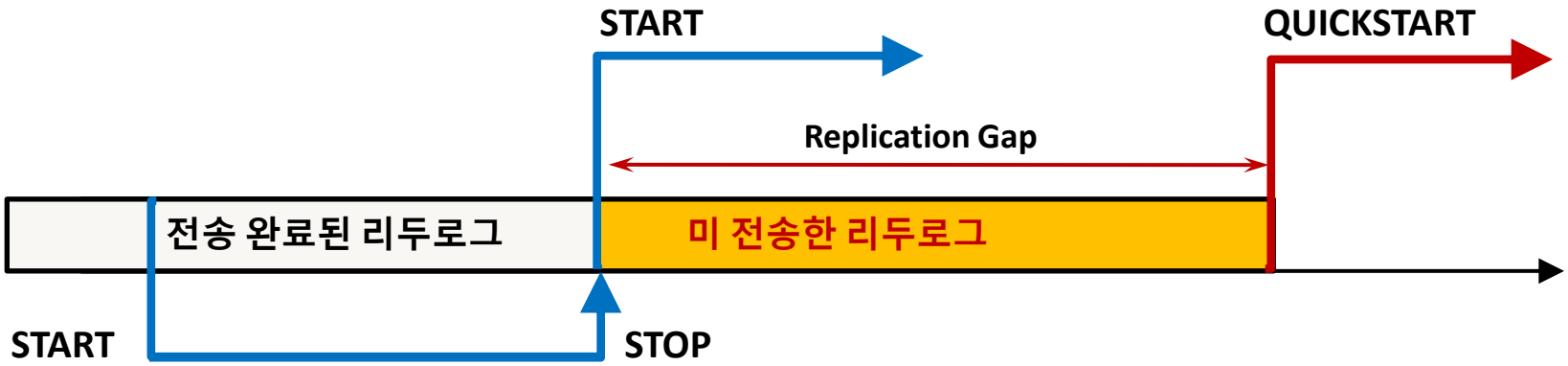
```
ALTER REPLICATION replication_name {ADD¦DROP} TABLE
FROM user_name.table_name TO user_name.table_name ;
```

❖ 이중화 객체 삭제 구문

```
DROP REPLICATION replication_name ;
```

❖ 이중화 제어

- START - 마지막 이중화 수행시점을 기점으로 이중화 시작 (일반구동)
- QUICKSTART - 미 전송 리두 로그를 포기, 현재를 기점으로 이중화를 시작 (긴급구동)
- STOP - 현재 기점으로 이중화를 중지
- * 그림 : stop 이후로 logfile 적체되는 상황



❖ 관련 구문

```
ALTER REPLICATION replication_name {START|QUICKSTART|STOP} ;
```

ALTIBASE REPLICATION

❖ 세션의 이중화 제어

- 수행 시점부터 해당 세션에서 발생하는 TRANSACTION에 대한 이중화 여부 지정
- TRANSACTION 특성에 따라 이중화 여부를 동적으로 선택 가능

❖ 관련 구문

```
ALTER SESSION SET REPLICATION = {NONE|DEFAULT} ;
```

- NONE - 이중화하지 않음
- DEFAULT - 이중화 객체에 설정된 복제방식으로 변경

```
ALTER SESSION SET REPLICATION = {TRUE|FALSE} ;
```

- FALSE - 이중화하지 않음
- TRUE - 이중화 객체에 설정된 복제방식으로 변경

❖ 이중화 대상 ALTIBASE 선정 조건

- Character Set 동일
- National Character Set 동일(5.3.3 higher)
- 이중화와 관련된 ALTIBASE 내부요소 버전 동일
  - ALTIBASE major 버전이 같다면 대부분 동일하나 반드시 확인 필요
  - Replication Protocol 버전
  - Meta 버전

```
# altibase -v
version 7.1.0.1.0 X86_64_LINUX_redhat_Enterprise_release6.0-64bit-7.1.0.1.3-release-GCC4.6.3
(x86_64-unknown-linux-gnu) May  4 2018 16:14:51, binary db version 6.5.1, meta version 8.5.1, cm
protocol version 7.1.6, replication protocol version 7.4.2, shard version 2.0.0
```

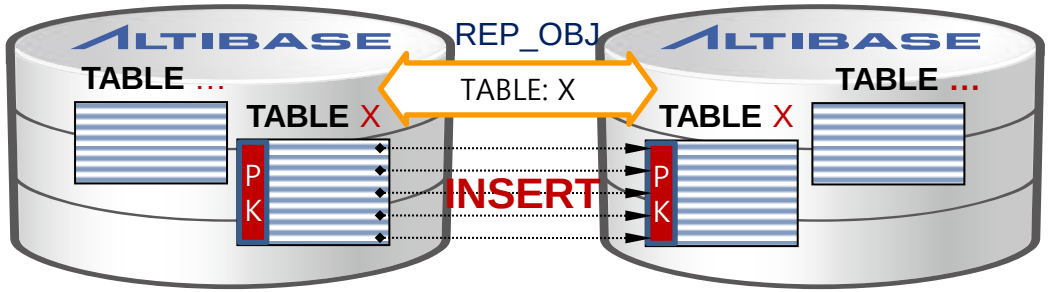
ALTIBASE REPLICATION

❖ 테이블 복제(SYNC)

- 이중화 객체를 이용한 테이블 복제 기능
- 지역SERVER 테이블의 모든 레코드를 원격SERVER의 테이블로 INSERT하는 방식
  - 이중화가 중지(STOP)된 상태에서만 수행 가능
  - 복제 완료 후 자동으로 이중화를 시작(START)
- 특정 테이블 복구, 장애 노드 복구, 신규 노드 추가 시 사용

❖ 간략 수행 절차

- 복제대상 테이블 TRUNCATE 또는 재생성
- 이중화 객체에 대상 테이블을 추가 또는 이중화 객체 재생성
- 지역SERVER에서 테이블 복제 구문 수행



❖ 테이블 복제(SYNC) 구문

```
ALTER REPLICATION replication_name SYNC [ONLY]
[PARALLEL parallel_factor ]
[TABLE user_name.table_name, … , user_name.table_name ];
```

- SYNC : 로컬 SERVER의 이중화 대상 테이블 데이터를 원격지로 복제 이후 이중화는 자동으로 START
- ONLY : 명시할 경우 테이블 복제만 수행, 자동으로 이중화를 시작(START)하지 않음
- parallel_factor : 테이블 복제를 수행할 쓰레드의 개수 (기본값 1, 최대 CPU*2)
- TABLE : 복제를 수행할 테이블을 명시, 생략 시 해당 이중화 객체의 모든 테이블이 대상

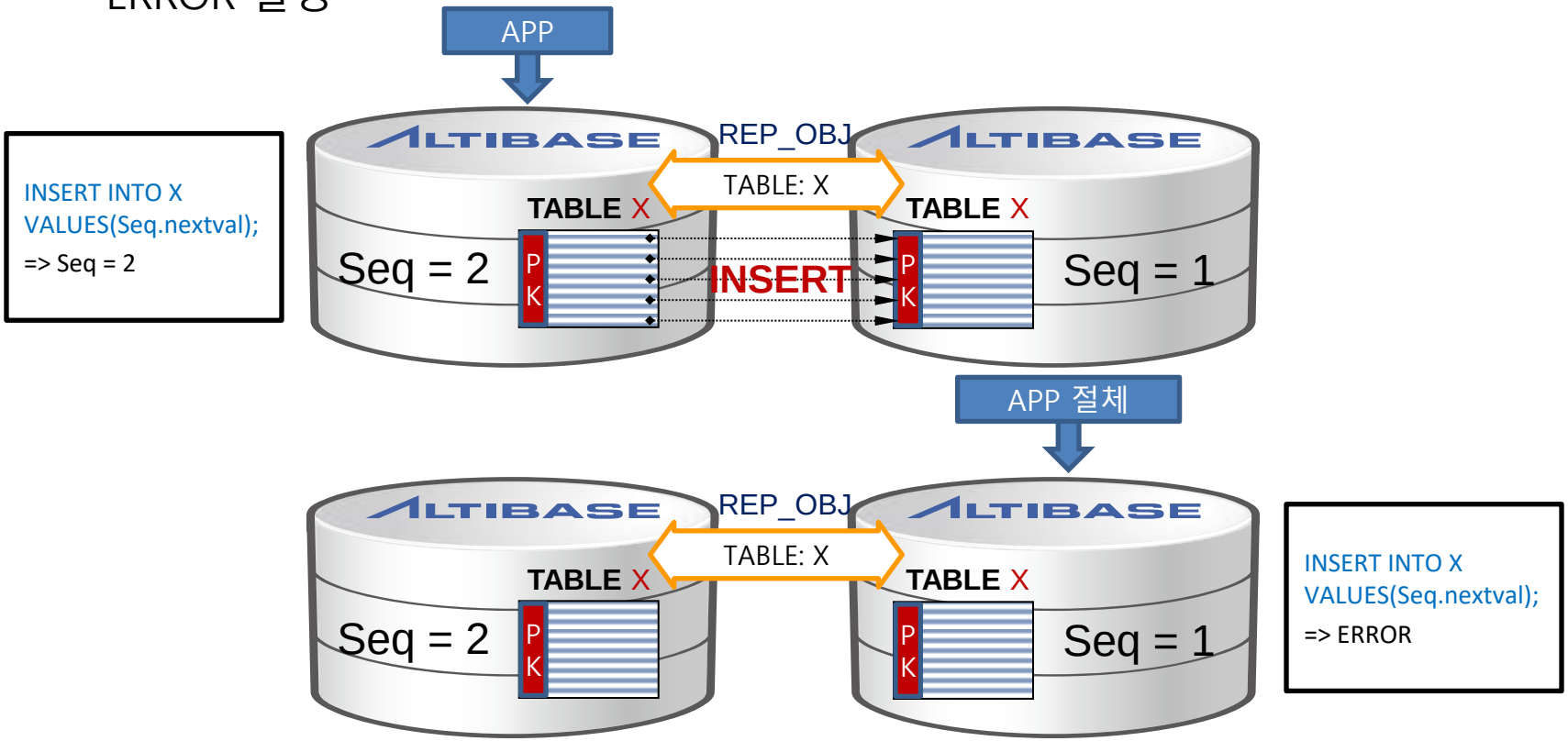
❖ 테이블 복제(SYNC) 시 유의 사항

- Active-Active 구성에서 시스템 서비스 중 수행 지양
  - 테이블 복제 완료 전까지 해당 이중화 객체의 다른 테이블들이 이중화되지 않음
  - Active-Standby 구성이라면 사용 가능하나 일시적인 부하 발생 고려
- 대응되는 원격SERVER 테이블의 모든 레코드 삭제 후 수행
  - 동일한 PK의 레코드가 이미 존재 시 삽입 충돌이 발생하여 복제 실패
  - 동일한 PK에 대한 INSERT 연산으로 인한 DBMS 부하 발생
  - 이중화 trace 로그파일에 다량의 삽입 충돌 에러 발생
- 원격SERVER 테이블 레코드 삭제 시에는 TRUNCATE 수행 권장
  - 사용자 실수 방지 차원
  - DELETE로 원격SERVER 테이블의 레코드 삭제하여 지역SERVER의 레코드도 삭제

ALTIBASE REPLICATION

❖ 시퀀스 이중화

- 이중화 환경에서 테이블만 이중화를 지원하기 때문에 FAIL-OVER 시 시퀀스의 관리가 필요
- 아래 예제와 같이 시퀀스(Seq)를 key로 사용 할 때 어플리케이션(APP) 절체 시 ERROR 발생



ALTIBASE 시퀀스 이중화

- ALTIBASE 이중화는 이를 해결하기 위해 시퀀스 자체 복제가 아닌 시퀀스 전용테이블로 시퀀스 이중화 지원

시퀀스 이중화를 위한 시퀀스 생성

- 시퀀스 생성 시 ENABLE SYNC TABLE 옵션을 명시하면 *seq_name$seq* 테이블이 생성
  - 캐쉬는 100이상 권장

```
CREATE SEQUENCE user_name.seq_name START WITH 1 CACHE 100 ENABLE SYNC TABLE;
```

시퀀스 이중화 생성

```
CREATE REPLICATION repl_name WITH 'remote_host_ip', remote_host_port_no FROM
user_name.seq_name$seq TO user_name.seq_name$seq;
```

❖ 시퀀스 이중화 유의사항

- Active-Active 환경에서 사용 불가
- 시퀀스 캐시 크기가 클수록 생성 속도가 향상
- FAIL-OVER 시점에는 캐시 크기 만큼 원격 SERVER의 시퀀스 공백 발생
- 이중화 재생성과 시퀀스 재생성 및 변경은 모든 SERVER 동일하게 적용

이중화 객체에 대한 DDL 수행

- 이중화와 관련된 모든 구문은 SYS 사용자로만 수행 가능
- 이중화 객체 변경(ALTER)에 대한 모든 구문은 이중화가 중지(STOP)된 상태에서만 가능
- 이중화 대상 테이블은 DDL 수행 불가
  - 이중화 대상 테이블 여부와 무관하게 수행 가능한 DDL 일부 제외
  - 프로퍼티를 통하여 추가적인 DDL 허용 가능

이중화 대상 테이블 여부와 무관하게 수행 가능한 DDL

- ALTER INDEX REBUILD PARTITION
- GRANT OBJECT
- REVOKE OBJECT
- CREATE TRIGGER
- DROP TRIGGER

❖ 이중화 객체에 대한 DDL 수행 절차

- REPLICATION_DDL_ENABLE 을 1로 설정 후, 허용가능 DDL 수행

❖ 프로퍼티를 통하여 허용 가능한 DDL (*버전 별 매뉴얼 참조)

```
ALTER TABLE table_name {ADD | DROP} COLUMN
ALTER TABLE table_name ALTER COLUMN column_name {SET | DROP} DEFAULT
ALTER TABLE table_name MODIFY COLUMN
ALTER TABLE table_name {ADD | DROP}  CONSTRAINT
{CREATE | DROP} {UNIQUE} INDEX index_name
ALTER TABLE table_name TRUNCATE PARTITION
TRUNCATE TABLE
{CREATE | DROP} INDEX
```

❖ 유의사항

- DDL 수행은 이중화되지 않으므로 관련된 SERVER 모두 동일하게 수행

❖ 특정 이중화 대상 테이블에 새로운 컬럼 추가 절차

➢ 작업 요건

- 예정된 작업시간 동안 해당 테이블에 변경 연산이 발생하지 않아야 함
- SELECT 연산은 변경연산과 무관하므로 조회 서비스 가능

SERVER A	SERVER B
테이블 rep_tbl에 변경연산 관련 서비스 종료여부 확인 iSQL> connect sys/manager; iSQL> ALTER REPLICATION rep_obj_01 STOP; iSQL> ALTER REPLICATION rep_obj_01 DROP TABLE FROM user01.rep_tbl TO user01.rep_tbl; ------DDL start---------- iSQL> connect user01/user01; iSQL> ALTER TABLE rep_tbl ADD COLUMN (new_col CHAR(9)); ------DDL end---------- iSQL> connect sys/manager; iSQL> ALTER REPLICATION rep_obj_01 ADD TABLE FROM user01.rep_tbl TO user01.rep_tbl; iSQL> ALTER REPLICATION rep_obj_01 START; 서비스 정상화	테이블 rep_tbl에 변경연산 관련 서비스 종료여부 확인 iSQL> connect sys/manager; iSQL> ALTER REPLICATION rep_obj_01 STOP; iSQL> ALTER REPLICATION rep_obj_01 DROP TABLE FROM user01.rep_tbl TO user01.rep_tbl; ------DDL start---------- iSQL> connect user01/user01; iSQL> ALTER TABLE rep_tbl ADD COLUMN (new_col CHAR(9)); ------DDL end---------- iSQL> connect sys/manager; iSQL> ALTER REPLICATION rep_obj_01 ADD TABLE FROM user01.rep_tbl TO user01.rep_tbl; iSQL> ALTER REPLICATION rep_obj_01 START; 서비스 정상화

REPLICATION SYSTEM DESIGN

REPLICATION SYSTEM DESIGN

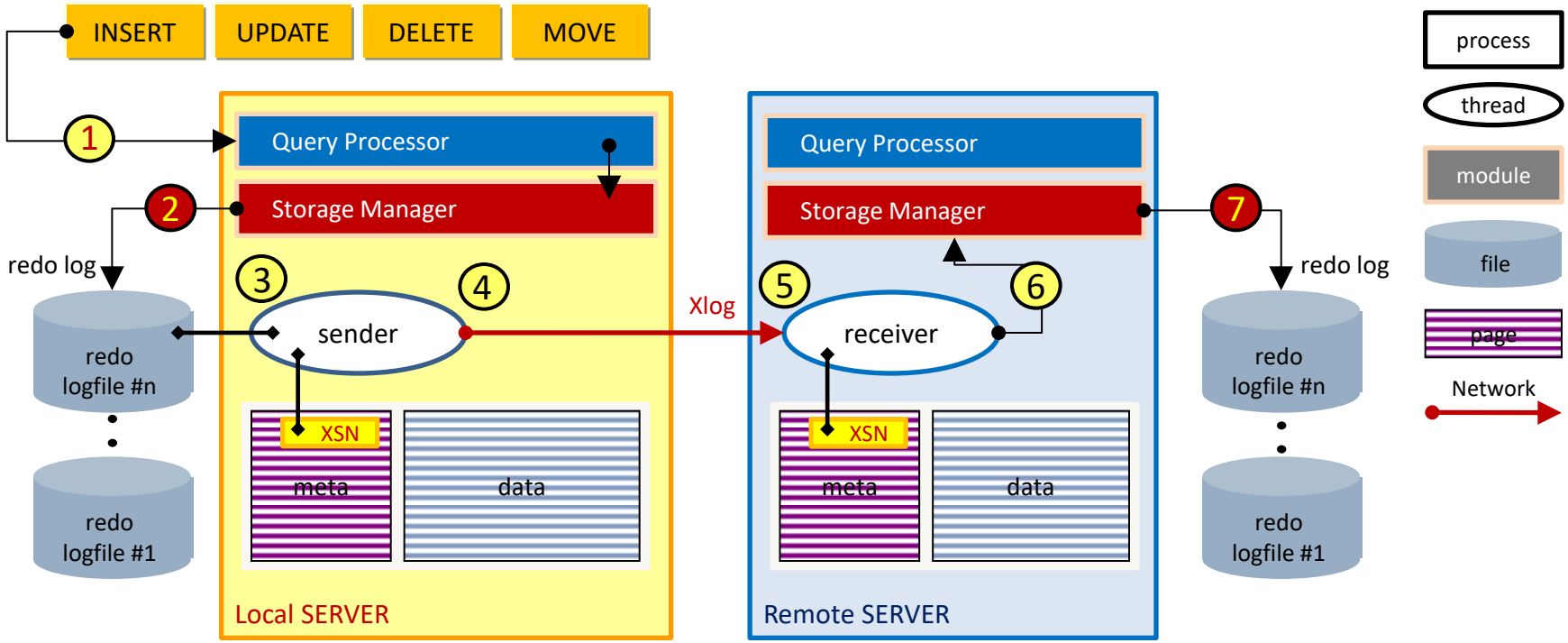
❖ 복제방식 구현

- lazy - 이중화를 모듈화하여 마스터 TRANSACTION에 영향을 주지 않는 방식
- eager - 2PC(2 Phase commit)와 유사한 방식

❖ 구성방식 구현

- Active-Active
  - 모두 Sender를 구동
- Active-Standby
  - Fail-over를 고려한 시스템 : Active-Active 와 동일하게 모두 sender 구동
  - 백업만을 위한 시스템 : Active만 sender 구동

REPLICATION SYSTEM DESIGN

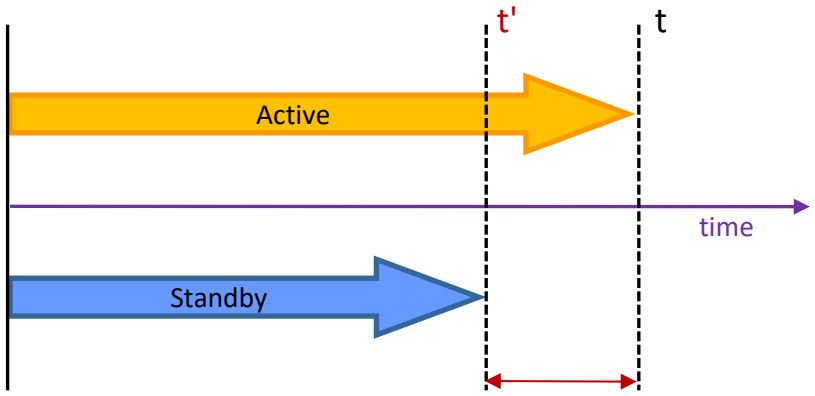


- **Lazy**
  - 마스터 TRANSACTION = 1 + 2
  - 이중화 TRANSACTION = 3 + 4 + 5 + 6 + 7
- **Eager**
  - TRANSACTION = 1 + 2 + 3 + 4 + 5 + 6 + 7
    - 이중화 TRANSACTION(7)까지 이상없이 반영되었을 때 마스터 TRANSACTION(2) 확정

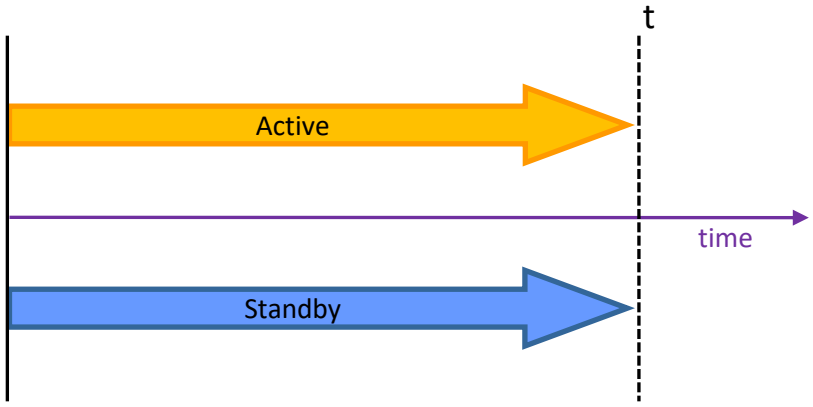
REPLICATION SYSTEM DESIGN

❖ 이중화 노드간 동기화 시점에 따른 이중화 복제방식

- lazy : 비동기식. 마스터 TRANSACTION과 이중화 TRANSACTION이 별개로 수행
  이중화 지연은 발생하나 TRANSACTION 처리 성능 빠름
- eager : 동기식. 마스터 TRANSACTION과 이중화 TRANSACTION이 하나로 수행
  이중화 지연은 발생하지 않으나 TRANSACTION 처리 성능 느림



[그림1. lazy]



[그림2. eager]

❖ 주요 특징

- 이중화 지연(Replication Delay or Gap) 발생과 TRANSACTION 처리 성능 측면에서의 trade-off

REPLICATION SYSTEM DESIGN

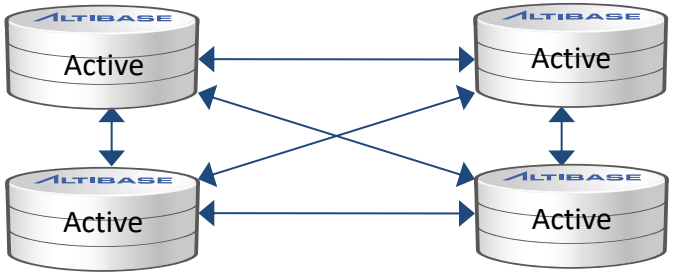
❖ 이중화 갭(REPLICATION GAP)

- ➢ 이중화 지연 정도를 성능 뷰 V$REPGAP 에서 수치로 제공
- ➢ 미 전송된 ONLINE LOG FILE의 SIZE를 표시
  - version 6.5.1 이전 제품은 ONLINE LOG FILE의 일련번호인 SN(Sequence Number)과 XSN을 통한 계산
    (이중화 갭 = [지역SERVER의 최신 SN] - [지역SERVER의 최신 XSN])

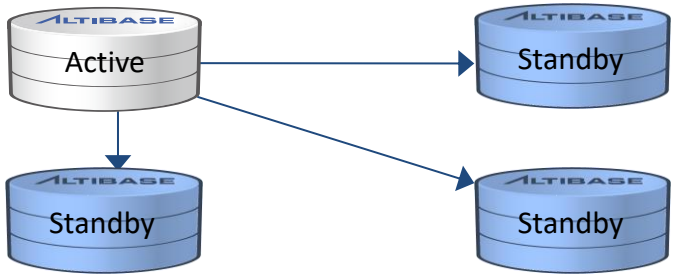
REPLICATION SYSTEM DESIGN

❖ 변경연산 주체에 따른 이중화 노드 구성방식

- Active-Active 모든 이중화 노드에서 변경연산이 가능하나 변경연산간 충돌 가능성 있음
- Active-Standby 특정 이중화 노드에서만 변경연산이 가능하나 변경 연산간 충돌 가능성 없음



[그림1. Active-Active]



[그림2. Active-Standby]

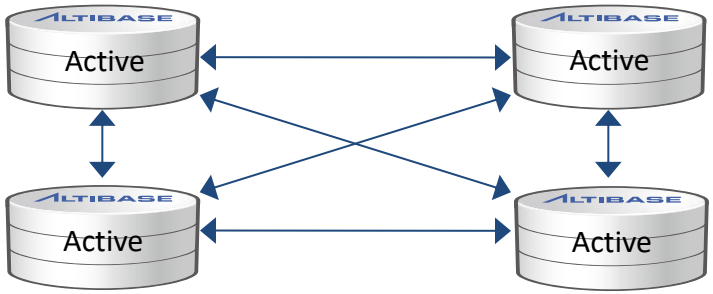
❖ 주요 특징

- 변경 연산의 부하 분산과 변경 연산간 충돌 가능성 측면에서의 trade-off
- 구성 방식에 따라 응용프로그램 고려 사항 발생
- Active-Active : lock 발생에 대한 고려
- Active-Standby : 노드 역할에 따른 응용프로그램 유지

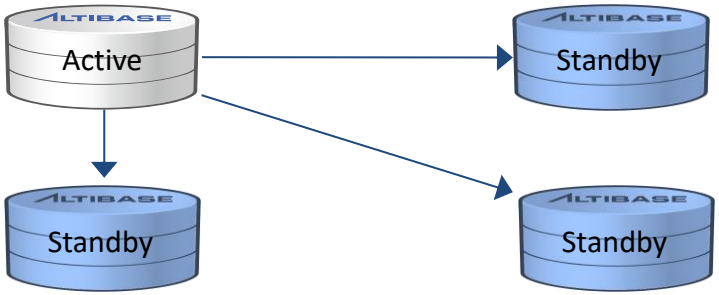
REPLICATION SYSTEM DESIGN

❖ 이중화 시스템 설정 방법

- Active-Active, Active-Standby(fail-over)
  - 한 개의 SERVER당 [전체SERVER개수 - 1] 개의 이중화 객체 생성
  - 모든 SERVER에서 sender 구동
- Active-Standby(backup)
  - Active SERVER는 [전체SERVER개수 - 1] 개의 이중화 객체 생성
  - Standby SERVER는 Active에 대응되는 1개의 이중화 객체만 생성
  - Active SERVER에서만 sender 구동



[그림1. Active-Active, Active-Standby(fail-over)]



[그림2. Active-Standby(backup)]

REPLICATION SYSTEM DESIGN

❖ 구성방식과 복제방식에 따른 고려사항

구분	Active-Active		Active-Standby	
	lazy	eager	lazy	eager
이중화 성능	빠름	느림	빠름	느림
이중화 지연	발생	발생하지 않음	발생	발생하지 않음
변경연산의 부하분산	모든 DML 가능 (INSERT,UPATE,DELETE,SELECT)		불가능 (SELECT만 가능)	
응용프로그램	lock 경합 고려		이중화 역할에 따라 최소 두 벌 유지	
이중화 충돌	발생	발생하지 않음		
데이터불일치	발생가능	발생하지 않음	발생가능	발생하지 않음

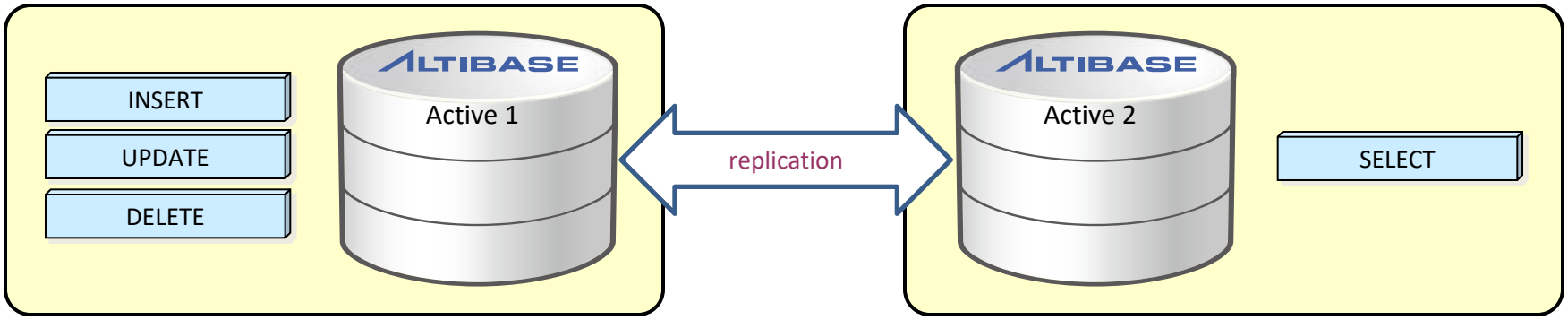
❖ 이중화 도입 절차

- 시스템 요건에 부합하는 구성방식과 복제방식 선택
- 빠른 성능이 장점인 lazy 복제방식을 채택하는 것이 일반적
- 가장 이상적인 것은 Active-Active & lazy 조합에 이중화 충돌을 회피한 시스템 설계
- 구성방식과 복제방식에 따라 발생 가능한 문제에 대한 대응 방안 수립
- Network 장애의 경우는 eager 복제방식이라 할지라도 장애 복구 방안 수립필요

REPLICATION SYSTEM DESIGN

❖ 변경연산 전용 SERVER 지정 방식

- ➢ SERVER1은 변경연산 전용, SERVER2는 조회 연산 전용 설계



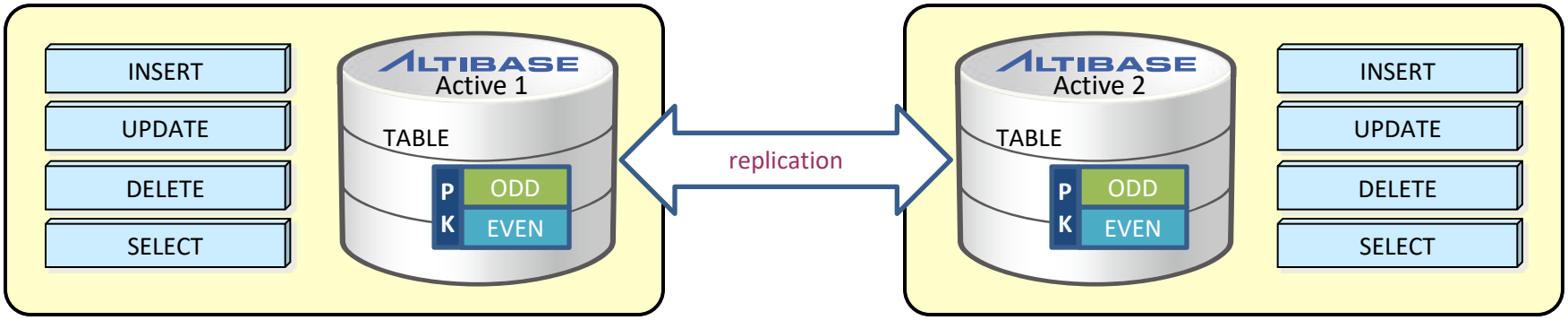
❖ 주요특징

- ➢ 조회 연산의 부하 분산만 가능하므로 응용이 제한적

REPLICATION SYSTEM DESIGN

❖ PK를 노드 개수 만큼 분할하는 방식

- ➢ SERVER1은 짝수 전용, SERVER2는 홀수 전용으로 설계



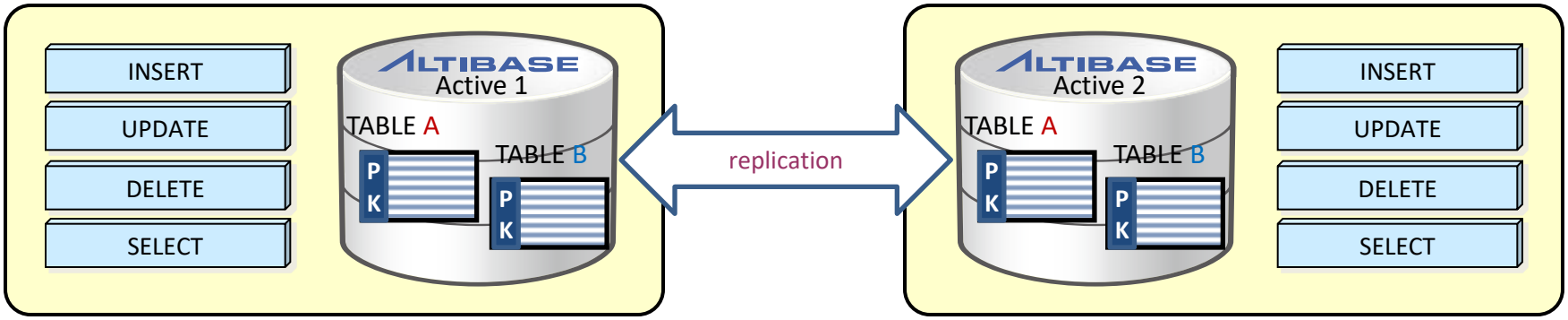
❖ 주요특징

- ➢ 변경 연산의 부하 분산이 가능하나 응용프로그램 구현에 유의 필요

REPLICATION SYSTEM DESIGN

❖ 업무에 따라 테이블을 분할하는 방식

- SERVER1은 A업무 테이블 변경 연산 전용, SERVER2는 B업무 테이블 변경 연산 전용 설계



❖ 주요특징

- 변경 연산의 부하 분산이 가능하나 복합 업무 처리 경우에 대한 고려 필요

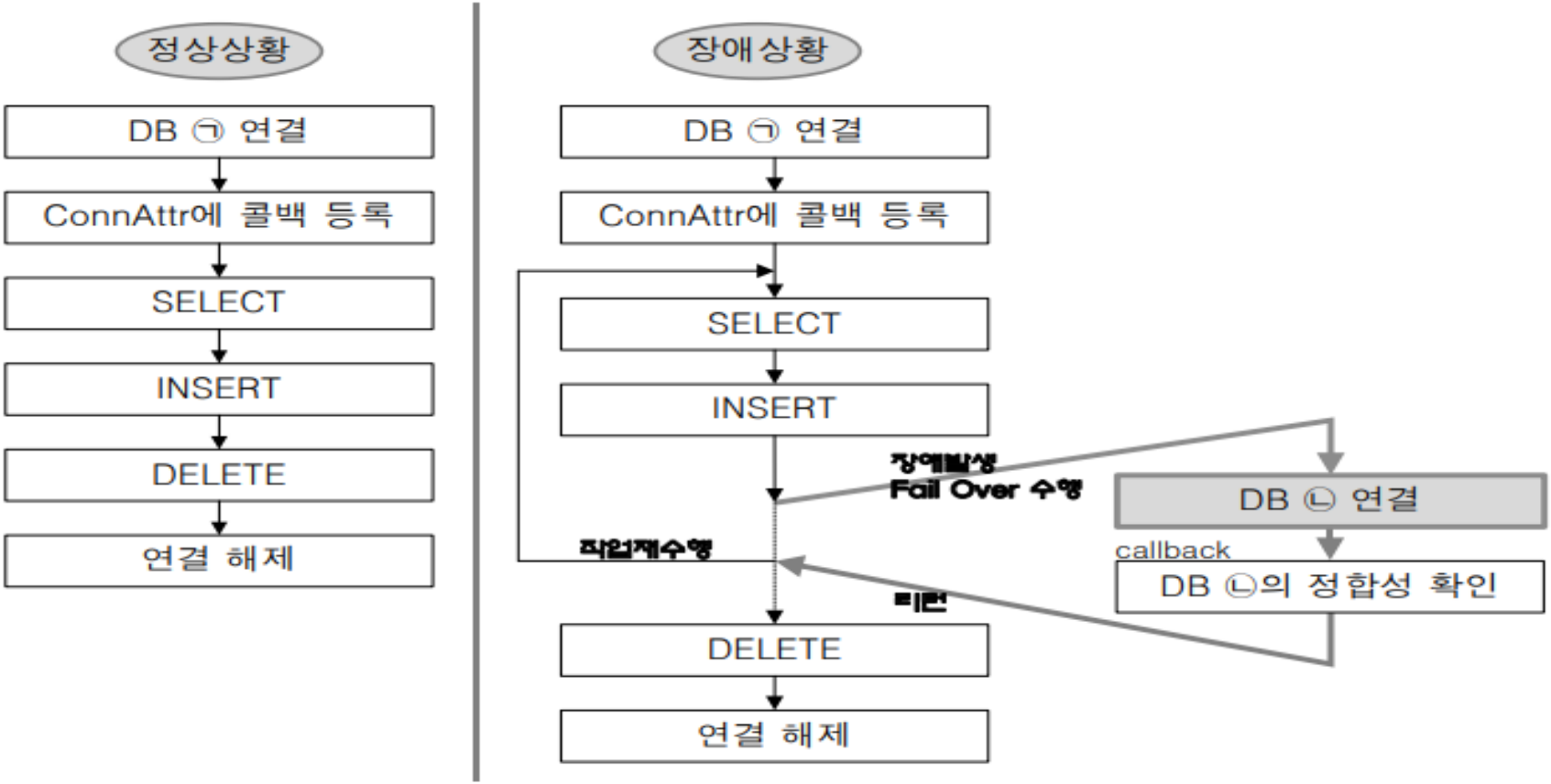
REPLICATION

FAIL-OVER

FAIL-OVER

❖ FAIL-OVER 란?

- 이중화로 구성된 DBMS 운영 중 장애가 발생하였을 때 이를 극복하고 장애가 발생하지 않은 것처럼 서비스를 계속 할 수 있도록 하는 것을 의미



FAIL-OVER

❖ 장애를 인식하는 시점에 따른 FAIL-OVER 분류

- CTF(Connection Time Fail-Over)
  - DBMS 접속 시점에 장애를 인식하여 장애가 발생한 DBMS 대신 다른 가용 노드의 DBMS로 접속하고 서비스 진행
- STF(Service Time Fail-Over)
  - DBMS 접속에 성공하여 서비스하는 도중 장애가 발생했을 때 다른 가용 노드의 DBMS에 다시 접속하여 세션의 프로퍼티를 복구한 후 사용자 응용 프로그램의 업무 로직을 계속 수행

FAIL-OVER

❖ FAIL-OVER 설정

➢ JDBC

- Connection url 부분에 Fail-Over 관련 속성 지정

```
jdbc:Altibase://192.168.3.52:20300/mydb?
  AlternateServers=(192.168.3.54:20300,192.168.3.53:20300)
  &ConnectionRetryCount=3&ConnectionRetryDelay=3
  &SessionFailOver=off
```

➢ ODBC, APRE

- 연결 스트링 부분에 Fail-Over 관련 속성 지정

```
DSN=192.168.3.52;UID=sys;PWD=manager;PORT_NO=20300;
AlternateServers=(192.168.3.54:20300,192.168.3.53:20300);
ConnectionRetryCount=3;ConnectionRetryDelay=3;
SessionFailOver=off
```

※ 7.1.0 버전부터 LoadBalance 항목 제거

FAIL-OVER

❖ FAIL-OVER 설정 속성

속성	상세설명
AlternativeServer	장애 발생 시 접속하게 될 가용 SERVER를 나타내며 다음과 같은 형식으로 추가 (IP Address1:port1, IP Address2:port2, ...)
ConnectionRetryCount	가용 SERVER 실패 시, 접속 시도 반복 횟수
ConnectionRetryDelay	가용 SERVER 접속 실패 시, 다시 접속을 시도하기 전에 대기하는 시간 (초 단위)
SessionFailOver	Fail-Over 할 때의 mode 결정 ON : STF, OFF: CTF

REPLICATION

CONFLICT RESOLUTION

CONFLICT RESOLUTION

❖ 이중화 충돌 유형

- 삽입충돌 - 동일한 PK에 대한 INSERT 연산
- 변경충돌(1) - 동일한 PK에 대한 UPDATE 연산
- 변경충돌(2) - 존재하지 않는 PK에 대한 UPDATE 연산
- 삭제충돌 - 존재하지 않는 PK에 대한 DELETE 연산

CONFLICT RESOLUTION

❖삽입 충돌, 변경 충돌(2), 삭제 충돌 발생 시나리오

time	node A	node B
T1	**Insert(PK1) & commit**	**Insert(PK1) & commit**
T2	send log(T1)	send log(T1)
T3		receive log(node A's T2) ***INSERT CONFLICT**
T4	**delete(PK1) & commit**	
T5	send log(T4)	
T6		receive log(node A's T5) ***SUCCESS**
T7	**receive log(node B's T2) *SUCCESS**	
T8	**select(PK 1) - 1 rows**	**select(PK 1) - no rows**
T9	**update(PK1) & commit**	
T10	send log(T9)	
T11		receive log(node A's T10) ***UPDATE CONFLICT(2)**
T12	**delete(PK1) & commit**	
T13	send log(T12)	
T14		receive log(node A's T13) ***DELETE CONFLICT**

CONFLICT RESOLUTION

❖ 변경충돌(1) 발생 시나리오

- ➢ 변경 충돌에 대한 감지를 하지 않을 경우 발생할 수 있는 시나리오
- ➢ 동일한 PK에 대한 서로 다른 노드의 변경 연산 결과가 성공, 별도 감지 필요

time	node A	node B
T1	**Update C1=C -> C1=A (update(PK1, 'A') & commit)**	
T2	send log(T1) C->A	
T3		C1=C -> C1=B (**update(PK1, 'B') & commit)**
T4		send log(T3) C->B
T5	receive log(node B's T4) ***SUCCESS**	
T6		**receive log(node A's T2) *SUCCESS**
T7	**select(PK1) - 'B' *UPDATE CONFLICT(1) C=A 인 상태에서 C->B로 update 시도는 fail**	**select(PK1) - 'A' *UPDATE CONFLICT(1) C=B 인 상태에서 C->A로 update 시도는 fail**

CONFLICT RESOLUTION

❖ ALTIBASE에서 제공하는 이중화 충돌 해결

- DBMS Level
  - User-oriented scheme
  - Timestamps-based scheme
  - Master-Slave scheme
- Utility Level
  - Alticomp(~ version 6.3.1 은 Audit)

❖ 이중화 충돌 유형에 따른 처리

충돌 유형	관련 연산	발생 상황	처리
삽입 충돌	INSERT	동일한 PK에 대한 INSERT 연산	수신 측에 설정된 이중화 충돌 해결 정책 따름
변경 충돌	UPDATE	동일한 PK에 대한 UPDATE 연산	
		존재하지 않는 PK에 대한 UPDATE 연산	이중화 충돌 리포트만 수행
삭제 충돌	DELETE	존재하지 않는 PK에 대한 DELETE 연산	

CONFLICT RESOLUTION

❖ USER-ORIENTED SCHEME

- 기본적으로 설정되어 있는 이중화 충돌 해결 정책
- 이중화 충돌 발생 시 해당 레코드와 관련된 연산 무시
- 사용자가 확인 후 조치할 수 있도록 이중화 trace 로그 파일에 리포트만 수행
- $ALTIBASE_HOME/trc/altibase_rp_conflict.log (6.1.1 이전버전은 altibase_rp.log)

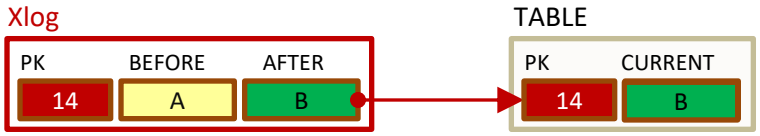
❖ 이중화 충돌 유형에 따른 감지와 처리

충돌 유형	발생 상황	충돌 감지	처리
삽입충돌	동일한 PK에 대한 INSERT 연산	PK 존재여부로 감지	관련연산을 모두 무시, 감지에 대한 리포트만 수행
변경충돌	동일한 PK에 대한 UPDATE 연산	**value-based 기법으로 감지**	
	존재하지 않는 PK에 대한 UPDATE 연산	PK 존재여부로 감지	
삭제충돌	존재하지 않는 PK에 대한 DELETE 연산	PK 존재여부로 감지	

CONFLICT RESOLUTION

❖ VALUE-BASED 기법

- ➢ 변경충돌을 감지하기 위한 기법
- ➢ 변경전의 값과 현재 값을 비교하여 일치하지 않을 경우 변경 충돌로 판별
  - UPDATE 연산에 대한 Xlog는 관련된 컬럼의 변경 전 값과 변경 후의 값으로 구성



[그림1. 정상적인 연산으로 판별하는 경우]

[그림2. 이중화 충돌연산으로 판별하는 경우]

❖ 충돌 감지와 처리

- ➢ 프로퍼티 설정으로 충돌에 대한 감지와 처리를 선택

구분	발생상황	처리
삽입충돌	동일한 PK에 대한 INSERT 연산	REPLICATION_INSERT_REPLACE 값이 0 이면, 이중화 충돌 리포트 REPLICATION_INSERT_REPLACE 값이 1 이면, 이중화 충돌을 무시하고 반영
변경충돌	동일한 PK에 대한 UPDATE 연산	REPLICATION_UPDATE_REPLACE 값이 0 이면, 이중화 충돌 리포트 REPLICATION_UPDATE_REPLACE 값이 1 이면, 이중화 충돌을 무시하고 반영

CONFLICT RESOLUTION

❖ 유의사항

- LOB 컬럼은 동일한 PK 변경에 대한 충돌을 감지하지 않음
- LOB 데이터 타입의 특성
  - 리두 로그 자체에 변경전의 값이 없어 감지 불가능

CONFLICT RESOLUTION

❖ MASTER-SLAVE SCHEME

- ➢ 이중화 객체 생성 시 Master, Slave 지정으로 적용 가능한 이중화 충돌 해결 정책
- ➢ 항상 Master를 기준으로 처리

❖ 상세 처리 방식

구분	발생상황	처리	
		Master	Slave
삽입충돌	동일한 PK에 대한 INSERT 연산	이중화 충돌 리포트	기존 레코드를 삭제하고 INSERT 연산 반영
변경충돌	동일한 PK에 대한 UPDATE 연산	이중화 충돌 리포트	UPDATE 연산 반영

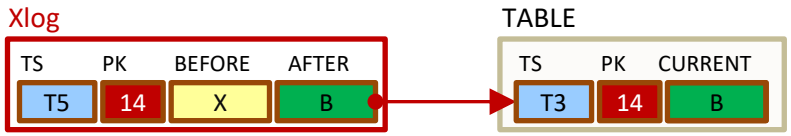
❖ 유의사항

- ➢ 하나의 Master 이중화 객체는 반드시 대응되는 Slave 객체가 있어야 이중화 가능
  - Master-Master (X), Slave-Slave (X), Master or Slave-NONE (X)

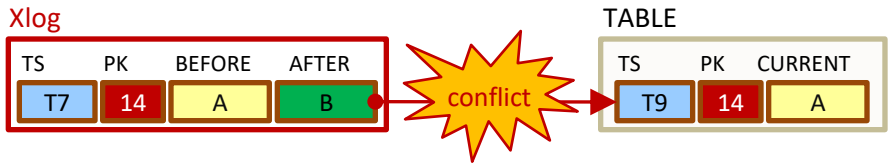
CONFLICT RESOLUTION

❖ TIMESTAMPS-BASED SCHEME

- ➢ 프로퍼티로 설정 가능한 이중화 충돌 해결 정책
  - REPLICATION_TIMESTAMP_RESOLUTION을 1로 설정 (기본값은 1)
- ➢ TIMESTAMP 컬럼을 활용한 우선순위 판별
  - 시간으로 순서화되어 상대적으로 신뢰성 있는 충돌 해결이 가능



[그림1. 정상적인 연산으로 판별하는 경우]



[그림2. 이중화 충돌연산으로 판별하는 경우]

❖ 상세 처리 방식

구분	발생상황	처리
삽입충돌	동일한 PK에 대한 INSERT 연산	TIMESTAMP가 같거나 크면, 기존 레코드를 삭제하고 INSERT 연산 반영. 그렇지 않으면, 이중화 충돌 리포트
변경충돌	동일한 PK에 대한 UPDATE 연산	TIMESTAMP가 같거나 크면, UPDATE 연산 반영. 그렇지 않으면, 이중화 충돌 리포트

CONFLICT RESOLUTION

❖ 유의사항

- 이중화 SERVER간 시간이 반드시 일치
- TIMESTAMP 컬럼 필수
  - 프로퍼티를 설정하였다 하더라도TIMESTAMP 컬럼 없는 테이블은 적용되지 않음
  - TIMESTAMP 컬럼은 사용자가 직접 추가 필요

❖ 참고사항

- TIMESTAMP 컬럼으로 인한 레코드당 8byte의 추가 저장 공간 발생
- TIMESTAMP 컬럼이 추가적으로 전송되므로 이중화를 위한 통신 비용 증가

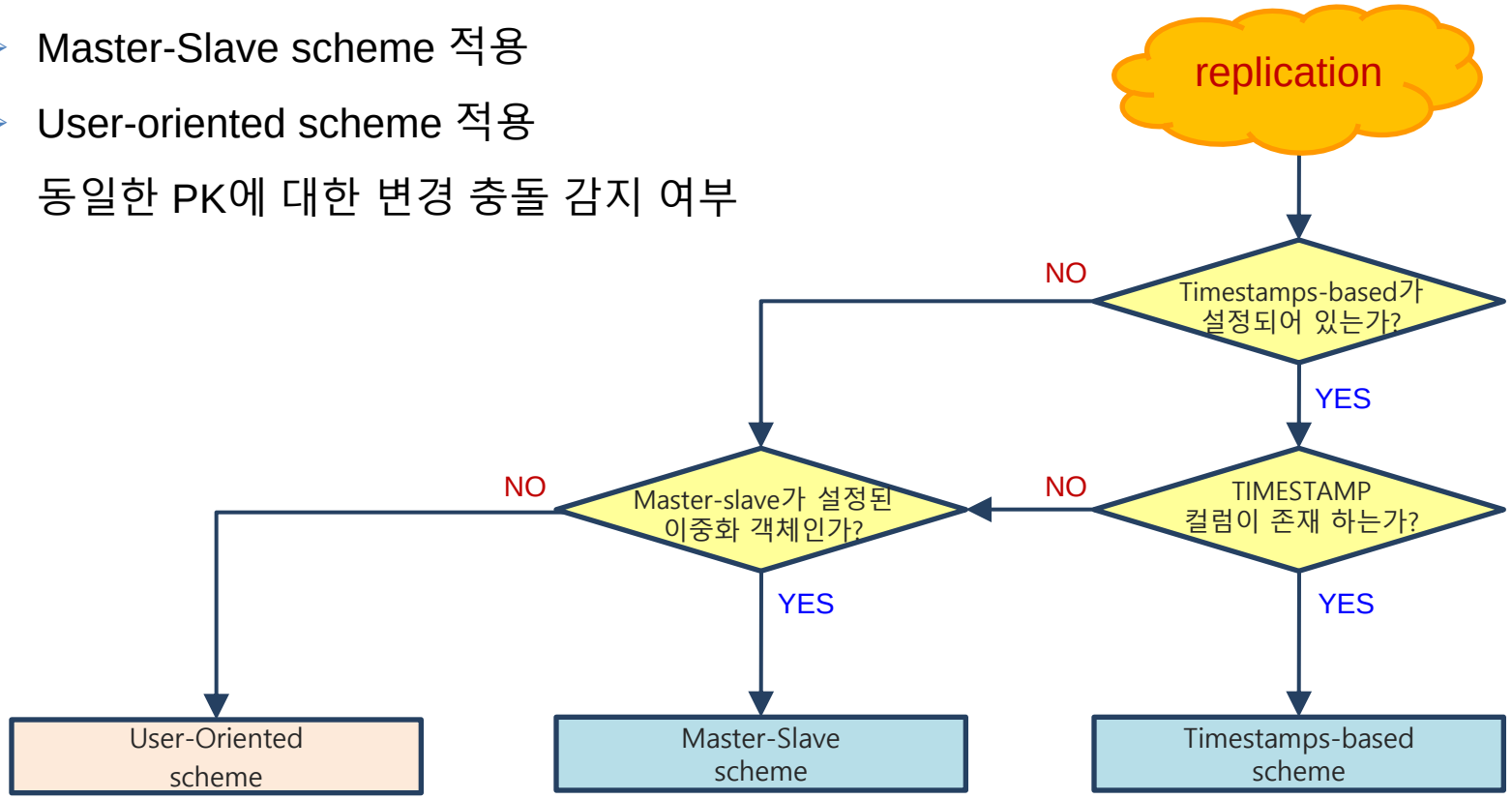
CONFLICT RESOLUTION

❖ 이중화 충돌 해결 SCHEME 혼용 시 처리 흐름

- Timestamps-based scheme가 우선적으로 적용

  단, Timestamp 컬럼 없다면 처리하지 않음
- Master-Slave scheme 적용
- User-oriented scheme 적용

  동일한 PK에 대한 변경 충돌 감지 여부



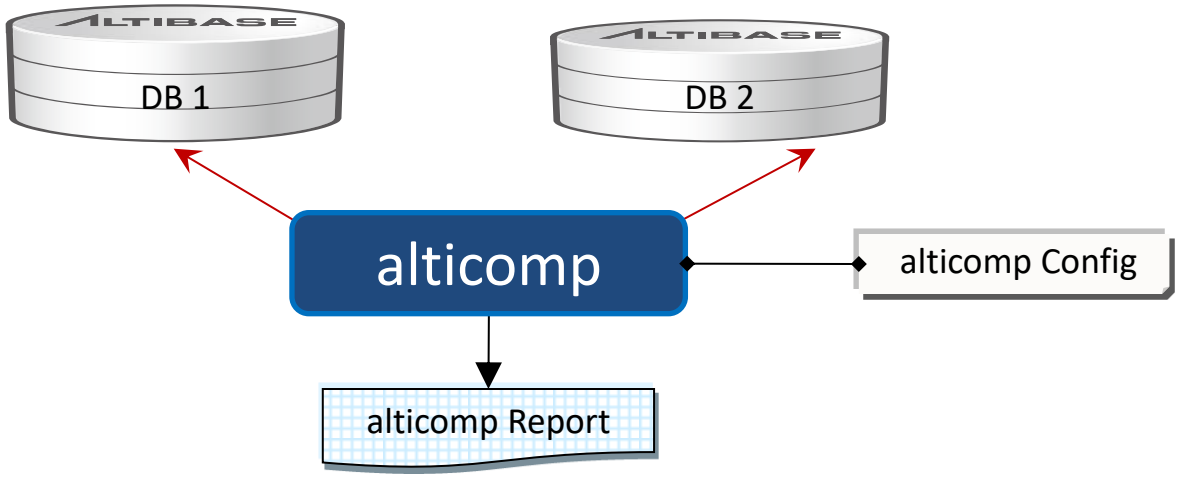
REPLICATION

ALTICOMP(UTILITY)

ALTICOMP(UTILITY)

❖ altiComp(6.3.1 이하 VERSION 은 audit 으로 가능함)

- 두 개 데이터베이스를 테이블 단위로 데이터 비교 또는 동기화 기능을 수행하는 유틸리티
- 이중화 충돌로 인한 데이터 불일치를 사용자 판단에 의해 일괄적으로 해결함이 목적
- 기본적으로 마스터 (MASTER) 데이터베이스를 기준으로 슬래이브(SLAVE) 데이터베이스를 일치시키는 정책 채택
- 대상 데이터베이스가 변경 중에는 정상적으로 수행되지 않을 수 있으므로 유의 필요



ALTICOMP(UTILITY)

❖ altiComp 데이터 불일치 종류

➢ Master 데이터베이스와 Slave 데이터베이스간의 데이터 불일치가 발생하는 세가지 경우

Master

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

Slave

KEY	VALUE
1	APPLE
2	ORANGE

MOSX

Master

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

Slave

KEY	VALUE
1	APPLE
2	ORANGE
3	GRAPE

MOSO

Master

KEY	VALUE
1	APPLE
2	ORANGE

Slave

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

MXSO

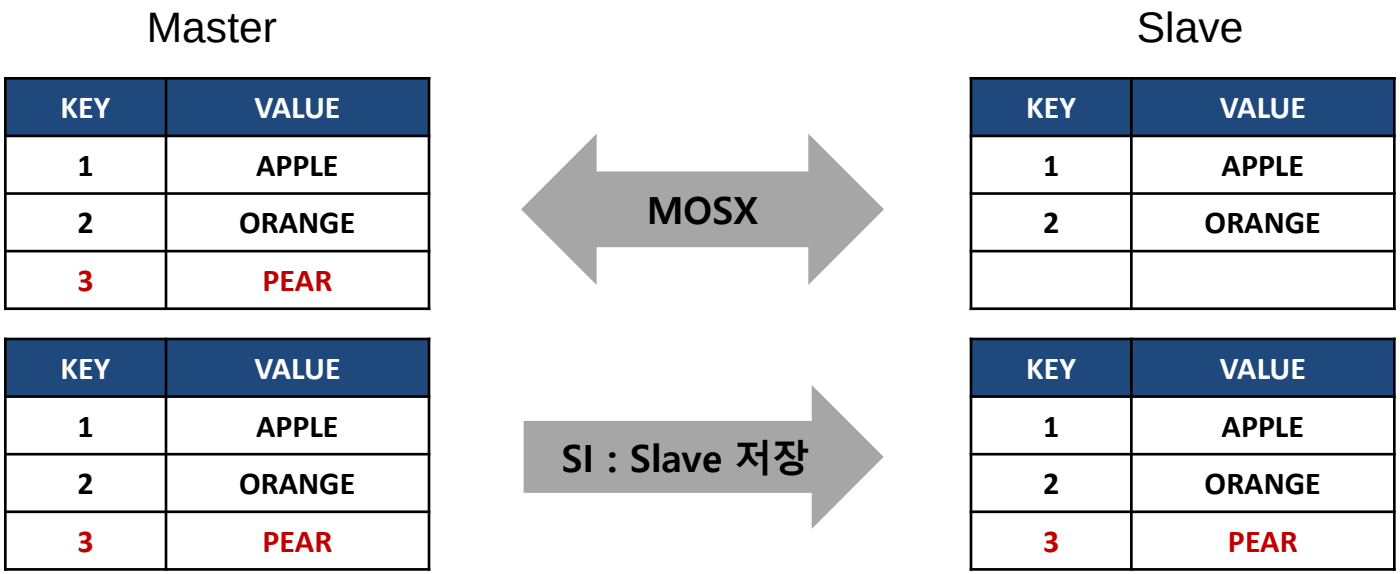
ALTICOMP(UTILITY)

❖ 데이터 동기화 정책

- altiComp 이후 Master 데이터베이스와 Slave 데이터베이스간의 데이터 불일치가 발생하는 세 가지 경우에 따라 다음의 데이터 동기화 정책 사용

❖ SI(SLAVE DATABASE INSERT)

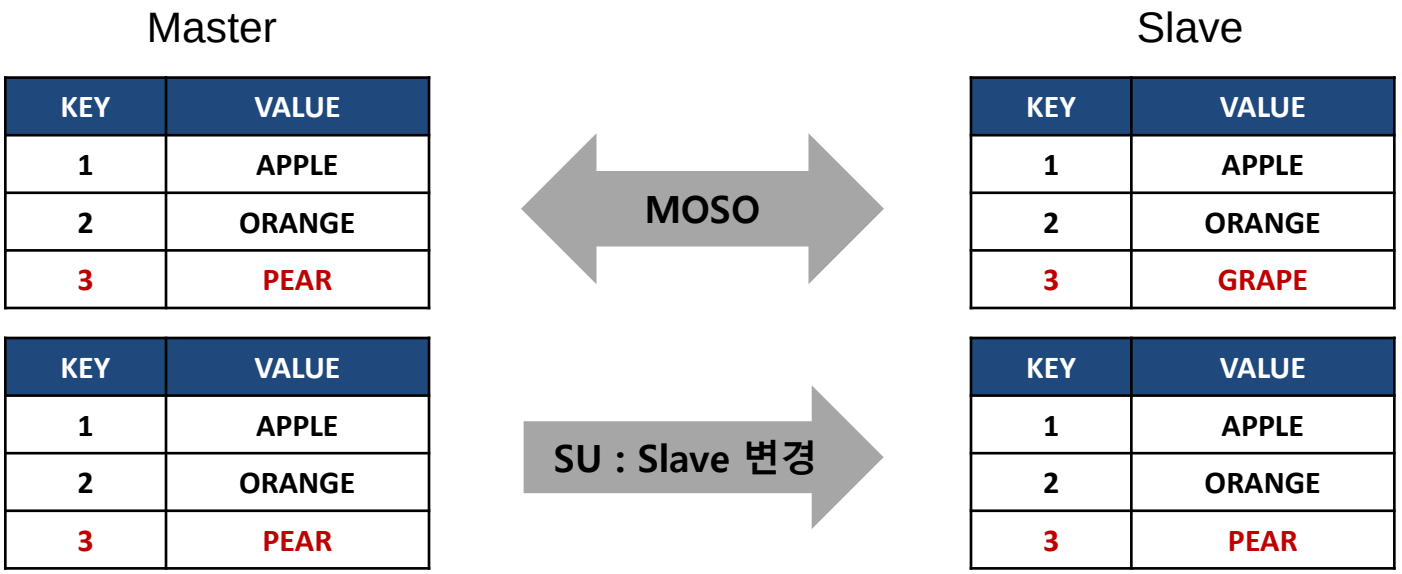
- MOSX 불일치를 해소하는 정책으로, Master 데이터베이스의 레코드를 Slave 데이터베이스에 삽입(Insert)



ALTICOMP(UTILITY)

❖ SU(SLAVE DATABASE UPDATE)

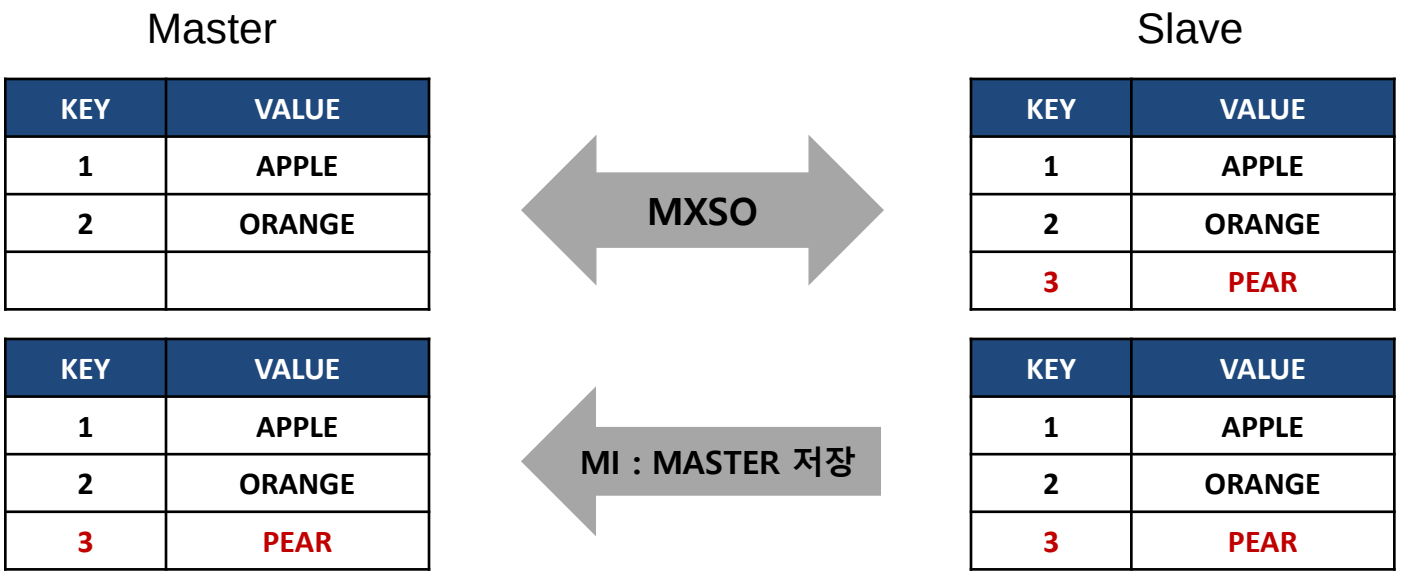
- MOSO 불일치를 해소하는 정책으로, Master 데이터베이스의 레코드 내용으로 Slave 데이터베이스 변경(Update)



ALTICOMP(UTILITY)

❖ MI(MASTER DATABASE INSERT)

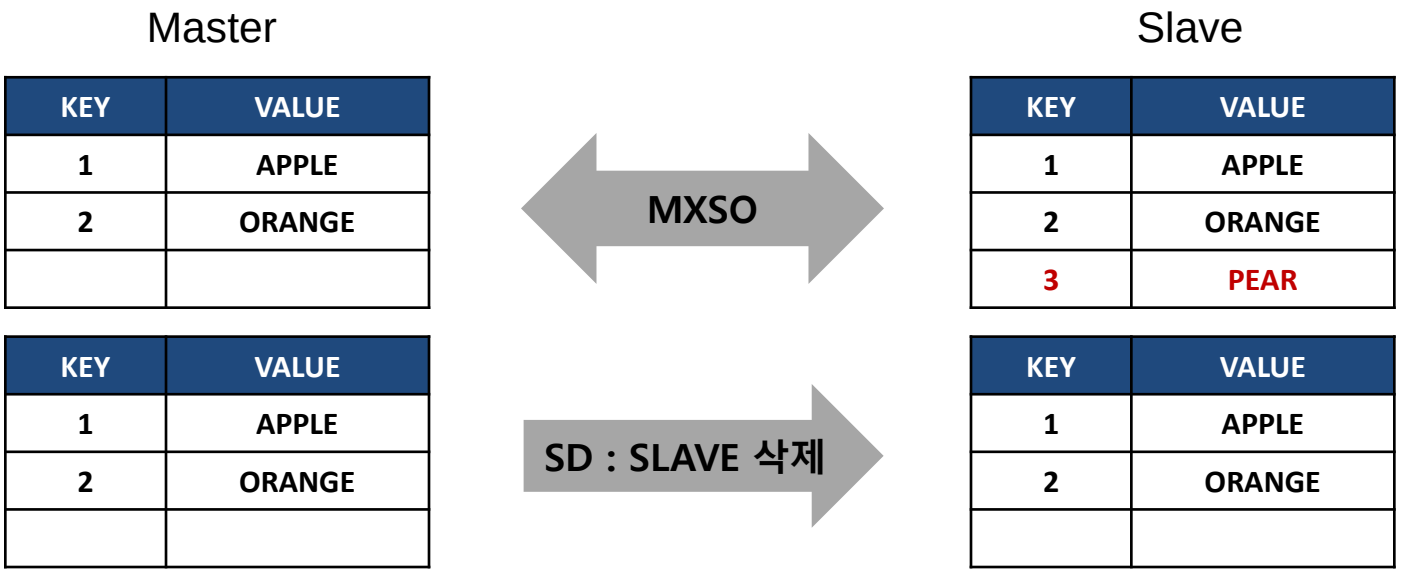
- MXSO 불일치를 해소하는 정책으로, Slave 데이터베이스의 레코드를 Master 데이터베이스에 삽입(Insert)



ALTICOMP(UTILITY)

❖ SD(SLAVE DATABASE DELETE)

- ➢ MXSO 불일치를 해소하는 정책으로, Slave 데이터베이스의 레코드 삭제(Delete)



❖ altiComp 구성

- altiComp 실행 시 서로 다른 두 개의 데이터베이스 간에 구성될 altiComp 정책을 기술한 설정 파일이 필요
- altiComp 설정 파일
  - altiComp 을 실행하기 위한 옵션을 지정하는 설정 파일
  - Connection정보, altiComp 기능 설정, 일치 정책 등의 내용 포함
  - $ALTIBASE_HOME/altiComp/altiComp.cfg 샘플 파일 참조
- 비교(DIFF) 기능
  - Master 데이터베이스와 Slave 데이터베이스의 불일치 데이터를 식별하여 결과 파일 생성
- 일치(SYNC)기능
  - 두 데이터베이스 간의 불일치를 해소
  - 불일치 데이터에 대한 결과 파일 생성

❖ 비교(DIFF) 기능

➢ Master 데이터베이스와 Slave 데이터베이스 간의 이중화 동작에서 발생한 불일치 데이터를 비교한 결과 파일 생성

```
# 비교 작업을 위한 감사 환경 파일 설정 시작

# Master 데이터베이스 접속 정보 설정
DB_MASTER=“ altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=MS949 ”

# Slave 데이터베이스 접속 정보 설정
DB_SLAVE=“ altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=MS949 ”

# Audit 작업 종류 기술 (비교 작업이므로 DIFF 설정)
OPERATION = DIFF

# 스레드 수를 지정 (무제한)
MAX_THREAD = -1
```

```
# 불일치 감사 정책 설정 (diff 의 경우 설정 의미 없음)
# DELETE_IN_SLAVE = ON
# INSERT_TO_SLAVE = ON
# INSERT_TO_MASTER = OFF
# UPDATE_TO_SLAVE = ON

# 실행 결과 파일이 생성될 위치 지정
LOG_DIR = “ ./ ”
LOG_FILE = “ sample.log ”

# Alticomp 대상 테이블 매칭 설정
# 마스터 테이블[ EMP]과 슬레이브 테이블 EMPLOYEE 와 비교
[EMP]
TABLE = EMPLOYEE
SCHEMA = SYS

# 마스터 테이블[DEPT] 와 슬레이브 테이블 DEPARTMENT 와 비교
[DEPT]
TABLE = DEPARTMENT
SCHEMA = SYS

# 감사 환경 파일 설정 끝
```

❖ altiComp 명령 실행(DIFF)

```
$ alticomp  - f sample.cfg
```

➢ 마스터테이블-사용자명.슬래이브 테이블.log' 내용

- (1) MOSX 일 경우

```
$ cat EMP-SYS.EMPLOYEE.log
MOSX[19,15]->ENO(19):PK->{19}
MOSX[20,15]->ENO(20):PK->{20}
```

- (2) MOSO 일 경우

```
$ cat EMP-SYS.EMPLOYEE.log
MOSO[10,10]->ENAME('JJLEE                ','YHBAE                ):PK->{10}
MOSO[11,11]->ENAME( ' MJYOO               ','MSKIM                ):PK->{11}
```

- (3) MXSO 일 경우

```
$ cat EMP-SYS.EMPLOYEE.log
MXSO[8,8]->ENO(8):PK->{8}
MXSO[8,9]->ENO(9):PK->{9}
```

❖ 일치(SYNC) 기능

➢ Master 데이터베이스와 Slave 데이터베이스간의 불일치 데이터를 식별하여 altiComp 환경 파일의 일치 정책에 따라 불일치 해소 기능

```
# 비교 작업을 위한 altiComp 환경 파일 설정 시작

# Master 데이터베이스 접속 정보 설정
DB_MASTER= " altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=MS949 "

# Slave 데이터베이스 접속 정보 설정
DB_SLAVE= " altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=MS949 "

# Audit 작업 종류 기술 (일치 작업이므로 SYNC 설정)
OPERATION = SYNC

# 스레드 수를 지정 (무제한)
MAX_THREAD = -1
```

ALTICOMP

```
# 불일치 ALTICOMP정책 설정
DELETE_IN_SLAVE = ON
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = OFF
UPDATE_TO_SLAVE = ON

# 실행 결과 파일이 생성될 위치 지정
LOG_DIR = “ ./ ”
LOG_FILE = “ sample.log ”

# ALTICOMP 대상 테이블 매칭 설정
# 마스터 테이블[ EMP]과 슬레이브 테이블 EMPLOYEE 와 비교
[EMP]
TABLE = EMPLOYEE
SCHEMA = SYS

# 마스터 테이블[DEPT]과 슬레이브 테이블 DEPARTMENT와 비교
[DEPT]
TABLE = DEPARTMENT
SCHEMA = SYS

# 감사 환경 파일 설정 끝
```

❖ altiComp 명령 실행(SYNC)

```
$ alticomp  -f sample.cfg
```

- 실행 결과 파일 내용은 다음과 같으며, 만일 실패한 레코드가 있다면 로그 파일에 원인과 레코드 내용 기록
  - sample.log 파일 내용

```
INFO[ MNG ] Tread # 0 init is OK!
INFO[ MNG ] Tread # 0 start is OK!

[EMP->EMPLOYEE]
Fetch Rec In Master: 3
Fetch Rec In Slave : 2
MOSX = -, SI
MXSO = -, SD
MOSO = -, SU
```

ALTICOMP

```
-------------------------------------------------------------
Operation    Type       MASTER    SLAVE
-------------------------------------------------------------
INSERT         Try             0               1
               Fail            0               0

UPDATE        Try             X               1
               Fail           X               0

DELETE        Try             X               0
               Fail           X               0
-------------------------------------------------------------
UPDATE        Try             0               2
               Fail           0               0

OOP   TPS: 13698.63
SCAN TPS: 20547.95

Time: 0.00 sec
```

❖ 이중화 환경에서의 일치(SYNC) 작업 절차

- Application 서비스 중지
  - AUDIT SYNC 중 변경 TRANSACTION 발생시 데이터가 불일치 될 수 있음
- 이중화 갭 확인
  - 이중화 로그 원격 SERVER에 모두 반영 되었는지 (rep_gap=0) 확인

```
iSQL> SELECT rep_gap FROM V$REPGAP;
```

- 이중화 중지

```
iSQL> ALTER REPLICATION replication_name  STOP;
```

- altiComp 수행
  - REPLICATION QUICKSTART 명령으로 Alticomp 로 반영된 TRANSACTION 로그가 이중화로 전송되는 것을 방지

```
iSQL> ALTER REPLICATION replication_name  QUICKSTART;
```

❖ altiComp 사용시 유의 사항

- PK가 없을 경우 Log 파일에 아래 에러 발생
  - FATAL[TASK] Process failure! [SCANER]: [ERR-910D8 : No Primary Key Column exist (T1:T1)]
- SD와 MI 정책 충돌 시 아래 에러 발생
  - Invalid Property Value SD and MI Incompatible was defined.
- 데이터 타입이 다를 경우 데이터 값이 같더라도 다르게 인식
  - char(10) vs. varchar(10)

REPLICATION

REPLICATION MONITORING

REPLICATION MONITORING

❖ DBMS 모니터링 분류

분류	설명
내부 모니터링	메타테이블 및 성능뷰에 대한 SQL를 통하여 DBMS 내부 요소를 모니터링
외부 모니터링	OS 명령어를 통하여 OS차원에서 DBMS 관련된 외부 요소를 모니터링
trace 로그 모니터링	DBMS에서 발생하는 각종 로그를 모니터링

❖ 모니터링 방법

- 관련 명령어를 수행하는 쉘 스크립트 작성하여 주기적으로 수행
  - 유틸리티 활용 (교육자료 2권 참조)
    - ALTIMON
  - 별도의 응용프로그램 작성

REPLICATION MONITORING

❖ 이중화 관련 주요 내부 모니터링 요소

- V$REPGAP
- V$REPSENDER
- V$REPRECEIVER

❖ 이중화 관련 주요 외부 모니터링 요소

- NETWORK
- 리두 로그 파일 시스템
- ALTIBASE 구동 상태
- OS 구동 상태

❖ 이중화 관련 주요 trace 로그 요소

- 이중화 trace 로그 파일

REPLICATION MONITORING

❖ 추가적인 내부 모니터링 요소

- 이중화에 직접적인 관련이 있는 요소는 아니나 영향을 줄 수 있는 항목
  - 벌크(BULK)성 UPDATE/DELETE 쿼리 수행
    - 성능뷰 V$STATEMENT, V$SESSION등을 활용
  - 장시간 수행되는 변경 연산 TRANSACTION
    - 성능뷰 V$TRANSACTION,V$STATMENT, V$SESSION등을 활용

REPLICATION MONITORING

❖ 이중화 관련 메타 테이블

- ➢ 이중화 객체 생성 만으로도 관련 정보 조회 가능

메타 테이블	설명
SYS_REPLICATIONS_	**이중화 객체 정보**
SYS_REPL_HOSTS_	**이중화 객체 별 이중화 대상 IP 정보**
SYS_REPL_ITEMS_	**이중화 객체 별 이중화 대상 테이블 정보**
SYS_REPL_OFFLINE_DIR_	오프라인 이중화 옵션 정보 (5.3.3 higher)
SYS_REPL_OLD_COLUMNS_	sender가 현재 사용중인 이중화 대상 칼럼 정보
SYS_REPL_OLD_INDEX_COLUMNS_	sender가 현재 사용중인 이중화 대상 인덱스 칼럼 정보
SYS_REPL_OLD_INDICES_	sender가 현재 사용중인 이중화 대상 인덱스 정보
SYS_REPL_OLD_ITEMS_	sender가 현재 사용중인 이중화 대상 테이블 정보
SYS_REPL_RECOVERY_INFOS_	이중화 복구를 위한 로그 정보 메타 테이블

REPLICATION MONITORING

❖ SYS_REPLICATIONS_

Column Name	설명
REPLICATION_NAME	이중화 이름
IS_STARTED	이중화 시작 여부
XSN	송신자가 Xlog 전송을 재개할 재시작 SN
ITEM_COUNT	이중화 대상 테이블 개수
CONFLICT_RESOLUTION	이중화 충돌 해결 방법
REPL_MODE	기본 이중화 모드

❖ SYS_REPL_HOSTS_

Column Name	설명
HOST_NO	호스트 식별자
REPLICATION_NAME	이중화 이름
HOST_IP	원격 SERVER IP 주소
PORT_NO	원격 SERVER 이중화 포트 번호

REPLICATION MONITORING

❖ SYS_REPL_ITEMS_

Column Name	설명
REPLICATION_NAME	이중화 이름
TABLE_OID	테이블 객체 식별자
LOCAL_USER_NAME	지역 SERVER의 대상 테이블 소유자 이름
LOCAL_TABLE_NAME	지역 SERVER의 대상 테이블 이름
REMOTE_USER_NAME	원격 SERVER의 대상 테이블 소유자 이름
REMOTE_TABLE_NAME	원격 SERVER의 대상 테이블 이름

REPLICATION MONITORING

❖ 이중화 관련 성능 뷰

- ➢ 이중화를 운영하여 sender 및 receiver가 활성화되었을 때만 조회 가능
  - v$repgap에서 제공하는 이중화 갭 수치는 sender가 구동중인 상태에서만 조회 가능

성능뷰	설명
v$repgap	**이중화갭 정보**
v$repsender	**sender 정보**
v$repsender_transtbl	sender의 TRANSACTION 테이블 정보
v$repreceiver	**receiver 정보**
v$repreceiver_column	receiver의 이중화 대상 칼럼 정보
v$repreceiver_transtbl	receiver의 TRANSACTION 테이블 정보
v$repsync	테이블 복제를 수행중인 테이블의 정보
v$repoffline_status	오프라인 이중화의 수행 상태 정보 (5.3.3 higher)
v$repexec	이중화 관리자 정보
v$replogbuffer	이중화 전용 로그 버퍼의 정보

REPLICATION MONITORING

❖ V$REPGAP(~ version 6.5.1)

Column Name	설명
REP_NAME	이중화 객체 이름
REP_LAST_SN	마지막 로그 레코드의 일련번호
REP_SN	현재 전송중인 로그 레코드 일련번호
REP_GAP	REP_LAST_SN과 REP_SN 차이
READ_FILE_NO	현재 읽고 있는 로그 파일 번호

❖ V$REPGAP(version 7.1.0 ~)

Column Name	설명
REP_NAME	이중화 객체 이름
REP_LAST_SN	마지막 로그 레코드 일련번호
REP_SN	현재 전송중인 로그 레코드 식별번호
REP_GAP	현재 전송중인 로그 레코드부터 마지막 로그 레코드까지 크기 (기본값 MB단위) REPLICATION_GAP_UNIT으로 단위 조정 가능
REP_GAP_SIZE	REPLICATION GAP 사이즈 (byte)

REPLICATION MONITORING

❖ V$REPSENDER

Column Name	설명
REP_NAME	이중화 객체 이름
XSN	현재 송신중인 로그 레코드의 SN
COMMIT_XSN	Commit 로그 레코드의 SN
STATUS	현재 상태
SENDER_IP	송신자 IP 주소
PEER_IP	원격 SERVER의 IP 주소
REPL_MODE	사용자가 지정한 이중화 모드

❖ V$REPRECEIVER

Column Name	설명
REP_NAME	이중화 객체 이름
MY_IP	지역SERVER의 IP 주소
PEER_IP	원격SERVER의 IP 주소
APPLY_XSN	처리중인 XSN

REPLICATION MONITORING

❖ 이중화 모니터링 예제

➢ 로컬SERVER

```
iSQL> SELECT rep_name, rep_sn, rep_last_sn, rep_gap, read_file_no, start_flag FROM V$REPGAP;
REP_NAME          REP_SN        REP_LAST_SN     REP_GAP       READ_FILE_NO    START_FLAG
------------------------------------------------------------------------------------------
REP1              5178653       7070623         2             1               0
1 row selected.

iSQL> SELECT rep_name, xsn, status, repl_mode FROM V$REPSENDER;
REP_NAME          XSN           STATUS          REPL_MODE
------------------------------------------------------------
REP1              5178653       1               LAZY
1 row selected.

iSQL> SELECT replication_name, xsn, is_started FROM SYSTEM_.SYS_REPLICATIONS_;
REPLICATION_NAME  XSN           IS_STARTED
-------------------------------------------
REP1              5178290       1
1 row selected.

iSQL> exit
$
$ ls $ALTIBASE_HOME/logs
do_not_remove_log_files  loganchor2  logfile2  logfile5  never_remove_log_files
loganchor0               logfile0    logfile3  logfile6
loganchor1               logfile1    logfile4  logfile7
```

REPLICATION MONITORING

❖ 이중화 모니터링 예제

➢ 원격SERVER

```
iSQL> SELECT rep_name, apply_xsn FROM V$REPRECEIVER;
REP_NAME          APPLY_XSN
---------------------------
REP1              5178290
```

REPLICATION MONITORING

❖ 대표적인 이중화 장애 상황에 대응되는 모니터링

- ➢ 모든 모니터링 요소는 이중화 갭(V$REPGAP) 모니터링 만으로도 간접적으로 감지 가능

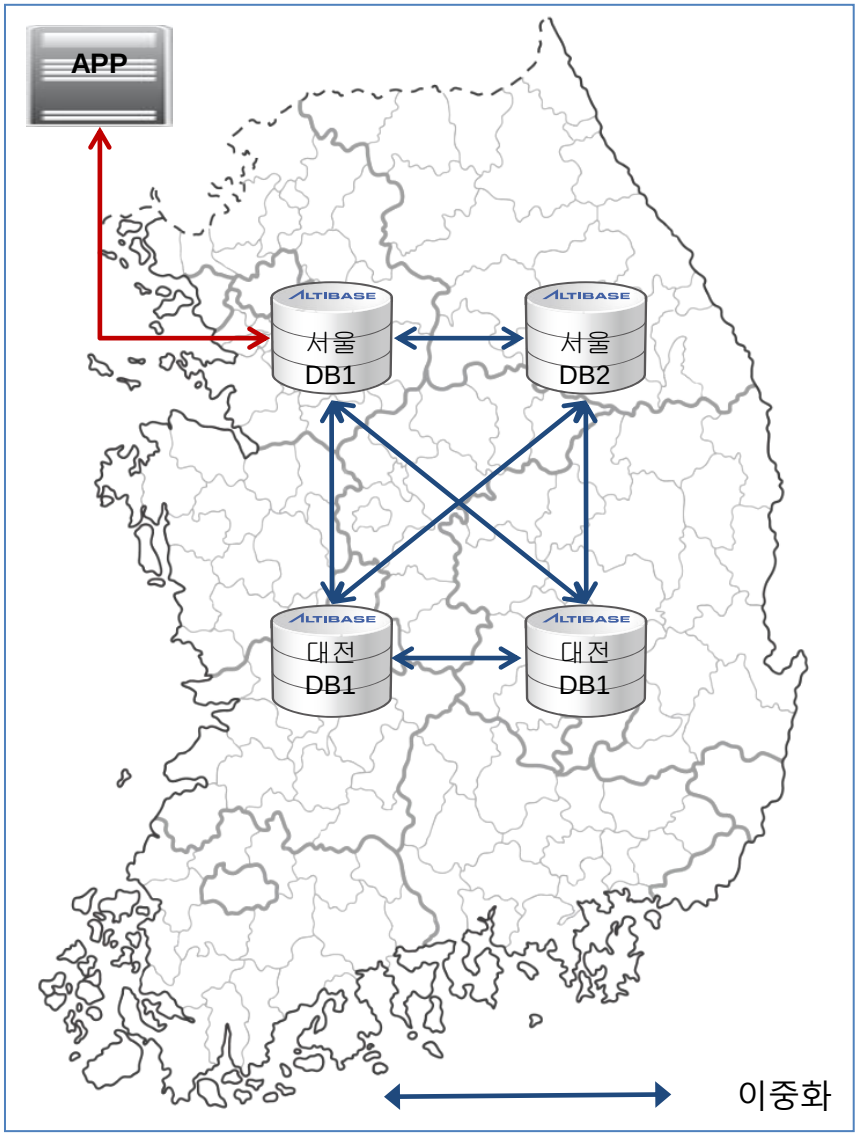
원인	상황	모니터링 요소	대상
대량 변경연산	대량(BULK)의 UPDATE/DELETE 수행	대량 변경연산 TRANSACTION	지역SERVER
Network 장애	일시적/주기적인 Network 결함 Network 단절	Network	지역SERVER, 원격SERVER
송신불가	지역SERVER의 이중화가 중지(STOP)된 상태	Sender	지역Server
수신불가	원격SERVER의 ALTIBASE 구동 중지	ALTIBASE SERVER	원격SERVER
	원격SERVER의 OS failure	OS	
	원격SERVER의 리두 로그 파일 관련 파일시스템의 full (1) 장시간 수행되는 원격SERVER의 변경 TRANSACTION으로 인한 대량의 리두 로그 파일 생성과 같은 ALTIBASE 내부 요인	장시간 수행되는 변경연산 TRANSACTION	
	원격SERVER의 리두로그파일 관련 파일시스템의 full (2) 사용자가 임의로 다른 작업을 위해 해당 파일시스템의 공간을 사용 중인 경우와 같은 ALTIBASE 외부 요인	리두 로그 파일시스템	

REPLICATION CASE

REPLICATION CASE

❖ 이중화 적용 사례 1

- ➢ 통신사 인증서비스
  - ▪ 주요특징
    - ◆ 총 4대 구성 Full-Mesh 이중화
    - ◆ 서울 대전간 이중화 구성
    - ◆ 이중화 SYNC를 이용한 무정지 구축

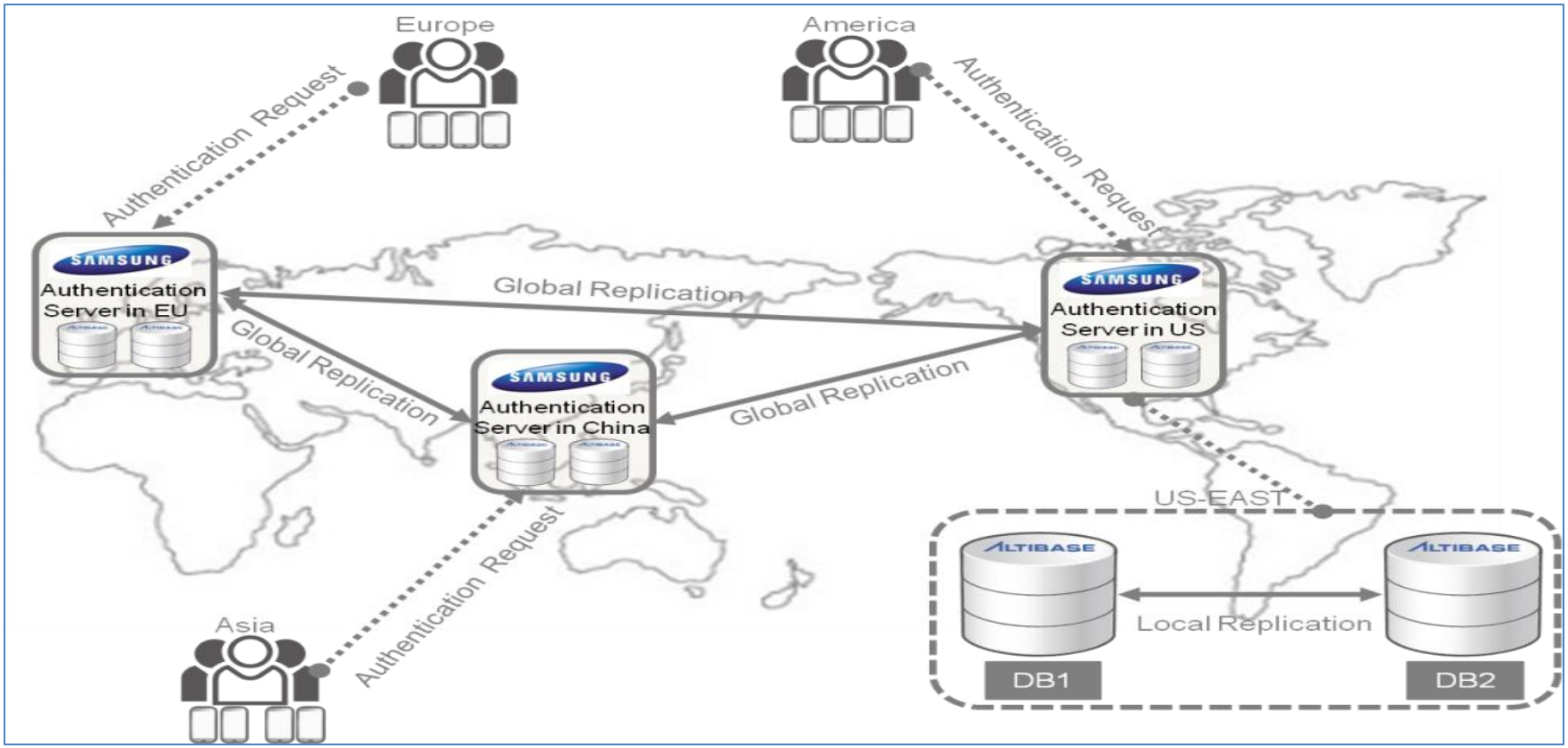


REPLICATION CASE

❖ 이중화 적용 사례 2

➢ 삼성전자 글로벌인증서비스

- 주요특징
  - 전세계 판매된 스마트 기기의 인증 처리를 위한 이중화 구성 구축
  - 대륙간 이중화 구성을 통해 무정지 서비스 제공



Thank you!

ALTIBASE