Real Alternative DBMS ALTIBASE, Since 1999

ALTIBASE 이중화 제약사항 가이드

2010. 04

ALTIBASE
PERFORMANCE SOLUTIONS

Document Control

Change Record

Date	Author	Change Reference
2010-03	swj0701	Created
2010-04	lim272	Modified

Reviews

Date	Name (Position)
2010-04	wlgml337

Distribution

Name	Location

목차

개요

현재 시장에 알려져 있는 시스템의 고가용성 구성을 위한 방법에는 디스크를 공유하는 방식과 네트웍을 이용한 데이터 동기화 방식이 있다. ALTIBASE는 현재 네트웍을 이용한 데이터 동기화 방법만을 채택하고 있으며 이를 이중화(Replication)라고 부른다.

네트웍을 이용한 데이터 동기화는 디스크를 공유하는 방식에 비해 성능이 빠른 반면 네트웍이 갖는 전송 지연, 단절 등에 의한 제약 사항이 존재한다.

ALTIBASE가 제공하는 이중화 기능 역시 네트웍에 의한 특성 때문에 불가피한 제약 사항들이 존재한다. 본 문서는 이러한 제약 사항들을 고려한 효율적인 이중화 구성이 설계 단계부터 가능하도록 가이드하는 내용을 담고 있다.

사용자는 다음의 문서를 같이 참고하도록 한다.

1. 효율적인 이중화 구성 가이드

ALTIBASE 이중화

본 장에서는 ALTIBASE 이중화에 대한 개념에 대하여 간략하게 설명한다.

ALTIBASE 이중화 개념

ALTIBASE는 고가용성을 위해 구성된 2 개 이상의 시스템에서 각각의 시스템마다 DB 저장 공간을 독립적으로 가져가는 형태로 운영되어야 한다. 이러한 이유는 ALTIBASE 이중화 기법이 서비스의 고가용성을 확보하면서 성능 저하를 최소화하기 위해 상대적으로 성능 저하가 많이 발생하는 디스크 공유 방식보다는 네트웍을 통한 데이터 동기화 방식을 채택하고 있기 때문이다.

ALTIBASE가 제공하는 이중화 기법은 트랜잭션이 발생할 때 기록되는 리두로그를 로컬 서버의 내부 쓰레드인 Sender가 xlog라는 이중화 로그 형태로 치환하여 네트웍을 통해 상대편 서버로 전송하고 수신된 서버의 내부 쓰레드인 Receiver가 이를 수신 서버에 반영하는 형태로 동작한다.

사용자가 고려할 중요한 사항은 네트웍의 특성상 데이터의 전송 지연 및 충돌이라는 문제를 유발할 수 있는 구조적 제약을 가지고 있기 때문에 앞으로 설명할 이중화의 제약 사항에 대한 충분한 이해를 갖고 활용해야지만 개발 단계부터 운영까지 발생할 수 있는 문제 가능성을 최소화 할 수 있다.

데이터 동기화 방식

이중화는 비동기(Lazy) 방식과 동기(Eager) 방식을 제공하고 있다. 비동기 방식은 성능 우선의 데이터 동기화 방식이며, 동기 방식은 데이터 일치성 우선의 데이터 동기화 방식이다. 동기 방식이 비동기 방식에 비해 성능 저하를 감수하고 높은 데이터 일치성을 보장하지만, 두 방식 모두 본질적으로 2 개 이상의 저장소의 데이터 일치성을 완전히 보장할 수는 없다.

2 개 방식의 중요한 차이는 아래와 같다.

구분	설명
동기 방식	로컬 서버에 발생한 트랜잭션이 상대편 서버에 반영될 때까지 대기한다.
비동기 방식	로컬 서버에 발생한 트랜잭션이 상대편 서버에 전송 및 반영에 대한 대기를 하지 않는다.

위의 표와 같이 동기 방식이 높은 데이터 일치성을 보장할 수 있으나 상대편 서버에 반영 여부를 기다려야 하기 때문에 비동기 방식에 비해 매우 낮은 성능을 갖게 된다. 따라서, 사용자는 데이터의 일치성이 업무상 중요한지 여부를 판단하여 동기/비동기 방식에 대한 선택을 해야 한다. (현재 ALTIBASE는 세션 단위의 동기/비동기 방식도 지원하기 때문에 업무별로 적절하게 구성을 하는 것도 고려할 수 있다.)

ALTIBASE 이중화 제약 사항

본 장에서는 ALTIBASE 이중화 기능을 사용할 때의 제약 사항에 대하여 정리하고, 해당 문제에 따른 해결책을 제시하여 데이터 충돌을 발생시키지 않고 효율적으로 이중화 기능을 사용할 수 있도록 가이드 한다.

이중화 구성의 제약 조건

ALTIBASE의 이중화는 1 개 이상의 테이블을 하나의 이중화 객체로 생성하여 운영하는 형태이다. 이렇게 이중화를 생성하여 사용할 때, 다음과 같은 제약 조건이 존재한다.

1. 데이터 제약 조건
   - 이중화 테이블에 대해 반드시 기본키가 존재해야 한다.
   - 이중화 테이블에 대해 기본키에 대한 수정이 없어야 한다.
   - LOB 칼럼은 기본키와 유니크 키로 지정할 수 없다.
   - 양쪽 테이블의 칼럼 정보, 기본키, Not Null 정보가 동일해야 한다. 칼럼의 개수나 정보가 다른 경우에는 이중화 구성은 성공하지만, 실질적인 데이터 동기화는 정보가 동일한 칼럼에 대해서만 이뤄 진다.
   - Memory Table에 대한 이중화인 경우 xLog 크기에 제한이 없다. 그러나, Disk Table에 대한 이중화인 경우에는 한 Row로 생성한 xLog의 크기가 128KB 보다 작아야 한다

2. 연결 제약 조건
   - 하나의 데이터베이스에 이중화 연결이 가능한 최대 개수는 32 개이다.
   - 이중화 연결 대상인 데이터베이스는 캐릭터 셋과 내셔널 캐릭터 셋이 동일하게 설정되어야 이중화 연결이 가능하다(ALTIBASE 5.3.3 버전).

3. 이중화 대상 칼럼의 제약 조건

   이중화 구성 시에 테이블의 칼럼 정보가 동일하지 않은 경우, 이러한 칼럼을 이중화 대상이 아닌 칼럼이라고 하며 다음과 같은 제약 조건을 갖게 된다.
   - 이중화된 트랜잭션에서 INSERT 할 때, 이중화 대상이 아닌 칼럼에는 NULL을 사용한다.
   - 이중화 대상 칼럼과 이중화 대상이 아닌 칼럼으로 Unique 인덱스를 구성하면, 이중화 생성은 성공하지만 진행은 실패한다.

4. 파티션드 테이블 제약 조건
   - 원격 서버와 지역 서버의 파티션 방법이 동일해야 한다.
   - 범위나 리스트 파티션은 파티션 조건이 동일해야 한다.

- 해시 파티션은 파티션의 개수가 동일해야 한다.

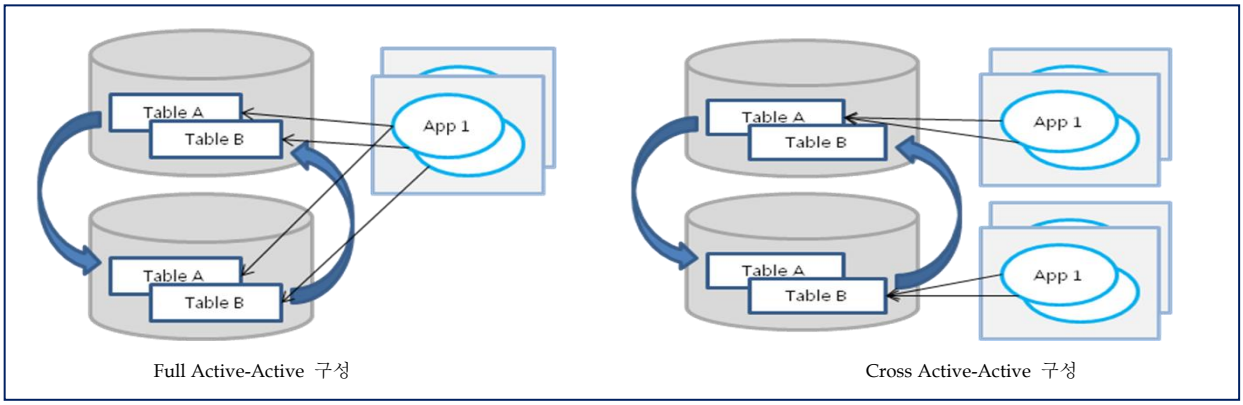
5. 이중화 테이블에 대한 DDL 제약 조건

   기본적으로 이중화 구동 중에는 DDL을 사용할 수 없다. 그러나 다음의 DDL은 Replication_ddl_enable 프로퍼티를 1 로 설정해 주면 이중화 구동 중에도 사용할 수 있다.

   - ADD/DROP Column
   - ALTER Column
   - TRUNCATE Table/Partition
   - CREATE/DROP Index
   - CREATE/DROP Trigger

Cross Active-Active 구성

ALTIBASE 이중화를 사용하여 Active-Active 구성을 하기 위해서는 반드시 Cross Active-Active 방식으로 구성해야 한다. Full Active-Active 방식이 불가한 점은 이중화 Conflict 관련 기술을 참고한다.



Full Active-Active 구성 Cross Active-Active 구성

기본키 칼럼에 UPDATE 불가

ALTIBASE 이중화는 기본키(Primary Key)를 조건으로 데이터 동기화를 지원하기 때문에 기본키 칼럼에 대한 UPDATE는 허용하지 않는다.

만일 이중화 환경에서 기본키 칼럼에 대하여 UPDATE를 수행해야 하는 경우에는 해당 레코드를 DELETE한 후, 다시 INSERT하는 형태의 흐름으로 개발 해야 한다.

SEQUENCE는 이중화 불가

ALTIBASE 이중화는 SEQUENCE에 대한 이중화를 제공하지 않는다. 따라서, 사용자는 SEQUENCE를 서버 별로 홀/짝으로 구성하고 각각을 2 씩 증가하는 형태의 구성 방식을 권장한다. (예) A서버는 홀수로 증가하는 SEQUENCE를 B서버는 짝수로 증가하는 SEQUENCE를 사용하는 형태.

만약 4-Way 이중화로 구성되어 있다면 각 장비 별로 고유ID를 SEQUENCE와 함께 구성하는 방법을 고려할 필요가 있다.

기타 고려 사항

1. 이중화의 성능을 위해 랜카드는 Giga-bit 랜카드를 권장한다.
2. 메모리/디스크 테이블이 함께 하나의 이중화 객체로 구성될 경우 디스크 테이블에 발생하는 트랜잭션의 성능이 상대적으로 메모리에 비해 느리기 때문에 전체적인 이중화 반영 속도가 지연될 수 있다. 따라서, 업무의 흐름상 메모리 및 디스크 테이블의 반영 순서가 중요하지 않은 경우라면 메모리 테이블로 구성된 이중화 객체 및 디스크 테이블로 구성된 이중화 객체로 각각 분리하는 것이 유리하다.

이중화 Conflict

ALTIBASE 이중화는 비동기 방식의 이중화를 기본적으로 채택하고 있기 때문에 논리적으로 회피할 수 없는 conflict 가 발행할 수 있고, 이를 DBMS 단에서 자동적으로 해소할 수 없기 때문에 conflict 가 발생하지 않도록 개발 단계에서 회피하여 설계할 필요가 있다.

이중화 Conflict 종류

이중화 Conflict는 아래와 같이 분류한다.

분류	발생 조건	에러 메시지
INSERT Conflict	수신 측에 해당 PK에 해당하는 레코드가 이미 존재하는 경우	ERR-11058(errno=0) The row already exists in a unique index. (altibase_rp.log)
DELETE Conflict	수신 측에 해당 PK에 해당하는 레코드가 이미 존재하지 않는 경우	ERR-61000(errno=0) The received record is not found in the database. (altibase_rp.log)
UPDATE Conflict	수신 측에 해당 PK에 해당하는 레코드가 존재하지 않는 경우	ERR-61000(errno=0) The received record is not found in the database. (altibase_rp.log)
	수신 측에 해당 PK에 해당하는 레코드의 변경 전 데이터와 수신 받은 변경 전 데이터가 일치 하지 않는 경우	ERR-61035(errno=0) [Receiver] An update conflict encountered. (altibase_rp.log)

- UPDATE Conflict에 대해서는 문서 후반에 자세히 설명한다.

정상적인 INSERT Conflict

예외적으로 정상적인 INSERT Conflict 도 있는데 다음과 같이 코드가 작성된 경우라고 할 수 있다.

```
INSERT INTO TABLE
If (SQLCODE == SQL_DUP_ERROR)
{
    UPDATE TABLE ~
}
```

위와 같은 코드의 흐름은 업무에서 흔히 사용되는 형태이나 ALTIBASE 이중화 환경에서는 $ALTIBASE_HOME /trc/altibase_rp.log에 INSERT Conflict 오류를 생성하게 된다.

로컬 서버에서 실패한 INSERT 트랜잭션이 상대편 서버로 전송되는 이유는 트랜잭션이 PK중복으로 인해 실패하는 과정에서 발생한 롤백 정보가 리두 로그에 기록되고 이 로그가 상대편 서버로 모두 전송이 되기 때문이다. 이런 과정에서 발생한 트랜잭션은

상대편 서버에서도 롤백 되기 때문에 무시해도 되지만 다른 이유에 의한 것일 수 있음으로 해당 SQL문을 유발하는 소스 코드를 살피고 가능한 다음과 같이 변경하는 것이 바람직하다.

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

이중화 UPDATE Conflict

ALTIBASE 이중화는 UPDATE 트랜잭션이 발생하게 되면 UPDATE 대상 칼럼의 이전 값 (Before Image)과 변경 값(After Image)을 모두 전송하여 수신 측의 현재 데이터를 비교하게 되는데, 이 때 비교한 값이 불일치한 경우를 UPDATE Conflict 라고 한다.

ALTIBASE는 UPDATE Conflict가 발생한 경우 기본적인 정책은 수신 측에 반영하지 않고 해당 Conflict오류에 대한 Trace-Log만 기록한다. 단, 사용자의 업무 목적상 강제로 반영해도 문제가 없는 경우에는 ALTIBASE 설정 항목 중 하나인 "REPLICATION_UPDATE_REPLACE"를 "1"로 설정하여 반영시키는 운영 정책도 가능하다.

즉, 해당 속성이 1 인 경우 UPDATE Conflict 상황에서도 Before Image의 값이 달라도 수신 측에서 UPDATE를 수행하며 Trace-Log에 에러 로그를 남기지 않는다.

(Conflict가 발생할 수 있는 예)

	A서버	B서버
현재 상태	C2=10	C2=10
A, B서버 동일PK에 대해 Update 동시 발생	C2=30 (10→30 update)	C2=40 (10→40 update)
트랜잭션 진행 후 상태	Sender: xLog (10 → 30)	Sender: xLog (10 → 40)

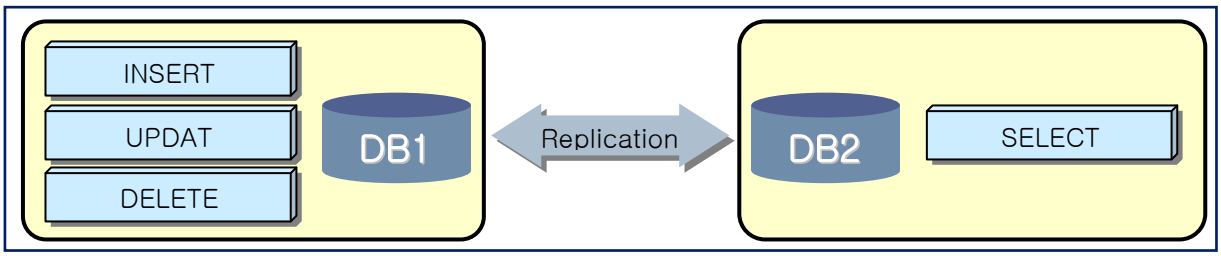
위의 예처럼 A, B서버 모두 동일한 (C2=10)의 상태였으나 동시에 UPDATE가 서로 다른 값으로 변경하는 트랜잭션이 발생하면 최종적으로 A, B서버는 서로 다른 값으로 변경된 상태가 된다. 이 상태에서 A서버는 (10 → 30)으로 변경하는 xLog를 B서버로 전송할 것이고 B서버는 (10 → 40)으로 변경하는 xLog를 A서버로 전송하게 된다. 하지만 현재 값이 이미 Update된 이후 값으로 xLog로 받은 "10"과 다르다고 판단되기 때문에 A, B서버 모두 반영할 수 없게 되고 결과적으로 A, B서버는 서로 다른 값을 갖게 된다.

이중화 Conflict 대안

Conflict를 회피하기 위한 중요한 설계 상의 원칙은 동일한 PK를 갖는 레코드의 접근을 원천적으로 방지하도록 해야 한다는 점이다. 아래와 같이 이중화 구성을 예로 들어 설명할 수 있다.

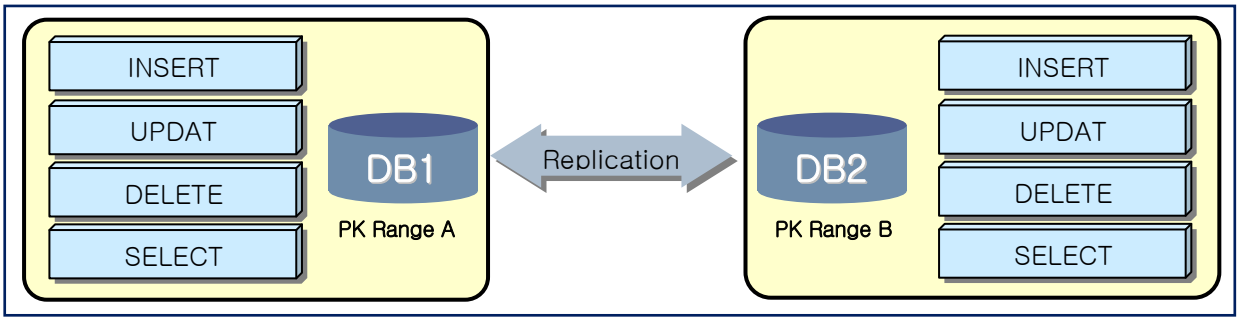
1. 업무의 분리 설계

   아래의 구성처럼 서버 1 에서만 변경을 발생시키고, 서버 2 는 조회용으로만 사용하여 Conflict를 회피할 수 있다



2. 기본키의 분리 설계

   아래의 그림처럼 INSERT, UPDATE, DELETE의 경우 기본키를 조건으로 양쪽 서버의 처리 영역을 분리하여 설계함으로써 Conflict를 회피할 수 있다.



RP_MSGLOG_FLAG 설정

이중화를 사용하면서 데이터의 Conflict가 발생할 경우 MSGLOG_FLAG를 설정하면 $ALTIBASE_HOME/trc/altibase_rp.log 파일에 Conflict가 발생한 테이블 및 SQL문의 내용을 함께 출력해 주기 때문에 문제를 추적하는데 필요한 정보를 확인할 수 있다.

RP_MSGLOG_FLAG 설정은 iSQL에서 다음과 같이 수행하여 설정할 수 있다.

```
iSQL> alter system set RP_MSGLOG_FLAG = 6 ;
```

ALTIBASE®

알티베이스㈜
서울특별시 구로구 구로 3 동 182-13
대륭포스트 2 차 1008 호
02-2082-1000
http://www.altibase.com

대전사무소
대전광역시 서구 둔산동 921
주은리더스텔 901 호
042-489-0330

기술본부
서울특별시 구로구 구로동
우림e-biz센터 11 층 1101 호
02-2082-1000

기술지원센터
02-2082-1114
support@altibase.com

ATC (ALTIBASE Technical Center)
http://atc.altibase.co.kr