ALTIBASE HDB ADMINISTRATION I

ALTIBASE

ALTIBASE HDB ADMINISTRATION I

INTRODUCTION to ALTIBASE HDB

ALTIBASE HDB CONCEPT
타 DBMS와의 비교
INSTALLATION
STARTUP & STOP
iSQL

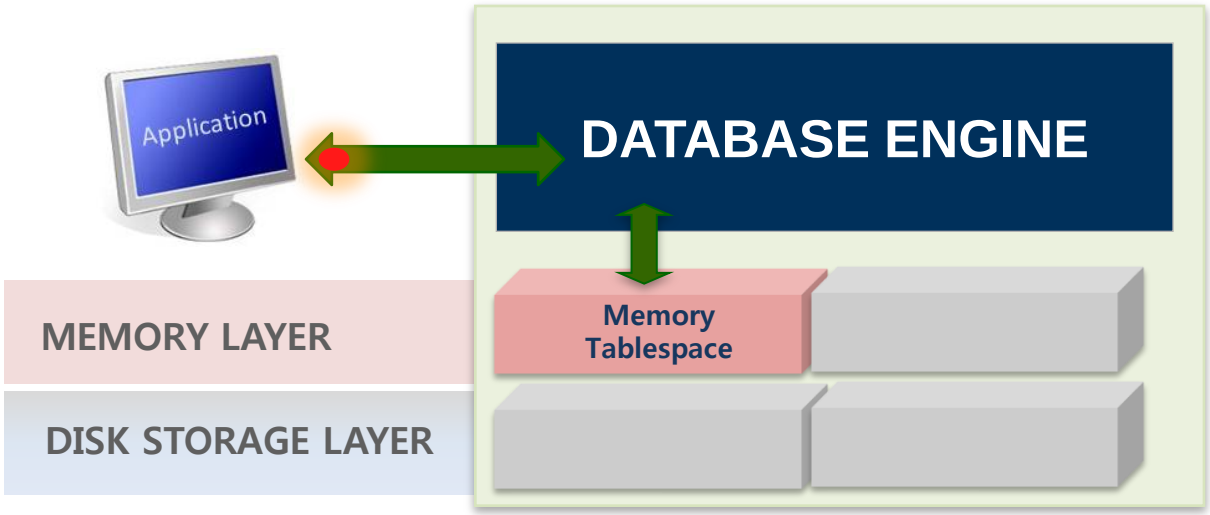
ALTIBASE

ALTIBASE HDB CONCEPT

DBMS 특징

❖메모리 DBMS의 특징

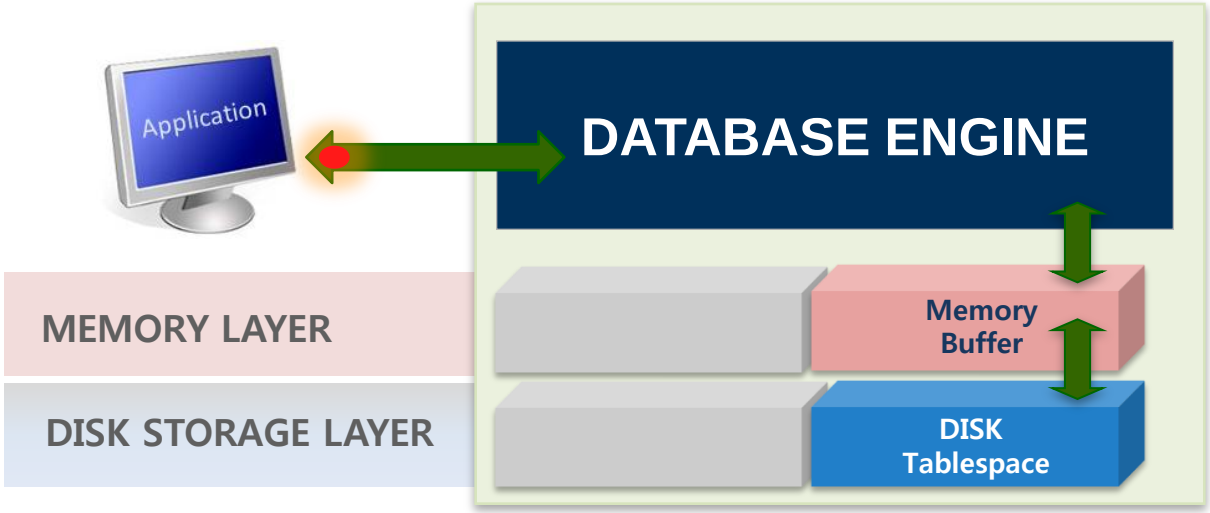
- ➢ 빠른 데이터 처리 성능
  - 데이터와 인덱스를 모두 메모리에 저장하여 처리하므로 빠른 데이터 처리 가능
  - 메모리 인덱스는 RID가 아닌 Physical한 포인터로 관리하기 때문에 액세스가 빠름
  - 디스크 I/O로 인한 성능 저하가 거의 발생하지 않음
  - OLTP성 업무에 사용하기 적합
- ➢ 물리적 메모리의 크기만큼만 데이터를 적재할 수 있는 제약사항이 있음



DBMS 특징

❖디스크 DBMS의 특징

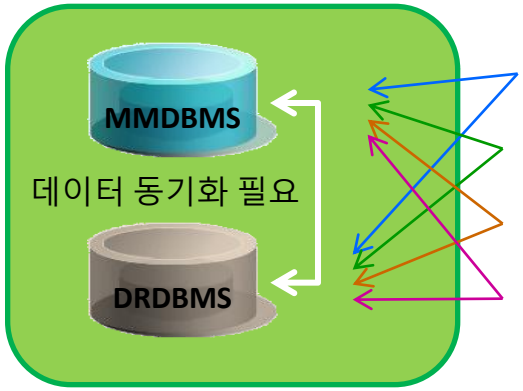
- ➢ 대용량 데이터 처리
  - 메모리 DBMS에 비해 데이터의 저장 공간에 대한 제약이 거의 없음
  - History성 데이터, DW 용으로 사용하기 적합
- ➢ 처리 성능에 제한
  - 디스크I/O로 인한 성능 저하 발생
  - Buffer에 데이터 상주시켜도 인덱스는 디스크에 생성됨



Hybrid DBMS 도입 배경

❖Hybrid DBMS를 통한 효율성 증대

일반적 MMDBMS의 적용(혼용 구조)



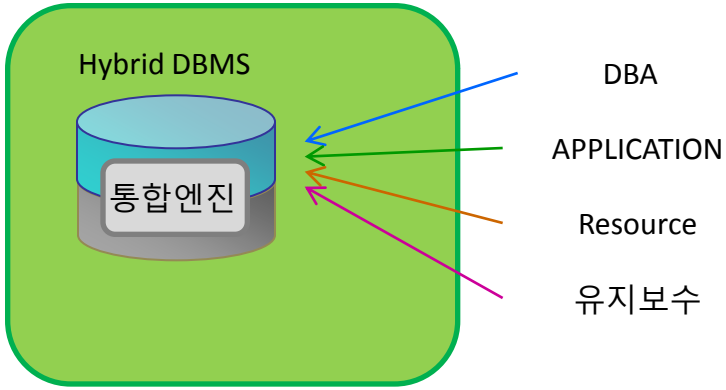
관리에서 운영, 구성적인 모든 비용과 노력들이 두 개의 DBMS를 전부 제어 해야만 함

관리적 비용 : DRDBMS + MMDBMS 비용 추가

운영적 비용 : DRDBMS + MMDBMS 비용 추가

구성적 비용 : DRDBMS + MMDBMS 비용 추가

Hybrid DBMS의 적용



메모리와 디스크 저장장치를 하나의 엔진에서 완벽하게 융합되어 제공함으로 효율성 극대화

관리적 비용 : Hybrid DBMS만을 관리 [비용 ½ 절감]

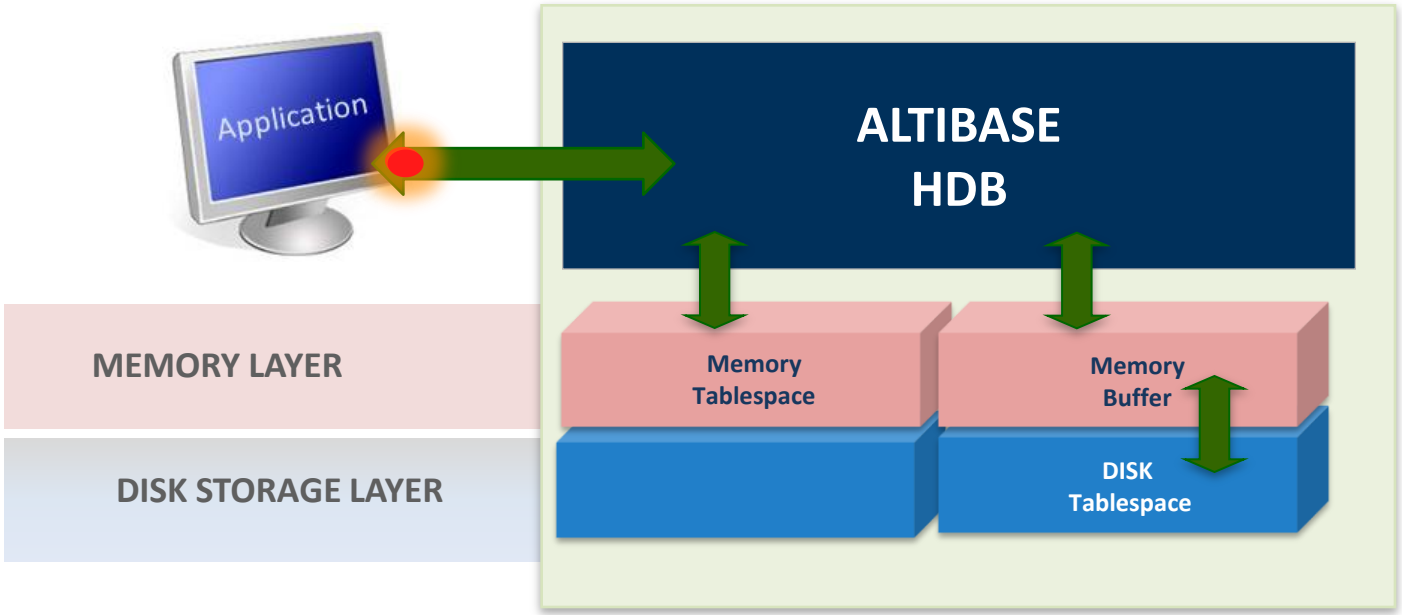
운영적 비용 : Hybrid DBMS만을 관리 [비용 ½ 절감]

구성적 비용 : Hybrid DBMS만을 관리 [비용 ½ 절감]

Hybrid DBMS

Hybrid DBMS Concept

- 사용자는 Memory DBMS, Disk DBMS 의 구분없이 하나의 DBMS만 접근



ALTIBASE

타 DBMS 와의 비교

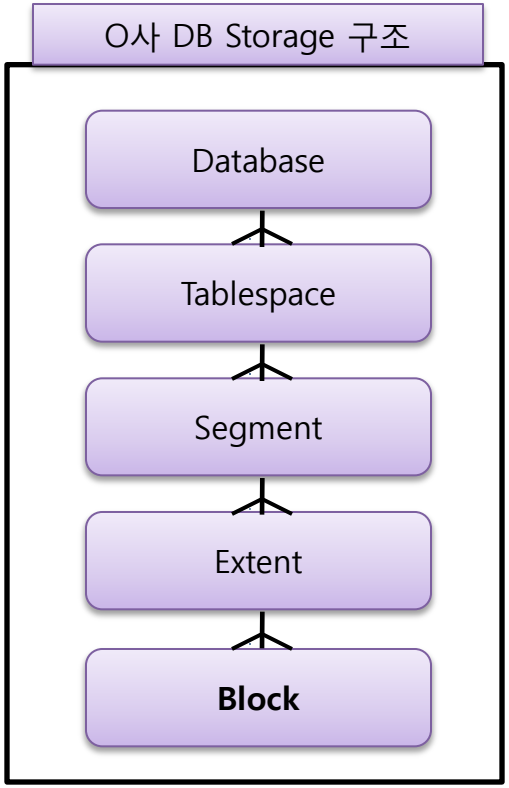
Architecture 비교

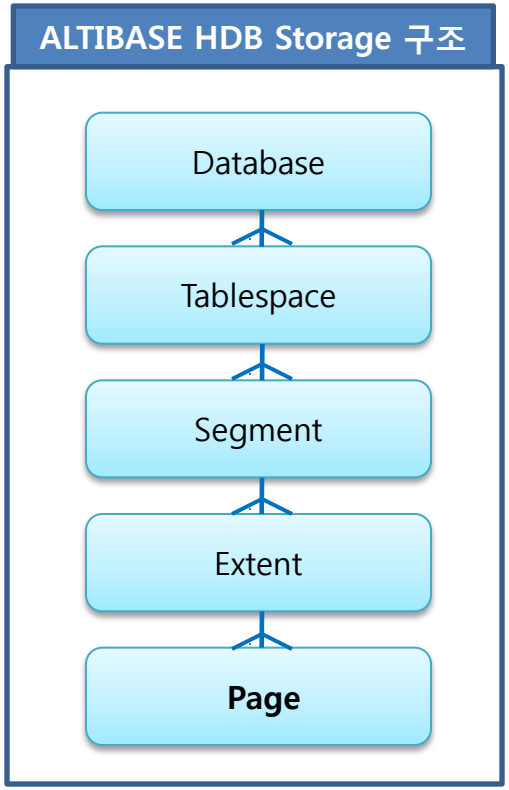
기능	O사 DB	ALTIBASE HDB	비고
DBMS Process구조	Multi Process 구조	Multi Thread 구조	
모 델	Relational DBMS 구조	Relational DBMS 구조	
아키텍처	Client-Server구조	Client-Server 구조	
High Availability 방식	데이터베이스 클러스터링 (RAC)	이중화 (Replication)	Replication 은 테이블의 데이터만 복제됨
	별개 Instance	별개 Instance	
	Storage 공유	Storage 별도	
	스키마 공유	스키마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

Architecture 비교

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

STORAGE 비교





기능	O사 DB	ALTIBASE HDB
관리 구조	데이터베이스(DB) 테이블스페이스 (Tablespace) 데이터파일 (Datafile) 세그먼트 (Segment) Extent Block	데이터베이스(DB) 테이블스페이스 (Tablespace) 데이터파일 (Datafile) 세그먼트 (Segment) Extent Page

일반 기능 비교

	ALTIBASE HDB	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 안됨	지원됨	
DB Link	지원됨	지원됨	

일반 기능 비교

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

SQL 비교

	ALTIBASE HDB	O사 DB	비고
SQL	표준 SQL (ANSI-SQL92지원)	표준 SQL , 변형 SQL (ANSI-SQL92, ANSI-SQL1999지원)	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 질의 CONNECT BY ~ WITH	지원됨	지원됨	
Array Processing	지원됨	지원됨	
Move 구문	지원됨	지원 안됨	

SQL 비교

	ALTIBASE HDB	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	지원 안됨	지원됨	
Parallel Insert	지원됨	지원됨	
Parallel Index Build	지원됨	지원됨	

ALTIBASE

New Features(6.3.1)

ALTIBASE

Job Scheduler

Job Scheduler

Job scheduler

- 저장 프로시저에 실행 일정을 결합한 JOB 객체를 제공
- 한 개의 JOB에는 한 개의 프로시저만 등록
- JOB_SCHEDULER_ENABLE 과 JOB_THREAD_COUNT 프로퍼티 설정 필요
- SYS 사용자만이 JOB을 생성, 변경, 삭제할 수 있음

Job scheduler 의 시작 및 종료

- Job scheduler 시작

```
iSQL> ALTER SYSTEM SET job_scheduler_enable = 1 ;
Alter  success.
iSQL> ALTER SYSTEM SET job_thread_count = 8 ;  => property 파일을 수정해야 함
Alter  success.
```

- Job scheduler 종료

```
iSQL> ALTER SYSTEM SET job_scheduler_enable = 0 ;
Alter  success.
```

작업 생성

❖구문

```
CREATE JOB job_name
EXEC procedure_name
START expr1
END expr2
INTERVAL number {YEAR | MONTH | DAY | HOUR | MINUTE} ;
```

❖예제

➢ 이름이 proc1인 프로시저가 현재부터 실행되어 1시간 주기로 작업을 실행 후, 3일 후에 끝나도록 JOB을 생성하라.

```
iSQL> CREATE JOB job1
      2 EXEC proc1
      3 START sysdate
      4 END sysdate + 3
      5 INTERVAL 1 HOUR;
Create  success.
```

작업 변경

❖구문

```
ALTER JOB job_name
SET  execute_procedure_statement  |
       START expr1 |
       END expr2 |
       INTERVAL number  {YEAR | MONTH | DAY | HOUR | MINUTE} ;
```

❖예제

➢ Job1이 실행되는 시작 시간을 '2014년 1월 1일' 로 변경하라

```
iSQL> ALTER JOB job1
      2 SET START to_date('20140101','YYYYMMDD');
Alter   success.
```

➢ Job1의 실행 주기를 10분 간격으로 변경하라

```
iSQL> ALTER JOB job1
      2 SET INTERVAL 10 MINUTE;
Alter   success.
```

작업 삭제

❖구문

```
DROP JOB job_name ;
```

❖예제

➢ 이름이 job1인 JOB을 제거하라

```
iSQL> DROP JOB job1;
Drop  success.
```

ALTIBASE

Function Based Index

함수 기반 인덱스

❖함수 기반 인덱스

- ➢ 함수 또는 수식의 결과 값을 기반으로 생성하는 인덱스
- ➢ WHERE절에 함수 또는 산술 표현을 자주 사용시 빠른 검색 속도를 보장
- ➢ QUERY_REWRITE_ENABLE 를 1로 변경한 후 인덱스 사용 가능

❖예제

```
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
Alter success.
iSQL> CREATE INDEX  NVL_IDX  ON  employee (NVL(salary, 0));
Create success.
 iSQL> SELECT eno, ename, salary FROM employee
      2 WHERE (NVL(salary, 0)) < 1000000;
ENO          ENAME                     SALARY
------------------------------------------------
1            EJJUNG
7            HJMIN                     500000
8            JDLEE
13           KWKIM                     980000
20           DIKIM
5 rows selected.
----------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 35, COST: 0.31 )
 SCAN ( TABLE: EMPLOYEE, FULL SCAN, ACCESS: 20, COST: 0.24 )
----------------------------------------------------------------
```

함수 기반 인덱스

예제

```
iSQL> ALTER SESSION SET QUERY_REWRITE_ENABLE = 1;
Alter success.
iSQL> SELECT eno, ename, salary FROM employee
      2 WHERE (NVL(salary, 0)) < 1000000;
ENO         ENAME                SALARY
-----------------------------------------------
1           EJJUNG
8           JDLEE
20          DIKIM
7           HJMIN                500000
13          KWKIM                980000
5 rows selected.
-------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 35, COST: 0.07 )
 SCAN ( TABLE: EMPLOYEE, INDEX: NVL_IDX, RANGE SCAN, ACCESS: 5, COST: 0.00 )
-------------------------------------------------------------
```

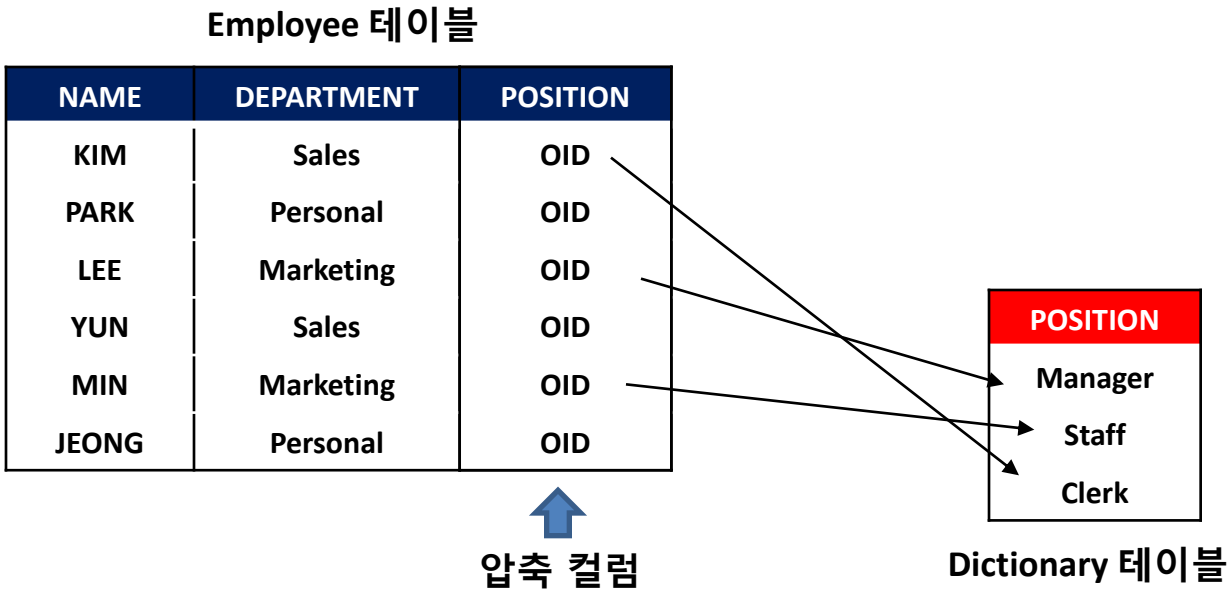
ALTIBASE

압축 컬럼

컬럼 압축

❖개념

- 압축 컬럼
  - 중복된 데이터를 딕셔너리 테이블에 별도로 저장하고, 그 데이터를 가리키는 OID로 대체함
- 압축 테이블
  - 압축 컬럼이 속해 있는 테이블
  - 압축 컬럼을 포함하는 테이블 생성 시, 서버에서 압축 컬럼 별로 딕셔너리 테이블과 유니크 인덱스를 자동으로 생성하여 관리함

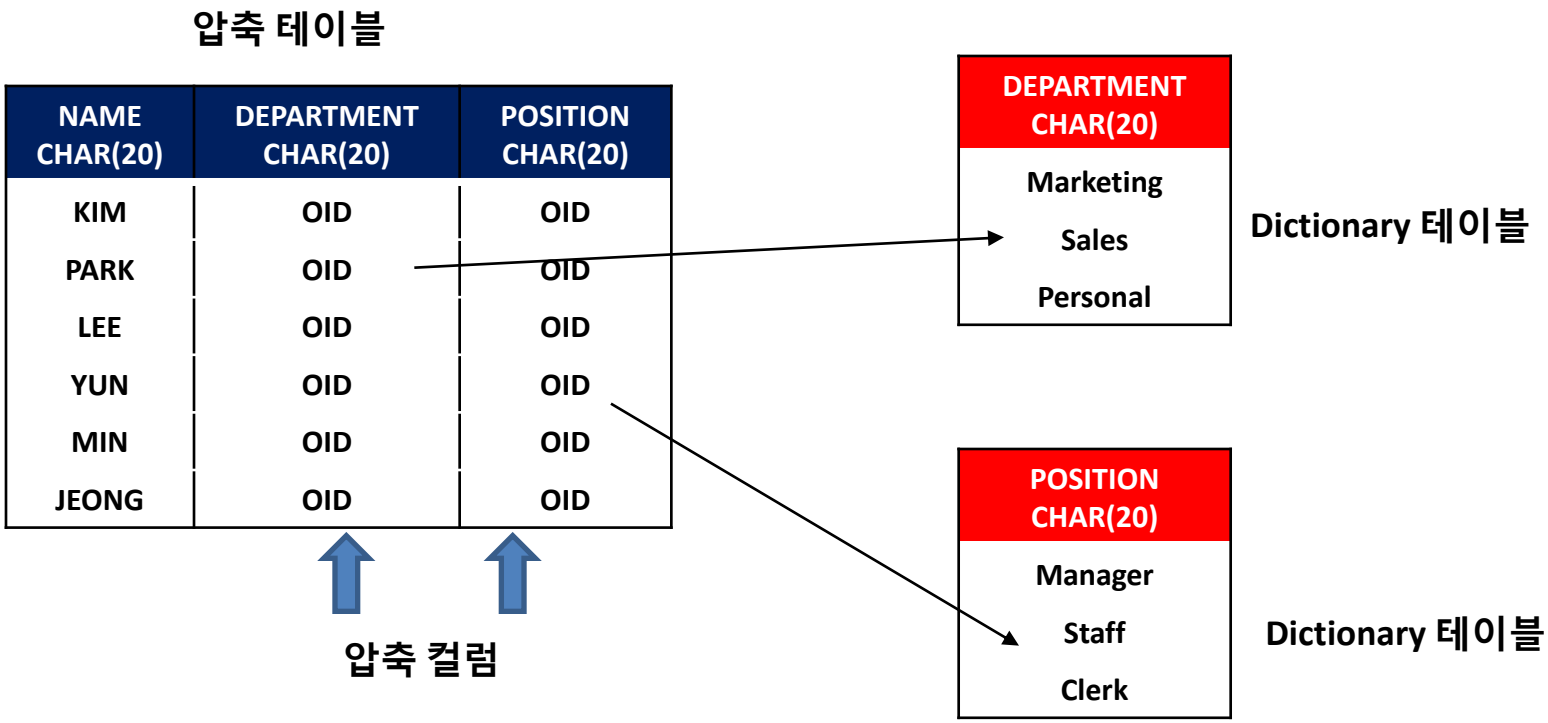


컬럼 압축

❖ 압축 테이블 생성

➢ 구문

```
iSQL> CREATE TABLE EMPLOYEE (
      2 NAME CHAR(20),
      3 DEPARTMENT CHAR(20),
      4 POSITION CHAR(20)
      5 ) COMPRESS ( DEPARTMENT, POSITION MAXROWS 100 );
```



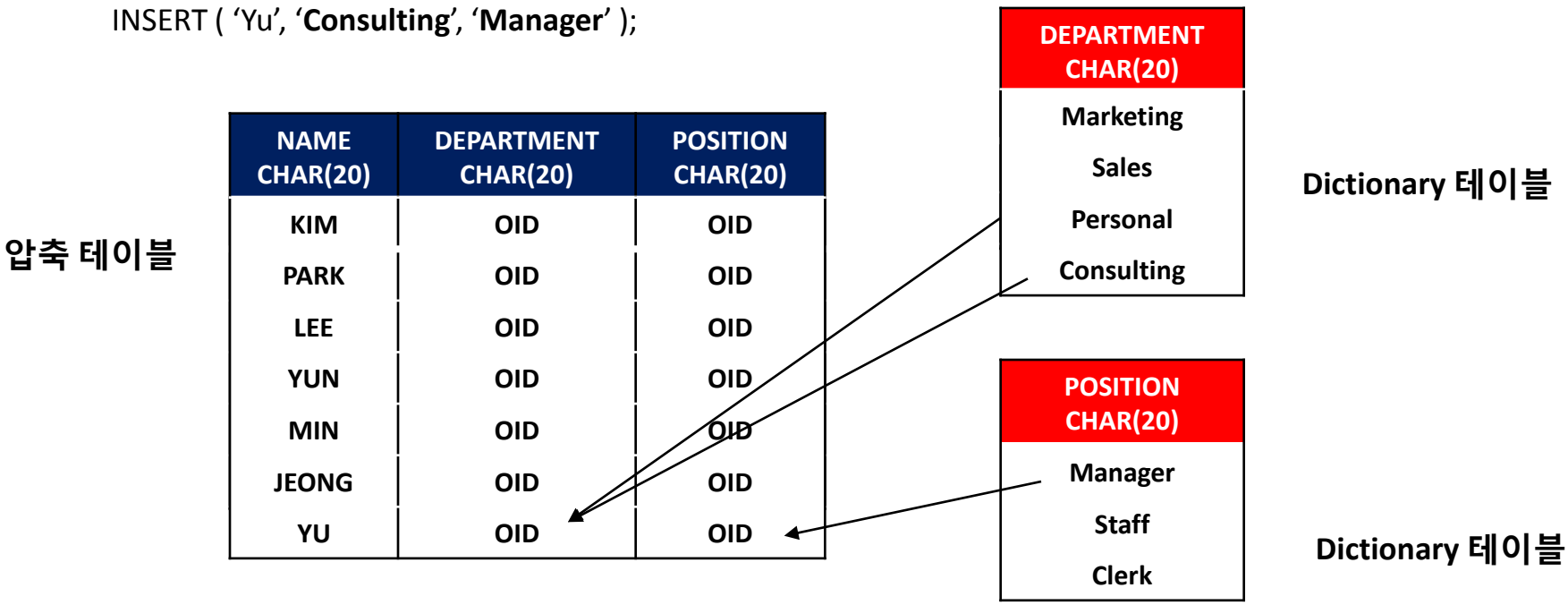
컬럼 압축

❖주의사항

- INSERT / UPDATE 시 주의사항
  - Insert 하는 값이 Dictionary 테이블에 존재하는지 조회 후,
    - 존재한다면 해당 OID 값을 저장함
    - 존재하지 않으면 Dictionary 테이블에 해당 값을 Insert 후, OID를 얻어서 테이블에 저장함 (성능저하 발생)

INSERT ('Yu', '**Sales**', '**Manager**');

INSERT ('Yu', '**Consulting**', '**Manager**');

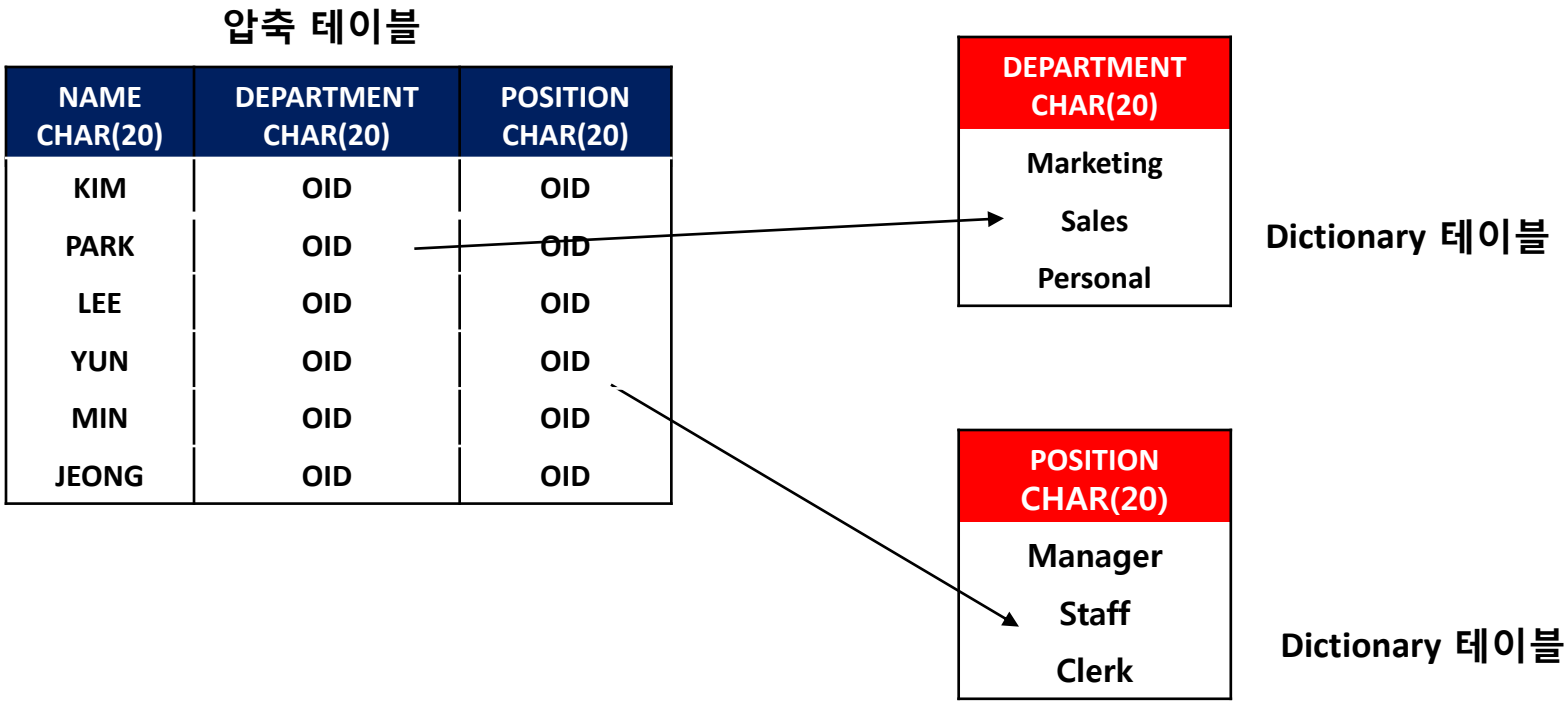


컬럼 압축

❖주의사항

- ➢ DELETE 시 주의사항
  - DELETE 실행 후, 딕셔너리 테이블에 unpointed data 가 남아 있을 수 있음
  - 테이블 Reorganize 작업을 통해 unpointed data 를 삭제할 수 있음

DELETE FROM EMPLOYEE ;

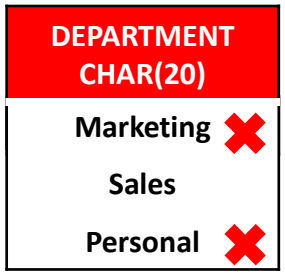


컬럼 압축

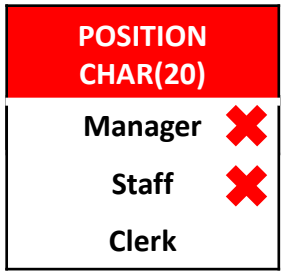
❖주의사항

- REORGANIZE
  - 테이블에 X Lock 을 획득 후, Reorganize 작업을 실행함
  - 딕셔너리 테이블과 인덱스의 unpointed 데이터를 삭제함

```
iSQL> ALTER TABLE EMPLOYEE
    2 REORGANIZE COLUMN ( DEPARTMENT, POSITION );
```



Dictionary 테이블



Dictionary 테이블

- 지원되는 Data Type
  - CHAR, VARCHAR, NCHAR, NVARCHAR, BYTE, BIT, VARBIT, NIBBLE, DATE
  - CHAR, VARCHAR 의 경우 precision이 8 이상인 경우에만 압축이 허용됨

컬럼 압축

❖주의사항

- ➢ 압축 컬럼 추가
  - ALTER TABLE ~ ADD COLUMN 구문을 사용하여 압축 컬럼을 추가할 수 있음
  - 컬럼 추가 없이 기존 컬럼 압축은 안됨(DROP COLUMN → ADD COLUMN)

ALTER TABLE EMPLOYEE
ADD COLUMN (JOIN_DATE CHAR(20)) **COMPRESS** (JOIN_DATE) ;

NAME CHAR(20)	DEPARTMENT CHAR(20)	POSITION CHAR(20)	JOIN_DATE CHAR(20)
KIM	Sales	Clerk	OID
PARK	Personal	Manager	OID
LEE	Marketing	Manager	OID
YUN	Sales	Clerk	OID
MIN	Marketing	Staff	OID
JEONG	Personal	Staff	OID

JOIN_DATE CHAR(20)
20120701
20131003

Dictionary 테이블

ALTIBASE

STARTUP & SHUTDOWN

STARTUP

❖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 프로세스를 구동
    - 프로퍼티 및 라이센스 파일의 유무 및 정합성을 체크
    - DB작업도 일체 수행할 수 없음 (단, CREATE/DROP DATABASE만 수행 가능)
  - CONTROL 단계
    - DB복구가 가능한 수준까지 내부 모듈을 준비
    - 복구 수행 및 데이터베이스 운영모드(archive모드)를 변경 가능함
    - DB 복구에 필요한 메타 및 성능 뷰를 조회할 수 있음
  - META 단계
    - DB 구동을 위해 메모리/디스크 데이터 파일의 정합성을 체크
    - Restart recovery를 수행
    - 내부 주요 쓰레드를 활성화
  - SERVICE 단계
    - 사용자가 DB를 사용할 수 있도록 모든 준비를 완료

STARTUP

❖구동하기

```
Shell::/home/alti1> isql –sysdba
----------------------------------------------------------------
      Altibase Client Query utility.
      Release Version 6.1.1.0.10
      Copyright 2000, ALTIBASE HDB 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 ---
```

STARTUP

❖구동 이후 프로세스 확인

- OS에서 제공하는 "ps" 명령을 통해 프로세스ID를 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 메모리의 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)
```

SHUTDOWN

❖SHUTDOWN 과정

- SHUTDOWN 과정은 STARTUP의 역순으로 진행
- STARTUP과 달리 단계의 전이는 존재하지 않음
- "SHUTDOWN" 구문을 통해 진행 가능
- 아래의 3가지 옵션에 따라 다른 동작을 수행
  - NORMAL 옵션
    - 접속한 세션이 정상 종료될 때까지 대기 후 STOP과정을 진행
  - IMMEDIATE 옵션
    - 접속한 세션을 강제 종료시킨 후 STOP과정을 진행
    - 강제 종료되는 세션의 트랜잭션은 모두 롤백처리
  - ABORT 옵션
    - "kill -9" 와 같이 프로세스를 강제로 종료시킴
    - 이 경우 정상 종료와 달리 재 구동 시에 restart recovery라는 자동복구과정이 진행됨

SHUTDOWN

❖종료하기

```
Shell::/home/alti1> isql –sysdba
----------------------------------------------------------------
      Altibase Client Query utility.
      Release Version 6.1.1.0.10
      Copyright 2000, ALTIBASE HDB 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.
```

SERVER SCRIPT

❖server 스크립트

- $ALTIBASE_HOME/bin/ 에 위치
- DB의 구동, 종료, 및 DB생성 등을 쉽게 할 수 있도록 제공되는 쉘 스크립트

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

SERVER SCRIPT

❖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
```

SERVER SCRIPT

❖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
```

SERVER SCRIPT

❖server 스크립트를 이용하는 구동

```
Shell::/home/alti1> server start
-----------------------------------------------------------------
      Altibase Client Query utility.
      Release Version 6.1.1.0.10
      Copyright 2000, ALTIBASE HDB 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.
```

SERVER SCRIPT

❖server 스크립트를 이용하여 종료

```
Shell::/home/alti1> server stop
-----------------------------------------------------------------
      Altibase Client Query utility.
      Release Version 6.1.1.0.10
      Copyright 2000, ALTIBASE HDB 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.
```

ALTIBASE

iSQL

iSQL

iSQL

- $ALTIBASE_HOME/bin 에 위치
- DB에 접속하여 질의 수행 및 결과를 조회하는 유틸리티
- DBA권한으로 DB 구동 및 종료, 백업 및 복구의 수행
- 세션의 강제 종료를 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 6.1.1.0.10
    Copyright 2000, ALTIBASE HDB 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 접속할 수 있도록 제공하는 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 HDB 서버가 위치한 IP를 지정
-u	ALTIBASE HDB 사용자 명을 지정
-p	DB 사용자의 패스워드를 지정
-port	ALTIBASE HDB Listen Port번호를 지정
-nls_use	ALTIBASE HDB 생성 시 입력한 문자셋을 지정
-o	iSQL에서 실행한 결과를 저장할 파일명을 지정
-f	iSQL에서 수행할 질의 및 명령을 저장한 입력 파일명을 지정
-h	입력 옵션에 대한 도움말을 출력
-silent	iSQL 배너를 화면에 출력하지 않게 할 경우 지정

iSQL

❖iSQL 실행 후 옵션 (1)

옵션	설명
desc	테이블 구성 정보를 확인
@	지정된 파일명을 실행
!	OS 명령을 수행하고자 할 경우
h	수행된 질의의 목록을 확인
/	직전에 수행한 질의를 재 수행
ed	직전에 수행한 질의를 편집하고자 할 경우
autocommit	현재 세션의 autocommit 모드를 변경할 경우
spool [fileName]	spool 명령에 입력된 파일에 현재 수행 결과를 기록
show all	iSQL의 현재 설정 상태 및 사용자 명을 출력
show user	iSQL에 접속한 사용자 명을 출력

iSQL

❖iSQL 실행 후 옵션 (2)

옵션	설명
colsize	컬럼사이즈의 길이를 지정
linesize	하나의 레코드의 출력라인의 길이를 지정
pagesize	지정된 개수만큼 레코드 출력 후 컬럼 타이틀을 출력
heading	출력 결과에서 컬럼 타이틀을 보이거나 감추도록 설정
timing	실행한 질의의 수행 시간을 1/100 초 단위로 출력
vertical	컬럼들을 세로로 출력 한 라인에 (컬럼 명 : Value) 형태로 결과를 출력
foreignkeys	desc 명령으로 테이블 조회 시 참조키 정보를 출력
querylogging	수행한 변경질의를 $ALTIBASE_HOME/trc/isql_query.log파일에 수행한 질의를 시간 순으로 저장

iSQL

glogin.sql

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

login.sql

- glogin.sql과 동일하며 iSQL을 실행하는 유저에게만 적용하고자 할 경우 사용
- iSQL을 실행하는 디렉토리에 login.sql 을 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 사용의 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 사용 예 : 실행계획을 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

❖ iSQL 사용 예 : 세션의 강제 종료

- ➢ 대상이 되는 세션의 고유번호를 확인해야 함

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

ALTIBASE HDB ADMINISTRATION I

ARCHITECTURES

FEATURES
ARCHITECTURE
TABLESPACE MANAGEMENT
TRANSACTION MANAGEMENT
BUFFER MANAGEMENT

ALTIBASE

FEATURES

Features

- **서버 클라이언트 엔진 구조**
  클라이언트-서버 프로토콜을 선택(TCP/IP, IPC, Unix Domain socket)
- **인터페이스**
  ANSI SQL 92 Full spec, ODBC 3.0, JDBC 2.0, C/C++, ADO.NET, C/C++ Precompiler 제공
- **다중버전 동시성 제어기법 (MVCC)**
  트랜잭션의 동시성 제어를 위해 다중 버전 동시성 제어 (MVCC, Multi-Version Concurrency Control) 기법을 사용
- **고립화 수준**
  트랜잭션 고립화 수준을 Read Commited(default), Repeatable Read, No Phantom Read(=Serializable) 지정
- **Fuzzy/Ping-Pong Checkpoint 제공**
  Checkpoint 수행 중에 발생한 트랜잭션을 처리할 수 있도록 Fuzzy Checkpoint를 제공하며 Checkpoint 시에도 트랜잭션 처리를 빠르게 하는 Ping-Pong Checkpoint를 제공

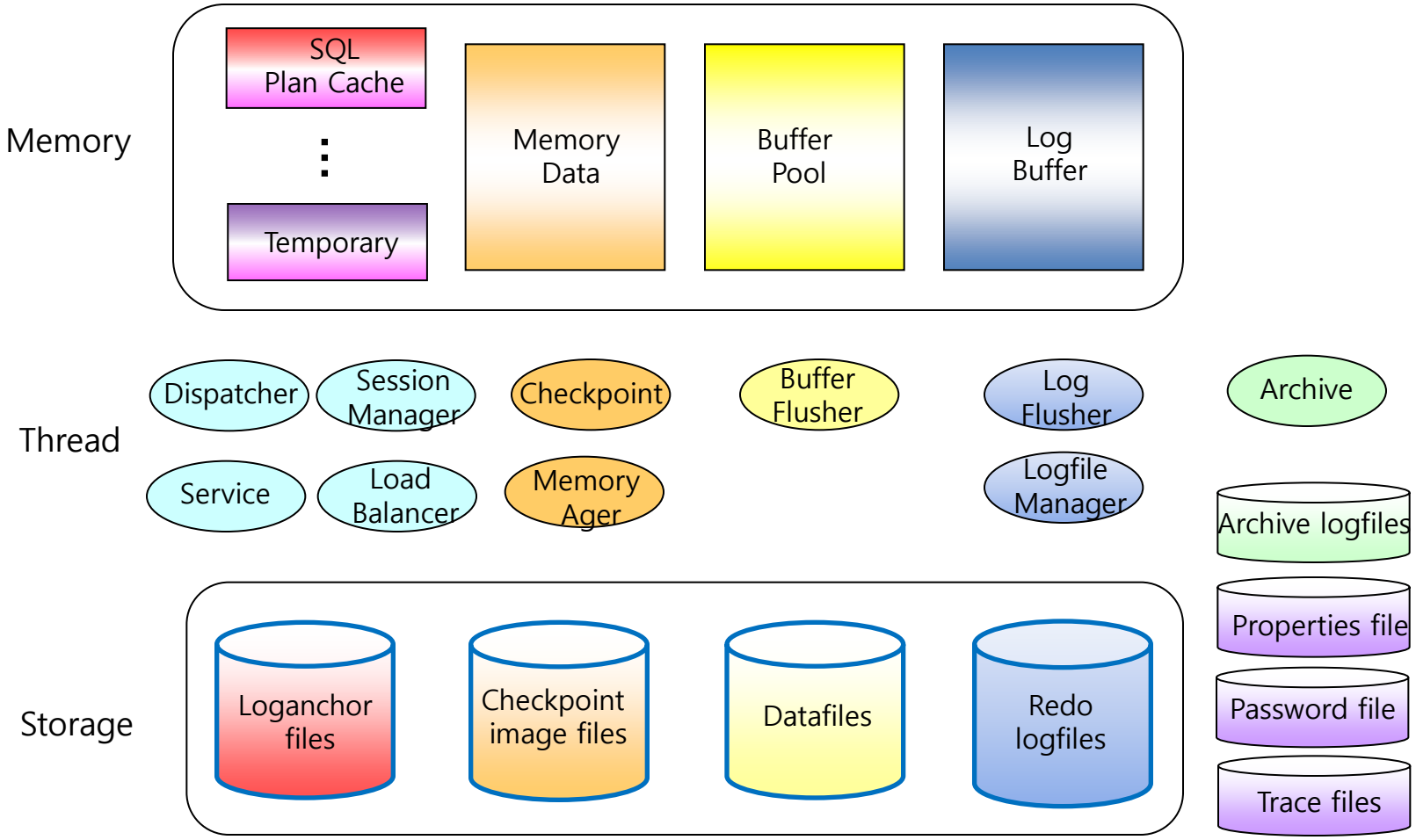
Features

- **DeadLock 감지**
  비정상적인 트랜잭션 감지를 위해 프로세스를 따로 두지 않고 DeadLock 발생 순간에 에러를 발생시켜 신속히 조치
- **Stored Procedure/Function 제공**
  업무 절차를 하나의 서버 모듈로 만든 후, DBMS에 저장해 두고, 모듈 이름만 호출하여 업무 프로세스를 DBMS에 수행할 수 있음
- **데이터베이스 이중화**
  TCP 기반의 네트워크를 통해 변경 트랜잭션 로그를 전송하고 원격 서버에서 이를 반영하여 데이터를 복제를 수행
- **데이터베이스 공간**
  ALTIBASE HDB는 하나 이상의 테이블스페이스로 구성되고, 테이블스페이스는 메모리와 디스크로 나눔

ALTIBASE

ARCHITECTURE

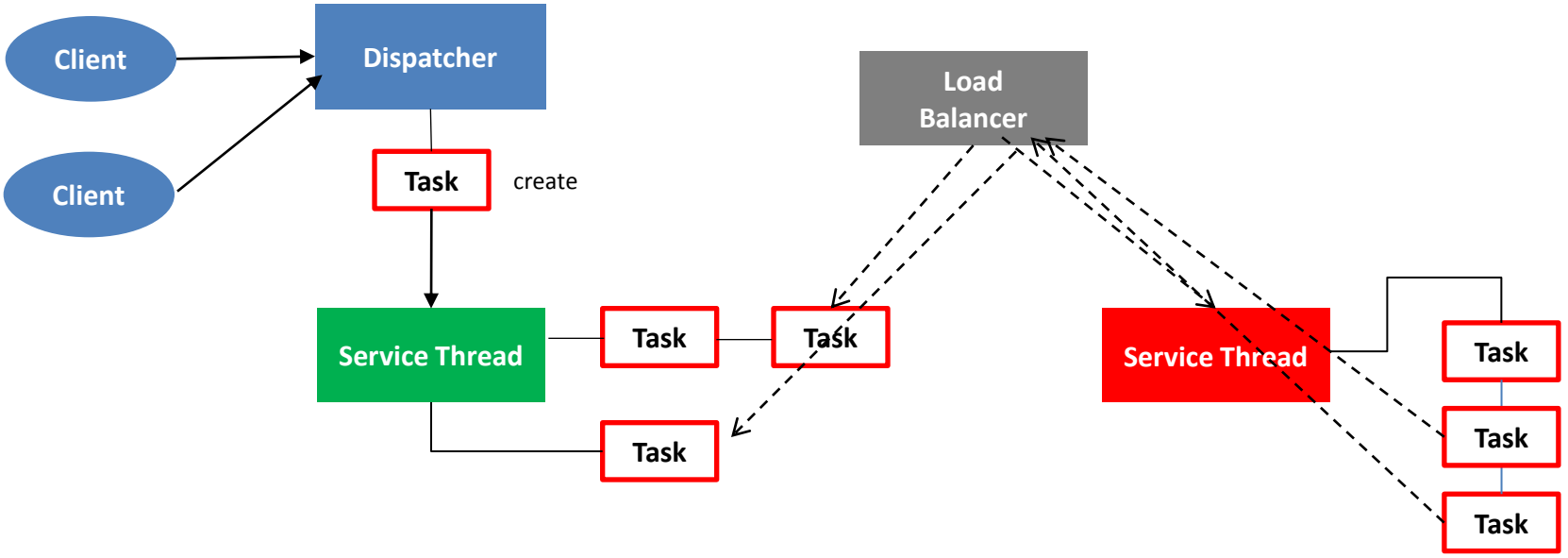
ARCHITECTURE(전체)



서비스 쓰레드

❖서비스 쓰레드 모드

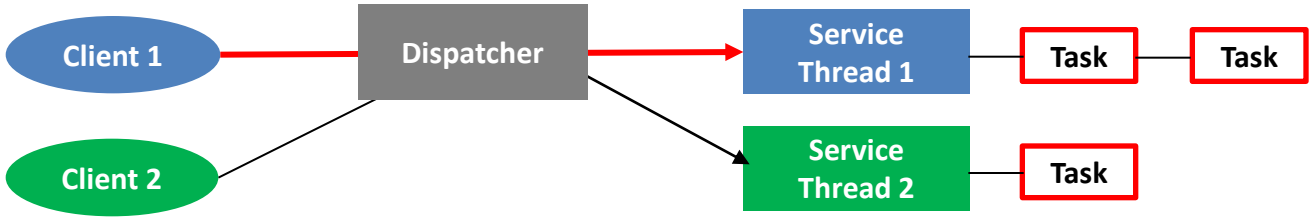
- multiplexing thread mode
  - 여러 개의 세션을 하나의 쓰레드가 관리하는 방식
  - 동시 접속 사용자가 많은 경우에 유리하며, 서버의 자원 사용량이 적음
  - 태스크 분배 시 오버헤드가 존재하여 사용자가 적은 경우 자원이 낭비될 수 있음



서비스 쓰레드

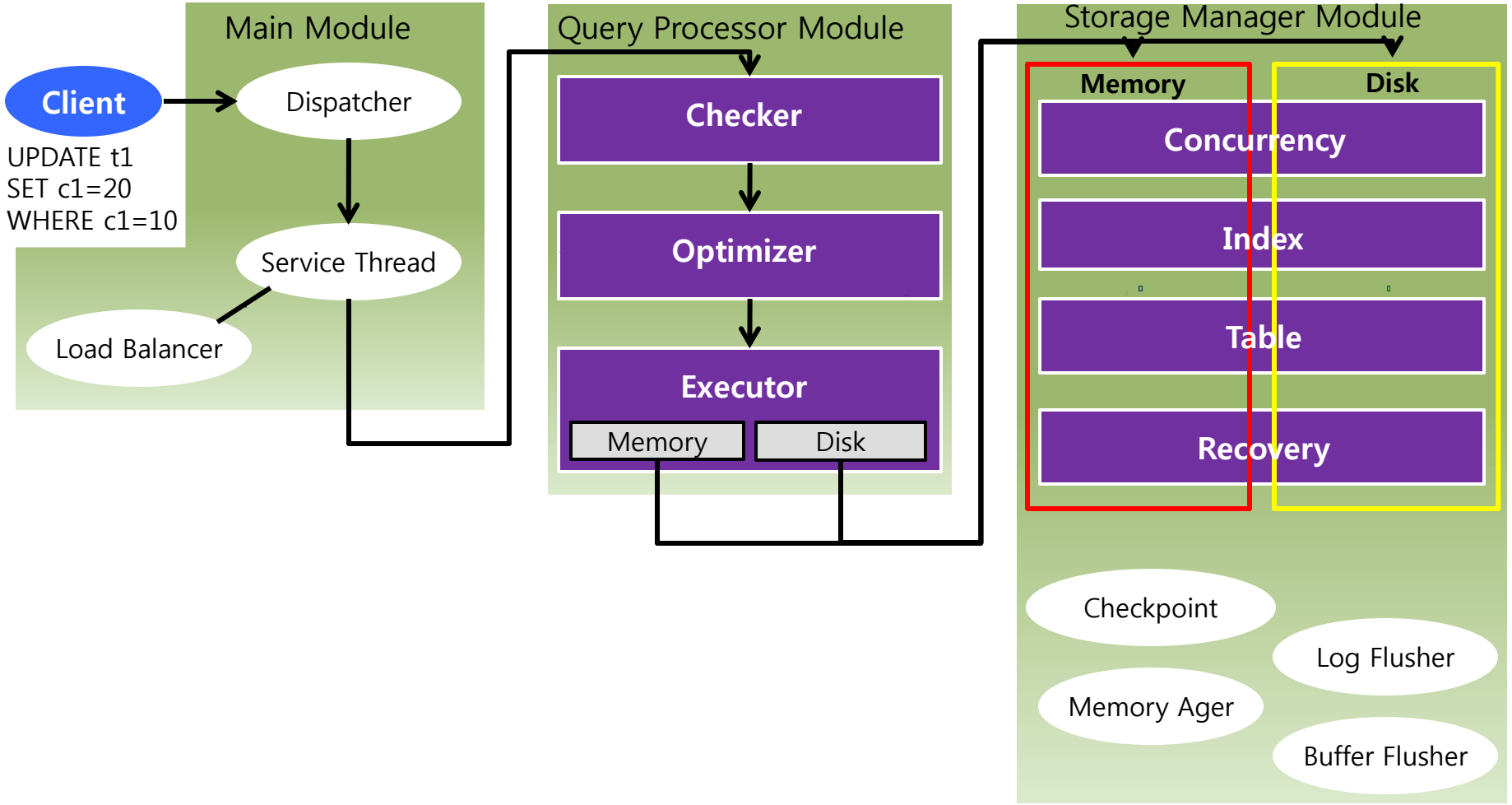
❖서비스 쓰레드 모드

- 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 용이함
  - 세션 연결 시마다 쓰레드를 늘려야 하기 때문에 사용자 증가에 따른 CPU 사용량도 증가함



- Dedicated thread mode 설정 방법
  - DEDICATED_THREAD_MODE 값을 1로 설정해야 함(기본값은 0)
  - CPU affinity 를 적용 시, multiplexing mode에 비해 성능이 월등히 향상됨
    - CPU affinity : CPU 연계 기능을 활성화하면 하나의 서비스 쓰레드는 동일한 CPU 코어에만 할당되어 다른 CPU 코어로 쓰레드 정보를 이동하는 비용이 줄어듬

내부 프로세스


Main Module
Client
Dispatcher
UPDATE t1
SET c1=20
WHERE c1=10
Service Thread
Load Balancer
Query Processor Module
Checker
Optimizer
Executor
Memory
Disk
Storage Manager Module
Memory
Disk
Concurrency
Index
Table
Recovery
Checkpoint
Log Flusher
Memory Ager
Buffer Flusher

ALTIBASE

TABLESPACE MANAGEMENT

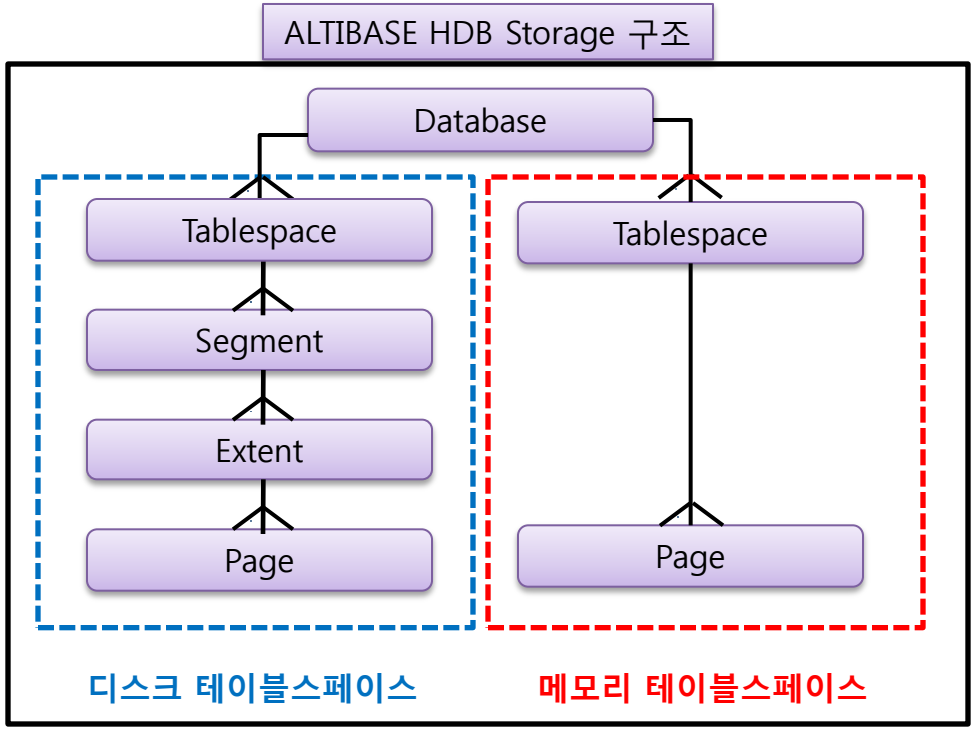
테이블스페이스 개념

- **테이블스페이스 (Tablespace / TBS)**
  - 데이터베이스를 구성하는 최상위 논리적인 구조
  - 테이블, 인덱스 등의 데이터베이스 객체들이 저장되는 논리적인 저장소
  - 데이터베이스 운영을 위해 기본적으로 하나 이상의 테이블스페이스가 필요

STORAGE

❖스토리지 구조

➢ 하나의 데이터베이스는 한 개 이상의 테이블스페이스로 구성되며, 하나의 테이블스페이스는 다수의 세그먼트 또는, 다수의 페이지로 구성됨



➢ 메모리 테이블스페이스
- 32K 크기의 페이지들로 구성

➢ 디스크 테이블스페이스
- 다수의 세그먼트로 구성
- 세그먼트는 다수의 익스텐트로 구성
- 익스텐트는 8K 크기의 페이지 64개로 구성됨(512K)

테이블스페이스 종류

❖ALTIBASE HDB에서 제공하는 테이블스페이스

- 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

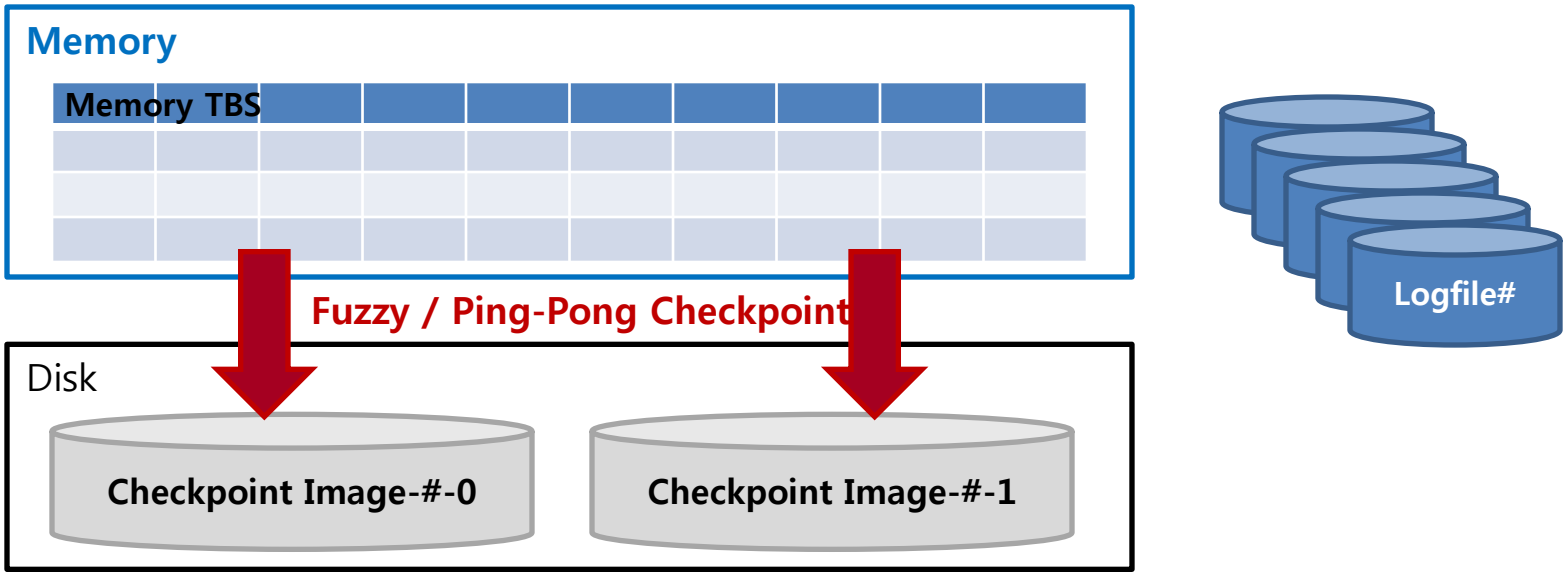
  - 사용자 테이블스페이스(User Tablespace)
    - 사용자의 필요에 따라 선택적으로 생성
    - 임시 TBS, 데이터 TBS(메모리 TBS, 휘발성 TBS, 디스크 TBS)

메모리 테이블스페이스

❖메모리 테이블스페이스

- 모든 데이터가 메모리 공간에 저장되는 테이블스페이스를 의미
- ALTIBASE HDB 4까지는 하나만 존재
- ALTIBASE HDB 5부터는 사용자/업무별로 확장하여 추가 할 수 있음

❖구조 (메모리 TBS + 체크포인트 이미지 파일)



메모리 테이블스페이스

❖메모리 테이블스페이스의 공간 할당

32K 페이지 단위로 테이블에 공간 할당

❖Page의 상태

객체	Free	Used
Tablespace	•특정 테이블에 할당되지 않은 공간 •1 Page 씩 특정 테이블에 할당 가능	•테이블에 할당한 공간 •테이블에서 반납하기 전까지 다른 테이블에서 사용할 수 없음
Table	•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

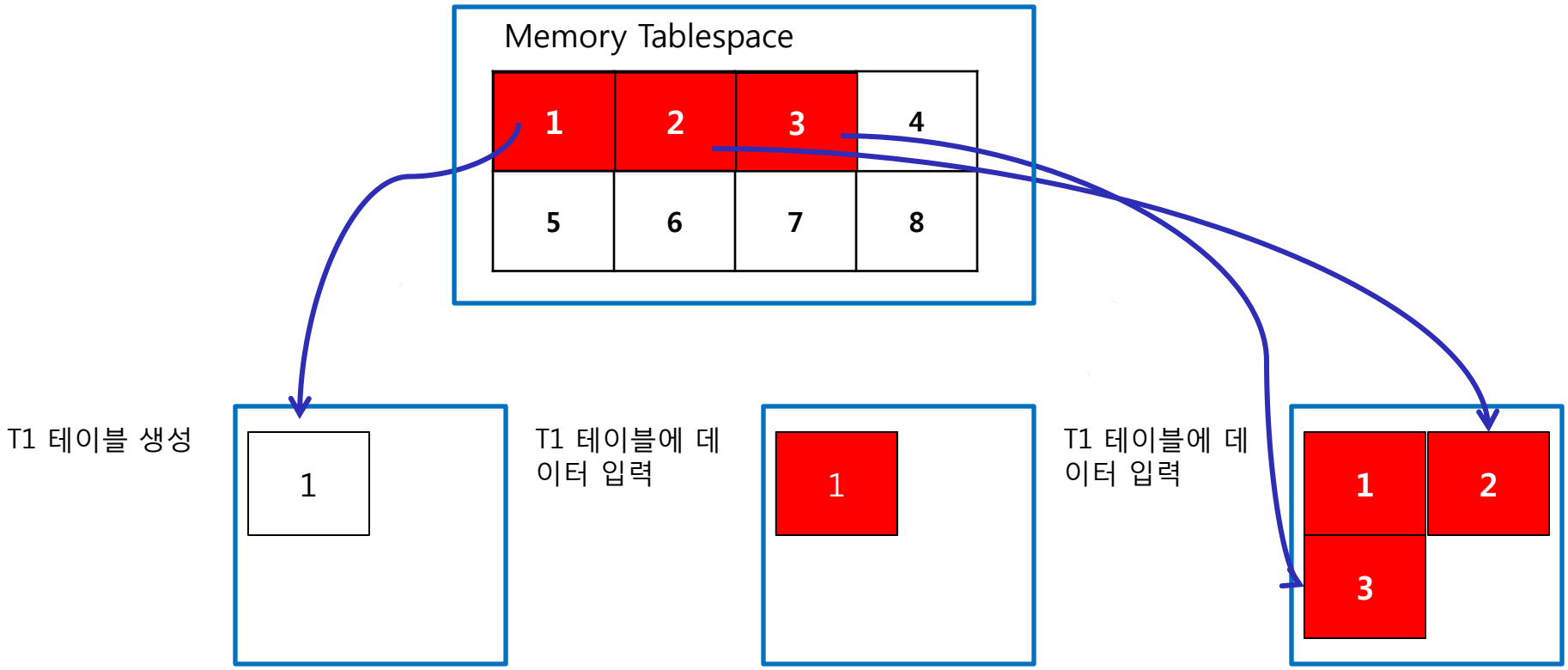
메모리 테이블스페이스

❖Page의 상태변화

- Table에 Delete 수행
  - Table 안에서 Page 상태가 Used → Free 로 전환(해당 Table 내에서 재사용 가능)
  - Tablespace로 Page를 반환하지 않음
  - Delete 수행 후 Compaction을 수행하면 해당 Page를 Tablespace에 반하고(다른 테이블에서 사용 가능), Tablespace의 Page 상태는 Used → Free로 변경됨
- Table에 Truncate 수행
  - Table에 할당 되었던 Page를 Tablespace에 반환함
  - Table에 할당되었던 Tablespace의 Page 상태는 Used → Free로 전환(다른 Table에서 할당 받아 사용 가능)
- Table에 Move 수행
  - Move 구문을 통해서 데이터를 다른 Table로 이동시켜도 Delete 한 것과 동일하게 해당 Table안에서만 재사용 가능
  - Move 수행 후, Compaction을 수행하면 해당 Page를 Tablespace에 반환함

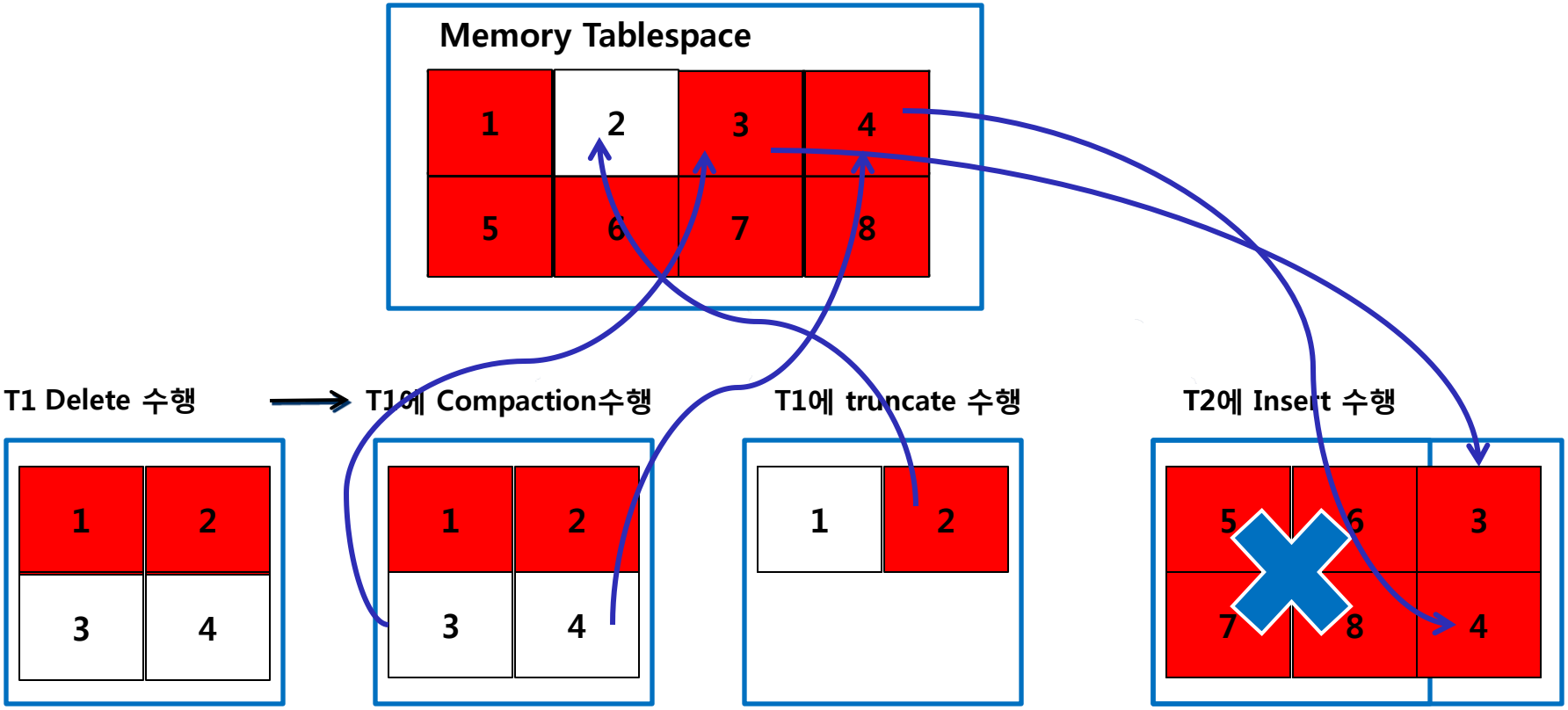
메모리 테이블스페이스의 공간 할당

❖공간 할당 구조



메모리 테이블스페이스의 공간 할당

❖공간 반납 구조



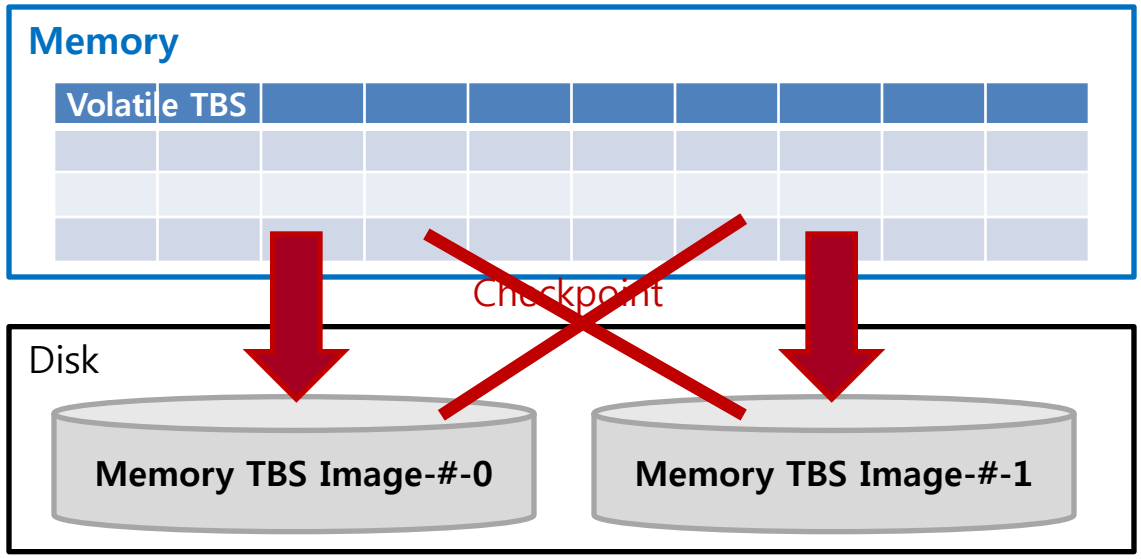
휘발성 테이블스페이스

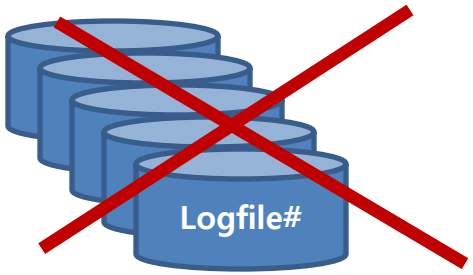
❖휘발성 테이블스페이스

- 데이터가 메모리에만 상주
- 디스크에 체크포인트 이미지 파일을 가지지 않음

❖특징

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





임시 테이블

❖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 트랜잭션이 종료되면 임시 테이블은 자동으로 truncate
- 임시 테이블내의 데이터는 백업이나 시스템 장애 시 복구가 불가능

❖제약사항

- 파티셔닝 불가
- 외래키 지정 불가
- 휘발성 테이블 스페이스에서만 저장 가능
- 분산 트랜잭션 지원 불가

임시 테이블 생성

❖구문

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

❖예제

➢ 트랜잭션에 한정되는 임시 테이블을 생성하라

```
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

❖ 예제

➢ 세션에 한정되는 임시 테이블을 생성하라

```
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.
```

임시 테이블 생성

❖ 예제(ON COMMIT DELETE ROWS)

➢ 임시 테이블을 생성하고 데이터를 입력한 후, 트랜잭션이 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.
```

트랜잭션에 한정되는 임시테이블(t1)

```
iSQL> COMMIT;
Commit success.

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

Commit, Rollback 시 데이터 지워짐

임시 테이블 생성

❖ 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)

임시 테이블 생성

```
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

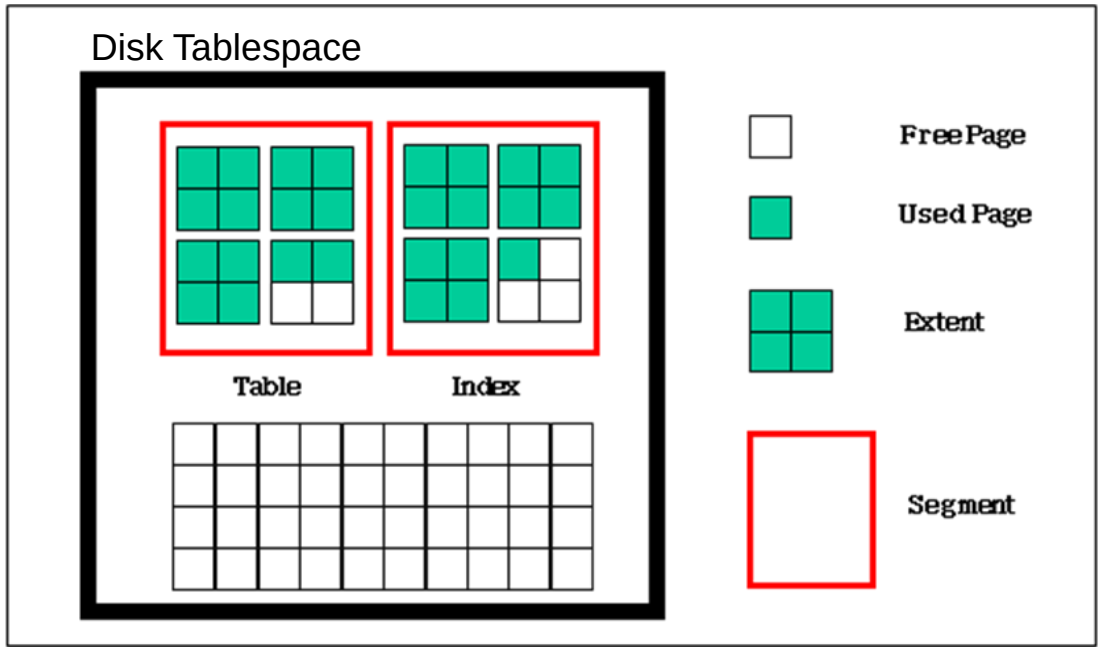
다른 세션에서 조회 결과

디스크 테이블스페이스

❖ **디스크 테이블스페이스**

모든 데이터가 디스크 공간에 저장되는 테이블스페이스를 의미

❖ **구조**



- 페이지 (Page)
  - 테이블과 인덱스의 레코드를 저장하는 최소단위 [디스크 페이지 : 기본 8KB]
- 익스텐트 (Extent)
  - 페이지를 할당하는 단위
  - 데이터 저장 시 free 페이지가 부족하면 테이블스페이스로부터 할당 받음
  - 기본 값 : 64개의 페이지 (512KB) 로 구성

디스크 테이블스페이스

❖ 디스크 테이블스페이스의 공간 할당

Table에 512K Extent 단위로 공간을 할당

❖ Page의 상태

객체	Free	Used
Tablespace	•특정 테이블에 할당되지 않은 공간 •Extent 단위로 특정 테이블에 할당 가능	•테이블에 할당한 공간 •테이블에서 반납하기 전까지 다른 테이블에서 사용할 수 없음
Table	•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 테이블스페이스의 공간 할당

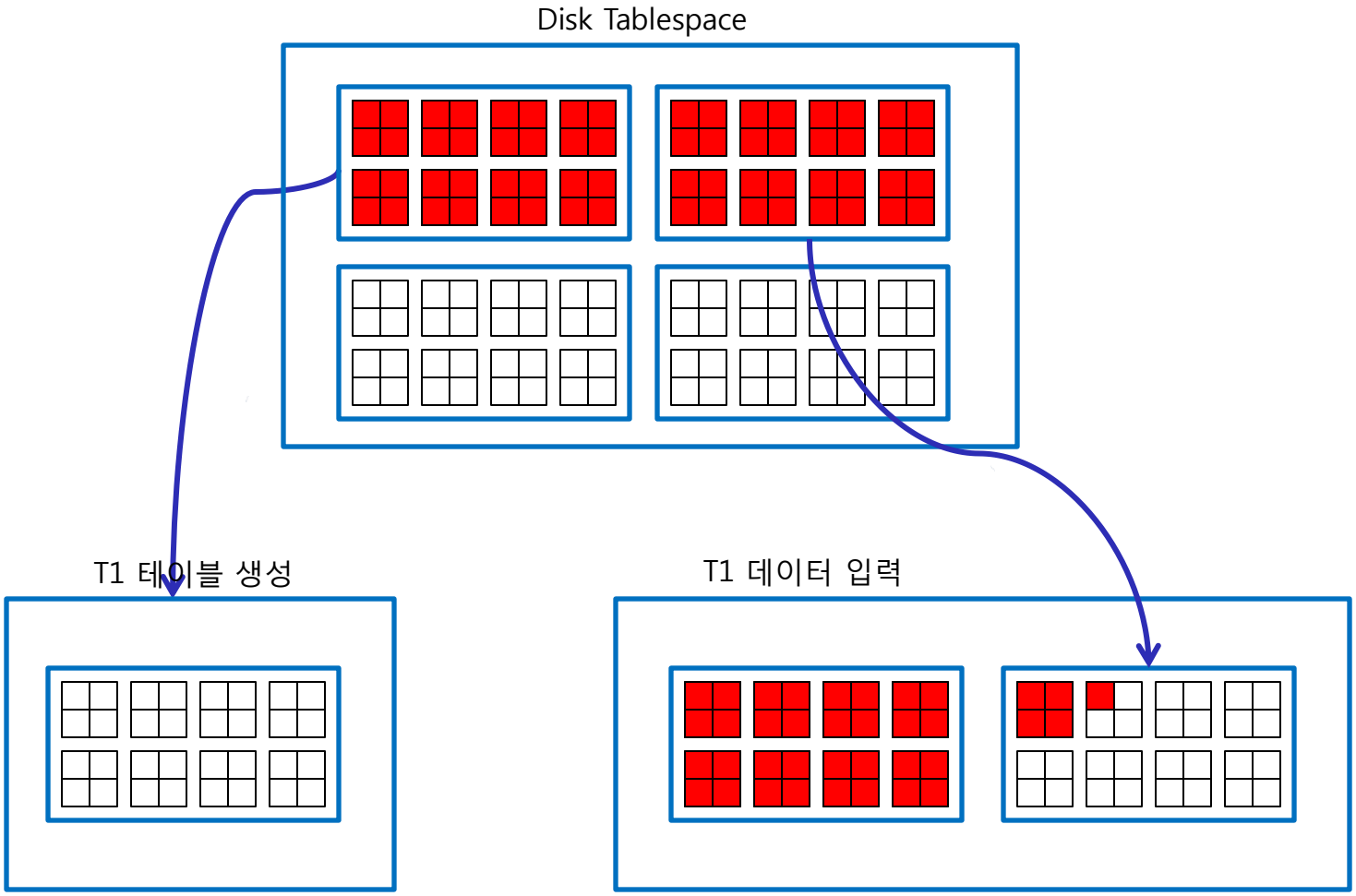
❖Page의 상태변화

- Table에 Delete 수행
  - Table 안에서 Page 상태가 Used → Free 로 전환(해당 Table 내에서 재사용 가능)
  - Tablespace로 Page를 반환하지 않음
- Table에 Truncate 수행
  - Table에 할당 되었던 Page를 Tablespace에 반환함
  - Table에 할당되었던 Tablespace의 Page 상태는 Used → Free로 전환(다른 Table에서 할당 받아 사용 가능)
- Table에 Move 수행
  - Move 구문을 통해서 데이터를 다른 Table로 이동시켜도 Delete 한 것과 동일하게 해당 Table안에서만 재사용 가능

❖ 디스크테이블에는 Compaction 기능이 없음

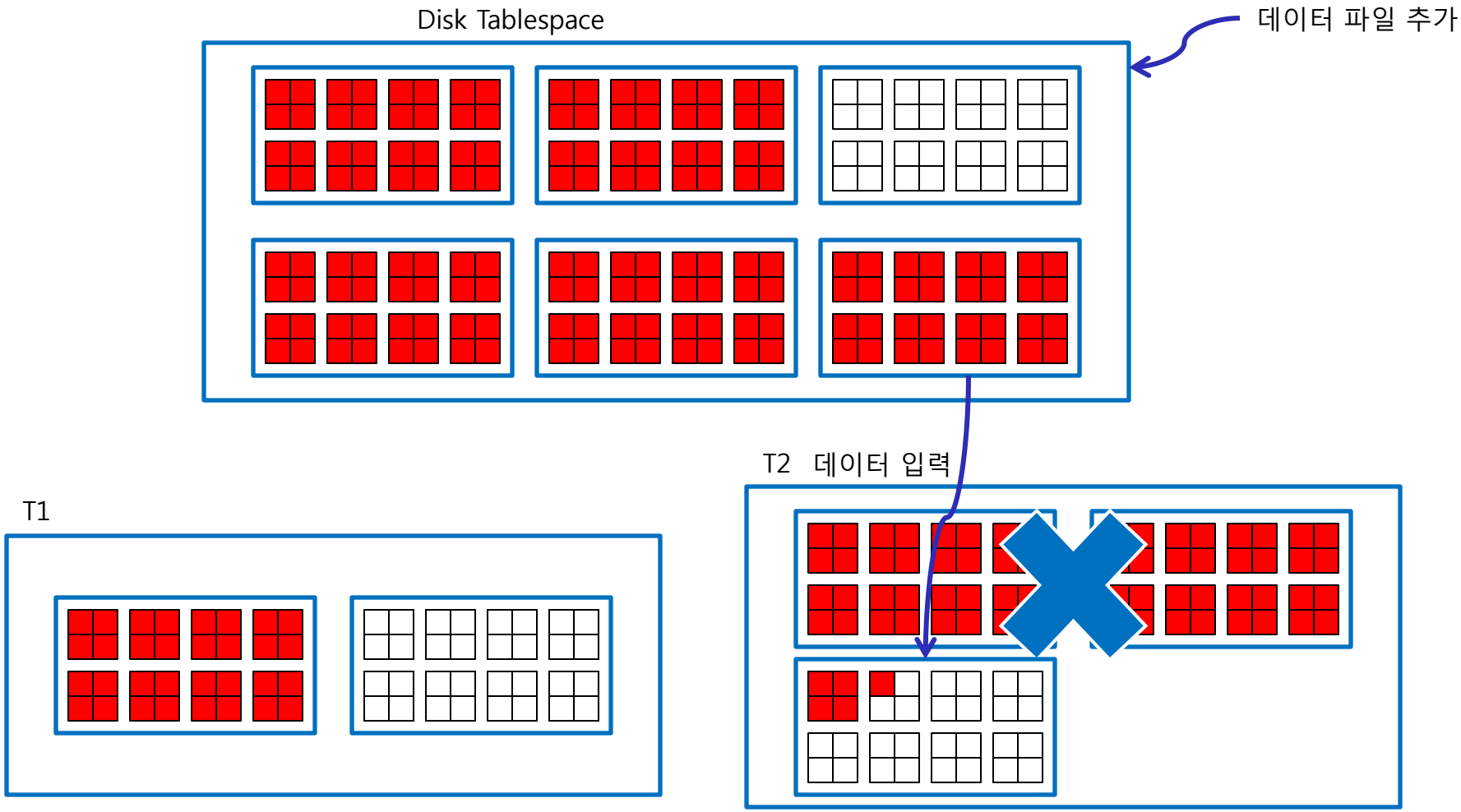
디스크 테이블스페이스의 공간 할당

❖공간 할당 구조



디스크 테이블스페이스의 공간 할당

❖공간 반납 구조



테이블스페이스 확인(3)

❖ 테이블스페이스 관련 Performance View

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

메모리 테이블스페이스 생성

❖메모리 테이블스페이스

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

❖구문(기본)

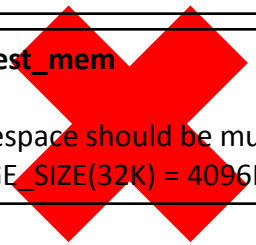
```
CREATE MEMORY [DATA] TABLESPACE tablespace_name
SIZE size (K | M | G) ;
```

❖예제

- 초기 사이즈가 512M인 메모리 테이블스페이스를 생성

```
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

메모리 테이블스페이스 생성

❖구문(자동확장 추가)

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

❖예제

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

```
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 테이블스페이스를 생성

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

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]] ;
```

❖예제

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

```
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.
```

휘발성 테이블스페이스 생성

❖휘발성 테이블스페이스

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

❖구문

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

❖예제

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

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

디스크 테이블스페이스 생성

❖ 디스크 테이블스페이스

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

❖ 구문(기본)

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

❖ 예제

- 기본 경로에 데이터 파일 test01.dbf 를 생성하는 test_disk 테이블스페이스를 생성

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

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)];
```

❖예제

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

```
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 테이블스페이스를 생성

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

임시 테이블스페이스 생성

❖ 임시 테이블스페이스

- 디스크 데이터에 대한 질의 수행 중 생성되는 임시 결과를 저장하기 위한 테이블스페이스
- 트랜잭션이 종료하는 시점에 해당 질의가 남긴 모든 데이터들은 사라짐

❖ 구문

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

❖ 예제

- tbs.temp로 구성된 test_temp 임시 테이블스페이스를 생성. 임시 파일의 크기는 10M이고, 5M 크기로 자동 확장.

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

디스크 테이블스페이스 변경

❖테이블스페이스 변경

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

❖구문

```
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']
};
```

❖예제

- test_disk 테이블스페이스에 64 MB의 데이터 파일 test01.dbf를 추가(공간이 더 필요할 때는 500K 씩 파일이 자동확장)

```
iSQL> ALTER TABLESPACE test_disk
      2 ADD DATAFILE 'test01.dbf' SIZE 64M
      3 AUTOEXTEND ON NEXT 500K;
Alter success.
```

메모리 테이블스페이스 변경

❖ 테이블스페이스 변경

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

❖ 구문

```
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) ]
};
```

❖ 예제

- Test_mem 테이블스페이스를 8M 단위로 최대 1GB 까지 자동 확장으로 변경

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

테이블스페이스 변경

- test_disk 디스크 테이블스페이스가 자동확장을 하지 않도록 변경

```
iSQL> ALTER TABLESPACE test_disk
      2 ALTER DATAFILE ' test01.dbf'
      3 AUTOEXTEND OFF;
Alter success.
```

- test_disk 디스크 테이블스페이스가 자동확장을 하도록 변경

```
iSQL> ALTER TABLESPACE test_disk
      2 ALTER DATAFILE ' test01.dbf‘
      3 AUTOEXTEND ON
Alter success.
```

- test_disk 테이블스페이스의 데이터파일 test01.dbf 를 삭제하시오.

```
iSQL> ALTER TABLESPACE test_disk
      2 DROP DATAFILE ‘test01.dbf‘;
Alter success.
```

테이블스페이스 삭제

❖ 테이블스페이스 삭제

- 데이터베이스에서 테이블스페이스를 제거함
- 시스템 테이블스페이스는 삭제할 수 없음

❖ 구문

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

❖ 예제

- 메모리 테이블스페이스 test_mem을 삭제

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

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

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

테이블스페이스 상태

❖테이블스페이스 상태

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

❖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 DB구동이 불가능할 시 특정 테이블스페이스를 폐기할 수 있도록 설정

ALTIBASE

TRANSACTION MANAGEMENT

TRANSACTION

❖ 트랜잭션의 정의

- 트랜잭션이란 하나 이상의 SQL문장들로 이루어진 논리적인 작업단위

❖ 트랜잭션 관리

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

❖ 정상적인 트랜잭션의 무결성 조건

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

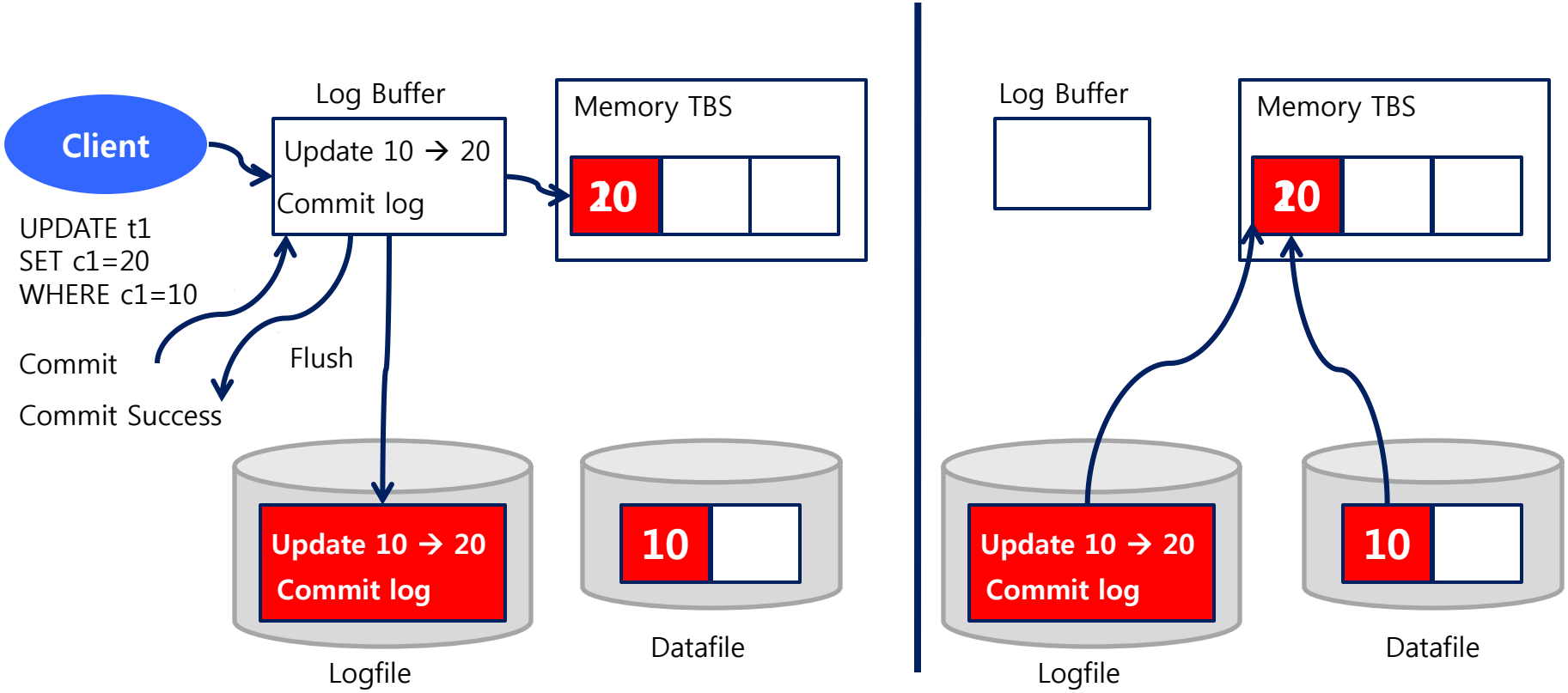
DURABILITY

❖ 트랜잭션의 영속성

- DBMS가 사용자에게 트랜잭션 커밋(commit) 응답을 했을 경우, 데이터베이스 객체에 대한 해당 변경 사항이 디스크에 반영(flush) 되기 전에 시스템 장애가 발생하였더라도 해당 트랜잭션의 커밋은 보장 되어야 한다는 속성
- 데이터베이스 관리 시스템은 트랜잭션의 durability를 제공하기 위해 로그(log) 기록을 관리한다.
- 로그 기록으로 인한 디스크 I/O는 트랜잭션 처리의 병목(bottleneck)으로 작용하게 되어 트랜잭션 처리 성능 저하의 원인이 될 수 있다.
- 완벽한 트랜잭션 durability과 트랜잭션 처리 성능 관계는 안정성과 성능이라는 각기 다른 목표를 가지는 상충(tradeoff) 관계에 있다.

DURABILITY

❖ 트랜잭션의 영속성과 복구



비정상 종료 발생 → SERVER Restart → Restart Recovery

DURABILITY LEVEL

❖ Durability Level 범위

- Durability Level은 0~5 Level로 구분
- ALTIBASE HDB는 0과 1 Level을 제외한 2에서 5 Level까지만 지원
- 기본 Durability Level은 3 Level이다.

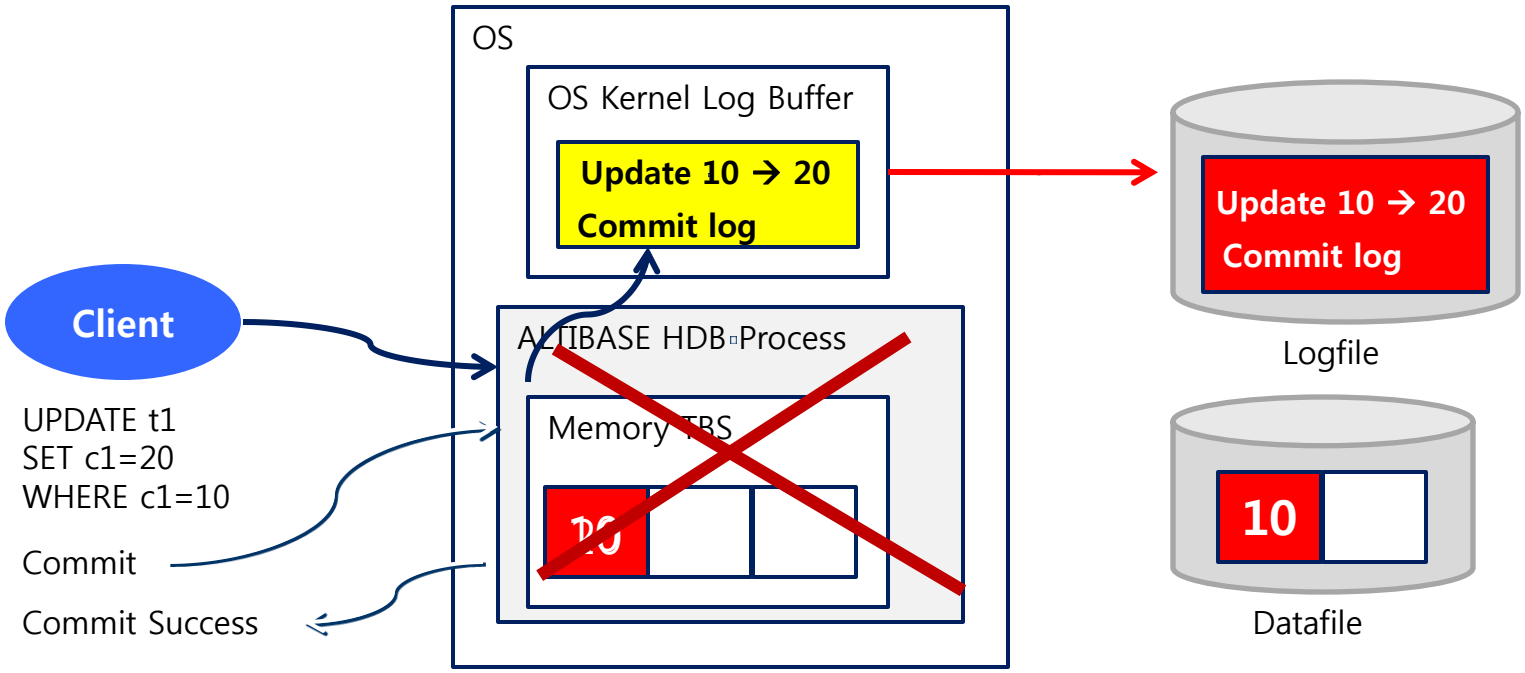
❖ Durability Level 설정방법

- altibase.properties 파일에서 관련 프로퍼티의 변경하여 설정
  - COMMIT_WRITE_WAIT_MODE, LOG_BUFFER_TYPE 프로퍼티로 설정
- DB 구동 시 설정된 Durability Level로 구동
- DB 운영 중에 실시간으로 변경 불가능

DURABILITY LEVEL

Durability Level 3

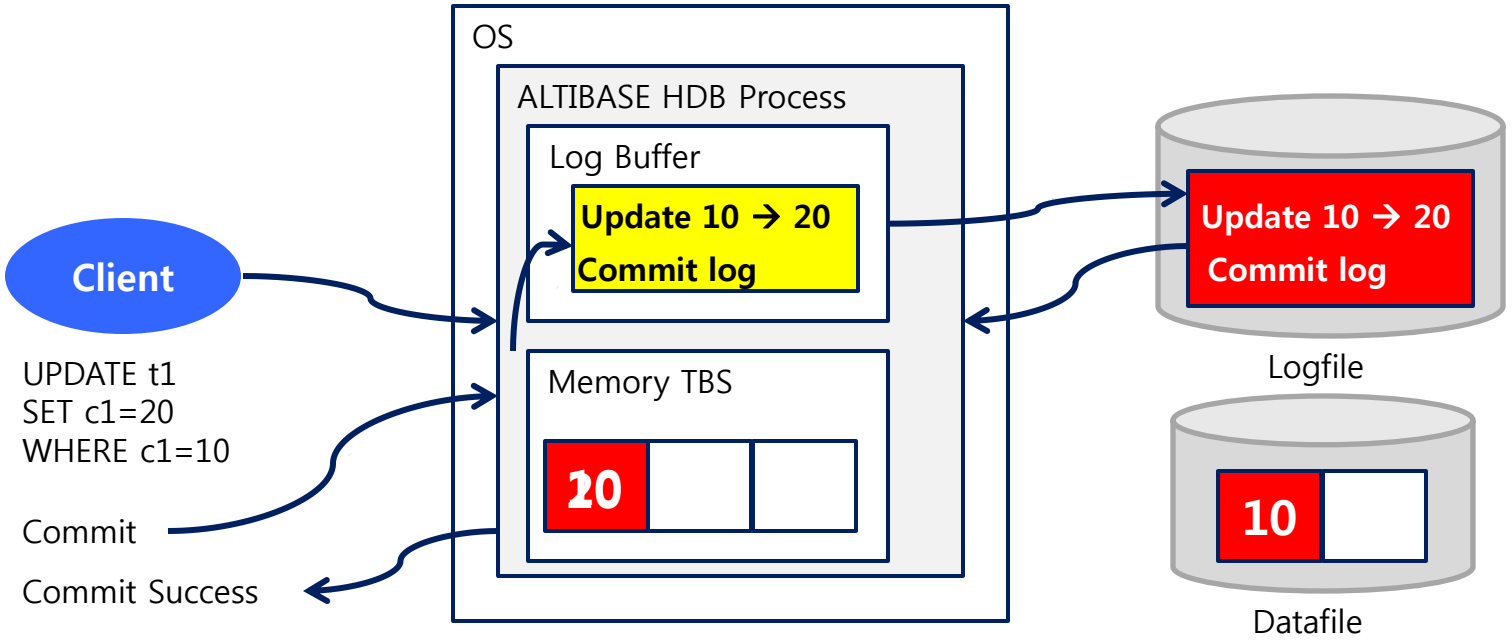
- OS Kernel 영역의 로그버퍼를 사용하기 때문에 ALTIBASE HDB 프로세스가 비정상 종료를 하더라도 트랜잭션이 commit한 로그는 운영체제의 의해 로그 파일에 반영
- ALTIBASE HDB의 기본 durability level(성능지향 설정 방법)
- OS의 crash 상황만 아니라면 트랜잭션 durability를 완벽하게 지원



DURABILITY LEVEL

Durability Level 5

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



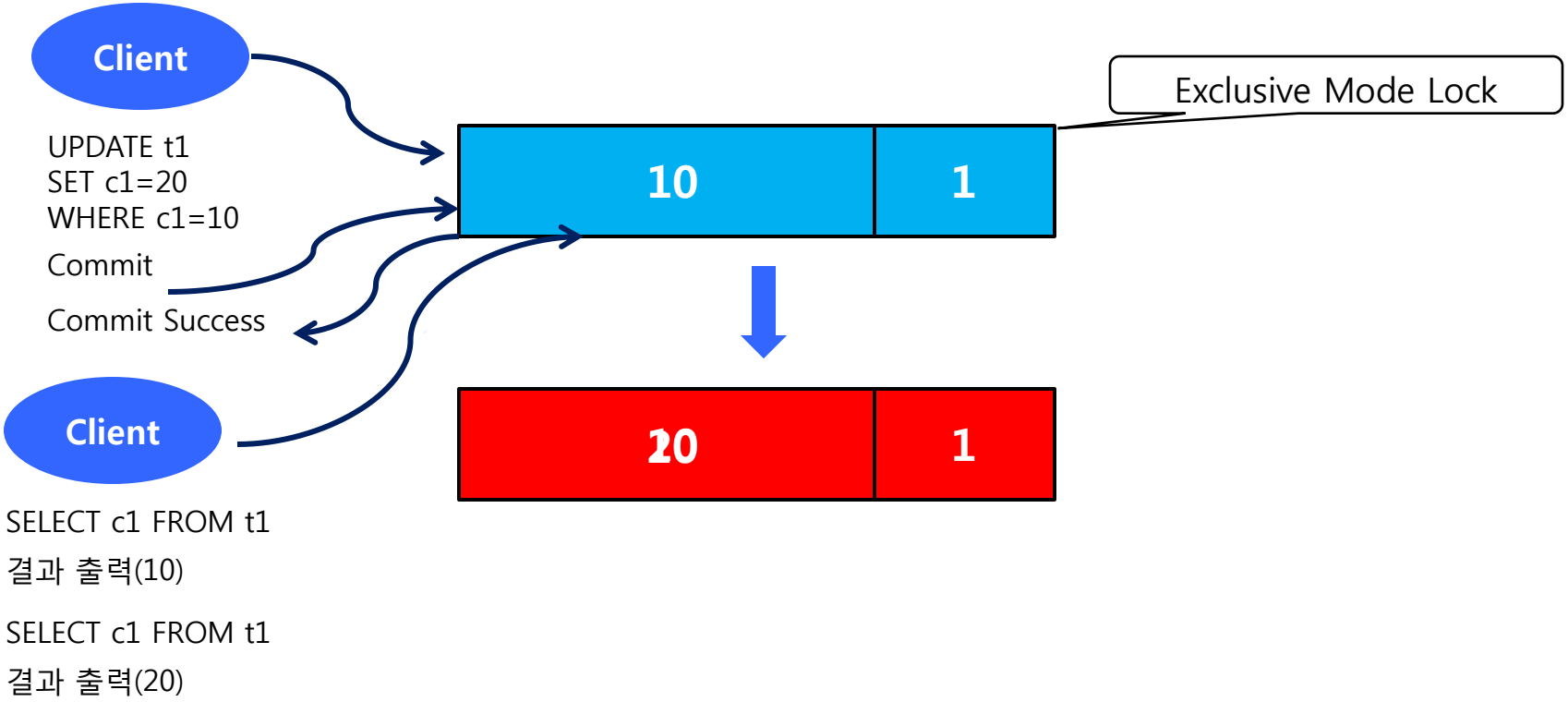
CONCURRENCY CONTROL

- **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 트랜잭션에게 영향을 미치게 됨
- **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 트랜잭션이 동일 레코드에 대해 연산을 수행하고 있더라도 그 레코드를 검색하는 다른 트랜잭션에게는 영향을 미치지 않도록 하는 동시성 제어 기법

MVCC

MVCC

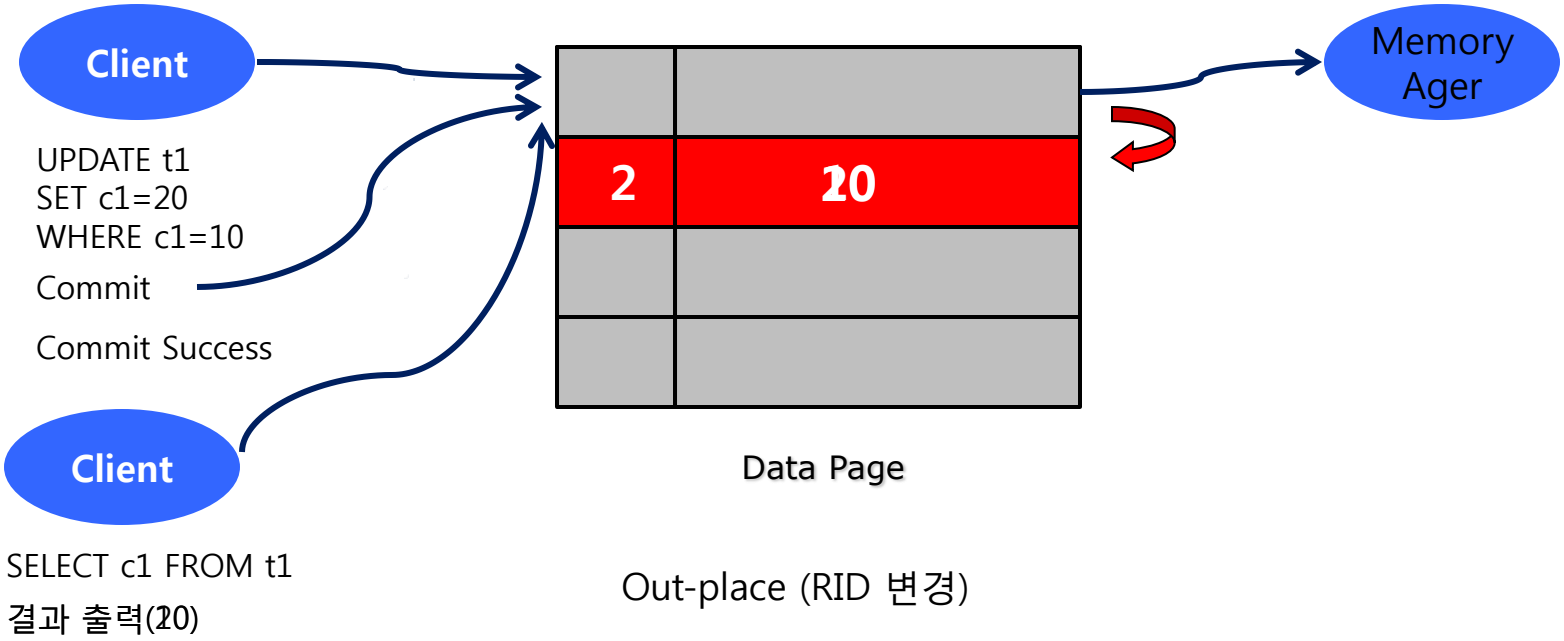
- 변경 연산 시에 Lock을 획득하고 새로운 버전을 생성하여 변경하기 때문에 Read/Modify 연산간에 충돌이 없고 복잡한 트랜잭션 환경에서 높은 성능을 보장



OUT-PLACE & IN-PLACE

❖메모리 테이블 : Out-place

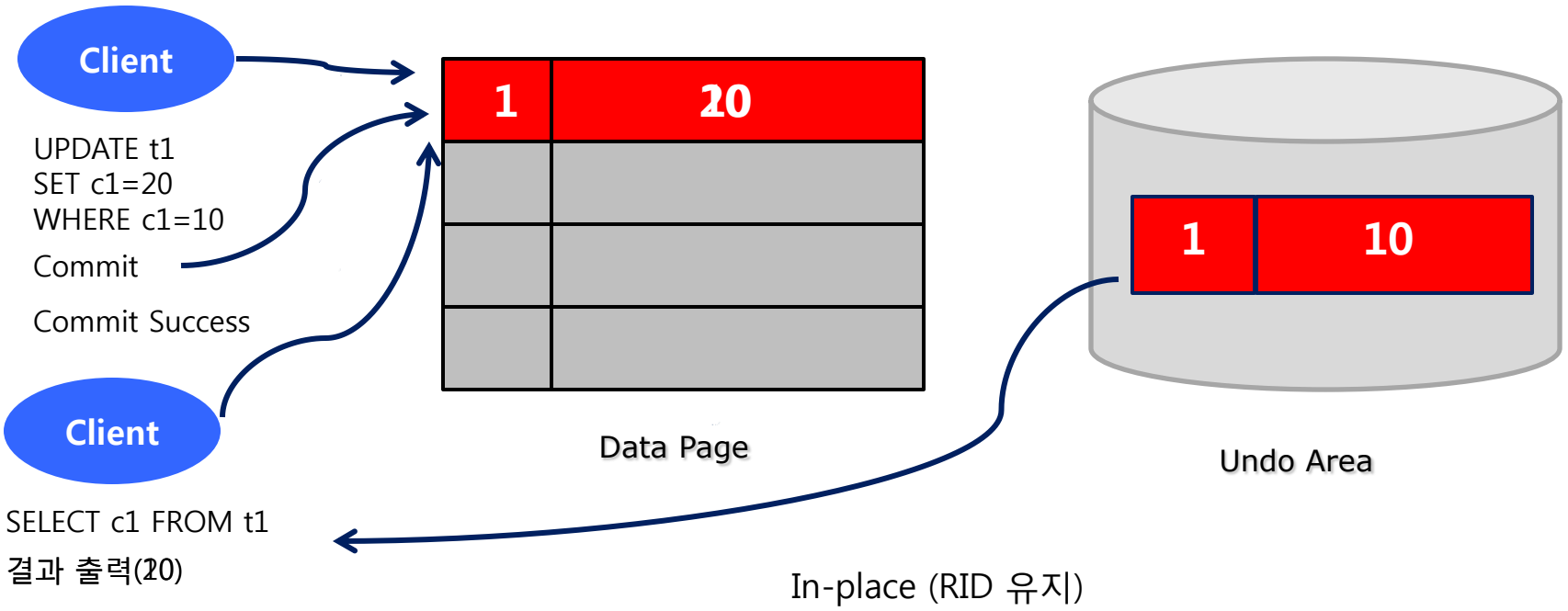
- 새로운 레코드 버전을 데이터 페이지의 별도 RID로 저장하는 Out-place 구조로 빠른 성능을 보장하나 데이터 페이지의 사용 효율성이 저하될 가능성이 있으므로 특정 레코드의 버전이 프로퍼티에 지정된 사이즈보다 커질 경우에는 In-place Update로 전환되도록 설계되어 있음



OUT-PLACE & IN-PLACE

❖ 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

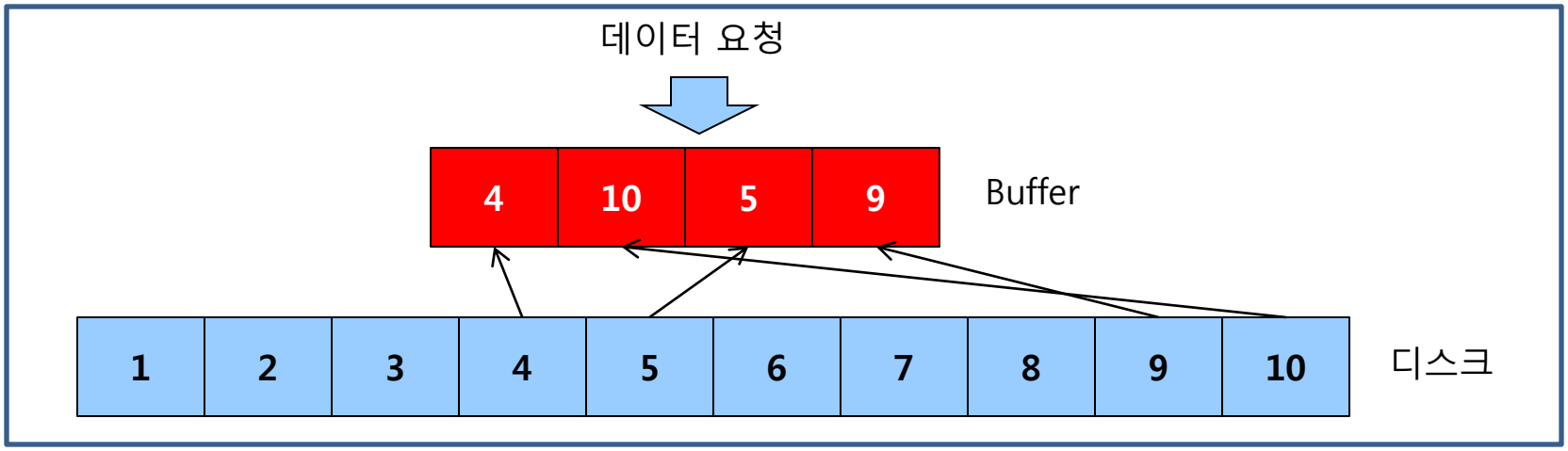


ALTIBASE

BUFFER MANAGEMENT

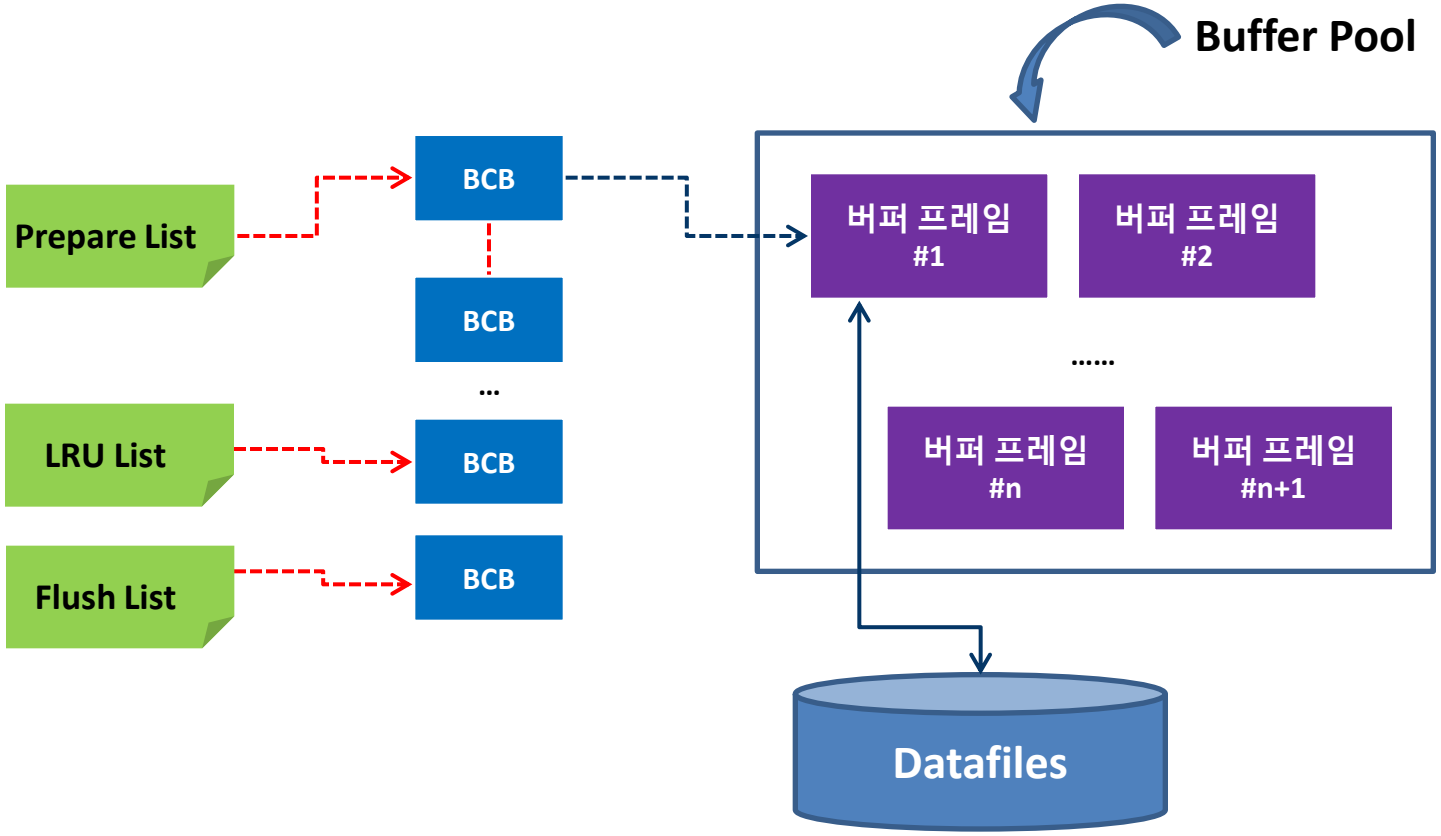
개요

- **Buffer**
  - 디스크 I/O를 줄이고 성능 향상을 위해서 디스크에 있는 페이지를 임시로 메모리에 적재하여 사용하는 메모리의 영역을 의미
- **Buffer Replace**
  - 디스크의 모든 페이지를 버퍼에 적재할 수 없으므로 새로운 디스크의 페이지를 적재할 때, 버퍼의 일부 페이지를 새로운 페이지로 교체하는 것
- **Victim**
  - Buffer Replace를 위해 선정된 버퍼 프레임과 해당 BCB를 아울러 지칭함



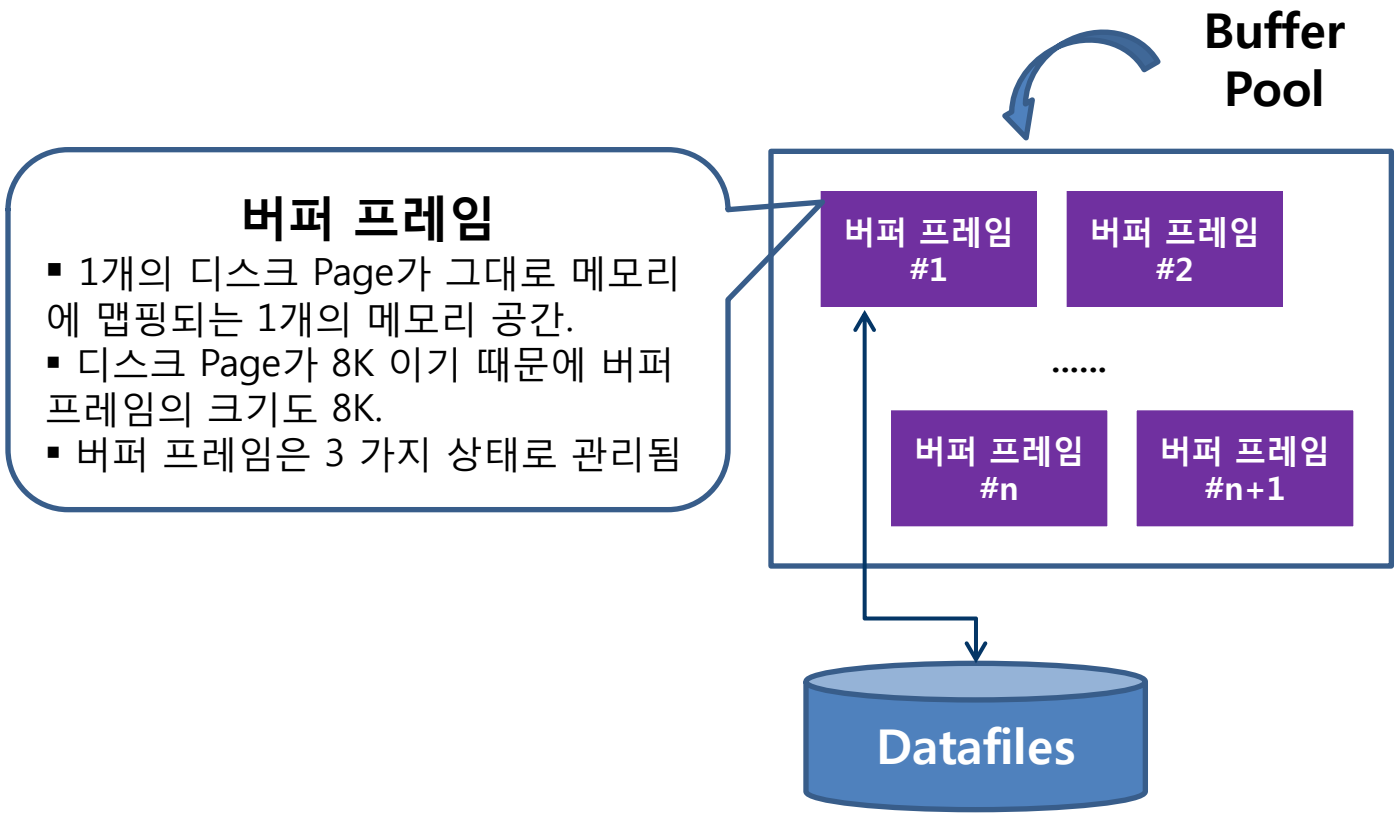
BUFFER 구성요소

Buffer 구성요소



BUFFER 구성요소

❖Buffer 구성요소



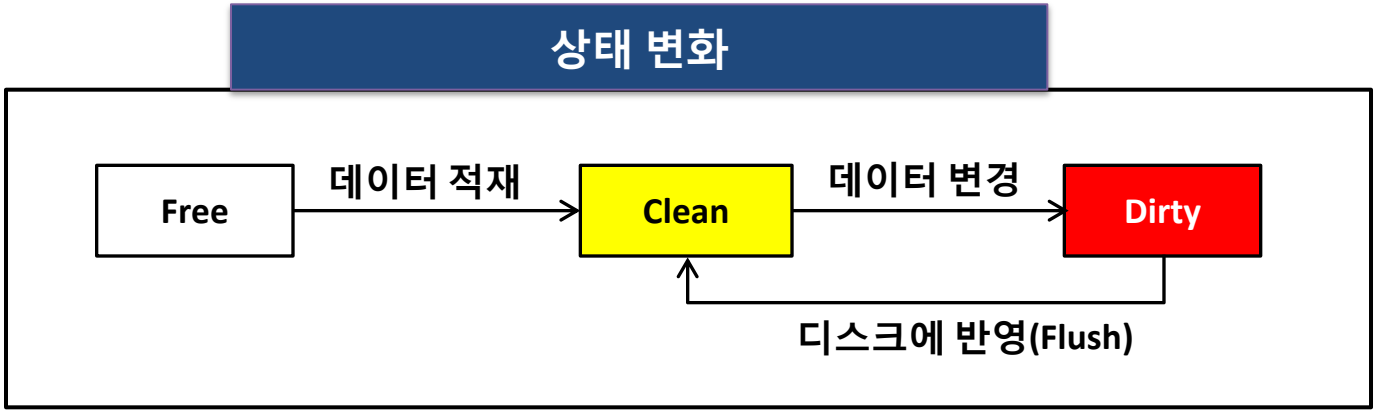
버퍼 프레임

- 1개의 디스크 Page가 그대로 메모리에 맵핑되는 1개의 메모리 공간.
- 디스크 Page가 8K 이기 때문에 버퍼 프레임의 크기도 8K.
- 버퍼 프레임은 3 가지 상태로 관리됨

BUFFER 구성요소

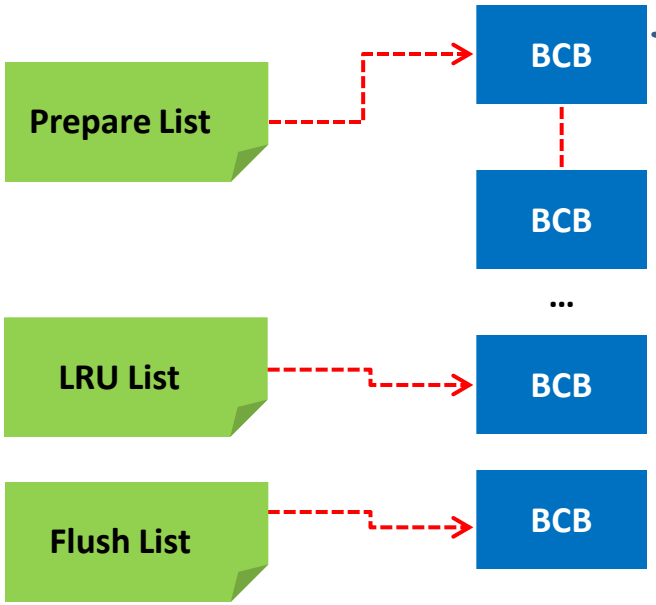
❖Buffer 구성요소

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 구성요소

❖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	디스크DB의 Page 식별번호
State	버퍼 프레임의 상태정보
ListItem	BCB가 어떤 리스트에 위치해 있는지 표시
ListNO	병렬화된 리스트 중 BCB가 위치한 리스트 번호

BUFFER 구성요소

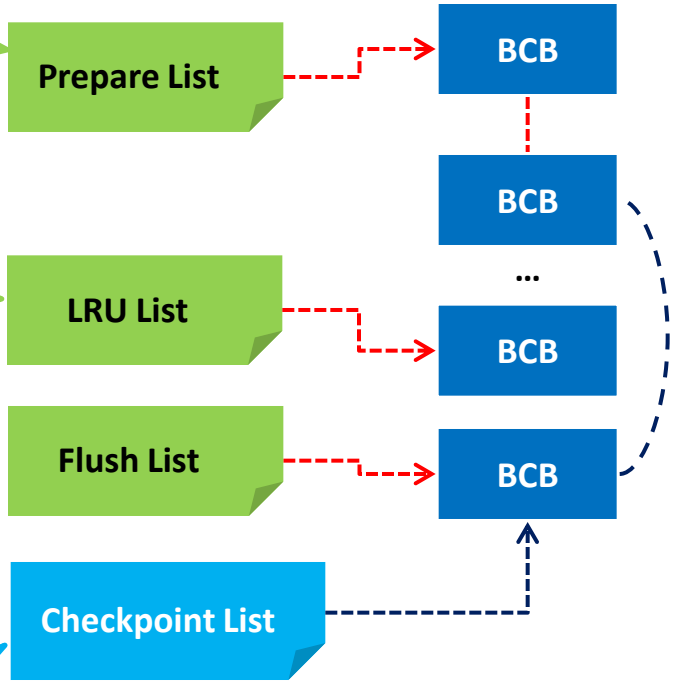
❖Buffer 구성요소

Prepare List
- DB 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

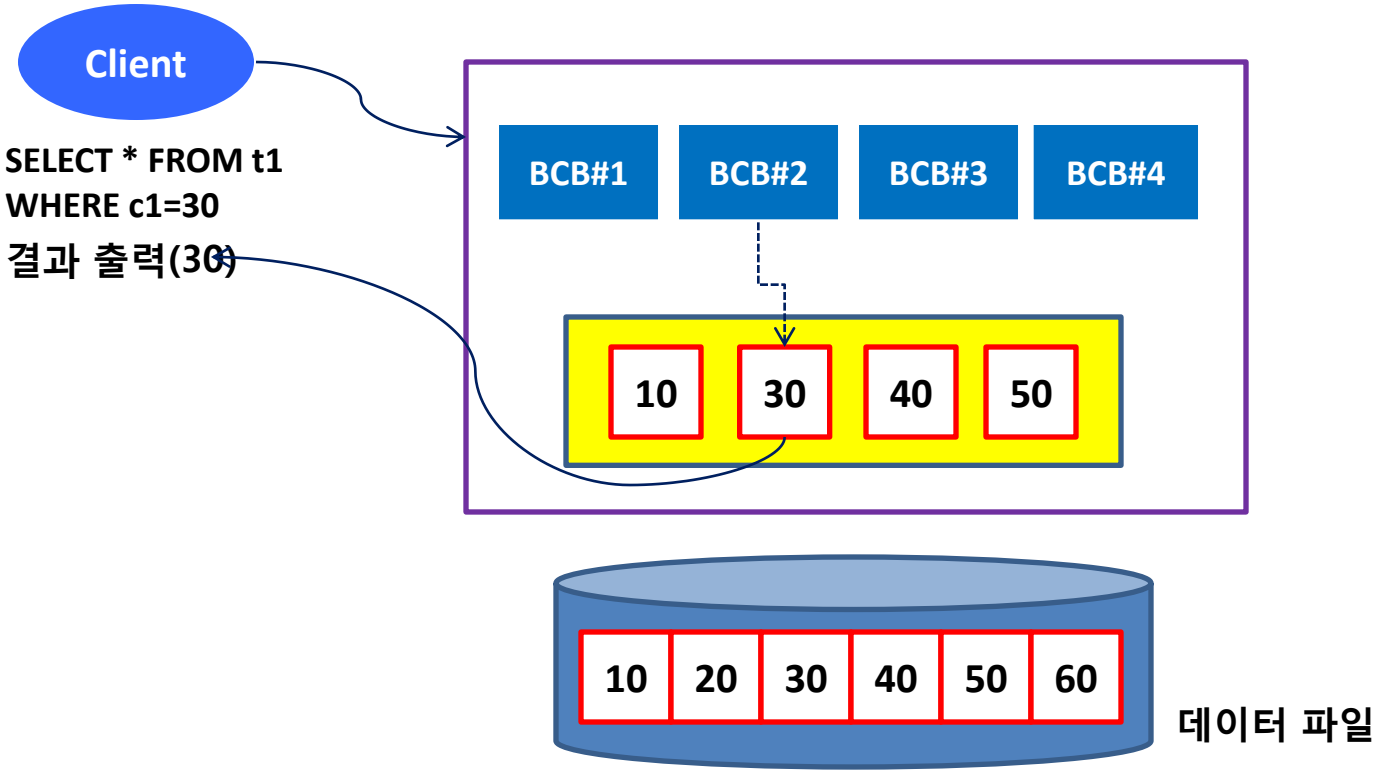
Flush List
- Dirty 상태의 BCB들을 관리하며, Buffer Flusher 에 의해서 디스크 Page로 flush됨
- Flush 가 되면, BCB의 상태는 Clean으로 변경되고 Replace를 위해 Prepare List 로 이동됨

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



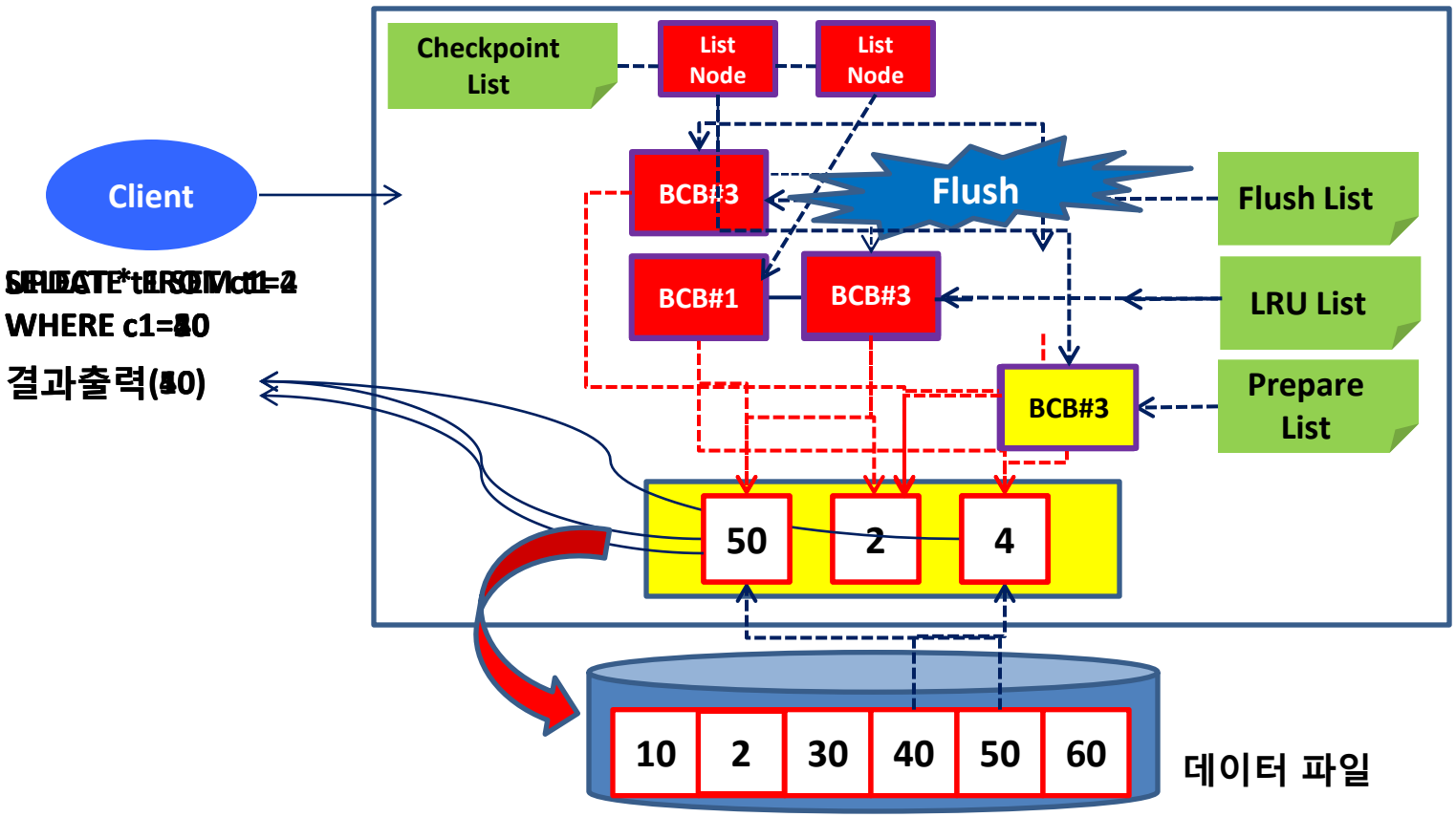
BUFFER 구성요소

Buffer 구성요소



BUFFER 구성요소

Buffer 구성요소



ALTIBASE

ALTIBASE HDB ADMINISTRATION I

REPLICATION

OVERVIEW
ALTIBASE REPLICATION
REPLICATION SYSTEM DESIGN
CONFLICT RESOLUTION
REPLICATION OPTIMIZATION
REPLICATION TROUBLESHOOTING
REPLICATION MONITORING
REPLICATION PROPERTY
REPLICATIO MANAGER

ALTIBASE

OVERVIEW

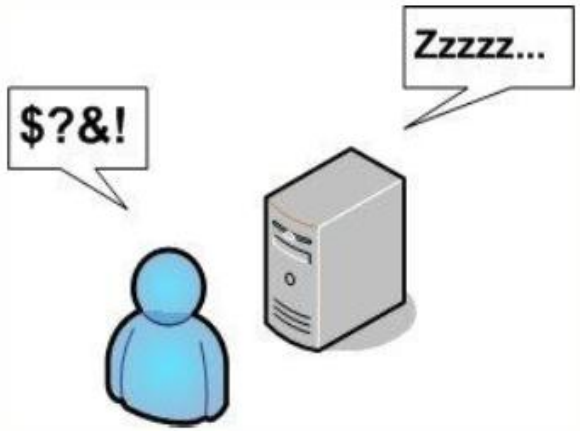
High Availabilility

❖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 기법이 존재



❖DBMS의 HA

- 노드 간 데이터베이스를 동기화함으로 실현
- 병렬 데이터베이스 아키텍처에 따라 기법이 다름

Shared Nothing vs. Shared Disk

구분	Shared Nothing Architecture	Shared Disk Architecture
공유자원	공유 자원이 없음	Disk
데이터동기화 방안	Network를 통한 복제	Disk를 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 적용에 큰 무리가 없음	Disk 공유를 위한 고비용의 전용망이 요구되므로 실질적으로는 거리제한이 존재
데이터정합성*	Network 복제 특성상 노드간 데이터불일치 현상을 억제하기 위한 별도의 고려가 필요	Disk를 공유하므로 노드간 데이터정합성이 보장
치명적인 Failure 요소	Network Failure시 관련 노드 데이터동기화 불가	Disk Failure시 시스템 전체 서비스 불가능
적합 시스템	**완벽한 데이터정합성보다는 빠른 성능 요구**	**성능보다는 완벽한 데이터정합성 요구**
관련 대표 DBMS 기술	**이중화 (replication)**	**RAC (Real Application Cluster)**
관련 DBMS 벤더	ALTIBASE HDB, DB2, MS-SQL, ORACLE, SYBASE	ORACLE

■ “성능, 시스템 구축비용, 거리적용”과 “데이터정합성” 측면에서의 **trade-off**

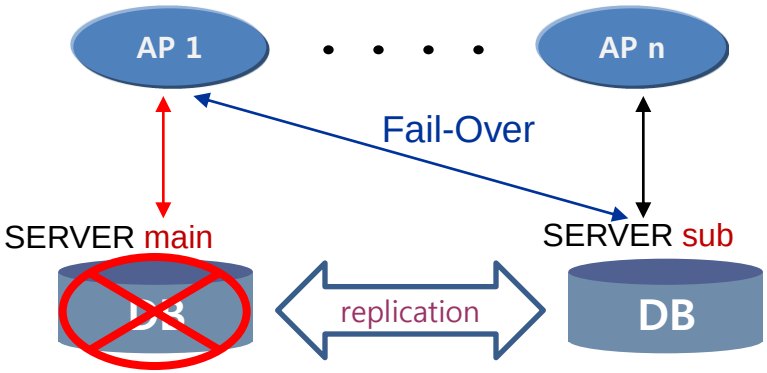
Replication

❖ 이중화의 정의

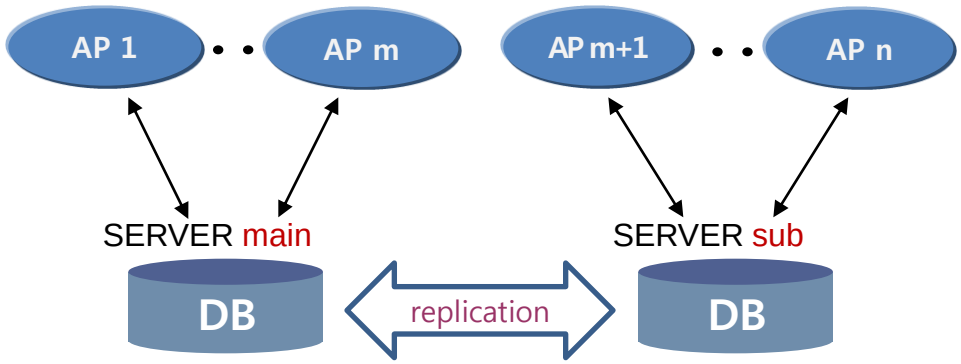
- 하나의 DB 의 변경된 내용을 Network를 통하여 다른 여러 DB 들에 복제하는 기술

❖ 이중화의 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]

ALTIBASE

ALTIBASE REPLICATION

ALTIBASE Replication Features

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 HDB패키지 불필요 사용자에 목적에 따라 ALTIBASE HDB를 가변적으로 사용 가능
리두로그 기반	리두로그를 실시간 전송하는 레코드 단위 이중화
테이블 단위 관리	이중화를 수행할 레코드를 테이블 단위로 관리 운영 중 이중화 할 테이블을 실시간으로 추가, 삭제 가능
lazy, eager 복제방식 지원	복제방식으로 lazy, eager 모두 지원
고속복제	lazy 복제방식의 경우 마스터 트랜잭션에 수행속도에 영향을 주지 않으면서 마스터 트랜잭션의 95% 이상에 달하는 빠른 속도로 복제 가능 (Giga bit LAN 환경의 UNIX 시스템에서의 성능테스트 측정치)
병렬 쓰레드 지원	Eager 모드 사용 시, 송신 쓰레드를 지정한 개수만큼 병렬로 사용하여 이중화 성능 향상
최대 32-way 이중화 지원	하나의 ALTIBASE HDB는 최대 32개의 ALTIBASE HDB와 이중화 가능 업무에 따라 부하분산 및 다양한 시스템 구성이 가능
Point-To-Point 방식의 복제	이중화 노드간 1:1로만 복제, 다른 이중화 노드로 전이되지 않음

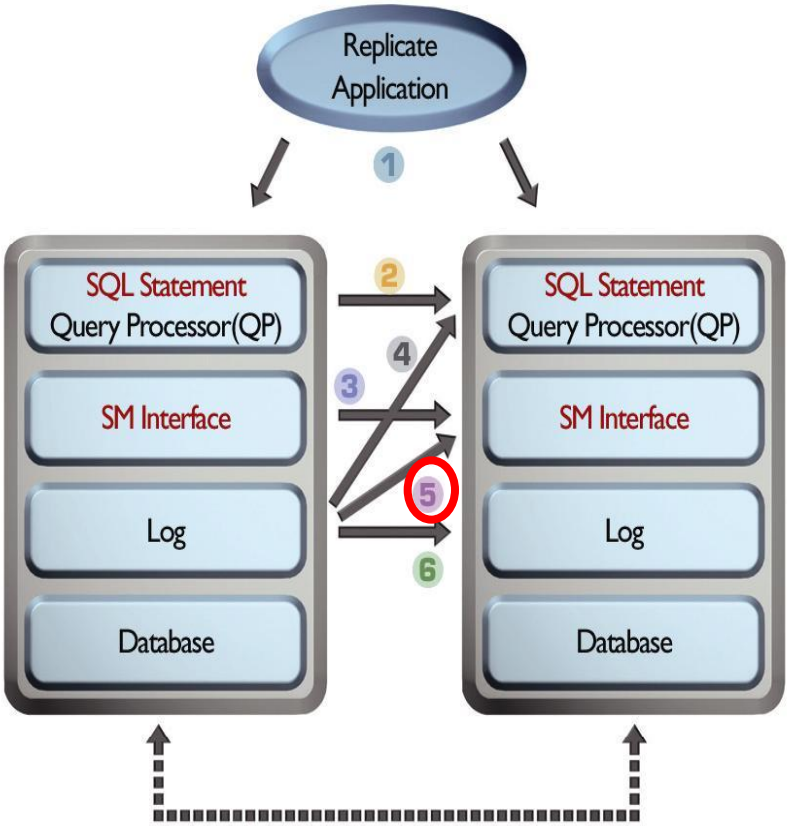
ALTIBASE Replication Features

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 서버에 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
4개 Conflict Resolution 제공	Conflict Resolution을 위한 3개의 Scheme 과 1개의 유틸리티 제공

ALTIBASE HDB Replication Methods

❖ ALTIBASE HDB 이중화 기법

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 가능
   - 치환 비용은 발생하나 이중화 성능이 빠름
6. 리두로그 전송 후 회복 방법으로 반영
   - 속도는 빠르나, Active-Active 불가



ALTIBASE HDB Replication Architecture

❖ Xlog

- SM에서 직접 실행(execution)이 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 서버로 전송
- 이중화 수신 쓰레드(receiver)
  - 전송 받은 Xlog를 SM에 통하여 실행

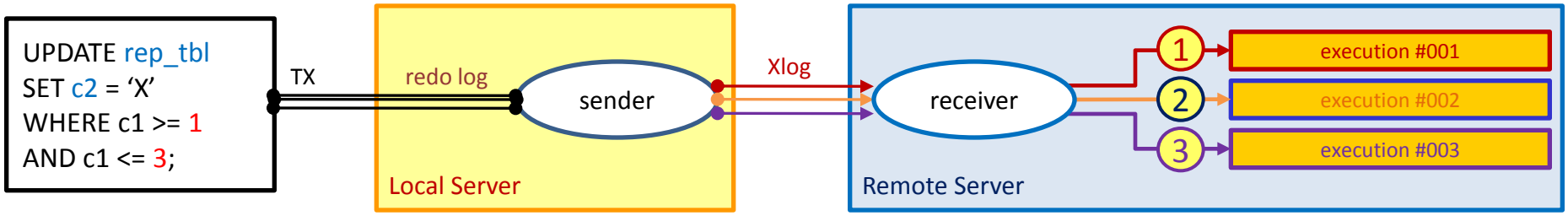
❖ XSN (Xlog Sequence Number)

- sender가 receiver에게 최종 전송한 리두로그의 위치
- 이중화가 중지되었다가 다시 시작할 때 이중화 시작 위치가 됨

ALTIBASE HDB Replication Architecture

❖SQL 구문 관점에서의 상세 이중화 과정

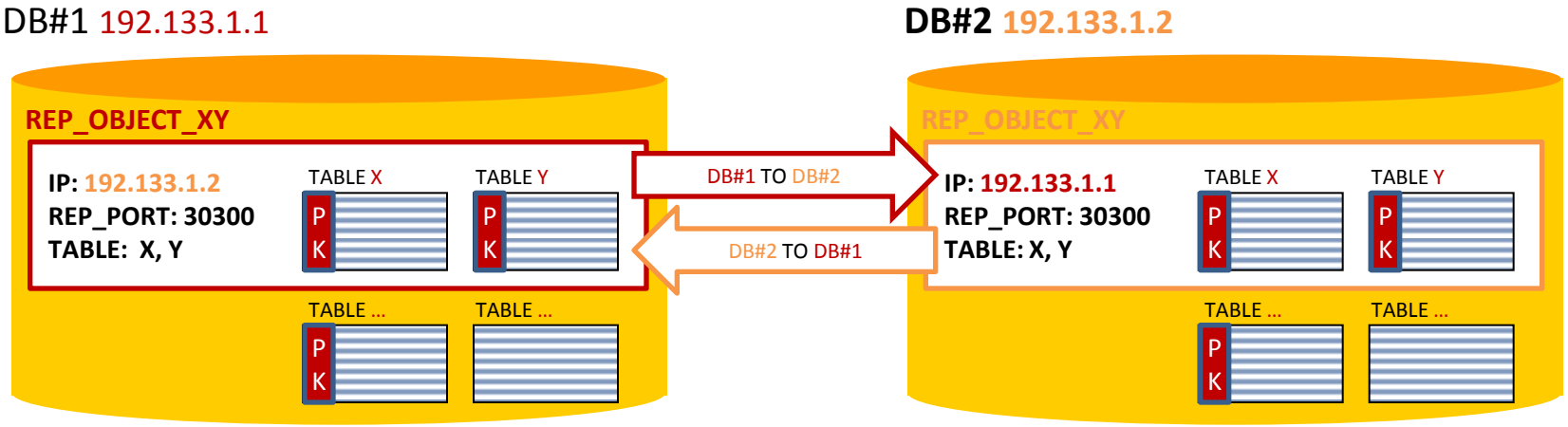
- 3개의 레코드를 변경하는 하나의 UPDATE 구문 수행
- 관련 리두로그가 최소 3개의 Xlog로 치환되어 전송
  - 치환될 때마다 Xlog 단위로 실시간 전송
- Xlog에 대응하는 레코드 변경연산 실행이 총 3회 발생



Replication Object

❖ 이중화 객체

- 이중화 수행을 위한 객체로 지역서버와 원격서버에 모두 존재해야만 이중화가 가능
  - 동일한 이름으로 대응되는 이중화 객체간 1:1로만 이중화 수행, 전이되지 않음
- IP, 이중화 포트(port)번호로 식별
  - 하나의 데이터베이스에 최대 32개 생성 가능
- 객체 별로 다중 IP 지원
- 객체 별로 lazy, eager 복제방식 지정 가능
- 객체 별로 이중화 대상 테이블의 컬럼 및 상세조건과 기타 정보 유지



Replication Object

❖ 이중화 대상 테이블의 요건

- PK 필수
- 이중화를 수행할 컬럼은 이름과 스펙이 완벽히 일치
  - 레코드 중 복제될 컬럼을 이름으로 식별
  - 컬럼 개수 및 순서는 지역서버와 원격서버가 각각 다르더라도 무방
  - 지역서버의 테이블에는 존재하고 원격서버에는 없는 컬럼은 NULL로 채워짐
  - 컬럼 이름은 동일한데 스펙이 다르면 이중화 구동(START)시점에 에러 발생
  - 데이터타입, 길이, 제약조건 등
  - FK 불가

❖ 이중화에 대한 오해

- 시퀀스도 이중화가 가능한가?

Replication Object

❖ 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 서버의 역할 지정, 생략 시 미지정으로 설정
- option : 오프라인 이중화 같은 이중화 객체에 대한 부가 기능
- replication_name : 이중화 객체 명, 이중화를 수행하려는 서버간 이름이 동일해야 함
- remote_host_ip : 원격서버의 IP 주소
- remote_replication_port_no : 원격서버의 이중화 수신 포트번호*
- FROM - TO : 이중화 대상 테이블을 지역서버, 원격서버 순으로 명시

Replication Object

❖ 이중화 대상 서버 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 ;
```

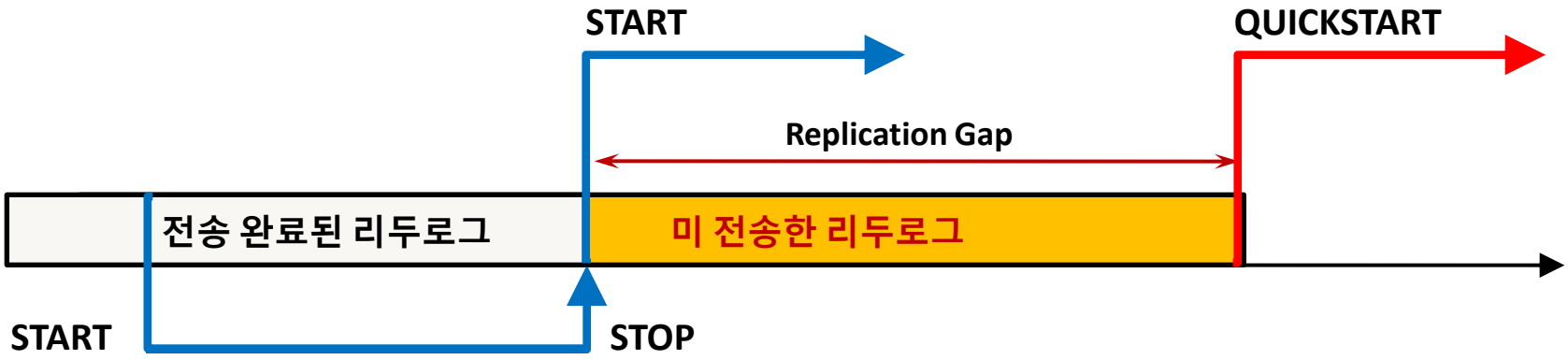
❖ 이중화 객체 삭제 구문

```
DROP REPLICATION replication_name ;
```

Replication Control

❖이중화 제어

- START - 마지막 이중화 수행시점을 기점으로 이중화를 시작 (일반구동)
- QUICKSTART - 미 전송 리두로그를 포기, 현재를 기점으로 이중화를 시작 (긴급구동)
- STOP - 현재를 기점으로 이중화를 중지



❖관련 구문

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

Session Control

❖세션의 이중화 제어

- 수행시점부터 해당 세션에서 발생하는 트랜잭션에 대한 이중화 여부를 지정
- 트랜잭션 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

Replication Setting Step

❖ 이중화 대상 ALTIBASE HDB 선정

- 캐릭터세트가 동일해야 함
- 내셔널 캐릭터세트가 동일해야 함 (5.3.3 higher)
- 이중화와 관련된 ALTIBASE HDB 내부요소의 버전이 동일해야 함
  - ALTIBASE HDB major 버전이 같다면 대부분 동일하나 반드시 확인이 필요
  - Replication Protocol 버전
  - Meta 버전

```
# altibase -v
version 6.1.1.0.5 XEON_LINUX_redhat_Enterprise_ES4-64bit-6.1.1.0.5-release-GCC3.4.6 (xeon-redhat-linux-gnu) Aug  7 2012 15:45:18, binary db version 6.1.1, meta version 5.10.1, cm protocol version 5.6.3, replication protocol version 6.1.1
```

Table Clone

❖테이블 복제

- 이중화 객체를 이용한 테이블 복제 기능
- 지역서버 테이블의 모든 레코드를 원격서버의 테이블로 INSERT하는 방식
  - 이중화가 중지(STOP)된 상태에서만 수행 가능
  - 복제완료 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
- 지역서버에서 테이블 복제 구문 수행

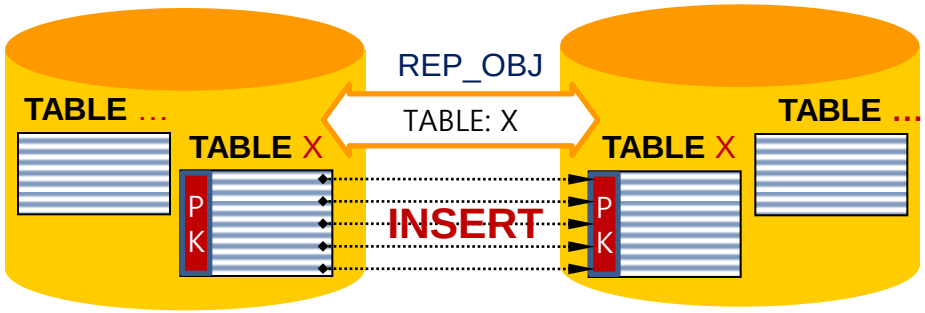


Table Clone

❖관련 구문

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

- SYNC: 이중화를 위한 ALTIBASE HDB간 요구사항 확인
- 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: 테이블 복제를 수행할 테이블을 명시, 생략 시 해당 이중화 객체의 모든 테이블이 대상

Table Clone

❖유의 사항

- Active-Active 구성에서는 시스템 서비스 중에 수행하지 않아야 함
  - 테이블 복제가 완료되기 전까지는 해당 이중화 객체의 다른 테이블들이 이중화되지 않음
  - Active-Standby 구성이라면 사용 가능하나 일시적인 부하발생 고려
- 대응되는 원격서버 테이블의 모든 레코드를 삭제 후 수행해야 함
  - 동일한 PK의 레코드가 이미 존재 시 삽입충돌이 발생하여 복제 실패
  - 동일한 PK에 대한 INSERT 연산으로 인한 DBMS 부하 발생
  - 이중화 trace 로그파일에 다량의 삽입충돌 에러 발생
- 원격서버 테이블의 레코드 삭제 시에는 TRUNCATE 수행을 권장
  - 사용자 실수 방지 차원
  - DELETE로 원격서버 테이블의 레코드 삭제하여 지역서버의 레코드도 삭제

DDL Operation

❖ 이중화 객체에 대한 DDL 수행

- 이중화와 관련된 모든 구문은 SYS 사용자로만 수행가능
- 이중화 객체 변경(ALTER)에 대한 모든 구문은 이중화가 중지(STOP)된 상태에서만 가능
- 이중화 대상 테이블은 DDL 수행이 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

DDL Operation

❖프로퍼티를 통하여 허용 가능한 DDL

- ALTER TABLE *table_name* {ADD | DROP} COLUMN
- ALTER TABLE *table_name* ALTER COLUMN *column_name* {SET | DROP} DEFAULT
- ALTER TABLE *table_name* TRUNCATE PARTITION
- TRUNCATE TABLE
- {CREATE | DROP} INDEX

❖수행 절차

- REPLICATION_DDL_ENABLE 을 1로 설정 후, 허용가능 DDL 수행

❖유의사항

- 세션의 이중화 속성이 NONE일 경우는 수행되지 않음
  - 사용자가 명시적으로 세션의 이중화 속성을 NONE으로 변경한 경우는 DEFAULT로 변경 필요
- DDL 수행은 이중화되지 않으므로 관련된 서버에서 모두 동일하게 수행해 주어야 함
- ALTER TABLE ~ MODIFY COLUMN은 수행할 수 없음

DDL Operation Job Example

❖특정 이중화 대상 테이블에 새로운 컬럼을 추가하는 절차

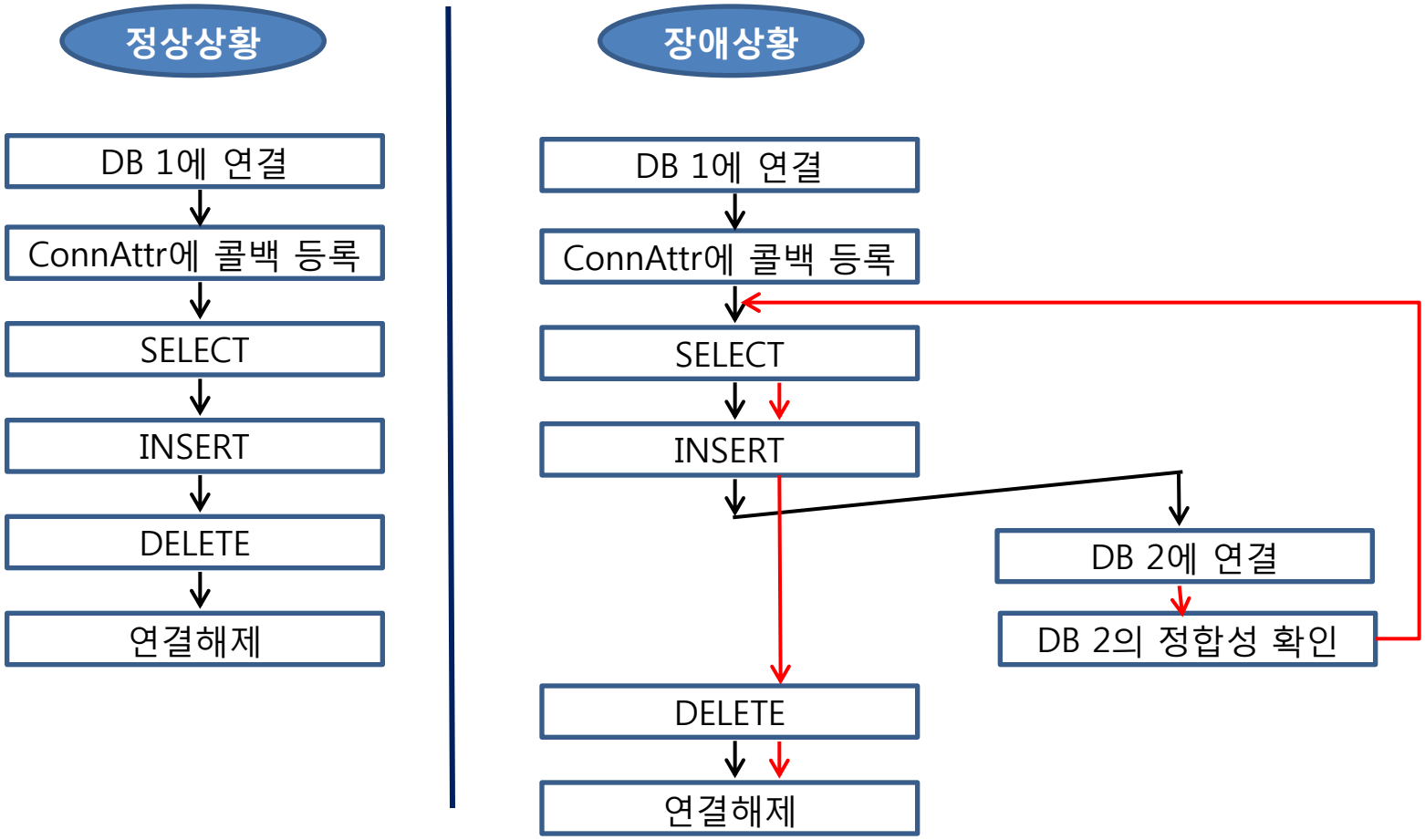
- ➢ 작업 요건
  - 예정된 작업시간 동안 해당 테이블에 변경연산이 발생하지 않아야 함
  - SELECT 연산은 변경연산과 무관하므로 조회 서비스는 가능

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 job start----------** iSQL> connect user01/user01; iSQL> ALTER TABLE rep_tbl ADD COLUMN (new_col CHAR(9)); **------DDL job 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 job start----------** iSQL> connect user01/user01; iSQL> ALTER TABLE rep_tbl ADD COLUMN (new_col CHAR(9)); **------DDL job 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**

Fail-Over

❖Fail-Over란?

- DBMS가 탑재된 장비에서 장애 발생, 네트워크 경로에서 장애 발생, DBMS가 비정상 종료되어 장애가 발생했을 경우 이를 극복하는 것을 의미



Fail-Over

❖장애를 인식하는 시점에 따른 Fail-Over의 분류

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

Fail-Over 속성

속성	상세설명
AlternativeServer	장애 발생 시 접속하게 될 가용 서버를 나타내며 기존의 IP Address 부분에 다음과 같은 형식으로 추가한다. (IP Address1:port1, IP Address2:port2, ….)
ConnectionRetryCount	가용 서버 실패 시, 접속 시도 반복 횟수
ConnectionRetryDelay	가용 서버 접속 실패 시, 다시 접속을 시도하기 전에 대기하는 시간 (초 단위)
LoadBalance	"ON" 으로 설정하면 최초 접속 시도 시에 기본 서버와 가용 서버를 포함하여 랜덤으로 선택한다. "OFF" 로 설정하면 최초 접속 시도 시에 기본 서버에 접속하고, 접속에 실패하면 AlternativeSever 로 기술한 서버에 접속한다.
SessionFailOver	Fail-Over 할 때의 종류를 결정한다. ON : STF, OFF: CTF

Fail-Over Configuration

❖JDBC

- Connection url 부분에 Fail-Over 관련 속성을 지정해준다.

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

❖ODBC, APRE

- 연결 스트링 부분에 Fail-Over 관련 속성을 지정해준다.

```
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;LoadBalance=off;
SessionFailOver=on
```

ALTIBASE

REPLICATION SYSTEM DESIGN

Replication Architecture

❖복제방식 구현

- lazy - 이중화를 모듈화하여 마스터 트랜잭션에 영향을 주지 않는 고속 복제로 구현
- eager - 2PC(2 Phase commit)와 유사한 형식으로 구현

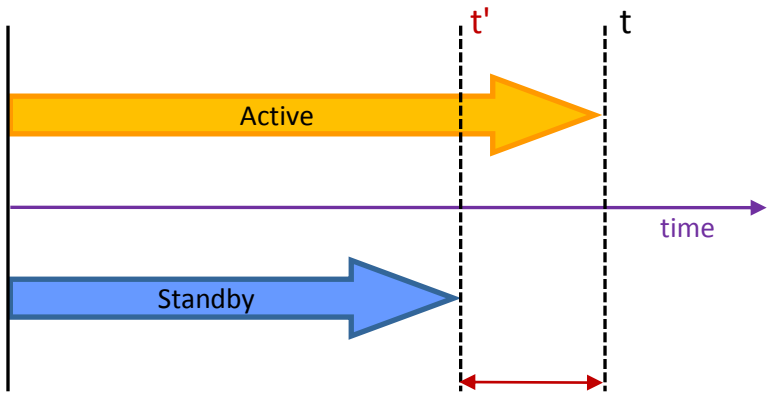
❖구성방식 구현

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

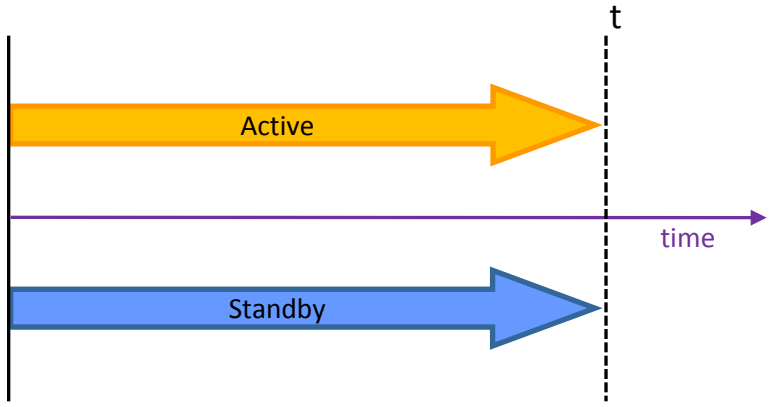
lazy vs. eager

❖이중화 노드간 동기화 시점에 따른 이중화 복제방식

- lazy: 비동기식. 마스터 트랜잭션과 이중화 트랜잭션이 별개로 수행
  이중화 지연은 발생하나 트랜잭션 처리 성능이 빠름
- eager: 동기식. 마스터 트랜잭션과 이중화 트랜잭션이 하나로 수행
  이중화 지연은 발생하지 않으나 트랜잭션 처리 성능이 느림



[그림1. lazy]

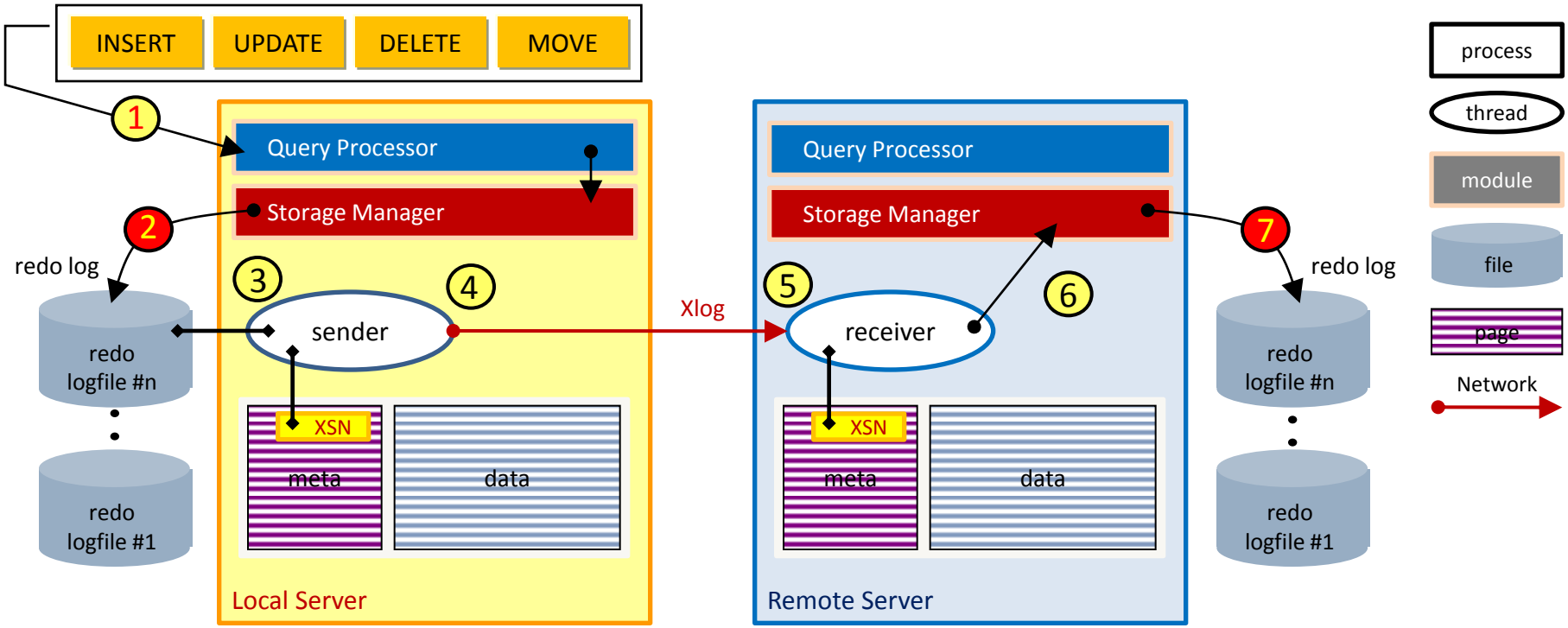


[그림2. eager]

❖주요 특징

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

Replication Architecture



Lazy

- 마스터 트랜잭션 = 1 + 2
- 이중화 트랜잭션 = 3 + 4 + 5 + 6 + 7

Eager

- 트랜잭션 = 1 + 2 + 3 + 4 + 5 + 6 + 7
  - 이중화 트랜잭션(7)까지 이상없이 반영되었을 때 마스터 트랜잭션(2)을 확정

Replication Architecture

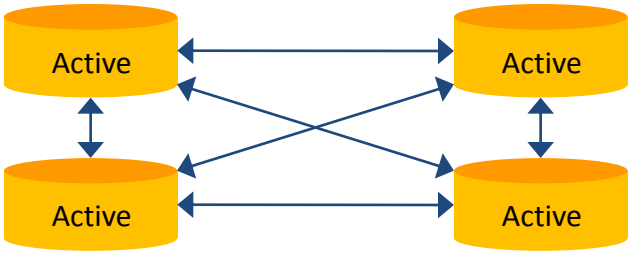
❖ 이중화 갭(Replication Gap)

- 이중화 지연 정도를 성능뷰 v$repgap 에서 수치로 제공
- 리두로그 일련번호인 SN(Sequence Number)과 XSN을 통한 계산
- 이중화갭 = [지역서버의 최신 SN] - [지역서버의 최신 XSN]

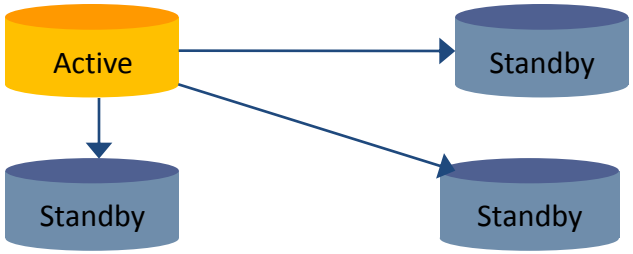
Active-Active vs. Active-Standby

❖ 이중화 갭(Replication Gap)

- 변경연산 주체에 따른 이중화 노드 구성방식
- Active-Active 모든 이중화 노드에서 변경연산이 가능하나 변경연산간 충돌 가능성 있음
- Active-Standby 특정 이중화 노드에서만 변경연산이 가능하나 변경연산간 충돌 가능성 없음



[그림1. Active-Active]



[그림2. Active-Standby]

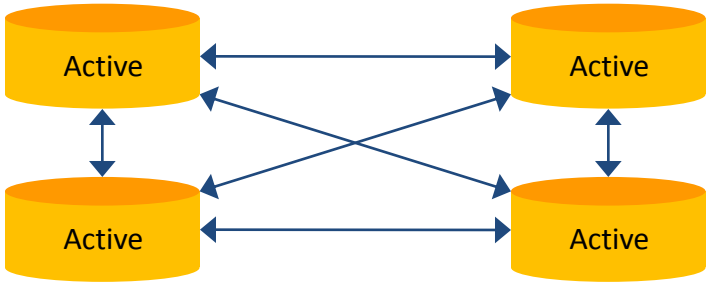
❖ 주요 특징

- 변경연산의 부하분산과 변경연산간 충돌 가능성 측면에서의 trade-off
- 구성방식에 따라 응용프로그램 고려사항 발생
- Active-Active - lock 발생에 대한 고려
- Active-Standby - 노드 역할에 따른 응용프로그램 유지

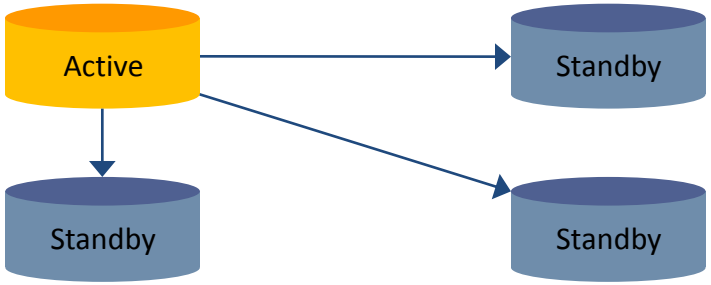
Replication System Setting

❖이중화 시스템 설정 방법

- Active-Active, Active-Standby(fail-over)
  - 한 개의 서버당 [전체서버개수 - 1] 개의 이중화 객체를 생성
  - 모든 서버에서 sender가 구동
- Active-Standby(backup)
  - Active 서버는 [전체서버개수 - 1] 개의 이중화 객체를 생성
  - Standby 서버는 Active에 대응되는 1개의 이중화 객체만 생성
  - Active 서버에서만 sender가 구동



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



[그림2. Active-Standby(backup)]

Replication System Matrix

❖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 대응방안 수립
- Network 장애*의 경우는 eager 복제방식이라 할지라도 장애복구 방안을 수립해야 함

Replication System Design

❖변경연산 전용 서버 지정 방식

- ➢ 서버1은 변경연산 전용, 서버2는 조회연산 전용으로 설계

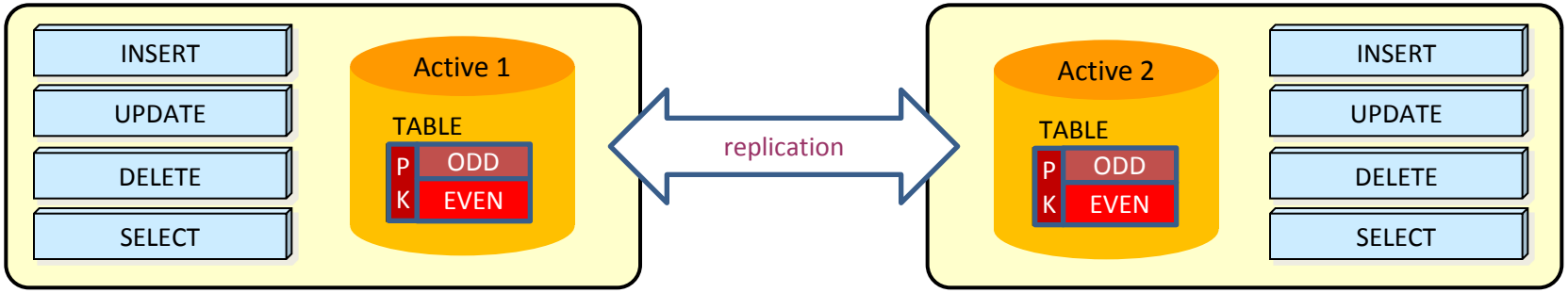
❖주요특징

- ➢ 조회 연산의 부하분산만 가능하므로 응용이 제한적

Replication System Design

❖PK를 노드 개수 만큼 분할하는 방식

- 서버1은 짝수전용, 서버2는 홀수전용으로 설계



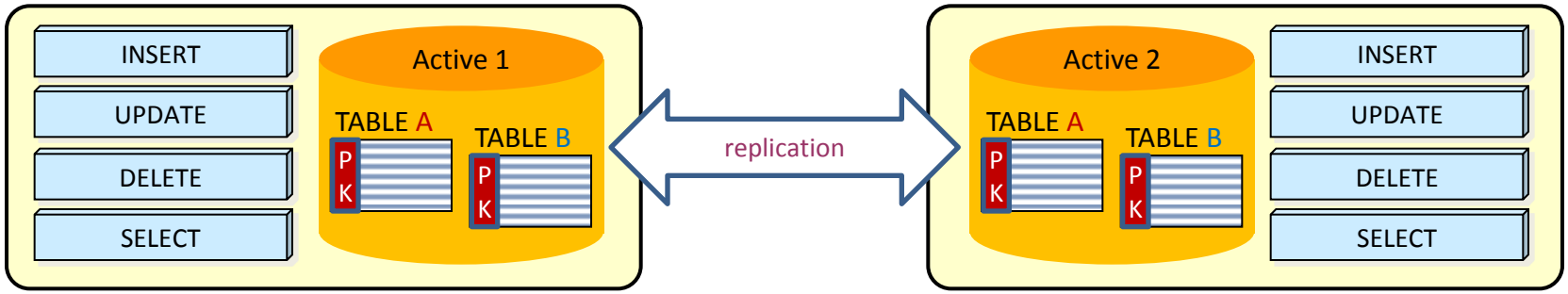
❖주요특징

- 변경 연산의 부하분산이 가능하나 응용프로그램 구현에 주의필요

Replication System Design

❖업무에 따라 테이블을 분할하는 방식

- 서버1은 A업무 테이블 변경연산 전용, 서버2는 B업무 테이블 변경연산 전용으로 설계



❖주요특징

- 변경 연산의 부하분산이 가능하나 복합 업무를 처리하는 경우에 대한 고려 필요

ALTIBASE

CONFLICT RESOLUTION

Replication Conflict

❖ 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

Replication Conflict Scenario

❖ 삽입충돌, 변경충돌(2), 삭제출동 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**

Replication Conflict Scenario

❖ 변경충돌(1) 발생 시나리오

- 변경충돌에 대한 감지를 하지 않을 경우에 발생할 수 있는 시나리오
- 동일한 PK에 대한 서로 다른 노드의 변경연산 결과가 성공, 별도의 감지가 필요

time	node A	node B
T1	**update(PK1, 'A') & commit**	
T2	send log(T1)	
T3		**update(PK1, 'B') & commit**
T4		send log(T3)
T5	receive log(node B's T4) ***SUCCESS**	
T6		**receive log(node A's T2) *SUCCESS**
T7	**select(PK1) - 'B' *UPDATE CONFLICT(1)**	**select(PK1) - 'A' *UPDATE CONFLICT(1)**

Conflict Resolution

❖ALTIBASE HDB에서 제공하는 이중화 충돌 해결

- DBMS Level
  - User-oriented scheme
  - Timestamps-based scheme
  - Master-Slave scheme
- Utility Level
  - 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 정책에 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	

User-oriented scheme

User-oriented scheme

- 기본적으로 설정되어 있는 이중화 충돌 해결 정책
- 이중화 충돌 발생 시 해당 레코드와 관련된 연산을 무시
- 사용자가 확인 후 조치할 수 있도록 이중화 trace 로그파일에 리포트만 수행
- $ALTIBASE_HOME/trc/altibase_rp.log

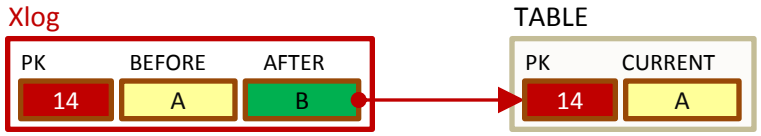
이중화 충돌 유형에 따른 감지와 처리

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	

User-oriented scheme

❖Value-based 기법

- 삽입/변경충돌을 감지하기 위한 기법
- 변경전의 값과 현재 값을 비교하여 일치하지 않을 경우 변경충돌로 판별
  - UPDATE 연산에 대한 Xlog는 관련된 컬럼의 변경전의 값과 변경후의 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
삽입 충돌	동일한 PK에 대한 INSERT 연산	REPLICATION_INSERT_REPLACE 값이 0 이면, 이중화 충돌 리포트 REPLICATION_INSERT_REPLACE 값이 1 이면, 이중화 충돌을 무시하고 반영
변경 충돌	동일한 PK에 대한 UPDATE 연산	REPLICATION_UPDATE_REPLACE 값이 0 이면, 이중화 충돌 리포트 REPLICATION_UPDATE_REPLACE 값이 1 이면, 이중화 충돌을 무시하고 반영

User-oriented scheme

❖유의사항

- LOB 컬럼은 동일한 PK 변경에 대한 충돌을 감지하지 않음
- LOB 데이터타입의 특성
  - 리두로그 자체에 변경전의 값이 없어 감지가 불가능

Master-Slave scheme

❖Master-Slave Scheme

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

❖상세 처리방식

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

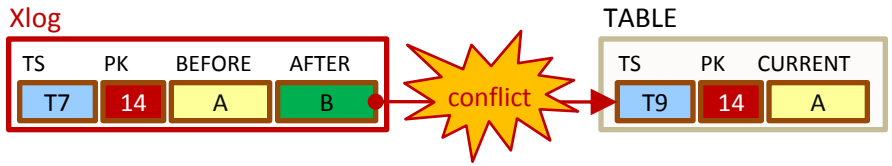
- ➢ 하나의 Master 이중화 객체는 반드시 대응되는 Slave 객체가 있어야만 이중화 가능
  - ▪ Master-Master (X), Slave-Slave (X), Master or Slave-NONE (X)

Timestamps-based scheme

❖Timestamps-based scheme

- ➢ 프로퍼티로 설정 가능한 이중화 충돌 해결 정책
  - REPLICATION_TIMESTAMP_RESOLUTION을 1로 설정 (기본값은 0)
- ➢ TIMESTAMP 컬럼을 활용한 우선순위 판별
  - 시간으로 순서화되므로 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

Timestamps-based scheme

❖유의사항

- 이중화 서버간 시간이 일치해야 함
- TIMESTAMP 컬럼 필수
  - 프로퍼티를 설정하였다 하더라도TIMESTAMP 컬럼이 없는 테이블은 적용되지 않음
  - TIMESTAMP 컬럼은 사용자가 직접 추가하여야 함

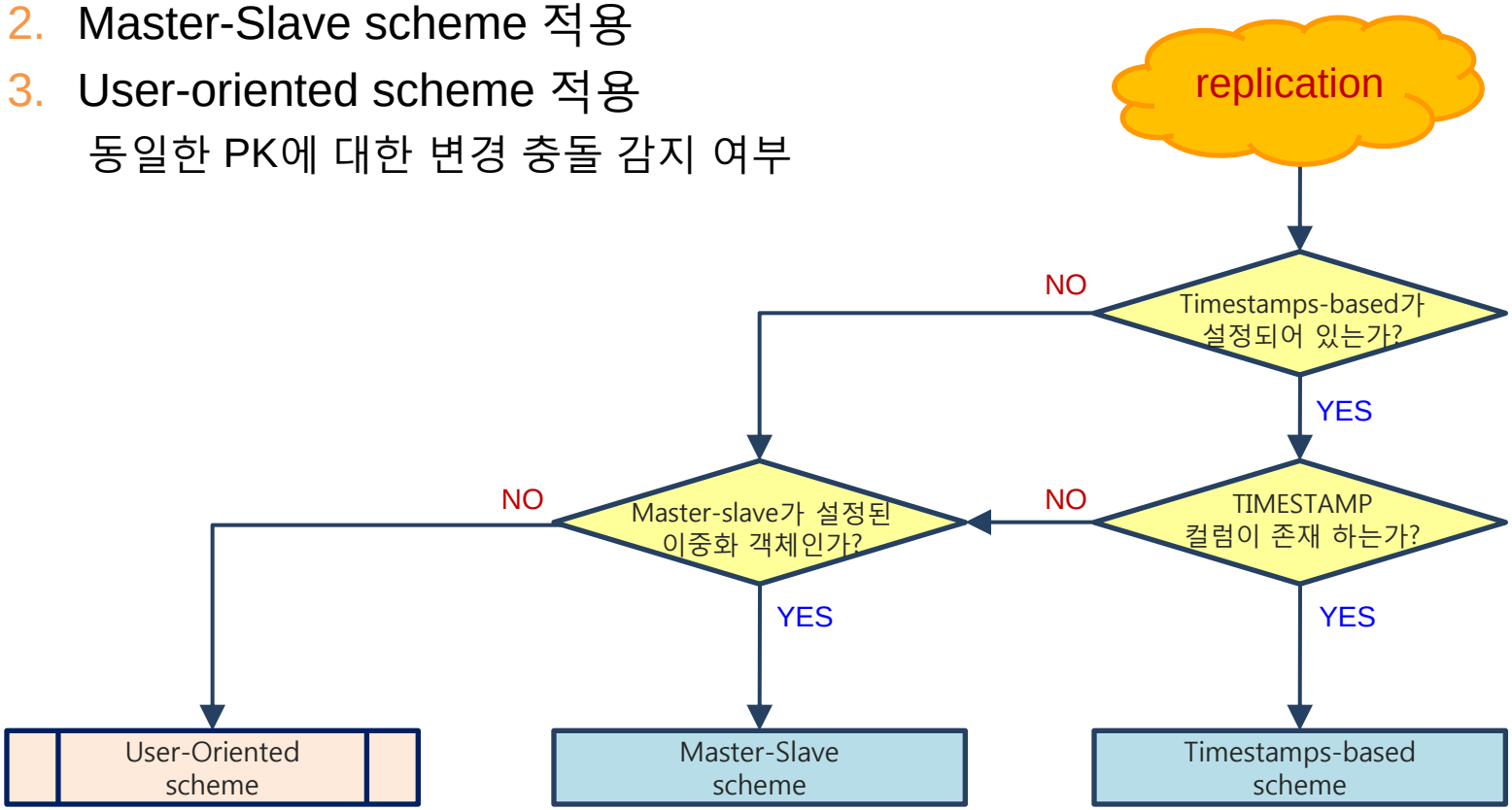
❖참고사항

- TIMESTAMP 컬럼으로 인한 레코드당 8byte의 추가 저장공간이 발생
- TIMESTAMP 컬럼이 추가적으로 전송되므로 이중화를 위한 통신 비용이 증가

Conflict Resolution Flow

❖ 이중화 충돌 해결 scheme 혼용시의 처리 흐름

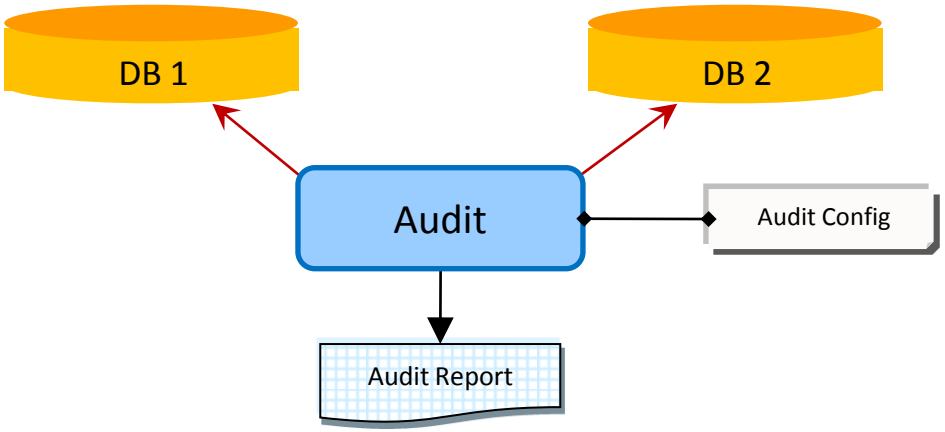
1. Timestamps-based scheme가 우선적으로 적용
   단, TIMESTAMP 컬럼이 없다면 처리하지 않음
2. Master-Slave scheme 적용
3. User-oriented scheme 적용
   동일한 PK에 대한 변경 충돌 감지 여부



AUDIT

AUDIT

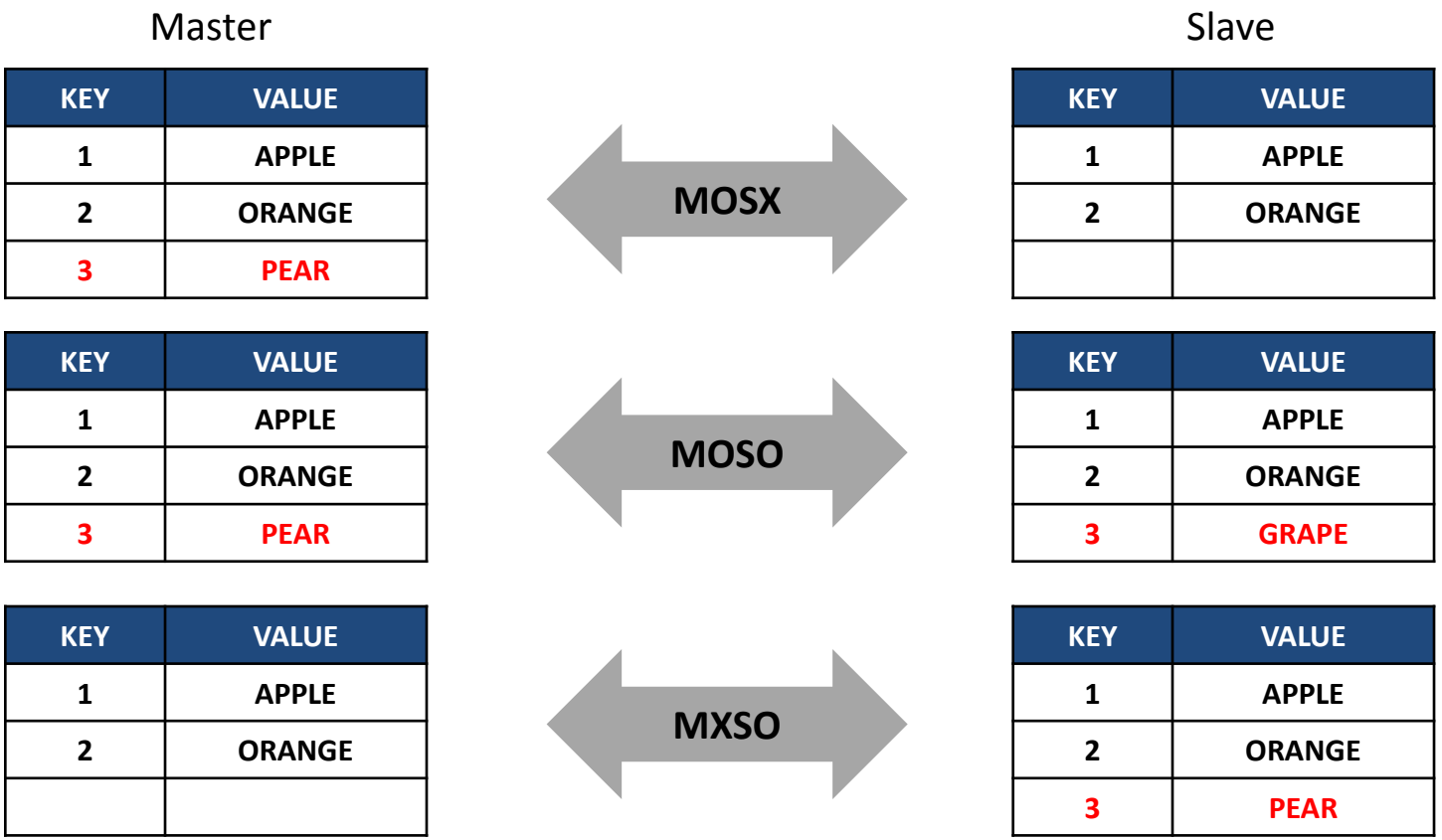
- 두개의 DB를 테이블 단위로 비교, 데이터 동기화 관련 기능을 수행하는 유틸리티
- 이중화 충돌로 인한 데이터불일치를 사용자 판단에 의해 일괄적으로 해결하게 함이 목적
- 기본적으로 마스터 (MASTER) DB를 기준으로 슬래이브(SLAVE) DB를 일치시키는 정책을 채택
- 대상 DB가 변경중에는 정상적으로 수행되지 않을 수 있으므로 주의가 필요



AUDIT

❖Audit 의 종류

➢ Master DB와 Slave DB간의 데이터 불일치가 발생하는 세가지 경우



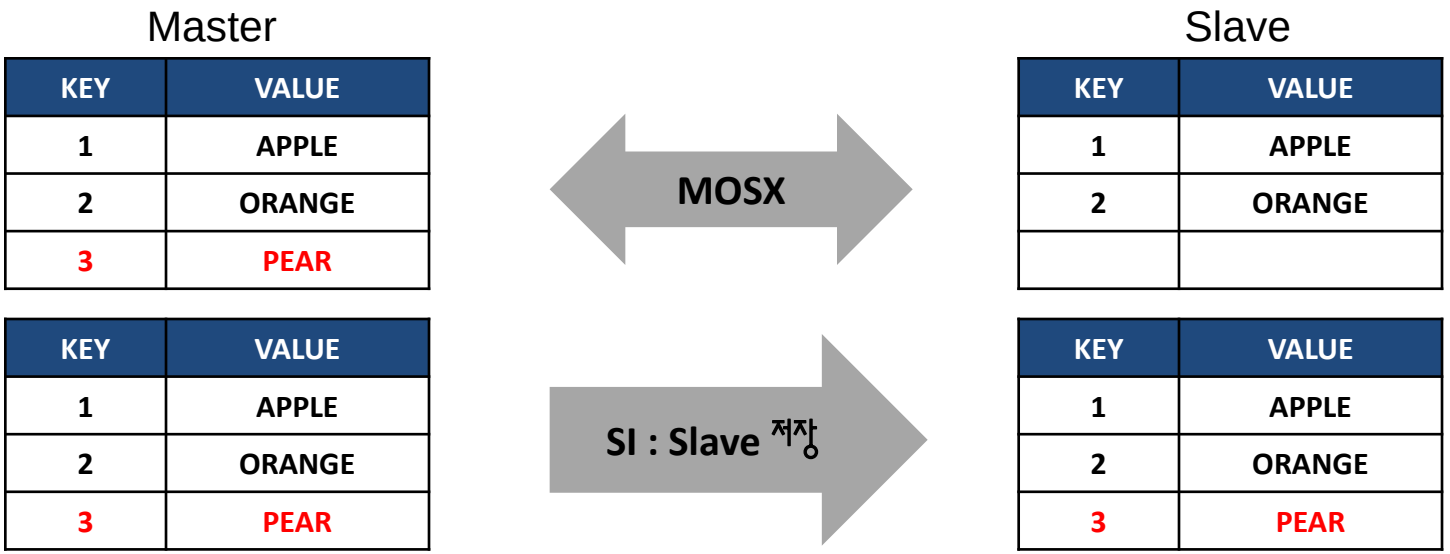
AUDIT

❖ 데이터 동기화 정책

- Audit 이후 Master DB와 Slave DB간의 데이터 불일치가 발생하는 세 가지 경우에 따라 다음의 데이터 동기화 정책을 사용

❖ SI (Slave Database Insert)

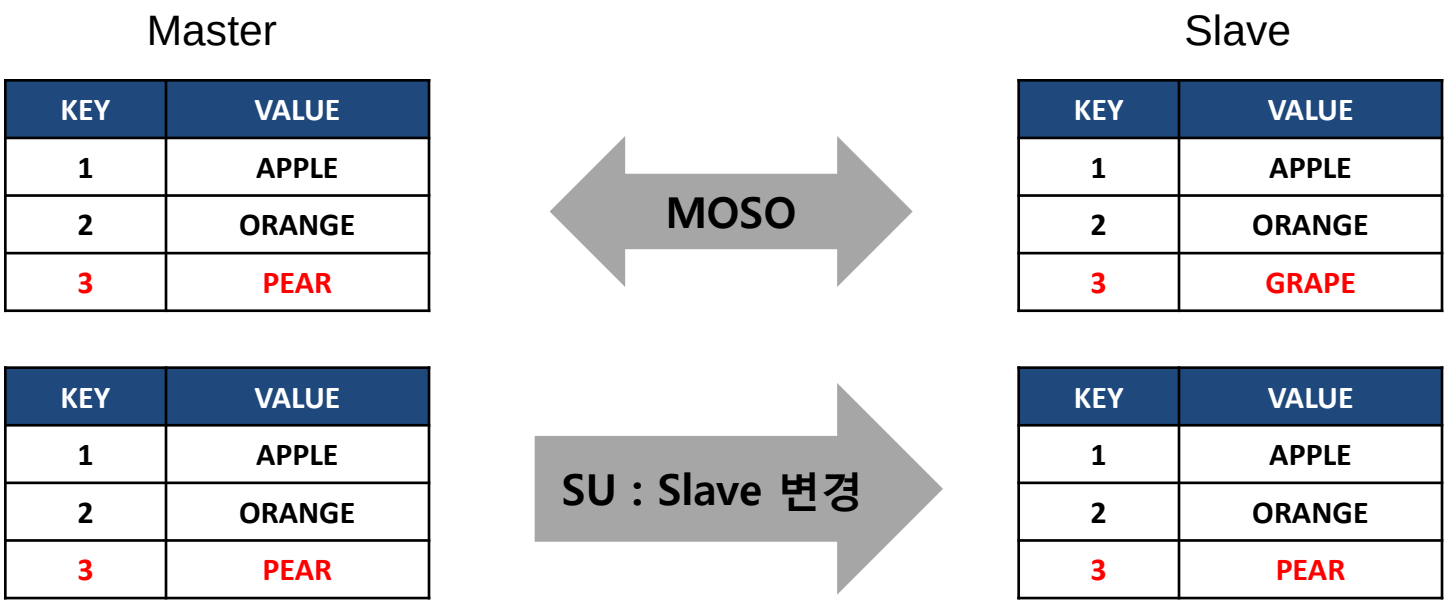
- MOSX 불일치를 해소하는 정책으로, Master DB의 레코드를 Slave DB에 삽입 (Insert) 한다.



AUDIT

SU (Slave Database Update)

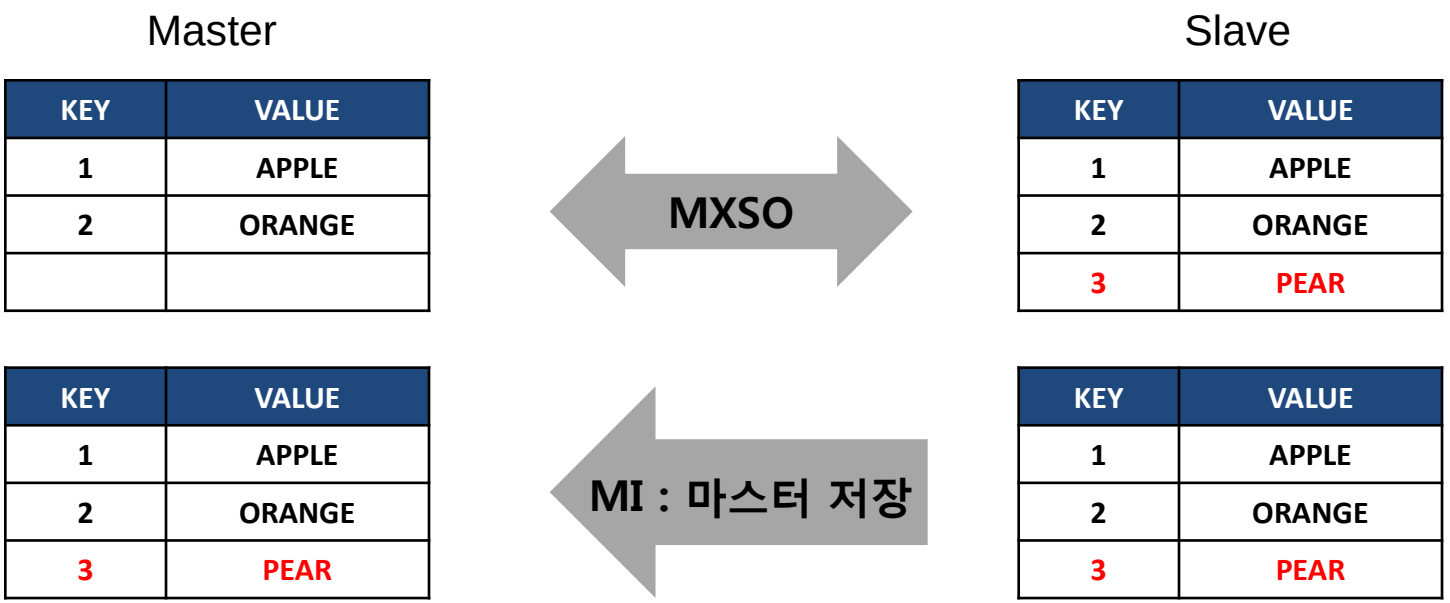
- MOSO 불일치를 해소하는 정책으로, Master DB의 레코드 내용으로 Slave DB를 변경(Update) 한다.



AUDIT

MI (Master Database Insert)

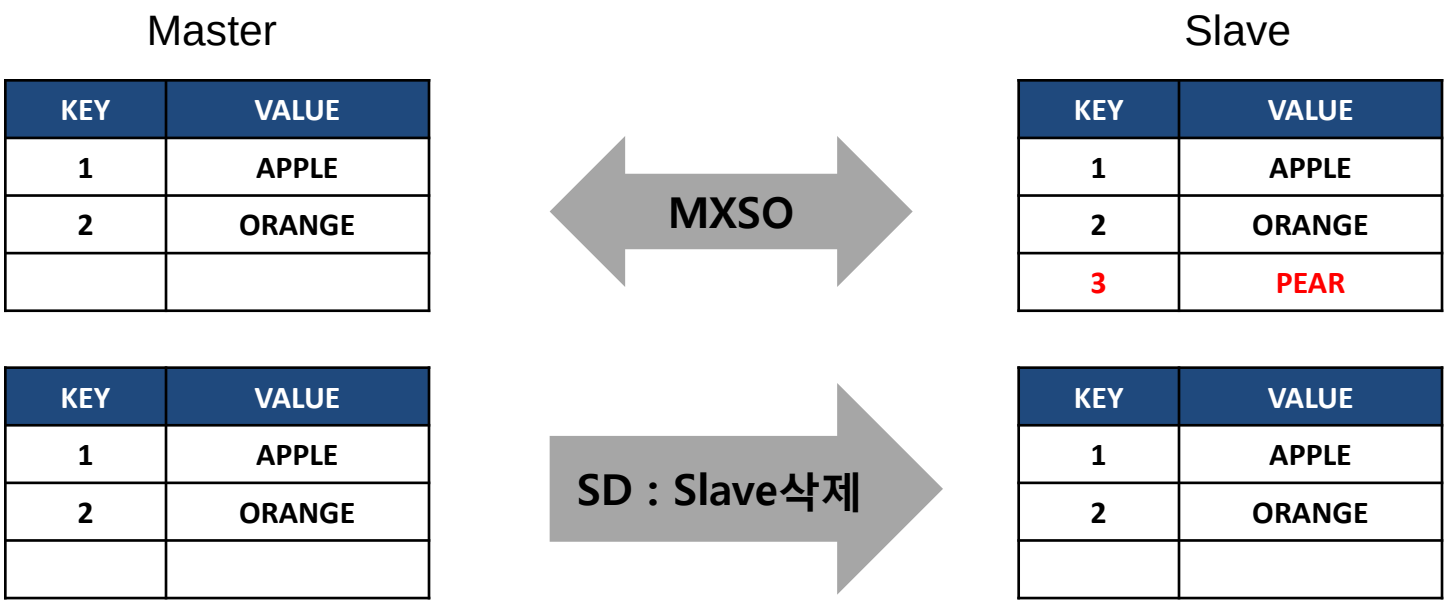
- MXSO 불일치를 해소하는 정책으로, Slave DB의 레코드를 Master DB에 삽입 (Insert) 한다.



AUDIT

SD (Slave Database Delete)

- MXSO 불일치를 해소하는 정책으로, Slave DB의 레코드를 삭제(Delete)한다.



AUDIT

AUDIT의 구성

- Audit 실행시 서로 다른 두개의 데이터베이스 간에 구성될 Audit정책을 기술한 설정 파일이 필요

- Audit 설정 파일 : Audit을 실행하기 위한 옵션을 지정하는 설정 파일로 이 파일은 Connection정보, Audit 기능 설정, 일치 정책 등의 내용을 포함하고 있으며 $ALTIBASE_HOME/audit/sample.cfg 샘플파일을 참조하여 작성

- 비교(DIFF) 기능 : Master DB와 Slave DB의 불일치 데이터를 식별하여 결과파일로 생성

- 일치(SYNC)기능 : 두 데이터베이스 간의 불일치를 해소하고, 불일치 데이터에 대한 정보를 결과 파일로 생성

AUDIT

❖ 비교(DIFF) 기능

- Master DB와 Slave DB 간의 이중화 작업에서 발생한 불일치 데이터를 식별하여 실행 결과 파일로 생성

```
# 비교 작업을 위한 감사 환경 파일 설정 시작

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

# Slave DB 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
```

AUDIT

```
# 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
# AUTODETECT_UNIQ_INX = ON

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

# Audit 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
```

AUDIT

Audit 명령 실행

```
$ audit –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}
```

AUDIT

❖ 일치(SYNC) 기능

- Master DB와 Slave DB간의 불일치 데이터를 식별하여 Audit 환경 파일의 일치 정책에 따라 불일치를 해소하는 기능

```
# 비교 작업을 위한 Audit 환경 파일 설정 시작

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

# Slave DB 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
```

AUDIT

```
# 불일치 Audit 정책 설정
DELETE_IN_SLAVE = ON
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = OFF
UPDATE_TO_SLAVE = ON
AUTODETECT_UNIQ_INX = ON

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

# Audit 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
```

AUDIT

- Audit 명령 실행

```
$ audit –f sample.cfg
```

- 실행 결과 파일 내용은 다음과 같으며 만일 실패한 레코드가 있다면, 해당 레코드는 로그 파일에 원인과 레코드 내용이 출력
  - 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
```

AUDIT

```
-------------------------------------------------------------
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
```

AUDIT

❖ 이중화 환경에서의 일치(SYNC) 작업 절차

- Application 서비스 중지
  - AUDIT SYNC 중 변경 트랜잭션 발생시 데이터가 불일치 될수 있음
- 이중화 갭 확인
  - 이중화 로그 원격 서버에 모두 반영 되었는지 (rep_gap=0) 확인

```
iSQL> SELECT rep_gap FROM v$repgap;
```

- 이중화 중지

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

- AUDIT 수행
  - REPLICATION QUICKSTART AUDIT으로 반영된 트랜잭션 로그가 이중화로 전송되는 것을 방지

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

AUDIT

❖Audit 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)

ALTIBASE

REPLICATION OPTIMIZATION

Overview

❖ALTIBASE 이중화 최적화를 위한 공통 고려사항

- 응용프로그램 고려요소
  - 제한사항
  - 권고되는 logic

- 이중화갭 최소화 및 TPS 향상을 위한 필수 고려요소
  - Network 최적화
  - 트랜잭션 튜닝

- Active-Active 구성 방식에서의 고려사항
  - 부분실패와 부분롤백
  - 이중화에서의 lock 체계
  - 이중화 lock timeout
  - 이중화 데드락
  - 이중화 sender 튜닝

OPTIMIZATION

❖PK에 대한 UPDATE 연산 불가능

- PK 변경 시 이중화 충돌이 발생 가능하므로 에러로 처리됨

❖이중화 객체 설계

- 이중화 객체간 트랜잭션 순서가 관계없다면 이중화 객체를 분리하는 것을 권장
  - 이중화 송신 쓰레드를 객체 수만큼 생성함으로 속도 향상 기대 가능
  - 메모리 DB 관련 이중화 객체와 디스크 DB 관련 이중화 객체의 분리는 필수
  - 디스크 DB의 느린 수행 속도로 인해 메모리 DB의 이중화 수행 속도까지 느려지는 현상 방지

Application Logic

❖INSERT 결과가 중복에러일 때 UPDATE를 수행하는 logic의 변경 권장

- 일반적으로 흔히 사용되지만 DBMS에 부하를 주는 잘못된 logic
  - 대부분의 DBMS는 무조건 INSERT 연산을 수행 후 결과를 반환하는 구조
- 리두로그 복제 기법의 ALTIBASE 이중화에서는 또 다른 문제가 발생
  - 실패한 INSERT 연산에 대한 INSERT, 롤백 연산도 이중화 되어 불필요한 연산부하 증가
  - 이중화 trace 로그파일에 다량의 삽입충돌 에러 메시지가 발생
    - 무시해도 되는 다량의 삽입충돌 에러로 인해 주요한 에러에 대한 식별이 어려워 짐
- UPDATE 결과가 없을 때 INSERT 하는 logic 으로 변경 권장

DBMS에 부하를 주는 잘못된 logic	권장되는 logic

…
INSERT INTO TABLE
If (SQLCODE == SQL_DUP_ERROR)
{
  UPDATE TABLE …
}
…
	…
	UPDATE TABLE …
	If (SQLCODE == SQL_NO_DATA)
	{
	  INSERT INTO TABLE …
	}
	…

Transaction Tuning

❖Network 대역폭을 초과하는 트랜잭션의 튜닝

- 기본적으로 벌크(bulk)성 UPDATE/DELETE 연산은 자제
  - 업무 요건상 발생 시 완화 방안
  - LIMIT 절을 활용한 분할 수행
  - 양쪽 서버에서 이중화 세션 제어 옵션을 NONE으로 설정 후 동일한 연산을 수행
- 테이블 전체를 주기적으로 삭제하는 경우 DELETE보다는 TRUNCATE를 활용
  - 프로퍼티 설정 후 양쪽서버에서 TRUNCATE 구문 수행

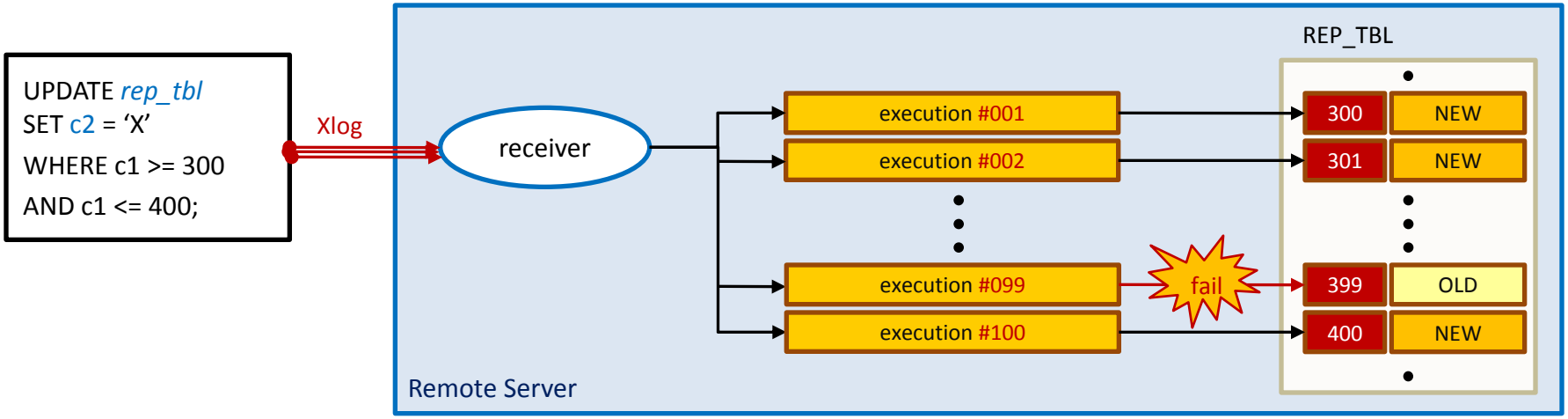
❖UPDATE 연산의 최소화

- 업무상 불필요한 UPDATE 연산의 최소화만으로도 Network 대역폭 사용량 감소 가능
  - UPDATE 연산에 대한 Xlog는 이중화 출동 감지를 위해 관련된 컬럼의 변경전의 값과 변경후의 값으로 구성되므로 가장 큼

Partial Fail & Partial Rollback

❖부분실패

- 원격서버에서 이중화를 반영할 때 트랜잭션의 일부인 특정레코드의 반영실패
- 특정 레코드 반영실패로 인해 전체 트랜잭션이 실패하는 상황 발생



[100건의 레코드를 변경하는 하나의 구문이 이중화 연산 중 1건의 반영실패 발생하는 상황]

❖반영실패 요인

- 이중화 충돌, 이중화 lock timeout 초과

Partial Fail & Partial Rollback

❖부분실패 발생시 복제방식에 따른 처리

구분	lazy	eager
지역서버	-	commit시 에러 발생
원격서버	해당레코드와 관련된 구문과 에러로그를 남김 해당레코드와 관련된 연산에 대한 부분롤백 수행	해당레코드와 관련된 구문과 에러로그를 남김
처리결과	해당레코드는 지역서버만 반영	관련 트랜잭션 전체가 지역서버와 원격서버 모두 반영되지 않음

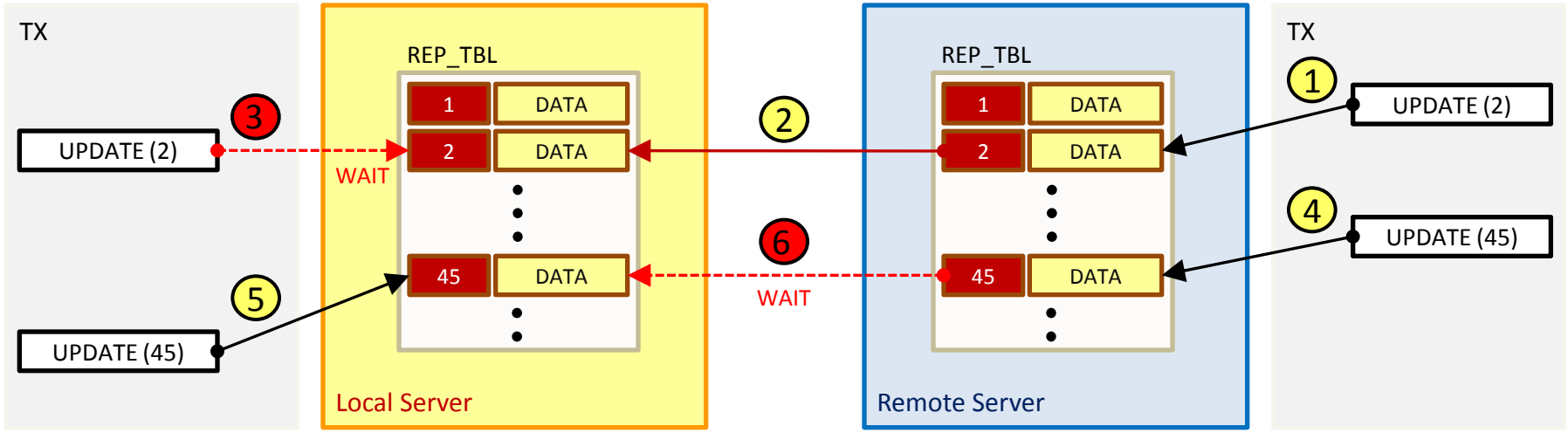
❖부분롤백

- 원격서버에서 이중화를 반영할 때 반영실패가 발생한 레코드만 롤백을 수행하는 기법
- 특정 레코드 반영실패로 인해 전체 트랜잭션이 반영 실패하는 상황을 방지하는 것이 목적
- 부분롤백이 수행된 레코드는 원격서버와 지역서버간의 불일치 상태로 남게 됨

Lock System in Replication

❖ 이중화에서의 lock 체계

- 변경연산에 한하여 단일서버를 사용하는 것과 유사한 레코드 단위 lock 체계를 가짐
  - 단, Network 특성상 더 늦게 발생한 변경연산(5)이 먼저 lock을 획득할 가능성 존재
- 동일한 레코드에 대한 변경연산 시 두 가지 형태로 lock이 발생
  - 이중화 변경연산이 먼저 lock을 획득한 경우 (1~3)
  - 마스터 변경연산이 먼저 lock을 획득한 경우 (4~6)



[lock 획득 시점에 따른 이중화 서버간 lock 획득 흐름]

Lock System in Replication

❖장시간 지속 시 lock 형태에 따른 현상

- ➢ 이중화 변경연산이 먼저 lock을 획득한 경우
  - 수신 서버의 lock 대기가 지속되나 DBMS 차원에서 정상적인 흐름
  - 동기화 측면에서는 크게 문제되지 않음
- ➢ 마스터 변경연산이 먼저 lock을 획득한 경우
  - receiver가 lock 획득을 위해 대기하므로 이중화 수행이 중지 됨
    - lazy - 이중화갭 증가로 인한 데이터불일치 현상 심화
    - eager - 이중화 대상 테이블에 대한 변경연산 후 commit을 수행한 세션은 대기 상태

❖대응방안

- ➢ 이중화 변경연산을 위한 대기시간의 한계를 설정
  - 대기시간 초과시 해당 레코드에 대한 변경연산을 포기하는 정책 적용

Replication Lock Timeout

❖ 이중화 lock timeout

- 이중화를 수행할 때 receiver가 lock을 획득하기 위한 최대 대기시간
- lock으로 인해 이중화 수행 전체에 영향을 미치는 것을 방지함이 목적
- 이중화 수신 측의 프로퍼티를 통해 적용
  - REPLICATION_LOCK_TIMEOUT, 기본값은 5 sec

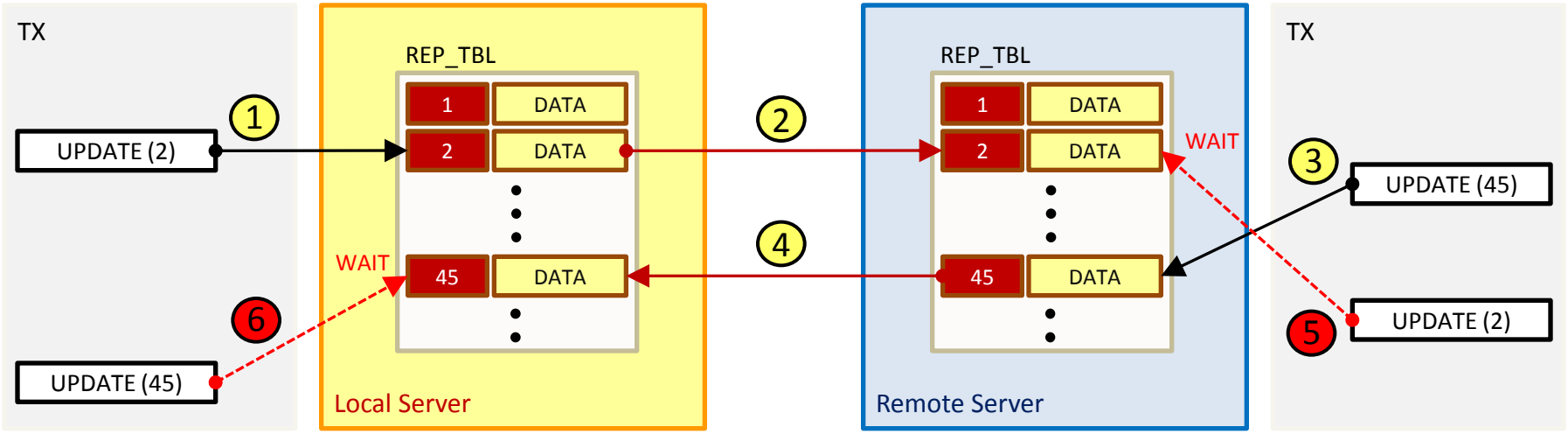
❖ 고려사항

- 시스템에 따른 적정한 이중화 lock timeout 수치 설정
  - 지나치게 클 경우, 이중화 수행 자체가 장시간 중지될 수 있음
  - 지나치게 작을 경우, 잦은 이중화 lock timeout 발생으로 데이터불일치가 심각해질 수 있음
- 이중화 대상 서버의 DML 수행시점
  - 이중화 lock timeout이 발생하여 일부 레코드가 반영되지 않을 수 있으므로 주의

Replication Deadlock

❖ 이중화 데드락

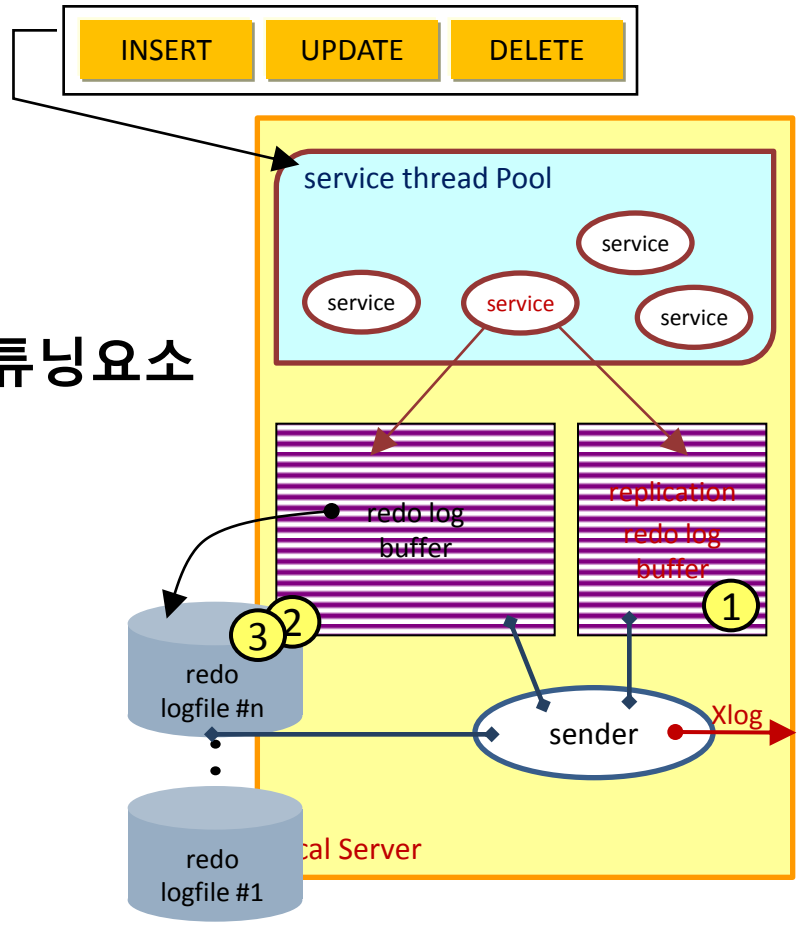
- 마스터 변경연산과 이중화 변경연산이 서로 엉켜 무한대기 상태
- DBMS에서 감지가 불가능한 Network 데드락
- 변경연산 트랜잭션의 최대시간을 제한하는 프로퍼티를 통하여 제한 가능
  - UTRANS_TIMEOUT
- 최선의 대안은 변경연산간의 경합을 고려한 시스템 도입



Replication sender Tuning

❖이중화 sender의 리두로그 접근 순서

- ➢ 이중화 전용 리두로그 버퍼
- ➢ 리두로그 버퍼
- ➢ 리두로그 파일



❖sender의 성능저하시 프로퍼티를 통한 튜닝요소

- ➢ 이중화 전용 리두로그 버퍼의 크기 증가
  - REPLICATION_LOG_BUFFER_SIZE
    - 최대 4G까지 설정 가능, 기본값은 30M
    - 리두로그 파일을 직접 접근하지 않기 위해 설정
- ➢ 미리 읽을 리두로그파일의 개수 증가
  - REPLICATION_PREFETCH_LOGFILE_COUNT
    - 최대 1024까지 설정 가능, 기본값은 0
    - 로그파일을 미리 읽어두므로 속도 향상
    - 2차적인 튜닝 대상

ALTIBASE

REPLICATION MONITORING

Overview

❖DBMS 모니터링 분류

분류	설명
내부 모니터링	메타테이블 및 성능뷰에 대한 질의를 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
    - ALTIMON
    - Replication Manager
  - 별도의 응용프로그램 작성

Replication Monitoring

❖ 이중화 관련 주요 내부 모니터링 요소

- 이중화 갭
- sender
- receiver

❖ 이중화 관련 주요 외부 모니터링 요소

- network
- 리두로그 파일시스템
- ALTIBASE HDB구동 상태
- OS 구동 상태

❖ 이중화 관련 주요 trace 로그 요소

- 이중화 trace 로그파일

Replication Monitoring

❖추가적인 내부 모니터링 요소

- ➢ 이중화와 직접적인 관련이 있는 요소는 아니나 영향을 미치므로 모니터링 필요
  - 벌크(BULK)성 UPDATE/DELETE 쿼리 수행
    - 성능뷰 v$statement, v$session 등을 활용
    - 장시간 수행되는 변경연산 트랜잭션
    - 성능뷰 v$transaction, v$statement, v$session 등을 활용

Replication Meta Table

❖ 이중화 관련 메타 테이블

➢ 이중화 객체 생성 만으로도 관련 정보를 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 Meta Table

❖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 서버 IP 주소
PORT_NO	원격 서버 이중화 포트 번호

Replication Meta Table

❖SYS_REPL_ITEMS_

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

Replication Performance View

❖ 이중화 관련 성능뷰

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

Replication Performance View

❖이중화 관련 성능뷰

- ➢ 이중화를 운영하여 sender 및 receiver가 활성화되었을 때만 조회 가
  - ▪ v$repgap에서 제공하는 이중화갭 수치는 sender가 구동중인 상태에서만 조회 가능

성능뷰	설명
v$repgap	**이중화갭 정보**
v$repsender	**sender 정보**
v$repsender_transtbl	sender의 트랜잭션 테이블 정보
v$repreceiver	**receiver 정보**
v$repreceiver_column	receiver의 이중화 대상 칼럼 정보
v$repreceiver_transtbl	receiver의 트랜잭션 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 Meta Table

v$repgap

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

v$repsender

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

Replication Meta Table

v$repreceiver

Column Name	설명
REP_NAME	이중화 객체의 이름
MY_IP	지역서버의 IP 주소
PEER_IP	원격서버의 IP 주소
APPLY_XSN	처리중인 XSN

Replication Monitoring Example (1)

❖ 지역서버

```
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            60602217     61617892       1015675      529             0
1 row selected.

iSQL> SELECT rep_name, xsn, status, repl_mode FROM v$repsender;
REP_NAME        XSN          STATUS         REPL_MODE
-----------------------------------------------------------
REP1            60602217     1              LAZY
1 row selected.

iSQL> SELECT replication_name, xsn, is_started FROM SYSTEM_.SYS_REPLICATIONS_;
REPLICATION_NAME  XSN          IS_STARTED
-------------------------------------------
REP1              60600750     1
1 row selected.

iSQL> exit
$
$ ls $ALTIBASE_HOME/logs
loganchor0  logfile529  logfile532  logfile535  logfile538  logfile541  logfile544
loganchor1  logfile530  logfile533  logfile536  logfile539  logfile542  logfile545
loganchor2  logfile531  logfile534  logfile537  logfile540  logfile543
```

Replication Monitoring Example (1)

❖ 원격서버

```
iSQL> SELECT rep_name, apply_xsn FROM v$repreceiver;
REP_NAME        APPLY_XSN
----------------------------
REP1            60600750
```

Replication Monitoring Example (2)

❖대표적인 이중화 장애 상황에 대응되는 모니터링 적용 예제

- 모든 모니터링 요소는 이중화 갭 모니터링 만으로도 간접적으로 감지가 가능

원인	상황	모니터링 요소	대상
대량 변경연산	대량(BULK)의 UPDATE/DELETE 수행	대량 변경연산 트랜잭션	지역서버
Network 장애	일시적/주기적인 Network 결함 Network 단절	Network	지역서버, 원격서버
송신불가	지역서버의 이중화가 중지(STOP)된 상태	Sender	지역서버
수신불가	원격서버의 ALTIBASE HDB구동 중지	ALTIBASE HDB server	원격서버
	원격서버의 OS failure	OS	
	원격서버의 리두로그파일 관련 파일시스템의 full (1) 장시간 수행되는 원격서버의 변경 트랜잭션으로 인한 대량의 리두로그파일 생성과 같은 ALTIBASE HDB 내부 요인	장시간 수행되는 변경연산 트랜잭션	
	원격서버의 리두로그파일 관련 파일시스템의 full (2) 사용자가 임의로 다른 작업을 위해 해당 파일시스템의 공간을 사용 중인 경우와 같은 ALTIBASE HDB 외부 요인	리두로그 파일시스템	

ALTIBASE

REPLICATION MANAGER

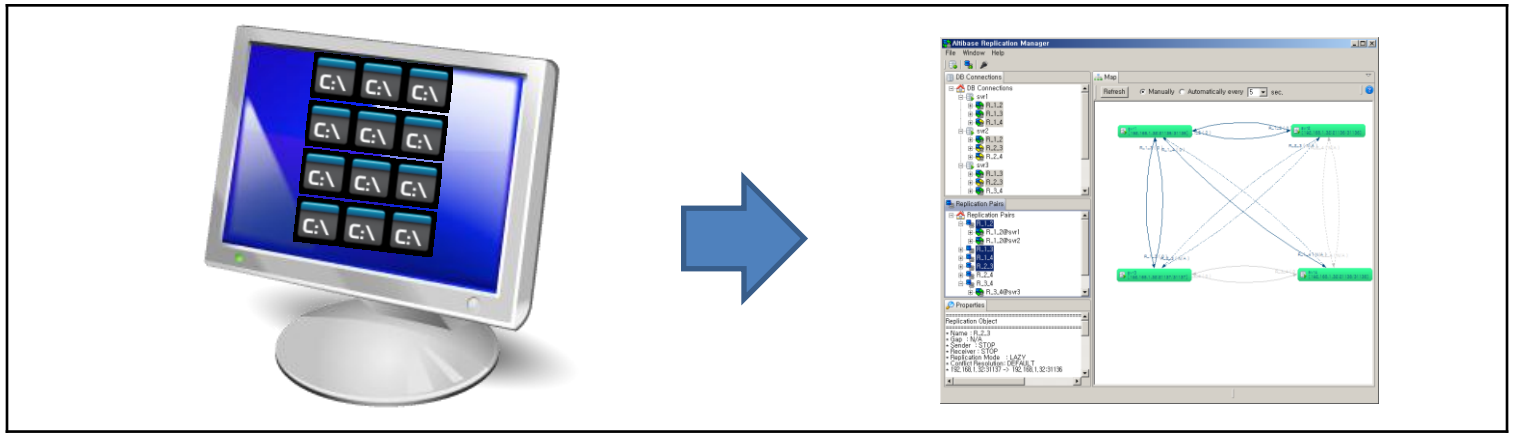
Replication Manager

❖Replication Manager

- ALTIBASE HDB 이중화 관리를 위한 GUI 툴로 다수의 서버에 기동되고 있는 Replication 객체를 편리하게 모니터링이 가능

❖주요 특징

- ALTIBASE HDB 4.3.9 이상에서 사용 가능
- 이중화 객체 상태와 관계 확인
- 마우스 클릭 한번으로 이중화 객체 관리
- 이중화 모니터링 및 상태 분석이 직관적으로 가능



Replication Manager

❖시스템 요구사항

- 하드웨어 요구사항
  - CPU : 800Mhz 펜티엄 III 이상
  - 메인메모리 : 512MB 이상
  - 디스크 : 50MB 이상의 여유공간 (JRE를 위한 별도 저장 공간 필요)
  - 화면 해상도 : 1024 x 768 화소 이상
- 패키지 종류

패키지 이름	컴퓨터 운영 체제	하드웨어	자바실행 환경	윈도우 시스템
ReplicationManager-win32.win32.x86.zip (JRE 포함)	Windows XP, Vista, 7	x86 32-bit	Java 6 or higher	Win32
ReplicationManager-linux.gtk.x86.zip (JRE 포함)	Linux	x86 32-bit	Java 6 or higher	GTK

Replication Manager

❖설치 및 제거

- 다운로드
  - 알티베이스 고객지원서비스 포털(http://support.altibase.com) 에서 다운로드
  - Zip 파일과 같은 압축된 형식으로 제공
- 설치
  - Replication Manager 설치 파일의 압축을 해제하고 디렉토리를 원하는 위치로 이동
  - ALTIBASE HDB 버전에 알맞은 JDBC 드라이버 필요
- 삭제
  - Replication Manager 설치 디렉토리 삭제

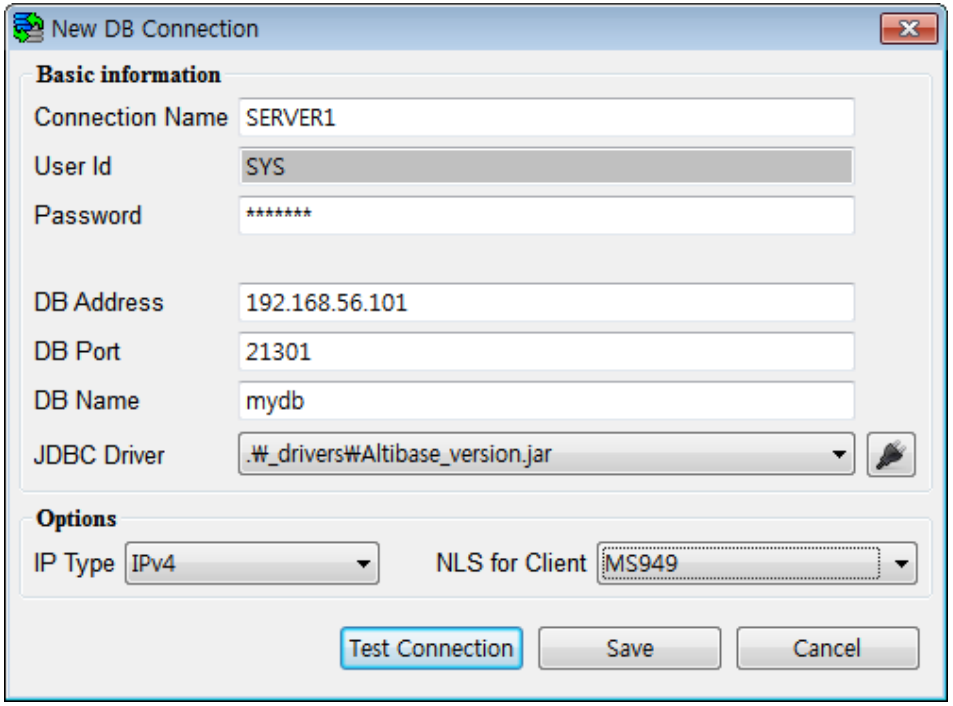
❖Replication Manager 시작과 종료

- 시작
  - 프로그램이 설치된 폴더에서, 윈도우의 경우 "Replication Manager.exe" 실행파일을 더블클릭하고, 리눅스의 경우 "Replication Manager" 응용 프로그램을 실행
- 종료
  - 상위 메뉴에서 "File"을 선택한 다음 "Exit"를 선택하거나, 오른쪽 상단에 있는 X 를 클릭

Replication Manager

❖ 데이터베이스 연결 추가

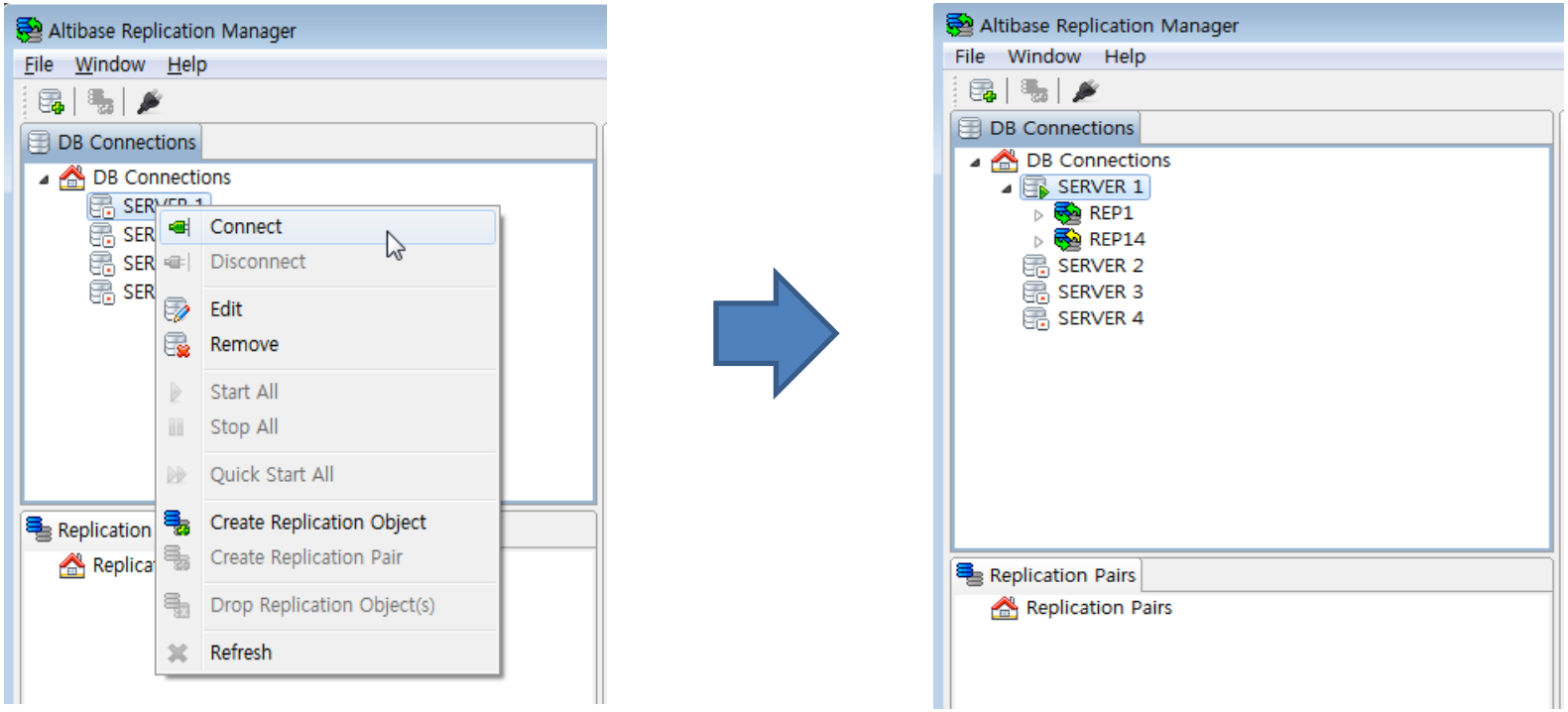
- ➢ 도구모음에서 "New DB Connection" 아이콘을 클릭하거나, "DBConnections"창에서 "DB Connections" 아이콘에 오른쪽 클릭한 다음 표시되는 콘텍스트 메뉴에서 "New DB Connection" 항목을 클릭



Replication Manager

❖ 데이터베이스 연결하기

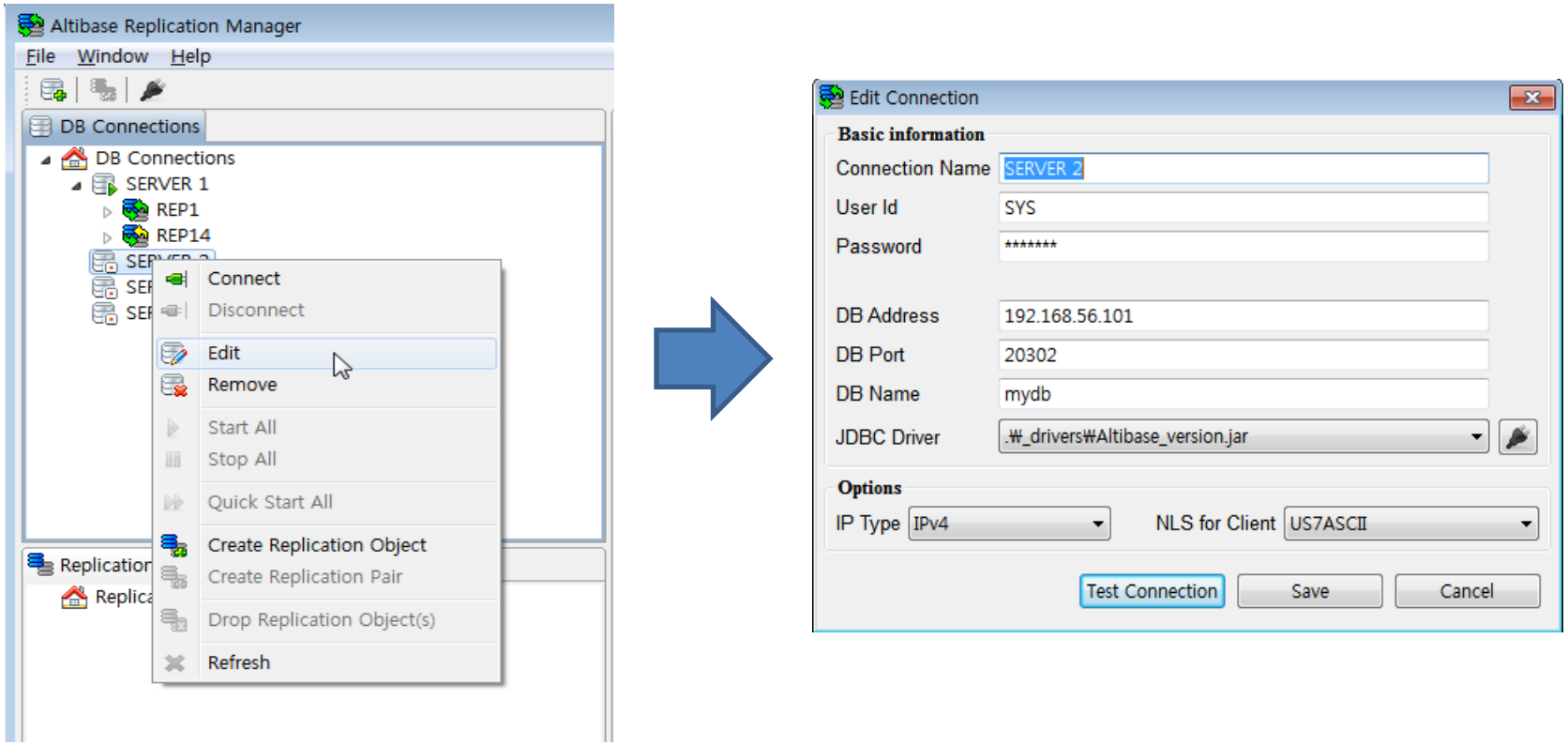
➢ 새로 추가된 데이터베이스에 오른쪽 클릭한 다음 "Connect" 항목을 선택. 시스템 환경에 따라 시간이 좀 걸릴 수도 있음.



Replication Manager

❖ 데이터베이스 연결 정보 편집

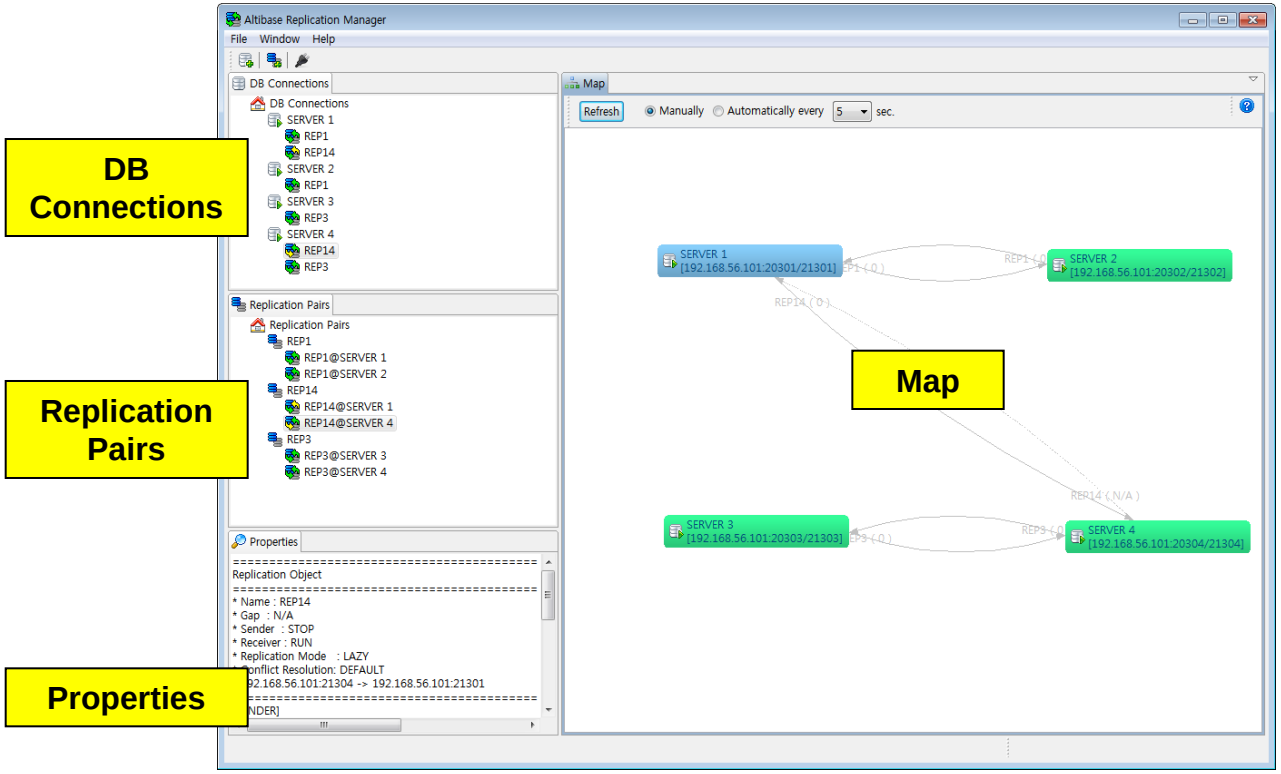
➢ 데이터베이스 연결 정보를 편집하고자 할때 데이터베이스에 오른쪽 클릭하여 콘텍스트 메뉴를 연후 “Edit” 항목을 선택한다.



Replication Manager

❖사용자 인터페이스

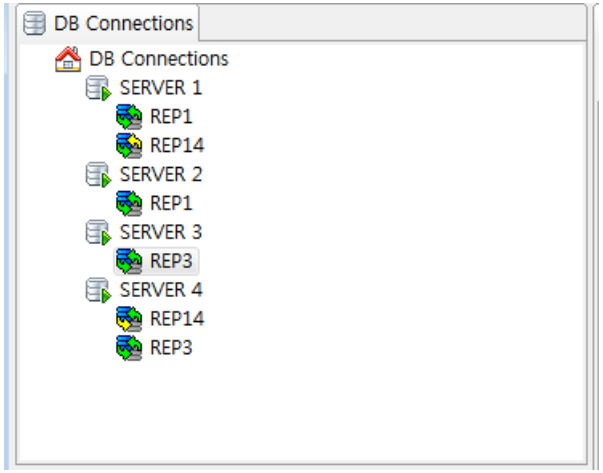
- Replication Manager GUI는 "DB Connections", "Replication Pairs", "Properties", "MAP" 4개의 창으로 구성

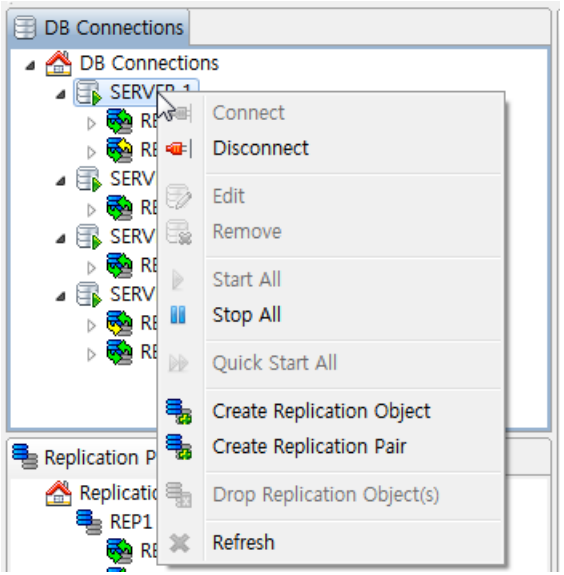


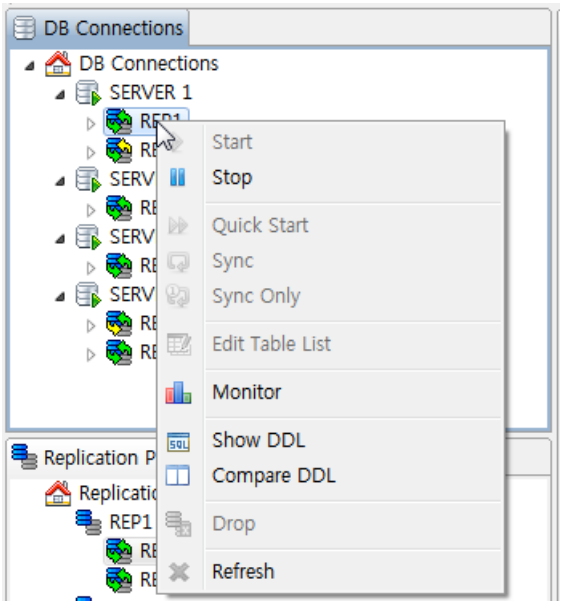
Replication Manager

DB Connections

- 프로그램의 시작 위치이며, 데이터베이스와 이중화 객체 간의 관계를 트리 구조로 보여주는 데이터베이스 중심의 뷰이다



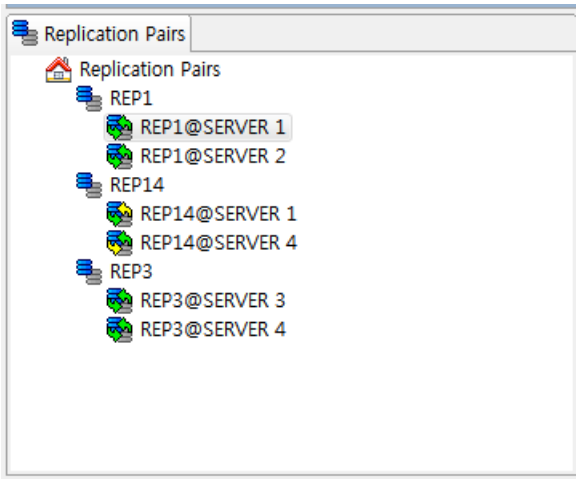


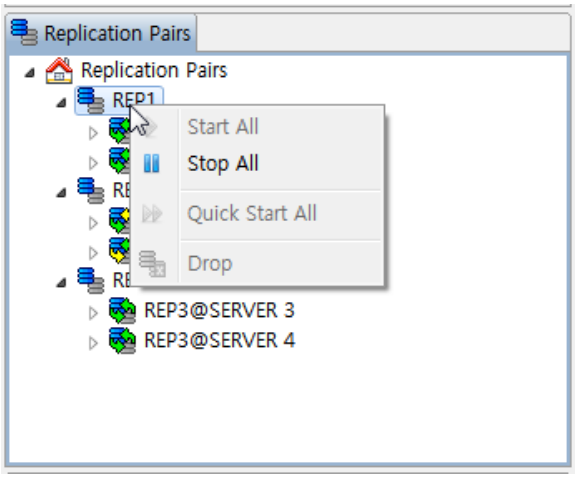


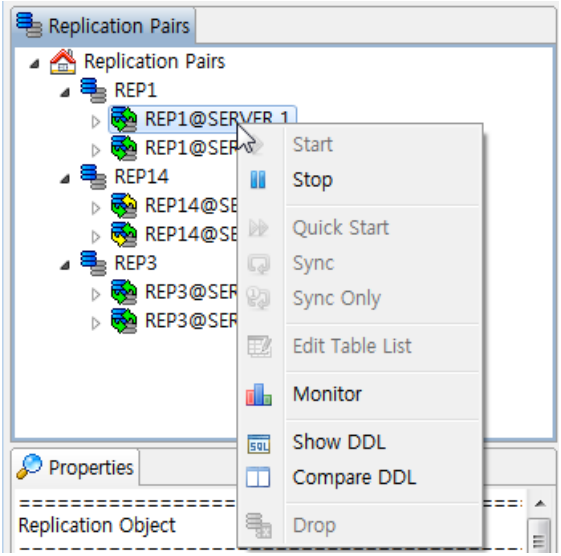
Replication Manager

Replication Pairs

- 이중화 개체를 한 쌍으로 표현하여 보여주는 논리적인 뷰이다. 두 개씩 짝지어 같은 이름을 갖고 서로 상호 작용하는 이중화 객체 그룹을 "이중화 쌍"이라고 부른다.



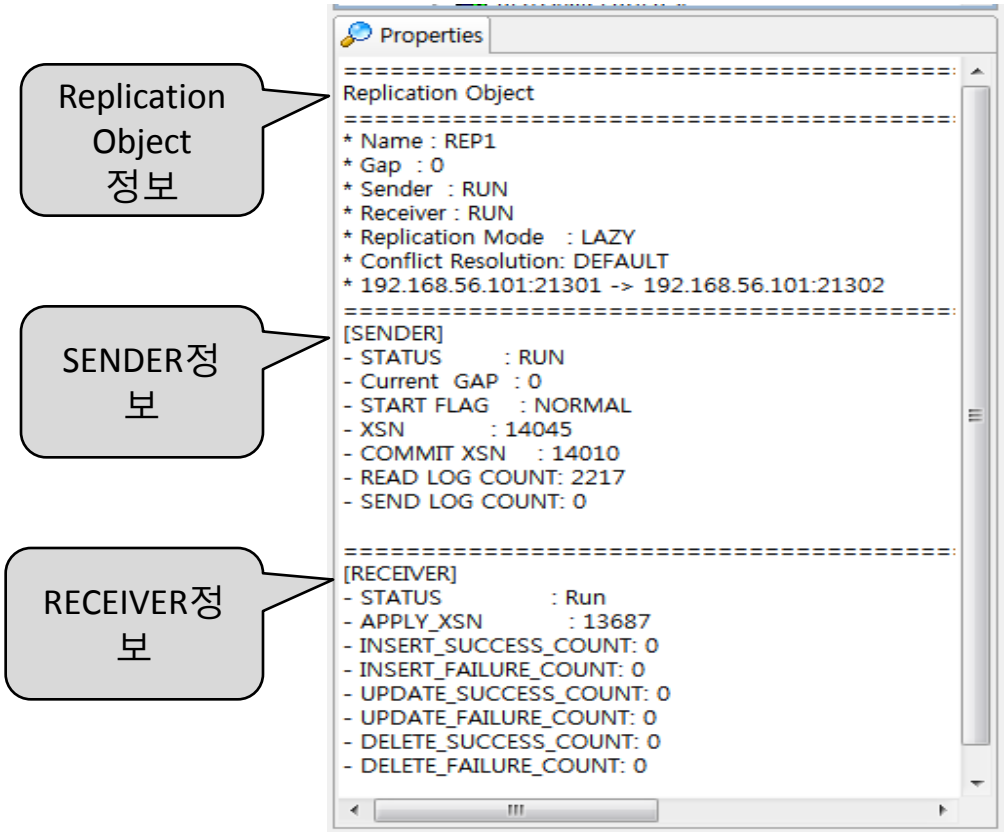




Replication Manager

❖ Properties

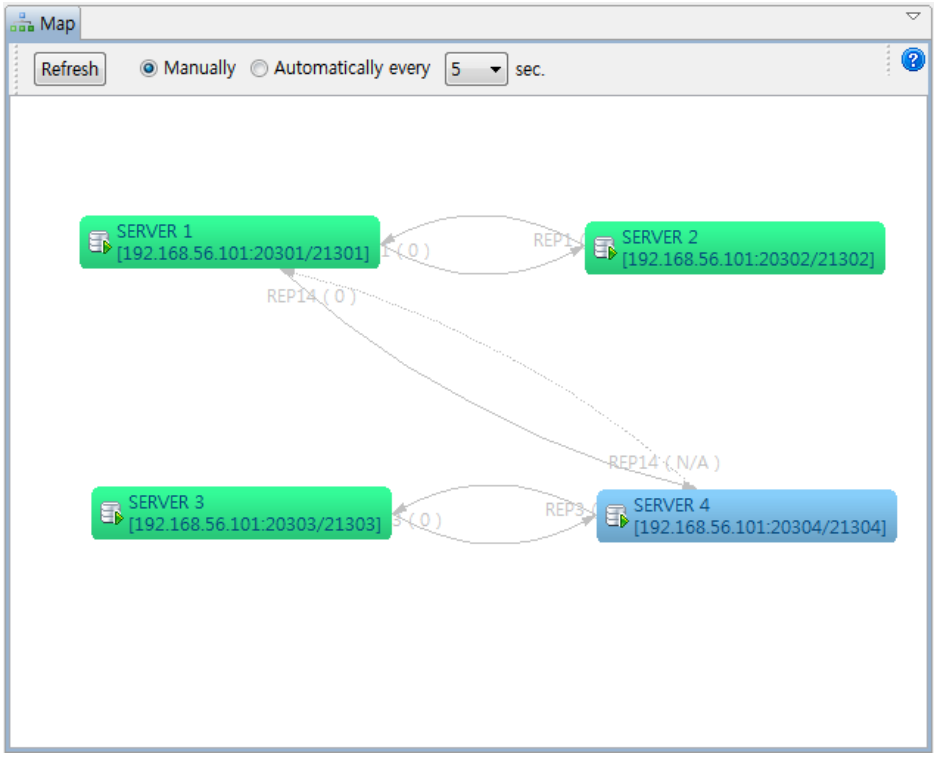
- ➢ 현재 선택된 객체의 속성을 보여준다. (예.데이터베이스 연결 또는 이중화 객체)



Replication Manager

Map

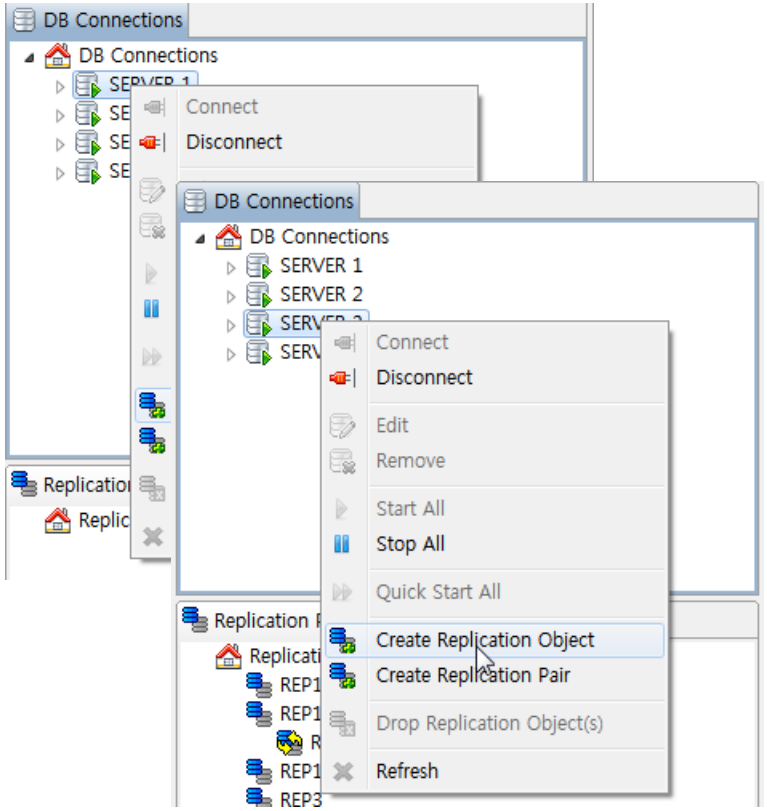
- 데이터베이스들과 이중화 객체들, 그리고 서로 간의 관계에 대한 물리적 구성과 상태를 그래프로 형상화 한다.

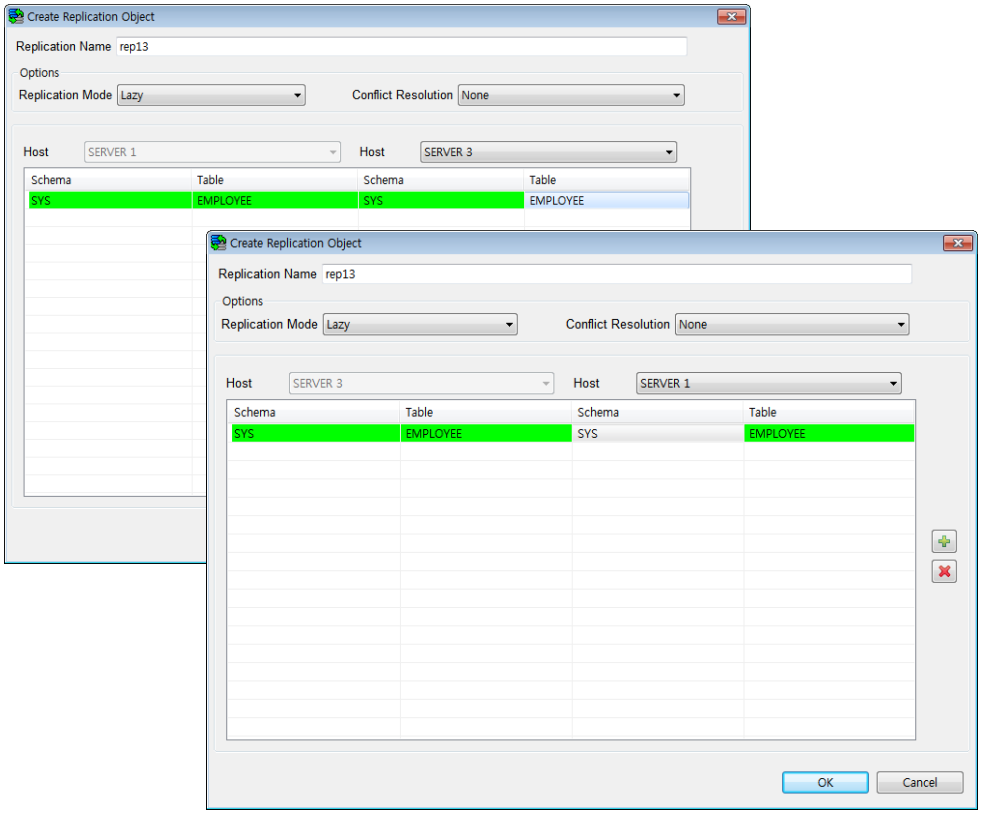


Replication Manager

❖ Replication Object 생성

- ➢ DB Connections 데이터베이스에 오른쪽 클릭한 다음 “Create Replication Object” 메뉴를 클릭한 후 정보를 입력

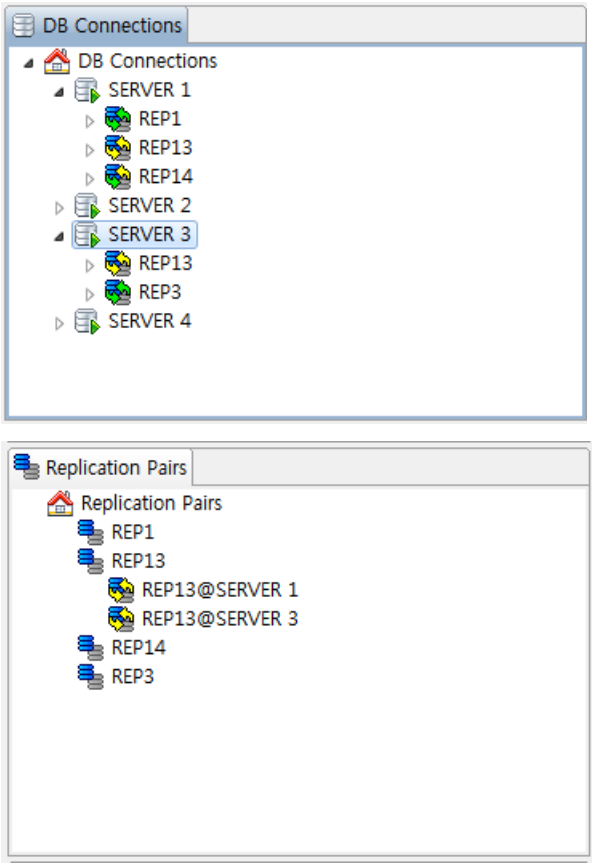


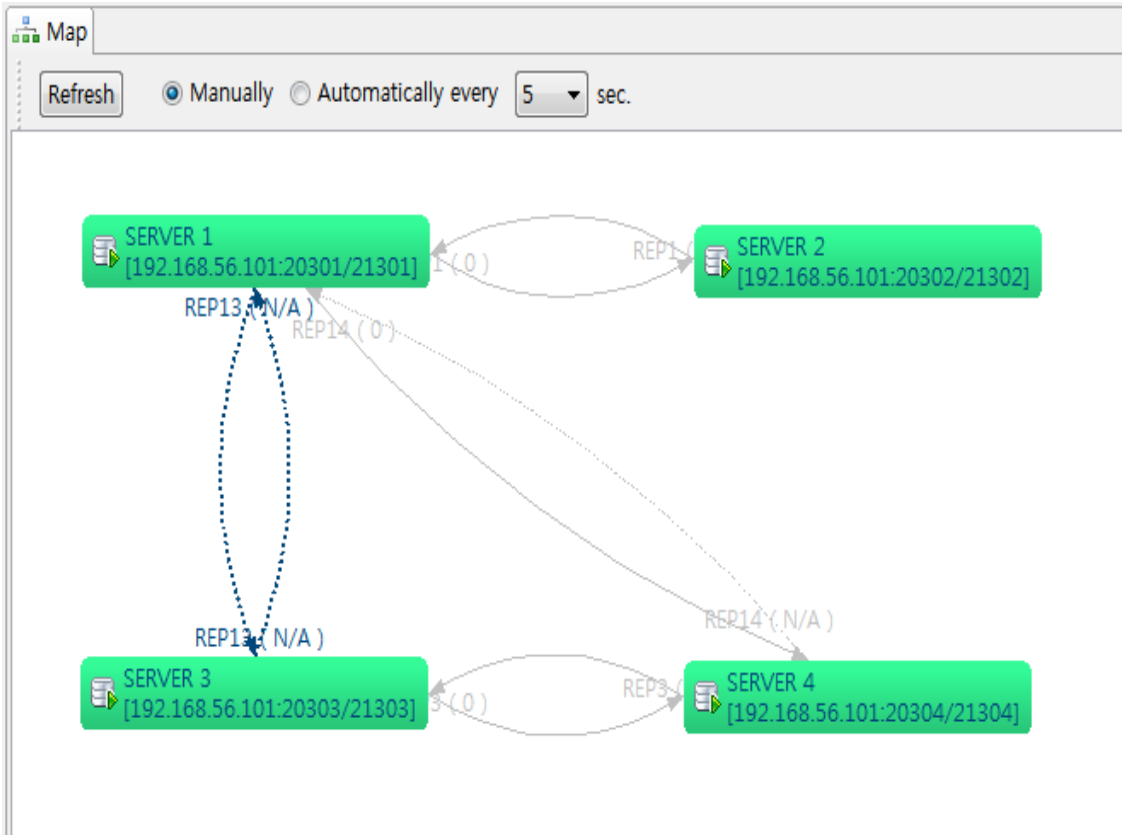


Replication Manager

❖ Replication Object 생성 확인

- ➢ 각 메뉴창에서 새로 생성한 Replication Object 를 확인

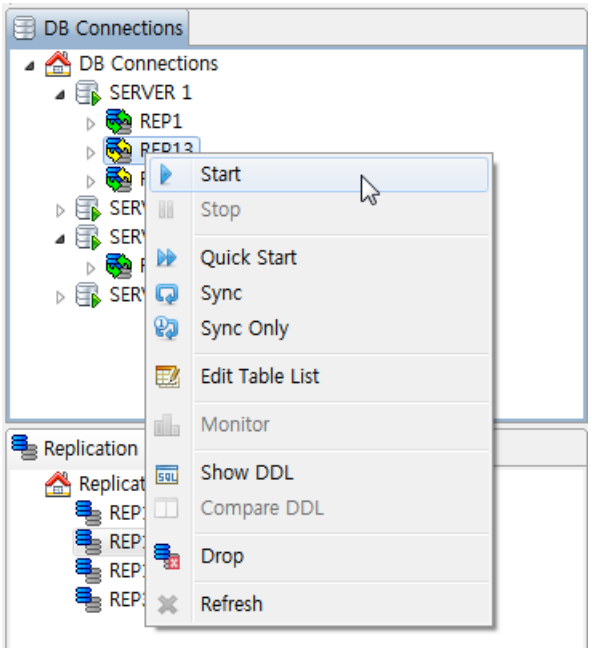


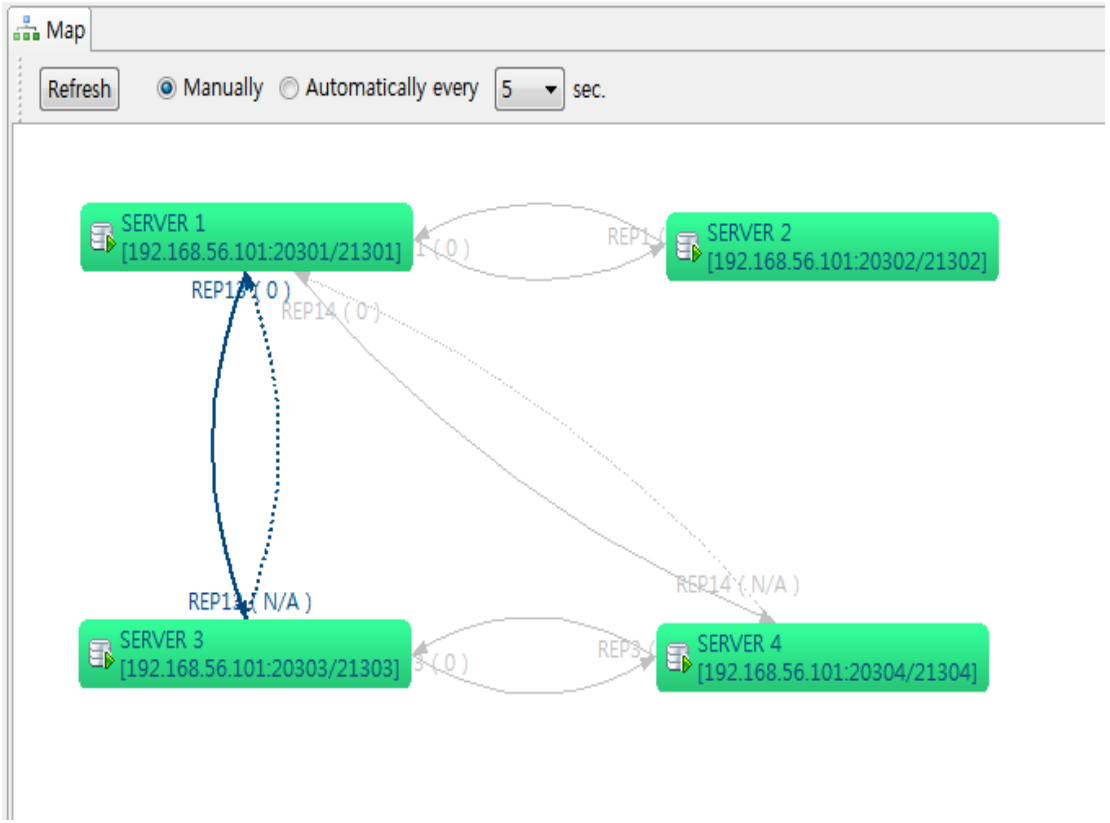


Replication Manager

❖ Replication Object 이중화 시작

- ➢ Replication Object 명에 오른쪽 클릭하여 이중화 시작

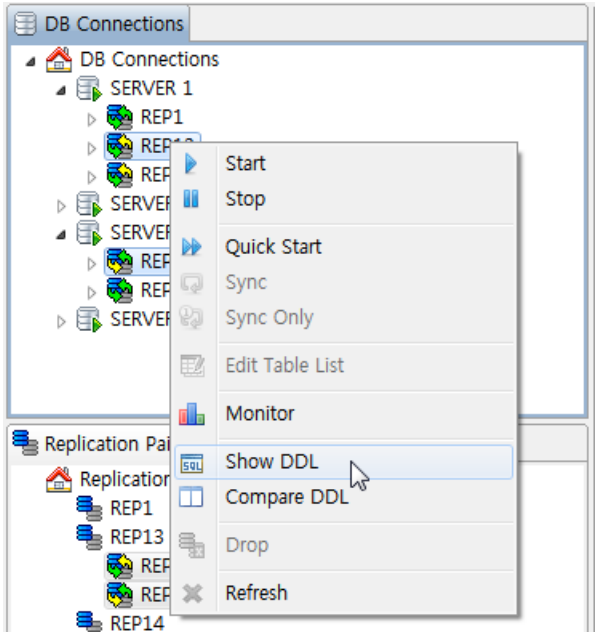


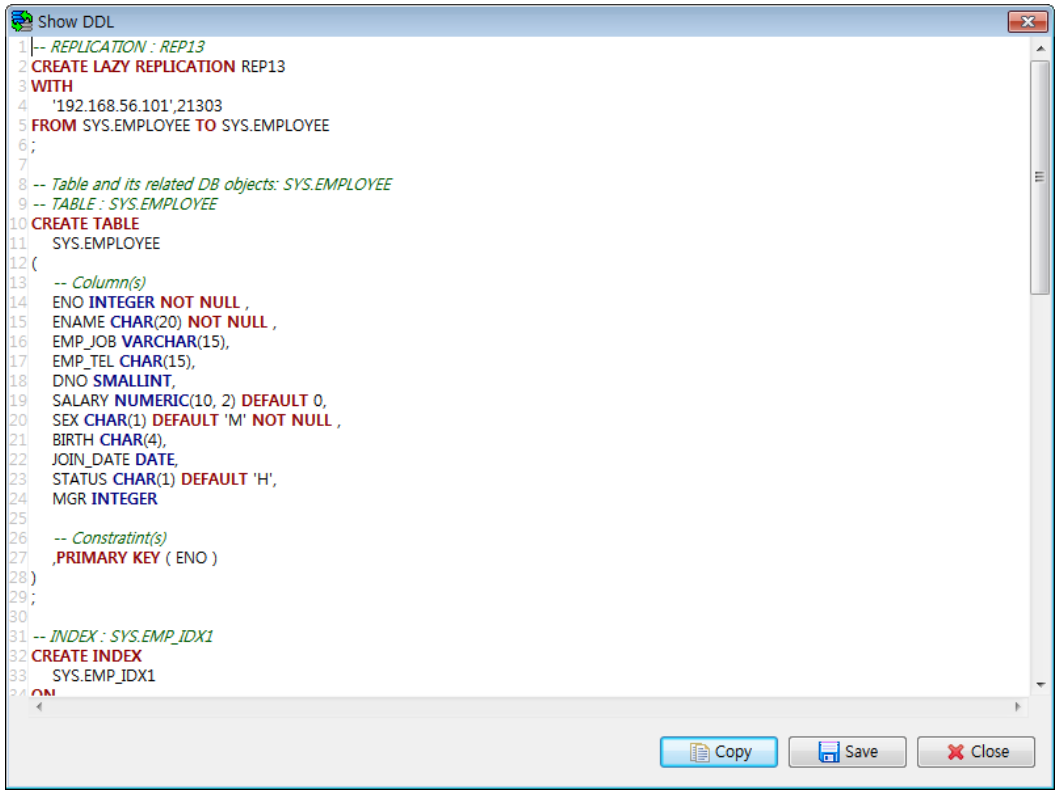


Replication Manager

Show DDL

- Replication Object 와 관련 Object (Table, Index 등)의 스키마 생성 정보를 보여줌

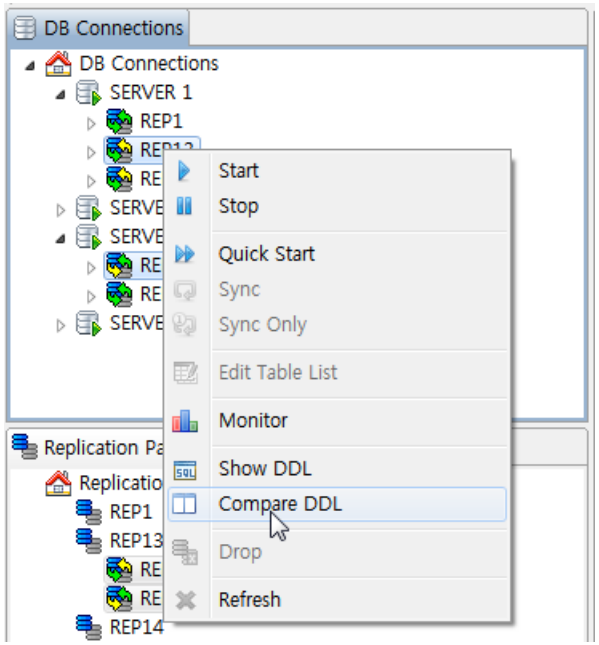


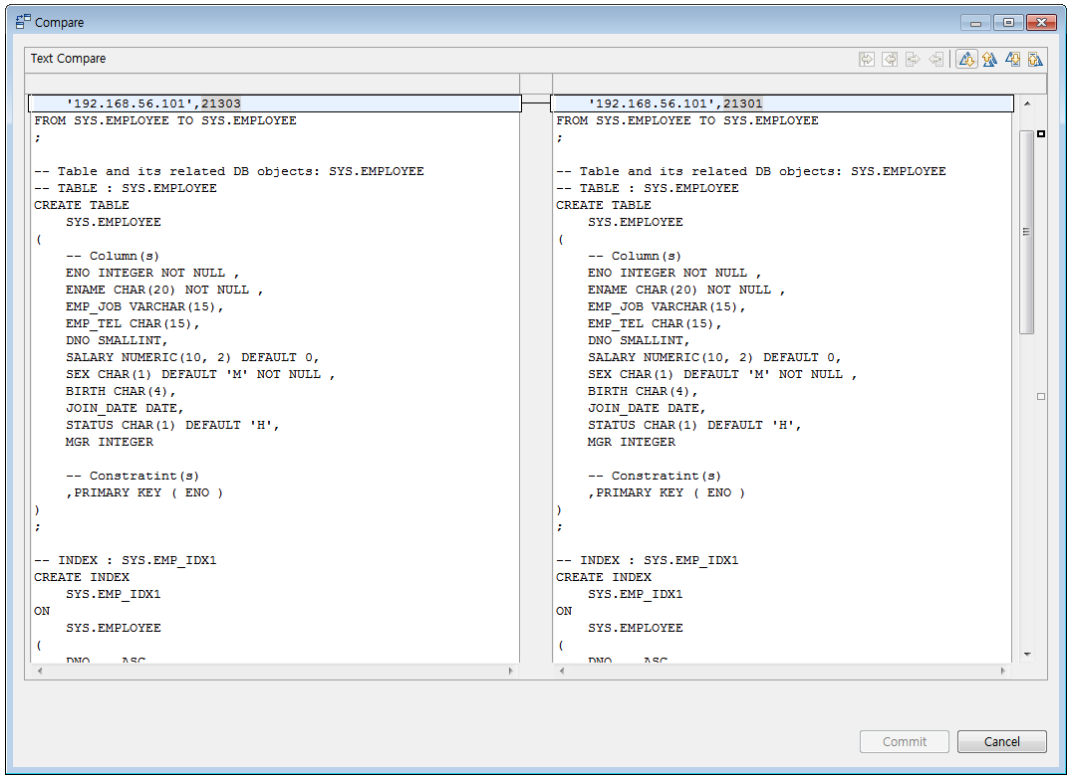


Replication Manager

Compare DDL

- Replication Object 와 상대 서버의 Object 간의 스키마 생성 정보를 비교





Replication Manager

❖ Monitor

- ➢ Replication Object 에 대한 모니터링 정보