Real Alternative DBMS ALTIBASE, Since 1999

ALTIBASE 환경의 개발 시 고려사항 가이드

2010. 07

Document Control

Change Record

Date	Author	Change Reference
2010-07	Lim272	Created

Reviews

Date	Name (Position)

Distribution

Name	Location

목차

개요

본 문서는 ALTIBASE DBMS을 이용한 개발환경에서 개발자가 고려해야 할 사항들을 설명한다. 특정 개발환경 별 참고 문서는 각 개발환경 별로 작성된 기술문서를 참고하도록 한다. 본 문서에는 일반적인 사항을 기술하며 문서 후반에는 ALTIBASE 에러메시지들에 대하여 설명하고 있다. 문서에서 설명하는 ALTIBASE 버전은 5.3 이상을 기준으로 한다.

다음의 문서들을 환경에 맞게 같이 참고하도록 한다.

1. 『디스크 IO병목을 고려한 볼륨구성 가이드』
2. 『ALTIBASE 백업정책 결정을 위한 고려사항』
3. 『효율적인 이중화 구성 가이드』
4. 『ALTIBASE 이중화 제약사항 가이드』
5. 『ALTIBASE 용량 산정 가이드』
6. 『ALTIBASE Precompiler 개발 가이드』
7. 『ALTIBASE SQL Tuning Guide』
8. 『ALTIBASE 자바 개발 가이드』
9. 『ALTIBASE & VC++2008 Express Edition 개발 가이드』
10. 『ORACLE To ALTIBASE 변환 가이드』
11. 『ALTIBASE & WAS 연동 가이드』

설계 단계의 고려사항

ALTIBASE를 이용하여 DB 또는 시스템 설계를 수행하는 경우 고려할 사항들을 설명한다. 설계 단계에서 고려하지 못한 사항들은 이후 개발뿐 아니라 운영 단계에까지 미치는 영향이 크기 때문에 반드시 아래 사항들을 고려하여 설계에 참고할 것을 권장한다.

이중화 환경의 고려사항

① ALTIBASE는 네트웍 기반의 이중화 방식

ALTIBASE는 네트웍을 기반으로 변경트랜잭션 로그를 송/수신하여 데이터를 동기화 하는 형태의 이중화 방식을 채택하고 있다. 즉, 일반적으로 사용자가 흔히 이해하는 디스크를 공유하는 방식이 아니다. 따라서, 네트웍 환경에서 발생할 수 밖에 없는 데이터 전송지연 등으로 인한 동기화 부분을 고려하여 설계를 해야 한다.

ALTIBASE의 이중화 방식은 Lazy방식과 Eager방식이 있다. Lazy방식은 데이터의 전송지연을 허용하는 형태이며 Eager방식은 데이터의 지연을 허용하지 않는 형태이다. 이와 같은 Lazy/Eager의 설정은 세션단위로도 설정이 가능하다. 따라서, 데이터의 전송지연이 허용되는 수준의 업무라면 Lazy방식을 채택하고 그렇지 않은 데이터의 동기화가 반드시 필요한 업무라면 Eager방식을 업무 별로 적절하게 설정하여 운용하도록 한다.

비교	Lazy 방식	Eager 방식
성능	단독 서버의 90% 성능 수준	Lazy 방식 대비 현저한 성능저하
동기화	상대편 서버에 변경트랜잭션 로그의 전송 성공만을 확인하는 구조이기 때문에 성능은 우수한 반면 동기화는 지연될 수 있음	상대편 서버에 데이터 불일치를 원천 방지하면서 반영완료가 된 후 로컬에도 Commit되기 때문에 성능은 지연되지만 데이터의 동기화 부분은 지연되지 않음.

② 데이터 불일치의 방지 방안 고려

동일한 Primary key를 갖는 레코드를 양쪽 서버에서 동시에 서로 다른 값으로 변경하는 경우 이중화 방식에서는 서로 다른 레코드로 변경되는 상황이 발생하게 된다. 이 경우 사용자가 원치 않는 데이터의 불일치가 발생하기 때문에 업무요건을 충분히 파악하여 변경 업무를 이중화로 구성된 그룹 내에서 한쪽 서버에서만 발생하도록 프로그램을 배치하거나 동일한 PK를 갖는 레코드가 이중화로 구성된 그룹 내에서 변경이 동시에 발생하지 않도록 배치를 해야만 한다.

예를 들면, L4 switch를 사용한다면 업무별로 프로그램을 각기 다른 서버에 배치하여 192.168.1.X망에서 접속한 프로그램은 1 번 DBMS서버로 접속을 하고 192.168.2.X망에서 접속한 프로그램은 2 번 DBMS서버로 접속을 수행하는 형태의 구성이 가능할 것이다. 또는, 사용자가 명시적으로 서비스 프로그램의 DBMS접속 부분에 IP를 지정하는 것도 방법이 된다.

③ 로그 기반의 이중화 방식

ALTIBASE의 이중화는 로컬서버에서 발생한 트랜잭션의 변경로그를 전송하는 형태이다. 한 개의 SQL문으로 백만 건을 변경하였다 하더라도 동일하게 원격서버로 한 개의 SQL문의 전송되는 형태가 아닌 백만 건의 변경트랜잭션 로그가 모두 전송되는 구조임을 의미한다.

이외도 이중화 환경의 주요 고려사항들은 ALTIBASE의 기술문서인『효율적인 이중화 구성 가이드』, 『이중화 제약사항 가이드』문서를 반드시 참고하도록 한다.

HA(High Availability) 및 백업에 대한 고려

일반적인 상황에서는 장애가 발생하고 복구가 되면 이중화는 자동으로 자신이 가지고 있는 장애시점에 미처 보내지 못한 트랜잭션로그를 상대편에 보내어 데이터 동기화를 수행하게 된다. 하지만, 네트웍의 특성상 장애 시점에 변경트랜잭션 로그의 전송불가에 의한 데이터 불일치로 인해 서비스를 바로 Fail-Over할 수 없는 업무가 있는지를 고려해야 한다.

반드시 데이터가 동기화 된 이후에만 서비스가 가능한 업무라면 ALTIBASE는 데이터 미지연분에 대한 반영을 위해 Off-Line replicator를 제공하고 있다. 다만, 이것은 장애 서버의 디스크에 접속이 가능한 상태에서만 사용이 가능하다는 제약조건이 있다. (이 기능을 쓰고자 한다면 공유디스크 장비를 구축하는 것을 권장한다.)

이러한 Off-Line replicator를 사용할 수 있는 환경이 아니라면 프로그램에서 장애시점의 Raw-Data를 누적하여 필요한 부분만큼 정상서버에 반영한 후 서비스를 이관하는 형태의 HA정책을 수립해야 한다. 예를 들면, 증권사의 경우 대외계를 통해 나가는 모든 데이터는 고유번호로 관리된다. 이 번호는 대외계 시스템과 DB에 동일하게 존재하기 때문에 장애 발생시 대외계와 DB시스템 간에 차이가 발생한 부분만큼을 정상 서버에 반영하는 업무로직을 사용하는 경우도 있다.

HA외에도 물리적인 디스크 장애로 인한 데이터파일의 손상 및 트랜잭션 로그파일의 손상으로 인한 장애에 대비하여 백업정책을 수립해야 한다. ALTIBASE의 백업 방식은 아카이브 백업 방식이며 증분 백업방식을 제공하고 있지 않다. 따라서, 사용자는 백업 시점에 모든 파일을 백업하거나 또는, 테이블스페이스 별로 순차적으로 백업을 하든 DB 전체를 백업해야 하는 방법을 선택해야 한다.

백업방식은 다음과 같은 형태 중 하나를 택해야 한다.

방식	설명
iloader	주요 테이블 별로 레코드의 스냅샷만을 저장하는 형태로 iloader로 백업을 받는 시점의 데이터만 백업된다.
아카이브 백업방식	트랜잭션 로그파일을 포함하여 백업하기 때문에 파일의 손상이 없는 시점까지 복구가 가능하다.
콜드 백업방식	ALTIBASE 프로세스를 내린 후 모든 필요 파일을 복사하여 보관하는 형태로 파일을 복사한 시점까지만 복구가 가능하다.

백업과 관련하여 『ALTIBASE 백업정책 결정을 위한 고려사항』문서를 참고하도록 한다.

테이블 설계의 고려사항

① 테이블의 저장 위치를 목적에 맞게 선정

ALTIBASE는 메모리/디스크 테이블스페이스를 모두 지원한다. 사용자는 빠른 처리성능이 요구되는 업무에 대해서는 어떤 테이블들을 메모리 테이블스페이스 위치 시킬 것인지를 선정해야 한다. 즉, 메모리/디스크 테이블스페이스에 각각 위치할 업무 목적 별 테이블의 목록을 작성할 필요가 있다.

다만, 하나의 테이블 명으로 메모리/디스크에 동시에 위치할 수 없기 때문에 예를 들어, 당일 처리를 메모리 테이블에 저장하고 이전 처리를 디스크에 저장하고 한다면 테이블을 2 개로 분리하여 EXEC_TABLE_MEM / EXEC_TABLE_DISK 와 같은 형태로 테이블을 생성해야 한다. 조회 시에는 2 개의 테이블을 UNION ALL 이용하여 조회하거나 별도의 View를 생성하여 조회하는 방법을 택하면 된다.

② 컬럼 타입에 대한 고려

ALTIBASE가 제공하는 숫자 타입의 경우 Numeric (9)와 같은 타입은 Integer타입으로도 충분히 표현이 가능한데 차지하는 공간은 8byte를 차지한다. Integer타입에 비해 레코드당 4 byte를 더 점유하게 된다. 처리 성능 측면에서도 내부적으로 Numeric의 경우 다시 Native타입으로 변환을 해야 하는 비용이 발생하기 때문에 Integer타입에 비해서는 다소 느릴 수 밖에 없다.

다음의 사항들을 컬럼 타입의 선정에 있어 고려할 것을 권장한다.

고려사항	설명
숫자형	1. Native ←→ Non-Native의 변환비용 최소화 되도록 고려 (Native타입[Integer, Double, BigInt]이 성능 면에서는 더 유리함) 2. 실수 형의 정밀도를 요하지 않는 경우 Double 타입의 선택 3. Sum, Avg가 사용되는 컬럼이라면 Double, BigInt 타입의 선택
날짜형	날짜 관련 연산이 중요한 경우라면 Date 타입을 선택하고 검색 및 비교 연산에 국한된 경우라면 Char/Varchar타입을 선택 (Date 타입은 8byte로 고정되어 있음)
Join	Join이 발생하는 컬럼이라면 형 변환이 발생하지 않도록 고려

③ 이중화 대상인 경우 반드시 Primary key를 가져야 한다.

④ Foreign Key는 트랜잭션의 성능을 저하시킴으로 무분별하게 사용하지 않는다.

파티션 테이블의 제한

ALTIBASE 파티션 테이블의 경우 현재까지 (버전 5.3.7 기준) Partition Global Index를 지원하고 있지 않다. 따라서, 파티션 테이블 전체를 조회하는 조건 절에 파티션 키가 배제된 형태의 조회 시에는 현저한 성능저하가 발생하기 때문에 파티션 테이블은 업무

목적상 히스토리성 데이터를 월별, 목적별 보관 형태의 목적으로 사용할 것을 권장한다.

하드웨어 준비의 고려사항

위에서 언급한 사항들과 관련된 물리적인 고려사항을 정리하여 설명한다.

고려사항	설명
이중화	네트웍을 이용하기 때문에 이중화의 연결은 별도의 Giga-bit 랜카드를 이용하여 서버간에 직접 연결하거나 Private망을 이용한 환경을 권장한다.
디스크	백업을 고려할 경우 예측된 DB용량에 추가적으로 테이블스페이스 전체와 함께 일부 아카이브 로그파일이 저장될 수 있는 디스크 공간이 필요하다.

추가적으로 ALTIBASE의 리두로그 파일에 대한 정책은 무한 생성이라고 볼 수 있다. 몇 개의 리두로그 파일을 생성하고 재사용하는 구조가 아니라 계속 새로운 리두로그 파일을 생성해 가는 정책을 쓰기 때문에 대량 변경작업등으로 인한 리두로그 파일의 대량 생성이 불가피할 경우들이 있기 때문에 리두로그 파일이 저장될 디스크의 공간은 충분히 책정하는 것이 바람직하다.

하드웨어적인 용량산정과 관련된 사항은 『ALTIBASE 용량산정 가이드』문서를 참고하도록 한다.

Not Support Feature

ALTIBASE는 SQL92 표준을 준수하지만 타 DBMS와 비교하여 약간 상이한 부분들이 존재한다. 이와 관련된 자세한 사항은 각 타 DBMS벤더 별로 제공되고 있는 기술문서를 참고하도록 한다. (Ex)『ORACLE To ALTIBASE 변환 가이드』

개발 단계의 고려사항

개발단계에서 필요한 고려사항들을 정리하여 설명한다.

DBMS 접속방식의 선택

① ALTIBASE 접속 방식의 선택

접속방식	설명
CONNTYPE=1	TCP/IP 방식의 접속, 일반적인 접속 방식
CONNTYPE=2	UNIX Domain Socket 방식의 접속, 로컬서버 내에서만 사용가능
CONNTYPE=3	IPC 방식의 접속, 로컬서버 내에서만 사용가능

CONNTYPE=n 의 의미는 접속방식을 의미하며 프로그램 내에서 접속 문자열을 지정할 경우 사용자가 명시적으로 지정할 수 있다. DBMS와 응용 프로그램이 별도의 분리된 서버에 위치하면 CONNTYPE=1 의 방법만 사용이 가능하다. 하지만, DBMS와 응용 프로그램을 같은 서버에서 구동할 수 있는 경우라면 통신비용의 감소를 위해서 CONNTYPE=2, 3 을 선택하는 것을 권장한다.

② ALTIBASE는 Auto-Commit 모드가 기본설정

JDBC는 표준 스펙상 Auto-Commit모드로 접속하게 된다. 그 외 응용프로그램은 사용자가 별도로 변경하지 않는 이상 기본적으로 $ALTIBASE_HOME/conf/altibase.properties 파일 내에 정의된 “AUTO_COMMIT” 속성에 의해 결정된다. 이 속성값이 “1” 이면 Auto-Commit 모드로 동작한다. (Auto-Commit이라 함은 변경트랜잭션 정상적으로 완료된 후 자동으로 Commit이 수행됨을 의미)

따라서, 사용자가 이 속성값을 Non Auto-Commit으로 변경하고자 한다면 설정파일에서 해당 속성값을 “0”으로 변경한 후 ALTIBASE를 재 구동 해야 한다. 만일, 세션 별로 제어를 할 경우에는 DBMS에 접속한 이후 다음과 같은 SQL을 수행하면 된다.

```
ALTER SESSION SET AUTOCOMMIT = FALSE;
```

(Java의 경우는 setAutoCommit함수를 이용하여 제어하도록 한다.)

(ALTER SYSTEM 명령을 위해 시스템 전체에 반영하도록 실시간으로 변경할 수 있으나 재 구동 후에 다시 원복 될 것임으로 전체에 영향을 미치는 속성의 변경은 프로퍼티 파일을 통해 수정하고 재 구동하는 방법을 권장한다.)

Thread 프로그램, Connection Pool의 관리

Thread 환경의 프로그램에서는 하나의 연결을 다수의 Thread가 공유할 경우 반드시 연결에 대한 동시성 제어를 개발자가 수행해야 한다. ALTIBASE는 세션과 서버간의 통신과정에서 약속된 프로토콜을 주고 받는다. 일반적으로 PREPARE→BIND→EXECUTE→FETCH등의 과정을 거치게 되는데 이 과정이 순차적으로 진행되어야 할 상황에서 다른 프로토콜이 인터럽트가 발생하여 세션처리 과정이 잘못 처리되는 과정에서 오류가 발생하게 되는 것이다.

```
Communication link failure (EXEC->INVL)
Communication link failure (PREP->EXEC)
Invalid request to process the SQL statement
```

위의 오류들은 Thread 프로그램 또는 Connection Pool을 사용하는 과정에서 하나의 연결이 어떠한 SQL을 수행 중에 동시성 제어가 안 된 상태에서 또 다른 SQL을 진행시킬 경우 발생한다. 따라서, 위와 같은 오류가 발생하면 먼저 개발자가 작성한 프로그램에서 연결 부분에 대한 동시성 제어나 처리과정 중에 잘못된 인터럽트가 발생하지 않는지 확인이 필요하다.

(간혹, 사용자 프로그램에서 Timer를 설정한 경우 위와 같은 동시성 제어 문제가 아니어도 발생할 수 있다. 예를 들면, SELECT~FETCH 과정 중에 Timer가 동작되어 아직 FETCH프로토콜이 완료되지 않은 상태에서 다른 SQL문이 수행될 경우 위와 같은 오류가 발생할 수 있다.)

Application의 연결해제 시 확인사항

프로그램에서 수행한 SQL문들은 개별 프로그램과 서버에 prepared된 상태로 존재한다. 이러한 SQL문들은 사용이 완료되면 자원을 해제하도록 코드를 개발하는 것이 바람직하다. 특히, 자바의 경우 메모리 증가 현상이 발생할 수 있으므로 다음과 같은 코드가 반드시 중요하다.

```
Connection con = pool.getConnection();
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery();

try
{
    ….
} exception ()
{
    …..
}
finally
{
    // 개발자가 프로그램 내에서 사용한 모든 자원을 명시적으로 해제하도록 한다.
    rs.close();
```

```
        stmt.close();
        conn.close();
}
```

CLI/ODBC등의 개발의 경우도 마찬가지로 SQLAllocStmt로 메모리가 할당된 이후 사용이 끝난 시점에는 반드시 SQLFreeStmt 함수를 통해 SQL_DROP을 수행해야 한다. 재사용이 된다면 SQL_CLOSE 옵션을 이용하도록 하며 그렇지 않고 완전히 해제할 경우라면 SQL_DROP 옵션을 통해 명시적으로 해제하도록 한다.

Session Timeout

세션은 DB에 접속한 프로그램의 연결을 의미한다. 각 세션은 ALTIBASE 고유의 Timeout정책에 의해 관리된다. ALTIBASE는 크게 5가지 Timeout정책을 제공하고 있으며 다음과 같다. (아래 색상이 표기된 것은 연결 문자열에서만 제어가 가능하다.)

구분	에러 후 세션상태	설명
CONNECTION_TIMEOUT	단절	DB에 접속된 상태에서 네트웍상의 송/수신간에 블로킹 된 상태의 타임아웃을 설정하는 옵션
TIMEOUT	단절	DB에 접속을 시도하는 시간이 이 설정 값을 초과하면 발생
QUERY_TIMEOUT	유지	DB에 접속된 상태에서 질의를 수행하는 전체 시간이 이 설정 값을 초과하면 발생
FETCH_TIMEOUT	단절	DB에 접속된 상태에서 SELECT문을 수행한 이후 FETCH 프로토콜이 발생하는 시간 간격이 이 설정 값을 초과하면 발생
IDLE_TIMEOUT	단절	DB에 접속된 상태에서 어떠한 질의도 수행하지 않고 대기하는 시간이 이 설정 값을 초과하면 발생
UTRANS_TIMEOUT	단절	DB에 접속된 상태에서 질의를 수행한 이후 Commit/Rollback 이 수행될 때까지 이 설정 값을 초과하면 발생 (수행 질의는 롤백처리 됨)

만일, 사용자가 연결시도에 대한 5초 시간제한과 네트웍상의 블로킹현상이 발생할 경우 이를 30초안에 응답이 없으면 연결을 해제하겠다고 설정한다면 다음과 같이 연결 문자열에 옵션을 추가하면 된다.

```
DSN=127.0.0.1;CONNTYPE=2;TIMEOUT=5;CONNECTION_TIMEOUT=30
```

그 외의 TIMEOUT설정은 기본적으로 $ALTIBASE_HOME/conf/altibase.properties에서 설정한 단위로 동작하게 되며 세션 별로는 다음과 같은 질의를 수행하여 제어가 가능하다. (“0” 으로 설정할 경우 무한대로 동작하게 됨)

```
ALTER SESSION SET QUERY_TIMEOUT = 30 (초 단위)
ALTER SESSION SET FETCH_TIMEOUT = 30
```

위의 TIMEOUT에 의한 오류가 발생하면 다음과 같은 에러를 응용 프로그램에서 확인할 수 있다.

구분	에러 메시지
CONNECTION_TIMEOUT	Connection time out
TIMEOUT	Client unable to establish connection
QUERY_TIMEOUT	Client's query exceeded in the execution time limitation
FETCH_TIMEOUT	Communication link failure. Server closed the connection.
IDLE_TIMEOUT	Communication link failure. Server closed the connection.
UTRANS_TIMEOUT	Communication link failure. Server closed the connection.

TIMEOUT 관련 에러는 $ALTIBASE_HOME/trc/altibase_boot.log에도 기록이 남게 된다.

```
[2010/07/06 13:10:35] [Thread-182894171744] [Level-1]
[Notify : UTrans Timeout] Session Closed by Server : Session ID = 53
CLIENT_INFO          => TCP 127.0.0.1:3992(PID : 13645)
Time Limit           => 3
Running Time         => 5
Last Query           => insert into t1 select * from t1 limit 1
Caused by Transaction  => 82368
```

위의 로그가 altibase_boot.log에 기록이 되면 개발자는 반드시 왜 Timeout으로 인한 오류가 발생했는지 확인하여 조치해야 한다. 각각의 원인은 앞에서 설명한 바와 같기 때문에 질의처리의 병목으로 발생했다면 해당 질의를 튜닝 해야 할 것이며 위의 예제처럼 UTrans_Timeout이 발생한다면 변경트랜잭션이 발생한 후 commit/rollback을 수행하지 않은 것임으로 접속정보를 통해 해당 프로그램의 트랜잭션 제어부분을 반드시 수정/조치해야 한다.

사용자가 위의 조치로 해결하기 어렵고 업무적으로 처리를 강제로 해야 한다면 DB전체의 속성 값을 변경하기 보다는 해당 프로그램의 DB세션에 대해 해당 TIMEOUT 설정 값을 변경하여 처리하도록 한다.

SELECT Cursor 사용 시 주의

CURSOR를 이용한 SELECT문을 구현할 때에는 반드시 FETCH 구문에 대해 에러체크를 수행하고 에러가 발생하거나 혹은 데이터를 모두 읽은 후에는 반드시 CURSOR를 CLOSE하도록 해야 한다.

만일, 이미 다 읽은 CURSOR를 FETCH하려고 하거나 아직 닫히지 않는 CURSOR에 어떤 작업을 시도할 경우 의도하지 않은 오류가 발생할 수 있으므로 CURSOR를 사용한 뒤에는 반드시 정상적으로 CLOSE하도록 해야 한다.

또 하나의 주의사항은 ALTIBASE의 경우 FETCH도중에 COMMIT/ROLLBACK이 발생하게 되면 자동적으로 열린 CURSOR를 닫게 된다. 따라서, FETCH를 수행하면서 다른 트랜잭션을 발생하고자 할 경우에는 별도의 DB세션을 생성 한 후 해당 세션을 통해 별도의 트랜잭션을 처리하도록 하거나 FETCH가 모두 완료된 이후 COMMIT/ROLLBACK을 수행하도록 해야 한다. 하지만, 처리해야 하는 레코드 건수가 많다면 대량 변경으로 수행하기 보다는 별도의 세션을 맺어 매건 처리하는 구조로 구현하는 것을 권장한다. 변경될 레코드의 개수가 하나의 트랜잭션으로 반드시 묶여야 한다면 조건을 주어 나누어서 처리가 되도록 프로그램을 개발하는 것이 바람직하다.

(아래의 대량 변경 작업의 주의 사항을 확인하도록 한다.)

SQL Error의 체크

ALTIBASE환경의 개발자는 모든 질의 처리 이후 반드시 SQL Error를 체크할 것을 권장한다. 특히, 다음의 경우를 반드시 체크하도록 해야 명확한 오류를 알아낼 수 있다.

```
EXEC SQL PREPARE ….
EXEC SQL EXECUTE ….
If (SQLCODE != SQL_SUCCESS)
    Error_log ();


또는,
EXEC SQL DECLARE CURSOR…
EXEC SQL OPEN ….
If (SQLCODE != SQL_SUCCESS)
    Error_log ();
```

위 코드에서는 PREPARE 또는 DECLARE CURSOR수행 후 에러 체크하는 부분이 없다. 이 경우 PREPARE (DECLARE CURSOR)단계가 오류였을 경우라면 EXECUTE (OPEN)단계에서 SQL오류는 "Not defined (XX)" 와 같은 오류가 발생하여 실제로 PREPARE (DECLARE CURSOR) 오류로 인함임에도 불구하고 해당 오류를 정확하게 확인할 수 없다.

ALTIBASE는 PREPARE (DECLARE CURSOR) 수행과정도 서버로 질의를 전송하여 Execute전 단계까지 질의처리 과정을 해놓기 때문에 PREPARE (DECLARE CURSOR)를 수행한 이후에도 반드시 에러를 체크하는 로직을 넣어야 한다.

중요한 점은 모든 SQL처리 문이 들어간 경우는 반드시 SQL에러체크를 수행하도록 해야 한다는 점을 알아두어야 한다.

LOB 타입의 주의

ALTIBASE에서 LOB타입을 조회할 때에는 반드시 Non-Auto-Commit모드로 동작해야 한다. Auto-Commit모드에서는 다음과 같은 오류가 발생할 수 있다.

```
[ERR-91101 : Connection is in autocommit mode. One can not operate on LOB datas with
```

autocommit mode on.]

테이블의 컬럼에 LOB를 빼고 조회를 할 경우는 상관이 없으나 SELECT 타겟절에 LOB 타입이 포함될 경우 위와 같은 오류가 발생한다. 따라서, LOB 타입을 조회할 때에는 반드시 세션의 속성을 Non-Auto-Commit모드로 변경을 한 후 수행하도록 해야 한다.

Prepared Statement의 사용

ALTIBASE는 질의처리의 성능을 빠르게 하기 위해 내부에 수행된 질의의 실행계획을 저장하고 있다. 하지만, 매번 PREPARE/EXECUTE를 하는 형태의 질의를 수행하게 되면 내부적으로 이미 PREPARE된 실행계획을 사용하게 되지만 PREPARE를 수행하는 네트웍 비용이 매번 발생하기 때문에 이러한 성능 감소 요인을 막기 위해서는 다음과 같이 코드를 작성할 것을 권장한다.

(O)	(X)

Prepare statement
while (1)
{
Execute statement
}
	while (1)
	{
	Prepare statement
	Execute statement
	}

오픈 소스 중에는 OTL (Oracle, Odbc Template Library)라는 DB개발 라이브러리가 있다. 이 라이브러리는 모든 질의를 매번 Prepare/execute를 수행하는 형태로 처리하고 있다. 단, streaming옵션을 활성화 시키면 한번 prepare된 질의는 execute만 수행하는 형태로 처리하게 되는데 이런 튜닝요소를 통해 (네트웍 전송비용의 감소) 응용 프로그램의 처리속도를 2 배 이상 향상 시키는 경우도 있으므로 개발자는 질의 처리를 수행하는 과정에서는 가능한 dynamic SQL을 최소화 시키고 동일한 질의를 매번 prepare하고 있지는 않은지 확인하도록 해야 한다.

다음과 같은 코드는 매우 불합리한 코드라고 할 수 있다.

```
sprintf (sql,  "INSERT INTO t1 VALUES (%d, %d)", szNum1, szNum2);
Exec sql Prepare s for :sql;
Execute s;
```

위의 코드는 매번 실행할 SQL문장을 만들고 prepare한 후 execute하도록 실행한다. 이런 경우는 다음과 같이 한다면 간결하고 합리적이라 할 것이다.

```
Exec sql INSERT INTO t1 VALUES (:szNum1, :szNum2);
```

SQL Tuning

DB업무와 관련해 개발 작업을 진행하다 보면 질의 처리가 느린 경우들이 발생한다. 혹은, 테스트 환경에서는 빠르게 처리되었는데 통합테스트 또는 운영 단계에서 질의가 매우 느린 경우들이 발생한다. 이것은 개발자 혹은 DBA가 수행되는 모든 질의에 대해 실행계획의 검증이나 테스트 과정을 소홀히 한 경우라고 볼 수 있다.

ALTIBASE는 질의의 처리과정 즉, 실행계획을 다음과 같이 확인할 수 있다.

```
ALTER SESSION SET EXPLAIN PLAN = ON;
```

위와 같은 명령을 수행한 후 iSQL이나 또는 AdminCenter2 와 같은 유틸리티에서 질의를 수행하면 수행된 실행계획을 확인할 수 있다.

```
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
Alter success.
iSQL> SELECT a, COUNT(*) FROM t1 WHERE a = 1 GROUP BY a;
A                COUNT
-----------------------------------
1                262145
1 row selected.
----------------------------------------------------------
PROJECT ( COLUMN_COUNT: 2, TUPLE_SIZE: 16 )
  GROUP-AGGREGATION ( ITEM_SIZE: 32, GROUP_COUNT: 1, BUCKET_COUNT: 1024, ACCESS: 1, SELF_ID: 3, REF_ID: 2 )
    SCAN ( TABLE: T1, FULL SCAN, ACCESS: 262145, SELF_ID: 2 )
----------------------------------------------------------
```

위의 실행계획은 T1 테이블을 FULL SCAN하고 레코드에 대해 262,145 번 접근했음을 확인할 수 있다. 만일, 인덱스가 존재하고 인덱스를 타도록 조건을 명시한다면 다음과 같이 출력될 수도 있을 것이다.

```
----------------------------------------------------------
PROJECT ( COLUMN_COUNT: 1, TUPLE_SIZE: 4 )
  SCAN ( TABLE: T1, INDEX: IDX_T1, ACCESS: 2, SELF_ID: 2 )
----------------------------------------------------------
```

SQL튜닝과 관련된 사항은 『ALTIBASE SQL 튜닝 가이드』문서를 참고하면 더 많은 정보를 얻을 수 있으므로 개발 전에 개발자들은 반드시 읽어볼 것을 권장한다.

대량 변경 작업의 주의

일반적인 대량 변경작업은 ALTIBASE뿐 아니라 타 DBMS에서도 부하가 크기 때문에 각 벤더 별로 요구하는 주의사항들이 존재한다. ALTIBASE는 다음과 같은 사항의 주의가 필요하다.

① **메모리 테이블의 대량 변경작업 시 메모리 증가의 가능성**

ALTIBASE는 MVCC(Multi Version Concurrency Control)기법을 지원한다. MVCC기법은 변경이 발생할 경우 동일 레코드를 해당 테이블의 빈 공간에 복제하여 변경을 가하게 된다. 만일, 천 만 건을 변경하면 천 만 건의 복제 본으로 인한 메모리 증가가 발생할 수 밖에 없다. (물론 이렇게 늘어난 영역은 다시 데이터 또는 복제영역으로 재활용이 되지만 실시간으로 메모리를 해제 시키지는 않는다.)

② **대량 변경작업을 수행함으로써 트랜잭션 로그파일의 지속적인 증가**

대량의 변경작업은 각 변경 대상 레코드에 대한 리두로그를 트랜잭션 로그파일에 기록해야 한다. 따라서, 하나의 트랜잭션 내에 변경되는 모든 레코드의 리두로그를 기록해야 하는데 만일, 이 대상의 범위가 많다면 그만큼의 트랜잭션 로그파일도 많이 생성될 수 밖에 없다. (ALTIBASE는 체크포인트 시점에 트랜잭션 로그파일을 삭제하게 되는데 로그파일에 트랜잭션이 지속중인 정보가 존재하면 삭제할 수 없고 이로 인해 로그파일 증가에 의한 디스크 부족의 장애가 발생할 가능성이 있다.)

③ 이중화 환경에서의 Dead-Lock 발생 가능성 및 반영 지연

이중화 환경에서는 앞서 설명한 바와 같이 하나의 트랜잭션이 변경한 모든 레코드의 변경정보를 상대편 서버로 전송하게 된다. 이 과정에서 상대편 서버의 해당하는 레코드에 대하여 락을 유지하게 되는데 만일, 상대편 서버에서 동일한 범위에 속하는 레코드를 변경할 경우 이중화 로그에 의해 발생한 트랜잭션과 상대편 서버 자체에서 발생하는 트랜잭션 간에 데드락이 유발될 수 있다.

ALTIBASE Trace 로그

ALTIBASE는 설치된 경로인 $ALTIBASE_HOME/trc 경로에 모듈 별로 중요한 상태정보 및 에러 등에 대해 Trace 로그를 기록하고 있다. 이 에러들에 설명한다.

Trace 로그의 분류

ALTIBASE가 기록하는 Trace로그는 다음과 같다.

분류	설명
altibase_boot.log	Start/Stop 과정 및 DBMS 전반의 로그를 기록
altibase_sm.log	체크포인트 및 테이블스페이스 관련 로그를 기록
altibase_qp.log	DDL수행 및 Query Processing과 관련된 로그를 기록
altibase_rp.log	이중화와 관련된 로그를 기록
altibase_lk.log	DB link 사용과 관련된 로그를 기록

해당 로그파일은 기본적으로 10 개까지 altibase_xx.log를 포함하여 altibase_xx.log-[1-10]까지 순환되면서 로그를 기록하게 된다.

이하에서는 사용자에게 의미 있는 중요한 메시지들에 대한 각 Trace로그 별로 분리하여 설명한다.

altibase_boot.log

사용자는 DBMS에서 오류가 발생하면 altibase_boot.log를 먼저 확인해야 한다. 표 중에 [ ] 표시는 사용자가 반드시 조치할 에러이다.

ERR-0108d (errno=0) License is invalid or expired.
구동단계에서 라이센스 파일이 없거나 잘못된 경우 발생한다. ALTIBASE에서 라이센스 정보를 확인하고 필요 시 재발급을 받도록 해야 한다.
ERR-01052(errno=13) Unable to invoke open() function on [/dbs/mydb-0-0]
최초에 구동 전에 DB를 생성하지 않았거나 해당 파일에 접근할 권한이 없는 경우 발생한다. DB를 생성하거나 또는 디렉토리 및 파일에 접근권한이 있는지 확인한다. 또는, 사용자의 실수로 데이터파일을 삭제한 경우일 수 있으므로 이 경우 백업 본을 이용하여 복구하는 절차가 필요하게 된다.
ERR-01052(errno=2) Unable to invoke open() function on [/dblog01/logfile536358]
ALTIBASE 구동단계에서는 자동복구를 수행하게 되는데 이 과정에 필요한 리두로그파일이 손상되었거나 존재하지 않는 경우에 발생한다. ALTIBASE 본사로 기술지원 요청을 해야 한다.
ERR-0001c(errno=76) Unable to shutdown the communication channel

ALTIBASE Shutdown 단계에서 세션과 관련된 정리과정에서 나오는 Warning메시지로 운영에는 영향이 없다.
ERR-71019 (errno=104) Failed to invoke a system function, write() (read, accept, select)
디스크 또는 네트웍상의 오류로 인해 발생한다. 파일시스템의 체크나 네트웍상의 문제가 없는지 확인이 필요하다. 또는, 메모리의 부족으로 시스템 콜이 실패하는 경우가 발생할 수 있다.
ERR-40029 (errno=16) Failed to invoke a system function, flock_trywrlock()
이미 ALTIBASE가 구동 중이거나 구동된 상태에서 다시 ALTIBASE를 재 구동하려고 하는 경우 발생한다. 또는, 메모리의 부족으로 시스템 콜이 실패하는 경우 발생할 수 있다.
ERR-10018 (errno=2) The version of data file for backup is not compatible with the version of storage manager. Backup DB => [Version ID = 5.4.1, Bit = 64, Endian = LITTLE LogSize = 10485760 Transaction Table Size = 1024] Server=>[Version ID = 5.4.1, Bit = 64, Endian = LITTLE LogSize = 10485760 Transaction Table Size = 2049]
백업 받은 데이터파일을 가지고 구동하는 과정에서 현재 사용중인 ALTIBASE 버전과 호환이 되지 않는 데이터파일을 가지고 복구하려고 시도하거나 백업시점에 설정한 ALTIBASE 설정 값과 다른 상태로 구동을 하려고 시도하는 경우에 발생한다.
[ERROR] first write () operation failed. errno=32
세션이 단절된 상태에서 read/write 시스템 콜이 오류가 발생할 수밖에 없으므로 그 정보를 Warning형태로 기록을 남길 경우이다. 응용 프로그램에서 정상적인 Disconnect 명령 없이 강제 종료시키는 경우에도 이 로그가 기록된다. 운영에는 영향이 없다.
ERR-7101d(errno=11) Protocol header error. (TCP 211.115.75.212:63847) Protocol processing failed. Close connection...
ALTIBASE의 접속포트를 통해 접속한 세션이 ALTIBASE 서버버전과 호환되지 않는 버전으로 접속을 시도한 경우로 출력된 접속정보를 통해 역추적하여 Server/Client버전간의 호환여부를 확인해야 한다. 운영에는 영향이 없다.
ERR-4000f(errno=2) No Error Message Loaded
$ALTIBASE_HOME/msg/ 경로에 위치한 파일들이 없거나 버전과 호환되지 않는 경우 발생한다. 또는, 해당 경로에 읽기 권한이 없을 경우도 발생한다.
[Notify : Query Timeout] Query Canceled by Server : Session ID = 102373 [Notify : Fetch Timeout] Session Closed by Server : Session ID = 80106 [Notify : Utrans Timeout] Session Closed by Server : Session ID = 80106 [Notify : Idle Timeout] Session Closed by Server : Session ID = 80106
QUERY / FETCH / UTRANS / IDLE TIMEOUT 설정에 의해 TIMEOUT이 발생한 경우 그에 대한 Warning형태의 로그를 기록하는 경우이다. 문제가 되는 경우 해당 세션정보가 메시지와 함께 기록되기 때문에 세션을 추적하여 문제라고 판단할 경우 해결하도록 해야 한다.
ERR-01027(errno=0) No more IPC channel (MAX=50, USED=50, BUFSIZE=65536)
IPC 접속 가능한 자원보다 더 많은 IPC접속을 시도할 경우 발생한다.
[Warning] Memory allocation failed. Size:524320 Timeout: 0

ERR-61055(errno=12) Memory allocation failure
메모리가 부족한 상황에서 내부적으로 필요한 메모리 할당에 실패하는 경우 발생한다. 이 경우 메모리가 부족한 원인을 파악하여 조치해야 한다. 만일, Swap 메모리까지 부족한 경우가 발생하면 Warning없이 ALTIBASE가 비정상 종료할 수 있다.
ioctl() failed
소켓 또는 디스크 I/O작업간에 시스템 콜 수행에 일시적인 문제가 발생한 경우 발생한다. 일반적으로 무시해도 상관없으나 네트웍 또는 디스크를 점검하는 것을 권장한다.
WANING : write error number : 0, file name : /dblog01/logfile537572, file handle : 440, intended size : 10485760, writed size : 10485760 또는, File Extending Failed. : The disk space has been exhausted. Error : Extending DataFile : from 0, size 10485760
디스크 용량이 부족하여 ALTIBASE가 필요로 하는 파일을 생성할 때 용량이 부족하여 정상적으로 생성하지 못했다는 경고로 반드시 디스크 용량을 체크하고 조치해야 한다.
BEGIN-STACK-[CRASH]========== END-STACK========== [========= FATAL Terminated ==========]
ALTIBASE가 버그로 인해 비정상 종료가 발생할 경우 어떤 진행과정에서 발생했는지 자체적인 Trace로그를 출력하게 된다. 이 에러가 발생한 경우는 ALTIBASE 프로세스가 종료된 상태임으로 우선 $ALTIBASE_HOME/trc/모든 파일을 백업하도록 한다. 다음 디스크가 충분하다면 리두로그파일을 확보한다. 이와 같은 파일백업이 완료되면 ALTIBASE를 재 구동하도록 조치하고 ALTIBASE에 기술지원을 요청한다.

altibase_qp.log

altibase_qp.log에는 사용자가 수행한 DDL의 수행기록 및 DB 속성값을 “ALTER SYSTEM” 구문으로 변경을 가하는 경우 등에 대한 로그가 기록된다. 아래와 같이 분류된다.

[EXEC_DDL_BEGIN : create index T1_IDX on T1 (A)]
사용자가 수행한 DDL의 수행시작을 의미하는 로그이다.
[EXEC_DDL_END : SUCCESS]
사용자가 수행한 DDL의 정상완료를 의미하는 로그이다.
[EXEC_DDL_END : FAILURE] errorcode 1627549735.
사용자가 수행한 DDL의 오류내역을 의미하는 로그이다
[SET-PROP] CHECKPOINT_BULK_WRITE_PAGE_COUNT=[0]
사용자가 시스템 전역에 영향을 미치는 DB속성 설정 값을 변경한 경우를 의미하는 로그이다.

altibase_sm.log

altibase_sm.log는 체크포인트의 수행여부 및 장시간 수행되는 질의의 존재 여부 등을 판단해볼 수 있는 중요한 근거가 된다.

[CHECKPOINT-BEGIN]
체크포인트가 시작되었음을 의미한다.
Remove Online Log File at LFG [0]: File[220 ~ 227]
체크포인트가 수행되면서 더 이상 복구에 필요하지 않은 트랜잭션 로그파일들을 정리하는 메시지이다. 만일, File [None] 이라고 지속적으로 표시될 경우 장시간 실행되는 질의가 없는지 또는, 이중화로 미 전송되고 있는 부분이 없는지 확인이 필요하다.
[CHECKPOINT-summary] BeginChkptLSN=[0,83,9820280], EndChkptLSN=[0,83,9820721], DiskRecLSN=[0,83,9820280]
체크포인트가 정상적으로 완료되는 시점에 체크포인트의 수행정보에 대한 요약정보를 로그에 남긴다.
[CHECKPOINT-END]
체크포인트가 정상적으로 완료 되었음을 의미한다.
Minimum LSN = [0,83,9824136]
위 값은 현재 진행중인 트랜잭션의 기록중인 로그파일의 offset위치이다. 따라서, 저 값이 변동하지 않고 계속 같은 값을 유지한다면 위의 Remove로그와 동일하게 장시간 수행되는 질의의 존재 여부나 이중화 미 전송여부를 반드시 확인해야 한다.
Database-Level Backup Completed [SUCCESS]
백업이 정상 수행되었음을 의미한다.
DISK TABLESPACE .... DATABASE ....
디스크 테이블스페이스의 백업이 수행되었음을 의미한다.
~/loganchor0 BACKUP TO
로그앵커의 백업이 수행되었음을 의미한다.
MEMORY TABLESPACE DATAFILE …. BACKUP TO ….
메모리 테이블스페이스의 백업이 수행되었음을 의미한다.

altibase_rp.log

altibase_rp.log에는 이중화의 동작상태 및 각종 Conflict발생 시의 질의가 기록되기 때문에 이중화 문제의 추적에 있어 중요한 단서제공을 하고 있다.

ERR-31017(errno=0) Replication not found
이중화 관련 DDL을 수행할 때 지정한 이중화 객체가 존재하지 않는 경우에 발생한다.

따라서, 지정한 이중화 객체가 존재하는지 확인이 필요하다.
ERR-6200f(errno=16) [Sender] Stop sender thread REP1 at [9765213], Restart SN[9765207]
이중화 Sender를 사용자가 명시적으로 Stop시킨 경우 발생하는 로그이다.
ERR-71017(errno=79) Failed to invoke a system function, connect() ERR-61012(errno=79) [Sender] Failed to connect to the peer server ERR-61022(errno=79) [Sender] Sender Sleep : 60 seconds [Sender] getNextLastUsedHostNo: from 192.168.1.81:53341 to 192.168.1.81:53341 ERR-61003(errno=9) Unable to read from a socket ERR-61012(errno=119) [Sender] Failed to connect to the peer server ERR-6100d(errno=119) [Sender] Failed to handshake with the peer server (Handshake Process Error)
이중화 Sender가 상대편 서버에 접속할 수 없는 경우와 재 접속을 시도하는 로그이다. 상대편의 서버, 네트웍 상태 및 ALTIBASE가 정상 구동되고 있는지 확인이 필요하다. 만일, 명시적으로 상대편 이중화를 Drop한 경우라면 정상적인 에러이다.
[Recovery Sender] Replication REP1 Start... at [60495917]
이중화가 정상 개시되어 Sender가 구동된 로그이다.
[Receiver] Replication REP1 Started ...
이중화가 정상 개시되어 Receiver가 구동된 로그이다.
RECEIVER:REPLICATION STOP MSG arrived! [ReceiverApply] REPLICATION STOP XLog arrived! Normal Stop! [Receiver] Replication REP1 Stopped ...
상대편 서버의 이중화가 사용자에 의해 명시적으로 Stop되어 중지된 경우이다.
ERR-61047(errno=0) [Receiver] REP1 receiver has error in recvXlog() ERR-61048(errno=0) [Receiver] REP1 receiver has recvXLog error in run() ERR-6104b(errno=0) [Receiver] REP1 receiver is ended (by thr_exit)
이중화 Receiver가 어떤 오류로 인해 쓰레드가 종료된 경우이다.
ERR-61036(errno=0) [Receiver] err_not found in deleteXlog() ERR-61000(errno=0) The received record is not found in the database.
이중화로 수신 받은 DELETE 트랜잭션을 수행하는 과정에서 해당하는 레코드가 존재하지 않을 경우 Conflict 오류를 기록한 경우이다.
ERR-61035(errno=0) [Receiver] An update conflict encountered. ERR-61001(errno=0) A conflict has been occurred while executing the received statement.
이중화로 수신 받은 UPDATE 트랜잭션을 수행하는 과정에서 해당하는 레코드의 Before/After 값이 달라 Conflict 오류를 기록한 경우이다.
ERR-6103a(errno=0) [Receiver] err_not_found in updateXlog()

ERR-61000(errno=0) The received record is not found in the database.
이중화로 수신 받은 UPDATE 트랜잭션을 수행하는 과정에서 해당하는 레코드가 존재하지 않을 경우 Conflict 오류를 기록한 경우이다.
ERR-11058(errno=0) The row already exists in a unique index.
이중화로 수신 받은 INSERT 트랜잭션을 수행하는 과정에서 해당하는 레코드가 이미 수신 측에 존재하는 경우이다. (동일한 PK를 가지는 데이터가 존재함)

위 예시에서 사용된 이중화 객체 명은 "REP1"이다. 사용자가 지정한 이중화 객체는 이름이 다를 것이므로 에러에 대한 조회 시에 이를 고려하도록 해야 하며 주로 이중화 Sender/Receiver의 동작 상태 및 Conflict 오류를 감지할 것을 권장한다.

또한, 이중화 Conflict가 발생하면 앞서 설명한 이중화 환경의 고려사항을 제대로 준수하지 않았거나 이중화 지연으로 데이터 불일치가 발생했을 가능성이 있기 때문에 어떠한 이유로 그러한지 Conflict 발생 시 남는 SQL문을 통해 원인을 찾는 것이 중요하다.

Client Application 에러 메시지

ALTIBASE를 사용하는 응용 프로그램을 운영하는 환경에서 자주 발생하는 에러 유형들을 소개하고 각 유형들에 대해 어떤 사항들을 개발자가 확인해야 하는지를 설명한다.

Connection does not exist

DBMS에 연결이 되지 않은 상태 또는, 단절된 상태에서 질의를 처리하려고 할 경우 발생하는 오류이다. 다음의 경우들에서 발생할 수 있다.

1. 앞에서 설명한 TIMEOUT정책에 의해 연결이 해제된 이후 질의를 수행할 때
2. 쓰레드 프로그램의 경우 존재하지 않는 연결 고유 명을 사용한 경우

위의 사항들을 점검하고 관련 사항에 적합한 조치를 취하도록 한다. 예를 들어, TIMEOUT정책에 의해 발생하였다면 해당 세션의 TIMEOUT 설정 값을 늘리거나 또는 질의의 튜닝을 진행 하는 식의 조치가 필요하다.

Communication link failure

일반적으로 위의 "Connection does not exist" 는 이 오류 다음으로 발생하게 된다. 이 에러는 질의를 수행과정에서 세션이 단절된 상태를 의미한다. 원인은 앞서 설명한 것처럼 TIMEOUT정책에 의해 해제되는 경우가 일반적이다. 간혹, 네트웍의 문제로 연결이 단절되는 경우가 발생할 수 있다.

이 문제는 통신소켓상의 문제로 다양한 원인으로 발생할 수 있기 때문에 다음의 사항들을 체크하고 확인할 것을 권장한다.

1. altibase_boot.log 에 세션이 단절된 로그가 있는지 확인
2. 쓰레드 프로그램인 경우 지켜야 할 주의사항이 제대로 준수되지 않았는지 확인
3. 네트웍상의 일시적인 문제가 없었는지 확인

Calculate stack overflow

ALTIBASE 내부적으로 질의를 처리하는 과정에서 사용되는 객체로 수행되는 질의가 필요로 하는 Stack이 부족한 경우 발생한다. 사용자는 해당 질의의 수행 전에 다음과 같이 질의를 수행한 후 에러가 발생한 질의를 수행하면 정상적으로 처리할 수 있다.

```
ALTER SESSION SET STACK SIZE = 8192;
```

이 값은 메모리의 증가를 유발하기 때문에 전체 시스템에 반영하여 사용하기 보다는 해당 세션만 변경하여 사용하도록 할 것을 권장한다.

Conversion not applicable

다음과 같은 경우에 발생한다.

```
CREATE TABLE t2 (a BYTE (2));
INSERT INTO t2 VALUES   (CLOB'1');
```

위의 예제처럼 BYTE컬럼에 CLOB의 데이터를 넣을 경우 형 변환을 CLOB을 BYTE로 변환할 수 없음으로 에러를 발생시킨다. 즉, 각 컬럼 타입끼리는 호환이 가능한 매트릭스가 있는데 그러한 호환 가능하지 않는 질의를 처리하게 될 경우 이 에러가 발생한다.

컬럼 타입 간의 호환은 ALTIBASE ODBC User 매뉴얼을 참고하면 된다.

Fixed record size exceed a page size

ALTIBASE에서는 한 페이지에 저장될 수 있는 Fixed 컬럼 크기의 최대는 8K로 제약이 되어 있기 때문에 발생한다. (버전 5.1.5 이하 버전)

최신의 5.3 버전부터는 디스크의 경우 이 제약사항이 해결 되었으며 메모리 테이블의 경우도 32K로 제약이 완화되어 있다. 따라서, 5.1.5 버전을 사용하는 경우 이 에러가 발생하면 FIXED컬럼 크기의 합계가 8K를 넘지 않도록 고려해야 한다.

```
CREATE TABLE t1 (c1 CHAR(4096));
CREATE TABLE t2 (c1 CHAR (4096));
iSQL> CREATE VIEW view1 AS SELECT a.c1 AS aa, b.c1 AS bb FROM t1 a, t2 b;
[ERR-31233: Fixed record size exceeds a page size.]
```

위의 예처럼 t1, t2 테이블은 각각 8K제약에는 해당되지 않지만 VIEW를 만드는 과정에서 생성된 FIXED컬럼의 합계는 8K를 넘기 때문에 동일한 에러가 발생한다. 마찬가지로 질의를 수행한 SELECT 타겟절의 컬럼 중 FIXED컬럼의 합계가 8K를 넘으면 이런 에러가 발생하게 된다.

➔ t1, t2 테이블의 컬럼을 생성시점부터 FIXED가 아닌 VARIABLE로 설정하도록 한다.

```
CREATE TABLE t1 (c1 CHAR(4096) VARIABLE)
```

Invalid cursor state

CURSOR를 사용할 때 다 사용된 CURSOR를 정상적으로 CLOSE CURSOR처리를 하지 않은 채 다시 OPEN CURSOR를 하려고 시도할 경우 발생한다. 또는 이미 Fetch가 완료된 CURSOR를 다시 Fetch하려고 시도하는 경우에도 발생한다.

간혹, 사용자 프로그램이 쓰레드인 경우 연결객체에 대한 동시성 제어를 제대로 하지 않아 자신의 쓰레드에 유효하지 않은 커서를 핸들링 할 경우도 발생하게 되는데 앞에서 설명한 것처럼 반드시 CURSOR를 사용 후에는 명시적으로 CLOSE CURSOR 구문을 사용함으로써 문제를 해결하도록 해야 한다.

```
DECLARE CURSOR
```

```
OPEN CURSOR
FETCH CURSOR
/** CLOSE CURSOR **/ ← 누락된 경우들이 대표적임.
```

Not defined cursor

CURSOR를 사용할 때 CURSOR NAME을 명시하는데 이 CURSOR NAME이 존재하지 않거나 혹은 DECLARE(PREPARE) CURSOR를 수행하는 과정에서 오류가 발생했는데 이를 체크하지 않아 EXECUTE 혹은 OPEN CURSOR단계에서 오류가 발생할 때 이 에러가 나타난다. 사용자는 반드시 다음과 같이 개발을 해야 한다.

```
DECLARE CURSOR1 FOR :statement;
If (SQLCODE != 0)                    ← PREPARE단계에서도 반드시 에러체크를 해야 함
    Error_process();
OPEN CURSOR1 USING :variable;
```

Invalid request to process the SQL statement

일반적으로 쓰레드 구조의 프로그램에서 자주 발생하며 앞에서 설명한 바와 같이 쓰레드 구조의 연결 객체에 대한 동시성 제어가 올바르지 않을 경우 ALTIBASE 서버와 클라이언트 간에 주고 받아야 하는 프로토콜의 순서가 틀려지는 경우 발생한다.

먼저, 프로그램의 구조가 쓰레드라면 이러한 동시성 제어가 올바른지 확인하도록 해야 한다. 혹은, PREAPRE->BINDING->EXECUTE와 같이 질의를 수행해야 하는 순서가 제대로 지켜지지 않은 경우가 있는지 확인해야 한다.

Invalid literal

이 에러는 숫자형 타입에 문자를 입력하는 등의 오류가 발생할 경우 나타난다.

```
CREATE TABLE t1 (a INTEGER);
INSERT INTO t1 VALUES ('AAAA');
```

위의 예처럼 컬럼 타입이 숫자형임에도 데이터를 문자열로 삽입하려고 시도할 경우 발생한다.

반대로, 문자열을 숫자형 변수에 담으려고 할 경우에는 다음과 같이 오류가 발생한다.

```
Invalid character value for cast specification.
```

Invalid length of data type

컬럼 타입보다 더 큰 값으로 데이터가 입력되는 경우 발생한다.

```
CREATE TABLE t1 (a CHAR(2));
INSERT INTO t1 VALUES ('aaaa');
[ERR-2100D : Invalid length of the data type]
```

위의 예처럼 컬럼 타입은 2byte까지 저장이 가능한데 입력을 4byte로 하려고 하면 지정된 컬럼 크기보다 데이터 값이 크기 때문에 에러가 발생한다.

Indicator variable required but not supplied error

이 에러는 Precompiler를 사용하는 개발환경에서 발생을 하는데 SELECT 결과에서 컬럼의 리턴 값이 NULL로 리턴 될 경우 INDICATOR변수가 지정되지 않으면 SQL_SUCCESS_WITH_INFO와 함께 이 오류메시지를 발생하게 된다.

```
int IND_C1;
int IND_C2;
EXEC SELECT C1, C2
   INTO :H_C1 INDICATOR :IND_C1 , :H_C2 INDICATOR :IND_C2;
```

위의 경우처럼 INDICATOR변수를 명시적으로 사용하거나 또는, Precompiler의 옵션에서 –unsafe_null 옵션을 이용하여 에러가 발생하지 않도록 할 수 있다.

Incompatible NLS between the client and the server

ALTIBASE 서버로 접속을 시도할 때 서버에 설정된 문자셋과 다른 문자셋 값으로 접속을 시도할 경우 발생한다. 예를 들어, 서버는 MS949 로 설정된 상태인데 클라이언트에서 US7ASCII와 값으로 접속을 시도할 경우 이러한 에러가 발생한다.

서버에 설정된 문자셋을 조회하는 방법은 다음과 같다.

```
SELECT * FROM V$NLS_PARAMETERS; (버전 5.3 이상)
SELECT name, value1 FROM V$PROPERTY WHERE name = 'NLS_USE'; (버전 5.1 이하)
```

이 에러는 5.3 이전 버전들에서만 발생하게 된다. 5.3 버전부터는 문자셋이 달라도 US7ASCII로 접속은 가능하다. 다만, 서버와 클라이언트간의 문자셋의 차이로 한글과 같은 문자를 저장하려 할 경우 올바르지 않은 데이터로 저장될 수 있음으로 주의가 필요하다.

Invalid size of data to bind to host variable [Data Size ...]

이 에러는 호스트 변수의 크기보다 더 큰 데이터를 넣어 질의가 처리될 경우 발생한다. 예를 들어, CHAR (20)으로 선언한 호스트변수에 20 보다 큰 문자열이 들어올 경우

발생한다. 일반적으로 개발자가 변수의 초기화의 실패 또는, 쓰레기 값이 들어올 경우 발생할 수 있다. 또는, 쓰레드 환경의 프로그램에서 동시성 제어 실패로 A 문장을 PREPARE한 후 바인딩을 해야 하는데 B 문장의 바인딩을 수행할 경우 발생할 수 도 있다. (A문장은 20byte를 binding하면 되는데 잘못 핸들링 하여 B문장의 100byte를 바인딩 하는 형태)

일단, 사용자는 해당 호스트 변수 값을 출력하여 명확히 길이를 넘지 않는지 체크하고 쓰레드 부분의 핸들링에 있어서도 동시성 부분에 문제가 없는지 확인하도록 해야 한다.

Invalid character in use

일반적으로 한글에 대해 LIKE 검색 시 발생을 하게 되는데 예를 들어 ALTIBASE 서버의 문자셋 값을 KO16KSC5601 로 설정하였는데 KO16KSC5601 의 표현 범위를 넘어선 문자가 뒤섞여서 저장된 경우 발생한다.

위의 오류가 발생한 경우에는 다음과 같이 작업을 하는 것을 권장한다.

1. iloader를 통해 테이블의 데이터를 다운로드
2. ALTIBASE 구동 중지
3. $ALTIBASE_HOME/conf/altibase.properties 의 NLS_USE값을 MS949 로 변경
4. ALTIBASE 구동 개시
5. 문제가 되는 대상 테이블을 truncate (delete는 안됨)
6. iloader를 통해 (1)에서 받은 데이터를 다시 업로드

즉, 한글을 사용하는 경우라면 KSC5601 보다 MS949 / UTF8 과 같은 문자셋을 사용하고 서버와 일치하는 문자셋으로 접속하여 처리할 것을 권장한다.

Too many pages are allocated

ALTIBASE 메모리 테이블스페이스의 경우는 모든 메모리 테이블스페이스의 합산이 $ALTIBASE_HOME/conf/altibase.properties 에서 MEM_MAX_DB_SIZE를 초과할 수 없다. 이 에러메시지는 모든 공간이 소진되어 발생한다.

따라서, 불필요한 테이블을 삭제하거나 MEM_MAX_DB_SIZE를 더 큰 값으로 늘리고 ALTIBASE를 재 구동해야 한다. 이러한 임시 조치 이후에는 어떤 이유로 메모리 테이블스페이스의 사용량이 증가 하였는지 각 테이블 별 사용량의 조회나 대량 변경작업등이 발생하지 않았는지 확인하여 조치할 필요가 있다.

The Tablespace does not have enough free space

테이블스페이스의 여유공간이 부족할 경우 발생한다. 디스크 테이블스페이스의 경우 데이터파일의 추가 작업이 필요하다.

The transaction exceeds lock timeout specified by user

DML간에는 Lock을 대기하고 서버에 설정된 TIMEOUT정책에 의해 동작을 하게 되지만 DML에 의해 잡힌 Lock으로 인해 DDL이 대기하게 될 경우 즉시, 이 에러가 발생하게 된다.

시간	A 세션	B 세션
T1	INSERT INTO x1 VALUES … (Lock획득)	
T2		ALTER TABLE x1 ADD COLUMN …(Error)

위의 표와 같이 A세션이 먼저 DML을 통해 Lock을 획득한 상태에서 B세션이 DDL을 수행하려고 할 경우 이 에러가 발생하기 때문에 해당 테이블에 Lock이 존재하는지 체크하고 조치한 후 작업을 진행하면 된다.

The update log size '...' is bigger than TRX_UPDATE_MAX_LOGSIZE '...'

ALTIBASE에서는 대량 변경작업이 유발하는 후속적인 장애상황을 방지하기 위해 변경 트랜잭션에 의해 발생하는 리두로그의 양이 $ALTIBASE_HOME/trc/altibase.properties에 TRX_UPDATE_MAX_LOGSIZE에 명시된 값보다 커질 경우 이 에러를 발생시키고 롤백처리를 하게 된다.

사용자는 해당 트랜잭션이 반드시 하나의 질의문으로 처리되어야 한다면 세션 레벨에서 다음과 같이 이 속성 값을 변경하여 처리하도록 한다.

```
ALTER SESSION SET TRX_UPDATE_MAX_LOGSIZE = 0;   (해당 제약을 없애는 설정)
```

이렇게 변경하여 처리할 경우 리두로그 파일의 생성이 많아져서 디스크 풀 장애와 같은 상황이 발생할 수 있으므로 주의가 필요하다. 또한, ALTIBASE는 대량 변경 작업 시 MVCC로 인한 리소스의 증가를 방지하기 위해 테이블에 X-Lock을 획득하려고 시도하기 때문에 X-Lock획득 이후 해당 테이블에 조회 및 변경이 X-Lock으로 인해 대기하게 됨으로 사용자의 주의가 필요하다.

String data right truncated

SELECT문 등에서 리턴 되는 문자타입의 값이 선언된 호스트변수의 크기보다 클 경우 발생한다. DB에 접속하여 조회하는 테이블의 컬럼 크기를 확인하고 사용된 호스트 변수의 크기를 (컬럼 크기 + 1 byte)로 선언하도록 해야 한다.

Value overflow

숫자형 컬럼의 표현 범위보다 더 큰 값이 입력될 경우 발생한다.

```
CREATE TABLE x2 (a NUMERIC(10, 5));
```

```
insert into x2 values (123444.5678);
[ERR-21010 : Value overflow]
```

Several statement still opened

Fetch 프로토콜이 진행 중에서 CREATE TABLE과 같은 질의가 수행될 때 발생한다. 따라서, 모든 동일 세션에서 CURSOR-FETCH가 완료된 후 DDL을 수행하거나 열려 있는 CURSOR를 CLOSE한 이후 DDL질의를 수행하도록 해야 한다. 또는, 별도의 세션에서 DDL질의를 수행하도록 코드의 변경이 필요하다.

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

기술지원본부
서울특별시 구로구 구로 3 동 182-13
대륭포스트 2 차 908 호
02-2082-1000

기술지원센터
02-2082-1114
http://support.altibase.com