

ALTIBASE Microsoft Windows User's Manual

Release 3.5.9.0

High Performance Main Memory DBMS [®]
ALTIBASE



ALTIBASE Microsoft Windows User's Manual
Release 3.5.9.0

Copyright © 2001 ALTIBASE Corporation.
Inyoung Bldg 5F, 44-11, Youido-dong, Youngdungpo-gu, Seoul, Korea
Tel: +82-2-769-7500 Fax: +82-2-769-7510
All Rights Reserved.

본 문서의 저작권은 ㈜알티베이스에 있습니다. 당사의 동의 없이 무단으로 복제
또는 전용할 수 없습니다.

㈜알티베이스
150-890
서울시 영등포구 여의도동 44-11 인영빌딩 5층
전화: 02-769-7500 팩스: 02-769-7510
e-mail: support@altibase.com
homepage: <http://www.altibase.com>

목 차

ALTIBASE Microsoft Windows User's Manual	1-1
1. 알티베이스 설치	1-1
알티베이스 설치하기	1-2
알티베이스 구성 요소	1-4
알티베이스 디렉터리	1-6
2. 데이터베이스 파일 생성과 삭제	2-1
데이터베이스 생성	2-2
메타 테이블	2-5
데이터베이스 삭제	2-14
3. 알티베이스 실행과 종료	3-1
알티베이스의 실행	3-2
알티베이스의 종료	3-5
4. 알티베이스 프로퍼티	4-1
환경 설정 방법	4-2
데이터베이스 구성 프로퍼티	4-3
성능 관련 프로퍼티	4-5
세션 연결 프로퍼티	4-8
트랜잭션 프로퍼티	4-11
로깅 프로퍼티	4-13
데이터베이스 이중화 프로퍼티	4-16
메시지 로그 프로퍼티	4-20
쿼리 최적화 프로퍼티	4-21
인덱스 타입 프로퍼티	4-22
기타 프로퍼티	4-23
5. 백업 및 복구	5-1
로그(Log)와 복구	5-2
체크포인트	5-4
백업 및 복구	5-6
6. 응용프로그램 작성	6-1
프로그램 작성 방법	6-2
Altibase CLI를 활용한 프로그램	6-3
JDBC를 활용한 프로그램	6-10
ODBC를 활용한 프로그램	6-14

7. 데이터베이스 이중화	7-1
데이터베이스 이중화 기능	7-2
데이터베이스 이중화 방법	7-3
이중화 기능의 사용 방법	7-4
8. 관리도구	8-1
알티베이스 관리도구 (dbadmin)	8-2
공유 메모리 관리도구 (shmutil)	8-17
9. 차기 구현 요소	9-1
누락된 기능 및 요소	9-2
10. FAQ	10-1
Starting FAQ	10-2

1. 알티베이스 설치

알티베이스 설치하기

- 알티베이스를 설치할 머신에 새로운 계정 예를 들어 **altibase**를 생성한다. 새로운 계정이 필요 없을 경우에는 관리자 아이디를 사용한다. 예) **administrator**

알티베이스 홈페이지 다운로드 페이지에서 윈도우 설치 버전을 선택하고, 다운받고 실행한다.

- 시스템의 환경변수에 아래의 내용을 추가한다.

```
set ALTIBASE_HOME c:\altibase3\altibase_home
set PATH .;%ALTIBASE_HOME%\bin;%PATH%
```

- 라이선스 파일 (license: 확장자 없음)을 %ALTIBASE_HOME%\conf 에 복사한다.

```
copy license %ALTIBASE_HOME%\conf\
* 라이선스는 license@altibase.com 으로 메일을 보내시면 라이선스를 발급해 드립니다.
```

- 설치 완료

알티베이스 설치에 대한 자세한 내용은 *ALTIBASE Administrator's Guide* 내에 1장. 설치 참고

환경 설정

알티베이스를 이상없이 설치하였으면 알티베이스가 실행될 수 있는 환경을 설정해야 한다.

알티베이스 패키지 설치 작업이 끝났다면 (1) 라이선스 발급 (2) 알티베이스 프로퍼티 설정 (3) 데이터베이스 생성 (4) 알티베이스 샘플 프로그램을 수행하여 알티베이스 사용에 문제가 없는지를 확인하여야 한다.

운영관리자

- 환경변수 설정 및 확인
 - 환경변수에 ALTIBASE_HOME 의 경로를 설정해야 한다.
 - PATH = %ALTIBASE_HOME%\bin
 - CLASSPATH = %ALTIBASE_HOME%\lib\Altibase.jar
- 알티베이스 프로퍼티 설정

자세한 내용은 제 4장 알티베이스 프로퍼티 참조

클라이언트 사용자

- 환경변수 설정 : 운영관리자와 동일

알티베이스 구성 요소

알티베이스 제품을 구성하고 있는 요소들은 크게 서버 부문, 프로그래밍 라이브러리 부문, 그리고 지원 도구 부문으로 구분된다. 각 부문에 대한 자세한 내용은 '[알티베이스 디렉터리](#)' 부분을 참조.

서버 구성 요소

서버의 구성 요소로는 다음과 같은 것들이 있다.

- 데이터베이스 파일을 생성하는 기능(createdb.exe).
- 데이터베이스 파일을 삭제하는 기능(destroydb.exe).
- 클라이언트-서버 구조로 운영할 때의 서버(Altibase.exe).

프로그래밍 인터페이스

알티베이스의 응용 프로그램을 작성할 때 필요한 구성 요소들로서, 다음과 같은 것들이 있다.

- C 또는 C++ 언어로 프로그램을 작성할 때 필요한 라이브러리.
- Altibase CLI 인터페이스를 제공하는 라이브러리(odbccli.lib)가 포함된다.
- 클라이언트 환경설정을 위한 라이브러리 id.lib를 제공한다.
- MS-Windows 환경에서 ODBC 드라이버를 활용하여 프로그래밍할 수 있는 라이브러리(altibase_odbc.dll).
- 자바 언어로 프로그래밍할 때 필요한 자바 클래스 라이브러리(Altibase.jar).
- 프로그래밍에 필요한 헤더 파일들. 이 것에 대한 자세한 설명은 [제 6장 응용프로그램 작성](#)에서 설명 한다.

관리 도구

dbadmin.exe

알티베이스 서버를 관리할 때 필요한 도구(dbadmin.exe). 이 도구는 알티베이스 서버의 실행과 종료, 서버의 운영 상태, 서버가 사용하는 자원의 관한 정보 등을 열람 및 관리할 수 있다. 이 도구에 대한 상세한 내용 및 사용법은 [제 8장 관리도구](#)에서 다룬다.

shmutil.exe

공유메모리 데이터베이스 관리 도구(**shmutil.exe**). 이 도구는 알티베이스의 공유 메모리를 메인 메모리 데이터베이스로 사용하는 경우, 공유메모리에 관련된 관리작업을 한다. 이 도구에 대한 자세한 내용은 제 8장 관리도구에서 다룬다.

지원 도구

알티베이스를 보다 쉽고 다양하게 활용할 수 있도록 하고 데이터베이스 관리에 필요한 정보를 얻는데 필요한 여러 가지 지원 도구가 있다. 다음과 같은 것으로 구성된다.

isql.exe

대화형으로 알티베이스 데이터베이스 질의를 수행할 수 있는 도구. 이 도구에 대한 자세한 내용은 *ALTIBASE iSQL User's Manual*을 참고.

iload.exe

데이터베이스의 특정 테이블을 로드 및 언로드할 수 있는 도구. 이 도구에 대한 자세한 내용은 *ALTIBASE iLoader User's Manual*을 참고.

killCheckServer.exe

실행중인 **checkServer.exe**를 종료시킨다.

checkServer.exe

알티베이스의 상태를 체크하여 비정상종료시 수행해야할 일을 스크립트 파일로 만들어 실행할수 있도록 한다.

- 사용법

Dos prompt> **checkServer [-f server-restart-script-file]**

- 사용예

다음과 같이 서버를 재시작하는 스크립트 파일을 생성한다.
(파일명 : **restart.bat**)

```
ALTIBASE_HOME=c:\altibase3\altibase_home\
%ALTIBASE_HOME%\bin\server.bat start
```

다음과 같이 프롬프트에서 실행시킨다.

```
DOS Prompt> start checkServer.exe -f
restart.bat
```

restoredb.exe

온라인 백업 기능을 이용하여 생성된 데이터베이스 파일들을 사용하여 데이터베이스를 이전의 상태로 복구한다. 자세한 설명은 제 5장 백업 및 복구의 **restoredb**를 참조하기 바란다.

알티베이스 디렉터리

알티베이스를 설치 하면 다음의 디렉터리가 생성된다. 각각의 디렉터리 역할과 이 디렉터리가 포함하는 내용에 관하여 설명한다.

알티베이스 홈 디렉터리

알티베이스 홈 디렉토리는 환경 변수 `ALTIBASE_HOME`에 지정된다. 이 홈 디렉토리가 포함하는 내용은 `bin`, `conf`, `lib`, `include`, `msg`, `db`, `logs`, `sample`, `install`, `audit`, `trc`, `admin`, 그리고 `arch_logs` 디렉토리를 포함하고 있다. 각각에 관하여 설명하도록 한다.

trc 디렉터리

알티베이스 운영 상태를 기록한 파일들이 존재한다.

altibase_boot.log

알티베이스 서버가 동작된 상태를 기록하고 있다. 이 파일이 기록하고 있는 정보로는 알티베이스 구동 및 종료시 얻어지는 시스템 정보에 대한 세부사항이 있으며, 또한 알티베이스의 비정상 종료시 알티베이스의 에러발생 상태를 기록한다.

bin 디렉터리

알티베이스를 포함한 알티베이스 관리도구와 알티베이스를 사용하는데 필요한 지원도구들의 실행 파일이 존재하는 디렉터리이다. 다음과 같은 파일이 존재한다.

`Destroydb.exe`, `createdb.exe`, `altibase.exe`, `altibase.map`, `dbadmin.exe`, `shmutil.exe`, `server.bat`, `checkServer.exe`, `killCheckServer.exe`, `isql.exe`, `iload.exe`, `audit.exe`, `sesc.exe`.

알티베이스 관리도구(`dbadmin`, `shmutil`)의 자세한 설명은 [제 8장 관리도구](#)를 참조.

알티베이스 지원도구의 자세한 설명은 [1-55 페이지 지원도구](#)를 참조하기 바란다.

conf 디렉터리

알티베이스 라이선스와, 알티베이스가 취할 수 있는 다양한 옵션을 수록한 프로퍼티 파일 (**altibase.properties**) 이 존재하는 디렉터리이다. 알티베이스가 실행될 때 라이선스를 확인하고 프로퍼티 파일을 참조하여 서버 상태를 초기화 한다. 알티베이스의 프로퍼티에 대한 자세한 설명은 [제 4장 알티베이스 프로퍼티](#)를 참조하기 바란다.

audit 디렉터리

이중화 동작시 발생한 불일치를 해결하는 알티베이스 유틸리티인 **audit**의 예제 스크립트 파일이 들어있는 디렉터리이다. 자세한 설명은 *ALTIBASE Audit User's Manual*을 참조하기 바란다.

lib 디렉터리

응용 프로그램 작성에 필요한 라이브러리 파일을 수록한 디렉터리이며 다음과 같은 파일이 있다. 각각의 라이브러리를 이용하여 응용 프로그램을 작성하는 방법은 [제 6장 응용프로그램 작성](#)에서 설명하였다.

Altibase.jar

알티베이스를 자바 응용프로그램에서 사용하기 위한 JDBC 드라이버이다.

sesc.lib

내장 SQL 프로그램을 작성할 때 필요한 라이브러리이다. 내장 SQL 프로그램 작성에 관한 자세한 내용은 *ALTIBASE Embedded SQL User's Manual*을 참조하기 바란다.

odbccli.lib

알티베이스 클라이언트-서버용 응용프로그램 작성을 위한 라이브러리이다.

include 디렉터리

Altibase CLI 라이브러리를 이용하여 응용 프로그램을 작성할 때 필요한 헤더 파일을 수록한 디렉터리이다.

sqlcli.h

클라이언트 응용 프로그램을 작성할 때 필요한 헤더 파일로서, **libid.a**를 링크할 때 사용한다.

idConfig.h

컴파일 과정을 위한 시스템 설정에 대한 정보를 담고 있다.

idTypes.h

Altibase CLI 응용 프로그램 개발 시 필요한 기초 데이터 타입에 대한 정보를 담고 있다.

iddTypes.h

Altibase CLI 응용 프로그램 개발 시 필요한 SQL 데이터 타입 및 연산자에 대한 정보를 담고 있다.

idcCli.h

Altibase CLI 응용 프로그램 개발 시 필요한 Altibase CLI 데이터 타입 및 연산자에 대한 정보를 담고 있다.

sescli.h

SES C/C++ 전처리기(precompiler)로 응용 프로그램 개발 시 필요한 헤더 파일.

ses.h

SES C/C++ 전처리기(precompiler)로 응용 프로그램 개발 시 에러처리 SQL문장 구조에 대한 정보를 담고 있다.

msg 디렉터리

오류 메시지를 수록한 파일들을 포함하는 디렉터리이다. 다음과 같은 파일이 있다. 오류 메시지는 영문과 한글 두가지로 제공된다.

- 영문 : US7ASCII 한글 : KO16KSC5601

E_SM_US7ASCII.msb

자료 저장 관리 모듈에서 발생할 수 있는 오류 메시지를 수록한 파일이다.

E_QP_US7ASCII.msb

질의 처리 모듈에서 발생할 수 있는 오류 메시지를 수록한 파일이다.

E_MM_US7ASCII.msb

알티베이스 서버 메인 모듈에서 발생할 수 있는 오류 메시지를 수록한 파일이다.

E_ID_US7ASCII.msb (E_MT_US7ASCII.msb)

함수 실행이나 데이터 타입과 관련된 오류 메시지를 수록한 파일이다.

dbms 디렉터리

데이터베이스가 존재하는 디렉터리이다. 이 디렉터리의 위치 및 디렉터리명은 프로퍼티 파일에 명시되어 있다.

logs 디렉터리

로그 파일 정보를 수록한 파일(loganchor)과 1개 이상의 로그 파일이 존재하는 디렉터리이다. 이 디렉터리의 위치 및 디렉터리명은 프로퍼티 파일에 명시되어 있다. 로그 파일명은 알티베이스 시스템에서 자동으로 결정된다.

install 디렉터리

알티베이스 응용프로그램 작성에 필요한 makefile을 위한 매크로 설정 등이 포함된 altibase_env.mk 파일과 README 파일이 있다.

sample 디렉터리

알티베이스의 응용 프로그램을 샘플로 제공한 디렉터리이다.

Altibase CLI, JDBC, ODBC, SES C/C++ 라이브러리를 이용하여 작성된 프로그램과 Makefile이 수록되어 있다.

admin 디렉터리

기존 (버전 3.5.7.0 이전) install 디렉토리 밑에 있던 sql 화일들과 fixed table 관련 view 생성 스크립트 파일이 있다.

arch_logs 디렉토리

Sync가 끝난 로그 파일이 존재하는 디렉토리이다. 이 디렉토리의 위치 및 디렉토리명은 프로퍼티 파일에 명시되어 있다.

2. 데이터베이스 파일 생성과 삭제

데이터베이스 생성

알티베이스를 사용하기 위해서는 반드시 데이터베이스 파일을 생성 시켜 놓아야 한다. 그렇지 않으면 알티베이스가 실행되지 않는다. 여기에서는 데이터베이스 파일의 생성 방법, 데이터베이스 파일의 구조에 관하여 설명한다.

생성하기

알티베이스를 실행하기 위해서는 반드시 먼저 데이터베이스 파일을 생성시켜 놓아야 한다. 데이터베이스 파일이 없는 상태에서는 알티베이스가 실행되지 않는다. 데이터베이스의 생성은 **createdb** 유틸리티를 이용하는데 사용 법은 다음과 같다.

```
createdb -p pagecount | -M dbsize [-d home_dir][-f
property_file]
```

- p pagecount** 이 옵션은 데이터베이스 파일이 포함할 페이지의 수를 나타내는 것으로서, 반드시 값을 지정해야 한다. 페이지의 크기는 32 KB 이기 때문에 **pagecount** 의 값에 32 KB 를 곱하면 데이터베이스의 크기가 결정된다.
- M dbsize** 이 옵션은 실제 생성될 데이터베이스 파일의 크기를 메가바이트 단위로 지정한다. 이 값은 지정된 파일 크기에 가장 근사한 페이지 개수로 변환되어 데이터베이스 파일을 생성한다.
- d home_dir** 이 옵션은 알티베이스의 홈 디렉터리 경로를 명시하여 주는 것이다. 데이터베이스 파일 경로는 이 디렉터리 경로에다 프로퍼티 파일에서 **DB_DIR** 에 명시된 데이터베이스 디렉터리 명을 합친 것으로 결정된다. 이 옵션 값을 입력하지 않으면 환경변수 **ALTIBASE_HOME** 에 명시된 알티베이스 홈 디렉터리 경로를 프로퍼티 파일에 명시된 데이터베이스 디렉터리 명을 합친 것 값으로 데이터베이스 파일 경로를 결정한다.
- f property_file** 알티베이스 프로퍼티 파일을 명시적으로 알려 주는 것이다. 즉, 이 옵션을 입력할 때는 반드시 파일이 포함된 절대 경로와 파일명을 명시해야 하며, 만일 이 값을 입력하지 않으면 알티베이스는 홈 디렉터리 안의 **conf** 디렉터리 (**\$ALTIBASE_HOME/conf**) 에 있는

altibase.properties 를 프로퍼티 파일로 결정한다.

createdb는 데이터베이스 파일뿐만 아니라 로그 파일도 함께 생성한다. 로그 파일의 경로는 home_dir 또는 ALTIBASE_HOME 환경변수가 포함하는 값에 프로퍼티 파일의 LOG_DIR 값을 합친 것으로 결정된다.

예 제

프로퍼티 파일에서 DB_NAME = mydb, DB0_DIR = ?/dbs, DB1_DIR = ?/dbs, LOG_DIR = ?/logs로 되어 있는 상황에서 아래와 같이 createdb를 실행하면, 다음과 같다. (?: 환경변수의 ALTIBASE_HOME 경로를 의미한다.)

```
createdb -M 100
```

\$ALTIBASE_HOME/dbs 경로에는 100MB의 mydb라는 데이터베이스 파일이 생성되고, \$ALTIBASE_HOME/logs에는 loganchor 파일과 logfile0라는 로그 파일이 생성된다.

```
Dos prompt> createdb -M 100
CreateDB: Release 3.2.0 - Production on Feb 27 2003 16:07:12
(c) Copyright 2001 ALTIBase Corporation. All rights reserved.
```

```
DB Info (Page Size = 32768)
      (Page Count = 3200)
      (File Size = 104857600)
```

```
Ready for Creating Database [mydb] (Y/N) ?y
```

```
BEGIN:
Database Creation ....
SUCCESS:
Database Creation Completed..
BEGIN:
QP Meta Information Creation ....
SUCCESS:
QP Meta Information Creation Completed.
BEGIN:
DB Writing (Page Count = 3200)
SUCCESS:
DB Writing Completed. All Done.
Dos prompt>
```

데이터베이스 구조

알티베이스 버전 2.0 부터 데이터베이스 메모리 영역은 동일한 크기의 페이지의 집합으로 구성되어 있다. 페이지는 데이터베이스를 관리하기 위한 정보를 담고 있는 catalog 페이지와 사용자 데이터를 저장하는 데이터 페이지로 나누어진다. catalog 페이지는 현재 생성된 데이터베이스에 대한 상세한 명세를 담고 있으며, 알티베이스의 구동 및 종료시 데이터베이스의 일관성 검사 및 변경정보를 유지하게 된다.

catalog 페이지는 데이터베이스에서 사용되는 자기자신을 제외한 나머지 데이터 페이지에 대한 리스트 및 사용정보를 담고 있으며, 백업 데이터베이스의 가장 첫번째 페이지에 위치하고 있는 매우 중요한 페이지 영역이다.

데이터 페이지는 실제로 사용자 데이터가 저장되는 영역이며, 페이지 헤더와 페이지 본체(**body**)로 나누어진다. 페이지 헤더는 서로 간의 리스트를 유지하기 위한 링크 정보와 타입, 그리고 자기 자신의 페이지 번호로 구성되어 있으며, 페이지 바디는 실제 데이터를 저장하기 위한 여러 개의 슬롯으로 분할된다. 이 슬롯이 실제 데이터가 저장되는 최종 저장소이다.

하나의 페이지 크기가 **32KB(2^{15})** 이기 때문에 **32** 비트 운영체제에서는 최대 2^{17} 개의 페이지가 존재할 수 있고, **64** 비트 운영체제에서는 최대 2^{49} 개의 페이지가 데이터베이스 내에 존재할 수 있다.

알티베이스 버전 **2.0** 부터는 위에서 기술한 페이지를 제외한 데이터베이스를 구성하는 나머지 어떠한 영역의 메모리도 백업 데이터베이스로 내려가지 않는다. 예를 들면, 인덱스 메모리 영역의 경우 임시 메모리 영역에 존재하기 때문에 런타임시 빈번하게 발생하는 인덱스 연산에 대해 로깅 작업을 하지 않고, 서버 종료시 모두 삭제된다. 이로 인해 인덱스에 대한 로깅 부하가 사라 졌으며, 로그의 양 뿐 아니라 실행 성능상의 이점도 함께 기대할 수 있게 되었다. 또한, 서버 종료시 소멸 되었던 인덱스 메모리 공간은 서버 재구동시 매우 빠른 속도로 재구성되기 때문에 실제 서비스에 아무런 영향을 미치지 않도록 설계되었다.

메타 테이블

메타 테이블이란 알티베이스 데이터베이스의 스키마에 관한 모든 정보를 수록하기 위한 시스템 정의 테이블이다. 여기에서는 메타 테이블의 종류 및 구조, 그리고 메타 테이블의 변경에 관하여 설명한다.

구조 및 기능

SYS_COLUMNS_

모든 테이블에서 정의된 열들의 정보를 기록하는 메타 테이블이다.

Column name	Type	Length	Description
COLUMN_ID	INTEGER	4	열의 식별자
DATA_TYPE	INTEGER	4	데이터 타입
LANG_ID	INTEGER	4	언어 식별자 (CHAR, VARCHAR 일 때만 의미 있음)
OFFSET	INTEGER	4	레코드의 오프셋
SIZE	INTEGER	4	길이
USER_ID	INTEGER	4	사용자 식별자
TABLE_ID	INTEGER	4	테이블 식별자
PRECISION	INTEGER	4	지정한 precision
SCALE	INTEGER	4	지정한 scale
COLUMN_ORDER	INTEGER	4	열의 생성순서
COLUMN_NAME	VARCHAR	40	열의 이름
IS_NULLABLE	CHAR	1	널 허용 여부 T: include NULL F: not include NULL
DEFAULT_VAL	VARCHAR	4000	기본 값
IS_VARING	CHAR	1	VARCHAR VARIABLE 형 인지 VARCHAR FIXED 형 인지를 나타냄 T: variable column F: fixed column

* 데이터타입은 `iddTypes.h`에 지정되어있는 값 (1: CHAR, 2: NUMERIC, 2: DECIMAL, 4: INTEGER, 5: SMALLINT, 6: FLOAT, 6: NUMBER, 7: REAL, 8: DOUBLE, 9: DATE, VARCHAR: 12, 30: BLOB, BIGINT: -5, 20001: HSS_BYTES, 20002: HSS_NIBBLE)

SYS_CONSTRAINTS_

사용자 정의 테이블의 제약 조건에 관한 정보를 포함하는 메타 테이블이다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	사용자 식별자
TABLE_ID	INTEGER	4	테이블 식별자
CONSTRAINT_ID	INTEGER	4	제약조건 식별자
CONSTRAINT_NAME	VARCHAR	40	제약조건 이름
CONSTRAINT_TYPE	INTEGER	4	제약조건 타입
INDEX_ID	INTEGER	4	제약조건의 인덱스 번호
COLUMN_CNT	INTEGER	4	제약조건의 칼럼 개수
REFERENCED_TABLE_ID	INTEGER	4	parent table의 식별자
REFERENCED_CONSTRAINT_ID	INTEGER	4	관련된 제약 조건 식별자

* 제약조건 타입은 0: FOREIGN KEY, 1: NOT NULL, 2: UNIQUE, 3: PRIMARY KEY, 4: NULL, 5: TIMESTAMP

SYS_CONSTRAINT_COLUMNS_

사용자 테이블에 정의된 모든 제한조건에 걸쳐있는 열의 정보를 기록하고 있는 메타 테이블이다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	사용자 식별자
TABLE_ID	INTEGER	4	테이블 식별자
CONSTRAINT_ID	INTEGER	4	제약조건 식별자
CONSTRAINT_COL_ORDER	INTEGER	4	제약 조건 칼럼 순서
COLUMN_ID	INTEGER	4	열 식별자

SYS_DATABASE_

데이터베이스 이름과 메타 버전 정보를 기록하는 테이블이다.

Column name	Type	Length	Description
DB_NAME	VARCHAR	40	데이터베이스 이름
META_MAJOR_VER	INTEGER	4	데이터베이스 메타 테이블 버전 (주)
META_MINOR_VER	INTEGER	4	데이터베이스 메타 테이블 버전 (부)

* 현재 알티베이스가 단일 데이터베이스 만을 지원하기 때문에 데이터베이스 이름은 저장되지 않는다.

* META_MAJOR_VER는 메타 테이블의 베이스가 변경될 경우 증가하는 값으로, 데이터베이스의 이 버전과 알티베이스 바이너리의 해당 버전이 일치하지 않은 경우 데이터베이스 마이그레이션 작업을 요한다.

* **META_MINOR_VER**는 메타 테이블의 일부 스키마 또는 레코드 값이 변경될 경우 증가하는 값으로 데이터베이스의 이 버전과 알티베이스의 해당 버전이 다른 경우 내부적으로 값을 비교해 상위 버전으로 메타 테이블에 대해 자동 업그레이드를 수행한다.

SYS_GRANT_OBJECT_

Column name	Type	Length	Description
GRANTOR_ID	INTEGER	4	권한 부여자의 식별자
GRANTEE_ID	INTEGER	4	권한 피수여자의 식별자
PRIV_ID	INTEGER	4	Privilege 식별자 1: all privilege 201 ~ 232
USER_ID	INTEGER	4	객체 소유자의 식별자
OBJ_ID	INTEGER	4	객체 식별자
OBJ_TYPE	CHAR	1	객체 타입 T: table, S: sequence, P: PSM, V: view
WITH_GRANT_OPTION	INTEGER	4	객체 접근 권한 부여시 WITH GRANT OPTION의 사용 유무 0: 사용 안 함 1: 사용

SYS_GRANT_SYSTEM_

Column name	Type	Length	Description
GRANTOR_ID	INTEGER	4	권한 부여자의 식별자
GRANTEE_ID	INTEGER	4	권한 피수여자의 식별자
PRIV_ID	INTEGER	4	Privilege 식별자 1: all privilege 201 ~ 232

SYS_INDEX_COLUMNS_

사용자 테이블에 정의된 모든 인덱스에 걸쳐있는 열의 정보를 기록하고 있는 메타 테이블이다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	사용자 식별자
INDEX_ID	INTEGER	4	인덱스 번호
COLUMN_ID	INTEGER	4	열의 식별자
INDEX_COL_ORDER	INTEGER	4	인덱스 칼럼 순서
SORT_ORDER	CHAR	1	ascending or descending
TABLE_ID	INTEGER	4	테이블 식별자

SYS_INDICES_

사용자 테이블에 정의된 모든 인덱스 정보를 기록하고 있는 메타 테이블이다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	사용자 식별자
TABLE_ID	INTEGER	4	테이블 식별자
INDEX_ID	INTEGER	4	인덱스 번호
INDEX_NAME	VARCHAR	40	인덱스 이름
INDEX_TYPE	INTEGER	4	인덱스 타입
IS_UNIQUE	CHAR	1	UNIQUE 여부
COLUMN_CNT	INTEGER	4	인덱스 열 개수
IS_RANGE	CHAR	1	Range scan 가능 여부
IS_PERS	CHAR	1	T: persistent mode F: non-persistent mode

* 인덱스 타입은 TTREE: 1, BTREE: 2, RTREE: 3

SYS_PRIVILEGES_

Column name	Type	Length	Description
PRIV_ID	INTEGER	4	Privilege 식별자 1: all privilege 201 ~ 232
PRIV_TYPE	INTEGER	4	Privilege Type 2: SYSTEM 또는 1: OBJECT
PRIV_NAME	VARCHAR	40	Privileges 이름

SYS_PROCEDURES_

사용자가 정의한 저장 프로시저와 저장 함수들에 대한 정보로 저장 프로시저 이름, 리턴 타입, 파라미터 개수, 실행 가능 여부 등을 기록하는 테이블이다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	사용자 식별자
PROC_OID	BIGINT	8	저장 프로시저 객체 식별자
PROC_NAME	VARCHAR	40	저장 프로시저 이름
OBJECT_TYPE	INTEGER	4	저장 프로시저인지 저장 함수인지를 구별하는 타입
STATUS	INTEGER	4	VALID: 0, INVALID: 1 (INVALID인 경우 재컴파일이 필요하며 재컴파일 후에도

			VALID 상태가 될 수 없으면 저장 프로시저를 실행 불가)
PARA_NUM	INTEGER	4	저장 프로시저 파라미터 개수
RETURN_DATA_TYPE	INTEGER	4	저장 함수의 리턴 데이터 타입
RETURN_SIZE	INTEGER	4	저장 함수의 리턴 데이터 타입 크기
RETURN_LANG_ID	INTEGER	4	리턴 타입 언어 식별자
RETURN_PRECISION	INTEGER	4	저장 함수의 리턴 데이터 타입 precision
RETURN_SCALE	INTEGER	4	저장 함수의 리턴 데이터 타입 scale
PARSE_NO	INTEGER	4	SYS_PROC_PARSE_에 저장된 텍스트 정보 레코드 수
PARSE_LEN	INTEGER	4	SYS_PROC_PARSE_에 저장된 텍스트 전체 길이
NATIVE_GROUP_OID	BIGINT	8	Native stored procedure의 group 식별자

SYS_PROC_PARAS_

사용자가 정의한 저장 프로시저와 저장 함수들의
인자(parameter)들에 대한 정보를 기록하는 테이블이다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	사용자 식별자
PROC_OID	BIGINT	8	저장 프로시저 식별자
PARA_NAME	VARCHAR	40	파라미터 이름
PARA_ORDER	INTEGER	4	파라미터의 순서로 첫번째 파라미터의 경우 1을 가짐
INOUT_TYPE	INTEGER	4	IN, OUT, INOUT
DATA_TYPE	INTEGER	4	파라미터의 데이터 타입
SIZE	INTEGER	4	파라미터 타입 size
LANG_ID	INTEGER	4	파라미터 타입 언어 식별자
PRECISION	INTEGER	4	파라미터 타입 precision
SCALE	INTEGER	4	파라미터 타입 scale
DEFAULT_VAL	VARCHAR	4000	파라미터의 기본 값

SYS_PROC_PARSE_

사용자가 정의한 저장 프로시저와 저장 함수들의 파싱할 텍스트
정보를 기록하는 테이블이다.

Column name	Type	Length	Description
-------------	------	--------	-------------

USER_ID	INTEGER	4	소유자 식별자
PROC_OID	BIGINT	8	저장 프로시저 객체 식별자
SEQ_NO	INTEGER	4	한 저장 프로시저의 텍스트 정보를 SYS_PROC_PARSE_에 여러 개의 레코드로 저장할 때 이들 레코드의 순서
PARSE	VARCHAR	100	저장 프로시저 텍스트

SYS_PROC_RELATED_

사용자가 정의한 저장 프로시저와 저장 함수들이 접근하는 테이블, 시퀀스, 저장 프로시저, 저장 함수, 또는 뷰들에 대한 정보를 기록하는 테이블이다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	소유자 식별자
PROC_OID	BIGINT	8	저장 프로시저 객체 식별자
RELATED_USER_ID	INTEGER	40	저장 프로시저에 사용된 테이블의 사용자 식별자
RELATED_OBJECT_NAME	VARCHAR	40	객체 이름
RELATED_OBJECT_TYPE	INTEGER	4	TABLE, SEQUENCE, PSM, or VIEW

저장프로시저 **PROC1**이 테이블 **t1**에 **INSERT** 작업을 수행하는 저장프로시저일 경우 **PROC1**의 소유자 식별자와 저장프로시저 식별자가 각각 **USER_ID**와 **PROC_OID**에 저장되고, 테이블 **t1**의 소유자 ID와 테이블 이름은 각각 **RELATED_USER_ID**, **RELATED_OBJECT_NAME**에 저장되며, **RELATED_OBJECT_TYPE**에는 **TABLE**임을 명시하게 된다.

SYS_REPLICATIONS_

이중화 관련 정보를 기록하고 있는 메타 테이블이다.

Column name	Type	Length	Description
REPLICATION_NAME	VARCHAR	40	이중화 이름
LAST_USED_HOST_NO	INTEGER	4	가장 최근에 사용한 주소 위치
HOST_COUNT	INTEGER	4	SYSTEM_SYS_REPL_HOSTS_ 에 있는 IP의 개수
IS_STARTED	INTEGER	4	이중화 시작 여부 (0: 중지, 1: 시작 상태)
XLSN_FILE_NO	INTEGER	4	원격 서버에 마지막으로 보낸 로그 파일의 파일번호
XLSN_OFFSET	INTEGER	4	원격 서버에 마지막으로 보낸 이중화 동작 파일의 오프셋
ITEM_COUNT	INTEGER	4	이중화 대상 테이블 개수
CONFLICT_RESOLUTION	INTEGER	4	0: 기존 방식과 동일, 1:

			Master로서 동작, 2: Slave로서 동작
--	--	--	----------------------------

SYS_REPL_HOSTS_

원격 서버에 관련된 정보를 가진 메타 테이블이다.

Column name	Type	Length	Description
HOST_NO	INTEGER	4	일련 번호
REPLICATION_NAME	VARCHAR	40	이중화 이름
HOST_IP	VARCHAR	40	원격 서버 IP
PORT_NO	INTEGER	4	원격 서버 포트 번호

SYS_REPL_ITEMS_

이중화 테이블에 관련된 정보를 가진 메타 테이블이다.

Column name	Type	Length	Description
REPLICATION_NAME	VARCHAR	40	이중화 이름
TABLE_OID	BIGINT	8	테이블 객체 식별자
LOCAL_USER_NAME	VARCHAR	40	지역 서버 사용자 이름
LOCAL_TABLE_NAME	VARCHAR	40	지역 서버 테이블 이름
REMOTE_USER_NAME	VARCHAR	40	원격 서버 사용자 이름
REMOTE_TABLE_NAME	VARCHAR	40	원격 서버 테이블 이름

SYS_TABLES_

메타 테이블들과 사용자가 정의한 테이블에 대한 정보를 기록하는 테이블이다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	사용자 식별자
TABLE_ID	INTEGER	4	테이블 식별자
TABLE_OID	BIGINT	8	테이블 객체 식별자
COLUMN_COUNT	INTEGER	4	테이블 열의 개수
TABLE_NAME	VARCHAR	40	테이블 이름
TABLE_TYPE	CHAR	1	T: table, S: sequence
REPLICATION_COUNT	INTEGER	4	테이블과 관련된 replication 개수
LOAD_STATUS	CHAR	1	T: pin, F: unpin
MAXROW	BIGINT	8	입력할 수 있는 최대 레코드 개수, 0: 제한 없음

SYS_USERS_

데이터베이스 사용자에 대한 정보를 기록하는 테이블이다. 여러 사용자에 대한 정보를 포함할 수 있다. 이 테이블의 구조는 아래와 같다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	사용자 식별자
USER_NAME	VARCHAR	40	사용자 이름
PASSWORD	VARCHAR	40	사용자 패스워드

* 패스워드는 ImMEdlAKKg4I로 암호화 되어 있다.

SYS_VIEWS_

뷰에 대한 기본 정보를 기록하는 테이블이다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	뷰의 소유자 식별자
VIEW_ID	INTEGER	4	뷰 식별자
STATUS	INTEGER	4	뷰 상태 (0: VALID, 1: INVALID)

SYS_VIEW_PARSE_

사용자가 정의한 뷰들의 파싱할 텍스트 정보를 기록하는 테이블이다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	뷰의 소유자 식별자
VIEW_ID	INTEGER	4	뷰 식별자
SEQ_NO	INTEGER	4	한 뷰 생성문 텍스트를 정보를 SYS_VIEW_PARSE_에 여러 개의 레코드로 저장할 때 이들 레코드의 순서
PARSE	VARCHAR	100	뷰 생성문 텍스트

SYS_VIEW_RELATED_

사용자가 정의한 뷰들이 접근하는 관련 객체에 대한 정보를 기록하는 테이블이다.

Column name	Type	Length	Description
USER_ID	INTEGER	4	뷰의 소유자 식별자
VIEW_ID	INTEGER	4	뷰 식별자
RELATED_USER_ID	INTEGER	4	뷰가 접근하는 객체의 소유자 식별자
RELATED_OBJECT_NAME	VARCHAR	40	뷰가 접근하는 객체의 이름
RELATED_OBJECT_TYPE	INTEGER	4	뷰가 접근하는 객체의 타입 (TABLE, PSM, VIEW)

SYS_XA_HEURISTIC_TRANS_

Column name	Type	Length	Description
FORMAT_ID	BIGINT	8	전역 트랜잭션의 트랜잭션 식별자를 지정하는 형식
GLOBAL_TX_ID	VARCHAR	64	전역 트랜잭션의 트랜잭션 식별자로 사용되는 값 중 하나
BRANCH_QUALIFIER	VARCHAR	64	전역 트랜잭션의 트랜잭션 식별자로 사용되는 값 중 하나
STATUS	INTEGER	4	전역 트랜잭션의 상태

변경

메타 테이블에 대한 조회 및 변경은 **DML**을 통하여 수행할 수 있다. 다만 시스템에서 정의된 시스템 사용자(사용자 이름: **SYSTEM_**)만이 메타 테이블을 변경할 수 있다. 그러나 메타 테이블 정보가 변경되면 시스템에 치명적인 손상이 발생할 수 있기 때문에 가급적 변경하지 말아야 한다.

데이터베이스 삭제

데이터베이스 파일을 삭제할 경우에는 **destroydb** 명령을 이용한다. 이 명령은 데이터베이스 파일은 물론 **loganchor** 및 로그 파일들을 모두 삭제한다. 데이터베이스 삭제는 반드시 알티베이스를 종료 시킨 후 수행하여야 한다. **destroydb**의 사용법은 아래와 같다.

destroydb -n dbname [-d home_dir] [-f property_file]

-n dbname 삭제할 데이터베이스 이름을 입력한다.

-d home_dir 이 옵션은 알티베이스의 홈 디렉터리 경로를 명시하여 주는 것이다. 데이터베이스 파일 경로는 이 디렉터리 경로에다 **dbname**을 합쳐 결정한다. 이 옵션 값을 입력하지 않으면 환경변수 **ALTIBASE_HOME**에 명시된 알티베이스 홈 디렉터리 경로를 프로퍼티 파일에 명시된 데이터베이스 디렉터리 명을 합친 것 값으로 삭제할 데이터베이스의 파일 경로를 결정한다.

-f property_file 프로퍼티 파일을 명시적으로 알려 주는 것이다. 만일 이 값을 입력하지 않으면 **destroydb**는 알티베이스 홈 디렉터리 안의 **conf** 디렉터리(**\$ALTIBASE_HOME/conf**)에 있는 **altibase.properties**를 프로퍼티 파일로 결정한다.

예 제

프로퍼티 파일에서 **DB_NAME = mydb**, **DB0_DIR = ?/dbs**, **DB1_DIR = ?/dbs**, **LOG_DIR = ?/logs**로 되어 있는 상황에서 아래와 같이 **destroydb**를 실행하면,

destroydb -n mydb

\$ALTIBASE_HOME/dbs 경로의 **mydb**라는 데이터베이스 파일이 삭제 되고, **\$ALTIBASE_HOME/logs** 경로의 **loganchor** 파일과 로그 파일들이 삭제된다.

```
Dos prompt> destroydb -n mydb
DestroyDB: Release 3.2.0 - Production on Feb 27 2003
16:07:12
(c) Copyright 2001 ALTIBase Corporation. All rights
reserved.

Ready for Destroying Database ? (y/N)y
[Ok] /user5/charlie/work/altibase_home/dbs/mydb-0-0 Exist.
[Ok] /user5/charlie/work/altibase_home/dbs/mydb-1-0 Exist.
[Ok] /user5/charlie/work/altibase_home/logs/loganchor
Exist.
SUCCESS:
Database Destruction End.
Dos prompt>
```

3. 알티베이스 실행과 종료

알티베이스의 실행

알티베이스 서버를 실행하는 방법은 **dbadmin**을 통해서 실행하는 것이다. 직접 알티베이스 파일을 실행시킬 수는 없다.

관리자 도구인 **dbadmin**을 통해 알티베이스 서버를 실행하면 알티베이스 서버 정보 및 관리 측면에서 여러 가지 이점이 있다. 이 방법을 간단하게 소개하면, **dbadmin**을 실행한 상태에서 먼저 **connect** 명령을 수행하고 난 다음, **startup** 명령을 수행하면 된다(아래 화면 참고). 또한, 공유 메모리와 서버 정보에 대한 자세한 내용은 [제 8장 관리도구](#)에서 설명하였다.

알티베이스 서버가 실행되는 과정을 살펴보면, 먼저 프로퍼티 로딩과 시스템 메모리 검사를 거친 후, 알티베이스 시스템 환경 초기화, 시스템 데이터 초기화, 시그널 핸들링, **DB** 공간의 메모리 초기화, 질의 처리 모듈 초기화, 마지막으로 쓰레드들을 초기화 함으로서 알티베이스 서버의 구동이 완료된다.

```
Dos prompt> dbadmin
dbadmin Utility Ver 3.2.0
Copyright 1999-2000, ALTIBase Corporation or its
subsidiaries.
All rights reserved.

=====
dbadmin(ALTIBASE ADMINISTRATION TOOL) HELP screen
=====
# General Commands
-----
CONNECT          - Connect to dbadmin
DISCONNECT       - Disconnect from dbadmin
HELP             - This screen
SHELL            - Execution of Shell
![shell command] - Execution of Shell Command ex) !ls
EXIT             - QUIT alias
QUIT            - Quit from dbadmin console
-----
# Management Commands : opt => options
-----
STARTUP          - Startup altibase server
STATUS [opt]     - Examine of Altibase status
                  - SESSION [opt] : View Session Status
                  - PROPERTY      : View All Property
                  - REPLICATION   : View Replication Information
                  - DB [opt]      : View Memory Usages.
                  - MEMORY       : View Internal Memory Usages.
                  - ALL          : View All Status
ex)
  * Status          - Overview Status
  * Status All      - View All Status
  * Status Session  - Overview Session Status
  * Status Session All - View All Session Status
  * Status Session Num - View Session Status of the
Number
  * Status DB       - View Database Informations
  * Status DB All   - View All Table Informations
  * Status DB Tbl-Name - View Certain Table
Informations
  TERMINATE Session-Number - Terminate a Session
Specified by the Number
```

```

SHUTDOWN [Normal | Immediate | Abort] - Shutdown altibase
server
DTX      [opt]      - Manipulate Distributed Transactions.
                  - SHOW          : View All Distributed Transactions.
                  - ROLLBACK [tid] : Rollback Specified Distributed
Transaction.
                  - COMMIT  [tid] : Commit Specified Distributed
Transaction.
=====

```

알티베이스 서버가 정상적으로 실행되면 아래와 같은 메시지가 화면에 출력된다.

```

Admin> connect
Connected Internally.
Admin> startup
Trying Connect to Altibase.. Connected with Altibase.
=== SYSTEM MEMORY CONTROL ===
PageSize = 8192 byte MaxMem = 8192M FreeMem = 377M
SwapSize = 10631M FreeSwap = 5708M
=== SYSTEM HANDLE CONTROL ===
[SUCCESS] MAX HANDLE IS VALID
Max Handle = 4096 and Max Thread = 100
[PREPARE] Setup Altibase Env...[SUCCESS]
[PREPARE] Signal Processing...[SUCCESS]
[SM-PREPARE] Database Object Preparing...[SUCCESS]
[SM-PREPARE] Log Manager Initialize...[SUCCESS]
[SM-PREPARE] Transaction Manager Initialize...[SUCCESS]
[SM-PREPARE] Index Pool Manager Initialize...[SUCCESS]
[SM-PREPARE] Lock Manager Initialize...[SUCCESS]
[SM-PREPARE] Memory Manager Initialize...[SUCCESS]
[SM-PREPARE] Dirty Page Manager Initialize...[SUCCESS]
[SM-PREPARE] Recovery Phase - 1 : Prepare
Database...[SUCCESS]
[SM-PREPARE] Recovery Phase - 2 : Recovery Skip & Start
Thread...[SUCCESS]
[SM-PREPARE] Recovery Phase - 3 :
Checkpointing...[CHECKPOINT-BEGIN]
[CHECKPOINT-step0] write begin_chkpt log<0,
703516>(recoveryLSN=<0, 703516>)
[CHECKPOINT-step1] Flush Dirty page - Phase1
                  [DirtyPageCount=0]
[CHECKPOINT-step2] Flush Dirty page - Phase2
                  [DirtyPageCount=1]
[CHECKPOINT-step3] sync Database File
[CHECKPOINT-step4] write end_chkpt log<0, 703592>
[CHECKPOINT-step5] Sync Log File
[CHECKPOINT-step6] Update and Flush Log Anchor
[CHECKPOINT-step7] Remove Archive Log File[0 ~ 0]
[CHECKPOINT-summary] TargetDB = 1, BeginChkptLSN =
<0,703516> EndChkptLSN = <0,703592> RecoveryLSN = <0,703516>
[CHECKPOINT-END]
[SUCCESS]
[SM-PREPARE] Loading Database : Dynamic Memory Version
=> Serial Bulk Loading
=> . is 8192k : *.....[SUCCESS]
[SM-PREPARE] Index Manager Initialize...[SUCCESS]
[SM-PREPARE] CheckPoint Manager Initialize...[SUCCESS]
[SM-PREPARE] Database
Refining.....[SUCCESS]
[SM-PREPARE] Index Rebuilding [Total Index
Count:35]****.....[SUCCESS]
[SM-PREPARE] Garbage Collector Manager
Initialize...[SUCCESS]
[SM-PREPARE] Delete Manager Initialize...[SUCCESS]
<< Summary >>

```

```

Database Name -- mydb
Database Type -- Dynamic Memory Version
Maximum Page -- 131072(4096M)
Allocated Page -- 3200(100M)
[PREPARE] Query Preprocessor Unit
Init(TB,QC,TC)...[SUCCESS]
[PREPARE] Replication Manager Init... [SUCCESS]
[PREPARE] Replication Manager Start... [SUCCESS]
[PREPARE] Read Replication Information... [SUCCESS]
[PREPARE] Initializing XA service... [SUCCESS]
[PREPARE] Thread Startup... [SUCCESS]
[PREPARE] Initializing Modules...[SUCCESS]
[PREPARE] Starting Modules...[SUCCESS]

--- STARTUP SUCCESS & Service Available now ---
altibase startup Success!! PID[4125]
Admin> exit
Good Bye!!

```

또는 \$ALTIBASE_HOME/bin 밑에 서버 스크립트 명령을 이용하여 서버를 구동할 수 있다.

```

Dos prompt> server start
Admin> Connected Internally.
Admin> Trying Connect to Altibase.. Connected with Altibase.
=== SYSTEM MEMORY CONTROL ===
PageSize = 8192 byte MaxMem = 8192M FreeMem = 5954M
SwapSize = 10837M FreeSwap = 10548M

```

... 중략

```

--- STARTUP SUCCESS & Service Available now ---
altibase startup Success!! PID[16769]
Admin> Good Bye!!

```


알티베이스의 종료

shutdown은 알티베이스 서버를 종료 시키는 명령이다. 이 명령은 세 가지 옵션이 있는데, 각 옵션에 따라 서버를 종료하는 방식이 다르다. 각각의 옵션은 다음과 같다.

normal

서버를 정상적으로 종료하는 방식으로, 클라이언트들이 모두 종료될 때까지 서버의 종료 작업을 대기하는 방법이다. 서버가 종료하면서 수행하는 일들은 클라이언트-서버간 통신 세션을 감지하는 쓰레드의 종료, 서비스 쓰레드의 종료, 자료저장 관리자의 종료, 그리고 알티베이스 서버 프로세스가 완전히 종료되기를 대기하는 일들을 수행함으로써, 알티베이스 서버를 종료한다.

immediate

서버를 종료할 때 현재 연결된 세션들을 강제로 단절시킨 다음, 알티베이스 서버가 현재 실행 중인 트랜잭션들을 롤백(rollback) 시키고 알티베이스 서버를 종료하는 방법이다.

abort

알티베이스 서버를 '**kill -9**' 시스템 명령을 사용하여 강제로 죽이는 방법이다. 이 방법으로 알티베이스 서버를 종료하면, 데이터베이스가 완전하지 못하여 다음에 알티베이스 서버를 실행할 때 데이터베이스 복구 과정을 거쳐야 한다.

알티베이스를 정상적인 방법(명령: **shutdown normal/immediate**)으로 종료하면 다음과 같은 종료 과정을 나타내는 메시지가 화면에 출력된다.

```
Admin> shutdown normal
Ok..Shutdown Proceeding....
[NORMAL SHUTDOWN]
[PREPARE] Stop Modules...[SUCCESS]
[PREPARE] Finalize Modules...[SUCCESS]
[PREPARE] Destroying Task Threads....[SUCCESS]
[PREPARE] Shutdown Replication...
[PREPARE] Replication Manager Shutdown... [SUCCESS]
[PREPARE] Replication Manager Destroy... [SUCCESS]
[SUCCESS]
[PREPARE] Destroy QP modules...
[SUCCESS]
[PREPARE] Finalize SM modules...
[SM-PREPARE] Garbage Collector Manager Shutdown...[SUCCESS]
[SM-PREPARE] Delete Manager Shutdown...[SUCCESS]
[SM-PREPARE] CheckPoint Manager Shutdown...[SUCCESS]
[SM-PREPARE] Garbage Collector Manager Destroy...[SUCCESS]
[SM-PREPARE] Delete Manager Destroy...[SUCCESS]
[SM-PREPARE] CheckPoint Manager Destroy...[SUCCESS]
[SM-PREPARE] Index Storage Destroy...[SUCCESS]
[SM-PREPARE] Index Manager Destroy...[SUCCESS]
[SM-PREPARE] TimeStamp Manager Destroy...[SUCCESS]
[SM-PREPARE] Log Manager Destroy...[SUCCESS]
[SM-PREPARE] Lock Manager Destroy...[SUCCESS]
[SM-PREPARE] Index Pool Manager Destroy...[SUCCESS]
[SM-PREPARE] Transaction Manager Destroy...[SUCCESS]
```

```
[SM-PREPARE] Memory Manager Destroy...[SUCCESS]
[SM-PREPARE] Dirty Page Manager Destroy...[SUCCESS]
[SUCCESS]
[PREPARE] Destroy Thread Manager...
[PREPARE] Thread Manager Destroy...[SUCCESS]
[PREPARE] Destroy Communication Channel...[SUCCESS]
[PREPARE] unlock Duplicated Checker File...[SUCCESS]
[===== SHUTDOWN ALTIBASE =====]
```

```
Waiting For Altibase Process[15347] To Die Forever...
altibase Shutdown Success!!
Admin>
```

또는 \$ALTIBASE_HOME/bin 밑에 서버 스크립트 명령을 이용하여 서버를 종료할 수 있다.

```
Dos prompt> server stop
Admin> Connected with Altibase.
Admin> Ok..Shutdown Proceeding....
[IMMEDIATE SHUTDOWN]
[PREPARE] Stop Modules...[SUCCESS]
```

... 중략

```
Waiting For Altibase Process[16341] To Die Forever..
altibase Shutdown Success!!
Admin> Good Bye!!
```

```
Admin> shutdown abort
altibase process killed..
Admin> exit
Good Bye!!
```

또는 \$ALTIBASE_HOME/bin 밑에 서버 스크립트 명령을 이용하여 서버를 강제 종료할 수 있다.

```
Dos prompt> server kill
Admin> Connected with Altibase.
Admin> altibase process killed..
Admin> Good Bye!!
```

* 기타 서버 스크립트 명령

```
Dos prompt> server status: 알티베이스 상태 확인
Dos prompt> server status session [all/session_number]:
알티베이스에 연결된 세션에 관한 정보 출력
Dos prompt> server status db [all/table_name]: DB(또는
테이블)에 할당된 데이터 페이지에 관한 정보 및 테이블 별 메모리 효율성
출력
Dos prompt> server status memory: 메모리 객체별 메모리 사용량
출력
Dos prompt> server status replication: 리플리케이션 상태 출력
Dos prompt> server status all: 위의 모든 정보를 출력
```

알티베이스 상태 모니터링 (세션 모니터링, 테이블 모니터링, 메모리 사용량 모니터링, 이중화 모니터링)에 대한 자세한 설명은 *ALTIBASE Administrator's Guide* 참고

4. 알티베이스 프로퍼티

환경 설정 방법

사용자는 알티베이스 서버를 다양한 모드로 운영할 수 있다. 알티베이스 서버의 환경 설정은 알티베이스 프로퍼티 파일을 이용하는 것이다. 프로퍼티 파일은 알티베이스 서버의 운용 방식과 튜닝에 관한 모든 구성 요소를 포함하고 있다.

알티베이스 서버와 관련된 환경을 설정하기 위한 방법은 두 가지가 있다. 첫째, 위에서 설명한 것처럼 알티베이스 프로퍼티 파일을 이용하는 방식이다. 이 방식은 알티베이스 서버가 실행되지 않은 상태에서 할 수 있는 정적인 환경 설정 방법으로서, 프로퍼티 파일에서 해당 구성 요소를 특정 값으로 설정한 후 알티베이스 서버를 재가동하여야만 수정된 값이 알티베이스 서버에 반영된다.

둘째, 알티베이스 서버가 가동 중이더라도 알티베이스 관련 환경 설정을 변경할 수 있는 동적인 방식이다. 알티베이스 서버를 내리지 않고도 변경할 수 있다는 장점이 있으나, 모든 프로퍼티에 적용되지는 않는다. **Alter system** 혹은 **alter session** 명령을 이용하여 알티베이스 서버 전체 혹은 세션 별로 환경 설정 값이 적용되도록 할 수 있다.

알티베이스 서버의 환경 설정에 관한 프로퍼티 파일은 **conf** 디렉터리 밑의 **altibase.properties**이며, 프로퍼티의 내용은 크게 다음과 같이 분류할 수 있다.

- 데이터베이스 구성에 관한 프로퍼티
- 성능에 관한 프로퍼티
- 세션 연결에 관한 프로퍼티
- 트랜잭션에 관한 프로퍼티
- 로깅에 관한 프로퍼티
- 데이터베이스 이중화에 관한 프로퍼티
- 메시지 로그 프로퍼티

데이터베이스 구성 프로퍼티

DB0_DIR, DB1_DIR

데이터베이스 파일이 존재할 경로를 지정한다.

알티베이스는 동시에 2벌의 데이터 파일을 유지하기 때문에 각각 저장될 디렉터리를 별도로 명시할 수 있다. 현재 기본 디렉터리는 **\$ALTIBASE_HOME/dbs** 경로로 동일하게 지정되어 있다.

DB_FILE_SIZE

데이터가 저장되는 데이터베이스 파일이 나뉘어지는 단위로서 일반적으로 파일 시스템이 **2G**이상 지원하지 않는 것이 디폴트이므로 **2G**보다 작게 설정하는 것이 백업이나 다른 알티베이스 관련 작업시 편리하다.

DB_NAME

데이터베이스 이름을 지정하며 임의의 이름일 수 있다. 데이터베이스 이름은 데이터베이스 파일을 생성할 때 데이터베이스 파일의 이름으로 사용된다.

LOG_DIR

로그 파일이 존재할 경로를 지정한다. 보통 **\$ALTIBASE_HOME/logs**가 기본 값으로 정의된다.

LOG_FILE_SIZE

로그가 계속 쌓이면 알티베이스는 여러 개의 로그 파일을 자동으로 생성하여 기록하는데, 각 로그 파일의 크기를 지정하는 것이다.

MAX_CLIENT

알티베이스에 접속할 수 있는 최대 클라이언트의 개수를 명시한다.

MAX_DB_SIZE

알티베이스 서비스 과정 중에 동적으로 늘어날 수 있는 데이터베이스의 최대 크기를 명시한다. 기본 값인 0을 가질 경우 32 비트와 64 비트 모드에 관계없이 **4G**로 설정된다.

MAX_DB_SIZE를 벗어난 크기로 데이터베이스 크기가 확장될 경우 그 트랜잭션은 에러 처리되며 이후 수행되는 **SELECT**문을 제외한 모든 **SQL**문은 에러로 간주된다.

MAX_LISTEN

클라이언트와 알티베이스 간의 통신시 **TCP/IP**, **UNIX DOMAIN** 사용하는 경우 **listen queue**의 크기를 지정하는 값이다.

MAX_THREAD

알티베이스 서버가 생성할 스레드 개수를 정의한다. 즉, 이 값은 서버에 동시에 연결할 수 있는 클라이언트 수로서, **MAX_CLIENT**와 동일한 의미이다.

PERS_PAGE_CHUNK_COUNT

영구 데이터 페이지를 한번에 할당하는 개수로서 기본값은 3200 이다. 즉, 명시된 값이 $128 \text{ page (4M)} * 25 = 3200$ 이므로 100 M 씩 만들어진다.

SHM_DB_KEY

데이터베이스를 가상 메모리 공간에 사용할 경우 0으로 설정하고, 공유 메모리 공간에 사용할 경우는 공유 메모리 키 값을 지정해야 한다. 공유 메모리 키 값은 시스템에서 사용되지 않는 임의의 값이면 된다.

공유 메모리 상에 데이터베이스를 두는 경우 알티베이스 서버의 재가동 단계에서 디스크로부터 페이지를 읽는 과정이 불필요하기 때문에 알티베이스 서버 가동 시간을 줄일 수 있다.(공유 메모리 상에 데이터베이스가 존재할 경우에 한한다.)

STARTUP_SHM_CHUNK_SIZE

SHM_DB_KEY의 값이 지정된 상태에서 알티베이스 구동시 생성되는 공유메모리 조각의 최대 크기를 명시한다.

TEMP_PAGE_CHUNK_COUNT

임시 데이터 페이지를 한번에 할당하는 개수로서 기본값은 128 이다. 즉, 명시된 크기만큼 만들어진다.

THREAD_POOL

Altibase 3.3.x 부터는 알티베이스 최초 구동 시에 이 프로퍼티에 명시된 값 만큼의 쓰레드가 시스템으로부터 생성되고 이 쓰레드들의 서비스 종료시에 해당 쓰레드 풀로 들어가 명시된 개수의 **Thread Pool**을 유지한다. 만일 명시된 개수를 넘어서는 요구가 들어올 경우 시스템으로부터 쓰레드를 할당받아 서비스를 수행한다.

기본값은 0 으로 설정되어 있으며, 이 경우는 매번 쓰레드의 생성 및 소멸이 발생하는 구조이다. 이 값이 0 보다 크게 설정되면 해당 값 만큼의 쓰레드가 대기 상태에 있게 된다.

TRC_DIR

메시지 로그 파일이 존재할 경로를 지정한다.

서버 구동 및 종료 등에 대한 시스템 정보가 기록되어 있는 **altibase_boot.log** 파일과 서버 관리 프로그램에서 사용하는 내부용 파일인 **altibase.lock** 이 있다. **\$ALTIBASE_HOME/trc** 가 기본 값으로 정의된다.

성능 관련 프로퍼티

AGER_WAIT_MAXIMUM, AGER_WAIT_MINIMUM

Ager 관련 스레드들이 **ager sleep** 시 시스템 호출인 **sleep**의 과도한 사용으로 인해 (특히, HP 시스템) 발생하는 서버의 성능 저하를 막기 위하여 이 값을 이용하여 서버 운영 중에 **ager sleep time**을 적절히 조절할 수 있도록 한다. **AGER_WAIT_MAXIMUM**의 기본값은 100000(ms), **AGER_WAIT_MINIMUM**의 기본값은 100(ms) 이다.

DATABASE_IO_TYPE

알티베이스는 메모리 상주형 데이터베이스 시스템이므로 데이터베이스와 관련된 디스크 I/O 작업은 최초 알티베이스 서버 가동시의 데이터 로딩 때와 알티베이스 운용 중 체크포인트가 발생하는 경우이다.

데이터베이스 파일과 관련하여 디스크 I/O를 수행하는 경우 **Direct I/O**와 **Buffered I/O** 두 가지 방법을 제공한다. **Direct I/O**를 사용하기 위해서는 이 프로퍼티의 값을 1로 설정하며 그렇지 않은 경우 0으로 설정한다. 기본값은 0이다.

Direct I/O는 디스크 I/O가 발생하는 동안 CPU 점유율을 줄인다는 장점이 있다. **Buffered I/O**는 **read-ahead, asynchronous write** 기법을 사용하므로 디스크 I/O 요구가 있을 때마다 실제로 디스크에 접근하지 않을 수도 있으므로 응용 프로그램 수준에서 볼 때 디스크 I/O가 훨씬 빠르다는 장점이 있다.

DDL_LOCK_TIMEOUT

DDL 문을 수행할 때 해당 테이블에 이미 다른 트랜잭션에 의해 잠금이 획득되어 있는 경우 잠금을 대기하는 옵션을 설정하는 것이다. 잠금을 요구하여 곧바로 획득되지 않을 경우 이 프로퍼티의 값이 -1로 설정되어 있으면 무한정 대기하고 양수인 경우 지정된 초(sec) 만큼 대기하고 다시 시도한다.

기본값은 0초로서, DDL 수행시 잠금을 요구한 시점에서 잠금을 획득할 수 없는 경우 해당 DDL은 즉시 에러 처리된다.

LOCK_ESCALATION_MEMORY_SIZE

다량의 업데이트 배치 작업을 할 때, **versioning**으로 인해 메모리가 크게 증가할 수 있으므로, 업데이트되는 메모리 크기가 명시한 값 이상이 되면 **versioning**을 하지 않고 **inplace update**¹를 함으로써 메모리가 크게 증가하는 것을 막는 프로퍼티이다. 기본값은 10(MB) 이다.

Versioning 기법을 사용하는 업데이트시 레코드 레벨의 X lock을 획득하고 테이블 레벨의 IX lock을 획득하지만 **inplace update**시

¹ inplace update란 update 대상이 되는 원래 레코드에 대해 해당 칼럼의 값이 수정되는 것을 의미한다.

테이블 레벨의 **X lock** 즉, 배타적 **lock**를 획득한다. 따라서, 이 값을 너무 작게 설정하면 해당 테이블에 대한 **scalability**가 떨어질 수 있으므로 유의해야 한다.

PARALLEL_LOAD_FACTOR

알티베이스 서버 가동시 **Database Refining** 혹은 **index rebuilding** 과정에서 생성되는 **Database Refining thread/index build thread** 개수를 조절하는 프로퍼티이다. 기본값은 0이며 0이나 1의 값이 설정되어 있으면 시스템의 **CPU** 개수 만큼 쓰레드가 생성되며 그 외의 값(**N**)에 대해서는 **N*2**개 만큼 병렬 작업 **thread**가 생성된다.

REFINE_PAGE_COUNT

알티베이스 서버 가동시 수행하는 단계 중 **Database Refining** 단계가 있다. 알티베이스 서버가 이전 종료할 당시, 트랜잭션들이 생성한 **versioning record**들이 **Garbage Collector**에 의해 처리되지 못해서 불필요한 레코드들이 데이터베이스에 존재하고 있을 수 있으며 또한 서버 가동시의 회복 과정에서 생성된 **versioning record**들이 존재할 수 있는데 이들 레코드들을 사용 가능하도록 처리하는 것이 **Database Refining** 단계이다.

Database Refining 대상이 되는 레코드들이 많을 경우 시간을 많이 소모할 수 있기 때문에 이 작업을 여러 쓰레드에 의해 병렬로 수행할 수 있는데 이 때 각 쓰레드가 처리하는 페이지 양을 지정할 수 있다. 기본값은 50개이다.

RESTORE_CHUNK_PAGE_SIZE

알티베이스 서버 가동시 디스크에 있는 데이터베이스 파일을 메모리 상으로 로딩하는데 이 경우 페이지 하나당 한번의 디스크 **I/O**를 수행한다면 많은 디스크 **I/O**로 인해 데이터 로딩 속도가 저하될 것이다.

따라서, 데이터 로딩 시 데이터 로딩 단위를 이 프로퍼티를 이용하여 지정할 수 있으며 기본값은 256으로서 단위는 페이지이다.

RESTORE_METHOD_

알티베이스 서버 가동시 디스크에 있는 데이터베이스 파일을 메모리 상으로 로딩 시 다음과 같은 방법으로 로딩하는 방식을 지정한다.

기본값은 0으로서 하나의 쓰레드가 여러 개의 테이블을 디스크에서 메모리로 로딩하는 방식이고 값을 1로 지정한 경우는 여러 개의 쓰레드가 각각의 테이블을 병렬로 로딩하는 방식이다.

UPDATE_IN_PLACE

알티베이스는 **update** 수행 시 **versioning** 기법이나 **inplace update**를 사용하는데 이 프로퍼티를 이용하여 **update** 방식을 지정할 수 있다. 0으로 설정한 경우 **versioning** 기법을 사용하는데 이는 **memory**의 증가와 **logging**의 양이 늘어날 수 있다. 반면에 **select** 수행 시 해당 버전을 바로 읽기 때문에 **update** 된 **row**에

대해 바로 읽기를 수행할 수 있다. 1로 설정한 경우 **inplace update** 방식을 이용하여 **update**를 수행한다. **Versioning** 기법을 이용한 방식 보다는 **memory**를 적게 쓰지만 해당 테이블에 **X Lock**을 잡고 수행하기 때문에 그 테이블에 대한 다른 트랜잭션 수행 시 **lock waiting**이 발생한다. 기본값은 0이다.

세션 연결 프로퍼티

알티베이스는 클라이언트-서버 구조로 사용 가능하며 세션 연결 프로퍼티는 클라이언트와 서버의 통신에 관한 프로퍼티를 규정하는 것이다. 다음과 같은 프로퍼티들이 있다.

Common 프로퍼티

CM_BUF_SIZE

클라이언트와 서버 간의 통신을 위해 세션 당 할당될 공유 메모리의 통신 버퍼 크기를 결정하는 것이다. 이 값은 알티베이스의 저장 관리자가 유지하는 한 개의 데이터 페이지의 2배 크기를 최적으로 추천하며, 일반적으로 **64K bytes**이다.

CM_DISCONN_DETECT_TIME

클라이언트와 서버의 연결이 단절되었는지를 검사하기 위해 알티베이스 서버에는 세션 관리 쓰레드(**cm detector**)가 존재한다. 세션 관리 쓰레드의 동작 주기를 설정하기 위한 프로퍼티로서 디폴트로 **3초**이다.

일반적으로 클라이언트 프로세스가 비정상 종료하면 그 클라이언트와 연결된 세션은 곧바로 그 상태를 감지할 수 있다.

그러나, 그 세션에서 수행 중인 작업이 세션 작업과는 무관한 알티베이스 서버 내부의 작업이고 또한 그 작업이 오랜 시간을 요구한다면 클라이언트와 연결이 종료되었는지의 여부를 해당 세션에서는 확인할 수 없기 때문에 클라이언트가 비정상 종료되었음에도 불구하고 알티베이스 서버는 그 작업을 계속 진행하게 된다.

위와 같은 경우 그러한 세션을 감지하여 해당 트랜잭션들을 롤백시킬 필요가 있으며 이를 위해 세션 관리 쓰레드가 주기적으로 세션들의 상태를 검사하게 된다.

CM_DISCONN_HIGHWATER_MARK

세션 관리 쓰레드가 동일한 클라이언트의 비정상 종료를 몇 회 이상 감지했을 때, 쓰레드의 작업을 취소시킬 것인지에 대해 결정하는 값으로 기본값은 **3(회)**이다.

ERROR_MSG_NLS

에러 메시지 언어를 설정하는 프로퍼티 이다. 이 프로퍼티가 설정되어 있지 않으면 기본값으로 **NLS_USE** 프로퍼티에 설정되어 있는 값을 따른다. 알티베이스는 현재 에러 메시지 언어로 한글과 영어만을 지원하기 때문에 **NLS_USE**가 **US7ASCII** 또는 **KO16KSC5601** 캐릭터 셋 이외인 경우 반드시 설정해야 한다.

IPC_CHANNEL_BUF_COUNT

클라이언트와 서버간 **IPC** 통신을 하기 위하여 반드시 설정되어야 하는 프로퍼티로, 하나의 **IPC** 채널이 사용할 수 있는 통신 버퍼의 개수를 의미한다.

채널당 사용되는 버퍼의 개수는 현재 **1**로 고정되어 있다.

IPC_CHANNEL_COUNT

클라이언트와 서버간 **IPC** 통신을 하기 위하여 반드시 설정되어야 하는 프로퍼티로, **IPC**를 이용한 클라이언트와 서버 통신 채널의 최대 개수를 지정한다. (각 채널 수에 비례해서 공유 메모리와 세마포를 할당받기 때문에 서버와 동시에 연결될 최대 **IPC** 연결 개수를 설정한다.)

IPC_CHANNEL_RETRY_COUNT

THREAD가 **IPC operation**을 끝낸 후 바로 **sleep**에 들어가지 않고 **THREAD**가 **CPU**를 **yield**하지 않도록 명시된 값만큼 **retry**를 하는 횟수를 지정한다.

NLS_USE

알티베이스가 지원하는 다국어 시스템은 현재 한글, 영어, 대만어, 중국어, 유니코드이다. 한글 시스템의 경우에는 **KO16KSC5601**로 지정하고 영문 시스템일 경우에는 **US7ASCII**로, 대만어, 중국어, 유니코드인 경우는 각각 **ZHT16BIG5**, **ZHS16CGB231280**와 **UTF8**로 지정해 주어야 한다. 데이터베이스에 한글을 사용할 경우에는 반드시 **KO16KSC5601**로 지정해 주어야 한다.

PORT_NO

TCP/IP로 클라이언트와 서버가 통신할 때 사용하는 포트 번호이다. 사용자는 루트영역(일반적으로 **1023**번까지)을 제외한 나머지 영역(최대 **65535**)에 대해 다른 프로그램에서 사용하지 않는 번호면 임의로 지정할 수 있다. 알티베이스 응용 프로그램은 이 포트 번호를 사용하여 서버와 연결하는데, **\$ALTIBASE_HOME/conf/altibase.properties**의 **PORT_NO**를 참조한다. 클라이언트가 서버와 다른 컴퓨터에 있는 경우에도 위의 경로를 설정해 주어야 한다.

세션별 타임아웃 관련 프로퍼티

FETCH_TIMEOUT

응용 프로그램 내에서 **SELECT** 문을 수행하는 시간이 길어짐에 따라 데이터베이스 메모리가 비정상적으로 증가하는 것을 막기 위하여 이 값을 설정한다. 질의 수행 시간이 프로퍼티 파일에 설정된 값보다 커지면 세션 연결을 해제하고 현재 트랜잭션을 철회한다. 기본값은 **0**이다.

프로퍼티 파일에 명시된 값은 시스템 제어문 또는 작업 제어문을 이용하여 변경할 수 있다.

IDLE_TIMEOUT

서버에 접속된 클라이언트가 비정상적으로 오랜 시간 연결을 맺고 있고, 만약 이러한 클라이언트의 수가 점차적으로 많아진다면 결국에는 서비스를 할 수 있는 연결 개수가 현저히 작아져, 나중에는 서비스가 불가능한 상황이 올 수 있다.

이러한 현상을 미리 방지하기 위해 이 값을 설정한다. 한 세션의 유휴 시간이 프로퍼티 파일에 설정된 값보다 커지면 세션 연결을 해제하고 현재 트랜잭션을 철회한다. 기본값은 0 이다.

프로퍼티 파일에 명시된 값은 시스템 제어문 또는 작업 제어문을 이용하여 변경할 수 있다.

QUERY_TIMEOUT

어떤 특정 질의들을 (sorting, 긴 join 등) 수행하는 시간이 길어짐에 따라 데이터베이스 크기가 비정상적으로 증가하는 것을 막기 위하여 이 값을 설정한다. 질의 수행 시간이 프로퍼티 파일에 설정된 값보다 커지면 현재 트랜잭션 연산을 자동적으로 부분 철회되게 한다. 기본값은 0이다.

프로퍼티 파일에 명시된 값은 시스템 제어문 또는 작업 제어문을 이용하여 변경할 수 있다.

UTRANS_TIMEOUT

변경 연산(update, insert, delete)을 수행하는 트랜잭션의 수행 시간이 길어짐에 따라 로그 파일의 개수가 비정상적으로 증가하는 것을 막기 위하여 이 값을 설정한다. 수행 시간이 프로퍼티 파일에 설정된 값보다 커지면 세션 연결을 해제하고 현재 트랜잭션을 철회한다. 기본값은 0이다.

프로퍼티 파일에 명시된 값은 시스템 제어문 또는 작업 제어문을 이용하여 변경할 수 있다.

트랜잭션 프로퍼티

AUTO_COMMIT

세션(session)에서 하나의 SQL 문을 수행할 때마다 이 하나의 SQL 문을 하나의 트랜잭션으로 처리할 것인지를 결정하는 프로퍼티이다. 이 값이 1 이면 **auto-commit**이고 0 이면 **non-autocommit**이다. **non-autocommit**의 경우에는 응용 프로그램에서 트랜잭션의 시작과 끝을 명시적으로 알려주어야 한다.

이와 같은 일은 세션 단위로 정해지는데, 만일 **AUTO_COMMIT = 1** 로 서버가 구동되었다 하더라도 세션 별로 이 모드를 변경할 수 있다. 예를 들어, 클라이언트에서 **ALTER SESSION SET AUTOCOMMIT = FALSE** 라는 SQL 문장을 실행하면 이 세션은 이후로 트랜잭션을 반영할 것인지 또는 롤백할 것인지를 명시해 주어야 한다.

ISOLATION_LEVEL

트랜잭션의 격리도를 설정하는 프로퍼티로서 아래 표와 같다.

Isolation Level	특징
0 (Committed Read)	검색 트랜잭션이 읽은 데이터는 완료한 트랜잭션에 의해 변경된 데이터임을 보장 기본 모드
1 (Repeatable Read)	한 트랜잭션 수행 동안 동일 레코드를 여러 번 반복해서 읽는 경우 항상 동일한 레코드의 내용을 검색하게 됨을 보장
2 (No Phantom)	Phantom 현상 방지

TRANSACTION_DURABILITY_LEVEL

트랜잭션의 지속성(**durability**)이란 어떠한 시스템 고장에도 불구하고 완료(롤백 혹은 커밋)한 트랜잭션이 작업한 내용은 데이터베이스에 영구적으로 반영되어 있어야 하는 트랜잭션의 성질이다.

알티베이스는 다음과 같은 트랜잭션 지속성을 지원한다.

Durability	특징
0	알티베이스가 온라인 중인 경우에만 지속성 보장 알티베이스가 재가동되는 경우 모든 데이터베이스는 없어짐
3	모든 시스템 고장에 대해 지속성 보장 기본 모드임

반드시 필요한 경우를 제외하고는 이 프로퍼티의 값을 변경해서는 안된다.

TRANSACTION_TABLE_SIZE

알티베이스 서비스 과정 중에 동시에 생성될 수 있는 최대 트랜잭션 개수로서 기본값은 1024이며, 이에 대한 메모리가 미리 할당되는 프로퍼티이다.

로깅 프로퍼티

데이터베이스가 변경될 때 변경 로그를 유지하는데, 이에 관한 처리를 어떻게 할 것인지를 정하는 프로퍼티들이 있다.

ARCHIVE_DIR

Archive log backup을 사용하는 경우 아카이브 로그 파일들이 백업될 디렉토리를 설정하는 프로퍼티이다. 사용자가 명시적으로 이 값을 지정하지 않으면 디폴트로 알티베이스가 설치된 디렉토리 밑의 **arch_logs** 디렉토리에 아카이브 로그 파일들이 백업된다.

사용자가 이 값을 명시적으로 지정할 수 있으나, 지정된 디렉토리는 미리 생성되어 있어야 하며 그렇지 않은 경우 에러 메시지를 출력함과 동시에 알티베이스 서버가 가동되지 않는다.

ARCHIVE_FULL_ACTION

ARCHIVE_DIR에 설정된 디렉토리가 속한 파일시스템에 충분한 디스크 공간이 없는 경우 **archive log backup**을 수행하는 **archive thread**의 **action**을 제어할 수 있는 프로퍼티이다.

ARCHIVE_FULL_ACTION의 값이 0인 경우 **archive thread**는 에러 메시지를 출력한 후, 아카이브 로그 파일을 백업하는 작업을 중지하게 된다. 이후 충분한 디스크 공간이 확보된 후에도 사용자가 명시적으로 아카이브 로그 백업을 활성화하는 명령을 입력하지 않는 한 아카이브 로그 백업은 수행되지 않는다. 이 경우 체크 포인트가 발생하면 아카이브 로그 파일이 백업되지 않았더라도 불필요한 로그 파일들은 삭제되기 때문에 운영시 주의가 필요하다.

ARCHIVE_FULL_ACTION의 값이 1인 경우 **archive thread**는 충분한 디스크 공간이 확보되어 아카이브 로그 파일을 백업할 수 있을 때까지 기다린다. 이 기간 동안 체크포인트가 발생한다면 아카이브 로그 파일이 백업되지 않아서 로그 파일들이 삭제될 수 없으므로 주의해야 한다.

ARCHIVE_THREAD_ENABLED

아카이브 로그 파일을 주기적으로 백업하는 스레드인 아카이브 스레드를 활성화시킬 지의 여부를 지정하는 프로퍼티로서, 1인 경우 아카이브 스레드를 활성화시키게 되며 디폴트 값은 1이다.

이 프로퍼티는 아카이브 로그 파일 백업을 위한 디렉토리에 충분한 디스크 공간이 없어서 아카이브 스레드가 비활성화된 경우, 이후에 디스크 공간을 확보하여 아카이브 스레드를 다시 활성화하고자 할 때 사용할 수 있다.

알티베이스 가동 중 다음과 같은 명령을 이용하여 이 프로퍼티의 값을 변경할 수 있다.

```
alter system set archive_thread_enabled = 1(또는 0)
```

AUTO_REMOVE_ARCHIVE_LOG

Archive log backup을 사용할 지의 여부를 지정할 수 있는 프로퍼티이며, 이 값이 1인 경우 archive log backup을 지원하지 않으며 0인 경우 archive log backup을 지원하도록 동작한다.

이 프로퍼티의 값이 1로 설정된 경우 archive log backup을 위한 archive thread가 생성되어 archive log file들이 백업되며, 그렇지 않은 경우 시스템 고장으로 인한 장애 시 복구에 필요한 로그 파일들을 제외한 모든 로그 파일들은 체크포인트가 수행되면 자동으로 삭제된다.

CHECK_POINT_ENABLED

체크포인트를 on 또는 off 시키는 프로퍼티이며 이 값을 0으로 지정하면 체크포인트 스레드가 동작하지 않으며, 사용자도 임의로 체크포인트를 수행할 수 없다. (0: off, 1: on) 기본값은 1이다.

CHECK_POINT_INTERVAL_IN_LOG

체크포인트 주기를 로그 파일이 생성되는 횟수로 정하는 것이다. 즉, 정해진 횟수 만큼 로그 파일이 교체되면 체크포인트를 자동으로 수행한다.

이 프로퍼티 값에 의해 체크포인트 요구가 발생할 당시 이미 체크포인트가 진행 중이거나 기타 다른 이유로 인하여 체크포인트가 수행되지 못하는 경우가 발생할 수 있다. 이 경우 이미 진행 중인 체크포인트가 끝난 후 바로 체크포인트를 수행하는 것이 아니라 현재의 체크포인트 요구는 바로 취소된다. 따라서, 최대 다음 체크포인트 요구는 이 프로퍼티에 설정된 값만큼 로그 파일이 새로 생기는 시점이 될 수 있다.

CHECK_POINT_INTERVAL_IN_SEC

체크 포인트 주기를 초 단위 시간으로 정하는 것이다.

LOGGING_LEVEL

트랜잭션 회복 및 재시작 회복을 위하여 로그가 반드시 필요하며 트랜잭션마다 로그를 생성해야 하기 때문에 시스템 성능이나 scalability에 많은 영향을 미치게 된다.

애플리케이션 특성에 따라 매번 로그를 생성하지 않아도 된다면 시스템 성능을 향상시킬 수 있기 때문에 여러 단계의 로깅 레벨을 지원한다.

Logging Level	특징
0	로그를 전혀 생성하지 않음 트랜잭션 롤백시 알티베이스 비정상 종료 재시작 회복 기능 사용할 수 없음
1	DML문에 대해서만 로그 생성 Transaction Consistent Checkpoint
2	모든 SQL문에 대해서 로그 생성 Fuzzy Checkpoint

	기본 모드
--	-------

로깅 레벨은 회복 기능에 중요한 영향을 미치기 때문에 기본 모드인 2를 유지하는 것이 좋으며 로깅 레벨 변경시 신중한 주의를 필요로 한다.

OPEN_LOG_FILE_COUNT

재시작 회복 성능을 향상시키기 위한 프로퍼티이며, 재시작 회복 동안 참조되는 로그 파일들을 명시된 개수만큼 **open**해 둬으로써 로그 파일을 접근할 때 마다 **open/close** 하는 비용을 감소시킨다. 기본값은 3이다.

PREPARE_LOG_FILE_COUNT

로그 생성시 해당 로그파일에 충분한 공간이 없으면 새로운 로그 파일을 생성하며, 이 경우 트랜잭션의 응답시간은 늦어지게 된다. 이처럼 로그파일 생성으로 인해 트랜잭션의 수행이 늦어지는 것을 막기 위해 알티베이스는 여분의 로그파일을 미리 생성하며 이 여분의 로그파일의 개수를 지정하는 것이 이 프로퍼티이다. 기본값은 2이다.

데이터베이스 이중화 프로퍼티

다음 속성값들은 데이터베이스의 이중화 기능을 위한 값들이다.
데이터베이스 이중화에 대한 자세한 내용은 [제 9장 데이터베이스 이중화](#)와 *ALTIBASE Replication User's Manual* 을 참조할 것.

REPLICATION_CONNECT_TIMEOUT

서버 구동시 이중화를 수행하기 위해 대상 호스트에 대한 이중화 연결 수행시 설정된 시간 값 이상 응답이 없을 경우 연결을 더 이상 시도하지 않고 진행한다. 서버가 구동 이후에는 이중화 송신 쓰레드가 별도로 연결을 다시 시도한다. 기본값은 225이다.

REPLICATION_ENABLE_ADD_COLUMN

이중화 대상 테이블에 컬럼 추가를 허용할지를 결정한다. 1일 경우 컬럼 추가가 가능하며 기본값은 0이다.

REPLICATION_GET_ACK

송신 쓰레드가 XLog 전송 후 수신 쓰레드로 부터 이에 대한 **acknowledge**를 받을지에 관한 유무를 지정한다. HP <-> SUN 과 같은 이기종 OS 간에 **sync** 방식 이중화 시 이 프로퍼티의 값이 0이면 100% **sync**를 보장 받지 못할 수 있다. 그러나 이 프로퍼티의 값을 1로 설정하면 100% **sync**가 보장 되지만 0인 경우보다 약 10%의 성능 저하가 예상된다. (시스템에 따라 차이가 있을 수 있음.) 기본값은 1(**acknowledge** 받음)이다.

REPLICATION_HBT_DETECT_HIGHWATER_MARK

Connection 응답을 하지 않는 경우 몇 회 이후에 장애로 판단할 것인지 결정한다. 기본값은 10이다. 따라서, 임의의 호스트의 장애를 판단하는 최대 시간은

REPLICATION_HBT_DETECT_TIME * REPLICATION_HBT_DETECT_HIGHWATER_MARK 이다.

HeartBeat 쓰레드는 기본 값인 30초 동안 연결이 되지 않을 경우 장애로 판단한다.

REPLICATION_HBT_DETECT_TIME

HeartBeat 쓰레드¹의 검사 주기를 설정한다. 기본값은 3초이고, 이 주기마다 **HeartBeat** 쓰레드는 해당 호스트에 대한 장애를 검사한다.

REPLICAION_LOCK_TIMEOUT

이중화 데드락이 발생하는 경우 **receiver** 쪽에서는 무한정 잠금을 기다리게 되어 서비스가 중단될 수 있으므로 이러한 경우를 방지하기 위해 잠금을 획득하기 위해 기다리는 최대 시간을

¹ HeartBeat 쓰레드: 알티베이스의 리플리케이션에서는 송신 쓰레드와 수신 쓰레드 간에 데이터 통신 수행 시 물리적인 장애가 발생했을 때 가능한 한 신속하게 장애를 검출하기 위하여 HeartBeat Thread 를 만들어서 상대 호스트의 상태를 주기적으로 검사하도록 하는 기법을 도입했다.

지정할 필요가 있다. **REPLICATION_LOCK_TIMEOUT**이 이러한 용도로 사용된다.

receiver에서 해당 작업을 수행하기 위해 잠금을 요구하게 되는데 최대 이 프로퍼티에 설정된 시간만큼만 잠금을 획득하기 위해 기다린다.

주어진 시간 내에 잠금을 획득하지 못하여 **REPLICATION_LOCK_TIMEOUT**이 발생하는 경우 해당 작업은 롤백된다.

REPLICATION_MAX_LOGFILE

체크포인트 수행 과정에서 로그 파일 삭제시 이중화에 필요한 로그 파일들은 삭제 대상이 되지 않는다. 따라서 네트워크 장애 등과 같은 원인으로 인해 이중화 데이터 전송이 느려지면 로그 파일들이 삭제되지 않고 계속 쌓이게 됨으로써 **disk full**이 발생하게 된다.

이러한 경우를 막기 위하여 비록 이중화 데이터가 모두 전송되지 않았다 하더라도 현재 생성되고 있는 로그 파일과 이중화에서 필요로 하는 최소 로그 파일의 개수의 차이가 어느 값 이상이 되면 이미 생성되어 있는 일부 로그 화일들을 삭제할 필요가 있다. 이를 설정해주는 프로퍼티가 **REPLICATION_MAX_LOGFILE**이다.

예를 들어, 이 값을 100으로 설정하고 체크포인트시 삭제할 로그 파일을 구분하는 과정에서 현재 사용 중인 마지막 로그 파일 번호가 250이고 이중화에서 필요한 최소 로그 파일의 번호가 50이고 시스템에 존재하는 최소 로그 파일의 번호가 45라면 45번 로그 파일부터 249번 로그 파일이 모두 삭제되게 된다.

아직 전송되지 않은 데이터가 많이 남아있는데도 불구하고 이 프로퍼티에 의해 로그 파일이 삭제되는 이러한 상황을 이중화 **give-up**이 발생했다라고 하며 이중화 **give-up**이 발생하면 전송되지 않은 데이터들을 모두 삭제하는 것이기 때문에 이중화 데이터 불일치가 발생할 수 있다. 따라서, 이 값은 알티베이스 운용 환경을 고려하여 적절하게 설정되어야 한다.

REPLICATION_PORT_NO

지역 서버에서 이중화 연결을 할 때 지역 서버의 이중화 포트 번호이다.

REPLICATION_PROPAGATION

이중화 **propagation** 기능을 사용할지를 설정하는 프로퍼티로서 1로 설정하면 이중화 **propagation** 기능을 사용한다.

알티베이스 서버 A, B, C가 있고 A->B, B->C로 이중화가 걸려있는 경우 일반 이중화 운영 모드에서는 A 서버에서 변경(Insert/Update/Delete)된 작업들이 B 서버에만 반영되며 C 서버로는 반영되지 않는 **point-to-point** 방식과 달리 이 프로퍼티

값을 1로 설정하면 A에서의 모든 변경 작업은 B 서버를 거쳐 C 서버까지 반영하게 된다.

REPLICATION_RECEIVE_TIMEOUT

송신 쓰레드와 수신 쓰레드에서 공통적으로 사용하는 프로퍼티로서 **sender/receiver**로부터 어떤 메시지를 기다리는 최대 시간을 지정하는 프로퍼티로서 기본값은 300초이다.

송신 쓰레드에서는 상대방 수신 쓰레드로부터 **ACK**를 기다리는 최대 시간이며 이 시간이 경과하게 되면

REPLICATION_SENDER_SLEEP_TIMEOUT에서 지정된 시간만큼 **sleep**한 후 다시 상대방 수신 쓰레드와의 접속을 시도한다. 이 경우 기존 사용하던 소켓은 닫고 새로운 소켓을 생성하여 재연결을 시도한다.

수신 쓰레드에서는 상대방 송신 쓰레드로부터 어떤 메시지를 기다리는 최대 시간이다. 이 시간이 경과하게 되면 수신 쓰레드는 자동 종료되며 이후 상대방 송신 쓰레드가 어떤 메시지를 다시 보내게 되는 경우 다시 생성된 기본값은 300초 이다.

REPLICATION_SENDER_AUTO_START

알티베이스는 서버 재 구동 시 종료를 시키지 않은 이중화의 송신 쓰레드가 존재하면 자동으로 띄우도록 되어 있다. 이 값을 0으로 설정함으로 써 송신 쓰레드를 자동으로 띄우지 않게 할 수 있다. 기본값은 1이다.

REPLICATION_SENDER_SLEEP_TIMEOUT

리플리케이션 송신 쓰레드가 예러 상황에서 **sleep**을 해야 할 때 얼마나 길게 할 것인지 결정한다. 기본값은 60초이다.

REPLICATION_SYNC_LOCK_TIMEOUT

이중화 **Sync** 수행시 이중화가 걸린 테이블에 이미 다른 트랜잭션에 의해 잠금이 획득되어 있는 경우 잠금을 대기하는 옵션을 설정하는 프로퍼티이다. 잠금을 요구하여 곧바로 획득되지 않을 경우 프로퍼티에 지정된 초(**sec**) 만큼 잠금을 대기한다. 기본값은 30초로서, 해당시간 동안 잠금을 획득할 수 없는 경우 **Sync**는 에러 처리된다.

REPLICATION_SYNC_LOG

이중화 수행 시 송신 쓰레드가 보내는 로그는 디스크에 내려 갔는지의 여부에 관계없이 메모리 상의 로그를 보내기 때문에 **system** 또는 **media failure** 같은 상황에서 데이터 불일치 등의 문제가 발생할 소지가 있다. 이러한 문제를 해결하기 위해서 이 값을 1로 설정하면 송신 쓰레드는 디스크에 내려간 로그만 보낼 수 있도록 한다. 기본값은 0이다.

REPLICATION_SYNC_MAX_LIMIT

Sync 방식으로 동작하는 도중 **async**로 전환되는 최대 한계값을 지정한다. **Sync** 방식으로 동작하는 도중에 현재 **LSN**과 메타 테이블의 **LSN**이 명시된 값보다 큰 경우 **async** 방식으로

동작한다. **async** 방식으로 동작하는 도중에 현재 **LSN**과 메타테이블의 **LSN**이 명시된 값보다 작아지게 되면 다시 **sync** 방식으로 동작한다. 기본값은 **3**(단위는 로그파일 개수)이다.

REPLICATION_UPDATE_REPLACE

이중화 작업중 변경작업 충돌(update conflict)시 변경된 내용의 반영을 결정한다. **0** 이면 충돌이 있을 경우 반영하지 않고 에러 처리하며, **1** 일 경우 충돌을 무시하고 반영한다.

메시지 로그 프로퍼티

MSGLOG_FILE_SIZE

알티베이스의 동작 상태를 기록하는 메시지 로그 파일(**altibase_boot.log**)의 크기를 설정한다. 기본값은 10M이다.

MSGLOG_MAX_FILE

메시지 로그 파일에 기록되는 메시지 양이 많아서 메시지 로그 파일의 크기를 넘을 경우 메시지 로그 백업 파일로 저장된 후 다시 처음부터 기록된다. 이 때, 백업될 메시지 로그 파일(**altibase_boot.log**)의 최대 개수를 지정할 수 있는 프로퍼티이다.

쿼리 최적화 프로퍼티

OPTIMIZER_MODE

이 값이 0으로 설정되면 질의문을 최적화하기 위하여 **cost_based optimization**을 지원하고, 1이면 **rule_based optimization**을 지원한다. 기본값은 0이다.

NORMALFORM_MAXIMUM

정규식(Normal Form) 노드 크기의 최대 개수를 지정할 수 있는 프로퍼티이다.

SELECT 질의문의 **WHERE**절에 존재하는 **predicate**들이 논리 연산자(**AND, OR**)로 복잡하게 연결되어 있을 때, 알티베이스는 **table**을 더 빠르게 탐색하기 위하여 **predicate**을 정규화 시킨다.

정규화 방법으로 **CNF**(conjunctive normal form)와 **DNF**(disjunctive normal form)가 있으며, 둘 중 어떤 **normal form**으로 변경되든지 **NORMALFORM_MAXIMUM**에 명시된 노드 크기를 초과하는 경우에 그 해당 **normal form**으로의 정규화를 시도하지 않는다.

만약, 두 **normal form** 모두 **NORMALFORM_MAXIMUM**을 초과하면 질의를 수행하기 위해서는 **normal form** 노드를 너무 많이 생성한다는 에러 메시지를 출력 후 질의 수행을 바로 종료한다.

이 규칙은 **On Condition** 조인의 **on predicate**에도 동일하게 적용된다.

인덱스 타입 프로퍼티

INDEX_BUILD_THREAD_COUNT

실행시간(run-time)시 index rebuilding 수행 과정에서 생성되는 index build thread 개수를 조절한다. 기본값은 0이며 0으로 설정된 경우 시스템의 CPU 개수 만큼 index rebuild thread가 생성된다.

INDEX_TYPE

인덱스를 생성할 때 인덱스 타입을 명시하지 않은 경우에 어떤 인덱스를 생성할 지를 지정한다. 이 값이 0으로 설정된 경우 TTREE가 생성되며, 1인 경우 BTREE로 생성된다. 기본값은 1이다.

기타 프로퍼티

ADMIN_MODE

ADMIN_MODE를 1(ON)로 설정하면 관리자 모드로 활성화되어 SYS 또는 SYSTEM_ 사용자만이 서버와 연결을 맺어 작업을 할 수 있고 그 외 일반 사용자들은 연결 자체가 실패한다. 이 프로퍼티는 ALTER SYSTEM SET ADMIN_MODE = 정수(0 또는 1)의 명령어로 서버 운영 중에 그 값을 변경할 수 있다. 기본값은 0이다.

* UNIX DOMAIN socket 프로토콜 만을 사용하여 서버에 접속하고자 할 때 이 값을 2로 설정하여 사용할 수 있다. (0 또는 1: IPC, TCP/IP, UNIX Domain 사용 연결 시)

EXEC_DDL_DISABLE

대개의 경우 초기 데이터베이스를 구축한 이후 DDL을 훨씬 더 빈번하게 수행하며 DDL 수행은 기존 데이터베이스 스키마를 변경시키는 작업이므로 상당한 주의를 요한다.

따라서 알티베이스 운영 중 DDL을 수행하지 못하도록 운영자가 설정할 수 있으며 이 프로퍼티의 값을 1로 설정하면 알티베이스 운영 중 DDL을 수행할 수 없으며 그렇지 않은 경우 DDL을 수행할 수 있다.

FIFO_QUEUE

큐의 기본 동작은 먼저 enqueue 된 레코드가 가장 먼저 dequeue 되는 것으로 이 프로퍼티의 값이 1(기본값)인 경우 동작하는 방식이다.

0으로 설정되면 가장 최근에 enqueue 된 레코드가 가장 먼저 dequeue 되는 stack 방식의 큐 동작을 한다.

MAX_QUEUE_COUNT

알티베이스 가동 시 이 프로퍼티에 지정된 값만큼의 condition variable이 초기화되어 생성되는 최대 큐의 개수이다.

알티베이스 큐는 dequeue 수행 시 옵션으로 waiting 기능을 허용한다. Waiting 기능은 dequeue 하고자 하는 레코드가 없을 때 dequeue 조건에 맞는 레코드가 enqueue 될 때까지 기다릴 시간을 지정하는 기능이다. 이 기능을 위해 내부적으로 condition variable이 사용되며, 이는 각 큐 테이블마다 존재한다.

기본값은 10이며 온라인 중에는 변경이 불가능하다.

SELECT_HEADER_DISPLAY

select 대상이 select all (*)로서, 결과 출력 시 iSQL 상에 칼럼 이름만 출력할 것인지, 테이블 이름과 함께 칼럼 이름을 출력할 것인지 설정하는 시스템 프로퍼티이다.

altibase.properties 파일에 명시할 수 있으며 ALTER SYSTEM 명령문으로 변경할 수 있다 (0 또는 1). 기본값은 0이며, 이 경우

iSQL 상에서 결과 출력 시 테이블 이름과 칼럼 이름이 함께 출력된다.

SUPPORT_SELECTIVE_LOADING

MMDBMS는 빠른 데이터 접근을 위해 버퍼 매니저를 유지하지 않는다. 따라서, 데이터베이스 내의 모든 테이블이 메모리에 올라와 있는 상태가 된다. 그러나 데이터베이스 운용 시 몇몇의 테이블을 제외하면, 실제로 사용되는 테이블은 소량이기 때문에 필요 없는 테이블일 경우 디스크로 내린다면, 나머지 메모리 공간을 효율적으로 재사용 할 수 있다. 즉, 이러한 선택적 테이블 로딩을 **selective loading** 이라고 하며, DB 관리 및 운용 시 시스템의 효율적 사용과 관리의 편의성을 극대화한다.

Selective loading을 사용하기 위해서는 이 프로퍼티를 1로 설정하며 그렇지 않은 경우 0으로 설정한다. 기본값은 0이다.

TRCLOG_LEVEL

알티베이스 서버에서 수행되는 SQL문의 정보, 에러 메시지 종류 또는 SQL 문의 수행 시간 등을 **altibase_boot.log** 파일에 기록하게 할 수 있는 프로퍼티로서 다음과 같은 내용을 **altibase_boot.log** 파일에 기록할 수 있다.

Level	설 명
0	altibase_boot.log에 receiver 측에 발생한 conflict message 기록
1	altibase_boot.log에 receiver 측에서 insertXLog 시 SM에서 발생하는 error message 기록
2	server status 결과를 altibase_boot.log에 기록
3	질의 수행 시간 측정
4	altibase_boot.log에 수행한 DML과 where 절의 predicate 분류 상태 기록
5	explain plan 기능 사용 시 where 절의 predicate 분류 상태도 함께 display
6	sequential fetch 시 sequential iterator의 trace 정보 기록
7	lock 설정 시간 기록 (server status db all)
8	DDL 문과 그 실행 성공 여부 기록
9	현재 HeartBeat Thread의 동작 유무를 판단할 수 있다. 또한, HeartBeat Thread에 등록된 모든 호스트에 대한 리스트를 주기적으로 altibase_boot.log에 남긴다.
10	PSM 관련 SQL문(create, procedure, execute procedure 등) 수행 시 오류 메시지를 altibase_boot.log에 남긴다.

위 표에서 레벨(level)은 **trclog_level**을 설정하기 위해 사용되는 비트 번호를 나타낸다.

Trclog_level을 설정하기 위해서는 `alter system set trclog_level = [number]` 구문을 사용할 수 있으며 여기에서 number는 2^{level} 을 의미한다. 기본값은 7이다.

예를 들어 위 표에서 level 0과 3을 설정하기 위해서는 $2^0 + 2^3 = 9$ 이므로 `alter system set trclog_level = 9` 구문을 사용한다.

5. 백업 및 복구

로그(Log)와 복구

데이터베이스의 복구는 시스템에서 발생할 수 있는 다양한 장애(failure)로부터 데이터베이스의 일관성(consistency)을 유지하기 위해 제공되는 방법이다.

장애(Failure)의 종류

트랜잭션 고장

트랜잭션 고장은 내부적 혹은 외부적인 요인에 의해 트랜잭션이 중단됨을 의미한다. 이 경우 트랜잭션의 정상적인 철회(rollback)를 통해 데이터를 복구한다.

시스템 고장

시스템 고장은 데이터베이스가 운영적 시스템의 결함 혹은 정전 등의 장애가 나타나는 경우이다. 시스템 재시동 시에 백업 데이터 파일과 **active** 로그를 통해서 시스템 고장 시점까지의 상태로 데이터베이스를 복구한다.

디스크 고장

디스크 고장의 경우 데이터 백업 파일이 저장된 디스크의 오류로 인해 백업 데이터를 손실 하였을 경우를 의미한다. 이 경우, 이전의 데이터 백업 파일이 있는 경우 이 파일을 통해 최근의 데이터베이스로 복구가 가능하다. 그러나 로그 디스크가 손상된다거나 **archive** 로그를 삭제하였을 경우 가장 최근의 상태로의 복구가 불가능하다.

로그와 백업 데이터 파일

백업 데이터 파일

백업 데이터 파일이란 어떤 시점에서 데이터베이스의 스냅샷(snapshot) 이미지이다. 이 파일은 데이터베이스의 생성(createdb)과 데이터베이스 종료(shutdown), 체크포인트 등에 의해 갱신된다.

로그(Log)

로그는 트랜잭션의 원자성과 지속성을 유지하기 위해 사용된다. 원자성은 트랜잭션의 철회(rollback)를 통해 트랜잭션 수행 이전의 상태로 복귀할 수 있도록 하는 것이고, 지속성은 정상적으로 종료(commit)된 트랜잭션이 다양한 데이터베이스 장애로부터 원래의 내용을 복구할 수 있도록 하는 것이다.

로그는 가지고 있는 내용에 따라 **active** 로그와 **archive** 로그로 구분 할 수 있다. **archive** 로그는 데이터베이스가 어느

시점(체크포인트 등)에 In-memory DB의 내용으로 백업 데이터 파일을 갱신한 경우, 이 시점 이전의 모든 로그를 가리킨다. 데이터베이스 운영 중에 발생한 다양한 갱신 작업은 **active** 로그로 간주한다.

로그는 현재의 데이터베이스 상태를 가지는 매우 중요한 파일로서, 만일 현재 로그 파일이 손상을 입었을 경우 당시 작업의 유무에 관계없이 데이터베이스 전체가 손상을 입게 된다. 기존 로그 파일은 일반적으로 데이터 볼륨이 손상되었을 경우 백업 볼륨과 함께 데이터베이스를 복구하는데 사용된다.

로그앵커(Loganchor)

로그앵커는 데이터 백업 파일과 로그와의 관계를 나타내는 중요한 정보를 포함한다. 이는 로그를 기준으로 한 시점에서의 데이터 백업 파일이 가지고 있는 정보의 시간적 위치를 나타낸다. 이 파일은 백업 데이터 파일과 함께 중요한 백업대상이 된다.

체크포인트

체크포인트(checkpoint)는 주기적으로 데이터베이스의 내용을 백업 데이터 파일에 저장함으로써, 시스템 장애로부터 데이터베이스 복구에 걸리는 시간을 줄이는데 그 의미가 있다.

알티베이스 2.0 부터는 퍼지 체크포인트(fuzzy checkpoint)와 핑퐁 체크포인트(ping pong checkpoint) 를 지원한다.

체크포인트의 수행

체크포인트의 수행은 주기적(시간 혹은 로그)인 수행과 함께 임의의 시간에 사용자에게 의해 체크포인트가 발생할 수 있다.

시간 주기 체크포인트

데이터베이스 운용 중 일정한 시간에 체크포인트를 수행하도록 결정할 수 있다. 이 주기는 데이터베이스 프로퍼티 중 **CHECK_POINT_INTERVAL_IN_SEC**에 의해 결정된다.

로그에 따른 체크포인트

데이터베이스 운영환경에 따라서 일정기간에 주로 데이터베이스의 갱신이 일어날 수 있다. 이 때문에 데이터베이스가 어느 정도 갱신 이벤트가 있을 경우, 이를 체크포인트를 통해서 백업 데이터 파일의 갱신을 유도할 수 있다. 이 값은 **log** 파일이 생성된 횟수를 이용하며, 데이터베이스 프로퍼티 중 **CHECK_POINT_INTERVAL_IN_LOG**를 이용한다.

임의적 체크포인트

사용자가 임의의 시간에 체크포인트를 발생 시킬 수 있다. 이 방법은 **SQL**을 통해 데이터베이스의 체크포인트를 지정하는 방법으로 **"ALTER SYSTEM CHECKPOINT"** 명령을 이용한다.

체크포인트와 Archive 로그

데이터베이스가 미디어 고장을 지원하지 않을 경우가 있을 수 있다. 이는 백업 데이터 파일의 장애가 없음을 보장하는 경우이다.

체크포인트의 발생은 현재 **active** 상태의 로그들의 내용이 백업 데이터 파일에 적용되는 것을 의미하고, 아울러 현재의 **active** 상태의 로그들은 **archive** 상태의 로그로 전환된다. 이 경우 백업 데이터 파일의 손실을 고려하지 않으면, **archive** 로그들은 더 이상 필요가 없게 된다. 이 때 불필요한 **archive** 로그들을 시스템에서 자동으로 제거하도록 설정할 수 있다. 이는 프로퍼티의 **AUTO_REMOVE_ARCHIVE_LOG**를 설정함으로써 가능한데, 이

값을 “1”로 설정할 경우 체크포인트 시점에 백업 데이터 파일에 적용된 로그는 자동으로 삭제된다.

백업 및 복구

데이터베이스의 고장으로부터의 복구는 데이터베이스가 어떤 장애에 의해 실행이 중단되었을 경우, 시스템을 재시동하였을 때 자동으로 복구된다(restart recovery).

restart recovery는 로그앵커로 부터 최근의 데이터베이스와 그에 따른 로그의 정보를 획득한 후, 데이터 백업 파일에 로그의 기록을 적용함으로써 복구를 수행한다.

만일 데이터베이스 백업 파일이 손실 되었을 경우, 다른 디바이스에 백업한 파일을 이용하여 시스템을 최신의 상태로 복구할 수 있다.

*주의) 데이터베이스 로그 파일이 손실되는 것은 고려하지 않는다. 따라서 로그가 저장된 디바이스가 고장이 나는 경우 데이터베이스를 최신의 상태로 유지하지 못할 수 있다.

On-line 백업

서버가 수행 중인 동안 다른 트랜잭션에게 영향을 주지 않고 현재의 **stable**한 데이터베이스를 다른 디스크 공간으로 백업한다.

사용 방법

ALTER SYSTEM BACKUP '*directory_name*';

directory_name: 온라인 백업으로 생성되는 파일을 저장할 디렉터리 경로명이며 512 character를 초과할 수 없다.

예제

```
iSQL> ALTER SYSTEM BACKUP 'backup_dir';
Alter success
```

이 명령의 수행 결과는 가장 최근에 수행된 검사점이 반영된 데이터베이스 디스크(이를 **stableDB**라 한다.)내의 데이터베이스 파일들, recovery에 필요한 로그 파일들(트랜잭션 rollback과 restart recovery에 필요한 로그 파일들), 로그 앵커 파일, ALTER TABLE *table_name* CLOSE 문의 수행으로 생성된 internal database file들이 path로 지정된 위치로 복사된다.

ex) 최근의 stableDB = 1, recovery에 요구되는 로그 파일 = logfile1

파일 종류	시스템에 존재하는 파일들	백업되는 파일들
-------	---------------	----------

데이터베이스 파일	mydb-0-0 mydb-0-1 mydb-0-2	mydb-1-0 mydb-1-1 mydb-1-2	mydb-1-0 mydb-1-1
로그 파일	logfile0 logfile1 logfile2 logfile3		logfile1
로그 앵커 파일	loganchor		Loganchor
내부 데이터베이스 파일	61312.tbl 983072.tbl		61312.tbl 983072.tbl

특징

- 백업할 파일들이 저장될 디렉터리는 백업 전에 미리 생성되어 있어야 한다.
- 백업되는 모든 종류의 파일들은 모두 동일한 디렉토리에 백업된다.
- Commit 되지 않은 데이터들도 백업될 수 있다. 이 경우 **restoredb** 수행시 자동으로 **recovery**를 수행한다.
- 백업된 데이터베이스의 버전 정보를 위해 **backup_ver** 파일이 자동으로 생성되며 **restoredb** 시 사용된다.
- 백업된 파일 중 사용자 실수로 인하여 어느 하나라도 제거되는 경우 백업된 일부 파일들을 이용하여 **restore**를 수행할 수 없다.

Restoredb

온라인 백업 기능을 이용하여 백업된 데이터베이스 파일들을 사용하여 데이터베이스를 이전의 상태로 복구한다.

사용 방법

```
restoredb -s 백업경로명 [-i] [-l 로그파일의 restore경로명]
                    [-d db0파일의 restore경로명] [-o
                    db1파일의 restore 경로명]
```

- **-i(interactive mode)**: 백업된 데이터 복구 작업을 계속 수행할 것 인지에 대해 결정한다.
- **-a**: 이 옵션을 지정한 경우 logs 디렉토리에 있는 logfile 까지 **recovery**를 수행한다.
지정하지 않은 경우는 logs 디렉토리에 logfile 이 있는 경우 오류가 발생한다.

- 백업경로명: 백업된 데이터가 존재하는 디렉터리 이름으로서, 반드시 주어져야 한다.
- 로그파일의 restore 경로명: restore 후 생성되는 로그 파일을 저장할 디렉터리 이름으로서 이 값이 주어지지 않는 경우 프로퍼티 파일에 설정된 디렉터리에 로그 파일이 저장된다.
- db0(db1) 파일의 restore 경로명: 백업된 하나의 데이터베이스 파일을 사용하여 2 개의 데이터베이스를 restore 하며 각 데이터베이스에 대해 별도의 디렉터리 경로명을 지정해 줄 수 있다.
 둘 다 주어진 경우 : 사용자가 지정한 디렉터리에 데이터베이스 파일을 각각 restore
 하나만 주어진 경우 : 사용자가 지정한 하나의 디렉터리에 2 개의 데이터베이스 모두를 restore
 둘 다 주어지지 않는 경우 : 프로퍼티 파일에 설정된 디렉터리에 데이터베이스를 restore

Restore후 생성되는 데이터베이스 파일들

내부 데이터베이스 파일은 db0가 저장되는 디렉터리에 restore된다.

파일 종류	백업된 파일들	Restore후 생성된 파일들
데이터베이스 파일	mydb-1-0 mydb-1-1 mydb-1-2	mydb-1-0 mydb-0-0 mydb-1-1 mydb-0-1 mydb-1-2 mydb-0-2
로그 파일	logfile1	logfile1
로그 앵커 파일	loganchor	loganchor
내부 데이터베이스 파일	61312.tbl 983072.tbl	61312.tbl 983072.tbl

제한 사항

- restoredb 수행 시 서버는 종료된 상태여야 한다.
- restoredb에서 옵션으로 사용된 디렉토리 이름은 restore 수행 전에 미리 생성되어 있어야 한다.
- 백업된 데이터베이스의 버전 정보와 restoredb 수행 당시의 프로퍼티 파일에 설정된 정보는 동일해야 한다.
- restore에 필요한 디스크 공간이 미리 확보되어 있어야 한다.

- **restore** 시에 공유메모리 데이터베이스가 남아있는 경우 이를 제거해야한다.

예제

```
Dos prompt> server stop
Dos prompt> restoredb -s backup_dir

RestoreDB: Release 3.2.0 Production on Apr 3
2003 15:06:09
(c) Copyright 2001 ALTIBase Corporation. All
rights reserved.

Restore DB: ##### Info #####
Source      : backup_dir
Target DB0 Dir:
/user5/charlie/work/altibase_home/dbs
Target DB1 Dir:
/user5/charlie /work/altibase_home/dbs
Target LOG Dir :
/user5/charlie /work/altibase_home/logs

BEGIN:    Database Restore Started..

[step0] check disk space...
          Required Space Info (Log Space =
10485816)
                                (DB Space = 20971520)
                                (Total    = 31457336)
          You must prepare enough space before
proceeding!!
[step1] restore Log Anchor...      [SUCCESS]
[step2] restore Database File...
[SUCCESS] Finish restore DB File[0 ~ 0]
[step3] restore Log File...
[SUCCESS] restore nothing Log File
[step4] restore Internal DB File... [SUCCESS]

SUCCESS:    Database Restore Completed.
```

기타 백업 정책

온라인 백업 외에도 알티베이스에서 제공하는 백업 정책은 다음과 같다.

- 서비스 중지없이 iLoader를 이용한 테이블 별 백업: 백업하기 전에 먼저 백업하고자 하는 테이블의 스키마에 관한 **FORM** 파일을 생성한 후, iLoader의 **out** 옵션을 이용하여 백업 파일을 생성하고, iLoader의 **in** 옵션을 이용하여 복구한다.
- 오프라인 백업: 유닉스 명령어 **cp**를 이용하여 데이터베이스 파일과 로그 파일을 백업 한 후 역시 명령어 **cp**를 이용하여

복구한다. 단, 2개의 데이터베이스 파일이 서로 다른 디렉토리에 있는 경우 모두 백업해야 한다.

- 아카이브 로그 백업: 아카이브 로그 백업은 디스크 장애와 같은 **media failure**(매체 고장)에 대한 복구를 지원하기 위한 방법이다. 백업된 아카이브 로그를 이용하여 데이터베이스를 복구하기 위해서는 과거에 온라인 백업 혹은 오프라인 백업을 통해 받은 데이터베이스 파일과 로그 앵커, 로그 파일이 존재해야 하며 다음과 같은 절차대로 수행한다.
 - 백업되어 있는 로그 파일과 로그 앵커를 백업된 아카이브 로그 파일이 있는 디렉토리로 이동한다.
 - **restoredb** 유틸리티를 이용하여 **restore**를 수행한다.

백업 및 복구에 대한 자세한 설명은 *ALTIBASE Administrator's Guide* 참고

6. 응용프로그램 작성

프로그램 작성 방법

본 장에서는 알티베이스 데이터베이스를 활용하는 응용 프로그램 작성법에 관하여 간략하게 설명하도록 한다. 알티베이스의 응용 프로그램 작성은 클라이언트-서버 구조로 작성하는 방법이 있다. **JDBC, ODBC, SES C/C++** 그리고 **Altibase CLI** 인터페이스를 사용하여 클라이언트 프로그램을 작성하는 방법으로서, 일반적으로 범용 데이터베이스 운용 환경에서의 프로그램 작성법이라 할 수 있다.

본 장에서는 알티베이스 데이터베이스를 활용하는 응용 프로그램 작성법에 관하여 간략하게 설명하도록 한다. 프로그램 작성의 자세한 내용은 *ALTIBASE CLI User's Manual*, *ALTIBASE Embedded SQL User's Manual*을 참고.

Altibase CLI 를 활용한 프로그램

알티베이스의 CLI (Call Level Interface)를 이용하여 클라이언트를 작성하는 방법이다. 알티베이스를 클라이언트-서버 구조로 운영하는 상황에서 C 또는 C++ 언어로 응용 프로그램을 작성할 때 사용할 API이다. 클라이언트 프로그램이 다중 스레드 또는 단일 프로세스 중 어느 형태로 작성되느냐에 따라 링크해야 할 클라이언트 라이브러리가 구분된다.

헤더 파일과 라이브러리

Altibase CLI를 이용하여 프로그램을 작성하기 위해서는 알티베이스의 홈 디렉토리의 서브 디렉토리인 `include`와 `lib`에 있는 헤더 파일과 라이브러리 파일이 필요하다.

```
%ALTIBASE_HOME%\include\sqlcli.h
%ALTIBASE_HOME%\lib\cli_conn.obj
%ALTIBASE_HOME%\lib\id.lib
%ALTIBASE_HOME%\lib\odbccli.lib
```

라이브러리 파일 중에서 클라이언트 환경설정을 위한 라이브러리 `libid.a`를 제공한다.

Makefile

작성된 Altibase CLI 소스를 컴파일하기 위한 Makefile에는 다음과 같은 내용이 포함된다.

```
include $(ALTIBASE_HOME)/install/altibase_env.mk
```

- 이 파일 안에는 컴파일 시 필요로 하는 라이브러리 패스와 라이브러리들을 링크시키기 위한 옵션들이 정의되어 있으며 오브젝트 파일을 만드는 규칙을 제공한다.

<Makefile 예제>

```
include
$(ALTIBASE_HOME)/install/altibase_env.mk

INCLUDES = -I../include -I.
LIBDIRS = -L../lib
SRCS=
OBJS=$(SRCS:.cpp=.o)

%.o: %.cpp
    $(CXX) $(CFLAGS) $(DEFINES) -
    I$(ALTIBASE_HOME)/include/ -o $@ $<
```

```

program_standalone : program.o
CC -L$(ALTIBASE_HOME)/lib/ -
I$(ALTIBASE_HOME)/include/ -o $@ $@.o
$(ALTIBASE_HOME)/lib/cli_conn.o -lsqlcli -lcli
-lcli_sa $(LIBS)

program_mutithread : program.o
CC -L$(ALTIBASE_HOME)/lib/ -
I$(ALTIBASE_HOME)/include/ -o $@ $@.o
$(ALTIBASE_HOME)/lib/cli_conn.o -lsqlcli -lcli
-lcli_mt $(LIBS)

```

다중 쓰레드 프로그램

다중 쓰레드 프로그램 작성시 주의할 점

1. 각 쓰레드 별로 환경 핸들, 연결 핸들을 각각 할당해야 한다.
2. 각 쓰레드에서 DB 관련 오류를 저장하는 데 필요한 저장소를 할당하는 함수 `ideAllocErrorSpace()`를 호출해야 한다. 이 함수는 쓰레드 생성 초기에 호출하면 된다.

프로그램 작성

Altibase CLI를 이용하여 작성하는 프로그램에서 알티베이스 서버에 접속하고 종료하는 방법을 간략하게 설명하도록 한다. 자세한 내용은 *ALTIBASE CLI User's Manual*을 참고

```

/* test_cli.cpp */

#include <sqlcli.h>
#include <stdio.h>
#include <stdlib.h>

#define SQL_LEN 1000
#define MSG_LEN 1024

SQLHENV env; // Environment 를 할당 받을 handle.
SQLHDBC con; // Connection 을 할당 받을 handle.
SQLHSTMT st1; // Statement 를 할당 받을 handle.
SQLHSTMT st2; // Statement 를 할당 받을 handle.
SQLHSTMT st3; // Statement 를 할당 받을 handle.
SQLHSTMT st4; // Statement 를 할당 받을 handle.

void execute_err(SQLHSTMT stat, char* q)
{
    printf("Error : %s\n",q);
}

```

```

SQLINTEGER errNo;
SQLSMALLINT msgLength;
SQLCHAR errMsg[MSG_LEN];

if (SQL_SUCCESS == SQLError ( env, con, stat,
NULL, &errNo, errMsg, MSG_LEN, &msgLength ))
{
    printf(" Error : # %ld, %s\n", errNo,
errMsg);
}

SQLFreeStmt(st1, SQL_DROP);
SQLFreeStmt(st2, SQL_DROP);
SQLFreeStmt(st3, SQL_DROP);
SQLFreeStmt(st4, SQL_DROP);

if (SQL_ERROR == SQLDisconnect(con)) {
    printf("disconnect error\n");
}

SQLFreeConnect(con);
SQLFreeEnv(env);

exit (1);
}

int main(int argc,char **argv)
{
    char    query[SQL_LEN];

    int     PORT_NO, age;
    char    *USERNAME, *PASSWD, *NLS,
connStr[1024], name[22];
    SQLINTEGER    len;

    if ( argc < 2 )
    {
        printf("Usage : %s port_no \n",argv[0]);
        exit(1);
    }

    PORT_NO = atoi(argv[1]);           // 서버의 접속
Port 번호.
    USERNAME = "SYS";                 // 사용자명.
    PASSWD = "MANAGER";               // 사용자 암호.
    NLS = "US7ASCII";                // 사용언어 ( KOI6KSC5601,
US7ASCII )

    /* Altibase Server 에 접속 */

    /* Environment 를 위한 메모리를 할당 */
    if (SQL_ERROR == SQLAllocEnv(&env))

```

```

    {
        printf("SQLAllocEnv error!!\n");
        exit(1);
    }

/* Connection 을 위한 메모리를 할당 */
if (SQL_ERROR == SQLAllocConnect(env, &con))
{
    printf("SQLAllocConnect error!!\n");
    SQLFreeEnv(env);
    exit(1);
}

sprintf(connStr,
"DSN=127.0.0.1;UID=%s;PWD=%s;CONNTYPE=%d;NLS_US
E=%s;PORT_NO=%d", USERNAME, PASSWD, 1, NLS,
PORT_NO);

/* Connection 형성 */
if (SQL_ERROR == SQLDriverConnect( con, NULL,
connStr, SQL_NTS, NULL, 0, NULL,
SQL_DRIVER_NOPROMPT ))
{
    printf("connection error\n");
    SQLFreeEnv(env);
    exit(1);
}

/* Statement 를 위한 메모리를 할당 */
if (SQL_ERROR == SQLAllocStmt(con, &st1))
{
    printf("SQLAllocEnv error!!\n");
    SQLFreeEnv(env);
    exit(1);
}

/* Statement 를 위한 메모리를 할당 */
if (SQL_ERROR == SQLAllocStmt(con, &st2))
{
    printf("SQLAllocEnv error!!\n");
    SQLFreeEnv(env);
    exit(1);
}

/* Statement 를 위한 메모리를 할당 */
if (SQL_ERROR == SQLAllocStmt(con, &st3))
{
    printf("SQLAllocEnv error!!\n");
    SQLFreeEnv(env);
    exit(1);
}

```

```

/* Statement 를 위한 메모리를 할당 */
if (SQL_ERROR == SQLAllocStmt(con, &st4))
{
    printf("SQLAllocEnv error!!\n");
    SQLFreeEnv(env);
    exit(1);
}

/* 쿼리수행 */

/* 쿼리 직접수행 */
sprintf(query,"DROP TABLE TEST001");
SQLExecDirect(st1,query, SQL_NTS);

// sprintf(query,"CREATE TABLE TEST001 ( name
varchar(20), age number(3) )");

sprintf(query,"CREATE TABLE TEST001 ( name
varchar(20), age integer )");

if (SQL_ERROR == SQLExecDirect(st2,query,
SQL_NTS))
{
    execute_err(st2, query);
}

/* Statement 를 준비하고 변수를 바인드한다. */
sprintf(query,"INSERT INTO TEST001
VALUES( ?, ? )");
if (SQL_ERROR == SQLPrepare(st3, query,
SQL_NTS))
{
    execute_err(st3, query);
}

if (SQL_ERROR == SQLBindParameter(st3, 1,
SQL_PARAM_INPUT, SQL_C_CHAR, SQL_CHAR,
sizeof(name), 0, name, 0, &len)) {
    printf("SQLBindParameter error!!! ==> %s
\n",query);
    exit(1);
}

if (SQL_ERROR == SQLBindParameter(st3, 2,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&age, 0, &len)) {
    printf("SQLBindParameter error!!! ==> %s
\n",query);
    exit(1);
}

/* 준비된 Statement 를 수행 */

```

```

    sprintf(name, "김민석");

    len = SQL_NTS;
    // sprintf(name, "0183194088");
    age = 28;
    if (SQL_ERROR == SQLExecute(st3))
    {
        execute_err(st3, query);
    }

    sprintf(name, "홍길동");

    // sprintf(name, "0012904321");
    age = 25;
    if (SQL_ERROR == SQLExecute(st3))
    {
        execute_err(st3, query);
    }

    sprintf(name, "아무개");

    // sprintf(name, "011985601234");
    age = 34;
    if (SQL_ERROR == SQLExecute(st3))
    {
        execute_err(st3, query);
    }

    sprintf(query, "SELECT * FROM TEST001");
    if (SQL_ERROR == SQLExecDirect(st4, query,
SQL_NTS))
    {
        execute_err(st4, query);
    }

    /* Select 의 결과값을 변수에 저장 */
    if (SQL_ERROR == SQLBindCol(st4, 1,
SQL_C_CHAR, name, sizeof(name), &len))
    {
        printf("SQLBindCol error!!!\n");
        exit(1);
    }

    if (SQL_ERROR == SQLBindCol(st4, 2,
SQL_C_SLONG, &age, 0, &len))
    {
        printf("SQLBindCol error!!!\n");
        exit(1);
    }

```

```

        while (SQLFetch(st4) == SQL_SUCCESS) //
결과값이 있는 동안 결과값을 받아 화면에 출력 */
        {
            int i;
            printf("Name : ");
            //          for (i = 0; i < 15; i++)
            //          printf("%02x ", (int)(unsigned
            char)(name[i]));

            printf("Name : %5s,
Age : %5ld\n",name,age);

        //          printf(", Age : %5ld\n",age);
        }

/* 모든 handle 을 해제하고 접속을 종료 */

SQLFreeStmt(st1, SQL_DROP);
SQLFreeStmt(st2, SQL_DROP);
SQLFreeStmt(st3, SQL_DROP);
SQLFreeStmt(st4, SQL_DROP);

if (SQL_ERROR == SQLDisconnect(con)) {
    printf("disconnect error\n");
}

SQLFreeConnect(con);
SQLFreeEnv(env);
}

```

실행결과

```

Dos prompt> make test_cli
Dos prompt> test_cli 20300 <- port_no
Name : 김민석, Age : 28
Name : 홍길동, Age : 25
Name : 아무개, Age : 34
Dos prompt>

```

JDBC 를 활용한 프로그램

알티베이스의 JDBC 드라이버를 이용하여 클라이언트를 작성하는 방법이다.

JDBC 드라이버

알티베이스는 기본적으로 %ALTIBASE_HOME%\lib 디렉토리 안에 Altibase.jar 파일을 JDBC 드라이버로 제공한다.

알티베이스 서버와 연결을 설정하기 위해, 먼저 드라이버를 로드하고, url과의 연결을 시도한다

단계 1) JDBC 드라이버를 로드할 때에는 프로그램 내에서 다음과 같은 코드로 JDBC 드라이버를 등록하여 사용한다.

```
Class.forName("Altibase.jdbc.driver.AltibaseDriver")
```

단계 2) 알티베이스의 url을 제공하고, url과 연결을 시도 하기 위한 일반적인 방법으로 다음과 같다. (Altibase에 로그인 하기 위한 id가 "SYS"이고, 패스워드가 "MANAGER"인 경우)

(JDBC url

"jdbc:Altibase://hostname:portnum/databasename")

(Connection

```
DriverManager.getConnection(url, "user", "password"))
```

```
String url =
    "jdbc:Altibase://127.0.0.1:20300/myddb";

Connection con =
    DriverManager.getConnection(url, "SYS",
    "MANAGER");
```

현재 JDBC 2.0 API 와 Standard Extension API 는 일부 지원한다.

Class Path

컴파일을 위한 class path는 다음과 같다.

```
classpath
%CLASSPATH%;%ALTIBASE_HOME%\lib\Altibase.jar
```

예를 들어, Altibase.jar를 이용하기 위해 환경변수에 다음과 같이 CLASSPATH를 설정한다. (compiler, tools, runtimes, and APIs 등을 제공하는 java1.2가 많은 사용자들이 이용 가능한 c:\ 밑에 설치 되었다고 가정)


```

set JAVA_HOME=c:\java1.2
export
CLASSPATH=%JAVA_HOME%\lib;%ALTIBASE_HOME%\lib\A
ltibase.jar;%CLASSPATH%

```

프로그램 작성

JDBC를 이용하여 작성하는 프로그램에서 알티베이스 서버에 접속하고 종료하는 방법은 대략 다음과 같다.

```

/* JdbcTest.java */

import java.util.Properties;
import java.sql.*;

class JdbcTest
{
    public static void main(String args[]) {

        Properties props = new Properties();
        Connection con = null;
        Statement stmt = null;
        PreparedStatement pstmt = null;
        ResultSet res;

        if ( args.length == 0 )
        {
            System.err.println("Usage : java
JdbcTest port_no \n");
            System.exit(1);
        }

        String port = args[0];
        String url = "jdbc:Altibase://127.0.0.1:"
+ port + "/mydb";
        String user = "SYS";
        String passwd = "MANAGER";
        String enc = "US7ASCII";

        props.put("user", user);
        props.put("password", passwd);
        props.put("encoding", enc);

        /* 알티베이스 JDBC 드라이버 등록 */
        try {

            Class.forName("Altibase.jdbc.driver.AltibaseDri
ver");
        } catch ( Exception e ) {

```

```

        System.err.println("Can't register
Altibase Driver");
        return;
    }

    /* 접속하고 Statment 를 할당. */
    try {
        con =
DriverManager.getConnection(url,props);
        stmt = con.createStatement();
    } catch ( Exception e ) {
        e.printStackTrace();
    }

    /* 쿼리수행 */

    try {
        stmt.execute("DROP TABLE TEST001");
    } catch ( SQLException se ) { }

    try {
        stmt.execute("CREATE TABLE TEST001
( name varchar(20), age number(3) )");

        pstmt = con.prepareStatement("INSERT
INTO TEST001 VALUES(?,?)");

        pstmt.setString(1,"김민석");
        pstmt.setInt(2,28);
        pstmt.execute();

        pstmt.setString(1,"홍길동");
        pstmt.setInt(2,25);
        pstmt.execute();

        pstmt.setString(1,"아무개");
        pstmt.setInt(2,34);
        pstmt.execute();

        res = stmt.executeQuery("SELECT *
FROM TEST001");

    /* 결과를 받아 화면에 출력 */
        while(res.next()) {
            System.out.println(" Name :
"+res.getString(1)+", Age : "+res.getInt(2));
        }

    /* 연결 해제 */
        stmt.close();
        pstmt.close();

```

```
        con.close();  
    } catch ( Exception e ) {  
        e.printStackTrace();  
    }  
}  
}
```

실행결과

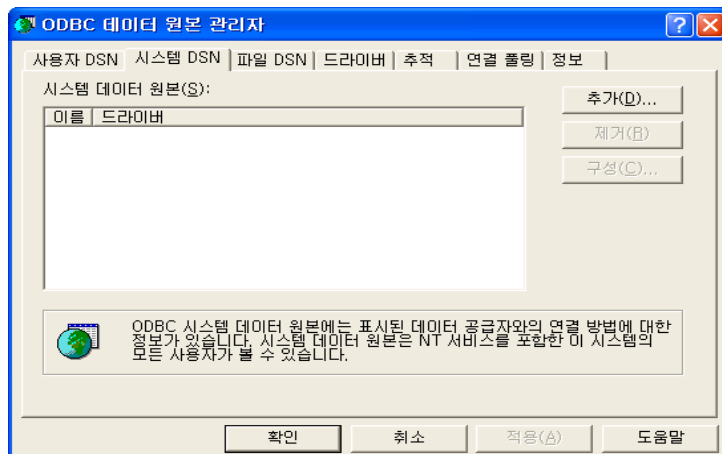
```
Dos prompt> make JdbcTest  
Dos prompt> java JdbcTest 20300 <- port  
Name : 김민석, Age : 28  
Name : 홍길동, Age : 25  
Name : 아무개, Age : 34
```

ODBC 를 활용한 프로그램

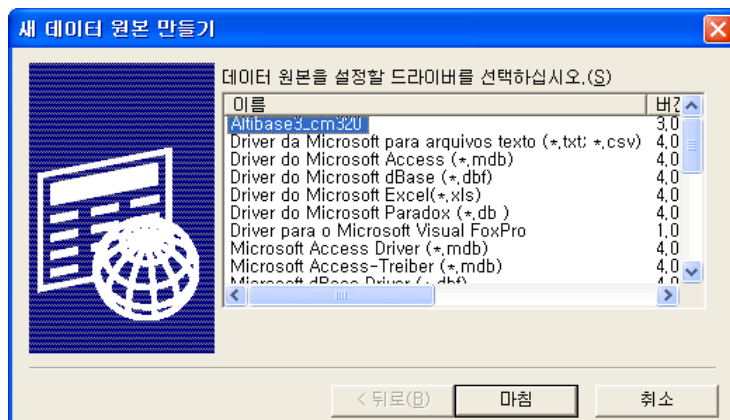
알티베이스의 ODBC 드라이버를 이용하여 클라이언트를 작성하는 방법이다.

ODBC 드라이버 설치

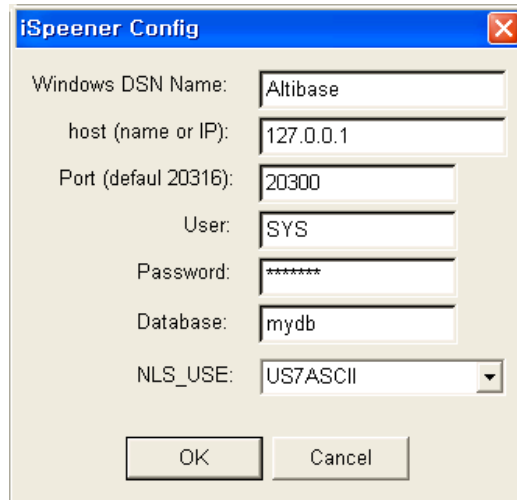
1. 윈도우 상에 Altibase ODBC를 설치하기 위해 www.altibase.com/download/etc/ODBC/ 에서 a3_odbc_3.X.X.exe 파일을 다운로드 한 후 실행하면 Altibase ODBC Driver의 DLL이 ODBC 드라이버 매니저와 윈도우 시스템 레지스트리에 설치된다.
2. 제어판에서 관리도구 메뉴를 선택하고, 데이터 원본 (ODBC)을 선택합니다. 그러면, 다음과 같은 ODBC 데이터 원본 관리자 윈도우가 나타난다. 시스템 DSN을 추가하기 위해 오른쪽에 위치한 추가 버튼을 클릭한다.



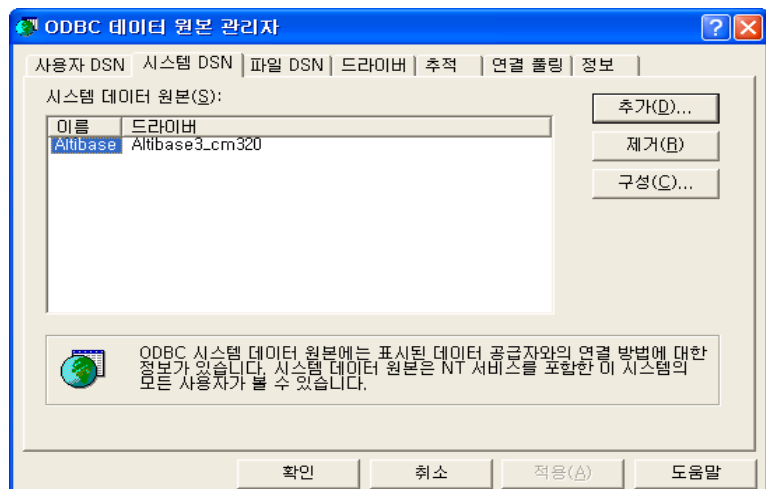
3. 데이터베이스의 원본을 설정할 드라이버 (Altibase3_cm320)를 선택하고 마침 버튼을 클릭한다.



4. Altibase 데이터베이스의 ODBC 세팅이 끝난 뒤, ODBC 데이터 원본 관리자에서 DSN (Data Source Name: Altibase)을 추가하고 DB 연결 시 필요한 정보(host, Port, User, Password)를 입력한다.



5. 이제 Altibase 서버의 ODBC 설정이 끝나고, 다음과 같이 시스템 DSN 탭을 클릭하면 데이터베이스 파일을 Altibase란 이름으로 사용할 수 있게 된다.



* 현재 ODBC Core 와 Level 1 일부분을 지원한다.

6. 위와 같이 ODBC 설정 후 ODBC 설치 폴더 내(예: C:\Program Files\Altibase\Altibase3_ODBC)의 iodbc.exe로 설정이 잘 되었는가를 확인 후 Altibase ODBC 원본 (Data Source)을 테스트하기 위해 SQL 문을 실행함으로써 Altibase ODBC 설정이 완벽하다는 것을 다시 한 번 확인할 수 있다.

e.g.)

```

C:\[InstallDir]> IODBC.exe -S altibase -H
127.0.0.1 -U SYS -P MANAGER
iodbc Ver. 0.11 (Beta 2) Copyright 1995 FFE
Software, Inc.
This is free software, and you are welcome to
redistribute it under certain
conditions; type 'help ?' for details.

1> create table t1 (i1 integer primary key, i2
varchar(10))
2> go
Msg 0, Level 16, State HYC00:
Optional feature not implemented

(0 rows affected)

```

프로그램 작성

ODBC를 이용하여 작성하는 프로그램에서 알티베이스 서버에 접속하고 종료하는 방법과 수행결과는 다음과 같다.

```

/* test_odbc.cpp */
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <stdio.h>
#include <stdlib.h>

#define SQL_LEN 1000
#define MSG_LEN 1024

SQLHENV      henv;
SQLHDBC      hdbc;
SQLHSTMT     hstmt;
SQLRETURN    retcode;

void execute_err(SQLHSTMT stat, char* q)
{
    printf("Error : %s\n",q);
    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLTCHAR errMsg[MSG_LEN];

    if (SQL_SUCCESS == SQLError ( henv, hdbc, stat,
        NULL, &errNo, errMsg, MSG_LEN, &msgLength ))
    {
        printf(" Error : # %lld, %s\n", errNo, errMsg);
    }

    SQLFreeStmt(stat, SQL_DROP);
    if (SQL_ERROR == SQLDisconnect(hdbc))
    {

```

```

printf("disconnect error\n");
}

SQLFreeConnect(hdbc);
SQLFreeEnv(henv);

exit (1);
}

void main()
{
    char    *DSN, *DBNAME, *USERNAME, *PASSWD,
    *PORTNO;
    char    query[SQL_LEN], name[21];
    int     age;

    SQLCHAR constr[100];
    SQLINTEGER len;
    DSN = "ALTIBASE"; // Domain Server Name
    DBNAME = "mydb"; // 데이터베이스 명.
    USERNAME = "SYS"; // 사용자명.
    PASSWD = "MANAGER"; // 사용자 암호.
    PORTNO = "20316"; // Port Number

    /* Environment 을 위한 메모리를 할당 */
    if(SQLAllocEnv(&henv) == SQL_ERROR)
    {
        printf("AllocEnv error!!\n");
        exit(1);
    }

    /* Connection 을 위한 메모리를 할당 */
    if(SQLAllocConnect(henv, &hdbc) ==
SQL_ERROR)
    {
        printf("AllocDbc error!!\n");
        SQLINTEGER errNo;
        SQLSMALLINT msgLength;
        SQLTCHAR errMsg[MSG_LEN];

        if (SQL_SUCCESS == SQLError ( henv,
NULL, NULL, NULL, &errNo, errMsg, MSG_LEN,
&msgLength ))
        {
            printf(" Error : # %lld, %s\n", errNo,
errMsg);
        }
        exit(1);
    }

    /* Connection 을 형성 */

```

```

sprintf((char*)constr, "DSN=%s; UID=%s; PWD=%s;
CONNTYPE=1; PORT_NO=%s", DSN, USERNAME, PASSWD,
PORTNO);

if ( SQLDriverConnect(hdbc, NULL, constr,
SQL_NTS, NULL, 0, NULL, SQL_DRIVER_COMPLETE))
{
printf("DBNAME = %s\n", DBNAME);
printf("USERNAME = %s\n", USERNAME);
printf("Connection error!!\n");
SQLINTEGER errNo;
SQLSMALLINT msgLength;
SQLTCHAR errMsg[MSG_LEN];

if (SQL_SUCCESS == SQLError ( henv, hdbc, NULL,
NULL, &errNo, errMsg, MSG_LEN, &msgLength ))
{
printf(" Error : # %lld, %s\n", errNo, errMsg);
}
SQLFreeConnect(hdbc);
SQLFreeEnv(henv);
exit(1);
}
printf("connected...\n");

/* statement 을 위한 메모리를 할당 */
if ( SQLAllocStmt(hdbc, &hstmt) == SQL_ERROR )
{
printf("AllocStmt error!!\n");
SQLDisconnect(hdbc);
SQLFreeConnect(hdbc);
SQLFreeEnv(henv);
exit(1);
}

/* 쿼리수행 */

/* 쿼리 직접수행 */
sprintf(query, "DROP TABLE TEST001");
SQLExecDirect(hstmt, (SQLTCHAR*)query, SQL_NTS);

sprintf(query, "CREATE TABLE TEST001 ( name
varchar(20), age number(3) )");
if (SQL_ERROR ==
SQLExecDirect(hstmt, (SQLTCHAR*)query, SQL_NTS))
{
execute_err(hstmt, query);
}

/* statement 를 준비하고 변수를 바인드한다. */
sprintf(query, "INSERT INTO TEST001
VALUES( ?, ? )");

```



```

if (SQL_ERROR == SQLPrepare(hstmt,
(SQLTCHAR*)query, SQL_NTS))
{
    execute_err(hstmt, query);
}

if (SQL_ERROR == SQLBindParameter(hstmt, 1,
SQL_PARAM_INPUT, SQL_C_CHAR, SQL_CHAR, 0, 0,
name, 19, &len))
{
    printf("SQLBindParameter error!!!
==> %s \n",query);
    exit(1);
}

if (SQL_ERROR == SQLBindParameter(hstmt, 2,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_NUMERIC, 0, 0,
&age, 0, &len))
{
    printf("SQLBindParameter error!!!
==> %s \n",query);
    exit(1);
}

/* 준비된 statement 를 수행 */
sprintf(name, "김민석");
age = 28;
if (SQL_ERROR == SQLExecute(hstmt))
{
    execute_err(hstmt, query);
}

sprintf(name, "홍길동");
age = 25;
if (SQL_ERROR == SQLExecute(hstmt))
{
    execute_err(hstmt, query);
}

sprintf(name, "아무개");
age = 34;
if (SQL_ERROR == SQLExecute(hstmt))
{
    execute_err(hstmt, query);
}

sprintf(query,"SELECT * FROM TEST001");
if (SQL_ERROR ==
SQLExecDirect(hstmt,(SQLTCHAR*)query, SQL_NTS))
{
    execute_err(hstmt, query);
}

```

```

/* Select 의 결과값을 변수에 저장 */
if (SQL_ERROR == SQLBindCol(hstmt, 1,
SQL_C_CHAR, name, 21, &len))
{
    printf("SQLBindCol error!!!\n");
    exit(1);
}

if (SQL_ERROR == SQLBindCol(hstmt, 2,
SQL_C_SLONG,&age, 0, &len))
{
    printf("SQLBindCol error!!!\n");
    exit(1);
}

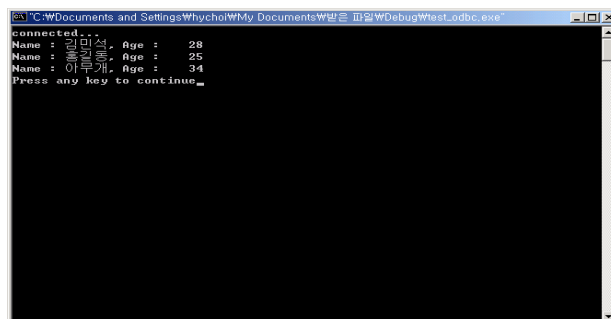
while ( SQLFetch(hstmt) == SQL_SUCCESS) //
결과값이 있는 동안 결과값을 받아 화면에 출력 */
{
    printf("Name : %5s, Age : %5ld\n",name,age);
}

/* 모든 handle 을 해제하고 접속을 종료 */
SQLFreeStmt(hstmt, SQL_DROP);
SQLDisconnect(hdbc);
SQLFreeConnect(hdbc);
SQLFreeEnv(henv);
}

```

실행결과

Visual C++ 에서 컴파일 후 exe file을 실행시키면 다음과 같은 결과를 볼 수 있다.



```

C:\Documents and Settings\Whychoi\My Documents\W\받은 파일\WDebug\West_odbc.exe
connected...
Name : 김민석, Age : 28
Name : 홍길동, Age : 25
Name : 이무개, Age : 34
Press any key to continue

```

7. 데이터베이스 이중화

데이터베이스 이중화 기능

알티베이스의 데이터베이스 이중화 기능은 서비스를 수행하고 있는 서버의 데이터베이스에 대한 최신(up-to-date) 백업 데이터베이스의 유지와, 서버의 예기치 않은 종료가 발생했을 때 대체 서버가 즉시 동일한 데이터베이스로 서비스를 재개할 수 있는 무정지 운영환경을 제공하는 것을 목적으로 한다.

데이터베이스 이중화 기능의 올바른 운영을 위해 다음과 같은 순으로 설명한다.

- 데이터베이스 이중화 방법
- 이중화 기능의 사용 방법

자세한 내용의 *ALTIBASE Replication User's Manual* 참조.

데이터베이스 이중화 방법

이중화 기능의 운영 방법

이중화 기능을 하기 위해서는 먼저 이중화를 만들 데이터가 있는 테이블을 정의하고, 이중화할 원격 서버와 이중화 이름, 프라이머리 키, 포트 번호 등의 스키마를 결정하고 지역 서버와 원격 서버에서 이중화 연결을 (CREATE REPLICATION) 만든다.

그 다음 원격 서버와 이중화를 시작 (ALTER REPLICATION START) 한다.

양 방향일 경우 원격 서버에서도 이중화 개시를 하면 양 방향으로 이중화를 사용할 수 있다.

이중화 기능의 사용 방법

알티베이스의 데이터베이스 이중화 기능을 위하여, 지역 서버(local server)는 시스템에서 발생하는 데이터베이스 변경 내용을 원격 서버(remote server)로 전송하며, 원격 서버는 전송 받은 내용을 자신의 데이터베이스에 반영하는 방법을 사용한다. 지역 서버와 원격 서버는 서비스 스레드와 별도로 이중화 관리에 필요한 스레드를 구동하며, 지역 서버의 이중화 송신 스레드(sender)는 데이터베이스의 변경에 따라 발생하는 내용을 원격 서버로 전송하며, 원격 서버의 이중화 수신 스레드(receiver)는 전송 받은 변경 내용을 데이터베이스에 반영시킨다. 또한 이중화 송,수신 스레드(sender와 receiver)는 대응 서버의 정상 및 비정상 종료를 자동 감지하며 이에 상응한 작업을 수행한다.

이중화 연결 생성

지역 서버와 원격 서버에 동일하게 이중화 정의를 한다.

예 제

지역 서버의 IP 주소가 192.168.1.11 이고 포트번호가 30524, 원격 서버의 IP 주소가 192.168.1.12 이고 포트번호가 50524 이라고 하자. 두 서버간의 EMPLOYEE 테이블과 DEPARTMENT 테이블을 이중화 한다고 가정하면, 필요한 이중화 정의는 다음과 같다.

지역 서버의 경우 (IP: 192.168.1.11)

```
iSQL> CREATE REPLICATION repl WITH
'192.168.1.12', 50524 FROM SYS.EMPLOYEE TO
SYS.EMPLOYEE, FROM SYS.DEPARTMENT TO
SYS.DEPARTMENT;
```

Create success

iSQL>

원격 서버의 경우 (IP: 192.168.1.12)

```
iSQL> CREATE REPLICATION repl WITH
'192.168.1.11', 30524 FROM SYS.EMPLOYEE TO
SYS.EMPLOYEE, FROM SYS.DEPARTMENT TO
SYS.DEPARTMENT;
```

Create success

이중화 시작

이중화를 시작한다.

예 제

'rep1' 이름으로 지역 서버의 이중화를 시작한다.

```
iSQL> ALTER REPLICATION rep1 START;
Alter success
```

다음은 원격 서버와 이중화를 시작(ALTER REPLICATION START)한 후, 이중화 상태를 화면으로 출력하였다.

```
iSQL> select * from system_.sys_replications_;
SYS_REPLICATIONS_.REPLICATION_NAME
SYS_REPLICATIONS_.LAST_USED_HOST_NO
-----
SYS_REPLICATIONS_.HOST_COUNT
SYS_REPLICATIONS_.IS_STARTED
SYS_REPLICATIONS_.XLSN_FILE_NO
-----
SYS_REPLICATIONS_.XLSN_OFFSET
SYS_REPLICATIONS_.ITEM_COUNT
SYS_REPLICATIONS_.CONFLICT_RESOLUTION
-----
REP1                                2
1              1              0
3472901        2              0
1 row selected.
```

```
Admin> connect
Connected with Altibase.
Admin> status replication;
*-----*
*           Replication           *
*-----*
=== Replication Sender List ===

REP1[192.168.1.12,50524](192.168.1.11,61812 =>
192.168.1.12,27255) Sender is running...
=== Replication Sender List END ===

=== Replication Receiver List ===
=== Replication Receiver List END ===
```

다음은 이중화 개시를 하기 전의 원격 서버의 상태를 화면으로 출력하였다.

```
iSQL> select * from system_.sys_replications_;
SYS_REPLICATIONS_.REPLICATION_NAME
SYS_REPLICATIONS_.LAST_USED_HOST_NO
-----
SYS_REPLICATIONS_.HOST_COUNT
SYS_REPLICATIONS_.IS_STARTED
SYS_REPLICATIONS_.XLSN_FILE_NO
-----
```

```

SYS_REPLICATIONS_.XLSN_OFFSET
SYS_REPLICATIONS_.ITEM_COUNT
SYS_REPLICATIONS_.CONFLICT_RESOLUTION
-----
REPl                                2
1              0              -1
-1             2              0
1 row selected.

Admin> connect
Connected with Altibase.
Admin> status replication;
*-----*
*           Replication           *
*-----*
=== Replication Sender List ===
=== Replication Sender List END ===

=== Replication Receiver List ===
REPl[192.168.1.11,30524](192.168.1.12,32857 =>
192.168.1.11,60211) Receiver is running...
=== Replication Receiver List END ===

```

다음 화면은 원격 서버에도 이중화 개시를 한 후 양방향 이중화의 진행 상태를 확인한 것이다.

```

iSQL> ALTER REPLICATION repl START;
Alter Success

Admin> status replication;
*-----*
*           Replication           *
*-----*
=== Replication Sender List ===

REPl[192.168.1.12,50524](192.168.1.11,61812 =>
192.168.1.12,27255) Sender is running...
=== Replication Sender List END ===

=== Replication Receiver List ===

REPl[192.168.1.11,30524](192.168.1.12,32857 =>
192.168.1.11,60211) Receiver is running...
=== Replication Receiver List END ===

```

또는

```

Dos prompt> server status replication
*-----*
*           Replication           *
*-----*
=== Replication Sender List ===

```



```

REP1[192.168.1.12,50524](192.168.1.11,61812 =>
192.168.1.12,27255) Sender is running...
=== Replication Sender List END ===

=== Replication Receiver List ===

REP1[192.168.1.11,30524](192.168.1.12,32857 =>
192.168.1.11,60211) Receiver is running...
=== Replication Receiver List END ===

```

이중화 테이블 삭제

이중화가 중지되어 있는 상태에서 이중화 테이블을 삭제할 수 있다.

예제

이중화 테이블 **EMPLOYEE**를 삭제한다.

```

iSQL> ALTER REPLICATION rep1 STOP;
Alter success
iSQL> ALTER REPLICATION rep1 DROP TABLE FROM
SYS.EMPLOYEE TO SYS.EMPLOYEE;
Alter success

```

이중화 테이블 추가

이중화가 중지되어 있는 상태에서 이중화 할 테이블을 추가할 수 있다.

예제

이중화 할 테이블 **EMPLOYEE**를 추가한다.

```

iSQL> ALTER REPLICATION rep1 STOP;
Alter success
iSQL> ALTER REPLICATION rep1 ADD TABLE FROM
SYS.EMPLOYEE TO SYS.EMPLOYEE;
Alter success

```

이중화 종료

이중화를 종료한다.

예제

'rep1' 이름의 이중화를 종료한다.

```

iSQL> ALTER REPLICATION rep1 STOP;
Alter success

```

이중화 연결 삭제

이중화 정의를 삭제한다. 이중화가 시작 되있는 상태에서는 이중화 연결을 삭제하기 전에 이중화 종료부터 해야 한다.

예 제

‘rep1’ 이름의 이중화 정의를 삭제하는 방법은 다음과 같다.

```
iSQL> ALTER REPLICATION rep1 STOP;  
Alter success  
iSQL> DROP REPLICATION rep1;  
Drop success
```

만약 이중화를 종료하지 않고 이중화 정의를 삭제하려고 하면 다음과 같은 에러메시지를 볼 수 있다.

```
iSQL> DROP REPLICATION rep1;  
[ERR-31089 : ERR-31089: The replication has  
already been started.]
```

8. 관리도구

알티베이스 관리도구 (dbadmin)

알티베이스 관리자 도구인 **dbadmin**은 알티베이스 서버의 시작/종료, 서버 동작 상태, 서버 기동 모드 등 기타 편리한 기능을 제공한다. 다음은 관리 명령을 중심으로 **dbadmin**의 간단한 사용 방법이다.

사용법

유닉스 프롬프트 상에서 **dbadmin** 실행 후 커맨드라인 명령을 이용한다.

기 능

```
Dos prompt> dbadmin
dbadmin Utility Ver 3.2.0
Copyright 1999-2000, ALTIBase Corporation or its
subsidiaries.
All rights reserved.

=====
                dbadmin(ALTIBASE ADMINISTRATION TOOL) HELP screen
=====
# General Commands
-----
CONNECT          - Connect to dbadmin
DISCONNECT       - Disconnect from dbadmin
HELP             - This screen
SHELL            - Execution of Shell
![shell command] - Execution of Shell Command ex) !ls
EXIT             - QUIT alias
QUIT            - Quit from dbadmin console
-----
# Management Commands : opt => options
-----
STARTUP          - Startup altibase server
STATUS [opt]     - Examine of Altibase status
    - SESSION [opt] : View Session Status
    - PROPERTY      : View All Property
    - REPLICATION   : View Replication Information
    - DB [opt]      : View Memory Usages.
    - MEMORY        : View Internal Memory Usages.
    - ALL           : View All Status
ex)
    * Status          - Overview Status
    * Status All       - View All Status
    * Status Session   - Overview Session Status
    * Status Session All - View All Session Status
    * Status Session Num - View Session Status of the
Number
    * Status DB        - View Database Informations
    * Status DB All    - View All Table Informations
    * Status DB Tbl-Name - View Certain Table
Informations
TERMINATE Session-Number - Terminate a Session
Specified by the Number
```

```

SHUTDOWN [NORMAL | IMMEDIATE | ABORT] - Shutdown altibase
server
DTX      [opt]      - Manipulate Distributed Transactions.
- SHOW          : View All Distributed Transactions.
- ROLLBACK [tid] : Rollback Specified Distributed
Transaction.
- COMMIT  [tid] : Commit Specified Distributed
Transaction.
=====

```

dbadmin 명령어

다음과 같은 dbadmin 명령어를 이용하여 dbadmin을 사용할 수 있다.

분류	명령어	옵션	기능
일반명령어	connect	-	dbadmin 접속
	disconnect	-	dbadmin 연결 해제
	Help	-	도움말
	shell	-	유닉스 프롬프트로 빠지기
	![shell command]	-	유닉스 명령어 실행
	exit	-	dbadmin 종료
	quit	-	dbadmin 종료
관리명령어	startup	-	알티베이스 구동
	status	session	알티베이스 상태 보기
		db	
		memory	
		property	
		replication	
		all	
	shutdown	normal	알티베이스 종료
		immediate	
		abort	
	terminate	session_number	특정 세션 강제 종료
	dtx	show	분산 트랜잭션 관련
		rollback tid	
		commit tid	

연결 / 해제

연결 (Connect)

시스템 관리자 모드로 진입하는 것으로서, 이 명령을 수행해야만 알티베이스 서버를 실행시킬 수 있을 뿐만 아니라 알티베이스

시스템 관리를 수행할 수 있다. **connect** 명령을 수행하지 않고서는 **dbadmin**이 제공하는 모든 기능(**quit**, **exit** 제외)을 사용할 수 없다.

자세한 설명은 제 3장 알티베이스 실행과 종료를 참고

```
Admin> connect
Connected Internally.
Admin> startup
Trying Connect to Altibase.. Connected with
Altibase.
=== SYSTEM MEMORY CONTROL ===
  PageSize = 8192 byte MaxMem = 8192M FreeMem =
377M
  SwapSize = 10631M FreeSwap = 5708M
...
...
```

해제 (Disconnect)

시스템 관리자 모드를 해제하는 것으로서, **connect** 명령과 서버 **shutdown** 이후에 수행시켜야 효력이 발생한다. 이 명령을 통하여 시스템 관리자 모드가 해제된 경우에는 다시 **connect** 명령을 수행해야 알티베이스 시스템을 관리할 수 있다.

자세한 설명은 제 3장 알티베이스 실행과 종료를 참고

```
Admin> connect
Connected with Altibase.
Admin> shutdown normal
Ok..Shutdown Proceeding....
[NORMAL SHUTDOWN]
[PREPARE] Stop Modules...[SUCCESS]
...
...
Admin> disconnect
dbadmin Disconnected....
```

시작 / 종료

시작 (Startup)

startup은 알티베이스 서버를 구동 시키는 명령이다. 이 명령을 수행하면 알티베이스 서버는 프로퍼티 로딩과 시스템 메모리 검사를 거친 후, 알티베이스 서버를 구동한다. 알티베이스 서버 구동 시에 수행되는 일들은 알티베이스 시스템 환경 초기화, 시스템 데이터 초기화, 시그널 핸들링, **DB** 공간의 메모리 초기화, 질의 처리 모듈 초기화, 마지막으로 쓰레드들을 초기화 함으로서 알티베이스 서버의 구동이 완료된다.

자세한 설명은 제 3장 알티베이스 실행과 종료를 참고

```
Admin> connect
Connected Internally.
```

```

Admin> startup
Trying Connect to Altibase. Connected with
Altibase.
=== SYSTEM MEMORY CONTROL ===
PageSize = 8192 byte MaxMem = 8192M FreeMem =
6214M
SwapSize = 10840M FreeSwap = 10557M
...

```

* 참고¹: 알티베이스 프로세스의 구동 시 해당 데이터베이스 파일을 로딩할 때의 동작을 세부적으로 나누면 **Dynamic Memory**, **Created Shared Memory**, 그리고 **Attached Shared Memory** 등 세 종류로 구별할 수 있다.

종료 (Shutdown)

shutdown은 알티베이스 서버를 종료 시키는 명령이다. 이 명령은 세 가지 옵션이 있는데, 각 옵션에 따라 서버를 종료하는 방식이 다르다. 각각의 옵션은 다음과 같다.

- normal 서버를 정상적으로 종료하는 방식으로, 클라이언트들이 모두 종료될 때까지 서버의

¹ Attached Shared Memory 모드(디스크를 접근하지 않고 공유 메모리상에 이미 로딩된 이미지를 attach 함)로 서버가 구동 될 때 해당 공유 메모리 버전과 디스크에 있는 버전이 서로 동일하지 않으면 (다수의 공유메모리로 만들어진 DB Image 에 대해 누군가가 한 개를 ipcrm 을 이용하여 제거한 경우, Created Shared Memory 로 로딩 중에 서버를 죽였고, 이후에 다시 Restart 를 한 경우, 공유 메모리 이미지가 존재하는 중에 Disk 의 이미지를 영동한 것으로 교체를 하고, Startup 을 시도한 경우 발생할 수 있다.) 서버 구동은 실패한다. 즉, 공유 메모리에 존재하는 일련 번호(Unique ID)와 disk 의 일련 번호를 검사하여 같은 경우 서버를 구동한다.

Unique ID 의 확인은 dbadmin 이 제공하는 status db 명령으로 (또는 서버 스크립트 명령 server status db 를 이용) Storage Info 에서 확인 할 수 있다.

Admin> status db

```

*-----*
*                      DB                      *
*-----*
<< Database Info >>
Version Info   -- 3110000
Database Name  -- mydb
Database Type  -- Dynamic Memory Version
Product Info   -- WIN_NT_5.0-32bit-3.5.2p2-release-VC7 (MS_WINDOWS)
Storage Info   -- -3FD43574:00066768-00003F38
OS/Bit  Env    -- 32
Log File Size  -- 10485760
DB File Size   -- 1073741824
DB File Count  -- (First)  = 1
DB File Count  -- (Second) = 1

```

```

<< Database Usage >>
Maximum Page   -- 131072 ( Available Size = 4096M)
Allocated Page -- 320 (Current Size = 10M)
Used Page      -- 53 (Current Size = 1M)
Free Page      -- 267 (Current Size = 8M)
DB File Count  -- (Second) = 1

```

종료 작업을 대기하는 방법이다. 서버가 종료하면서 수행하는 일들은 클라이언트-서버간 통신 세션을 감지하는 쓰레드의 종료, 서비스 쓰레드의 종료, 자료저장 관리자의 종료, 그리고 알티베이스 서버 프로세스가 완전히 종료되기를 대기하는 일들을 수행함으로써, 알티베이스 서버를 종료한다.

- **immediate** 서버를 종료할 때 현재 연결된 세션들을 강제로 단절시킨 다음, 알티베이스 서버가 현재 실행 중인 트랜잭션들을 철회(rollback) 시키고 알티베이스 서버를 종료하는 방법이다.
- **abort** 'kill -9' 시스템 명령을 사용하여 알티베이스 서버를 강제로 중단시키는 방법이다. 이 방법으로 알티베이스 서버를 종료하면, 데이터베이스가 완전하지 못하여 다음에 알티베이스 서버를 실행할 때 데이터베이스 복구 과정을 거쳐야 한다.

자세한 설명은 제 3장 알티베이스 실행과 종료를 참고

```
Admin> connect
Connected with Altibase.
Admin> shutdown normal
Ok..Shutdown Proceeding....
[NORMAL SHUTDOWN]
[PREPARE] Stop Modules...[SUCCESS]
...
...
```

상 태(Status)

클라이언트-서버 구조의 알티베이스 시스템 정보를 제공하는 명령이다. 제공하는 정보로는 컴퓨팅 시스템의 자원, 알티베이스 프로퍼티, 서비스 쓰레드들의 상태, 연결된 세션의 통신 방법 등에 관한 것이 있다. 아래 그림은 지역 서버에서 출력된 **status all** 명령의 결과 화면을 보여준 것이다. 알티베이스 프로퍼티, 연결된 세션의 통신 방법, 이중화 정보, 데이터베이스 정보 등에 대한 각각의 결과에 대해서는 다음의 명령으로 알 수 있다.

```
Admin> status: 알티베이스 상태 확인
Admin> status session [all/session_number]:
알티베이스에 연결된 세션에 관한 정보 출력
Admin> status db [all/table_name]: DB(또는
테이블)에 할당된 데이터 페이지에 관한 정보 및 테이블 별
메모리 효율성 출력
Admin> status memory: 메모리 객체별 메모리 사용량
출력
```



```
Admin> status replication: 리플리케이션 상태 출력
Admin> status all: 위의 모든 정보를 출력
```

알티베이스 상태 모니터링 (세션 모니터링, 테이블 모니터링, 메모리 사용량 모니터링, 이중화 모니터링)에 대한 자세한 설명은 *ALTIBASE Administrator's Guide* 참고

```
Admin> connect
Connected with Altibase.
Admin> status all
*-----*
*              Overview              *
*-----*

Start   Time = Mon Dec 08 20:15:17 2003
Current Time = Mon Dec 08 20:18:35 2003
Running Time = 0 day 0 hour 3 min 17 sec
Process ID  = 364

*-----*
*              Property              *
*-----*
*=====*
* Common Properties                  *
*=====*
DB_NAME      = [mydb]
LOG_DIR      = [c:\altibase3\altibase_home\logs]
DB0_DIR      = [c:\altibase3\altibase_home\dbs]
DB1_DIR      = [c:\altibase3\altibase_home\dbs]
SHM_DB_KEY   = [0]
LOG_FILE_SIZE = [10485760]
DB_FILE_SIZE = [1073741824]
PORT_NO      = [20300]
PORT_DIR     =
[c:\altibase3\altibase_home\trc\altibased20300]
IPC_CHANNEL_COUNT      = [0]
IPC_CHANNEL_BUF_COUNT  = [0]
NLS_USE                 = [US7ASCII]
PARALLEL_LOAD_FACTOR   = [1]
TRANSACTION_TABLE_SIZE = [1024]
TRANSACTION_DURABILITY_LEVEL = [3]

*=====*
* Session Properties    *
*=====*
MAX_DB_SIZE   = [4194304K]
MAX_THREAD    = [100]
MAX_CLIENT    = [100]
MAX_LISTEN    = [128]
CM_BUF_SIZE   = [65536]
CM_DISCONN_DETECT_TIME = [3]
CM_DISCONN_HIGHWATER_MARK = [3]

*=====*
* Logging Properties    *
*=====*
AUTO_COMMIT    = [auto commit]
CHECK_POINT_INTERVAL_IN_SEC = [6000]
CHECK_POINT_INTERVAL_IN_LOG = [100]
AUTO_REMOVE_ARCHIVE_LOG = [Yes]
PREPARE_LOG_FILE_COUNT = [2]
OPEN_LOG_FILE_COUNT = [3]
MAX_KEEP_LOG_FILE = [100]
STARTUP_SHM_CHUNK_SIZE = [2147483648]
```

```

*=====*
* Replication Properties*
*=====*
REPLICATION_PORT_NO      = [0]
REPLICATION_MAX_LOGFILE  = [0]
REPLICATION_UPDATE_REPLACE = [0]
REPLICATION_RECEIVE_TIMEOUT = [300]
REPLICATION_SYNC_LOG      = [0]

*=====*
* Boot Log Properties *
*=====*
MSGLOG_FILE_SIZE = [0]
MSGLOG_MAX_FILE  = [0]

*-----*
*           Session           *
*-----*
Session Count = 1

Next   SessionID = [4]
Max     Session  = [100]
Max     Thread   = [100]
Created Thread   = [3]
Waiting Thread   = [0]
Current Session  = [1]
        Running Session = [1]
        Queueing Session = [0]

Session [3] <Type : TCP/IP INET > <From :
127.0.0.1:39952,0(Opened)>
        Thread ID : 836
        State      : Session_State_Service [Running]
        Attribute   : <AutoCommit> <Tx Inactivated>
        Isolation Level : 0
        Open Stmt Count : 0           Current stmt ID : 0
        Protocol      : [Admin Stat Protocol]
        Idle Time      : 0 / 0
        Statement [0] : Mode = Allocated. : SCN=[0]   Time
[U=>0, Q=>0, F=>0]
=> ""

*-----*
*           Replication           *
*-----*
... No Replications..

*-----*
*           DB                     *
*-----*
Version Info  -- 3110000
Database Name -- mydb
Database Type -- Dynamic Memory Version
Product Info  -- WIN_NT_5.0-32bit-3.5.2p2-release-VC7
               (MS_WINDOWS)
Storage Info  -- -3FD43574:00066768-00003F38
OS/Bit Env    -- 32
Log File Size -- 10485760
DB File Size  -- 1073741824
DB File Count -- (First) = 1
DB File Count -- (Second) = 1

<< Database Usage >>
Maximum Page  -- 131072 ( Available Size = 4096M)
Allocated Page -- 320 (Current Size = 10M)

```

```

Used Page      -- 53 (Current Size = 1M)
Free Page      -- 267 (Current Size = 8M)

<< Table Usage >>
Table : SYS_TABLES_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
    Used Mem = Total:0.003M, Fixed:0.003M,
Variable:0.000M
    Memory Efficiency = 9.23%
Table : SYS_COLUMNS_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.594M, Fixed:0.594M,
Variable:0.000M
    Used Mem = Total:0.511M, Fixed:0.511M,
Variable:0.000M
    Memory Efficiency = 86.03%
Table : NEXT_TABLE_ID
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.00%
Table : NEXT_USER_ID
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.00%
Table : NEXT_INDEX_ID
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.00%
Table : NEXT_CONSTRAINT_ID
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.00%
Table : NEXT_PROC_ID
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.00%
Table : NEXT_PROC_PARA_ID
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.00%
Table : NEXT_REPL_HOST_NO
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.00%

```

```

Table : SYS_DATABASE_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.00%
Table : SYS_USERS_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.68%
Table : SYS_INDICES_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
    Used Mem = Total:0.004M, Fixed:0.004M,
Variable:0.000M
    Memory Efficiency = 13.65%
Table : SYS_INDEX_COLUMNS_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
    Used Mem = Total:0.004M, Fixed:0.004M,
Variable:0.000M
    Memory Efficiency = 12.01%
Table : SYS_CONSTRAINTS_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.00%
Table : SYS_CONSTRAINT_COLUMNS_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.00%
Table : SYS_NATIVE_PROCEDURE_GROUPS_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.00%
Table : SYS_PROCEDURES_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.094M, Fixed:0.094M,
Variable:0.000M
    Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
    Memory Efficiency = 0.28%
Table : SYS_PROC_PARAS_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
    Used Mem = Total:0.008M, Fixed:0.008M,
Variable:0.000M
    Memory Efficiency = 25.15%
Table : SYS_PROC_PARSE_
    LockMode: [No Lock], Grant Count: 0, Request Count: 0
    Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M

```

```

        Used Mem = Total:0.001M, Fixed:0.001M,
Variable:0.000M
        Memory Efficiency = 3.71%
Table : SYS_PROC_RELATED_
        LockMode: [No Lock], Grant Count: 0, Request Count: 0
        Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
        Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
        Memory Efficiency = 0.00%
Table : SYS_REPLICATIONS_
        LockMode: [No Lock], Grant Count: 0, Request Count: 0
        Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
        Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
        Memory Efficiency = 0.00%
Table : SYS_REPL_HOSTS_
        LockMode: [No Lock], Grant Count: 0, Request Count: 0
        Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
        Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
        Memory Efficiency = 0.00%
Table : SYS_REPL_ITEMS_
        LockMode: [No Lock], Grant Count: 0, Request Count: 0
        Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
        Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
        Memory Efficiency = 0.00%
Table : SYS_PRIVILEGES_
        LockMode: [No Lock], Grant Count: 0, Request Count: 0
        Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
        Used Mem = Total:0.003M, Fixed:0.003M,
Variable:0.000M
        Memory Efficiency = 10.01%
Table : SYS_GRANT_SYSTEM_
        LockMode: [No Lock], Grant Count: 0, Request Count: 0
        Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
        Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
        Memory Efficiency = 0.98%
Table : SYS_GRANT_OBJECT_
        LockMode: [No Lock], Grant Count: 0, Request Count: 0
        Alloc Mem = Total:0.031M, Fixed:0.031M,
Variable:0.000M
        Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
        Memory Efficiency = 0.00%
Table : SYS_XA_HEURISTIC_TRANS_
        LockMode: [No Lock], Grant Count: 0, Request Count: 0
        Alloc Mem = Total:0.063M, Fixed:0.031M,
Variable:0.031M
        Used Mem = Total:0.000M, Fixed:0.000M,
Variable:0.000M
        Memory Efficiency = 0.15%
*-----*
*           Memory Usages(Internal)           *
*-----*
*-----*
*           Session Manager                     *
*-----*
[ Statement Object Memory Pool ]

```

```

- Memory List Count:    1
- Slot  Size: 2744
- Memory List *** [1]
  Mutex Internal State
    Lock      Frequency:    6
    Average Try Frequency:  0
    Miss      Frequency:    0
    Memory Usage: 85 KB
[ Task Session Object Memory Pool ]

- Memory List Count:    1
- Slot  Size:13464
- Memory List *** [1]
  Mutex Internal State
    Lock      Frequency:    5
    Average Try Frequency:  0
    Miss      Frequency:    0
    Memory Usage: 420 KB
[ Task Session Mutex ]

    Lock      Frequency:    7
    Average Try Frequency:  0
    Miss      Frequency:    0
-----*
*                Thread Manager                *
*-----*
[ Task Thread Object Memory Pool ]

- Memory List Count:    1
- Slot  Size:    88
- Memory List *** [1]
  Mutex Internal State
    Lock      Frequency:    3
    Average Try Frequency:  0
    Miss      Frequency:    0
    Memory Usage: 2 KB
[ Task Connection Detector Mutex ]

    Lock      Frequency:    12
    Average Try Frequency:  0
    Miss      Frequency:    0
-----*
*                Storage Memory Manager          *
*-----*
[ Temp Memory Pool ]
- Memory List Count:    1
- Slot  Size:32776
- Memory List *** [1]
  Mutex Internal State
    Lock      Frequency:    370
    Average Try Frequency:  0
    Miss      Frequency:    0
    Memory Usage: 12288 KB
[ PCH Memory Pool ]
- Memory List Count:    1
- Slot  Size:    48
- Memory List *** [1]
  Mutex Internal State
    Lock      Frequency:    320
    Average Try Frequency:  0
    Miss      Frequency:    0
    Memory Usage: 40 KB
[ Memory Mgr Mutex ]
    Lock      Frequency:    3
    Average Try Frequency:  0
    Miss      Frequency:    0
[ Dirty Page List Mgr ]

```

```

- Dirty Page List Count:2
- Dirty Page List ***[0]
  Dirty Page List Mutex
    Lock      Frequency:      6
    Average Try Frequency:    0
    Miss      Frequency:      0
  <Memory List>
  Mutex Internal State
    Lock      Frequency:      0
    Average Try Frequency:    0
    Miss      Frequency:      0
  Memory Usage: 8 KB
- Dirty Page List ***[1]
  Dirty Page List Mutex
    Lock      Frequency:      6
    Average Try Frequency:    0
    Miss      Frequency:      0
  <Memory List>
  Mutex Internal State
    Lock      Frequency:      0
    Average Try Frequency:    0
    Miss      Frequency:      0
  Memory Usage: 8 KB

*-----*
*      Storage Transaction Manager      *
*-----*
Total Slot Count: 1024
Transaction Pool Size: 1260KB
Transaction OID List Pool
  Mutex Internal State
    Lock      Frequency:      2
    Average Try Frequency:    0
    Miss      Frequency:      0
Memory Usage: 304 KB
Free List Count: 4
- Commit Mutex
  Lock      Frequency:      1
  Average Try Frequency:    0
  Miss      Frequency:      0
- Free List ***[1]
  Free Slot Count:256
  List Mutex Internal State
    Lock      Frequency:      73
    Average Try Frequency:    0
    Miss      Frequency:      0
- Free List ***[2]
  Free Slot Count:256
  List Mutex Internal State
    Lock      Frequency:      0
    Average Try Frequency:    0
    Miss      Frequency:      0
- Free List ***[3]
  Free Slot Count:256
  List Mutex Internal State
    Lock      Frequency:      0
    Average Try Frequency:    0
    Miss      Frequency:      0
- Free List ***[4]
  Free Slot Count:256
  List Mutex Internal State
    Lock      Frequency:      0
    Average Try Frequency:    0
    Miss      Frequency:      0

*-----*
*      Storage Lock Manager      *

```

```

*-----*
[ Lock Item Memory Pool ]
- Memory List Count:      2
- Slot Size: 168
- Memory List *** [1]
  Mutex Internal State
    Lock      Frequency:      2
    Average Try Frequency:    0
    Miss      Frequency:      0
  Memory Usage: 160 KB
- Memory List *** [2]
  Mutex Internal State
    Lock      Frequency:      0
    Average Try Frequency:    0
    Miss      Frequency:      0
  Memory Usage: 160 KB
*-----*
*           Storage Recovery Manager           *
*-----*
[ Log Tail Mutex ]
  Lock      Frequency:      35
  Average Try Frequency:    0
  Miss      Frequency:      0
[ Transaction Log Pool ]
- Memory List Count:      4
- Slot Size: 65544
- Memory List *** [1]
  Mutex Internal State
    Lock      Frequency:      0
    Average Try Frequency:    0
    Miss      Frequency:      0
  Memory Usage: 1280 KB
- Memory List *** [2]
  Mutex Internal State
    Lock      Frequency:      0
    Average Try Frequency:    0
    Miss      Frequency:      0
  Memory Usage: 1280 KB
- Memory List *** [3]
  Mutex Internal State
    Lock      Frequency:      0
    Average Try Frequency:    0
    Miss      Frequency:      0
  Memory Usage: 1280 KB
- Memory List *** [4]
  Mutex Internal State
    Lock      Frequency:      0
    Average Try Frequency:    0
    Miss      Frequency:      0
  Memory Usage: 1280 KB
[ Open Log File Info ]
  3
*-----*
*           Storage Index Manager           *
*-----*
[ TTree Index Node Memory Pool ]
  Mutex Internal State
    Lock      Frequency:      40
    Average Try Frequency:    0
    Miss      Frequency:      0
  Memory Usage: 128 KB
*-----*
*           Storage Garbage Collector           *
*-----*
Index Slot Garbage Collecting Thread
System SCN:80003
Minimum SCN In Transactions: 80003

```



```

Minimum SCN In Garbage List: 80002
Thread    Count:      3
Add       Count:      1
Processed Count:      0
Mutex Internal State
  Lock     Frequency:   5388
  Average Try Frequency: 0
  Miss     Frequency:   0
TTree Index Node Garbage Collecting Thread
Thread Count:      2
- Thread *** [1]

Add       Count:      1
Processed Count:      0
Mutex Internal State
  Lock     Frequency:   4
  Average Try Frequency: 0
  Miss     Frequency:   0
- Thread *** [2]

Add       Count:      1
Processed Count:      0
Mutex Internal State
  Lock     Frequency:   4
  Average Try Frequency: 0
  Miss     Frequency:   0
Table Garbage Collecting Thread
Thread    Count:      3
Added     Count:      0
Processed Count:      0
Mutex Internal State
  Lock     Frequency:   5388
  Average Try Frequency: 0
  Miss     Frequency:   0

```

셸

이 명령을 수행하면 **dbadmin** 상태에서 셸 명령 모드로 전환되어 현재 사용하고 있는 셸 명령들을 수행할 수 있다. 다시 **dbadmin** 상태로 돌아 가려면 **exit** 명령을 수행하면 된다. **dbadmin** 상태에서 바로 한 개의 셸 명령을 수행하려면 예를 들어, 디렉터리의 내용을 보려면 **!dir**와 같이 셸 명령 앞에 **!**를 붙이면 된다.

```
Admin> !dir
```

기 타

help

dbadmin에서 제공하는 명령등에 대한 도움말 제공

quit

dbadmin을 종료

exit

quit과 같이 dbadmin을 종료

공유 메모리 관리도구 (shmutil)

공유 메모리를 이용하여 알티베이스 데이터베이스를 사용할 수 있다.

사용법

```
Dos prompt> shmutil [OPTIONS -pwe]
                        [ -d HOME_DIR]
                        [ -f PROPERTY_FILE]

[-d HOME_DIR] 알티베이스 의 홈 디렉터리 경로
[-f PROPERTY_FILE] 알티베이스 프로퍼티 파일의 경로
```

옵션

-p: 공유 메모리 정보 확인
-w: 공유 메모리 저장
-e: 공유 메모리 제거

공유 메모리 정보 확인

현재 공유 메모리 정보와 공유 메모리 세그먼트가 여러 개 생성되었다면 각각의 정보를 확인 할 수 있다.

예제

```
Dos prompt> shmutil -p

정상적인 경우 다음과 같은 메시지가 출력된다.

Dos prompt> shmutil -p
ShmUtil: Release 3.5.2p2 - Production on Nov 28 2003
12:03:02
(c) Copyright 2001 ALTIBase Corporation. All rights
reserved.

##### IPC Information #####
SHM BASE KEY = 10(0xa) ID = 0(0x0)

##### Brief Information #####
#####
      SIZE(MB)      PAGE( # )
TOTAL      10.0000      320
USED       1.6875       54
FREE       8.3125      266
Timestamp is 0(sec) / 0(usec)
```

```
##### Detail Information
#####
## Shared Memory Chunk ##
Shm Key   = 10
Shm Id    = 0
Page Count = 320
Size(MB)  = 10.0000
...

```

* 공유 메모리 사용량 결과 분석에 대한 자세한 내용은
ALTIBASE Administrator's Guide 참고

공유 메모리 저장

현재 공유 메모리의 DB를 mydbkim으로 백업 작업한다. 반드시 서버 종료 후 사용할 것.

예 제

```
Dos prompt> shmutil -w
```

정상적인 경우 다음과 같은 메시지가 출력된다.

```
Dos prompt> shmutil -w
```

```
ShmUtil: Release 3.2.0 - Production on Feb 27 2003 16:07:12
(c) Copyright 2001 ALTIBase Corporation. All rights
reserved.
```

```
!!!!!! WARNING !!!!!!
Duplicated DB-Name will overwrite original DB-file.
Use Different DB-Name.
Original DB Name 1) =>
/user5/charlie/work/altibase_home/dbs/mydb-0
2) =>
/user5/charlie/work/altibase_home/dbs/mydb-1
Input DB-Name => mydbkim
Database File
saving.../user5/charlie/work/altibase_home/dbs/mydbkim
[SUCCESS] Database File Saved
```

공유 메모리 제거

현재 공유메모리상 DB를 제거. 반드시 서버 종료 후 사용할 것.

예 제

```
Dos prompt> shmutil -e
```

정상적인 경우 다음과 같은 메시지가 출력된다.

```
Dos prompt> shmutil -e
```

```
ShmUtil: Release 3.2.0 - Production on Feb 27 2003 16:07:12
(c) Copyright 2001 ALTIBase Corporation. All rights
reserved.
```

```
Ready for Destroying Shared Memory? (y/N)y  
[SUCCESS] Destroyed Shared Memory.
```


9. 차기 구현 요소

누락된 기능 및 요소

프로토콜

문서가 작성된 현재까지 지원하는 프로토콜은 TCP와 UNIX 타입 IPC 프로토콜이다. 앞으로 지원할 프로토콜은 UNIX Domain Socket을 대신할 NamedPipe와 윈도우 IPC 프로토콜을 조사하여 좀더 빠른 것을 찾아 이식하려 한다.

비정상 종료 처리

현재 윈도우 버전은 `altibase.map` 파일을 지원하므로, 알티베이스 비정상 종료시 발생하는 `call stack` 정보를 이용하여 어느 모듈의 어느 위치에서 문제가 발생했는지 파악할 수 있게 되어 있다. 그러나, 유닉스 버전과 달리, `altibase_boot.log` 에 이러한 정보가 기록되진 못한다. 즉, Dr.Watson이나 윈도우 운영체제가 지원하는 `dumpstack` 프로그램을 사용하여야 한다.

IPC 지원

윈도우는 유닉스의 공유 메모리 기능을 어느 정도 지원하고 있다. 이러한 사실에서 현재 알티베이스 3은 공유 메모리 방식의 데이터베이스를 작성하고 운영할 수 있다. UNIX의 공유 메모리와 매우 다른 점은 공유 메모리를 생성하고 핸들하는 프로세스가 종료되면 공유 메모리에 접근할 방법이 없다는 것이다. 이러한 제약 사항을 극복하기 위해서, 현재 알티베이스 3가 제공하는 방식은 `ipc-daemon.exe` 라는 데몬 프로그램이다. 이 프로그램이 동작하고 있을때만, 알티베이스의 IPC 기능을 사용할 수 있다.

시그널 처리

윈도우 운영체제는 기본적인 시그널 기능을 제공한다. 그러나, UNIX 처럼 `signal masking` 처리는 불가능하다. 공개된 몇 개의 프로그램이 이러한 기능을 에뮬레이션하지만, 알티베이스 3에 접목하기에는 인터페이스 면에서 많은 문제가 있었다. 따라서, 알티베이스 UNIX 버전이 제공하는 정도의 시그널 기능을 거의 사용하지 않도록 막아둔 상태이다. `Dumpstack` 기능도 이러한 이유와 연관되어 올바르게 동작하지 않고 있다.

Pthread 지원

알티베이스는 내부의 몇 부분에서 **pthread**를 사용한다. **Mutex**가 가장 대표적인 부분이라고 할 수 있다. 현재 알티베이스 윈도우 버전을 작성하면서, 이러한 **pthread** 라이브러리의 몇몇 함수를 윈도우 고유 쓰레드 함수를 이용하여 구현을 하였다. 이러한 구현은 어디까지나, 알티베이스 3를 작성하기 위함이었을 뿐, 사용자에게 제공하기 위해 만들어 둔 것은 아니다. 현재 공개 소프트웨어로 윈도우용 **pthread** 라이브러리를 제공하는 곳이 한 군데 있다. Redhat.com의 **cygwin** 관련 자료와 함께 있는데, 특별히 **pthread** 라이브러리 방식의 클라이언트 쓰레드 프로그램을 작성하려 한다면, 이것이 적당할 것 같다.

ODBC 관련

알티베이스 3에는 UNIX 버전의 **Altibase CLI**의 모체이자 윈도우의 표준 인터페이스인 **ODBC**를 레벨 2까지 완벽히 지원한다.(레벨 3 함수도 몇가지 지원하고 있는데, 이러한 내용은 *ALTIBASE CLI User's Manual*을 참고하기 바란다.) 또한 **native** 응용 프로그램 작성을 위해서 **sqlcli.lib**도 UNIX 처럼 제공한다.

10. FAQ

Starting FAQ

Startup 관련 FAQ

질문1

licence 파일을 conf 디렉터리에 copy를 했고, manual에 명시된 대로 Altibase DB server를 가동하고자 dbadmin을 shell 상에서 실행해서 connect, startup 명령을 차례대로 입력 했습니다.

그런데, 아래와 같은 에러 메시지가 뜨면서 구동이 안 됩니다.

```
Admin> connect
```

```
Connected Internally.
```

```
Admin> startup
```

```
Trying Connect to Altibase.....Startup Failure. Check Your Environment.
```

답변

구동이 안 되는 이유는 2가지가 있습니다.

첫째 환경변수가 잘못 설정되어 있는 경우 입니다.

둘째 데이터베이스 파일을 생성하지 않고 구동을 하는 경우 입니다.

첫째 방법으로는 (bash shell인 경우)

```
export ALTIBASE_HOME=${HOME}/altibase_home
```

```
export PATH=${ALTIBASE_HOME}/bin:${PATH}
```

이 설정됐는지 확인해 보시고, 둘째 방법으로는 (알티베이스 매뉴얼 ALTIBASE_Starting 참고)

```
$createdb -M <dbsize>
```

를 사용하여 데이터베이스 파일을 생성하셨는지 확인해 보세요.

질문2

서버 설치 후 구동이 안되고, Unable to invoke read() 에러 메시지가 나타납니다.

답변

생성될 데이터베이스 파일의 크기를 너무 크게 잡으셔서 디스크 공간이 부족할 때 발생할 수 있습니다. 디스크 공간을 더 확보하시든지 생성할 데이터베이스 파일의 크기를 작게 잡고 구동을 다시 시도해 보세요.

질문3

Altibase DB server를 install한 후 dbadmin을 실행하여 startup을 했는데, 다음과 같은 에러가 발생하여 문의 드립니다.

```
-----
[PREPARE] Query Preprocessor Unit Init(TB,QC,TC)....[SUCCESS]
```

```
Can't read from altibase..
```

```
Admin>
-----
```

답변

altibase.properties에 IPC관련 파라미터가 너무 크게 잡혀있지 않았나 확인 해보세요.

IPC_CHANNEL_COUNT, IPC_CHANNEL_BUF_COUNT

이 값을 작게 설정하시거나 아니면 관련된 시스템 파라미터를 변경하여 재부팅 해보세요.

질문4

conf/altibase.properties file에 아래와 같이 되어 있는 부분을 수정하면 어떻게 되는지요.

Fixed Properties

should not be modified after createdb

단지 반영이 안 되는 건지 DBMS 구동 시 문제가 생기는지, 어떤 영향을 미치는지 알고 싶습니다.

답변

해당 block에 있는 properties중 고치면 db start가 안 되는 것이 있습니다.

예를 들자면 LOG_FILE_SIZE 같은 property는 db create할 때 적용되는 property로서 한 번 create한 후 이 값을 고치면 db start가 안됩니다. db name도 마찬가지 입니다.

위치와 상관 있는 (LOG_DIR, DB0_DIR, DB1_DIR) property들은 바꾸어도 상관없습니다.

반드시 starting manual을 참조하여서 바꾸어야 합니다.

질문5

log파일이 많이 생겨서 file system full이 발생하고 log파일을 생성할 수 없어서 다시 구동을 시켰는데, 안됩니다.

답변

Altibase 로그 파일은 circular 로그가 아니므로 재사용되지 않습니다. 체크포인트 발생시 recovery를 위해 필요한 파일만

유지하고, 나머지는 삭제할 수 있게끔 프로퍼티 중
AUTO_REMOVE_ARCHIVE_LOG = 1로 설정하세요.

질문6

logs 디렉터리에 있는 파일들을 모두 지워서 알티베이스가 **startup** 되지 않는 것 같아서 **db**s 디렉터리의 파일까지 모두 지우고 다시 **createdb** 로 새 **db**를 생성하였습니다. 이렇게 하면 기존의 **db**가 모두 삭제되고 새로운 **db**가 제대로 생성되는지 알고 싶습니다.

답변

데이터 파일과 로그파일, 로그엔커 파일들은 데이터베이스를 구성하는 중요한 파일들로서, 하나라도 없으면 데이터베이스 구성이 안 되는 파일들입니다. 만약 이중에 하나라도 어쩌다가 지우셨다면, 데이터베이스를 재 구성하기 위해서는, 말씀하신 것처럼, **db**s 디렉터리 밑에 데이터 파일들과, **logs** 디렉터리 밑에 로그파일 및 로그엔커파일, 그리고, 만약 **shared memory**를 사용하셨다면, 메모리에 있는 **altibase**용 **shared memory**를 모두 삭제하고, **createdb**로 새 데이터베이스를 만드시면 됩니다.

예를 들면 다음과 같습니다.

1. **rm logs/***
2. **rm dbs/***
3. **shared memory**를 사용했을 경우, **ipcrm -m <Altibase db용 shared memory id>**

Properties 관련 FAQ

질문1

2G가 넘는 **data**를 하드 디스크에서 **memory**로 **loading**시의 소요시간이 2G 이하일 때보다 많이 소요된다고 하셨는데, 5-6G의 **data**를 **loading** 시키려면 어떻게 해야 합니까?

답변

Chunk를 해서 올리는 방법이 있습니다.

Chunk는 **shared memory**에 **DB**를 생성했을 때, 가능한 방법입니다. **Process memory**에 **DB**를 생성했을 때는 전체를 로딩하는 방법 외에는 없습니다. **Shared memory**에 **DB**를 생성했을 때의 이점은, **DB**가 비정상 종료 시에도 데이터가 **shared memory** 영역에 그대로 남아있기 때문에 **startup** 할 때, 속도가 빠릅니다.

Shared memory로 **DB**를 생성 후 **chunk size**를 설정하는 방법은

\$ALTIBASE_HOME/conf/altibase.properties 파일에 있는 프로퍼티 **STARTUP_SHM_CHUNK_SIZE = 2G**로 설정하면 됩니다.

Memory, CPU 관련 FAQ

질문1

향후 메모리부족이 발생할 때를 대비할 방법은?

답변

full scan을 하는 응용프로그램을 가능한 full scan을 하지 않도록 수정하는 방법과 db size를 줄이거나, 하드웨어 메모리를 늘리는 방법 등이 있습니다.

질문2

CPU 점유율이 13% ~ 25%까지 너무 많이 차지하고 있습니다.

답변

많은 이유가 있습니다. 클라이언트와 서버가 통신할 때의 세션 연결 타입에 따라서 (속도: IPC > Unix Domain > TCP/IP), 그리고 PREPARE_LOG_FILE_COUNT가 작게 설정이 되 있을 때, 또는 index를 사용하지 않는 SQL문이 있을 때 등인 경우 CPU 점유율이 높게 나옵니다. 클라이언트 프로그램과의 연결 세션이 많을 경우 역시 CPU를 많이 점유합니다.

질문3

메모리에서의 table이 차지하는 공간을 알고 싶습니다.

답변

dbadmin을 수행하고 connect한 후 server status all command를 사용하면 알 수가 있습니다. 그리고 index size는 다른 disk기반 database와는 다르게 key value를 가져가는 것이 아니고 pointer만 가져 갑니다. 이것의 공간은 index node별로 64개의 key 값을 저장할 수 있습니다. 하나의 node는 (72byte + 64*8 byte)의 크기를 가집니다. 즉, 64 record를 가진 table에 어떤 index가 존재할 경우 이 index가 차지하는 공간은 하나의 node 584 byte입니다. 따라서 이 table이 100만 record라면 9125000 bytes의 공간을 차지합니다. index가 두개라면 2배의 공간을 필요로 하겠죠.

JDBC 관련 FAQ

질문1

JDBC 드라이버를 설치하고 sample 디렉터리에 있는 java sample을 돌렸는데 에러가 납니다.

답변

서버와 JDBC 드라이버간의 버전이 맞지 않거나 이전에 자바의 버전이 낮게 설정된 path를 그대로 이용하고 있어서 정상 작동을 안 할 수 있습니다.

질문2

Altibase의 홈페이지에서 Altibase.jar (JDBC)를 다운 받으려고 하니 두개의 버전이 있던데요. (Altibase & Altibase2) 두 버전의 차이점이 무엇인지요?

답변

Altibase와 Altibase2 두 버전에는 메타 테이블이 약간 바뀌었습니다.

질문3

Altibase에서 제공하는 JDBC 드라이버는 버전이 어떻게 되나요?

답변

스펙버전에서는 JDBC 1.0은 모두 지원하고 JDBC 2.0은 일부만 지원합니다.

Altibase는 JDBC 버전이 따로 없고, 서버에 같이 따라 나갑니다.

질문4

현재의 JDBC Driver는 J2EE 1.3 스펙을 지원하나요?

답변

지원 안 합니다.

타 DB 관련 FAQ

질문1

Altibase와 Oracle의 연동 시 두 소프트웨어는 같은 시스템에 있어야 하나요?

답변

Oracle과 Altibase는 같은 시스템에 있을 필요가 없습니다. Oracle로의 접속은 Oracle에서 지원하는 sqlnet을 이용하면 됩니다. 그리고 Altibase로의 접속은 TCP/IP 방식을 이용하면 client program이 Altibase DB server가 있는 곳에 있을 필요는 없습니다.

질문2

한 Program에서 Proc와 SES를 동시에 이용할 수 있나요?

답변

네, 이용할 수 있습니다. 다만 제약점은 둘 다 precompile을 하여야 하고 구문이 비슷하기 때문에 file은 따로 존재하여야 하며 나중에 link시 각각의 object를 link하면 됩니다.

질문3

Daemon Program의 위치는 어느 server에 두어야 하나요?

답변

Daemon Program의 위치는 Oracle이 있는 server에 두어도 되고 Altibase Server가 있는 곳에 있어도 됩니다. 다만 Altibase가 존재하는 곳에 두면 이곳에 Oracle Client Module이 install되어야 합니다. (Proc까지는 필요 없을 것 같습니다. Compile은 Proc가 있는 곳에서 하면 되니까요).

질문4

사용자 인증 및 세션 관리는 Altibase를 쓰고 실제 사용자의 가입자 정보 및 과금 정보는 Oracle을 사용할 때, Altibase 내의 테이블에 trigger를 걸어서 테이블이 변경되는 내용을 Oracle 내의 다른 테이블에 반영시키고 싶은 데 이게 가능한가요?

답변

알티베이스는 trigger를 제공하지 않습니다.

그러므로, Altibase와 Oracle의 연동 시 Altibase CLI program으로 반영시키는 방법을 제공하고 있습니다.

ODBC 관련 FAQ

질문1

PC에 C: 드라이브가 없고, D:만 있을 경우, ODBC 설치프로그램이 동작중 오류를 발생합니다.

답변

ODBC 설치 프로그램이 동작중 레지스트리 등록 작업을 수행합니다. 이때에 c: 드라이브를 가정하고, DLL 파일의 위치를 등록하기 때문에 이런 오류 메시지가 발생합니다. 일단, 이러한 경우에는 “무시” 버튼을 눌러 일단 설치를 완료합니다. 이후에 regedit.exe 라는 레지스트리 에디트 프로그램을 수행하여, HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\로 들어가서 Altibase3_cmXXX 항목에 등록되어 있는 Driver, Setup 항목의 값을 자신의 시스템에 맞춰 변경 등록합니다.

기타 FAQ

질문1

DB 업그레이드, 옵티마이저에 관한 질문

답변

DB의 버전 명이 다르면 데이터호환이 되지 않습니다. 따라서, DB 업그레이드 시 데이터를 초기화 하시고 (destroydb, createdb) 서버를 재기동 하시면 됩니다.

그리고 옵티마이저에 대해서 **cost-based optimization**을 제공합니다.

질문2

실행계획 보는 법

답변

Altibase 3에서는 이 기능을 지원합니다. ALTIBASE_iSQL User's Manual 내에 **Explain Plan**을 참고하세요.

질문3

기존 LDAP 기반 **Directory** 서비스와의 호환성, X.509 기반의 인증 **Mechanism** 지원, 사용자 ID, IP주소 또는 도메인 네임에 기반한 액세스 제어

답변

현재 **OpenLDAP** 지원으로 표준과 호환됩니다.

찾아보기

ㄱ

공유메모리 관리도구	8-17
관리 도구	1-4
기타 FAQ	10-7

ㄴ

다국어	4-9
데이터베이스 구성 프로퍼티	4-3
데이터베이스 구조	2-3
데이터베이스 삭제	2-14
데이터베이스 생성	2-2
데이터베이스 이중화	7-2
데이터베이스 이중화 프로퍼티	4-16
디스크 고장	5-2

ㄷ

로그	5-2
로그앵커	5-3
로그에 따른 체크포인트	5-4
로깅 프로퍼티	4-13

ㄹ

메시지 로그 프로퍼티	4-20
메타 테이블	2-5

ㅁ

백업 데이터 파일	5-2
복구	5-2, 5-7

ㅂ

세션 연결 프로퍼티	4-8
스냅샷	5-2

시간 주기 체크포인트	5-4
시스템 고장	5-2

ㅇ

아카이브 로그 백업	5-10
알티베이스 디렉토리	1-6
알티베이스 프로퍼티	4-2
알티베이스 프로퍼티 파일	1-7
알티베이스 홈 디렉터리	1-6
알티베이스의 실행	3-2
알티베이스의 종료	3-5
오프라인 백업	5-9
온라인 백업	5-6
응용 프로그램 작성	6-2
인덱스 타임 프로퍼티	4-22
임의적 체크포인트	5-4

ㅊ

지원 도구	1-5
-------	-----

ㅋ

체크포인트	5-4
-------	-----

꺄

쿼리 최적화 프로퍼티	4-21
-------------	------

ㅌ

타 DB 관련 FAQ	10-6
테이블별 백업	5-9
통신 버퍼	4-9
통신 버퍼 크기	4-8
통신 채널	4-9
트랜잭션 고장	5-2

트랜잭션 프로퍼티	4-11
트랜잭션의 철회	5-2

II

퍼지&핑퐁 체크포인트	5-4
페이지	2-2, 2-3

A

abort	3-5, 8-6
active 로그	5-2
admin 디렉터리	1-9
ADMIN_MODE	4-23
AGER_WAIT_MAXIMUM	4-5
AGER_WAIT_MINIMUM	4-5
ALTER REPLICATION START	7-4
ALTER REPLICATION STOP	7-7
Altibase CLI	6-3
Altibase connect	8-3
Altibase disconnect	8-3
Altibase status	8-6
Altibase status all	8-7
Altibase.jar	1-7
altibase.properties	1-7
altibase_boot.log	1-6
arch_logs 디렉터리	1-9
archive log backup	5-10
ARCHIVE_DIR	4-13
ARCHIVE_FULL_ACTION	4-13
ARCHIVE_THREAD_ENABLED	4-13
audit 디렉터리	1-7
AUTO_COMMIT	4-11
AUTO_REMOVE_ARCHIVE_LOG	4-14

B

bin 디렉터리	1-6
----------	-----

C

CHECK_POINT_ENABLED	4-14
---------------------	------

CHECK_POINT_INTERVAL_IN_LOG	4-14
CHECK_POINT_INTERVAL_IN_SEC	4-14
checkServer	1-5
classpath	6-10
CM_BUF_SIZE	4-8
CM_DISCONN_DETECT_TIME	4-8
CM_DISCONN_HIGHWATER_MARK	4-8
conf 디렉터리	1-6
configuration	4-2
CREATE REPLICATION	7-4
createdb	2-2

D

DATABASE_IO_TYPE	4-5
DB_FILE_SIZE	4-3
DB_NAME	4-3
DB0_DIRR	4-3
DB1_DIRR	4-3
dbadmin	1-4, 3-2, 8-2
dbs 디렉터리	1-8
DDL_LOCK_TIMEOUT	4-5
destroydb	2-14
DROP REPLICATION	7-8

E

ERROR_MSG_NLS	4-8
EXEC_DDL_DISABLE	4-23

F

FAQ	9-2, 10-2
FETCH_TIMEOUT	4-9
FIFO_QUEUE	4-23

I

IDLE_TIMEOUT	4-10
iload	1-5
immediate	3-5, 8-6

찾아보기

include 디렉터리	1-7
index rebuilding	4-6
INDEX_BUILD_THREAD_COUNT	4-22
INDEX_TYPE	4-22
install 디렉터리	1-9
IPC_CHANNEL_BUF_COUNT	4-8
IPC_CHANNEL_COUNT	4-9
IPC_CHANNEL_RETRY_COUNT	4-9
ISOLATION_LEVEL	4-11
isql	1-5

J

JDBC	6-10
JDBC 관련 FAQ	10-5
JDBC 드라이버	6-10

K

killCheckServer	1-5
KO16KSC5601	1-8

L

lib 디렉터리	1-7
LOCK_ESCALATION_MEMORY_SIZE	4-5
LOG_DIR	4-3
LOG_FILE_SIZE	4-3
loganchor	1-9
logancor	5-3
LOGGING_LEVEL	4-14
logs 디렉터리	1-9

M

MAX_CLIENT	4-3
MAX_DB_SIZE	4-3
MAX_LISTEN	4-3
MAX_QUEUE_COUNT	4-23
MAX_THREAD	4-3
Memory, CPU 관련 FAQ	10-5
msg 디렉터리	1-8

MSGLOG_FILE_SIZE	4-20
MSGLOG_MAX_FILE	4-20

N

NLS_USE	4-9
normal	3-5, 8-5
NORMALFORM_MAXIMUM	4-21

O

ODBC	6-14
ODBC 드라이버	6-14
Offline backups	5-9
online backups	5-6
OPEN_LOG_FILE_COUNT	4-15
OPTIMIZER_MODE	4-21

P

page	2-2, 2-3
PARALLEL_LOAD_FACTOR	4-6
PERS_PAGE_CHUNK_COUNT	4-4
PORT_NO	4-9
PREPARE_LOG_FILE_COUNT	4-15
Properties 관련 FAQ	9-2, 10-4
pthread	9-3

Q

QUERY_TIMEOUT	4-10
---------------	------

R

REFINE_PAGE_COUNT	4-6
REPLICAION_LOCK_TIMEOUT	4-16
REPLICATION_	
HBT_DETECT_HIGHWATER_MARK	4-16
REPLICATION_HBT_DETECT_TIME	4-16
REPLICATION_RECEIVE_TIMEOUT	4-18
REPLICATION_	
SENDER_AUTO_START	4-18

REPLICATION_	
SENDER_SLEEP_TIMEOUT	4-18
REPLICATION_CONNECT_TIMEOUT4-	
16	
REPLICATION_ENABLE_ADD_COLU	
MN	4-16
REPLICATION_GET_ACK	4-16
REPLICATION_MAX_LOGFILE	4-17
REPLICATION_PORT_NO	4-17
REPLICATION_PROPAGATION	4-17
REPLICATION_SYNC_LOCK_TIMEOU	
T	4-18
REPLICATION_SYNC_LOG	4-18
REPLICATION_SYNC_MAX_LIMIT	4-
18	
REPLICATION_UPDATE_REPLACE	4-
19	
RESTORE_CHUNK_PAGE_SIZE	4-6
RESTORE_METHOD_	4-6
restoredb	1-5, 5-7
rollback	5-2

S

sample 디렉터리	1-9
SELECT_HEADER_DISPLAY	4-23
server stop	3-6
SES 관련 FAQ	10-8
shell	8-15
SHM_DB_KEY	4-4
shmutil	1-5, 8-17
shutdown	3-5, 8-4
startup	3-2, 8-4
Startup 관련 FAQ	9-2, 10-2
STARTUP_SHM_CHUNK_SIZE	4-4
SUPPORT_SELECTIVE_LOADING	4-24
SYS_COLUMNS_	2-5
SYS_CONSTRAINT_COLUMNS_	2-6
SYS_CONSTRAINTS_	2-5

SYS_DATABASE_	2-6
SYS_GRANT_OBJECT_	2-7
SYS_GRANT_SYSTEM_	2-7
SYS_INDEX_COLUMNS_	2-7
SYS_INDICES_	2-8
SYS_PROC_PARAS_	2-9
SYS_PROC_PARSE_	2-9
SYS_PROC_RELATED_	2-10
SYS_PROCEDURES_	2-8
SYS_REPL_HOSTS	2-11
SYS_REPL_ITEMS	2-11
SYS_REPLICATIONS_	2-10
SYS_TABLES_	2-11
SYS_USERS_	2-12
SYS_VIEW_PARSE_	2-12
SYS_VIEW_RELATED_	2-12
SYS_VIEWS_	2-12
SYS_XA_HEURISTIC_TRANS_	2-13
SYS_XA_PRIVILEGES_	2-8

T

Table backups	5-9
TEMP_PAGE_CHUNK_COUNT	4-4
THREAD_POOL	4-4
TRANSACTION_DURABILITY_LEVEL	4-11
TRANSACTION_TABLE_SIZE	4-11
trc 디렉터리	1-6
TRC_DIR	4-4
TRCLOG_LEVEL	4-24

U

unique ID	8-5
UPDATE_IN_PLACE	4-6
US7ASCII	1-8
UTRANS_TIMEOUT	4-10

찾아보기