

- Utilities Manual
 - 서문
 - 이 매뉴얼에 대하여
 - 1.aexport
 - aexport소개
 - aexport사용방법
 - 2.altiComp
 - altiComp 소개
 - altiComp 사용 방법
 - 비교(DIFF)기능
 - 일치(SYNC) 기능
 - 3.dataCompJ
 - dataCompJ 소개
 - dataCompJ 사용 방법
 - 비교(DIFF) 기능
 - 일치(SYNC) 기능
 - 이기종간 데이터 타입 호환 테이블
 - 4.기타 Utilities
 - altiAudit
 - altibase
 - altiMon
 - altierr
 - altipasswd
 - altiProfile
 - altiwrap
 - awrite
 - checkServer
 - dumpbi
 - dumpct
 - dumpdb
 - dumpddf
 - dumpla
 - dumplf
 - dumptrc
 - killCheckServer
 - server

Utilities Manual



Altibase Tools & Utilities Utilities Manual

Release 7.1

Copyright © 2001~2019 Altibase Corp. All Rights Reserved.

본 문서의 저작권은 (주)알티베이스에 있습니다. 이 문서에 대하여 당사의 동의 없이 무단으로 복제 또는 전용할 수 없습니다.

(주)알티베이스

08378 서울시 구로구 디지털로 306 대륭포스트타워II 10층

전화: 02-2082-1114 팩스: 02-2082-1099

고객서비스포털: <http://support.altibase.com>

homepage: <http://www.altibase.com>

서문

이 매뉴얼에 대하여

이 매뉴얼은 Altibase를 사용하면서 필요한 유틸리티에 대해서 설명한다.

대상 사용자

이 매뉴얼은 다음과 같은 Altibase 사용자를 대상으로 작성되었다.

- 데이터베이스 관리자
- 성능 관리자
- 데이터베이스 사용자
- 응용 프로그램 개발자
- 기술지원부

다음과 같은 배경 지식을 가지고 이 매뉴얼을 읽는 것이 좋다.

- 컴퓨터, 운영 체제 및 운영 체제 유틸리티 운용에 필요한 기본 지식
- 관계형 데이터베이스 사용 경험 또는 데이터베이스 개념에 대한 이해

- 컴퓨터 프로그래밍 경험
- 데이터베이스 서버 관리, 운영 체제 관리 또는 네트워크 관리 경험

소프트웨어 환경

이 매뉴얼은 데이터베이스 서버로 Altibase 버전 7.1을 사용한다는 가정 하에 작성되었다.

이 매뉴얼의 구성

이 매뉴얼은 다음과 같이 구성되어 있다.

- 제 1장 aexport
이 장은 Altibase 데이터베이스 간 자동화된 데이터 마이그레이션(migration)을 지원하기 위한 도구인 aexport에 대해서 설명한다.
- 제 2장 altiComp
이 장은 altiComp 유틸리티의 기능을 소개하고, 불일치하는 데이터를 비교하고 일치시키는 기능 등을 설명한다.
- 제 3장 dataCompJ
Altibase 데이터베이스에서 이기종 데이터베이스로 데이터를 복제할 때 데이터의 정합성 확인과 불일치 해소 기능을 제공하는 dataCompJ를 설명한다.
- 제 4장 기타 Utilities
이 장은 aexport, altiComp, dataCompJ 등을 제외한 나머지 유틸리티에 대해서 설명한다.

문서화 규칙

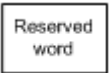
이 절에서는 이 매뉴얼에서 사용하는 규칙에 대해 설명한다. 이 규칙을 이해하면 이 매뉴얼과 설명서 세트의 다른 매뉴얼에서 정보를 쉽게 찾을 수 있다.



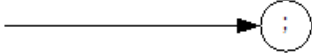
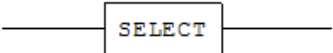
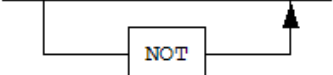
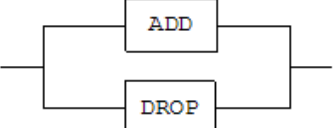
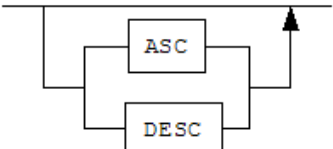
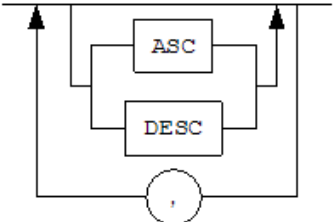
여기서 설명하는 규칙은 다음과 같다.

- 구문 다이어그램
- 샘플 코드 규칙

구문 다이어그램

이 매뉴얼에서는 다음 구성 요소로 구축된 다이어그램을 사용하여, 명령문의 구문을 설명한다.

구성 요소	의미
	명령문이 시작한다. 완전한 명령문이 아닌 구문 요소는 화살표로 시작한다.

구성 요소	의미
	명령문이 다음 라인에 계속된다. 완전한 명령문이 아닌 구문 요소는 이 기호로 종료한다.
	명령문이 이전 라인으로부터 계속된다. 완전한 명령문이 아닌 구문 요소는 이 기호로 시작한다.
	명령문이 종료한다.
	필수 항목
	선택적 항목.
	선택사항이 있는 필수 항목. 한 항목만 제공해야 한다.
	선택사항이 있는 선택적 항목
	선택적 항목. 여러 항목이 허용된다. 각 반복 앞부분에 콤마가 와야 한다.

샘플 코드 규칙

코드 예제는 SQL, Stored Procedure, iSQL 또는 다른 명령 라인 구문들을 예를 들어 설명한다.

아래 테이블은 코드 예제에서 사용된 인쇄 규칙에 대해 설명한다.

규칙	의미	예제
[]	선택 항목을 표시	VARCHAR [(size)] [[FIXED]] VARIABLE]
{ }	필수 항목 표시. 반드시 하나 이상을 선택해야 되는 표시	{ ENABLE DISABLE COMPILE }

규칙	의미	예제
	선택 또는 필수 항목 표시의 인자 구분 표시	{ ENABLE DISABLE COMPILE } [ENABLE DISABLE COMPILE]
...	그 이전 인자의 반복 표시 예제 코드들의 생략되는 것을 표시	SQL> SELECT ename FROM employee; ENAME ----- SWNO HJNO HSCHOI . . . 20 rows selected.
그 밖에 기호	위에서 보여진 기호 이 외에 기호들	EXEC :p1 := 1; acc NUMBER(11,2);
기울임 꼴	구문 요소에서 사용자가 지정해야 하는 변수, 특수한 값을 제공해야만 하는 위치 지정자	SELECT * FROM <i>table_name</i> ; CONNECT userID/password;
소문자	사용자가 제공하는 프로그램의 요소들, 예를 들어 테이블 이름, 칼럼 이름, 파일 이름 등	SELECT ename FROM employee;
대문자	시스템에서 제공하는 요소들 또는 구문에 나타나는 키워드	DESC SYSTEM_.SYS_INDICES_;

관련 자료

자세한 정보를 위하여 다음 문서 목록을 참조하기 바란다.

- Installation Guide
- Administrator's Manual
- Replication Manual
- iSQL User's Manual
- iLoader User's Manual

Altibase는 여러분의 의견을 환영합니다.

이 매뉴얼에 대한 여러분의 의견을 보내주시기 바랍니다. 사용자의 의견은 다음 버전의 매뉴얼을 작성하는데 많은 도움이 됩니다. 보내실 때에는 아래 내용과 함께 고객센터포털(<http://support.altibase.com/kr/>)로 보내주시기 바랍니다.

- 사용 중인 매뉴얼의 이름과 버전
- 매뉴얼에 대한 의견
- 사용자의 성함, 주소, 전화번호

이 외에도 Altibase 기술지원 설명서의 오류와 누락된 부분 및 기타 기술적인 문제들에 대해서 이 주소로 보내주시면 정성껏 처리하겠습니다. 또한, 기술적인 부분과 관련하여 즉각적인 도움이 필요한 경우에도 고객센터포털을 통해 서비스를 요청하시기 바랍니다.

여러분의 의견에 항상 감사드립니다.

1.aexport

aexport소개

개요

aexport는 Altibase 데이터베이스 간의 자동화된 데이터 마이그레이션(migration)을 지원하기 위한 도구이다. aexport는 데이터베이스의 논리적인 구조 및 데이터를 텍스트로 저장하고, 이를 다시 새로운 Altibase 데이터베이스로 로딩하기 위한 스크립트를 자동으로 생성하여 준다.

aexport가 접속한 데이터베이스로부터 추출할 수 있는 객체 및 구성 요소는 데이터베이스 사용자, 사용자 권한, 테이블, 테이블스페이스, 테이블 제약조건, 인덱스, 뷰, Materialized View, 저장 프로시저, 시퀀스, 그리고 이중화 객체이다.

aexport는 데이터베이스의 논리적인 구조를 SQL 스크립트로 변환하여 저장하고, 모든 데이터를 텍스트로 내려 받기 때문에 상이한 Altibase 버전간, 그리고 상이한 플랫폼 간의 데이터 이동 시에 유용하게 사용될 수 있다. Altibase 서버는 구동했지만 서비스는 하지 않는 상태 즉, 클라이언트 연결이 없을 때 aexport를 사용하기를 권장한다.

aexport 기능 소개

aexport가 추출할 수 있는 데이터베이스 객체 및 구성 요소는 다음과 같다.

- 데이터베이스 사용자
- 사용자 권한
- 롤(Role)
- 테이블스페이스
- 테이블
- 테이블 제약 조건
- 인덱스
- 뷰
- Materialized View

- 저장 프로시저
- 이중화 객체

aexport를 수행하면 위에 열거한 데이터베이스 구성 요소를 생성하기 위한 SQL 스크립트와 이를 구동하기 위한 쉘 스크립트가 생성된다.

aexport 모드와 SQL 스크립트 파일

aexport는 데이터베이스의 어떤 부분을 추출하는지에 따라서 다른 모드로 실행할 수 있다. 모드는 aexport 실행 시 커맨드 라인에서 지정할 수 있다.

aexport 실행 모드와 각 모드 별로 생성되는 SQL 스크립트 파일은 아래의 절에서 설명한다.

전체DB 모드

이 모드는 전체 데이터베이스를 추출한다. SYS 사용자만이 이 모드로 aexport를 실행할 수 있다.

이 모드로 aexport를 실행할 때 생성되는 SQL 스크립트 파일은 아래와 같다.

- SYS_CRT_DIR.sql: 모든 디렉토리 객체 생성
- SYS_CRT_USER.sql: 모든 사용자와 롤(Role) 생성
- SYS_CRT_SYNONYM.sql: 모든 시노님(Synonym) 객체 생성
- SYS_CRT_REP.sql: 모든 이중화 객체 생성
- ALL_CRT_VIEW_PROC.sql: 모든 뷰와 프로시저 생성
- ALL_CRT_TBS.sql: 모든 테이블스페이스 생성
- ALL_CRT_TBL.sql: 모든 사용자 테이블 생성
- ALL_CRT_INDEX.sql: 모든 사용자 인덱스 생성
- ALL_CRT_FK.sql: 모든 사용자의 외래 키 생성
- ALL_CRT_TRIG.sql: 모든 사용자의 트리거 생성
- ALL_CRT_SEQ.sql: 모든 사용자의 시퀀스 생성
- ALL_CRT_LINK.sql: 모든 사용자의 데이터베이스 Link 생성
- ALL_EXE_STATS.sql: 모든 사용자의 테이블, 칼럼, 인덱스의 통계 정보 생성
- ALL_REFRESH_MVIEW.sql: 모든 사용자의 Materialized View를 리프레쉬
- ALL_ALT_TBL.sql : 모든 사용자의 테이블과 파티션에 대한 접근 모드를 변경

주의: 롤(Role)은 비스키마 객체이므로 전체 DB 모드에서만 추출할 수 있다.

사용자 모드

이 모드는 옵션으로 지정하는 사용자가 소유한 모든 객체를 추출한다. SYS 사용자와 옵션으로 지정하는 사용자만이 이 모드로 aexport를 실행할 수 있다. 사용자 모드로

aexport를 실행하려면, -u 커맨드라인 옵션에 사용자 이름을 지정하면 된다.

이 모드로 실행할 때 생성되는 SQL 스크립트 파일은 아래와 같다.

- {사용자명}_CRT_TBL.sql: 지정한 사용자의 모든 테이블 생성
- {사용자명}_CRT_INDEX.sql: 지정한 사용자의 모든 인덱스 생성
- {사용자명}_CRT_FK.sql: 지정한 사용자의 모든 외래 키 생성
- {사용자명}_CRT_TRIG.sql: 지정한 사용자의 모든 트리거 생성
- {사용자명}_CRT_SEQ.sql: 지정한 사용자의 모든 시퀀스 생성
- {사용자명}_CRT_LINK.sql: 지정한 사용자의 모든 데이터베이스 Link 생성
- {사용자명}_EXE_STATS.sql : 지정한 사용자의 모든 테이블, 칼럼, 인덱스의 통계 정보를 설정
- {사용자명}_REFRESH_MVIEW.sql: 지정한 사용자의 모든 Materialized View를 리프레쉬
- {사용자명}_ALT_TBL.sql : 지정한 사용자의 테이블과 파티션에 대한 접근 모드를 변경

객체 모드

이 모드는 옵션으로 지정하는 모든 객체들을 추출한다. 객체 모드로 aexport를 실행하려면, -object 커맨드라인 옵션에 추출하고자 하는 객체들을 *user.object* 의 형태로 쉼표(",")로 분리 하여 명시하면 된다 (공백문자가 포함되면 안됨). SYS 사용자와 지정하는 객체의 소유자만이 이 모드로 aexport를 실행할 수 있다.

지정한 모든 객체의 소유자는 같아야 한다. 그러나, SYS 사용자로 실행할 때는 지정한 객체들의 소유자가 각각 다른 사용자여도 무방하다.

이 모드로 실행할 때 생성되는 SQL 스크립트 파일은 아래와 같다.

- {사용자명}_{객체명}_CRT.sql: 지정한 사용자 객체 생성
- {사용자명}_{객체명}_STATS.sql: 지정한 사용자 통계 정보 생성

셸 스크립트 파일

위에서 기술한 SQL 스크립트 파일 외에, aexport 수행 시 생성되는 셸 스크립트 파일들은 다음과 같다.

- run_il_in.sh: 데이터 로드 스크립트
- run_il_out.sh: 데이터 다운로드 스크립트
- run_is.sh: 스키마 생성 스크립트
- run_is_con.sh: constraint 생성 스크립트. TWO_PHASE_SCRIPT 프로퍼티를 ON으로 설정하면, 이 셸 스크립트 파일이 생성된다. 이 파일은 인덱스, 외래 키, 트리거, 및 이중화 객체를 생성하는 SQL 스크립트를 포함한다.

- run_is_fk.sh: 외래키와 트리거 생성 스크립트. TWO_PHASE_SCRIPT 프로퍼티를 ON으로 설정하면, 이 쉘 스크립트 파일은 생성되지 않는다.
- run_is_index.sh: 인덱스 생성 스크립트. TWO_PHASE_SCRIPT 프로퍼티를 ON으로 설정하면, 이 쉘 스크립트 파일은 생성되지 않는다.
- run_is_repl.sh: 이중화 생성 스크립트. TWO_PHASE_SCRIPT 프로퍼티를 ON으로 설정하면, 이 쉘 스크립트 파일은 생성되지 않는다.
- run_is_refresh_mview.sh: Materialized View를 refresh하는 스크립트. TWO_PHASE_SCRIPT 프로퍼티를 ON으로 설정하면, 이 쉘 스크립트 파일은 생성되지 않는다.
- run_is_alt_tbl.sh : 테이블과 파티션에 대한 접근 모드를 변경하는 스크립트 (TWO_PHASE_SCRIPT 프로퍼티를 ON으로 설정하면, 이 쉘 스크립트 파일은 생성되지 않는다.).

aexport 수행 후 생성된 쉘 스크립트를 대상 데이터베이스에 대해 실행하면, 데이터베이스의 논리적 구조가 자동으로 생성된다. 또한 기존의 데이터가 모두 대상 데이터베이스에 로딩된다. 쉘 스크립트는 Altibase iLoader를 사용하여 데이터 다운로드 및 업로드를 수행한다. 쉘 스크립트 내에서 iLoader를 사용하는 과정은 완전히 자동화되어 있기 때문에 사용자가 iLoader의 사용법에 익숙하지 않아도 사용이 가능하다.

aexport에 의해 생성된 모든 파일은 텍스트 파일이기 때문에 사용자의 필요에 따라 변경 후 사용이 가능하므로, 좀 더 유연하게 활용할 수 있다.

aexport 프로퍼티와 스크립트 파일

이 절은 aexport 프로퍼티 설정에 따라서 생성되는 스크립트 파일에 대해서 설명한다.

프로퍼티에 대한 상세한 설명은 “aexport 프로퍼티” 절을 참조한다.

- INVALID_SCRIPT = ON 일 때, INVALID.sql 파일이 생성된다. 이 파일은 유효하지 않은 모든 뷰와 저장 프로시저를 생성하는 SQL 스크립트를 포함하는데, 이 파일을 실행하는 쉘 스크립트 파일은 생성되지 않는다.
- TWO_PHASE_SCRIPT = ON 일 때, ALL_OBJECT.sql 파일과 ALL_OBJECT_CONSTRAINTS.sql 파일이 생성된다. ALL_OBJECT.sql 파일은 모든 객체를 생성하는 SQL 스크립트를, ALL_OBJECT_CONSTRAINTS.sql 파일은 모든 인덱스, 외래키, 트리거와 이중화 객체를 생성하는 SQL 스크립트를 포함한다. 또한, ALL_OBJECT_CONSTRAINTS.sql을 실행하는 run_is_con.sh 쉘 스크립트 파일이 생성된다.

aexport 설정

aexport는 서버에 접속하기 위해서 다음과 같은 정보가 필요하다.

- ALTIBASE_HOME
서버 혹은 클라이언트가 설치된 경로
- server_name
Altibase 서버가 구동되어 있는 컴퓨터 서버의 이름(또는 IP 주소)
- port_no
TCP 또는 IPC로 접속할 때 사용할 포트 번호
- user_id
데이터베이스에 등록된 사용자 ID
- Password
사용자 ID와 일치하는 암호
- NLS_USE
데이터 검색 시, 사용자에게 보여주는 문자 집합

ALTIBASE_HOME은 환경 변수로 설정하도록 되어 있으며, 나머지는 커맨드 라인 옵션을 통해서 설정할 수 있다.(자세한 내용은 “aexport 사용방법”을 참고한다.)

aexport를 실행하려면 ALTIBASE_HOME 환경 변수와 aexport 설정 파일(aexport.properties)이 필요하다. 실행 전에 ALTIBASE_HOME이 바르게 설정되어 있어야 하며, 샘플로 제공되는 aexport.properties.sample 파일을 이용해 aexport용 설정 파일을 생성해 두어야 한다.

ALTIBASE_HOME은 일반적으로 서버가 설치될 때 자동으로 설정되는데 클라이언트의 경우에는 사용자가 직접 설정해야 한다. 설정되지 않았을 경우에는 제대로 동작하지 않을 수 있으므로 실행 전에 바르게 설정되어 있는지 확인할 것을 권장한다.

port_no와 NLS_USE는 환경 변수 또는 서버 설정 파일(altibase.properties)을 이용해서 설정할 수도 있다. 세 가지 방법으로 모두 설정되어 있을 경우 적용 우선 순위는 다음과 같다.

1. 커맨드 라인 옵션
2. 환경 변수 (ALTIBASE_PORT_NO, ALTIBASE_NLS_USE)
3. 서버 설정 파일(altibase.properties)

그러므로 설정된 값과 다른 옵션으로 연결하고자 할 경우, 커맨드 라인 옵션을 사용하면 서버 설정 파일이나 환경 변수를 변경하지 않아도 된다.

옵션이 설정되어 있지 않을 경우에는 aexport가 처음 실행될 때 옵션 입력 프롬프트를 띄우고 사용자에게서 해당 값을 입력 받는다. 이 때 바르지 않은 형식이나 유효하지 않은 값을 입력할 경우, aexport는 제대로 동작하지 않을 수 있다.

특히 NLS_USE 옵션은 사용자가 설정하지 않았더라도 실행시에 입력 프롬프트를 띄우지 않는다. 만약 사용자가 NLS_USE 옵션을 설정하지 않았다면 현재 데이터베이스의

캐릭터 셋을 이용해 접속을 시도한다. 이 때 데이터베이스의 문자집합이 데이터베이스 캐릭터 셋이 아닐 경우에는 올바르게 실행되지 않거나 사용자 데이터가 일부 깨질 수 있으므로 반드시 NLS_USE를 사용 환경에 맞는 값으로 설정해야 한다.

원활한 aexport 사용을 위해 다음 환경 변수를 설정할 것을 권장한다.

- ALTIBASE_HOME: 서버 혹은 클라이언트가 설치된 경로
- ALTIBASE_PORT_NO: 서버에 접속할 때 사용할 포트 번호
- ALTIBASE_NLS_USE: 데이터 검색 시, 사용자에게 보여주는 문자집합
- PATH: 실행파일이 있는 경로인 \$ALTIBASE_HOME/bin 추가

환경변수

ALTIBASE_HOME

패키지가 설치된 디렉토리를 설정한다. aexport 사용을 위해 반드시 설정해야 하는 환경변수이다.

ALTIBASE_PORT_NO

접속할 서버의 포트 번호를 설정하는 환경 변수이다. -PORT 옵션 또는 altibase.properties를 통해서 지정할 수도 있다.

포트 번호 설정의 우선순위는 -PORT 옵션, 환경 변수 ALTIBASE_PORT_NO, altibase.properties 순이며 설정되지 않았을 경우에는 포트 번호 입력 프롬프트가 출력된다.

ALTIBASE_SSL_PORT_NO

aexport가 SSL/TLS 통신으로 접속할 서버의 포트 번호이다.

SSL 포트 번호 설정의 우선 순위는 -PORT 옵션, 환경변수 ALTIBASE_SSL_PORT_NO, altibase.properties 파일 내의 프로퍼티 순이다. 만약 아무것도 설정되지 않았을 경우에는 포트 번호 입력 프롬프트가 출력된다.

ALTIBASE_NLS_USE

서버에 연결할 때 사용할 캐릭터 셋을 설정한다. -NLS_USE 옵션 또는 altibase.properties를 통해서 지정할 수도 있다.

NLS_USE 설정의 우선순위는 -NLS_USE 옵션, 환경 변수 ALTIBASE_NLS_USE, altibase.properties 순이며 설정되지 않았을 경우에는 데이터베이스 캐릭터 셋을 사용한다.

주의) 서버 캐릭터 셋과 ALTIBASE_NLS_USE에 설정한 값이 다를 경우에는 정상적으로 동작하지 않을 수 있다. 반드시 적절한 값을 설정할 것을 권장한다.

ALTIBASE_UT_FILE_PERMISSION

aexport, iLoader, iSQL이 생성하는 파일들의 권한을 설정하는 공통 환경변수이다. 값을 설정하지 않으면 666 (user:rw, group:rw, other: rw)로 설정된다.

예) user:rw, group:--, other:--로 설정하는 경우,
export ALTIBASE_UT_FILE_PERMISSION=600

ISQL_FILE_PERMISSION, AEXPORT_FILE_PERMISSION, 또는 ILO_FILE_PERMISSION이 설정된 경우,
ALTIBASE_UT_FILE_PERMISSION 환경 변수 보다 우선 처리된다.

예)export ALTIBASE_UT_FILE_PERMISSION=660; export ISQL_FILE_PERMISSION=600;
iSQL에서 생성되는 파일의 권한은 ISQL_FILE_PERMISSION=600이 우선처리되어 user:rw, group:--, other:--으로 설정된다.
aexport, iloader가 생성하는 파일의 권한은 ALTIBASE_UT_FILE_PERMISSION=660에 따라 user:rw, group:rw, other:--으로 설정된다.

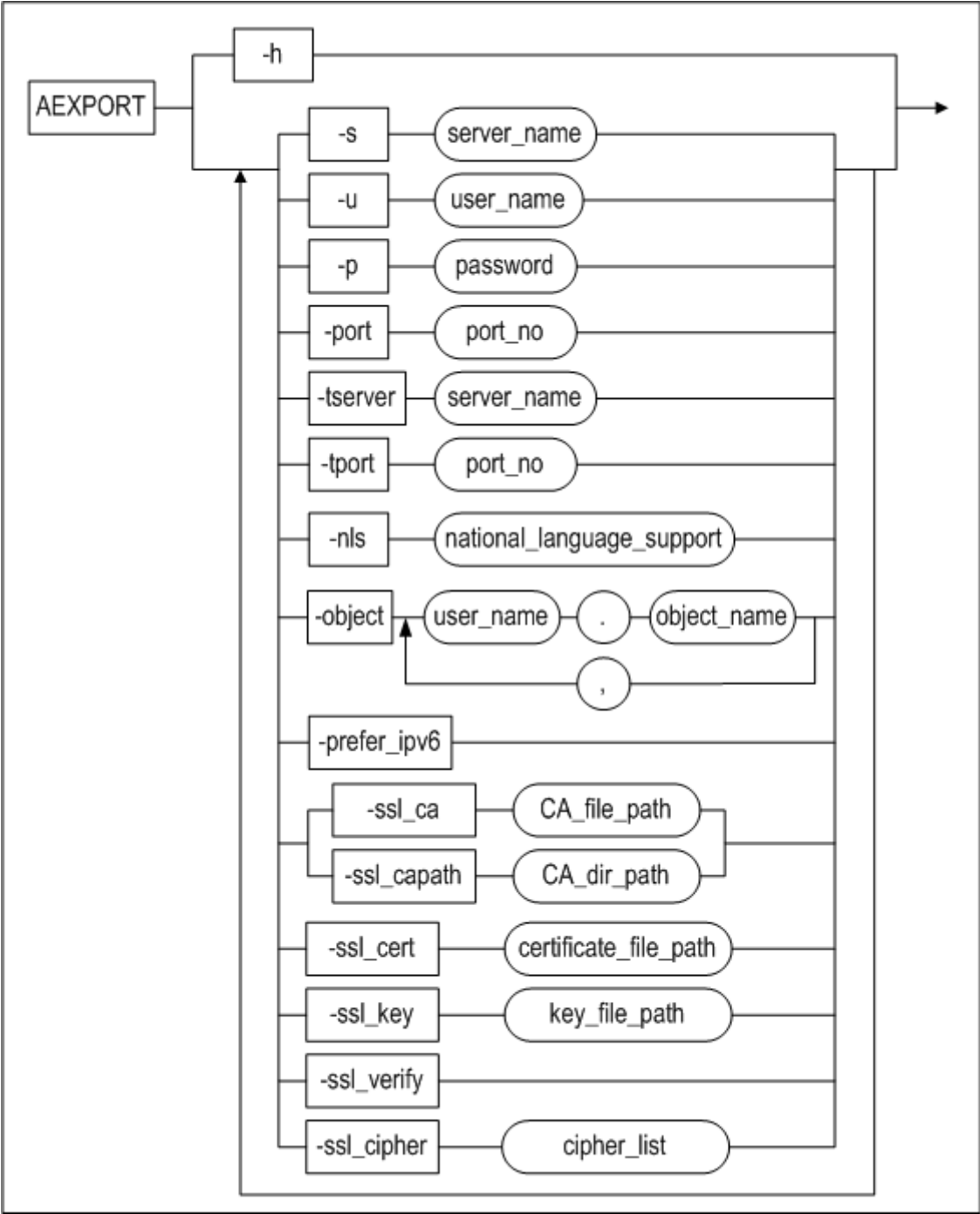
AEXPORT_FILE_PERMISSION

aexport가 생성하는 파일 권한을 설정하는 환경 변수이다. 값을 설정하지 않으면 666 (user:rw, group:rw, other: rw)로 설정된다.

예) user:rw, group:--, other:--로 설정하는 경우,
export AEXPORT_FILE_PERMISSION=600

aexport사용방법

구문



파라미터

파라미터	설명
-h	도움말을 출력한다.

파라미터	설명
-s	데이터를 다운로드할 서버의 호스트 명 또는 IP 주소를 설정한다. 생략시 호스트 명 입력 프롬프트가 출력된다. IP 주소의 경우, IPv4 주소 또는 IPv6 주소를 사용할 수 있다. IPv6 주소는 “[과 “]”로 에워싸야 한다. 예를 들어, localhost 를 명시하고자 할 때, 가능한 값은 다음과 같다. localhost (호스트 이름), 127.0.0.1 (IPv4주소), [::1] (IPv6주소) Altibase에서 IPv6 주소를 사용하는 방법에 대해서는 <i>Administrator’s Manual</i> 을 참고하기 바란다.
-u	접속할 Altibase의 사용자명을 설정한다. 생략 시 사용자명 입력 프롬프트가 출력된다. 전체 DB 모드로 실행하려면, 이 옵션에 SYS 사용자를 지정해야 한다. 사용자 이름에 소문자, 특수 문자 또는 공백이 포함된 경우 큰 따옴표를 사용해야 한다. -u \"user name\"
-p	접속할 Altibase 사용자의 패스워드를 설정한다. 생략시 패스워드 입력 프롬프트가 출력된다.
-port	접속할 Altibase의 포트번호를 설정한다. 생략 시 환경 변수 ALTIBASE_PORT_NO, altibase.properties를 차례로 참조하며 설정되어 있지 않은 경우에는 포트 번호 입력 프롬프트가 출력된다.
-object	추출할 객체를 소유자 이름과 함께 명시한다. 추출할 객체 이름에 소문자, 특수 문자 또는 공백이 포함된 경우 큰 따옴표를 사용해야 한다. -object \"user name\".\"table name\"
-tserver	추출한 데이터를 업로드할 대상 서버를 지정한다. 이 정보는 aexport가 생성하는 쉘 스크립트 파일 안에 사용된다. -s 옵션과 마찬가지로 호스트 이름, IPv4 주소 또는 IPv6 주소 모두 가능하다.
-tport	접속할 대상 서버의 포트번호를 설정한다. 이 정보는 aexport가 생성하는 쉘 스크립트 파일 안에 사용된다.
-nls_use	데이터베이스에 데이터 저장 또는 데이터 추출 시에 사용되는 클라이언트 캐릭터 셋을 지정한다 (US7ASCII, KO16KSC5601, MS949, BIG5, GB231280, MS936, UTF8, SHIFTJIS, MS932, EUCJP). 생략 시 환경 변수 ALTIBASE_NLS_USE, altibase.properties를 차례로 참조하며, 설정되지 않았을 경우에는 데이터베이스 캐릭터 셋을 사용한다.

파라미터	설명
-prefer_ipv6	-s 옵션으로 호스트 이름을 입력했을 때, 접속할 IP 주소의 버전을 결정하는 옵션이다. 이 옵션을 명시하면, 호스트 이름을 IPv6 주소로 바꾸어 접속한다. 이 옵션을 명시하지 않으면, isql은 IPv4 주소로 접속한다. 선호하는 버전의 IP 주소로의 접속이 실패하면, 다른 IP 버전 주소로 접속을 다시 시도한다. 예를 들어, -S 옵션에 "localhost"를 입력하고 이 옵션을 명시하면, isql 은 처음에 IPv6 주소인 [::1]로 접속하고, 이 접속이 실패하면 IPv4 주소인 127.0.0.1로 접속을 다시 시도한다.
-ssl_ca CA_file_path	접속할 알티베이스 서버의 공개키(public key)가 포함된 CA(인증 기관, Certification Authority) 인증서 파일의 위치를 지정한다.
-ssl_capath CA_dir_path	접속할 알티베이스 서버의 공개키가 포함된 CA 인증서 파일이 저장되어 있는 디렉토리를 지정한다.
-ssl_cert certificate_file_path	클라이언트 인증서 파일의 위치를 지정한다.
-ssl_key key_file_path	클라이언트 개인키 파일의 위치를 지정한다.
-ssl_verify	이 옵션을 지정하면 클라이언트가 서버로부터 전달받은 인증서를 검증한다.
-ssl_cipher cipher_list	SSL 암호화를 위해 사용할 알고리즘의 이름 후보들을 지정한다. General Reference에서 SSL_CIPHER_LIST 프로퍼티를 참고한다.

SSL 접속에 대한 자세한 설명과 예제는 *iSQL User's Manual*의 '접속 연결 및 해제'를 참조하기 바란다.

수행 절차

aexport를 사용한 데이터베이스 마이그레이션 절차는 크게 다음과 같이 분류된다.

- 원본 데이터베이스로부터 구조 추출
- 원본 데이터베이스로부터 데이터 추출
- 대상 데이터베이스에 데이터베이스 구조 생성
- 대상 데이터베이스에 데이터 로딩
- 대상 데이터베이스에 Materialized View를 리프레쉬한 다음, 인덱스와 외래키 생성, 접근 모드 변경

데이터베이스 구조 추출

aexport를 사용하여 데이터베이스 구조를 추출한다.

- aexport를 수행한다.

```
$ aexport -s 127.0.0.1 -u sys -p manager
```

- Altibase 데이터베이스 사용자의 암호를 입력한다. (대상 데이터베이스에서 각 사용자가 사용하게 될 암호)
- 원격지에 있는 Altibase의 데이터를 aexport를 이용하여 백업을 할 때에는 원격지 서버의 주소와 포트(PORT)를 명시한다.

```
$ aexport -s 222.112.84.200 -port 20300 -u sys -p manager
```

데이터 추출

aexport에 의하여 생성된 쉘 스크립트를 실행하여 테이블 데이터를 추출한다.

- 현재 작업중인 디스크에 충분한 공간이 있는지 확인한다. 텍스트 데이터의 경우 데이터베이스 내부 자료 형태보다 더 많은 용량을 차지할 수 있기 때문에 데이터 파일 사이즈의 두 배 가량의 여유공간 확보를 권장한다.
- "run_il_out.sh" 스크립트를 수행한다.

```
$ sh run_il_out.sh
```

대상 데이터베이스에 데이터베이스 구조 생성

- aexport와 "run_il_out.sh"에 의해 생성된 모든 SQL 스크립트와 쉘 스크립트, 그리고 확장자가 "fmt", "log", "dat"인 파일을 대상 데이터베이스가 존재하는 시스템으로 복사한다. 대상 데이터베이스가 동일한 시스템에 존재하는 경우 이 과정은 생략한다.
- 대상 데이터베이스를 구동시킨다.
- "run_is.sh" 스크립트를 수행한다.

```
$ sh run_is.sh
```

- iSQL로 데이터베이스에 접속하여 데이터베이스 구조가 올바르게 생성되었는지 확인한다. 만약, 데이터베이스 구조가 올바르게 생성되지 않았다면, run_is.sh 수행 당시의 화면출력을 검사하여 문제를 파악한다.

대상 데이터베이스에 데이터 로딩

- "run_il_in.sh" 스크립트를 수행한다.

```
$ sh run_il_in.sh
```

- 'run_il_in.sh' 디렉토리에서 확장자가 "bad"인 파일들 중 사이즈가 0이 아닌 파일이 있는지 검사한다. 해당 파일이 있을 경우 해당 테이블에 관련된 log 파일

및 bad 파일의 내용을 검사하여 조치한다. 이에 관련된 자세한 사항은 *iLoader User's Manual*을 참조한다.

대상 데이터베이스에 **Materialized View**를 리프레쉬한 다음, 인덱스 및 외래키 생성, 접근 모드 변경

TWO_PHASE_SCRIPT 프로퍼티가 OFF일 때,

- “run_is_refresh_mview.sh” 스크립트를 수행한다.

```
$ sh run_is_refresh_mview.sh
```

- “run_is_index.sh” 스크립트를 수행한다.

```
$ sh run_is_index.sh
```

- “run_is_fk.sh” 스크립트를 수행한다.

```
$ sh run_is_fk.sh
```

- “run_is_alt_tbl.sh” 스크립트를 수행한다.

```
$ sh run_is_alt_tbl.sh
```

TWO_PHASE_SCRIPT 프로퍼티가 ON일 때,

- “run_is_con.sh” 스크립트를 수행한다.

```
$ sh run_is_con.sh
```

주의 사항

- SYS 사용자가 아닌 일반 사용자로 aexport를 실행할 경우 해당 사용자가 생성한 스키마에 대해서만 스크립트를 생성한다.
- SYS 사용자가 아닌 일반 사용자로 aexport를 실행할 경우 이중화 객체는 생성되지 않는다.
- 일반 사용자로 aexport를 실행할 경우 테이블 생성 권한이 필요하다. aexport는 객체간 의존성 분석을 위해 임시 테이블을 생성하기 때문이다.
- 동시에 여러 개의 aexport 프로세스를 실행하면 안 된다. aexport는 생성된 SQL 스크립트를 저장하기 위해 임시 테이블을 사용하기 때문에, 동시에 두 개 이상의 aexport 프로세스를 실행하면 예상치 못한 결과가 나올 수 있다.
- EXECUTE와 TWO_PHASE_SCRIPT 프로퍼티를 ON 으로, OPERATION 프로퍼티를 IN으로 설정하고 데이터를 업로딩할 때는, INDEX 프로퍼티 값의 영향을 받지 않는다. 이는 인덱스만 생성하는 SQL 스크립트가 따로 생성되지 않기 때문이다.

그러므로 INDEX 프로퍼티를 ON으로 업로딩 작업 (EXECUTE=ON and OPERATION=IN)을 하려면, TWO_PHASE_SCRIPT 프로퍼티를 OFF로 해야 한다.

- “run_is.sh” 스크립트 구동 시 기존에 있는 모든 사용자 및 객체를 삭제하므로, 소스 데이터베이스에서 해당 스크립트를 수행하지 않도록 주의해야 한다.
- 파라미터 -s, -p는 일반적으로 서버에 접속하기 위한 용도뿐 아니라 스크립트에서도 함께 사용된다. 그러나 -tserver, -tport와 함께 사용될 경우에는 -s와 -p는 데이터를 다운로드 할 서버에 접속하는 용도로만 사용되고, 대상 데이터베이스에 대해 수행할 스크립트에는 -tserver와 -tport로 명시한 값들이 사용된다.

```
$ aexport -s 127.0.0.1 -u sys -p manager -tserver 192.168.1.10 -tport 21300
```

```
$ cat run_il_in.sh
```

```
iload -s 192.168.1.10 -port 21300 -u SYS -p MANAGER in -f SYS_T1.fmt -d  
SYS_T1.dat -log SYS_T1.log -bad SYS_T1.bad
```

- 사용자 생성 시 PASSWORD_VERIFY_FUNCTION을 사용해서 콜백 함수를 지정했다면, 사용자를 import 하기 전에 사용자의 패스워드를 검증 함수에 부합하도록 설정해야 한다. 또한 대상 데이터베이스에 사용자를 import 하기 전에 검증 함수를 먼저 import 해야 한다.
- 사용자 이름 또는 추출할 객체 이름에 소문자, 특수 문자 또는 공백이 포함된 경우 큰 따옴표를 사용해야 한다.

aexport의 한계

- 저장 프로시저 생성 시 참조해야 할 저장 프로시저가 미리 생성되어 있지 않으면 작업이 실패하게 된다. aexport는 저장 프로시저 간의 의존성에 대한 정보에 접근할 수가 없기 때문에 저장 프로시저 생성 순서를 보장할 수 없다. 이런 경우 저장 프로시저 생성에 실패할 수 있기 때문에 대상 데이터베이스에 저장 프로시저를 수동으로 생성해야 한다.
- aexport는 시퀀스의 메타 정보에 제한적인 접근만이 가능하다. 이러한 제약 때문에 SYS 사용자 이외의 계정에서 생성한 시퀀스의 경우 INCREMENT BY에 의해서 지정된 시퀀스 특성만 반영이 되고 나머지 특성은 기본 값으로 설정된다. 이러한 제약사항이 문제가 되는 경우 시퀀스를 대상 데이터베이스에 수동으로 생성해야 한다.
- 대상 데이터베이스에 객체를 생성하는 과정에서, Material View 생성 전에 기반 테이블이 미리 생성되어 있어야 한다. aexport는 materialized view 생성을 위한 테이블 생성 순서를 보장하지 않기 때문에, materialized view 생성에 실패할 수도 있다. 이 경우, 사용자가 수동으로 materialized view를 생성해야 한다.

- aexport 가 원본 데이터베이스에서 materialized view 생성 구문을 추출할 때, 최초로 그 materialized view를 생성했던 구문을 가져온다. 즉, 원본 데이터베이스에서 materialized view에 대해서 refresh 방법 또는 refresh 변경 시기를 변경하는 DDL문을 수행하더라도 aexport가 추출하는 구문에는 해당 변경사항이 반영되지 않는다.

SSL 접속과 스크립트 파일

- SSL 접속으로 aexport를 실행하는 경우, 원본 데이터베이스 접속용 스크립트(run_il_out.sh) 파일에는 aexport를 실행할 때 사용한 SSL 옵션이 그대로 적용된다.
- 대상 데이터베이스에 SSL 접속을 원한다면 프로퍼티 파일에 SSL 관련 프로퍼티를 설정해야 한다. 자세한 설명은 aexport 프로퍼티 절의 ILOADER_ARRAY
ILOADER_ARRAY = *count* (기본값: 1)
iLoader로 데이터를 다운로드 또는 업로드 할 때 한 번에 처리할 row 개수를 지정한다.
- ILOADER**_COMMIT
ILOADER_**COMMIT = *count* (기본값: 1000)
iLoader로 데이터를 업로드할 때 커밋할 단위(개수)를 지정한다. 이 프로퍼티로 -commit옵션의 값을 지정할 수 있다.
- ILOADER_PARALLEL
ILOADER_PARALLEL = *count* (기본값: 1)
iLoader로 데이터를 다운로드 또는 업로드 할 때 병렬로 처리할 스레드 개수를 지정한다.
- ILOADER_ASYNC_PREFETCH
ILOADER_ASYNC_PREFETCH = OFF|ON|AUTO (기본값 OFF)
iLoader로 데이터를 다운로드할 때 비동기 prefetch 동작을 설정한다. 자세한 설명은 iLoader User's Manual의
'-async_prefetch' 옵션을 참고하기 바란다.
- SSL_ENABLE 프로퍼티를 참조하기 바란다.

aexport 프로퍼티

aexport 프로퍼티 설정

aexport.properties 파일의 프로퍼티 값을 조정해야 한다. 이 파일은 \$ALTIBASE_HOME/conf 디렉터리에 있어야 한다 (같은 디렉터리의 altibase.properties 파일과 혼동하지 말 것). 이 파일이 없으면 aexport는 구동되지 않는다.

Altibase를 설치할 때, aexport.properties 파일은 \$ALTIBASE_HOME/conf에 존재하지 않는다. 다만, 같은 디렉터리에 aexport.properties.sample이라는 이름의 샘플 파일을

제공하는데, 이 파일을 aexport.properties 이름으로 복사한 후, 수정해서 사용하면 된다.

aexport 프로퍼티

- OPERATION

OPERATION = IN/OUT

OUT으로 설정할 경우, 모든 스키마와 데이터를 추출할 수 있는 스크립트가 생성된다. 데이터 추출 스크립트(iLoader 실행 명령어로 구성)를 실행하면 form 파일(.fmt)과 데이터 파일(.dat)이 만들어진다.

IN으로 설정할 경우, OUT에서 생성된 스키마 생성 스크립트와 데이터 로딩 스크립트를 실행하여 대상 데이터베이스에 스키마를 만들고 데이터를 입력한다. 스키마 생성 스크립트와 데이터 입력 스크립트는 aexport를 실행하지 않고 셸 프롬프트에서 수동으로 실행할 수도 있다.

- EXECUTE

생성한 스크립트를 자동으로 수행할 것인지 여부를 설정한다.

EXECUTE = ON/OFF

ON일 경우 OPERATION 에 지정된 작업에 맞는 적절한 스크립트 파일을 자동으로 실행한다. 스크립트 파일 이름은 ILOADER_OUT, ILOADER_IN, ISQL, ISQL_CON, ISQL_INDEX, ISQL_FOREIGN_KEY, ISQL_REPL, ISQL_REFRESH_MVIEW, 그리고 ISQL_ALT_TBL 프로퍼티로 설정된다.

OFF일 경우 스크립트를 생성하기만 하고 실행하지는 않는다.

- INVALID_SCRIPT

이 프로퍼티는 유효하지 않은 객체들을 생성하는 SQL 스크립트를 하나의 스크립트 파일에 모을 것인지 여부를 결정한다.

INVALID_SCRIPT = ON/OFF

ON 일 때, INVALID.sql 파일이 생성된다. 이 파일은 유효하지 않은 뷰와 저장 프로시저를 생성하는 모든 SQL 스크립트를 포함한다.

OFF이면, 유효하지 않은 각 객체를 생성하는 SQL 스크립트가 따로 생성된다. 즉, 유효한 객체와 같은 방식으로 다뤄진다.

- TWO_PHASE_SCRIPT

이 프로퍼티는 객체 생성 스크립트를 두 개의 스크립트 파일로 나눠서 생성할 것인지를 결정한다.

TWO_PHASE_SCRIPT = ON/OFF

ON 일 때, 아래와 같이 두 개의 SQL 스크립트 파일과 두 개의 쉘 스크립트 파일만이 생성된다: ALL_OBJECT.sql, ALL_OBJECT_CONSTRAINTS.sql, ALL_OBJECT.sql, run_is.sh, run_is_con.sh

OFF이면, 각 객체를 생성하는 SQL 스크립트 파일을 따로 생성한다.

- CRT_TBS_USER_MODE

CRT_TBS_USER_MODE = ON/OFF (기본값: OFF)

사용자 모드에서 테이블스페이스 생성 구문을 추출할 것인지 여부를 결정한다. 이 프로퍼티를 ON으로 설정하면, 사용자 모드에서 해당 사용자와 관련된 테이블스페이스를 생성하는 SQL문을 추출한다. 사용자 관련 테이블스페이스는 기본 테이블스페이스, 기본 임시 테이블스페이스, 접근 가능 여부가 설정된 테이블스페이스이다.

- INDEX

INDEX = ON/OFF

대상 데이터베이스에 스키마 구성 시 인덱스 포함여부를 결정한다. 데이터 로딩 후에 인덱스를 생성하고자 할 경우 이 프로퍼티를 ON으로 설정한다.

TWO_PHASE_SCRIPT 프로퍼티는 OFF일 때 제대로 동작한다.

- USER_PASSWORD

USER_PASSWORD = *password*

원본 데이터베이스에서 추출된 사용자를 대상 데이터베이스에 생성할 때 사용할 암호를 지정한다. (aexport는 사용자 객체 추출 시 사용자의 암호를 알 수 없기 때문에 수동 설정이 필요하다.) 이 프로퍼티가 설정되어 있지 않을 경우 각 사용자의 암호를 묻는 프롬프트가 나타난다.

- VIEW_FORCE

VIEW_FORCE = ON/OFF

ON이면, 뷰의 기본 테이블 등이 존재하지 않아도 뷰를 강제로 생성한다.

- DROP

생성 스크립트 내부에 DROP 구문을 포함할 것인지 여부를 결정한다.

DROP = ON/OFF

ON이면, 객체를 제거하는 구문이 SQL 스크립트에 포함되어 대상 데이터베이스 내에 이미 객체가 존재할 경우 기존 객체를 제거하게 된다. 기존 객체를 삭제하기 때문에 사용에 주의를 요한다.

주의) 객체 모드로 실행하면, 이 프로퍼티 값에 상관없이 DROP 구문이 생성되지 않는다.

- ILOADER_OUT

ILOADER_OUT = *run_il_out.sh*

원 데이터베이스에서 데이터를 추출하기 위해 생성되는 쉘 스크립트 파일명을 설정한다. OPERATION 프로퍼티를 OUT으로 설정할 경우에 사용된다.

- ILOADER_IN

ILOADER_IN = *run_il_in.sh*

대상 데이터베이스에 데이터 로딩을 위해 사용될 쉘 스크립트의 파일명을 설정한다.

- ISQL

ISQL = *run_is.sh*

대상 데이터베이스에 데이터베이스 스키마를 구성하기 위한 SQL 스크립트를 실행하는 쉘 스크립트 파일의 이름을 설정한다.

- ISQL_CON

ISQL_CON = *run_is_con.sh*

대상 데이터베이스에 인덱스, 외래키, 트리거와 이중화 객체를 생성하는 SQL 스크립트를 실행하는 쉘 스크립트 파일의 이름을 설정한다. TWO_PHASE_SCRIPT 프로퍼티가 ON일 때 사용된다.

- ISQL_INDEX

ISQL_INDEX = *run_is_index.sh*

대상 데이터베이스에 인덱스를 생성하는 SQL 스크립트를 실행하는 쉘 스크립트 파일의 이름을 설정한다. 이 프로퍼티를 설정하지 않으면 쉘 스크립트 파일은 생성되지 않는다.

- ISQL_FOREIGN_KEY

ISQL_FOREIGN_KEY = *run_is_fk.sh*

대상 데이터베이스에 외래키 생성하는 SQL 스크립트를 실행하는 쉘 스크립트 파일의 이름을 설정한다. 이 프로퍼티를 설정하지 않으면 쉘 스크립트 파일은 생성되지 않는다.

- ISQL_REPL

ISQL_REPL = *run_is_repl.sh*

대상 데이터베이스에 이중화를 생성하기 위한 쉘 스크립트의 파일명을 설정한다. 이 프로퍼티를 설정하지 않으면 쉘 스크립트 파일은 생성되지 않는다.

- COLLECT_DBMS_STATS

이 프로퍼티는 사용자의 테이블, 칼럼, 인덱스의 통계 정보를 추출할지 여부를 결정한다.

COLLECT_DBMS_STATS = ON/OFF

기본값은 OFF이며 통계 정보를 추출하지 않는다. 이 프로퍼티의 값을 ON으로 할 경우 통계 정보를 추출하도록 한다.

- ISQL_REFERSH_MVIEW

ISQL_REFERSH_MVIEW = *run_is_refresh_mview.sh*

대상 데이터베이스의 Materialized View를 리프레쉬하는 SQL 스크립트를 실행하는 쉘 스크립트 파일의 이름을 설정한다. 이 프로퍼티를 설정하지 않으면, 쉘 스크립트 파일이 생성되지 않는다.

- ISQL_ALT_TBL

ISQL_ALT_TBL = *run_is_alt_tbl.sh*

대상 데이터베이스의 테이블과 파티션에 대한 접근 모드를 변경하는 SQL 스크립트를 실행하는 쉘 스크립트 파일의 이름을 설정한다. 이 프로퍼티를 설정하지 않으면, 쉘 스크립트 파일이 생성되지 않는다.

- ILOADER_FIELD_TERM

ILOADER_FIELD_TERM = *field_term*

테이블의 데이터를 텍스트로 다운로드 할 때 사용할 필드 구분자를 설정한다. 설정하지 않을 경우 기본값은 쉼표(,)로 구분되며, 숫자는 그대로, 문자형 칼럼은

큰 따옴표(“”)로 에워싸여서 출력된다.

주의) 프로퍼티 파일 내에서 # 문자를 구분자로 사용할 경우, # 이하를 주석으로 처리하기 때문에, #는 구분자로 사용할 수 없다.

- ILOADER_ROW_TERM

ILOADER_ROW_TERM = *row_term*

테이블 데이터를 텍스트로 내릴 때 사용할 레코드 구분자를 설정한다. 설정하지 않을 경우 기본값은 <LF>이다.

주의) 프로퍼티 파일 내에서 # 문자를 구분자로 사용할 경우, # 이하를 주석으로 처리하기 때문에, #는 구분자로 사용할 수 없다.

- ILOADER_PARTITION

이 프로퍼티는 파티션 생성을 위한 SQL 스크립트와 쉘 스크립트를 만들 것인지를 결정한다.

ILOADER_PARTITION = ON/OFF

이 값이 ON일 경우, 파티션들의 데이터를 추출하는 스크립트, 파티션드 테이블과 그 테이블의 파티션들을 생성하는 스크립트, 그리고 각 파티션으로 데이터를 입력하는 스크립트를 생성한다. 즉, 원본 데이터베이스의 테이블 파티션의 데이터를 대상 데이터베이스의 상응하는 파티션드 테이블의 파티션에 입력하게 된다.

이 값이 OFF일 경우, 원본 데이터베이스에서 파티션드 테이블이라 하더라도 대상 데이터베이스에 non-partitioned 테이블을 생성하는 스크립트를 작성하고, 원본 데이터베이스의 파티션드 테이블의 모든 파티션의 데이터를 대상 데이터베이스의 non-partitioned 테이블에 입력한다.

ILOADER 관련 프로퍼티에 대한 더 자세한 설명은 *iLoader User's Manual* 을 참고하기 바란다.

- ILOADER_ERRORS

ILOADER_ERRORS = *count* (기본값: 50)

iLoader로 데이터를 업로드할 때 허용 가능한 최대 에러 개수를 지정한다. 이 프로퍼티의 기본값은 50이며, 0으로 설정하면 발생하는 에러 개수와 무관하게 업로드가 계속 실행된다.

- ILOADER_ARRAY

ILOADER_ARRAY = *count* (기본값: 1)

iLoader로 데이터를 다운로드 또는 업로드 할 때 한 번에 처리할 row 개수를 지정한다.

- ILOADER**_COMMIT

ILOADER_COMMIT** = *count* (기본값: 1000)

iLoader로 데이터를 업로드할 때 커밋할 단위(개수)를 지정한다. 이 프로퍼티로 -commit 옵션의 값을 지정할 수 있다.

- ILOADER_PARALLEL

ILOADER_PARALLEL = *count* (기본값: 1)

iLoader로 데이터를 다운로드 또는 업로드 할 때 병렬로 처리할 스레드 개수를 지정한다.

- ILOADER_ASYNC_PREFETCH

ILOADER_ASYNC_PREFETCH = OFF|ON|AUTO (기본값 OFF)

iLoader로 데이터를 다운로드할 때 비동기 prefetch 동작을 설정한다. 자세한 설명은 iLoader User's Manual의

'-async_prefetch' 옵션을 참고하기 바란다.

- SSL_ENABLE

대상 데이터베이스에 SSL 프로토콜로 접속할 것인지 여부를 지정한다.

SSL_ENABLE = ON/OFF

ON이면 대상 데이터베이스에 실행하는 쉘 스크립트(run_is.sh, run_il_in.sh)

내의 isql, iloader 명령에 SSL 관련 옵션이 지정된다.

SSL 관련 옵션 값은 프로퍼티(SSL_CA, SSL_CAPATH, SSL_CERT, SSL_KEY, SSL_CIPHER, SSL_VERIFY)로 명시할 수 있다. 각 프로퍼티의 역할은 aexport 접속 파라미터와 동일하므로 해당 절을 참조한다. SSL_ENABLE 프로퍼티 설정에 대한 예제는 SSL 프로퍼티 설정을 참고한다.

사용 예제

전체 DB 모드로 실행


```
$ aexport -s 127.0.0.1 -u sys -p manager
```

```
-----  
Altibase Export Script Utility.  
Release Version 7.1.0.0.0  
Copyright 2000, ALTIBASE Corporation or its subsidiaries.  
All Rights Reserved.  
-----
```

```
##### TBS #####  
##### USER #####  
##### SYNONYM #####  
##### DIRECTORY #####  
##### TABLE #####  
##### QUEUE #####  
##### SEQUENCE #####  
##### DATABASE LINK #####  
##### VIEW #####  
##### MATERIALIZED VIEW #####  
##### STORED PROCEDURE #####  
##### STORED PACKAGE #####  
##### TRIGGER #####  
##### LIBRARY #####  
##### REPLICATION #####  
##### JOB #####
```

```
-----  
##### The following script files were generated. #####
```

```
1. run_il_out.sh           : [ iloader formout, data-out script ]  
2. run_is.sh              : [ isql table-schema script ]  
3. run_il_in.sh           : [ iloader data-in script ]  
4. run_is_refresh_mview.sh : [ isql materialized view refresh script ]  
5. run_is_index.sh        : [ isql table-index script ]  
6. run_is_fk.sh           : [ isql table-foreign key script ]  
7. run_is_repl.sh         : [ isql replication script ]  
8. run_is_job.sh          : [ isql job script ]  
9. run_is_alt_tbl.sh      : [ isql table-alter script ]  
-----
```

```
$ ls -l
```

```
ALL_ALT_TBL.sql  
ALL_CRT_DIR.sql  
ALL_CRT_FK.sql  
ALL_CRT_INDEX.sql  
ALL_CRT_JOB.sql  
ALL_CRT_LIB.sql  
ALL_CRT_LINK.sql  
ALL_CRT_REP.sql  
ALL_CRT_SEQ.sql  
ALL_CRT_SYN.sql  
ALL_CRT_TBL.sql  
ALL_CRT_TBS.sql  
ALL_CRT_TRIG.sql  
ALL_CRT_USER.sql
```

ALL_CRT_VIEW_PROC.sql
ALL_REFRESH_MVIEW.sql
run_il_in.sh
run_il_out.sh
run_is.sh
run_is_alt_tbl.sh
run_is_fk.sh
run_is_index.sh
run_is_job.sh
run_is_refresh_mview.sh
run_is_repl.sh

사용자 모드로 실행

```
isQL> CREATE USER user1 IDENTIFIED BY user1;
```

Create success.

```
$ aexport -s 127.0.0.1 -u user1 -p user1
```

Altibase Export Script Utility.

Release Version 7.1.0.0.0

Copyright 2000, ALTIBASE Corporation or its subsidiaries.

All Rights Reserved.

USER

SYNONYM

TABLE

QUEUE

SEQUENCE

DATABASE LINK

VIEW

MATERIALIZED VIEW

STORED PROCEDURE

STORED PACKAGE

TRIGGER

LIBRARY #####

The following script files were generated.

1. run_il_out.sh : [iloader formout, data-out script]
 2. run_is.sh : [isql table-schema script]
 3. run_il_in.sh : [iloader data-in script]
 4. run_is_refresh_mview.sh : [isql materialized view refresh script]
 5. run_is_index.sh : [isql table-index script]
 6. run_is_fk.sh : [isql table-foreign key script]
 7. run_is_repl.sh : [isql replication script]
 8. run_is_job.sh : [isql job script]
 9. run_is_alt_tbl.sh : [isql table-alter script]
-

```
$ ls -l
```

USER1_ALT_TBL.sql

USER1_CRT_DIR.sql

USER1_CRT_FK.sql

USER1_CRT_INDEX.sql

USER1_CRT_LIB.sql

USER1_CRT_LINK.sql

USER1_CRT_SEQ.sql

USER1_CRT_SYN.sql

USER1_CRT_TBL.sql

USER1_CRT_TRIG.sql

USER1_CRT_USER.sql

USER1_CRT_VIEW_PROC.sql

USER1_REFRESH_MVIEW.sql

run_il_in.sh

run_il_out.sh

run_is.sh

```
run_is_alt_tbl.sh  
run_is_fk.sh  
run_is_index.sh  
run_is_job.sh  
run_is_refresh_mview.sh  
run_is_repl.sh
```

객체 모드로 실행

```

iSQL> CREATE USER user1 IDENTIFIED BY user1;
Create success.
iSQL> CONNECT user1/user1;
iSQL> CREATE TABLE t1(i1 INTEGER);
Create success.
iSQL> CREATE VIEW v1 AS SELECT i1 FROM t1;
Create success.
iSQL> CREATE MATERIALIZED VIEW m1 AS SELECT * FROM t1;
Create success.
iSQL> CREATE OR REPLACE PROCEDURE proc1(p1 IN INTEGER)
AS a INTEGER;
BEGIN
SELECT * INTO a FROM t1 WHERE i1 = 1;
END;
/
Create success.

```

```
$ aexport -s 127.0.0.1 -u user1 -p user1 -object user1.t1
```

```

-----
Altibase Export Script Utility.
Release Version 7.1.0.0.0
Copyright 2000, ALTIBASE Corporation or its subsidiaries.
All Rights Reserved.
-----

```

```
##### TABLE #####
```

```
$ ls
user1_t1_CRT.sql
```

```
$ aexport -s 127.0.0.1 -u user1 -p user1 -object user1.m1
```

```

-----
Altibase Export Script Utility.
Release Version 7.1.0.0.0
Copyright 2000, ALTIBASE Corporation or its subsidiaries.
All Rights Reserved.
-----

```

```
##### MATERIALIZED VIEW #####
```

```
$ ls
user1_m1_CRT.sql
```

```
$ aexport -s 127.0.0.1 -u user1 -p user1 -object user1.t1,user1.v1,user1.proc1
```

```

-----
Altibase Export Script Utility.
Release Version 7.1.0.0.0
Copyright 2000, ALTIBASE Corporation or its subsidiaries.
All Rights Reserved.
-----

```

```
##### TABLE #####
```

```
##### VIEW #####
```

```
##### STORED PROCEDURE #####
```

```
$ ls
user1_proc1_CRT.sql
```

```
user1_t1_CRT.sql
user1_v1_CRT.sql
```

SSL 프로퍼티 설정

```
SSL_ENABLE = ON # OFF
SSL_CA      = ${ALTIBASE_HOME}/cert/ca-cert.pem
#SSL_CAPATH = ${ALTIBASE_HOME}/cert
SSL_CERT    = ${ALTIBASE_HOME}/cert/client-cert.pem
SSL_KEY     = ${ALTIBASE_HOME}/cert/client-key.pem
SSL_CIPHER  = RC4-SHA:RC4-MD5
SSL_VERIFY  = ON # OFF
```

2.altiComp

이 장은 altiComp 유틸리티의 기능을 소개하고, 불일치하는 데이터를 비교하고 일치시키는 기능 등을 설명한다.

altiComp 소개

Altibase AltiComp 유틸리티의 사용 목적은 두 Altibase 간의 이중화 진행 상태를 모니터링하고 이중화 중에 발생한 데이터 불일치를 해소하는데 있다.

altiComp은 Altibase 데이터베이스를 또 다른 Altibase 데이터베이스와 테이블 단위로 비교, 검사하여 불일치 정보를 출력한다. 또한 불일치 데이터가 발생한 경우에는 두 데이터베이스를 일치시키는 기능도 제공한다.

altiComp 용어

Master Server

두 서버 간의 불일치 레코드 발견 시 수정을 하는 기준이 되는 서버이다. altiComp 실행 시 어느 쪽 서버도 master로 지정될 수 있다.

Master DB

Master Server의 데이터베이스이다.

Slave Server

두 서버 간의 불일치 레코드 발견 시 기준 데이터베이스에 따라서 수정이 되는 쪽 서버이다. altiComp 실행 시 어느 쪽 서버도 slave로 지정될 수 있다.

Slave DB

Slave Server의 데이터베이스이다.

불일치 레코드(Different Record)

Master DB의 지정 테이블과 Slave DB의 지정 테이블간에 주요 키(Primary Key)를 기준으로 칼럼 값이 일치하지 않는 레코드를 의미한다.

불일치 레코드가 발생하는 이유는 다음의 세 가지 경우이다.

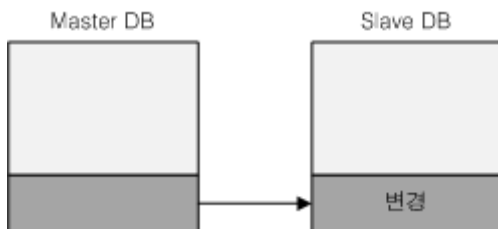
- MOSX 불일치: 특정 레코드가 주요 키를 기준으로 Master DB에는 있으나, Slave DB에는 없는 경우
- MOSO 불일치: 특정 레코드가 주요 키를 기준으로 Master DB와 Slave DB에도 있지만, 레코드의 내용이 다른 경우
- MXSO 불일치: 특정 레코드가 주요 키를 기준으로 Master DB에는 없고, Slave DB에는 있는 경우

일치 정책(Synchronization Policy)

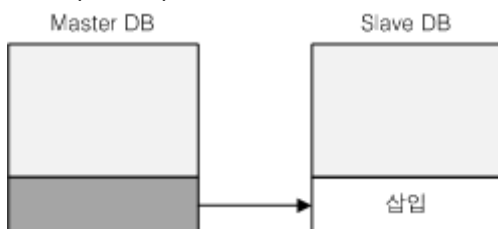
일치 정책은 불일치 레코드를 일치시키는 방법을 명시하는 정책이다. altiComp 프로그램은 기본적으로 Master DB를 기준 DB로 고정하고, Slave DB를 일치시키는 정책을 채택하고 있다.

Altibase는 일치 정책으로 다음과 같은 네 가지 방법을 제공한다.

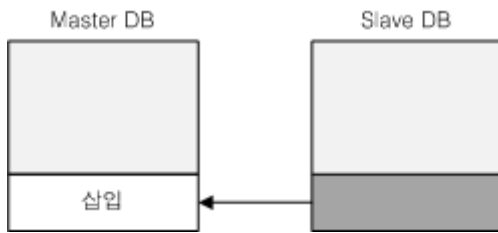
- SU 정책: MOSO 불일치를 해소하는 정책으로, Master DB의 레코드 내용으로 Slave DB를 변경(update)한다.



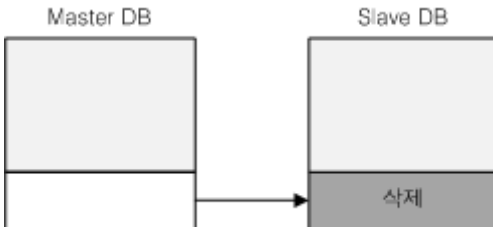
- SI 정책: MOSX 불일치를 해소하는 정책으로, Master DB의 레코드를 Slave DB에 삽입(insert)한다.



- MI 정책: MXSO 불일치를 해소하는 정책으로, Slave DB의 레코드를 Master DB에 삽입(insert)한다.



- SD 정책: MXSO 불일치를 해소하는 정책으로, Slave DB의 레코드를 삭제(delete)한다.



SU정책, SI정책, MI정책, SD정책은 사용자가 altiComp 환경 파일을 사용하여 지정할 수 있다. 단, MI정책과 SD정책은 서로 배타적이므로, 동시에 지정할 수 없다는 것을 주의하라.

DIFF

Master DB와 Slave DB간의 이중화 작업에서 발생할 수 있는 불일치 레코드를 식별하여 실행 결과 파일로 생성하는 기능을 가진다.

SYNC

Master DB와 Slave DB 사이의 불일치 레코드를 식별하여 altiComp 환경 파일에 기술된 일치 정책에 따라 양방향으로 반영하여 불일치를 해소하고, 실행 요약정보와 에러 정보를 포함하는 실행 결과 파일을 생성한다.

altiComp 환경 파일

altiComp를 실행하기 위한 옵션을 지정하는 환경 파일이다. 이 파일은 연결정보, altiComp 기능 설정, 일치 정책 등의 내용을 포함한다.

altiComp 사용 방법

이 절에서는 altiComp를 실행하기 위한 정보를 기술하는 altiComp 환경 파일을 먼저 설명하고, 비교(DIFF)와 일치(SYNC) 기능에 대해 설명한다.

altiComp 실행 방법

altiComp 기능을 사용하기 위해서는 먼저 DIFF 또는 SYNC를 수행할 테이블에 대한 정보를 가지고 있는 altiComp 환경 파일을 만들어야 한다. altiComp 환경 파일은 운용 방법의 altiComp 환경 파일에서 설명한다.

altiComp 명령은 다음과 같은 형태를 갖는다.

```
$ altiComp -f script_file_name
```

script_file_name : 환경 파일의 경로명을 포함한 파일 이름

현재 디렉토리가 /user/charlie/altibase_home/altiComp 라면

```
/user/charlie/altibase_home/altiComp> altiComp script_file_name
```

또는

```
/user/charlie/altibase_home/altiComp> altiComp ./script_file_name
```

altiComp 프로퍼티 설정

비교 또는 일치 작업의 내용을 기술하는 환경 파일로, 각각의 고유한 프로퍼티를 포함한다. 프로퍼티는 altiComp 유틸리티 실행에 필요한 정보를 기술한다.

(\$ALTIBASE_HOME/ altiComp 디렉터리 밑에 제시된 sample.cfg 참조)

설정 규칙

프로퍼티는 “**프로퍼티 이름 = 프로퍼티 값**”의 형식으로 구성되며, 대소문자 구분이 없다.

환경 파일은 다음과 같은 특별한 의미를 가지는 기호를 포함할 수 있다.

- “#” 기호는 주석(comment)으로 이후의 문자열은 무시된다.
- “{ }” 기호는 프로퍼티 값을 여러 줄로 기술하기 위한 목적으로 사용된다.
- “;” 기호는 여러 개의 값을 분리하기 위한 구분자 역할을 한다.
- ““ ”” 기호: 사용자 이름, 비밀번호 또는 테이블 이름, 칼럼 이름에 예약어 또는 특수 문자가 포함되어 있다면, 이 기호로 해당 문자열을 감싸준다.
Altibase에서 특수 문자는 ~, !, @, #, \$, %, ^, &, *, (,), _, +, | 이다.

프로퍼티 이름

공백이 없는 문자로 구성되며, 프로퍼티 그룹 내에서 식별하도록 하는 이름이다.

프로퍼티 값

단일 값(single value), 다중 값(multi value) 또는 표현식(expression)을 가질 수 있다.

- 표현식은 공백이 가능하며, 대부분의 프로퍼티는 이 형태의 값을 가진다.
예) TABLE = EMPLOYEE
- 다중 값은 “;” 분리자로 구분되는 여러 개의 단일 값을 가지며, 특히 여러 줄에 기술할 때는 “{ }” 기호 안에 값을 기술한다(예2). 다중 값이 허용되는 프로퍼티는 EXCLUDE 이다.
예) EXCLUDE = ENO; DNO; ENAME
또는 EXCLUDE = {ENO; DNO; ENAME}
- 표현식은 공백을 포함한 문자열을 표현하기 위한 것이며, 반드시 “{ }” 기호 안에 기술되어야 한다. 표현식을 가지는 프로퍼티는 WHERE 이다.
예) WHERE = { ENO > '1000' and ENO < '2000' }

데이터 타입 지원

altiComp 대상에서 특정 칼럼을 제외시키려면 프로퍼티 EXCLUDE를 다음과 같이 사용한다.

예) EMP 테이블에 CLOB 칼럼이 존재할 때, 해당 칼럼을 altiComp 대상에서 제외한다.

```
TABLE = EMP
EXCLUDE = { CCC }
```

프로퍼티 옵션

altiComp 유틸리티는 아래의 프로퍼티로 Local Server와 Remote Server의 접속 정보 및 비교(DIFF) 작업, 일치(SYNC)작업, 불일치 레코드에 대한 일치 정책을 지정할 수 있다.

DB_MASTER

두 서버 간의 불일치 레코드 발견 시 데이터 수정의 기준으로 하려는 서버를 지정한다.

값은 사용자명, 패스워드, 서버의 IP 주소 또는 서버명, NLS_USE 를 지정한다. 각 프로퍼티의 값은 Altibase 홈 디렉터리의 프로퍼티 파일에 있는 정보와 일치하여야 한다.

- TCP 접속:

```
DB_MASTER = altibase://sys:manager@DSN=192.188.1.1;PORT_NO=20300;NLS_USE=US7ASCII
```

- SSL 접속:

```
DB_MASTER = altibase://sys:manager@DSN=192.188.1.1;PORT_NO=${ALTIBASE_SSL_PORT_NO};NLS_USE=US7ASCII;CONNTYPE=SSL
```

연결 문자열의 SSL 관련 속성은 SSL/TLS User's Guide를 참고하기 바란다.

DB_SLAVE

상대 서버를 지정한다.

값은 사용자명, 패스워드, 서버의 IP 주소 또는 서버명, NLS_USE 를 지정한다.
프로퍼티 값은 Altibase 홈 디렉터리의 프로퍼티 파일에 있는 정보와 일치하여야 한다.

OPERATION

비교 작업을 하는 경우에는 “DIFF”를, 일치 작업을 하는 경우에는 “SYNC”를 지정한다.

INSERT_TO_SLAVE

MOSX 불일치에 대한 SI정책을 지정. 해당 레코드의 Slave DB에 삽입(insert) 여부를 결정한다. 프로퍼티 값은 “ON”, “OFF”를 가질 수 있으며, “ON”이면 삽입하고, “OFF”이면 삽입하지 않는다.

INSERT_TO_MASTER

MXSO 불일치에 대한 MI정책을 지정. 해당 레코드의 Master DB에 삽입(insert) 여부를 결정. 프로퍼티 값은 “ON”, “OFF”를 가질 수 있으며, “ON”이면 삽입하고, “OFF”이면 삽입하지 않는다.

DELETE_IN_SLAVE 프로퍼티 값과 동시에 “ON”을 가질 수 없다.

DELTE_IN_SLAVE

MXSO 불일치에 대한 SD정책을 지정. 해당 레코드의 Slave DB에서 삭제(delete) 여부를 결정. 프로퍼티 값은 “ON”, “OFF”를 가질 수 있으며, “ON”이면 삭제하고, “OFF”이면 삭제하지 않는다.

INSERT_TO_MASTER 프로퍼티 값과 동시에 “ON”을 가질 수 없다.

UPDATE_TO_SLAVE

MOSO 불일치에 대한 SU정책을 지정. 해당 레코드의 Slave DB에서 수정(update) 여부를 결정. 프로퍼티 값은 “ON”, “OFF”를 가질 수 있으며, “ON”이면 수정하고, “OFF”이면 수정하지 않는다.

CHECK_INTERVAL

테이블에 대하여 SYNC 작업을 한 후 다음 테이블의 SYNC 작업을 하기 전에 지정한 시간만큼 간격을 둔다. 단위는 MS(Millisecond)이다.

MAX_THREAD

운영하기 위한 스레드의 개수를 지정한다. 작업하려는 스레드의 개수를 제한 없이 사용하려면 “-1”을 명시한다.

FILE_MODE_MAX_ARRAY

이 값이 1보다 크면 작업 대상 테이블의 데이터를 파일에 저장한 후, 파일의 데이터에 대해서 SYNC 또는 DIFF 작업을 수행하게 된다. 이 값은 array fetch 하기 위한 array의 사이즈로 사용되고, altiComp는 지정한 값만큼의 레코드를 array fetch하여 csv 형식으로 파일에 저장한다.

이 옵션은 대용량의 테이블에 대한 altiComp 작업 성능을 향상시키기 위해 사용할 수 있으나, 대상 테이블에 LOB타입의 칼럼이 있으면 이 옵션을 사용해도 성능 향상이 미미할 수 있다.

이 옵션은 Altibase 서버간의 SYNC 또는 DIFF 작업 시에만 사용할 수 있다.

예) FILE_MODE_MAX_ARRAY = 1000

LOG_EQ_MOSO

PK를 포함한 모든 컬럼의 값이 일치하는 MOSO 레코드 정보를 실행 결과 파일에 기록할 지 결정하는 프로퍼티이다.

프로퍼티 값은 “ON”, “OFF”를 가질 수 있으며, “ON”이면 기록하고, “OFF”이면 기록하지 않는다. 프로퍼티를 지정하지 않으면 “OFF”로 동작한다.

기록 형식은 *MOSO[m,n]->EQ:PK->{PCOL_V}* 이다.

- m : Master Server의 레코드 순서
- n : Slave Server의 레코드 순서

이 옵션은 대용량 테이블을 비교할 때, 부하를 많이 줄 수 있으므로 주의해서 사용해야 한다.

TABLES 그룹

실행 대상이 되는 테이블에 관련된 정보를 정의한다. 이 그룹은 실행 대상이 되는 테이블의 개수만큼 기술되어야 하며, 반드시 그룹명은 Master DB의 테이블 이름이어야 한다.

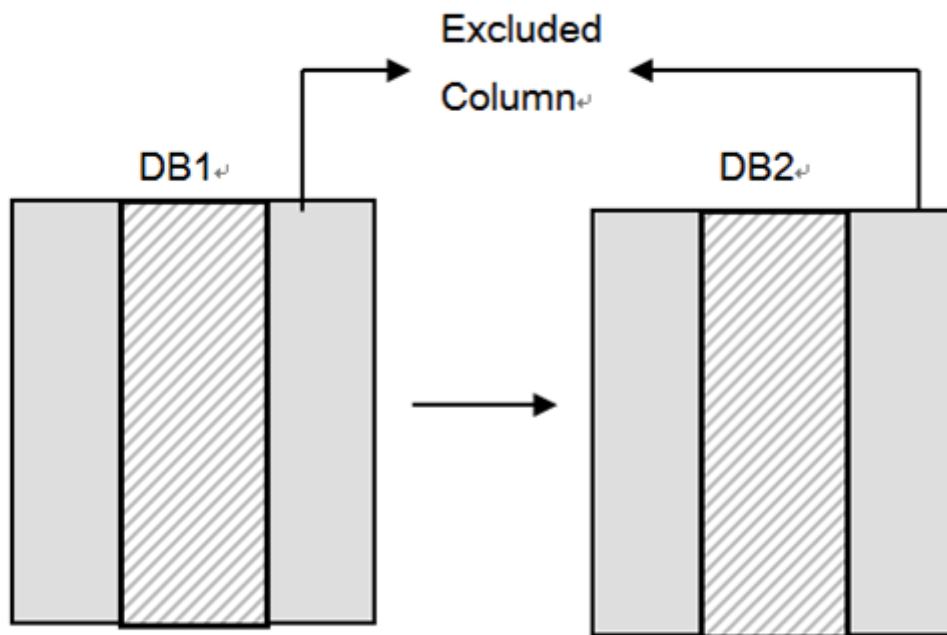
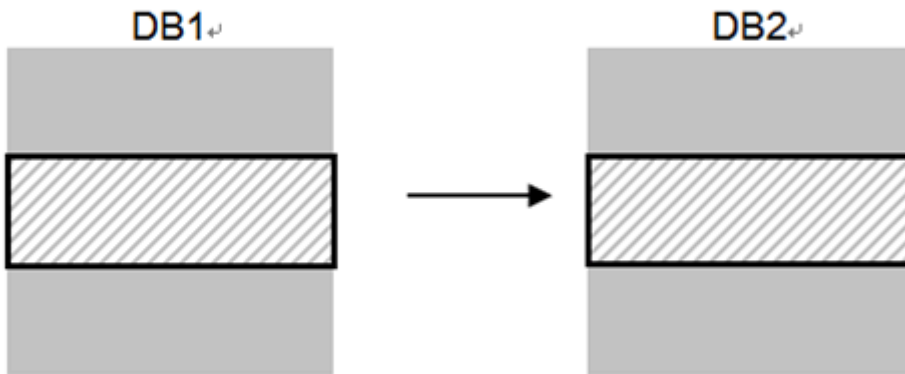
이 그룹에서 지정할 수 있는 프로퍼티는 다음과 같다.

WHERE

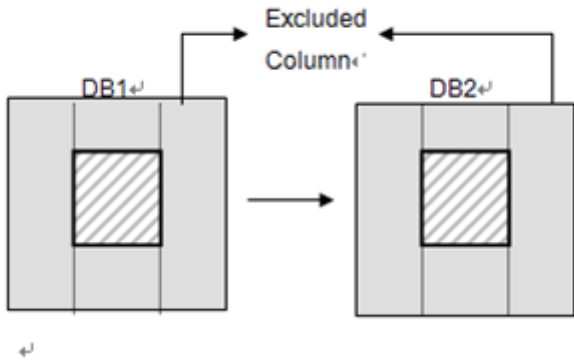
테이블의 레코드를 selection하는 조건을 지정한다. SQL 문의 WHERE 절에 기술하는 방식과 동일하게 기술한다. 다중 값을 허용하며, “;” 구분자로 여러 조건을 지정하는 것은 불가능하다. 또한 이 프로퍼티에는 주석을 달 수 없다. 비교(DIFF)와 일치(SYNC) 기능에 적용된다.

EXCLUDE

테이블의 레코드를 projection하는 조건을 지정한다. 프로퍼티 값은 다중 값으로 기술해야 한다. 기술된 칼럼들은 비교와 일치 작업에서 제외된다.



WHERE와 EXCLUDE를 적절히 조합하면, selection과 projection을 결합한 결과에 대해 altiComp 작업을 수행할 수 있다.



TABLE

Slave DB의 테이블 이름을 지정한다. Master DB와 Slave DB의 테이블 이름이 다른 경우에는 반드시 기술해야 하며, 비교(DIFF)와 일치(SYNC)기능에 적용된다. 생략하면 Master DB의 테이블 이름과 동일하게 지정된다.

단, 테이블 이름은 반드시 영문 및 숫자와 해당 특수문자((공백,~,!,@,#,\$,% ^,&,*,(,),_,+,|)로 지정해야 하며, 한글은 사용할 수 없다.

SCHEMA

Slave Db의 테이블 스키마를 기술한다.

Slave의 접속하는 사용자의 스키마 이름과 대상 테이블의 스키마가 다른 경우에 반드시 기술해야 하며, 생략하면 Slave의 접속 사용자의 스키마를 사용하게 된다.

비교(DIFF)기능

Master DB와 Slave DB 간의 이중화 작업에서 발생할 수 있는 불일치 레코드를 식별하여 실행 결과 파일로 생성하는 기능을 가진다.

환경 파일

altiComp 환경 파일의 OPERATION 프로퍼티 값을 “DIFF”로 지정한다.

실행 옵션의 모든 프로퍼티 파일을 기술해야 하며, 테이블 그룹의 WHERE, EXCLUDE, TABLE, SCHEMA 프로퍼티를 선택적으로 기술한다.

실행방법

비교(DIFF)기능은 다음과 같이 실행한다.

```
$ altiComp -f script_file_name
```

script_file_name : 환경 파일의 경로명을 포함한 파일 이름

실행결과

실행 결과로, 실행결과에 대한 요약 정보를 포함하는 실행 로그 파일과 테이블 별로 Master Database와 Slave Database의 내용을 비교하여 불일치가 발생한 레코드의 불일치 칼럼 내용을 포함하는 실행 결과 파일이 생성된다.

예를 들어,

/user/charlie/altibase_home/altiComp > altiComp sample.cfg 과 같이 altiComp 명령을 실행했을 때, altiComp 명령이 성공적으로 실행되면 altiComp 디렉터리 하위에는 sample.log와 각각의 테이블에 대한 “마스터 테이블-사용자명.슬레이브 테이블.log” 파일이 생성된다.

실행 로그 파일

“script_file_name.log” 이름으로 생성되며 실행한 환경 파일의 내용을 출력하고, 각 TABLES 그룹의 테이블에 대한 비교(DIFF)작업의 요약 정보를 출력한다.

환경 파일의 내용은 다음과 같이 출력된다.

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

```
[TAB_2->TAB_2]
Fetch Rec In Master: 3
Fetch Rec In Slave : 2
MOSX = DF, Count :      1
MXSO = DF, Count :      0
MOSO = DF, Count :      1
MOSO = EQ, Count :      1
```

```
SCAN TPS: 20547.95
Time:      0.00 sec
```

실행 결과 파일

“마스터 테이블-사용자명.슬레이브 테이블.log” 이름으로 생성되며, 비교 결과를 다음과 같은 형식으로 표현한다.

```
DF[m,n]-> COL_N (Vn_M, Vn_S):PK->{ PCOL_V }
```

- DF : 불일치 원인 (MOSX, MOSO, MXSO)
- m : Master Server의 레코드 순서

- n : Slave Server의 레코드 순서
- COL_N : 비교결과 다른 값을 가지는 첫 컬럼 이름
- Vn_M : Master Server의 해당 컬럼의 값
- Vn_S : Slave Server의 해당 컬럼의 값

단, 데이터 타입이 LOB인 컬럼이 있는 레코드의 경우 해당 컬럼의 값이 기록되지 않는다.

비교(DIFF) 예제

host1의 EMP 테이블과 host2의 EMPLOYEES 테이블, 그리고 host1의 DEPARTMENTS 테이블과 host2의 DEPARTMENTS 테이블에 대해 비교(DIFF) 기능을 수행하는 예제는 다음과 같다.

DIFF 예제 1

DB_MASTER를 host1, DB_SLAVE를 host2로 지정하고, 각각의 테이블의 모든 레코드를 비교하는 경우에 환경설정 파일은 다음과 같다

```
DB_MASTER = "altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=US7ASCII"
DB_SLAVE = "altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=US7ASCII"
OPERATION = DIFF
MAX_THREAD = -1
```

```
DELETE_IN_SLAVE = ON
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = ON
UPDATE_TO_SLAVE = ON
```

```
LOG_DIR = "./"
LOG_FILE = "sample.log"
```

```
[EMP]
TABLE = EMPLOYEES
SCHEMA = SYS
```

```
[DEPARTMENTS]
TABLE = DEPARTMENTS
SCHEMA = SYS
```

위 예제와 같이 Master Server(host1)의 대상 테이블 이름과 Slave Server(host2)의 대상 테이블 이름이 다를 수 있다.

DIFF 예제 2

EMP 테이블의 ENO 칼럼을 기준으로 selection하고, JOIN_DATE, SEX 칼럼은 비교 대상에서 제외하는 경우는 다음과 같다.

CONDITION 프로퍼티 값에 따라, 비교 대상이 되는 EMP 레코드는 “ENO가 1 이상이고 20 이하”인 레코드로 제한된다.

또한 EXCLUDE 프로퍼티 값에 따라, JOIN_DATE 와 SEX는 비교 대상에서 제외된다.

즉, 다른 모든 칼럼은 같고, JOIN_DATE와 SEX만 다르다면 레코드는 동일한 것으로 처리된다.

```
[EMP]
TABLE = EMPLOYEES
WHERE = {ENO >= 1 and ENO <= 20}
EXCLUDE = {JOIN_DATE; SEX}
[DEPARTMENTS]
```

DIFF 예제 3

EMP 테이블의 ENO 칼럼과 JOIN_DATE를 기준으로 selection하고, SEX 칼럼은 비교 대상에서 제외하는 경우는 다음과 같다.

```
[EMP]
TABLE = EMPLOYEES
WHERE = {(ENO >= 1 and ENO <= 20) or (JOIN_DATE >= '20001010')}
EXCLUDE = {SEX}
```

```
[DEPARTMENTS]
```

WHERE 프로퍼티 값에 따라, 비교 대상이 되는 EMP 레코드는 “ENO가 1 이상이고 20 이하” 또는 “JOIN_DATE가 2000년 10월 10일 이후”인 레코드로 제한된다.

또한 EXCLUDE 프로퍼티 값에 따라, SEX는 비교 대상에서 제외된다.

일치(SYNC) 기능

Master DB와 Slave DB 사이의 불일치 레코드를 식별하여 altiComp 환경 파일에 기술된 일치 정책에 따라 양 방향으로 반영하여 불일치를 해소하고, 실행요약정보와 에러정보를 포함하는 실행 결과 파일을 생성한다.

환경 파일

altiComp 환경 파일의 OPERATION 프로퍼티 값을 “SYNC”로 지정한다.

실행 옵션의 모든 프로퍼티 파일을 기술해야 하며, 테이블 그룹의 WHERE, EXCLUDE, TABLE, SCHEMA 프로퍼티는 선택적으로 기술한다

실행방법

일치(SYNC)기능은 다음과 같이 실행한다.

```
$ altiComp -f script_file_name
```

script_file_name : 환경 파일의 경로명을 포함한 파일 이름

실행결과

실행 결과로, 실행의 요약정보를 포함하는 실행 로그 파일과 테이블 별로 Master Database와 Slave Database의 내용을 비교하여 불일치가 발생한 레코드에 대한 일치 작업의 정보를 포함하는 실행 결과 파일, 일치 작업중 발생한 에러에 관한 정보를 포함하는 에러로그 파일로 구성된다.

실행 로그 파일

“script_file_name.log” 이름으로 생성되며 실행한 환경 파일의 내용을 출력하고, 각 TABLES 그룹의 테이블에 대한 일치(SYNC)작업의 요약 정보를 출력한다.

환경 파일의 내용은 다음과 같이 출력된다.

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

```
[TAB_2->TAB_2]
Fetch Rec In Master: 3
Fetch Rec In Slave : 2
MOSX = -, SI
MXSO = -, -
MOSO = -, SU
MXSX = -, -
```

Operation	Type	MASTER	SLAVE

INSERT	Try	0	1
	Fail	0	0
UPDATE	Try	X	1
	Fail	X	0
DELETE	Try	X	0
	Fail	X	0

UPDATE	Try	0	2
	Fail	0	0
OOP TPS:	13698.63		
SCAN TPS:	20547.95		
Time:	0.00 sec		

만일 실패한 레코드가 있다면, 해당 레코드는 에러로그 파일에 원인과 레코드 내용이 출력된다.

일치(SYNC) 예제

불일치 레코드를 해소하기 위한 일치 정책에 관련된 OPERATION과 TABLE을 지정하는 예제를 제시한다.

SYNC 예제 1

MOSX 불일치레코드(Master Server에는 있지만, Slave Server에는 없는 레코드)를 Slave Server에 삽입하고, MXSO 불일치 레코드 (Slave Server에는 있지만, Master Server에는 없는 레코드)는 무시한다고 가정하자.

```
Master Server = "altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=US7ASCII"
Slave Server = "altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=US7ASCII"
OPERATION = SYNC
MAX_THREAD = -1

DELETE_IN_SLAVE = OFF
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = OFF
UPDATE_TO_SLAVE = ON

LOG_DIR = "./"
LOG_FILE = "sample.log"

[EMP]
TABLE = EMPLOYEES
SCHEMA = SYS

[DEPARTMENTS]
TABLE = DEPARTMENTS
SCHEMA = SYS
```

MOSX 불일치 레코드를 해소하기 위해 필요한 일치 정책은 SI 정책이므로, INSERT_TO_SLAVE 프로퍼티 값을 “ON”으로 지정하였다. 또한 MXSO 불일치 레코드는 무시하므로, 필요한 일치 정책인 MI, SD 정책에 관련된 INSERT_TO_MASTER와 DELETE_IN_SLAVE 프로퍼티는 “OFF”로 지정한다.

SYNC 예제 2

MOSX 불일치레코드(Master Server에는 있지만, Slave Server에는 없는 레코드)를 Slave Server에 삽입하고, MXSO 불일치 레코드(Slave Server에는 있지만, Master Server에는 없는 레코드)는 Master Server에 삽입한다고 가정하자.

```
Master Server = "altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=US7ASCII"
Slave Server  = "altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=US7ASCII"
OPERATION = SYNC
MAX_THREAD = -1
```

```
DELETE_IN_SLAVE = OFF
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = ON
UPDATE_TO_SLAVE = ON
```

```
LOG_DIR = "./"
LOG_FILE = "sample.log"
```

```
[EMP]
TABLE = EMPLOYEES
SCHEMA = SYS
```

```
[DEPARTMENTS]
TABLE = DEPARTMENTS
SCHEMA = SYS
```

MOSX 불일치 레코드를 해소하기 위해 필요한 일치 정책은 SI 정책이므로, INSERT_TO_SLAVE 프로퍼티 값을 “ON”으로 지정한다. 또한 MXSO 불일치레코드는 Master Server에 삽입해야 하기 때문에, 필요한 일치 정책은 MI 정책이다. 따라서 이에 관련된 INSERT_TO_MASTER 프로퍼티를 “ON” 으로 지정하고, DELETE_IN_SLAVE 프로퍼티를 “OFF”로 지정한다.

SYNC 예제 3

Master Server와 동일하게 Slave Server를 일치시킨다고 가정하자.

```

Master Server = "altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=US7ASCII"
Slave Server  = "altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=US7ASCII"
OPERATION = SYNC
MAX_THREAD = -1

DELETE_IN_SLAVE = ON
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = OFF
UPDATE_TO_SLAVE = ON

LOG_DIR = "./"
LOG_FILE = "sample.log"

[EMP]
TABLE = EMPLOYEES
SCHEMA = SYS

[DEPARTMENTS]
TABLE = DEPARTMENTS
SCHEMA = SYS

```

Master Server와 동일하게 Slave Server를 일치시키기 위해서 필요한 일치 정책은 SI정책, SD 정책이다. 따라서, INSERT_TO_SLAVE와 DELETE_IN_SLAVE 프로퍼티 값을 “ON”으로 지정하였다.

SYNC 예제 4

\$ALTIBASE_HOME/sample/APRE/schema 디렉터리의 schema.sql을 참조하여 지역서버 host1의 EMPLOYEES 테이블과 원격 서버 host2의 EMPLOYEES 테이블(ENO에서 16번부터 20번까지 삭제), 그리고 host1 서버의 DEPARTMENTS 테이블과 host2 서버의 DEPARTMENTS 테이블에 대해 일치(SYNC) 기능을 수행하는 간단한 예제이다.

먼저 지역서버와 원격서버에 이중화 연결을 생성한다.

지역서버의 경우(IP: 192.168.1.11)

```

iSQL> CREATE REPLICATION rep1 WITH '127.0.0.1', 56342 FROM sys.employees TO sys.employees, FROM sys.departments 1
Create Success
iSQL>

```

원격서버의 경우(IP: 127.0.0.1)

```

iSQL> CREATE REPLICATION rep1 WITH '192.168.1.11', 65432 FROM sys.employees TO sys.employees, FROM sys.department
Create Success
iSQL>

```

현재 디렉터리가 /user/charlie/altibase_home/ altiComp 이라면

```
$ vi sample.cfg
Master Server = "altibase://sys:manager@DSN=127.0.0.1;PORT_NO=20582;NLS_USE=US7ASCII"
Slave Server  = "altibase://sys:manager@DSN=192.168.1.11;PORT_NO=20582;NLS_USE=US7ASCII"

OPERATION  = SYNC
MAX_THREAD = -1

DELETE_IN_SLAVE = ON
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = OFF
UPDATE_TO_SLAVE = ON

LOG_DIR = "./"
LOG_FILE = "sample.log"
```

```
[ EMPLOYEE S]
WHERE  = {ENO >= 1 and ENO <= 20}
TABLE  = EMPLOYEES
SCHEMA = SYS
[ DEPARTMENTS ]
TABLE  = DEPARTMENTS
SCHEMA = SYS
```

```
$ altiComp -f sample.cfg
$ cat sample.log
INFO[ MNG ] Tread # 0 init is OK!
INFO[ MNG ] Tread # 1 init is OK!
INFO[ MNG ] Tread # 0 start is OK!
INFO[ MNG ] Tread # 1 start is OK!
```

```
[DEPARTMENTS->DEPARTMENTS]
Fetch Rec In Master: 5
Fetch Rec In Slave : 5
MOSX = NO
MXSO = NO
MOSO = SU
```

Operation	Type	MASTER	SLAVE

INSERT	Try	0	0
	Fail	0	0
UPDATE	Try	X	0
	Fail	X	0
DELETE	Try	X	0
	Fail	X	0

UPDATE	Try	0	0
	Fail	0	0


```
OOP TPS:      0.00
SCAN TPS:    60240.96
Time:        0.00 sec
```

[EMPLOYEES->EMPLOYEES]

Fetch Rec In Master: 20

Fetch Rec In Slave : 15

MOSX = NO

MXSO = NO

MOSO = SU

Operation	Type	MASTER	SLAVE
INSERT	Try	0	5
	Fail	0	0
UPDATE	Try	X	0
	Fail	X	0
DELETE	Try	X	0
	Fail	X	0
UPDATE	Try	0	5
	Fail	0	0

```
OOP TPS:      576.04
SCAN TPS:    2304.15
Time:        0.01 sec
```

3.dataCompJ

이 장은 dataCompJ 유틸리티의 기능을 소개하고, 불일치하는 데이터를 비교 및 일치시키는 기능 등을 설명한다.

dataCompJ 소개

dataCompJ는 Altibase 데이터베이스에서 이기종 데이터베이스로 데이터를 복제한 이후, 데이터 정합성 확인과 데이터 불일치를 해소하기 위한 도구이다. dataCompJ는 데이터 복제 톨로써 Altibase에서 제공하는 Altibase Adapter for Oracle (이하 oraAdapter) 또는 Altibase Adapter for JDBC(이하 jdbcAdapter)를 사용하는 것을 전제한다.

dataCompJ는 환경파일에 사용자가 지정한 테이블들을 대상으로 두 이기종 데이터베이스간에 복제된 데이터를 검사하며, 데이터 정합성 여부를 판단하기 위해

파일로 출력하거나 불일치 데이터를 Slave 데이터베이스에 직접 적용하여 데이터를 일치시키는 기능을 제공한다.

시스템 요구사항

이 절은 dataCompJ를 설치하고 실행하기 위해 필요한 시스템 사양에 대해 설명하고, dataCompJ와 호환되는 데이터베이스 관리 시스템을 열거한다.

하드웨어 요구사항

- CPU: 800MHz 펜티엄 III 이상
- 메인 메모리: 1024MB 이상
- 디스크: 50MB 이상의 여유 공간

소프트웨어 요구사항

- Oracle database를 Slave DB로 사용하는 경우:
Oracle 또는 IBM Java 5 이상의 Java Runtime Environment (JRE)
- MariaDB를 Slave DB로 사용하는 경우:
Oracle 또는 IBM Java 6 이상의 Java Runtime Environment (JRE)

dataCompJ는 순수 자바 애플리케이션이기 때문에 하드웨어 및 운영 체제에 상관없이 독립적으로 실행되지만, 자바 런타임 환경(JRE)에 의존적이다. 따라서 환경에 맞는 JRE를 설치해야 한다. 또한 설치된 자바의 경로를 가리키는 JAVA_HOME 환경변수가 설정되어 있어야 한다.

호환되는 데이터베이스 관리 시스템

- Master DB: Altibase 5.3.3 이상.
- Slave DB
 - Oracle 데이터베이스 9i 버전 이상
 - MariaDB 5.5.x 버전 이상

Master/Slave DB의 의미는 '[dataCompJ 용어](#)'절을 참조한다.

dataCompJ는 Altibase 및 이기종 데이터베이스들과 함께 사용할 수 있다. 이 도구는 데이터베이스 접속을 위해 JDBC 드라이버를 사용하므로, 원본 데이터베이스와 대상 데이터베이스에 호환되는 JDBC 드라이버를 준비하는 것이 중요하다. 사용자 편의를 위해, 지원되는 데이터베이스에 적합한 몇 가지 JDBC 드라이버를 dataCompJ와 함께 제공한다.

dataCompJ 용어

Master DB

비교 대상인 두 이기종 데이터베이스 중 원본 데이터를 가진 Altibase 데이터베이스이다.

Slave DB

비교 대상인 두 이기종 데이터베이스 중 복제된 데이터가 적용되는 다른 벤더의 데이터베이스이다.

TablePair

사용자가 지정하는 dataCompJ의 비교 대상을 의미하며 테이블 단위이다. Master DB에 있는 하나의 테이블(Master table)과 Slave DB에 있는 하나의 테이블(Slave table)이 쌍을 이룬다.

Master 테이블

TablePair 중 사용자가 지정한 Master DB의 비교 대상 테이블이다.

Slave 테이블

TablePair 중 사용자가 지정한 Slave DB의 비교 대상 테이블이다.

불일치 레코드(Inconsistent record)

불일치 레코드

Master 테이블과 Slave 테이블의 주요 키(Primary Key)를 기준으로 칼럼 값이 일치하지 않는 레코드들을 의미한다. 불일치 레코드 종류는 다음과 같이 세 가지 타입이 있다.

1. MOSO 불일치: 동일한 주요 키 칼럼들(Primary key columns) 값을 가진 특정 레코드가 Master 테이블과 Slave 테이블, 두 곳에 존재하지만, 주요 키가 아닌 칼럼들(Non primary key columns)의 값이 서로 다른 경우.
2. MOSX 불일치: 특정 주요 키 칼럼들(Primary key columns) 값을 가진 레코드가 Master 테이블에만 존재하고 Slave 테이블에는 존재하지 않는 경우.
3. MXSO 불일치: 특정 주요 키 칼럼들(Primary key columns) 값을 가진 레코드가 Master 테이블에는 존재하지 않고 Slave 테이블에만 존재하는 경우.

불일치 레코드 처리

dataCompJ는 불일치 레코드 발견 시 dataCompJ 환경 파일에 기술된 사용자의 선택에 따라 다음의 두 가지 기능을 제공한다.

1. DIFF: Master 테이블과 Slave 테이블 간의 불일치 레코드를 식별하여 CSV 형식의 파일에 기록한다.

2. SYNC: Master 테이블과 Slave 테이블 간의 불일치 레코드를 식별하여 dataCompJ 환경 파일에 기술된 일치 정책에 따라 불일치를 해소한다.

일치 정책 (Synchronization policy)

일치 정책은 dataCompJ 프로그램이 불일치 레코드 발견 시, Master 테이블의 레코드를 기준으로, Slave 테이블의 레코드를 어떻게 일치시킬지를 명시하는 정책이다. 일치 정책으로 다음과 같은 세 가지 방법을 제공한다..

1. MOSO 불일치: UPDATE_TO_SLAVE 정책으로 Master 테이블의 레코드를 기준으로, 동일한 주요 키를 가진 Slave 테이블의 레코드를 변경(update)한다.
2. MOSX 불일치: INSERT_TO_SLAVE 정책으로 Master 테이블에만 존재하고 Slave 테이블에는 존재하지 않는 레코드를 Slave 테이블에 삽입(insert)한다.
3. MXSO 불일치: DELETE_FROM_SLAVE 정책으로 Master 테이블에는 존재하지 않고 Slave 테이블에만 존재하는 레코드를 Slave 테이블에서 삭제(delete)한다.

dataCompJ 사용 방법

이 절에서는 dataCompJ를 실행하기 위한 방법을 설명한다. dataCompJ의 환경 파일 설정과 설정 시 고려할 점들을 설명한다.

설치 및 제거

dataCompJ는 공식 Altibase 고객센터포털 <http://support.altibase.com> 에서 내려 받을 수 있다. dataCompJCLI라는 파일 이름으로 zip 또는 tar.gz 파일 형식으로 제공된다. 이 파일은 실행 파일과 JDBC 드라이버들을 포함하고 있다.

dataCompJ 설치의 압축을 풀기만 하면 된다. 결과로 생긴 디렉토리 안에는 dataCompJCLI.sh를 포함해서 jdbc 하위 디렉토리 및 xml 파일이 존재한다. 이 폴더를 원하는 위치로 옮기면 dataCompJ를 사용할 준비가 된 것이다.

dataCompJ를 제거하려면, dataCompJ가 설치되어 있는 디렉토리를 삭제하기만 하면 된다.

dataCompJ 실행 방법

dataCompJ를 Command Line Interface (CLI)에서 수행하기 위한 명령은 다음과 같은 형태를 갖는다.

- Linux

```
$ dataCompJCLI.sh -f dataCompJ_env_file_path
```

- Windows

```
C:\dataCompJ> dataCompJ.bat -f dataCompJ_env_file_path
```

dataCompJ_env_file_path

dataCompJ 환경 파일의 경로를 의미하며 dataCompJ 설치 시 dataCompJ.xml 파일이 기본으로 제공된다.

dataCompJ 수행 단계

dataCompJ는 사용자가 설정한 환경파일을 기반으로 동작하며, 크게 구축(Build)과 실행(Run)이라는 두 단계로 수행된다.

구축(Build) 단계

구축 단계는 주어진 환경파일을 기반으로 실행(Run) 단계가 수행 가능한지 판단하기 위해 초기 조사를 수행하는 단계이다. 만약, 하나의 문제라도 발견되면 리포트 파일(dataCompJ_report.txt)에 출력하고 실행(Run) 단계로 진행되는 것이 중지되고, dataCompJ는 종료된다.

1. 사용자가 지정한 환경파일을 읽는다.
2. 환경파일에 기술된 접속정보가 유효한지 확인한다.
3. 환경파일에 기술된 대상 테이블들의 유효성을 검증하기 위해 양쪽 데이터베이스에 접속하여 메타 정보를 확인한다. 테이블 중 어느 한쪽이라도 문제를 발견하면 리포트 파일에 출력하고 dataCompJ를 종료한다.

실행(Run) 단계

실행 단계는 대상 테이블들의 데이터를 비교하여, 사용자의 선택에 따라 비교(DIFF) 또는 일치(SYNC) 기능을 수행하는 단계이다.

각 TablePair의 수행 결과는 출력 파일 (dataCompJ_report.txt)에 출력된다.

출력 파일

dataCompJ를 수행하면 텍스트 형식의 리포트 파일 1개, 로그 파일 2개가 생성된다.

비교(DIFF) 기능에 의해 생성되는 출력 파일은 '[비교\(DIFF\) 기능](#)'에서 상세히 설명한다.

- dataCompJ_report.txt: 수행 결과를 정리하여 사용자에게 전달하기 위한 텍스트 형태의 리포트 파일이다.
- dataCompJ.log: 프로그램 수행 시 발생하는 이벤트를 기록하는 로그 파일이다. 프로그램의 상세 수행이력을 추적하기 위해 사용하는 파일이다.

- dataCompJ_data.log: 실행(Run) 단계에서 생성되는 로그 파일로, 사용자가 설정 파일의 <TraceInconsistentRecord>를 true로 설정한 경우 비교(diff)/일치(sync) 수행 시 처리되는 불일치 레코드의 상세 내용을 기록한다. 불일치 레코드가 많은 경우 대용량의 파일이 생성되고 프로그램의 처리 성능이 저하되기 때문에, 불일치 레코드 처리 내역에 대한 상세 내용이 필요한 경우에만 사용을 권장한다.

dataCompJ 환경파일 설정

dataCompJ를 수행하기 위해서는 환경파일 입력이 필수적이다. dataCompJ 설치 시 dataCompJ.xml 라는 파일이 기본 환경 파일로 제공되고 있으며, 사용자가 임의의 파일을 작성 후 사용도 가능하다. 하지만, 환경파일은 dataCompJ.xml에 기술된 XML 규칙에 따라 작성되어야 하며 다국어가 포함된 경우 반드시 UTF-8로 인코딩 되어야 한다.

dataCompJ 환경파일은 크게 [Connections](#), [Options](#), [TablePairs](#) 3개 부분으로 구성된다.

Connections

Connections 부분은 Master DB 및 Slave DB에 접속하기 위한 정보를 기록하는 곳이다.

<MasterDB>

Master DB의 접속 정보를 기록한다. MasterDB의 종류는 반드시 Altibase 데이터베이스여야 한다. 아래 XML Element는 MasterDB의 하위 요소들이다.

<JdbcUrl>

JDBC 접속에 사용되는 정보로 데이터베이스 사용자 아이디와 암호를 제외한 나머지 정보를 기록하는 문자열이다.

<JdbcFilePath>

Master DB 접속에 사용할 JDBC jar file이 존재하는 경로를 지정한다.

<UserId>

데이터베이스 접속을 위한 사용자 아이디를 지정한다.

<Password>

데이터베이스 접속을 위한 사용자 아이디의 암호를 지정한다.

<FetchSize>

데이터베이스에서 데이터를 가져오는 작업을 할 때, 한번에 가져오는 레코드의 개수를 지정한다. 이 항목은 선택 항목이며, 기본값은 1,000이다.

<BatchSize>

데이터베이스에 insert/delete/update 같은 변경 작업을 수행할 때, 한 번에 수행되는 레코드 개수를 지정한다. 예를 들어 이 항목에 10을 지정하면, 10개 레코드 단위로 insert/delete/update를 수행한다. 이 항목은 선택 항목으로, 기본값은 1,000이다.

<SlaveDB>

Slave DB의 접속 정보를 기록한다. SlaveDB의 하위 요소들은 위에서 기술한 [MasterDB](#)의 하위 요소들과 동일하다.

Options

Options 부분은 dataCompJ의 수행에 필요한 사용자 설정 값을 입력하는 곳이다.

<Operation>

불일치 데이터 처리를 위한 기능을 지정하는 항목으로 비교(DIFF)나 일치(SYNC) 둘 중 하나의 기능을 선택한다.

<FileEncoding>

dataCompJ 수행 시 생성되는 파일들의 encoding 타입을 지정한다.

<Diff>

<DirPath>

비교(DIFF) 수행 시 생성되는 비교 대상 테이블 별 결과 CSV 파일이 생성되는 디렉토리 경로를 지정한다.

<Sync>

일치(SYNC) 기능 수행 시 옵션들

<MOSO UPDATE_TO_SLAVE="true"/>

MOSO 불일치 데이터 발견 시 Master 테이블 레코드 기준으로 Slave 테이블의 레코드 업데이트(Update) 여부. false를 선택하게 되면 MOSO 불일치 데이터를 처리하지 않게 된다.

<MOSX INSERT_TO_SLAVE="true"/>

MOSX 불일치 데이터 발견 시 Master 테이블에만 존재하는 레코드를 Slave 테이블에 삽입(Insert) 여부. false를 선택하게 되면 MOSX 불일치 데이터를 처리하지 않게 된다.

<MXSO DELETE_FROM_SLAVE="true"/>

MXSO 불일치 데이터 발견 시 Master 테이블에 존재하지 않는 Slave 테이블의 레코드를

삭제(Insert) 여부. false를 선택하게 되면 MXSO 불일치 데이터를 처리하지 않게 된다.

<Log>

아래 XML Element는 Log의 하위 요소들이다.

<DirPath>

dataCompJ 수행 시 생성되는 로그 파일들의 디렉토리 경로를 지정한다.

<TraceInconsistentRecord>

diff/sync 수행 시 발견되는 모든 불일치 레코드의 상세 내역을 dataCompJ_data.log 파일에 기록여부를 true/false 값으로 결정한다.

<MaxThread>

dataCompJ에게 할당 가능한 최대 스레드 개수. 0으로 지정하는 경우, dataCompJ가 해당 머신의 CPU core 개수를 MaxThread으로 할당한다.

TablePairs

TablePairs 부분은 비교 대상 테이블들을 기록하는 곳이다. 비교 대상 테이블에 대한 개별 정보를 기록하는 방법과 일괄적으로 테이블 이름들이 나열되어 있는 텍스트 파일 경로를 지정하는 방법 두 가지를 제공한다. 그리고 이 두 가지 방법은 각 개별로 사용가능하고 함께 사용도 가능하다.

개별 테이블에 대한 정보를 제공하는 방법은 아래와 같으며, 데이터 비교 방법에 대한 상세한 제어가 가능하다는 장점이 있다. 예를 들어, 데이터 비교 시 특정 칼럼을 제외시키거나 특정 조건을 만족하는 데이터들만 비교 대상으로 할 수 도 있다.

비교 대상인 테이블 이름은 Master DB와 Slave DB 모두에서 허용하는 이름이어야 한다. 만약 xml 파일에 있는 테이블 이름에 공백, 특수 문자 등이 포함되었거나 대소문자를 구분해야 할 경우 큰따옴표(")를 사용해야 한다. 예를 들어, 비교 대상 테이블 이름이 SYS schema의Employee 01이면, 테이블 이름이 공백을 포함하기 때문에 SYS."Employee 01"과 같이 테이블 이름에 큰 따옴표를 사용한다.

<TablePair>

데이터 비교 대상 단위로 하나의 MasterTable과 하나의 SlaveTable로 구성된다. 아래의 XML element는 TablePair의 하위 요소들이다.

<MasterTable>

MasterDB에 존재하는 비교 대상 테이블의 이름으로 [SchemaName].TableName 형식으로 지정 가능하다. 스키마 이름을 별도로 지정하지 않으면 MasterDB의 UserId를

기본 스키마 이름으로 사용하게 된다. 이 항목은 필수 항목으로 기술되지 않으면 프로그램 수행 시 에러가 발생한다.

<SlaveTable>

SlaveDB에 존재하는 비교 대상 테이블의 이름으로 [SchemaName].TableName 형식으로 지정 가능하다. 스키마 이름을 별도로 지정하지 않으면 SlaveDB의 UserId를 기본 스키마 이름으로 사용하게 된다. 이 항목은 선택 항목으로, 기술되지 않으면 <MasterTable>과 같은 이름으로 간주된다

<Exclude>

테이블의 레코드를 projection하는 조건을 지정한다. 쉼표(,)를 구분자로 사용하여 여러 개의 칼럼을 지정할 수 있다. 이 항목은 선택 항목으로, 기술되지 않으면 dataCompJ가 지원하는 데이터 타입을 가진 모든 칼럼들을 비교대상으로 선택한다.

<Where>

테이블의 레코드를 selection하는 조건을 지정한다. SQL 문의 WHERE 절에 기술하는 방식과 동일하게 기술하며 다중 조건을 허용한다. 하지만, 양쪽 데이터베이스에 동일하게 적용되기 때문에 양쪽 데이터베이스 문법에 어긋나지 않아야 한다. 이 항목은 선택 항목으로, 기술되지 않으면 모든 레코드들을 비교 대상으로 한다.

<TableNameFilePath>

일괄적으로 테이블 이름들을 나열한 텍스트 파일 경로를 제공하는 방법은 아래와 같으며, 다수의 테이블들을 일괄적으로 비교해야 하는 경우 테이블 입력을 손쉽게 할 수 있다는 장점이 있다.

```
<TableNameFilePath>table_name_file_path</TableNameFilePath>
```

table_name_file_path는 테이블 이름들을 나열한 텍스트 파일의 경로이다. 비교 대상 테이블의 이름은 [SchemaName].TableName 형식으로 지정할 수 있으며, 각각의 테이블 이름은 개행 문자로 구분된다. MasterDB의 비교 대상 테이블 이름과 SlaveDB의 비교 대상 테이블 이름이 반드시 일치해야 한다.

제약 사항

비교 대상의 테이블을 선택할 때 다음의 제약 사항을 반드시 고려해야 한다. 하나의 제약 사항이라도 위반한 경우, dataCompJ는 구축 단계에서 발생한 문제점을 리포트 파일(dataCompJ_report.txt)에 출력하고, 실행 단계를 수행하지 않는다.

1. 하나의 비교대상 테이블은 MasterDB와 SlaveDB에 반드시 동일하게 (칼럼 이름, 칼럼 순서, 데이터 타입, Primary key) 구성되어야 한다. 단, 데이터 타입은 이기종 데이터베이스간 서로 비교 가능한 타입이면 가능하다.

2. 지원하지 않는 데이터 타입의 컬럼은 비교 대상에서 자동으로 제외된다. (예, LOB과 같은 binary type)
3. 주요 키(primary key) 외에 값을 비교할 수 있는 컬럼이 적어도 한 개 이상이어야 한다.
 - (예제 1) table1 (c1 int, c2 int, c3 CLOB, primary key (c1, c2))
(예제 2) table1 (c1 int, c2 int, c3 varchar(100), primary key (c1, c2))예제 1에서 '제약조건 3'을 만족하는 컬럼은 c3 뿐이다. 하지만 c3 컬럼의 데이터 타입인 CLOB은 dataCompJ가 지원하지 않기 때문에, '제약 사항 2'를 위반하게 된다. 따라서 table1에 대한 비교는 허용되지 않는다.
예제 2에서 '제약 사항 3'을 만족하는 컬럼은 c3 뿐이다. 그리고 c3의 데이터 타입이 dataCompJ가 지원하는 varchar 타입이기 때문에 table1에 대한 비교가 가능하다.

비교(DIFF) 기능

비교(DIFF) 기능은 Master DB와 Slave DB 간의 이기종 복제 작업에서 발생할 수 있는 불일치 레코드들을 식별하고 이들을 파일에 기록하여, 두 데이터베이스 시스템간의 불일치 레코드들을 확인하기 위한 기능을 가진다.

환경 파일

dataCompJ 환경 파일의 <Options> - <Operation> 값을 “DIFF”로 지정한다.
<Connections>와 <Options> 항목들은 모두 필수적으로 기입해야 하며, 대상 테이블에 대해 <TablePairs>에 기술해야 한다.

dataCompJ.xml

```

<?xml version="1.0" encoding="UTF-8" ?>
<dataCompJ>
<Connections>
  <MasterDB>
    <JdbcUrl>jdbc:Altibase://192.168.1.94:21135/mydb</JdbcUrl>
    <JdbcFilePath>./jdbc/Altibase710.jar</JdbcFilePath>
    <UserId>sys</UserId>
    <Password>manager</Password>
  </MasterDB>
  <SlaveDB>
    <JdbcUrl>jdbc:oracle:thin:@//192.168.3.18:1521/xe</JdbcUrl>
    <JdbcFilePath>./jdbc/ojdbc5.jar</JdbcFilePath>
    <UserId>altibase</UserId>
    <Password>altibase</Password>
  </SlaveDB>
</Connections>

<Options>
  <Operation>DIFF</Operation>
  <FileEncoding>UTF-8</FileEncoding>
  <Diff>
    <DirPath>./diff</DirPath>
  </Diff>
  <Sync>
    <MOSO UPDATE_TO_SLAVE="true"/>
    <MOSX INSERT_TO_SLAVE="true"/>
    <MXSO DELETE_FROM_SLAVE="true"/>
  </Sync>
  <Log>
    <DirPath>./</DirPath>
    <TraceInconsistentRecord>>false</TraceInconsistentRecord>
  </Log>
  <MaxThread>0</MaxThread>
</Options>

<TablePairs>
  <TablePair>
    <MasterTable>EX1</MasterTable>
  </TablePair>
</TablePairs>
</dataCompJ>

```

실행 방법

비교(DIFF)는 다음과 같이 실행한다.

```
$ dataCompJCli.sh -f dataCompJ.xml
```

실행 결과

실행 결과로 수행 결과가 요약된 리포트 파일(dataCompJ_report.txt)과 수행 중 이벤트가 기록되는 로그파일들(dataCompJ.log, dataCompJ_data.log)이 생성된다. "출력 파일"에 이들 파일들에 대해 상세히 설명되어 있다.

예제에 사용되는 테이블의 구조는 아래와 같다.

```
CREATE TABLE ex1 (C1 int primary key, C2 varchar(20), C3 CLOB, C4 int);
```

DIFF 수행

```
$ dataCompJCli.sh -f dataCompJ.xml
```

```
-----  
Data Comparison Utility for Java (dataCompJ)  
Copyright 2000, ALTIBASE Corporation or its subsidiaries.  
All Rights Reserved.  
-----
```

```
* Build started   at 2016-12-14 16:30:03.673  
* Build finished at 2016-12-14 16:30:04.357 (Elapsed: 0:00:00.691)  
* Run   started   at 2016-12-14 16:30:04.374  
* Run   finished at 2016-12-14 16:30:04.905 (Elapsed: 0:00:00.531)  
-----
```

Refer to the following output files for details.

- ```

- dataCompJ_report.txt: Report file to provide summarized execution result.
- dataCompJ.log: Log file to keep track of detailed program events during execution.
- dataCompJ_data.log: Log file to keep track of data events during diff/sync operation as well as different rec
```

## 리포트 파일 내용 확인

```

$ cat dataCompJ_report.txt
=====
Build
- Started : 2016-12-14 16:30:03.673
- Finished : 2016-12-14 16:30:04.357
- Elapsed : 0:00:00.691

[User input information]
...

[Problematic table(s): 0]

[Candidate table(s) for data comparison: 1]

1: SYS.EX1 -> ALTIBASE.EX1
* SELECT SQL : SELECT C1,C2,C4 FROM SYS.EX1 ORDER BY C1 ASC
* Excluded columns :
* Where condition :
* N/A data type columns: C3(CLOB)
=====
Run
- Started : 2016-12-14 17:19:42.731
- Finished : 2016-12-14 17:19:42.822
- Elapsed : 0:00:00.091

1: SYS.EX1 -> ALTIBASE.EX1
 Fetched record count from MASTER: 10
 Fetched record count from SLAVE : 10

Type Records

MOSO Matched 7
MOSO Diff 2
MOSX Master only 1
MXSO Slave only 1

```

## 실행 데이터 파일

비교(DIFF) 기능 선택의 결과로 TablePair 단위로 다음 4개의 Comma-separated values (CSV) 형식의 데이터 파일들이 생성된다.

- *SchemaName.TableName\_MASTER\_diff.csv*:  
MOSO 불일치 발생시 Master table의 해당 레코드 값을 CSV 형태로 저장하는 파일.
- *SchemaName.TableName\_SLAVE\_diff.csv*:  
MOSO 불일치 발생시 Slave table의 해당 레코드 값을 CSV 형태로 저장하는 파일.

- *SchemaName.TableName\_MASTER\_only.csv*:  
MOSX 불일치 발생으로 Master table에만 존재하는 레코드 값을 CSV 형태로 저장하는 파일.
- *SchemaName.TableName\_SLAVE\_only.csv*:  
MXSO 불일치 발생으로 Slave table에만 존재하는 레코드 값을 CSV 형태로 저장하는 파일.

*SchemaName.TableName\_MASTER\_diff.csv* 파일과

*SchemaName.TableName\_SLAVE\_diff.csv* 파일에는 MOSO 불일치 데이터가 같은 순서로 기록되기 때문에, diff 프로그램으로 두 개의 파일을 비교함으로써 MOSO 불일치 발생부분을 직관적으로 비교 가능하다.

## 예제

예제에 사용되는 테이블의 구조는 아래와 같다.

```
CREATE TABLE ex1 (C1 int primary key, C2 varchar(20), C3 CLOB, C4 int); /* C3 CLOB dataCompJ에서 지원하지 않는 데0
CREATE TABLE ex2 (C1 int primary key, C2 varchar(20), C3 int);
```

### DIFF 예제 1

두 개의 테이블을 대상으로 schema 이름을 생략 또는 명시하는 예제이다.

아래의 예제에서 EX1테이블의 경우 <MasterTable>항목은 스키마 이름 없이 테이블 이름만 기술하고, <SlaveTable> 항목은 생략했다. 스키마 이름 없이 테이블 이름만 기술한 <MasterTable>은 <MasterDB>-<UserId>에 입력된 값을 스키마 이름으로 사용하게 된다. 생략된 <SlaveTable>은 <MasterTable>과 같은 값으로 가정하고 처리된다.

EX1테이블의 C3 칼럼은 dataCompJ가 지원하지 않는 데이터 타입인 CLOB으로 비교 대상에서 자동으로 제외되며, 리포트의 Candidate tables 항목에 이 내용이 기록된다.

EX2 테이블은 스키마 이름과 테이블 이름을 사용자가 명시적으로 입력하였고, 입력 받은 스키마 이름과 테이블 이름을 기반으로 처리된다.

### dataCompJ.xml

```

<?xml version="1.0" encoding="UTF-8" ?>
<dataCompJ>
<Connections>
 <MasterDB>
 <JdbcUrl>jdbc:Altibase://192.168.1.94:21135/mydb</JdbcUrl>
 ...
 <UserId>sys</UserId>
 ...
 </MasterDB>
 <SlaveDB>
 <JdbcUrl>jdbc:oracle:thin:@//192.168.3.18:1521/xe</JdbcUrl>
 ...
 <UserId>altibase</UserId>
 ...
 </SlaveDB>
</Connections>
<Options>
 <Operation>DIFF</Operation>
 ...
</Options>
<TablePairs>
 <TablePair>
 <MasterTable>EX1</MasterTable>
 </TablePair>
 <TablePair>
 <MasterTable>SYS.EX2</MasterTable>
 <SlaveTable>ALTIBASE.EX2</SlaveTable>
 </TablePair>
</TablePairs>
</dataCompJ>

```

**dataCompJ\_report.txt**

```

=====
Build
- Started : 2016-12-14 17:06:53.021
- Finished : 2016-12-14 17:06:53.679
- Elapsed : 0:00:00.669

[User input information]
...
[Problematic table(s): 0]

[Candidate table(s) for data comparison: 2]

1: SYS.EX1 -> ALTIBASE.EX1
* SELECT SQL : SELECT C1,C2,C4 FROM SYS.EX1 ORDER BY C1 ASC
* Excluded columns :
* Where condition :
* N/A data type columns: C3(CLOB)
2: SYS.EX2 -> ALTIBASE.EX2
* SELECT SQL : SELECT C1,C2,C3 FROM SYS.EX2 ORDER BY C1 ASC
* Excluded columns :
* Where condition :
* N/A data type columns:
=====

Run
- Started : 2016-12-14 17:56:04.516
- Finished : 2016-12-14 17:56:04.593
- Elapsed : 0:00:00.076

1: SYS.EX1 -> ALTIBASE.EX1
 Fetched record count from MASTER: 10
 Fetched record count from SLAVE : 10

Type Records

MOSO Matched 7
MOSO Diff 2
MOSX Master only 1
MXSO Slave only 1

2: SYS.EX2 -> ALTIBASE.EX2
 Fetched record count from MASTER: 10
 Fetched record count from SLAVE : 10

Type Records

MOSO Matched 10
MOSO Diff 0
MOSX Master only 0
MXSO Slave only 0

```



## DIFF 예제 2

대상 테이블의 특정 칼럼을 사용자가 명시적으로 제외하고(Exclude), 특정 조건을 만족하는 레코드(Where)만을 대상으로 비교(Diff) 수행하는 예제이다.

dataCompJ.xml에 EX1 테이블에 대해 주요 키가 아닌 칼럼들 중 C4 칼럼을 비교대상에서 제외시키고 C1 칼럼 값이 5보다 큰 레코드만 비교 작업을 수행하도록 사용자가 명시적으로 입력한다.

### dataCompJ.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<dataCompJ>
...
<Options>
 <Operation>DIFF</Operation>
...
</Options>
<TablePairs>
 <TablePair>
 <MasterTable>EX1</MasterTable>
 <Exclude>C4</Exclude>
 <Where>C1 > 5</Where>
 </TablePair>
 <TablePair>
 <MasterTable>SYS.EX2</MasterTable>
 <SlaveTable>ALTIBASE.EX2</SlaveTable>
 </TablePair>
</TablePairs>
</dataCompJ>
```

### dataCompJ\_report.txt

=====

#### Build

- Started : 2016-12-14 17:06:53.021  
- Finished : 2016-12-14 17:06:53.679  
- Elapsed : 0:00:00.669

-----

[ User input information ]

...

[ Problematic table(s): 0 ]

[ Candidate table(s) for data comparison: 2 ]

1: SYS.EX1 -> ALTIBASE.EX1

\* SELECT SQL : SELECT C1,C2 FROM SYS.EX1 WHERE C1 > 5 ORDER BY C1 ASC  
\* Excluded columns : C4  
\* Where condition : C1 > 5  
\* N/A data type columns: C3(CLOB)

2: SYS.EX2 -> ALTIBASE.EX2

\* SELECT SQL : SELECT C1,C2,C3 FROM SYS.EX2 ORDER BY C1 ASC  
\* Excluded columns :  
\* Where condition :  
\* N/A data type columns:

=====

#### Run

- Started : 2016-12-14 17:57:09.895  
- Finished : 2016-12-14 17:57:09.974  
- Elapsed : 0:00:00.080

-----

1: SYS.EX1 -> ALTIBASE.EX1

Fetch record count from MASTER: 5  
Fetch record count from SLAVE : 5

-----

Type	Records
MOSO Matched	5
MOSO Diff	0
MOSX Master only	0
MXSO Slave only	0

-----

2: SYS.EX2 -> ALTIBASE.EX2

Fetch record count from MASTER: 10  
Fetch record count from SLAVE : 10

-----

Type	Records
MOSO Matched	10
MOSO Diff	0
MOSX Master only	0
MXSO Slave only	0

-----

# 일치(SYNC) 기능

일치(SYNC)기능은 Master DB와 Slave DB 간의 이기종 복제 작업에서 발생할 수 있는 불일치 레코드들을 식별하여 이를 Slave DB에 적용하는 기능이다. 두 데이터베이스 시스템의 데이터 동기화(Synchronization)를 손쉽게 할 수 있다.

## 환경 파일

dataCompJ 환경 파일의 <Options> - <Operation> 값을 “SYNC”로 지정한다. <Connections>와 <Options> 항목들은 모두 필수적으로 기입해야 하며, 대상 테이블에 대해 <TablePairs>에 기술해야 한다.

### dataCompJ.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<dataCompJ>
 <Connections>
 <MasterDB>
 <JdbcUrl>jdbc:Altibase://192.168.1.94:21135/mydb</JdbcUrl>
 ...
 <UserId>sys</UserId>
 ...
 </MasterDB>
 <SlaveDB>
 <JdbcUrl>jdbc:oracle:thin:@//192.168.3.18:1521/xe</JdbcUrl>
 ...
 <UserId>altibase</UserId>
 ...
 </SlaveDB>
 </Connections>
 <Options>
 <Operation>SYNC</Operation>
 ...
 </Options>

 <TablePairs>
 <TablePair>
 <MasterTable>EX1</MasterTable>
 </TablePair>
 </TablePairs>
</dataCompJ>
```

## 실행 방법

일치(SYNC) 기능은 다음과 같이 실행한다.

```

$ dataCompJCli.sh -f dataCompJ.xml
$ dataCompJCli.sh -f ./dataCompJ.xml
 Data Comparison Utility for Java (dataCompJ)
 Copyright 2000, ALTIBASE Corporation or its subsidiaries.
 All Rights Reserved.

* Build started at 2016-12-14 18:08:40.407
* Build finished at 2016-12-14 18:08:41.076 (Elapsed: 0:00:00.676)
* Run started at 2016-12-14 18:08:41.090
* Run finished at 2016-12-14 18:08:41.654 (Elapsed: 0:00:00.564)

Refer to the following output files for details.

- dataCompJ_report.txt: Report file to provide summarized execution result.
- dataCompJ.log: Log file to keep track of detailed program events during execution.
- dataCompJ_data.log: Log file to keep track of data events during diff/sync operation as well as different recs

```

## 실행 결과

실행 결과로 수행 결과가 요약된 리포트 파일, dataCompJ\_report.txt, 그리고 수행 중 이벤트가 기록되는 로그 파일들(dataCompJ.log, dataCompJ\_data.log)이 생성된다. "출력 파일"에 이들 파일들에 대해 상세히 설명되어 있다.

### dataCompJ\_report.txt

```
$ cat dataCompJ_report.txt
```

```
=====
Build
- Started : 2016-12-14 18:08:40.407
- Finished : 2016-12-14 18:08:41.076
- Elapsed : 0:00:00.676

[User input information]
...
[Problematic table(s): 0]

[Candidate table(s) for data comparison: 1]

1: SYS.EX1 -> ALTIBASE.EX1
* SELECT SQL : SELECT C1,C2,C4 FROM SYS.EX1 ORDER BY C1 ASC
* Excluded columns :
* Where condition :
* N/A data type columns: C3(CLOB)

=====
Run
- Started : 2016-12-14 18:08:41.090
- Finished : 2016-12-14 18:08:41.654
- Elapsed : 0:00:00.564

1: SYS.EX1 -> ALTIBASE.EX1
 Fetched record count from MASTER: 10
 Fetched record count from SLAVE : 10

Type Resolution Try Fail

MOSO UPDATE TO SLAVE 2 0
MOSX INSERT TO SLAVE 1 0
MXSO DELETE FROM SLAVE 1 0

```

모든 대상 테이블과 레코드들에 대해 SYNC가 정확히 수행된 경우, 동일한 대상 테이블에 대해 DIFF 기능을 재수행시 모든 레코드가 동일하다는 비교 결과를 얻게 된다.

```
$ dataCompJCli.sh -f ./dataCompJ.xml
```

```

Data Comparison Utility for Java (dataCompJ)
Copyright 2000, ALTIBASE Corporation or its subsidiaries.
All Rights Reserved.

```

```
* Build started at 2016-12-14 18:11:54.924
* Build finished at 2016-12-14 18:11:55.561 (Elapsed: 0:00:00.645)
* Run started at 2016-12-14 18:11:55.580
* Run finished at 2016-12-14 18:11:56.190 (Elapsed: 0:00:00.610)

```

Refer to the following output files for details.

- ```
-----  
- dataCompJ_report.txt: Report file to provide summarized execution result.  
- dataCompJ.log: Log file to keep track of detailed program events during execution.  
- dataCompJ_data.log: Log file to keep track of data events during diff/sync operation as well as different records  
-----
```

```
$ cat dataCompJ_report.txt
```

```
=====
```

| Build | |
|------------|---------------------------|
| - Started | : 2016-12-14 18:11:54.924 |
| - Finished | : 2016-12-14 18:11:55.561 |
| - Elapsed | : 0:00:00.645 |

```
-----
```

[User input information]

...

[Problematic table(s): 0]

[Candidate table(s) for data comparison: 1]

1: SYS.EX1 -> ALTIBASE.EX1

| | |
|--------------------------|--|
| * SELECT SQL | : SELECT C1,C2,C4 FROM SYS.EX1 ORDER BY C1 ASC |
| * Excluded columns | : |
| * Where condition | : |
| * N/A data type columns: | C3(CLOB) |

```
=====
```

| Run | |
|------------|---------------------------|
| - Started | : 2016-12-14 18:11:55.580 |
| - Finished | : 2016-12-14 18:11:56.190 |
| - Elapsed | : 0:00:00.610 |

```
-----
```

1: SYS.EX1 -> ALTIBASE.EX1

| | |
|---------------------------------|----|
| Fetch record count from MASTER: | 10 |
| Fetch record count from SLAVE : | 10 |

```
-----
```

| Type | Records |
|------|---------|
|------|---------|

```
-----
```

| | | |
|-------|-------------|----|
| MOSO | Matched | 10 |
| MOSO | Diff | 0 |
| MOSX | Master only | 0 |
| MXSO | Slave only | 0 |
| ----- | | |

이기종간 데이터 타입 호환 테이블

dataCompJ가 지원하는 이기종 데이터베이스 간의 호환되는 데이터 타입 매핑 테이블이다.

dataCompJ가 지원하지 않는 데이터 타입을 가진 테이블을 대상으로 데이터 비교를 수행하면 다음과 같이 처리된다.

- Master 테이블과 Slave 테이블 모두 dataCompJ가 지원하지 않는 데이터 타입인 경우: 구축(Build) 단계에서 지원하지 않는 데이터 타입이라는 에러를 리포트 파일(dataCompJ_report.txt)에 출력하고, 실행(Run) 단계가 수행되지 않는다.
- Master 테이블과 Slave 테이블 둘 중 한쪽은 지원되고 한쪽은 지원되지 않는 데이터 타입인 경우: 구축(Build) 단계에서 테이블의 비교 대상 칼럼의 개수가 다르다는 에러를 리포트 파일(dataCompJ_report.txt)에 출력하고, 실행(run) 단계가 수행되지 않는다.

dataCompJ가 지원하는 데이터 타입이지만, 비교 대상 테이블의 칼럼 데이터 타입이 이기종간 데이터 타입 호환 테이블과 다른 데이터 타입 조합을 사용하는 경우 다음과 같이 처리된다.

- 데이터 비교 자체가 아예 불가능한 타입의 경우 (예를 들어, 정수형과 문자형 데이터 타입간의 비교): 구축(build) 단계에서 리포트 파일에 호환되지 않는 타입이라고 출력하고, 실행(run) 단계가 수행되지 않는다.
- 비교가 애매한 데이터 타입의 경우 (예를 들어 정수형과 부동 소수점 형): 구축(build) 단계에서 처리할 수 없으므로, 그대로 실행(run) 단계가 수행된다. 하지만 수행 결과에 모든 레코드가 불일치 레코드로 처리된다.

Altibase to Oracle

| Altibase | Oracle | 설명 |
|----------|--------|----|
| SMALLINT | NUMBER | |
| INTEGER | NUMBER | |
| BIGINT | NUMBER | |

| Altibase | Oracle | 설명 |
|----------|-------------------|---|
| NUMERIC | NUMBER | |
| REAL | NUMBER | |
| DOUBLE | NUMBER | |
| FLOAT | FLOAT | |
| DATE | DATE
TIMESTAMP | DATE - TIMESTAMP의 경우 precision이 짧은 쪽을 기준으로 비교하게 된다. 예를 들어 Altibase DATE는 microsecond까지 지원되고, Oracle의 TIMESTAMP는 나노 초까지 지원 가능하다. 이 경우, 레코드의 동일 여부 비교는 나노 초를 기준으로 한다. DATE-DATE의 경우 Oracle DATE 타입은 YYYY-MM-DD HH:MM:SS까지 지원된다. 따라서, 이 경우 Altibase DATE 타입의 데이터에서 소수점 이하의 초는 무시하고 YYYY-MM-DD HH:MM:SS형식으로만 비교하게 된다. |
| CHAR | CHAR | |
| VARCHAR | VARCHAR2 | |
| NCHAR | NCHAR | |
| NVARCHAR | NVARCHAR2 | |

Altibase to MariaDB

| Altibase | MariaDB | 설명 |
|----------|----------|---|
| SMALLINT | SMALLINT | |
| INTEGER | INT | |
| BIGINT | BIGINT | |
| NUMERIC | DECIMAL | |
| REAL | FLOAT | |
| DOUBLE | DOUBLE | |
| FLOAT | N/A | Altibase Float 타입에 해당되는 MariaDB의 데이터 타입 없음. |

| Altibase | MariaDB | 설명 |
|----------|---|---|
| DATE | DATE,
DATETIME,
TIMESTAMP | DATE-DATE: MariaDB의 DATE는 YYYY-MM-DD까지 지원된다. 따라서, Altibase DATE 데이터에서 해당부분만 추출하여 비교한다. DATE-DATETIME: MariaDB의 DATE는 YYYY-MM-DD HH:MM:SS까지 지원된다. 따라서, Altibase DATE 데이터에서 해당부분만 추출하여 비교한다. DATE-TIMESTAMP: MariaDB의 DATE는 YYYY-MM-DD HH:MM:DD Microseconds 0 ~ 6 (default: 0)까지 지원된다. 따라서, Altibase DATE 데이터에서 해당부분만 추출하여 비교한다. |
| CHAR | CHAR | |
| VARCHAR | VARCHAR,
TINYTEXT,
TEXT,
MEDIUMTEXT,
LONGTEXT | |
| NCHAR | CHAR with
Character set | |
| NVARCHAR | VARCHAR
with Character
set | |

4.기타 Utilities

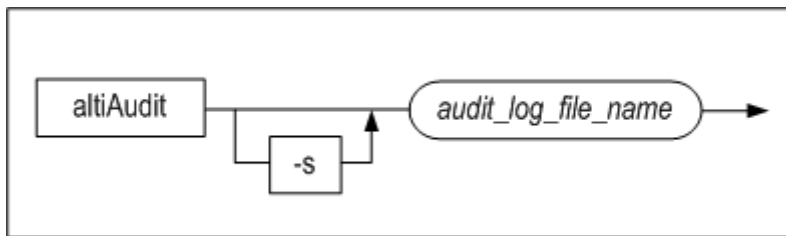
altiAudit

개요

감사가 Altibase 서버에서 수행 중이라면, \$ALTIBASE_HOME/trc 디렉토리에 기본적으로 감사 로그 파일이 생성되고 로그가 기록될 것이다. 감사 로그 파일의 위치는 AUDIT_LOG_DIR 프로퍼티로 변경 가능하다. 이 파일에는 사용자가 읽을 수 없는 바이너리 형태의 감사 로그가 기록되어 있다. altiAudit 유틸리티는 감사 로그 파일을 문자 형태로 변환하여 출력함으로써 사용자가 감사 로그를 분석할 수 있게 한다.

```
altiAudit [-s] {audit_log_file_name}
```

구문



설명

서버가 남긴 감사 로그를 문자 형태로 변환하여 출력한다.

`-s` 옵션을 사용해서 CSV 형태로 출력할 수도 있다.

사용예

아래는 감사 로그를 일반 텍스트 형태로 출력하는 명령어이다.

```
$ altiTudit $ALTIBASE_HOME/trc/alti-1366989680-0.aud
```

아래와 같은 형태의 결과가 출력된다.

[2015/03/05 14:59:29]

Session Info

| | | |
|-----------------|-------------|---------------------------------|
| User Name | = SYS | |
| Session ID | = 1 | |
| Client IP | = 127.0.0.1 | |
| Client Type | = CLI-64LE | |
| Client App Info | = isql | |
| Action | = INSERT | |
| Auto Commit | = 1 | (0:non-autocommit 1:autocommit) |

Query Info

| | | |
|----------------|-----------|---|
| Statement ID | = 65540 | |
| Transaction ID | = 150657 | |
| Execute result | = 4 | (0:failure 1:rebuild 2:retry 3:queue empty 4:success) |
| Fetch result | = 2 | (0:failure 1:success 2:no result set) |
| Success count | = 1 | |
| Failure count | = 0 | |
| Return code | = 0x02000 | |
| Processed row | = 1 | |
| Used memory | = 0 | bytes |
| XA flag | = 0 | (0:non-XA 1:XA) |

Query Elapsed Time

| | |
|-------------------|-----|
| Total time | = 0 |
| Soft prepare time | = 0 |
| Parse time | = 0 |
| Validation time | = 0 |
| Optimization time | = 0 |
| Execution time | = 0 |
| Fetch time | = 0 |

SQL

```
-----  
insert into t1 values ('aaaa', 1)  
-----
```

아래는 감사 로그를 CSV 형태로 출력하는 명령어이다.

```
$ altiAudit -s $ALTIBASE_HOME/trc/alti-1366989680-0.aud
```

아래와 같이 CSV 형태의 결과가 출력된다.

```
1425535169,SYS,1,127.0.0.1,CLI-64LE,isql,INSERT,1,65540,150657,4,2,1,0,1,0,0,0,0,0,0,0,0,0,"insert  
into t1 values ('aaaa', 1)"
```

출력 항목

출력 항목과 의미는 아래 표와 같다.

| 필드 이름 | 타입 | 설명 |
|---------------------|---------|--|
| Session Info | | |
| User Name | 문자열 | 세션에 접속한 사용자 이름 |
| Session ID | INTEGER | 세션 ID |
| Client IP | 문자열 | 클라이언트 IP 주소 |
| Client Type | 문자열 | 접속한 클라이언트의 타입 |
| Client App Info | 문자열 | 애플리케이션 정보 |
| Action | 문자열 | 실행된 구문의 종류 |
| Auto Commit | INTEGER | 0: Non-auto commit mode 1: auto commit mode |
| Query Info | | |
| Statement ID | INTEGER | Statement ID |
| Transaction ID | INTEGER | 트랜잭션 ID |
| Execute result | INTEGER | 수행 결과 0: failure 1: rebuild 2: retry 3: query empty 4: success |
| Fetch result | INTEGER | Fetch 결과 0: failure 1: success 2: no result set |
| Success count | INTEGER | 감사 조건에 부합하는 구문이 성공한 횟수이다. BY SESSION 조건일 경우, 감사 조건에 부합하는 구문의 수행에 성공한 횟수의 누적 값이다. BY ACCESS 조건일 경우, 감사 조건에 부합하는 구문이 성공적으로 수행된 경우 1이 표시된다. |
| Failure count | INTEGER | 감사 조건에 부합하는 구문이 실패한 횟수이다. BY SESSION 조건일 경우, 감사 조건에 부합하는 구문의 수행에 실패한 횟수의 누적 값이다. BY ACCESS 조건일 경우, 감사 조건에 부합하는 구문의 수행이 실패한 경우 1이 표시된다. |

| 필드 이름 | 타입 | 설명 |
|---------------------------|---------|---|
| Return code | INTEGER | 감사 조건에 부합하는 구문이 수행된 결과 코드이다. BY ACCESS 조건일 경우에만, 수행 결과가 표시된다. |
| Processed row | INTEGER | 처리된 레코드의 개수 |
| Used memory | INTEGER | 사용된 메모리 (향후 확장 예정) |
| XA flag | INTEGER | 0: Non-XA 1: XA |
| Query Elapsed Time | | |
| Total time | BIGINT | 쿼리 수행에 소요된 총 시간 |
| Soft prepare time | BIGINT | Prepare 수행에 소요된 시간 |
| Parse time | BIGINT | 파싱 수행에 소요된 시간 |
| Validation time | BIGINT | 정당성 검사에 소요된 시간 |
| Optimization time | BIGINT | 최적화 수행에 소요된 시간 |
| Execution time | BIGINT | 실행에 소요된 시간 |
| Fetch time | BIGINT | Fetch 수행에 소요된 시간 |
| SQL | | |
| | 문자열 | 실행된 SQL 구문 |

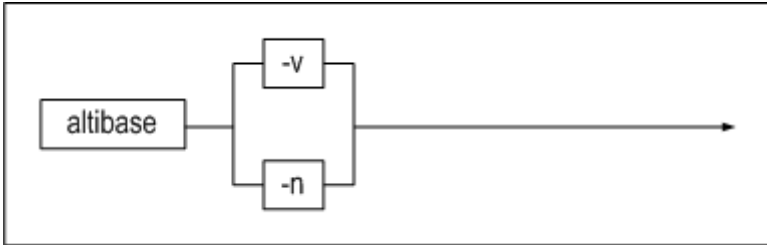
altibase

개요

‘altibase’는 Altibase의 모든 서비스를 관장하는 서버 프로세스를 실행하는 파일이다.

altibase {-v|n}

구문



파라미터

| 파라미터 | 설명 |
|------|-----------------------------------|
| -v | 설치되어 있는 Altibase 제품의 버전 정보를 출력한다. |
| -n | Altibase를 포그라운드에서 수행한다. |

설명

‘altibase’는 Altibase의 모든 서비스를 관장하는 서버 프로세스의 실행 가능(executable) 파일이다.

Altibase를 구동하거나 종료하기 위해 이 커맨드를 직접 사용하면 안 된다. 대신에 iSQL에 SYSDBA 모드로 접속한 뒤, 구동이나 종료 명령을 사용하거나 server 커맨드를 사용해야 한다. server 커맨드는 실제로 Altibase 구동과 종료에 관련된 일련의 명령을 묶어놓은 쉘 스크립트이다. 관련 내용은 본 문서의 server 항목을 참조한다.

보다 자세한 Altibase의 구동 및 종료에 관한 내용은 *iSQL User's Manual*이나 *Getting Started Guide* 를 참조한다.

iSQL로 구동된 Altibase 서버 프로세스는 백그라운드에서 운영된다. 반면, 쉘 프롬프트에서 altibase 커맨드를 -n 옵션으로 수행할 경우 Altibase는 포그라운드에서 구동된다. 이는 Altibase 디버깅을 위한 용도로만 사용되며, 실 운영 시에는 사용하지 않도록 한다.

-v 옵션으로 altibase 커맨드를 실행하면 현재 설치된 Altibase 제품의 버전 정보를 출력한다.

참고사항

Getting Started Guide

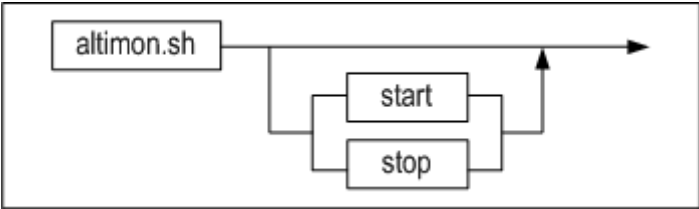
altiMon

개요

Altibase 서버와 altiMon이 구동된 호스트 장비를 모니터링 한다.

`altimon.sh {start | stop}`

구문



파라미터

| 파라미터 | 설명 |
|-------|------------|
| start | altimon 시작 |
| stop | altimon 종료 |

설명

altiMon은 Altibase 서버와 altiMon이 구동된 호스트 장비를 지속적으로 모니터링하여 로그 파일에 수집한 데이터를 기록한다.

altiMon은 주로 OS 정보와 DB 정보를 모니터링하며, 자세한 설명은 altiMon 설정을 참고한다.

start

1. JAVA_HOME 환경 변수를 설정한다.
2. 운영 체제에 맞는 명령을 실행한다.

- Unix 계열

```
$ altimon.sh start
```

3. 구동을 실패하면, \$ALTIBASE_HOME/altiMon/logs/altimon.log 파일을 확인한다.

stop

- Unix 계열

```
$ altimon.sh stop
```

운영체제

altiMon은 OS 정보를 수집하기 위해 C언어로 작성된 PICL 라이브러리를 사용한다.

PICL 라이브러리를 사용할 수 있는 운영체제는 아래와 같다.

| OS | CPU | Version | PICL Library |
|-------|---------------------|-------------------------------|--|
| AIX | ppc64 | OS Version 5.3, 6.1, 7.1 | aix-ppc64-5.so |
| HP-UX | ia64 | IA64 | hpux-ia64-11.sl |
| LINUX | X86_64
ppc64(le) | OS Version 2.6
glibc 2.5이상 | linux-x64.so
linux-ppc64.so |

지원하지 않는 OS 버전에서 아래 방법으로 하위 버전용 PICL이 동작하는지를 확인한 후에 사용할 수도 있다.

```
$ cd $ALTIBASE_HOME/altiMon # java -Dpicl="<picl_lib_file>" -jar lib/com.altibase.picl.jar
```

사용 예)

지원하지 않는 버전의 AIX 운영체제에서 "[aix-ppc64-5.so](#)" PICL 라이브러리가 사용 가능한지 검증하고 altimon을 구동하는 절차이다.

1. 하위 버전용 PICL의 동작 여부 확인

```
$ cd $ALTIBASE_HOME/altiMon  
$ java -Dpicl="aix-ppc64-5.so" -jar lib/com.altibase.picl.jar
```

2. 정상 동작이 검증된 후 \$ALTIBASE_HOME/bin/altimon.sh 파일을 열어서 PICL_LIB 변수에 해당 PICL 파일을 설정한다.

```
PICL_LIB=-Dpicl="aix-ppc64-5.so"
```

3. altimon을 구동한다


```
$ altimon.sh start
```

주의사항

altiMon은 java 1.5 이상에서 동작한다.

Java 버전은 PICL c 라이브러리의 비트 수와 일치하는 것을 선택한다. 예를 들어 PICL c 라이브러리가 [linux-x64.so](#) 인 경우 64 bit Java를 사용해야 한다.

altiMon 설정(Configuration)

altiMon을 사용하기 위해 \$ALTIBASE_HOME/altiMon 디렉토리의 conf 디렉토리에 있는 아래의 파일들을 설정한다.

- [config.xml](#)
- [Metrics.xml](#)
- [GroupMetrics.xml](#)

config.xml

모니터링 대상인 Altibase의 접속 정보와 altiMon 제어 정보를 설정하는 파일이다.

| 태그 이름 | 필수 여부 | 설명 |
|---|-------|--|
| <Altimon Name='String'
monitorOsMetric="true false"> | 필수 | monitorOsMetric 속성은 OsMetric을 측정할 것인지 여부를 지정한다. 기본값: true 사용자 환경과 호환되는 PICL c 라이브러리가 없는 경우에 false로 지정한다. |
| <DateFormat> | 옵션 | 로그 기록시 사용할 날짜 시간 포맷 기본값: yyyy-MM-dd HH:mm:ss 설정 가능한 날짜 형식은 자바 문서 참조한다. |
| <Interval> | 옵션 | 데이터 수집 주기. 기본값: 60 (초) <OSMetric>또는 <SQLMetric> 설정 부분에 Interval을 지정하지 않으면 여기에 설정한 값이 적용된다. 단, <GroupMetric>은 영향을 받지 않는다. |
| <LogDir> | 옵션 | 별도의 디스크 사용시 설정하면 된다. 설정하지 않으면 아래의 디렉토리가 기본으로 설정된다.
\$ALTIBASE_HOME/altiMon/logs |
| <MaintenancePeriod> | 옵션 | 로그 파일 유지 기간 기본값: 3 (일) |

| 태그 이름 | 필수 여부 | 설명 |
|---------------------------------|-------|--|
| <Target Name='String'> | 필수 | 모니터링 대상 데이터베이스 Name: 이름 |
| <HomeDirectory> | 옵션 | 알티베이스 홈 디렉토리를 절대 경로로 설정한다. 설정하지 않으면 ALTIBASE_HOME 환경변수 값이 사용된다. |
| <DBConnectionWatchdogCycle> | 옵션 | DB connection watchdog 실행 주기 DB connection watchdog은 모니터링 대상인 데이터베이스가 shutdown된 후에 일정 시간을 주기로 데이터베이스에 접속을 시도하여 모니터링을 지속할 수 있도록 한다. 기본값: 60 (초) |
| <User> | 옵션 | 접속 사용자. 설정하지 않으면 SYS 사용자로 접속한다. |
| <Password Encrypted="Yes No"> | 필수 | 비밀 번호 Encrypted 속성이 "No"일 때 사용자가 비밀번호를 등록하여도, altimon을 구동하면 Encrypted 속성은 "Yes"로 변경되고 비밀번호도 암호화된다. |
| <Port> | 필수 | 포트 번호 |
| <NLS> | 필수 | NLS_USE |
| <DbName> | 옵션 | 데이터베이스 이름 기본값: mydb |
| <IPv6> | 옵션 | IPv6 사용 여부 기본값: false |
| <ConnectionProperties> | 옵션 | 추가 연결 속성을 지정한다.
예)
<ConnectionProperties>
login_timeout=3;fetch_timeout=60
</ConnectionProperties> |

Metrics.xml

사전 정의된 OS Meric, SQL Metric과 사용자 정의된 OS Metric(Command Metric)을 설정하는 파일이다.

| 태그 이름 | 설명 |
|-------|----|
|-------|----|

| 태그 이름 | 설명 |
|---|---|
| <OSMetric Name='PROC_CPU_USER'
Activate='true' Interval='30' Logging='true'> | <p>사전 정의된 OS Metric을 설정한다. 아래의
Predefined OS Metric들을 사용할 수 있다.</p> <p>TOTAL_CPU_USER: 사용자 모드(user mode)에서
CPU 사용률(%) TOTAL_CPU_KERNEL: 커널 모드
(kernel mode)에서 CPU 사용률(%)</p> <p>PROC_CPU_USER: 사용자 모드(user mode)에서
Altibase 프로세스의 CPU 사용률(%)</p> <p>PROC_CPU_KERNEL: 커널 모드(kernel mode)
에서 Altibase 프로세스의 CPU 사용률(%)</p> <p>TOTAL_MEM_FREE: 사용할 수 있는 메인 메모리
(RAM)의 크기(KB)</p> <p>TOTAL_MEM_FREE_PERCENTAGE: 사용할 수
있는 메인 메모리(RAM)의 비율(%)</p> <p>PROC_MEM_USED: Altibase 프로세스가 사용중인
메인 메모리(RSS)의 크기(KB)</p> <p>PROC_MEM_USED_PERCENTAGE: Altibase
프로세스가 사용중인 메인 메모리(RSS)의 비율(%)</p> <p>SWAP_FREE: 사용 가능한 SWAP의 크기(KB)</p> <p>SWAP_FREE_PERCENTAGE: 전체 SWAP
공간에서 사용 가능한 SWAP의 비율(%)</p> <p>DISK_FREE: 지정한 디스크에서 사용되지 않은
디스크의 크기(KB) DISK_FREE_PERCENTAGE:
지정한 디스크에서 사용되지 않은 디스크의 비율
(%) DISK_FREE, DISK_FREE_PERCENTAGE는
아래의 태그를 추가해야 하며, 'Disk Name'은
고유한 이름으로 설정해야 한다.</p> <p><OSMetric Name='DISK_FREE' Activate='true'>
<Disk Name='disk1'>/home</Disk> </OSMetric>
<OSMetric Name='DISK_FREE' Activate='true'>
<Disk Name='disk2'>/home2</Disk> </OSMetric></p> |

| 태그 이름 | 설명 |
|---|---|
| <code><SQLMetric
Name='MEM_DATABASE_USAGE'
Activate='true' Interval='60' Logging='true'></code> | <p>SQL Metric을 설정하며, Name은 필수로 정의해야 한다.</p> <p>Name: metric 이름(필수)</p> <p>Activate: 경고 동작 여부</p> <p>지정 가능한 값은 true 또는 false이며, 기본값은 true이다. Interval: 데이터 수집 주기</p> <p>설정하지 않으면 config.xml에 설정한 값을 따른다.</p> <p>Logging: 데이터 수집 결과를 파일에 로깅할 지 여부를 설정한다. 지정 가능한 값은 true 또는 false이며, 기본값은 true이다. alert 정보만 기록하고 싶을 때는 false로 지정한다.</p> |
| <code><CommandMetric Name='MEM_VSZ'
Activate='true' Interval='60' Logging='true'></code> | <p>사용자가 정의하는 OS Metric을 설정한다. 아래의 <code><Command></code> 태그에 설정한 커맨드 또는 스크립트를 실행하고, stdout으로 출력되는 값을 측정값으로 사용한다. 해당 metric에 alert을 설정하려면, 이 측정값이 숫자여야 한다.</p> |
| <code><Query></code> | <p>쿼리 설정. SQLMetric일 경우 필수</p> |
| <code><Command></code> | <p>커맨드 또는 스크립트 파일을 설정하며, CommandMetric일 경우 필수이다. 스크립트 파일은 절대 경로 또는 상대 경로로 설정할 수 있다. 상대 경로의 기준은 \$ALTIBASE_HOME/altiMon 디렉토리이다.</p> |
| <code><Alert Activate='true'
ComparisonColumn='ALLOC_MEM_MB'
ComparisonType='gt'></code> | <p>경고 설정. 옵션.</p> <p>threshold 값과 비교되는 측정값은 숫자여야 한다.</p> <p>Activate: Alert 동작 여부, 옵션. 기본값: true</p> <p>ComparisonColumn: threshold 값을 비교할 대상 칼럼. 옵션.</p> <p>ComparisonType: 비교 연산자. 필수.</p> <ul style="list-style-type: none"> -eq: is equal to the threshold value -ne: is not equal to the threshold value -gt: is greater than the threshold value -ge: is greater than or equal to the threshold value -lt: is less than the threshold value -le: is less than or equal to the threshold value |

| 태그 이름 | 설명 |
|--|--|
| <WarningThreshold Value='500'> or
<CriticalThreshold Value='800'> | threshold 값 설정 (critical 또는 warning 두 가지 레벨로 설정 가능) <Alert Activate= true>인 경우 필수.
Value: threshold 값 |
| <ActionScript>db_usage.sh</ActionScript> | threshold 값을 벗어난 경우 수행할 스크립트 파일 이름 설정.
\$ALTIBASE_HOME/altiMon/action_scripts 디렉토리에 해당 스크립트 파일이 있어야 한다.
action script 수행시 metric 이름, 레벨 (CRITICAL 또는 WARNING), threshold 값, 측정값이 인자로 전달된다. |

GroupMetrics.xml

Group Metric을 정의하는 파일이다. OS Metric, Command Metric과 SQL Metric의 조합으로 이루어진다. Metrics.xml에 정의한 metric을 사용해서 GroupMetric을 정의할 수 있다.

Group Metric으로 수집한 데이터는 일반 Metric 측정값이 *.log 파일에 저장되는 것과 달리, CSV 형식의 파일로 저장된다.

| 태그 이름 | 설명 |
|--|--|
| <GroupMetric Name='group1'
Interval='30'> | Group Metric을 설정한다.
Name: metric 이름 (필수)
Activate: 데이터 수집 여부 (옵션) 지정 가능한 값은 true 또는 false이며, 기본값은 true이다.
Interval: 데이터 수집 주기 (필수) |
| <Target
MetricName='PROC_CPU_USER'/> | Group Metric에 포함될 base metric을 지정한다.
MetricName: Metrics.xml에 정의된 OSMetric, Command Metric 또는 SQLMetric의 이름
SQL Metric은 반드시 한 건의 SELECT 문의 실행 결과를 가져야 한다. DISK_FREE 또는 DISK_FREE_PERCENTAGE를 target으로 지정하는 경우 'Metric Name'과 'Disk Name'을 '.'으로 연결한 문자열이 MetricName으로 설정돼야 한다.
<Target MetricName='DISK_FREE.disk1'/> |

| 태그 이름 | 설명 |
|---------------------------|--|
| <Column Name='LOG_GAP' /> | <p>SQL Metric의 경우 select target들 중에서 Group Metric에 포함하고자 하는 칼럼을 지정한다.</p> <p>이 태그를 지정하지 않으면 모든 select target이 해당 Group Metric에 포함된다.</p> <p>Name: select 쿼리에 사용한 target 칼럼. alias를 지정한 경우 alias로 표기한다.</p> |

기타

- action_scripts 디렉토리
 <ActionScript> 태그에 설정하는 스크립트 파일은
 \$ALTIBASE_HOME/altiMon/action_scripts 디렉토리에 위치한다.
 예를 들어 <ActionScript>db_usage.sh</ActionScript>의 'db_usage.sh'
 파일은 action_scripts 디렉토리에 있어야 스크립트 파일이 정상적으로 수행된다.

사용예

altiMon 설정 파일들에 대한 예제이다.

config.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<config>
  <Altimon Name='rnd1'>
    <!-- <LogDir>/home/bethy/arugs/logs</LogDir> -->
    <DateFormat>yyyy-MM-dd HH:mm:ss</DateFormat>
    <MaintenancePeriod>3</MaintenancePeriod>
    <Interval>60</Interval>
  </Altimon>
  <Target Name='Altibase1'>
    <HomeDirectory>/home/bethy/work/altibase_trunk/altibase_home</HomeDirectory>
    <User>sys</User>
    <Password Encrypted="NO">manager</Password>
    <Port>20020</Port>
    <DbName>mydb</DbName>
    <NLS>KSC5601</NLS>
    <IPv6>FALSE</IPv6>
  </Target>
</config>
```

OSMetrics.xml

```

<?xml version="1.0" encoding="UTF-8"?>

<Metrics>
  <OSMetric Name='PROC_CPU_USER' Activate='true' Interval='60'>
    <Alert Activate='true' ComparisonType='gt'>
      <WarningThreshold Value='80'>
        <ActionScript>cpu_act.sh</ActionScript>
      </WarningThreshold>
    </Alert>
  </OSMetric>

  <SQLMetric Name='MEM_DATABASE_USAGE' Activate='true' Interval='30'>
    <Query>select
      trunc(mem_alloc_page_count*32/1024, 2) as alloc_mem_mb,
      trunc(mem_free_page_count*32/1024, 2) as free_mem_mb
    from v$database</Query>
    <Alert Activate='true' ComparisonColumn='ALLOC_MEM_MB' ComparisonType='GT'>
      <CriticalThreshold Value='8000' >
        <ActionScript>db_usage.sh</ActionScript>
      </CriticalThreshold>
    </Alert>
  </SQLMetric>

  <CommandMetric Name='MEM_VSZ'>
    <Command>scriptsDir/vsz.sh</Command>
    <Alert Activate='true' ComparisonType='gt'>
      <WarningThreshold Value='100000000'> <!-- in kB -->
        <ActionScript>mem_act.sh</ActionScript>
      </WarningThreshold>
    </Alert>
  </CommandMetric>
</Metrics>

```

GroupMetrics.xml

```

<GroupMetrics>
  <GroupMetric Name='group1' Interval='30'>
    <Target MetricName='TOTAL_CPU_USER' />
    <Target MetricName='PROC_CPU_USER' />
    <Target MetricName='LOGFILE_COUNT'>
      <Column Name='LOG_GAP' />
    </Target>
  </GroupMetric>
  <GroupMetric Name='group2' Interval='60'>
    <Target MetricName='PROC_MEM_USED_PERCENTAGE' />
    <Target MetricName='MEM_VSZ' />
    <Target MetricName='LOGFILE_COUNT' />
  </GroupMetric>
</GroupMetrics>

```

출력 항목

다음과 같은 형식으로 아래의 디렉토리에 출력된다.

logs 디렉토리

- altimon.log
altiMon 데몬의 각종 로그 (info, warn, error)를 기록하는 파일이다.
- alert.log
<Alert> 설정에 해당하는 데이터를 기록하는 파일이다.
- OsMetrics.log
모든 OS Metric들에 의해 수집된 데이터를 기록하는 파일이다.
- [SQLMetric_Name].log
SQL Metric에 의해 수집된 데이터가 Metric별로 log 파일에 기록된다.
- [GroupMetric_Name].csv
Group Metric에 의해 수집된 데이터가 Metric별로 csv 파일에 기록된다.
- report.html
사용자가 설정한 Configuration을 html 형식으로 리포트하는 파일이다.

logs/archive 디렉토리

logs 디렉토리의 altimon.log, *.csv 제외한 모든 파일이 archive/YYYY-MM-DD 디렉토리로 매일 백업된다.

이 디렉토리들 중에서 <MaintenancePeriod>에 설정한 유지 기간이 지난 디렉토리는 매일 1:50am에 삭제된다.

logs/csv_backup 디렉토리

Group Metric은 설정이 변경되지 않는 한 계속 같은 csv 파일에 측정 데이터가 추가된다.

그러나 metric 설정이 변경된 경우, 즉 Group Metric의 대상이 되는 target이 변경된 경우에는 기존의 csv 파일이 logs/csv_backup 디렉토리로 백업되고 동일한 이름의 새로운 파일에 데이터가 기록된다.

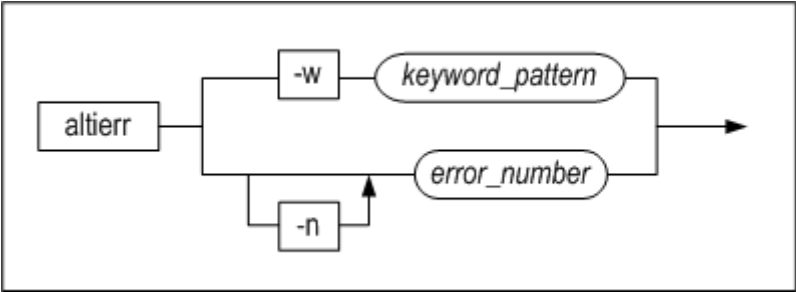
altiern

개요

altiern은 Altibase 서버 에러 코드에 대한 상세한 설명을 검색하여 출력하는 유틸리티이다. 에러 번호나 에러 메시지 패턴을 이용하여 검색이 가능하다.

```
altiern {-w keyword pattern | [-n] error number}
```

구문



파라미터

파라미터	설명
-w	지정한 검색 패턴을 포함하는 에러 메시지를 찾는다. 검색 패턴을 포함하는 모든 에러 메시지를 출력한다.
-n	에러 번호를 사용하여 검색한다. 에러 번호는 16진수/양의 정수/음의 정수 형태로 입력 가능하다. 에러 코드와 정확히 일치하는 레코드만 검색된다. 에러 번호를 사용하여 검색할 경우 파라미터(“-n”)를 생략할 수 있다.

설명

altiern은 Altibase 서버의 에러 코드 번호나 에러 메시지에 포함된 문자열을 이용하여 에러에 대한 상세한 설명을 검색하고 출력해 주는 유틸리티이다.

에러 상세 설명에는 에러 코드 번호, 에러 코드 문자열, 에러 설명, 에러가 발생한 원인, 그리고 에러에 대한 조치 사항이 포함된다.

Altibase 서버는 에러 상황 발생시 해당 코드를 “ERR-{에러번호}” 형태로 altibase_boot.log에 출력한다. 여기에서 에러 번호는 16진수 값으로, 해당 에러에 대한 상세 설명을 검색하기 위해 altiterr을 다음과 같이 사용할 수 있다.

```
For 'ERR-00015'  
$ altiterr 0x00015  
$ altiterr -w 00015  
$ altiterr 21
```

C/C++ Precompiler나 ODBC로 작성된 애플리케이션의 경우 SQL 관련 기능 수행 시 SQLCODE나 ODBC 함수의 리턴 코드에 에러 코드가 설정이 된다. 이 때 설정되는 에러 코드는 음의 정수 값이 설정되며, 해당 에러에 대한 상세 설명을 검색하기 위해 다음과 같이 altiterr을 사용할 수 있다.

```
For -266286  
$ altiterr -266286  
$ altiterr 266286  
$ altiterr 0x4102E
```

altiterr은 에러 메시지에 포함된 키워드를 이용한 검색을 지원하며, 이 경우 여러 개의 레코드가 검색될 수 있다. 키워드 패턴을 이용한 검색은 다음과 같다.

```
$ altiterr -w connect  
$ altiterr -w “does not”
```

참고 사항

Error Message Reference 참조

altipasswd

개요

SYSDBA 모드 접속을 위한 SYS 사용자의 암호를 변경한다. 데이터베이스 상에서 ALTER USER 문으로 암호를 변경하는 경우, 이 커맨드로 한 번 더 암호 변경 작업을 해야 한다. ALTER USER 문으로만 암호를 변경했을 경우, 데이터베이스 구동, 종료 등 SYSDBA 작업을 했을 때 오류가 발생하게 된다.

altipasswd

구문



설명

SYS 사용자의 암호를 변경한다.

사용예

SYS 사용자의 암호 “manager”를 “manager1234”로 변경한다면 쉘 프롬프트 상에서 다음과 같이 수행한다.

```
$ altipasswd
Previous Password : manager
New Password : manager1234
Retype New Password : manager1234
```

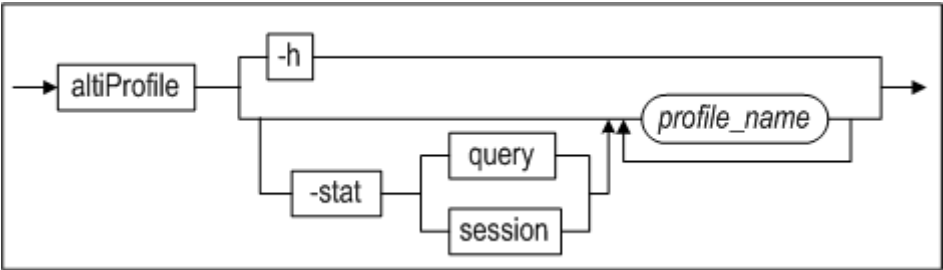
altiProfile

개요

Altibase 서버 내에서 수행되는 작업과 서버의 상태 정보를 파일로 기록하여 분석할 수 있다. 서버가 남긴 상태 파일을 프로파일이라 하며 altiProfile은 이 프로파일을 문자 형태로 변환하여 출력하거나, STATEMENT 관련 데이터만 추출하여 통계 정보를 구축하여 출력한다. 이 정보를 사용하여 사용자는 시스템의 상태를 분석할 수 있다.

```
altiProfile [-stat query|session] {profile_name [profile_name2 [profile_name3] ...]}
```

구문



파라미터

파라미터	설명
-h	도움말을 출력한다.
-stat query/session	서버가 남긴 프로파일에서 STATEMENT의 통계 정보를 구축하여, 텍스트와 CSV 형태의 파일로 출력한다. 구축되는 통계 정보에 대한 자세한 설명은 아래의 "사용 방법"을 참고하기 바란다.

설명

서버가 남긴 프로파일을 문자 형태로 변환하거나, STATEMENT 관련 데이터만 추출하여 통계 정보를 구축한다.

사용예

```
iSQL> ALTER SYSTEM SET QUERY_PROF_FLAG = 1;
Alter success.
iSQL> ALTER SYSTEM SET TIMED_STATISTICS = 1;
Alter success.
iSQL>    --(Execute an SQL query here.)

$ cd $ALTIBASE_HOME/trc
$ altiProfile alti-1286503704-0.prof

$ altiProfile -stat query $ALTIBASE_HOME/trc/*.prof
```

사용 방법

서버의 상태 및 작업 내용을 기록하려면 QUERY_PROF_FLAG 프로퍼티를 0보다 큰 값으로 설정하여야 한다. QUERY_PROF_FLAG 프로퍼티의 값에 따라 다음의 정보들이 기록된다.

값	이름	설명
0		기록하지 않음
1	[STATEMENT]	SQL 문이 실행될 때마다 실행된 SQL문, 실행시간, 실행정보, 색인 및 디스크 접근 정보 출력. 단, 실행시간이 제대로 출력되게 하려면 TIMED_STATISTICS 프로퍼티를 1로 설정해야 한다. TIMED_STATISTICS 프로퍼티에 대한 자세한 설명은 <i>General Reference</i> 를 참조하기 바란다.
2	[BIND]	SQL 문이 실행될 때마다 BIND 파라미터 출력

값	이름	설명
4	[PLAN]	SQL 문이 실행될 때마다 실행계획 출력
8	[SESSION STAT]	3초마다 세션 정보 출력(V\$SESSTAT 정보)
16	[SYSTEM STAT]	3초마다 시스템 정보 출력(V\$SYSSTAT 정보)
32	[MEMORY STAT]	3초마다 메모리 정보 출력(V\$MEMSTAT 정보)

위의 값을 조합하여 원하는 정보를 기록하도록 설정한다. 예를 들어, 프로퍼티를 1+4+32=37로 설정하면 SQL 문이 실행될 때마다 SQL 문의 실행정보와 실행계획을 출력하고 3초마다 메모리 정보를 출력한다.

프로퍼티가 설정되면, 서버는 alti-#시간-#번호.prof 이름의 파일에 정보를 남긴다. 사용자는 altiProfile 명령어로 상태 파일을 변환하여 분석할 수 있다.

통계 정보 출력

altiProfile은 -stat 옵션을 사용해서 실행된 SQL문에 대한 통계 정보를 구축하고 출력할 수 있다. 이 정보는 튜닝 대상이 되는 SQL문을 찾는 데 도움이 된다.

-stat query 옵션을 실행하여 구축되는 통계 정보는 아래와 같다.

- COUNT: QUERY가 실행된 횟수
- AVG: QUERY 수행에 걸린 평균 시간 (microseconds 단위)
- TOTAL: QUERY 수행에 걸린 시간 합계 (microseconds 단위)
- MIN: QUERY 수행에 걸린 최소 시간 (microseconds 단위)
- MAX: QUERY 수행에 걸린 최대 시간 (microseconds 단위)
- SUCCESS: QUERY 실행에 성공한 횟수
- FAIL: QUERY 실행에 실패한 횟수
- QUERY: 수행된 SQL문

-stat session 옵션으로 수행한다면, query 옵션을 실행하여 구축되는 통계 정보에 SESSION ID가 추가된다.

아래는 \$ALTIBASE_HOME/trc 디렉토리에 생성된 모든 프로파일을 분석하여 SQL문 별로 통계 정보를 구축하는 altiProfile 실행 예제이다.

```
$ altiProfile -stat query $ALTIBASE_HOME/trc/*.prof

### Processing [/altibase_home/trc/alti-1423543095-0.prof]...
100% [=====]

### Writing CSV File [alti-prof-stat-1423543711.csv]...

### Writing TEXT File [alti-prof-stat-1423543711.txt]...

### Successfully done.
```

위에 실행 로그를 살펴보면, 통계 정보는 CSV 형식과 텍스트 형식의 파일로 저장된다. 출력된 파일의 이름은 'alti-prof-stat-#시간.csv'과 'alti-prof-stat-#시간.txt'로 자동 생성된다.

다음은 텍스트 파일의 내용이다. 통계 정보는 TOTAL 값으로 정렬되어 출력된다.

```
$cat alti-prof-stat-1423543711.txt
```

COUNT	AVG	TOTAL	MIN	MAX	SUCCESS	FAIL	QUERY
5	0.003730	0.018650	0.003035	0.004640	5	0	DROP VIEW REVENUE
5	0.003523	0.017616	0.003004	0.003745	5	0	CREATE VIEW REVENUE (
...							

다음은 CSV 파일의 내용이다. 텍스트 파일과 동일한 내용이 CSV 형식으로 출력된다. CSV 형식의 파일은 excel 같은 스프레드시트 애플리케이션을 사용해서 사용자가 원하는 포맷으로 가공할 수 있다.

```
$ cat alti-prof-stat-1423543711.csv
```

COUNT	AVG	TOTAL	MIN	MAX	SUCCESS	FAIL	QUERY
5	0.003730	0.018650	0.003035	0.004640	5	0	"DROP VIEW REVENUE"
5	0.003523	0.017616	0.003004	0.003745	5	0	"CREATE VIEW REVENUE (
...							

주의 사항

프로파일링 기능을 동작시킬 경우, 서버 내에서 실행되는 모든 SQL 문에 대해 실행 정보를 기록하며 또한 매 3초마다 세션 및 시스템 정보 등 서버의 상태를 기록하므로 시스템에 부하를 줄 수 있다.

또한 프로파일링 기능을 설정할 때 프로파일이 커져 디스크가 꽉 찰 수 있으므로 신중해야 한다..

출력항목

다음과 같은 형식으로 출력된다.

```
[STATEMENT]
..
[BIND]
..
[PLAN]
..
[SESSION STAT]
..
[SYSTEM STAT]
..
[MEMORY STAT]
..
```

각 정보는 아래와 같은 형식으로 출력된다.

[STATEMENT]

다음의 표는 실행한 STATEMENT에 대한 정보를 나타낸다.

필드 이름	값	설명
SQL	문자열	실행된 SQL 문
User Info		
User ID	INTEGER	사용자 식별자
Client PID	BIGINT	클라이언트 프로세스 아이디
Client Type	VARCHAR(40)	접속한 클라이언트의 타입
Client AppInfo	VARCHAR(128)	클라이언트의 정보 문자열
Elapsed Time for this SQL statemen		
Total	BIGINT	총 쿼리 수행 시간
Parse	BIGINT	파싱 수행 시간
Valid	BIGINT	정당성 검사 수행 시간
Optim	BIGINT	최적화 수행 시간

필드 이름	값	설명
Execu	BIGINT	실행 수행 시간
Fetch	BIGINT	Fetch 수행 시간
Query Execute Info		
EXECUTE Result	INTEGER	0: failure 1: rebuild 2: retry 3: queue empty 4: success
Optimizer Mode	BIGINT	최적화 모드
Cost Mode	BIGINT	최적화 비용
Used Memory	BIGINT	향후 확장 예정
SUCCESS SUM	BIGINT	실행 성공 횟수의 총합
FAILURE SUM	BIGINT	실행 실패 횟수의 총합
PROCESSED ROW	BIGINT	처리된 레코드 개수
Result Set Info		
FETCH Result	INTEGER	0: failure 1: success 2: no results
Index Access Info		
Memory Full Scan Count	BIGINT	메모리 테이블에 대한 Full Scan이 발생한 횟수
Memory Index Scan Count	BIGINT	메모리 테이블에 대한 Index Scan이 발생한 횟수
Disk Full Scan Count	BIGINT	디스크 테이블에 대한 Full Scan이 발생한 횟수
Disk Index Scan Count	BIGINT	디스크 테이블에 대한 Index Scan이 발생한 횟수
Disk Access Info		
READ DATA PAGE	BIGINT	질의 수행시 디스크 페이지에 대한 읽기 연산 횟수
WRITE DATA PAGE	BIGINT	사용하지 않음

필드 이름	값	설명
GET DATA PAGE	BIGINT	질의 수행 시 디스크 페이지에 대한 버퍼 접근 횟수
CREATE DATA PAGE	BIGINT	질의 수행 시 디스크 페이지 생성 횟수
READ UNDO PAGE	BIGINT	질의 수행 시 UNDO 영역 디스크 페이지에 대한 읽기 연산 횟수
WRITE UNDO PAGE	BIGINT	사용하지 않음
GET UNDO PAGE	BIGINT	질의 수행 시 UNDO 영역 디스크 페이지에 대한 버퍼 접근 횟수
CREATE UNDO PAGE	BIGINT	질의 수행 시 UNDO 영역 디스크 페이지 생성 횟수

[BIND]

SQL 문에 바인드 되는 변수들에 대한 정보이다.

[PLAN]

실행된 SQL 문의 실행 계획이 출력된다. 실행 계획에 관한 자세한 내용은 *Performance Tuning Guide*를 참고한다.

[SESSION STAT]

매 3초마다 V\$SESSTAT 정보가 출력된다. V\$SESSTAT의 자세한 내용은 *General Reference*의 성능 뷰 부분을 참고한다.

[SYSTEM STAT]

매 3초마다 V\$SYSSTAT 정보가 출력된다. V\$SYSSTAT의 자세한 내용은 *General Reference*의 성능 뷰 부분을 참고한다.

[MEMORY STAT]

매 3초마다 V\$MEMSTAT 정보가 출력된다. V\$MEMSTAT의 자세한 내용은 *General Reference*의 성능 뷰 부분을 참고한다.

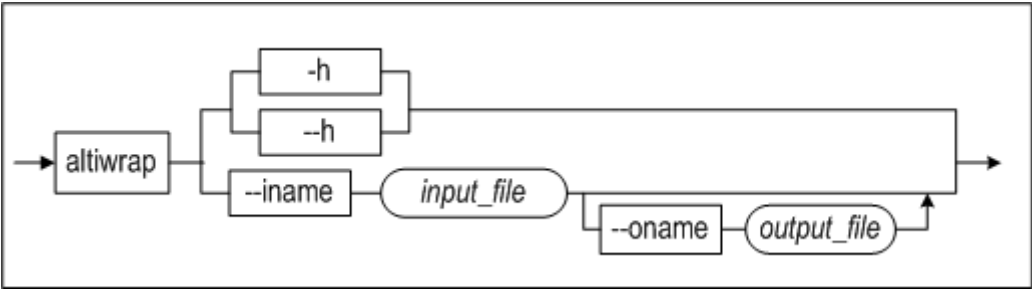
altwrap

개요

altiwrap은 PSM(Persistent Stored Module)으로 작성된 코드 프로그램을 암호화할 수 있는 유틸리티이다. altiwrap을 사용하여 코드를 암호화하면 다른 개발자에게 저장 프로시저 및 저장 함수 등의 PSM 코드가 노출되는 것을 방지할 수 있다.

```
altiwrap {--iname input_file} [--oname output_file]
```

구문



파라미터

파라미터	설명
-h/--h	도움말을 출력한다.
--iname	암호화 할 파일의 이름을 지정한다. 파일의 확장자를 생략하면, '.sql'로 인식한다.
--oname	코드 프로그램이 암호화되어 저장될 파일의 이름을 지정한다. 파일의 확장자를 생략하면, '.plb'로 저장된다.

설명

altiwrap을 실행하여 개발자가 작성한 저장 프로시저 및 저장 함수의 코드 프로그램을 암호화하여, 다른 개발자 또는 경쟁사에 노출되는 것을 방지할 수 있다.

Altibase가 암호화할 수 있는 구문은 아래와 같다.

- CREATE [OR REPLACE] PROCEDURE
- CREATE [OR REPLACE] FUNCTION
- CREATE [OR REPLACE] TYPESET
- CREATE [OR REPLACE] PACKAGE
- CREATE [OR REPLACE] PACKAGE BODY

주의 사항

- 코드 프로그램이 암호화 된 후에는 프로그램을 변경할 수 없다. 프로그램을 변경할 때에는 암호화 전의 원본 프로그램에서 변경한 후 다시 암호화해야 한다.

- 트리거(trigger)는 암호화할 수 없다.
- 암호화된 프로그램 코드는 구문(syntax) 및 시맨틱(semantic)의 오류를 검사하지 못한다.

사용예제

altwrap을 사용하여 sample1.sql 파일을 암호화한 후, 암호화된 파일을 출력한다.

아래는 sample1.sql 파일이다.

```
iSQL> create or replace procedure proc1 as
type arr1 is table of char(30) index by integer;
v1 arr1;
begin
v1[0] := 'create or replace';
v1[1] := 'typeset';
v1[2] := 'is';
v1[3] := 'success';
println( v1[0] || v1[1] || v1[2] || v1[3] || '!' );
end;
/
```

sample1.sql 파일을 암호화한다.

```
$ altwrap --iname sample1.sql --oname --sample1.plb
```

암호화된 코드 프로그램을 isql에서 실행하면 된다.

```
iSQL> @sample1.plb
iSQL> create or replace procedure proc1 WRAPPED
'MjQz
MTk2
AAhjcmVhdGUgb3IgcmlVwbGFjZSBwcm9jZWR1cQQAQxIGFzCnR5cGUgYXJyMSBpcyB0YWJsQAYAAWgY2hhcigzMCKgaW5kZXggYn1AAQZ0ZWdlc
';
/
Create success.
iSQL> exec proc1;
create or replace typeset is success!
Execute success.
```

awrite

개요

로그 파일을 생성할 때 사용되는 시스템 콜의 응답 시간을 출력한다. 출력 값은 LOG_CREATE_METHOD 프로퍼티에서 시스템 콜을 결정하는데 참고된다.

구문



설명

write()와 fallocate() 시스템 콜의 응답 시간을 출력한다.

사용예

아래와 같은 결과를 출력한다.

```
$ awrite
fallocate to expand file size to 1GB
Elapsed Time ==>          2.564 seconds
write to expand file size to 1GB
Elapsed Time ==>          4.020 seconds
```

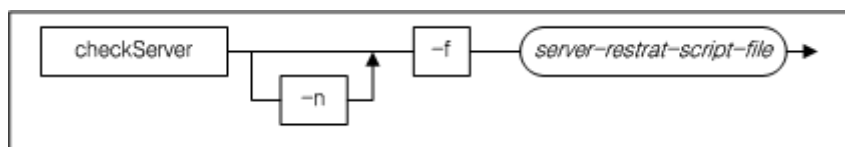
checkServer

개요

Altibase 프로세스를 감시하며, 프로세스가 비정상 종료하면 지정된 스크립트 파일을 실행한다.

```
checkServer [-n] {-f server-restart-script-file}
```

구문



파라미터

파라미터	설명
-n	checkServer를 foreground에서 수행 파라미터 생략시 checkServer는 background에서 수행

파라미터	설명
-f	Altibase 종료시 실행할 스크립트 파일의 이름

설명

checkServer는 주기적으로 Altibase 프로세스가 실행 중인지를 검사하여 Altibase 프로세스가 종료 되었을 경우 사용자가 지정한 스크립트를 수행한다.

Altibase가 비정상 종료 되었을 경우 Altibase 재구동 스크립트를 수행하는 것이 일반적이며, 재구동 스크립트는 다음과 같은 형식으로 작성할 수 있다.

- Altibase 구동 스크립트 '[restart.sh](#)'

```
#!/bin/sh
${ALTIBASE_HOME}/bin/server start
```

checkServer는 구동시 \$ALTIBASE_HOME/trc 디렉토리에 checkServer.pid와 checkServer.log 파일을 생성한다. checkServer.pid파일은 또 다른 checkServer가 동시에 수행되지 못하도록 하는 락의 역할을 하며 checkServer.log에는 checkServer의 동작 상태가 주기적으로 기록된다.

checkServer를 'kill -9'등의 방법으로 강제 종료했을 경우에 checkServer.pid 파일이 디렉토리에 계속 남아있게 되어 checkServer를 다시 수행할 수 없게 된다. 이 때는 checkServer.pid파일을 삭제하면 checkServer를 정상적으로 시작할 수 있다.

checkServer를 종료하기 위해선 반드시 killCheckServer 유틸리티를 이용한다.

주의 사항

checkServer는 Altibase 서버가 "server stop" 커맨드를 사용하지 않고 종료된 경우에만, 지정한 재시작 스크립트를 실행한다. "server stop" 커맨드를 사용하여 Altibase 서버를 정상 종료하는 경우, checkServer 프로세스도 역시 종료되어, 재시작 스크립트를 실행하지 않게 된다. 이것은 checkServer가 "server stop" 커맨드를 사용하여 종료하는 것만 정상 종료라고 판단하기 때문이다.

사용예

셸 프롬프트 상에서 다음과 같이 수행한다.

```
$ checkServer -f restart.sh &
```

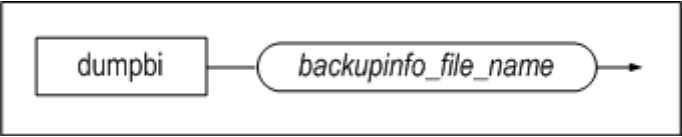
dumpbi

개요

dumpbi는 이진 형식으로 저장된 backupinfo 파일의 백업 정보를 문자 형식으로 보여준다.

```
dumpbi <backupinfo_file_name>
```

구문



설명

backupinfo 파일의 내용을 텍스트 형식으로 보여준다.

사용예

셸 프롬프트 상에서 다음과 같이 수행한다.

```
$ dumpbi backupinfo
```

출력 항목

dumpbi는 backupinfo 의 백업 정보를 다음과 같은 섹션으로 구분하여 출력한다.

[BACKUO INFO FILE HDR]

필드 이름	값(단위 바이트)	설명
Backup info slot count	0 ~ unsigned int 타입의 최대값	저장되어 있는 backupinfo slot의 수 (= 현재까지 백업된 파일의 수)
Last backup LSN	FileNo, Offset	가장 최근에 수행된 백업 시점의 LSN (유효한 backupinfo 파일인지 검증하기 위한 값)
Database name	문자열	데이터베이스 이름

[BACKUP INFO SLOT]

필드 이름	값(단위 바이트)	설명
Slot index	0 ~ unsigned int 타입의 최대값	slot의 순번
Begin backup time	YYYY-MM-DD HH:MM:SS	백업 시작 일시
End backup time	YYYY-MM-DD HH:MM:SS	백업 완료 일시
Incremental backup chunk cnt	0 ~ unsigned int 타입의 최대값	데이터파일에서 변경된 페이지를 포함한 incremental chunk의 수 (= 백업된 incremental chunk의 수)
Incremental backup chunk size	0 ~ unsigned int 타입의 최대값	백업이 수행될 당시의 INCREMENTAL_BACKUP_CHUNK_SIZE의 값
Backup target	1: DATABASE 2: TABLESPACE	백업 대상
Backup level	1: level 0 2: level 1	백업 레벨
Backup Type	1: full backup 2: differentail backup 3: cumulative backup	백업 유형
Tablespace ID	0 ~ unsigned short 타입의 최대값	백업 대상 데이터파일이 속한 테이블스페이스의 ID
File ID	0 ~ unsigned short 타입의 최대값	백업 대상 데이터파일의 ID
Original file size	0 ~ unsigned long 타입의 최대값	백업된 시점의 데이터파일 크기
Backup Tag	문자열	백업 태그 이름
Backup file name	문자열	백업 파일의 경로와 이름

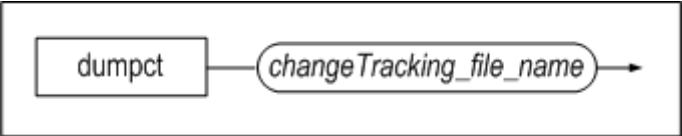
dumpct

개요

dumpct는 이진 형식으로 저장된 changeTracking 파일의 정보를 문자 형식으로 보여준다.

```
dumpct <changeTracking_file_name>
```

구문



설명

changeTracking 파일의 내용을 텍스트 형식으로 보여준다.

사용예

셸 프롬프트 상에서 다음과 같이 수행한다.

```
$ dumpct changeTracking
```

출력

dumpct는 changeTracking 파일 내의 변경 추적 정보를 다음과 같은

[^]:

으로 구분하여 출력한다.

[CHANGE TRACKING FILE HDR]

필드 이름	값(단위 바이트)	설명
Change tracking body count	0 ~ unsigned int 타입의 최대값	changeTracking 파일이 가진 body의 수 (changeTracking 파일은 header와 body로 나뉜다. body의 크기는 10Mbytes이며 공간이 부족하면 바디 단위로 확장된다.)
Incremental backup chunk size	0 ~ unsigned int 타입의 최대값	changeTracking 파일 생성시점의 INCREMENTAL_BACKUP_CHUNK_SIZE 프로퍼티 값

필드 이름	값(단위 바이트)	설명
Last flush LSN	FileNo, Offset	메모리상의 변경된 데이터가 파일로 기록된 시점의 LSN
Database name	문자열	데이터베이스 이름

[CHANGE TRACKING FILE BODY]

필드 이름	값(단위 바이트)	설명
Change tracking body ID	0 ~ unsigned int 타입의 최대값	body의 ID
Flush LSN	FileNo, Offset	body가 flush 된 시점의 LSN(파일 검증용)
Datafile descriptor slot		
Slot ID	0 ~ unsigned int 타입의 최대값	slot의 ID
Tracking state	0: 추적 비활성화 1: 추적 활성화	데이터파일 변경 사항 추적 상태
Tablespace type	0: memory TBS 1: disk TBS	테이블스페이스 유형
Page size	0 ~ unsigned int 타입의 최대값	페이지 크기
Bitmap extent count	0 ~ unsigned short 타입의 최대값	할당된 비트맵 익스텐트 개수
Current tracking list ID	0 ~ unsigned short 타입의 최대값	현재 추적중인 bitmap extent list의 ID
Differential0 BmpExt list		
List	0 ~ unsigned int 타입의 최대값	bitmap extent list에 매달린 block ID

필드 이름	값(단위 바이트)	설명
Hint	0 ~ unsigned int 타입의 최대값	
Differential1 BmpExt list		
List	0 ~ unsigned int 타입의 최대값	bitmap extent list에 매달린 block ID
Hint	0 ~ unsigned int 타입의 최대값	
Cumulative BmpExt list		
List	0 ~ unsigned int 타입의 최대값	bitmap extent list에 매달린 block ID
Hint	0 ~ unsigned int 타입의 최대값	
Tablespace ID	0 ~ unsigned short 타입의 최대값	데이터파일이 속한 테이블스페이스의 ID
File ID	0 ~ unsigned short 타입의 최대값	데이터파일의 ID

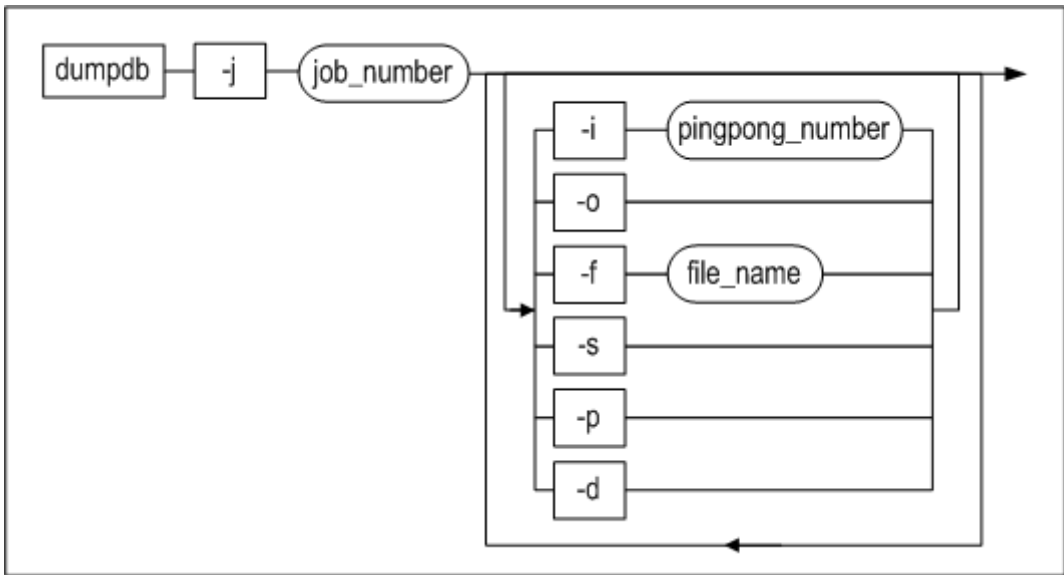
dumpdb

개요

dumpdb는 메모리 체크포인트 이미지 파일로부터 메모리 테이블스페이스에 대한 정보를 출력하거나 또는 메모리 테이블스페이스의 증분 백업 파일의 내용을 문자 형식으로 보여주는 유틸리티이다.

```
dumpdb {-j job_number } [-i pingpong_number] [-o] [-f file_name] [-s] [-p] [-d]
```

구문



파라미터

파라미터	설명
-j job_number	<p>어떤 정보를 출력할 것인지를 지정한다. 지정 가능한 값과 각 값에 대해 추가로 지정할 수 있는 인자는 다음과 같다.</p> <p>0: META (-s -f)</p> <p>1: TABLESPACE (-s -f) 2: TABLESPACE-FLI (-s -d)</p> <p>3: TABLESPACE-FREE-PAGE-LIST (-s)</p> <p>4: TABLE (-o -d)</p> <p>5: TABLE-ALLOC-PAGE-LIST (-o)</p> <p>6: PAGE (-s -p -d)</p> <p>7: INCREMENTAL_BACKUP_META (-f)</p>
-i pingpong_number	<p>체크포인트 이미지 파일의 핑퐁 번호이다. 생략하면 0이 사용된다.</p>
-o	<p>분석할 객체의 ID를 지정한다.</p>
-f file_name	<p>체크포인트 이미지 파일의 이름을 지정한다.</p>
-s	<p>분석할 테이블스페이스의 ID를 지정한다.</p>
-p	<p>분석할 페이지의 ID를 지정한다.</p>
-d	<p>정보를 상세히 출력한다.</p>

설명

메모리 체크포인트 이미지 파일을 분석하여 메타 헤더, 페이지 등의 정보를 텍스트 형태로 출력하거나 또는 메모리 테이블스페이스의 증분 백업 파일에서 백업 정보를 텍스트 형태로 출력한다.

이 유틸리티는 디스크에 저장된 체크포인트 이미지 파일을 분석하기 때문에, Altibase 서버가 구동 중이 아니더라도 이 유틸리티를 수행하여 데이터베이스에 생성되어 있는 스키마를 확인할 수 있다.

하지만 DDL 직후 서버가 비정상 종료되어 갱신된 스키마가 디스크에 기록되지 못하면 해당 정보를 확인할 수 없다.

사용예

셸 프롬프트 상에서 다음과 같이 수행한다.

```
$ dumpdb -j 1
$ dumpdb -j 1 -s 0
$ dumpdb -j 2
$ dumpdb -j 3
$ dumpdb -j 4
$ dumpdb -j 4 -d
$ dumpdb -j 4 -o 65536
$ dumpdb -j 5 -o 65536
$ dumpdb -j 6 -s 0 -p 4
```

<예제 1> 다음은 메모리 테이블스페이스에 관한 정보를 출력한다. -s 인자를 추가하여 특정 테이블스페이스에 관한 정보만 출력할 수도 있다.

```
$ dumpdb -j 1
```

<예제 2> 다음은 메모리 테이블스페이스의 FreeListInfo(FLI) 페이지에 관한 정보를 출력한다. -s 인자를 추가하여 특정 테이블스페이스에 관한 정보만 출력할 수 있고, -d 인자를 추가해서 FLI 페이지의 유효하지 않은 부분도 출력할 수 있다.

```
% dumpdb -j 2
```

<예제 3> 다음은 메모리 테이블스페이스의 FreePage들을 출력한다. -s 인자를 추가하여 특정 테이블스페이스에 관한 정보만 출력할 수도 있다.

```
% dumpdb -j 3
```

<예제 4> 다음은 데이터베이스에 생성되어 있는 모든 객체 정보를 출력한다. -o 인자(아래 예제에서 SelfOID 항목)를 추가하여 특정 객체에 관한 정보만 상세히 출력할 수도 있다. -d 인자를 추가해서 객체의 칼럼 정보와 인덱스 정보 등도 출력할 수 있다.

```
% dumpdb -j 4
```

<예제 5> 다음은 데이터베이스에 생성되어 있는 모든 객체의 정보를 칼럼 정보와 인덱스 정보 등과 함께 출력한다.

```
% dumpdb -j 4 -d
```

<예제 6> 다음은 특정 테이블의 스키마와 데이터를 출력한다.

```
% dumpdb -j 4 -o 65568
```

<예제 7> 다음은 특정 테이블이 사용하는 페이지 목록을 출력한다.

```
% dumpdb -j 5 -o 65568
```

<예제 8> 다음은 메모리 데이터베이스의 특정 페이지를 출력한다.

```
% dumpdb -j 6 -s 0 -p 4
```

<예제 9> 다음은 증분 백업 파일에 대해 dumpdb를 수행하여 백업 정보를 출력한다.

```
% dumpdb -j 7 -f SYS_TBS_MEM_DATA-0-0_TAG_MONDAY.ibak
dumpdb: Release 6.3.1.0.0 - Production on Oct 31 2012 22:12:21
(c) Copyright 2001 ALTIBase Corporation. All rights reserved.
```

```
[BEGIN CHECKPOINT IMAGE HEADER]
Binary DB Version      [ 6.2.1 ]
Redo LSN               [ 1, 5867599 ]
Create LSN             [0, 1385 ]
DataFileDescSlot ID    [ 1, 1 ]
```

//백업파일에 저장된 incremental backup정보

```
[BEGIN BACKUPFILE INFORMATION]

Begin Backup Time      [ 2012_11_06 23:18:43 ]
End Backup Time        [ 2012_11_06 23:18:44 ]
IBChunk Count          [ 0 ]
Backup Target          [ DATABASE ]
Backup Level           [ LEVEL0 ]
Backup Type            [ FULL ]
TableSpace ID          [ 1 ]
File ID                [ 0 ]
Backup Tag Name        [ MONDAY ]
Backup File Name       [ /backup_dir/TAG_MONDAY/SYS_TBS_MEM_DATA-0-0_TAG_MONDAY.ibak ]
```

```
[END BACKUPFILE INFORMATION]
```

```
[END CHECKPOINT IMAGE HEADER]
```

Dump complete.

출력 항목

다음의 표는 증분 백업 파일에 대해 dumpdb 유틸리티를 실행하여 출력되는 항목들에 한하여 설명한다.

필드 이름	설명
Binary DB Version	데이터 파일의 버전
Redo LSN	미디어 복구를 위한 Redo LSN. 로그 앵커의 Redo LSN값이 데이터 파일의 Redo LSN보다 크면, 이 항목에 출력되는 Redo LSN 부터 미디어 복구가 필요하다.
Create LSN	체크포인트 이미지 생성 시점의 LSN

필드 이름	설명
DataFileDescSlot ID	메모리 체크포인트 이미지와 묶여있는 ChangeTracking의 DataFileDescSlot ID

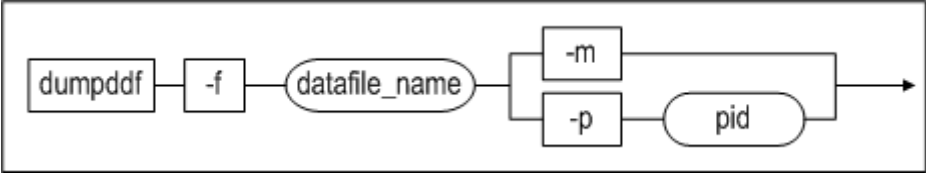
dumpddf

개요

dumpddf는 데이터 파일의 헤더 정보 또는 데이터 파일 내의 특정 페이지를 출력한다. 또는 증분 백업 파일에 대해 dumpddf를 수행하면 백업 파일의 헤더 정보와 백업 정보가 출력된다.

```
dumpddf {-f datafile_name} {-m | -p pid}
```

구문



파라미터

파라미터	설명
-f	정보를 얻고 싶은 데이터 파일의 이름 이 옵션은 반드시 입력해야 한다. 생략하면 dumpddf는 에러 메시지를 출력하고 종료한다.
-m	데이터 파일의 헤더 정보를 출력한다.
-p	정보를 얻고 싶은 데이터 파일내의 페이지 ID

설명

데이터 파일의 헤더의 정보 또는 데이터 파일 내의 특정 페이지를 텍스트 형태로 출력한다. 지정한 페이지가 테이블이나 인덱스의 페이지이면 논리적으로 구성하여 출력한다.

사용예

셸 프롬프트 상에서 다음과 같이 수행한다.

```
$ dumpddf -f datafile -m
$ dumpddf -f datafile -p page_id
```

출력 항목

다음은 dumpddf의 출력 예이다.

```
[BEGIN DATABASE FILE HEADER]
Binary DB Version      [ 5.4.1 ]
Redo LSN                [0, 734497 ]
Create LSN              [0, 1886 ]
MustRedo LSN           [0, 0 ]
```

출력 내용 중, 필드들은 다음과 같은 의미를 갖는다.

필드 이름	설명
Binary DB Version	데이터 파일의 버전
Redo LSN	미디어 복구를 위한 Redo LSN. 로그앵커의 Redo LSN값이 데이터 파일의 Redo LSN보다 크면, 이 Redo LSN 부터 미디어 복구가 필요하다.
Create LSN	데이터 파일 생성 시점의 LSN
MustRedo LSN	미디어 복구 시 이 Redo LSN까지 수행해야 한다.
DataFileDescSlot ID	Disk datafile과 묶여있는 ChangeTracking의 DataFileDescSlot ID

다음은 증분 백업 파일을 dumpddf로 출력한 예이다.


```
% dumpddf -m -f system001.dbf_TAG_MONDAY.ibak
```

```
-----  
Altibase Client Dump Disk Database File utility.  
Release Version 6.3.1.0.0  
Copyright 2000, ALTIBASE Corporation or its subsidiaries.  
All Rights Reserved.  
-----
```

```
[BEGIN DATABASE FILE HEADER]
```

```
Binary DB Version      [ 6.2.1 ]  
Redo LSN                [1, 5867599 ]  
Create LSN              [0, 1914 ]  
MustRedo LSN           [0, 0 ]  
DataFileDescSlot ID    [ 1, 2 ]
```

```
[BEGIN BACKUPFILE INFORMATION] -->백업파일에 저장된 incremental backup정보
```

```
Begin Backup Time      [ 2012_11_06 23:18:44 ]  
End Backup Time        [ 2012_11_06 23:18:46 ]  
IBChunk Count          [ 0 ]  
Backup Target          [ DATABASE ]  
Backup Level           [ LEVEL0 ]  
Backup Type            [ FULL ]  
TableSpace ID          [ 2 ]  
File ID                [ 0 ]  
Backup Tag Name        [ MONDAY ]  
Backup File Name       [ /backup_dir/TAG_MONDAY/system001.dbf_TAG_MONDAY.ibak ]
```

```
[END BACKUPFILE INFORMATION]
```

```
[END DATABASE FILE HEADER]
```

dumpla

개요

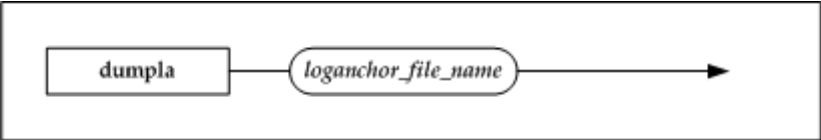
dumpla는 이진 형식으로 저장된 loganchor 파일의 내용을 문자 형식으로 보여준다.

loganchor는 Altibase의 물리적인 저장 정보(테이블스페이스, 데이터 파일)와 복구를 위해 필요한 정보들이 저장되는 파일이다. Altibase는 데이터베이스 생성 시(CREATE DATABASE 실행 시)에 이 파일을 생성하고 altibase.properties에 설정된 세 개의 LOGANCHOR_DIR 디렉토리에 loganchor#(#은 1, 2, 3)의 이름으로 저장, 관리한다. 세 개의 파일은 모두 같은 내용으로 파일이 손상될 경우를 대비해 복사본을 유지하는 것이다. 데이터베이스 생성 시에 현재 데이터베이스에 존재하는 테이블스페이스와

이에 속한 데이터 파일, 그리고 복구 관련 정보들이 저장된다. 데이터베이스 구동 시에 이 정보들을 이용하여 데이터베이스를 메모리에 적재하고 서비스를 준비한다.

```
dump1a <loganchor_file_name>
```

구문



설명

loganchor 파일의 내용을 텍스트 형식으로 보여준다.

사용예

셸 프롬프트 상에서 다음과 같이 수행한다.

```
$ dump1a loganchor0
```

출력 항목

dump1a는 loganchor의 내용을 다음과 같은 섹션으로 구분하여 출력한다.

[LOGANCHOR ATTRIBUTE SIZE]

현재 Loganchor에 어떤 데이터들이 얼마만큼의 공간을 사용하고 있는지를 나타낸다. 내용은 다음과 같다.

필드 이름	값(단위 바이트)	설명
Loganchor Static Area	0 ~ unsigned int 타입의 최대값	Loganchor에 고정적으로 들어가는 정보의 크기를 나타낸다. 주로 복구 시 필요한 정보들이다.
Tablespace Attribute	0 ~ unsigned int 타입의 최대값	저장된 Tablespace Attribute의 크기
Checkpoint Path Attribute	0 ~ unsigned int 타입의 최대값	저장된 Checkpoint Path의 크기
Checkpoint Image Attribute	0 ~ unsigned int 타입의 최대값	저장된 Checkpoint Image Attribute의 크기

필드 이름	값(단위 바이트)	설명
Disk Datafile Attribute	0 ~ unsigned int 타입의 최대값	저장된 Disk Datafile Attribute의 크기

[LOGANCHOR HEADER]

데이터베이스의 버전 및 체크포인트 Log Sequence Number (LSN) 등의 loganchor Header 정보를 나타낸다. LSN에 대한 자세한 설명은 “dumpIf의 출력”절을 참고하기 바란다.

필드 이름	값	설명
Binary DB Version	Major.minor.patch ex) 6.2.1	현재 Loganchor를 생성한 데이터베이스 실행 파일의 데이터베이스 버전이다.
Archivelog Mode	Archivelog No-Archivelog	데이터베이스가 아카이브 모드로 운영 중인지를 나타낸다.
Transaction Segment Entry Count		
Begin Checkpoint LSN	FileNo, Offset	Begin Checkpoint의 LSN을 나타낸다.
End Checkpoint LSN	FileNo, Offset	End Checkpoint의 LSN을 나타낸다.
Disk Redo LSN	FileNo, Offset	DRDB의 Redo 시작 위치를 나타낸다.
LSN for Recovery from Replication	FileNo, Offset	이중화를 이용한 복구 시 이 LSN부터 시작한다.

필드 이름	값	설명
Server Status	SERVER_SHUTDOWN SERVER_STARTED	서버의 상태를 기록한다. 서버 시작 시 이 값이 SERVER_STARTED로 바뀌고 종료 시 SERVER_SHUTDOWN으로 바뀐다. 서버 시작 시에 이 값이 SERVER_STARTED로 되어있다면 이전에 서버가 비정상 종료한 것이므로 재구동시 복구를 수행한다.
End LSN	FileNo, Offset	서버가 정상 종료한 후 재구동시 첫 번째 기록되는 로그의 LSN
ResetLog LSN	FileNo, Offset	불완전 복구시 설정된 Reset LSN
Last Created Logfile Num	0 ~ unsigned int 타입의 최대값	가장 최근에 생성된 로그 파일 번호
Delete Logfile(s) Range	삭제되는 첫 번째 ~ 마지막 파일 번호	가장 최근에 삭제된 로그 파일의 범위. 체크포인트 완료 후 더 이상 필요 없는 파일을 삭제하는데 이 때 지워지는 파일의 범위를 기록한다.
Update And Flush Count	0 ~ unsigned int 타입의 최대값	logAnchor 파일이 변경되고 플러시된 횟수
New Tablespace ID	0 ~ unsigned int 타입의 최대값	새로운 테이블스페이스 식별자. 테이블스페이스가 생성될 경우 이 값을 자신의 테이블스페이스 식별자로 사용하고 이 값을 증가시킨다.

[TABLESPACE ATTRIBUTE]

메모리 테이블스페이스 정보를 나타낸다. 내용은 다음과 같다.

필드 이름	값	설명
Tablespace ID	0 ~ unsigned int 타입의 최대값	테이블스페이스 식별자
Tablespace Name	문자열 ex) SYS_TBS_MEM_DIC	테이블스페이스 이름

필드 이름	값	설명
New Database File ID	0 ~ unsigned int 타입의 최대값	테이블스페이스에서 새로 파일이 추가될 때 파일이 가지게 될 파일의 식별자
Extent Management	FREE EXTENT BITMAP TABLESPACE	디스크 테이블스페이스의 extent가 어떤 방식으로 관리되는지를 나타낸다. 현재는 FREE EXTENT BITMAP TABLESPACE만 지원한다. FREE EXTENT BITMAP TABLESPACE를 enable하면, 디스크 테이블스페이스의 free extents를 관리하는데 bitmaps을 사용한다.
Tablespace Status	아래 TABLESPACE ATTRIBUTE 설명 참조	테이블스페이스의 현재 상태를 나타낸다.
Tablespace Type	0 ~ 8 아래 TABLESPACE ATTRIBUTE 설명 참조	테이블스페이스의 타입
Checkpoint Path Count	Checkpoint Path 개수	체크포인트 이미지 파일들의 경로. 메모리 테이블스페이스에만 적용된다.
Autoextend Mode	AutoExtend Non-AutoExtend	테이블스페이스가 자동으로 크기가 확장되는지 여부를 가리킨다. 메모리 테이블스페이스에만 적용된다.
Shared Memory Key	0 ~ unsigned int 타입의 최대값	데이터베이스가 공유 메모리 버전이면 이 때 공유 메모리에 데이터베이스가 사용한 공유 메모리의 키
Stable Checkpoint Image Num	0 1	체크포인트가 완료된 체크포인트 이미지의 파일 번호 메모리 테이블스페이스에만 적용된다.
Init Size	0 ~ unsigned int 타입의 최대값	초기 크기(Mbyte)
Next Size	0 ~ unsigned int 타입의 최대값	자동 확장되는 증가분의 크기(Mbyte)
Maximum Size	0 ~ unsigned int 타입의 최대값	최대 크기

필드 이름	값	설명
Split File Size	0 ~ unsigned int 타입의 최대값	메모리 테이블스페이스는 파일을 이 크기 단위로 생성한다. 예를 들어 테이블스페이스가 1G이고, Split file size가 100MB이면 10개의 파일이 생성된다.

[TABLESPACE ATTRIBUTE] 내용 중 Tablespace Status의 의미는 다음과 같다.

값	설명
OFFLINE	오프라인 상태
ONLINE	온라인 상태
INCONSISTENT	비일관적인 상태
CREATING	생성 중
DROPPING	삭제를 수행한 트랜잭션이 아직 커밋하지 않아 삭제 중인 상태
DROP_PENDING	삭제를 수행한 트랜잭션이 커밋 이후 Pending 연산 중인 상태
DROPPED	삭제된 상태
DISCARDED	Discard된 상태
BACKUP	백업 중
SWITCHING_TO_OFFLINE	OFFLINE으로 바뀌고 있는 중
SWITCHING_TO_ONLINE	ONLINE으로 바뀌고 있는 중

[TABLESPACE ATTRIBUTE] 내용 중 Tablespace Type의 의미는 다음과 같다.

값	설명
0	MEMORY SYSTEM DICTIONARY
1	MEMORY SYSETM DATA
2	MEMORY USER DATA
3	DISK SYSTEM DATA
4	DISK USER DATA
5	DISK SYSTEM TEMP

값	설명
6	DISK USER TEMP
7	DISK SYSTEM UNDO
8	VOLATILE USER DATA

[MEMORY CHECKPOINT PATH ATTRIBUTE]

메모리 테이블스페이스의 체크포인트 저장 경로를 나타낸다. 내용은 다음과 같다.

필드 이름	값	설명
Tablespace ID	0 ~ unsigned int 타입의 최대값	테이블스페이스 식별자
Checkpoint Path	문자열	체크포인트 이미지 파일의 경로

[MEMORY CHECKPOINT IMAGE ATTRIBUTE]

메모리 테이블스페이스의 체크포인트 이미지 정보를 나타낸다. 내용은 다음과 같다.

필드 이름	값	설명
Tablespace ID	0 ~ unsigned int 타입의 최대값	테이블스페이스 식별자
File Number	0 ~ unsigned int 타입의 최대값	파일 번호
Create LSN	< FileNo, Offset>	데이터 파일 생성 시점의 LSN
Create On Disk (PingPong 0)	Created None	0번 체크포인트 이미지 파일 생성 유무
Create On Disk (PingPong 1)	Created None	1번 체크포인트 이미지 파일 생성 유무
ChangeTracking DataFileDescSlot ID	0 ~ unsigned int 타입의 최대값	Memory checkpoint image와 묶여있는 ChangeTracking의 DataFileDescSlot ID

[DISK DATABASE FILE ATTRIBUTE]

디스크 테이블스페이스의 데이터 파일 저장 경로를 나타낸다. 내용은 다음과 같다.

필드 이름	값	설명
-------	---	----

필드 이름	값	설명
Tablespace ID	0 ~ unsigned int 타입의 최대값	테이블스페이스 식별자
Database File ID	0 ~ unsigned int 타입의 최대값	데이터 파일 식별자
Database File Path	문자열	데이터 파일의 경로
Create LSN	< FileNo, Offset>	데이터 파일이 생성된 시점의 시스템 로그의 마지막 LSN
Database File Status	아래 DISK_DATABASE_FILE_ATTRIBUTE 설명 참조	파일의 상태
Autoextend Mode	AutoExtend Non-AutoExtend	자동 확장 설정 여부
Create Mode	0 1	0: 재사용 1: 새로 생성한 파일
Initialize Size	0 ~ unsigned int 타입의 최대값	초기 크기(MByte)
Current Size	0 ~ unsigned int 타입의 최대값	현재 크기(MByte)
Next Size	0 ~ unsigned int 타입의 최대값	확장 단위 크기(MByte)
Maximum Size	0 ~ unsigned int 타입의 최대값	최대 크기(MByte)
ChangeTracking DataFileDescSlot ID	0 ~ unsigned int 타입의 최대값	Disk datafile과 묶여있는 ChangeTracking의 DataFileDescSlot ID

[DISK DATABASE FILE ATTRIBUTE] 내용 중 Database File Status의 의미는 다음과 같다.

값	설명
OFFLINE	오프라인 상태
ONLINE	온라인 상태
CREATING	생성 중
BACKUP_BEGIN	백업을 시작한 상태
BACKUP_END	백업을 종료 중인 상태
DROPPING	삭제 중인 상태

값	설명
RESIZING	크기를 바꾸고 있는 상태
DROPPED	삭제된 상태

다음은 dumpla의 출력 결과 중 일부분을 예로 보인 것이다.

```
[ DISK DATABASE FILE ATTRIBUTE ]
Tablespace ID           [ 2 ]
Database File ID        [ 0 ]
Database File Path C:\altibase_home\dbs\system001.dbf]
Create LSN               [0, 4443 ]
Database File Status    [ ONLINE ]
Autoextend Mode         [ Non-Autoextend ]
Create Mode              [ 0 ]
Initialize Size          [10 MBytes(1280 Pages)]
Current Size             [10 MBytes(1280 Pages)]
Next Size                [0 MBytes(0 Pages)]
Maximum Size             [0 MBytes(0 Pages)]
```

[Change Tracking ATTRIBUTE]

changeTracking 파일에 대한 정보를 나타낸다. 내용은 다음과 같다.

필드 이름	값	설명
Last Flush LSN	FileNo, Offset	메모리상의 변경된 데이터가 파일로 기록된 시점의 LSN
Change Tracking Manager State	문자열 ex) CHANGE TRACKING MGR ENABLED	페이지 변경 추적 기능의 상태
Change Tracking File Name	문자열	변경 추적 파일의 경로

[Backup Info ATTRIBUTE]

backupinfo 파일에 대한 정보를 나타낸다. 내용은 다음과 같다.

필드 이름	값	설명
Delete Archivelog File Range	FileNo, Offset	삭제해도 완전 복구가 가능한 아카이브 로그 파일의 번호
Last Backup LSN	FileNo, Offset	최근에 수행된 백업 시점의 LSN

필드 이름	값	설명
Before Backup LSN	FileNo, Offset	최근에 수행된 백업이전에 수행된 LSN
Backup Info Manager State	문자열 예) BACKUP INFO MGR INITIALIZED	백업 정보 파일 관리자의 상태
Backup Directory Path	문자열 예) /backup_dir/	백업 경로
Backup Info File Name	문자열	백업 정보 파일 이름

dumplf

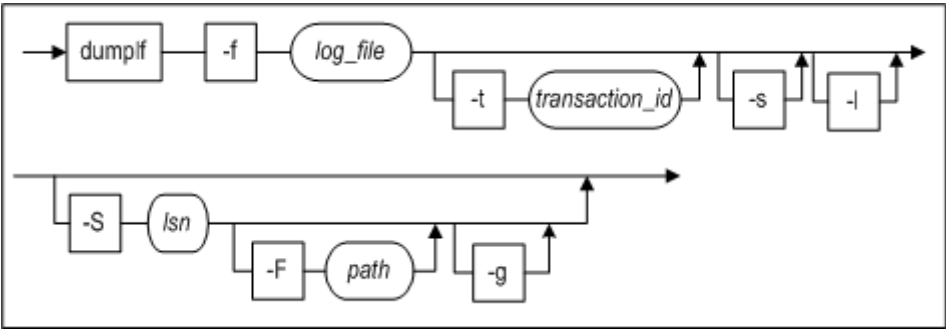
개요

Altibase는 복구를 위해서 트랜잭션이 변경 연산(INSERT, DELETE, UPDATE 등)을 수행할 때, 데이터베이스 영역(DB File)외의 별도 파일에 로그를 남긴다. 이 로그는 IO 최소화를 위해서 이진(binary) 형식으로 기록되어 있다. 이 로그가 기록되는 로그 파일은 altibase.properties의 LOG_DIR 프로퍼티에 설정되는 디렉토리에 logfile#(#은 로그파일 번호로 계속 증가한다) 형태로 저장된다.

dumplf는 이 로그 파일의 내용을 텍스트 형식으로 변환해서 보여주는 유틸리티이다. 이 로그를 통해서 데이터베이스에 수행된 연산의 종류 및 변경 연산을 수행하는 트랜잭션의 빈도를 측정할 수 있다.

```
dumplf {-f log_file_name} [-t transaction_id][-s] [-l][-S lsn [-F path] [-g]]
```

구문



파라미터

파라미터	설명
------	----

파라미터	설명
-f	출력할 로그 파일의 이름
-t	출력할 로그에 해당하는 트랜잭션 ID
-s	로그 파일의 헤더만 출력하고자 할 때 명시
-l	로그의 유형 (LT 필드) 또는 서브로그유형 (OPTYPE 과 UTYPE 필드)과 관련된 정보만 출력
-S	MMDB에서 INSERT, UPDATE, DELETE, COMMIT, ROLLBACK의 로그 개수를 출력한다. 특정 lsn을 입력하면 해당 lsn 이후의 로그만, '0,0'을 입력하면 전체 로그를 보여준다.
-F	로그를 분석할 대상 경로를 지정할 수 있으며, 생략할 경우 \$ALTIBASE_HOME/logs의 경로를 분석한다.
-g	테이블의 오브젝트 ID별 통계가 전체 통계 정보와 함께 출력된다. 생략하면 전체 통계 정보만 출력된다.

설명

로그 파일의 내용을 텍스트 형식으로 변환해서 보여준다.

사용예

셸 프롬프트 상에서 다음과 같이 수행한다.

```
$ dump1f -f logfile0
```

출력 항목

다음은 dump1f의 출력 예이다.

```
LSN=<0,820>, COMP:N, MAGIC:820, TID: 6400,BE: N, REP: Y, ISVP: N, ISVP_DEPTH: 0 PLSN=<0,739>, LT: SMR_LT_MEMTRANS
```

로그 파일의 내용 중, 필드들은 다음과 같은 의미를 갖는다.

필드 이름	값	설명
-------	---	----

필드 이름	값	설명
LSN	형식: (FileNo, Offset) 각 값의 범위: 0 - unsigned int 타입의 최대값 ,	로그의 sequence number로 로그의 물리적인 위치를 표시한다. LSN은 파일 번호, 오프셋으로 구성된다.
COMP	Y N	Y: 로그가 압축(compress)되었음 N: 압축되지 않음
MAGIC	0 - unsigned short 타입의 최대값	로그 레코드가 유효하다는 것을 나타내기 위해서 로그 LSN의 로그 파일 번호와 오프셋으로 이 값을 만들고 Redo, Undo시 로그를 읽었을 때 이 값이 현재 로그의 LSN과 동일한지 검사한다.
TID	0 - unsigned int 타입의 최대값	트랜잭션 식별자
BE	Y N	Y : Begin Transaction Log
REP	Y N	Y: 이중화가 걸려있는 경우 Sender가 Standby로 보내거나 참조하는 로그이다. N: Sender는 이 로그를 무시한다
ISVP	Y N	Y: 묵시적 저장점 (Implicit Savepoint) 로그. 문장(Statement) 시작 후 처음으로 찍힌 로그이다. 트랜잭션이 문장을 수행했을 때 오류가 발생하면 이 로그까지 부분 철회 (Partial Rollback)을 한다.
ISVP_DEPTH	0 - 255	어떤 SQL 구문이 다른 구문 내에 중첩되었을 때 묵시적 저장점의 중첩 깊이
PLSN	형식: (FileNo, Offset) 각 값의 범위: 0 - unsigned int 타입의 최대값	Previous Log LSN으로 이 값을 이용하여 같은 트랜잭션이 기록한 로그들끼리 체인 (Chain)으로 연결되어 있다.
LT	문자열	로그의 유형을 나타낸다. 자세한 내용은 아래 로그 유형에 대한 표를 참고한다.
SZ	0 - unsigned int 타입의 최대값	로그 크기 (단위: bytes)
RdSz	0 - unsigned int 타입의 최대값	로그 레코드의 크기 (단위: bytes)

필드 이름	값	설명
DMIOff	0 - unsigned int 타입의 최대값	논리적 로그의 위치. 트랜잭션 언두 또는 이중화에서 사용한다.
TableOID	0 - unsigned int 타입의 최대값	테이블의 오브젝트 ID
OID	0 - unsigned int 타입의 최대값	테이블을 제외한 모든 오브젝트의 ID. 레코드 오브젝트도 포함한다.
ContType	0, 1	이중화를 위해 사용하는 내부 값
OPTYPE	LogTypeName<LogTypeNumber>	Nested Top Action (NTA) 로그의 작업 유형
AFTER	SZ: <size>, Value: <value>	로그 레코드의 변경 후 이미지 (after image)
BEFORE	SZ: <size>, Value: <value>	로그 레코드의 변경 전 이미지 (before image)
UTYPE	LogTypeName<LogTypeNumber>	UPDATE 로그의 작업 유형
UPOS	형식: (SPACEID:<SpaceID>, PID:<PageID>, OFFSET:<Offset> => OID:<OID>)	변경된 객체의 주소. 또한 변경 작업 동안 발생한 정보도 포함한다.
SPACEID	0 - unsigned short 타입의 최대값	변경된 객체를 포함하고 있는 테이블스페이스의 ID
PID	0 - unsigned int 타입의 최대값	변경된 객체를 포함하고 있는 페이지의 ID
Offset	0 - unsigned int 타입의 최대값	변경된 객체를 포함하고 있는 페이지 내에서의 offset
FLISlot PrevPID NextPID	형식: (<BeforePID> => <AfterPID>)	메모리 테이블스페이스를 관리하기 위해 사용하는 내부 값
ESLSN	형식: (FileNo, Offset) 각 값의 범위: 0 - unsigned int 타입의 최대값	복구 수행 시 시작할 LSN
Lob Locator	0 - unsigned long 타입의 최대값	이중화에서 LOB타입과 관련해서 내부적으로 사용하는 값

dumpLf의 결과 중 LT (Log Type: 로그 유형)의 의미는 다음과 같다.

값	설명
SMR_LT_DUMMY	Dummy 로그
SMR_LT_CHKPT_BEGIN	Checkpoint Begin 로그
SMR_LT_DIRTY_PAGE	Dirty Page 로그
SMR_LT_CHKPT_END	Checkpoint End 로그
SMR_LT_MEMTRANS_COMMIT	Memory Transaction Commit 로그
SMR_LT_MEMTRANS_ABORT	Memory Transaction Abort 로그
SMR_LT_DSKTRANS_COMMIT	Disk Transaction Commit 로그
SMR_LT_DSKTRANS_ABORT	Disk Transaction Abort 로그
SMR_LT_SAVEPOINT_SET	Savepoint Set 로그
SMR_LT_SAVEPOINT_ABORT	Savepoint Abort Begin 로그
SMR_LT_XA_PREPARE	XA Prepare 로그
SMR_LT_TRANS_PREABORT	Abort Begin 로그
SMR_LT_DDL	DDL (Data Definition Language) 로그
SMR_LT_XA_SEGS	XA Prepare Transaction Segment Information
SMR_LT_LOB_FOR_REPL	이중화를 위한 LOB 로그
SMR_LT_UPDATE	MMDB(Main Memory Database) Update 로그
SMR_LT_NTA	MMDB NTA(Nested Top Action) 로그
SMR_LT_COMPENSATION	Compensation 로그
SMR_LT_DUMMY_COMPENSATION	Dummy Compensation 로그
SMR_LT_FILE_BEGIN	File Begin 로그
SMR_LT_FILE_TBS_UPDATE	Tablespace Update 로그
SMR_LT_FILE_END	File End 로그
SMR_DLT_READONLY	DRDB(Disk Resident Database) Redo Only 로그
SMR_DLT_UNDOABLE	DRDB Undo 로그

값	설명
SMR_DLT_NTA	DRDB NTA 로그
SMR_DLT_COMPENSATION	DRDB Compensation 로그
SMR_DLT_REF_NTA	DRDB Reference NTA 로그
SMR_LT_TABLE_META	이중화를 위한 테이블 메타 로그

OPTYPE 과 UTYPE 에 가능한 로그 타입 이름

값	설명
SMR_OP_SMM_PERS_LIST_ALLOC SMR_OP_SMC_FIXED_SLOT_ALLOC SMR_OP_SMC_VAR_SLOT_ALLOC SMR_OP_SMC_FIXED_SLOT_FREE SMR_OP_SMC_VAR_SLOT_FREE	MMDB의 페이지와 슬롯에 관련된 로그
SMR_OP_CREATE_TABLE SMR_OP_CREATE_INDEX SMR_OP_DROP_INDEX SMR_OP_ALTER_TABLE SMR_OP_SMM_CREATE_TBS SMR_OP_INSTANT_AGING_AT_ALTER_TABLE SMR_OP_SMC_TABLEHEADER_ALLOC	MMDB에서 DDL 구문 수행과 관련된 로그
SMR_MEM_LOB_CURSOR_OPEN SMR_DISK_LOB_CURSOR_OPEN SMR_LOB_CURSOR_CLOSE SMR_PREPARE4WRITE SMR_FINISH2WRITE	MMDB에서 LOB 데이터 제어와 관련된 로그

값	설명
SDR_OP_SDP_CREATE_TABLE_SEGMENT SDR_OP_SDP_CREATE_LOB_SEGMENT SDR_OP_SDP_CREATE_INDEX_SEGMENT SDR_OP_SDP_ADD_LOB_PAGE_TO_AGINGLIST SDR_OP_SDC_ALLOC_UNDO_PAGE SDR_OP_SDPTB_ALLOCATE_AN_EXTENT_FROM_TBS SDR_OP_SDPTB_ALLOCATE_AN_EXTDIR_FROM_LIST SDR_OP_SDPTB_RESIZE_GG SDR_OP_SDPST_ALLOC_PAGE SDR_OP_SDPSF_ALLOC_PAGE SCT_UPDATE_MRDB_CREATE_TBS SCT_UPDATE_MRDB_CREATE_CIMAGE_FILE SCT_UPDATE_MRDB_DROP_TBS SCT_UPDATE_MRDB_ALTER_AUTOEXTEND SCT_UPDATE_MRDB_ALTER_TBS_ONLINE SCT_UPDATE_MRDB_ALTER_TBS_OFFLINE SCT_UPDATE_DRDB_CREATE_TBS SCT_UPDATE_DRDB_DROP_TBS SCT_UPDATE_DRDB_ALTER_TBS_ONLINE SCT_UPDATE_DRDB_ALTER_TBS_OFFLINE SCT_UPDATE_DRDB_CREATE_DBF SCT_UPDATE_DRDB_DROP_DBF SCT_UPDATE_DRDB_EXTEND_DBF SCT_UPDATE_DRDB_SHRINK_DBF SCT_UPDATE_DRDB_AUTOEXTEND_DBF SCT_UPDATE_DRDB_ALTER_DBF_ONLINE SCT_UPDATE_DRDB_ALTER_DBF_OFFLINE SCT_UPDATE_VRDB_CREATE_TBS SCT_UPDATE_VRDB_DROP_TBS SCT_UPDATE_VRDB_ALTER_AUTOEXTEND SCT_UPDATE_COMMON_ALTER_ATTR_FLAG	테이블스페이스와 세그먼트에 관련된 로그

값	설명
SDR_OP_SDPST_UPDATE_WMINFO_4DPATH SDR_OP_SDPST_UPDATE_MFNL_4DPATH SDR_OP_SDPST_UPDATE_BMP_4DPATH SDR_OP_SDPSF_ADD_PIDLIST_PVTFREEPIDLIST_4DPATH SDR_OP_SDPSF_MERGE_SEG_4DPATH SDR_OP_SDPSF_UPDATE_HWMINFO_4DPATH SDR_OP_SDP_DPATH_ADD_SEGINFOSET	DRDB에서 Direct Page Insert를 위한 페이지 관리와 관련된 로그
SDR_OP_SDN_INSERT_KEY_WITH_NTA SDR_OP_SDN_DELETE_KEY_WITH_NTA	DRDB에서 B-tree 인덱스를 위한 NTA 로그
SDR_OP_STNDR_INSERT_KEY_WITH_NTA SDR_OP_STNDR_DELETE_KEY_WITH_NTA	DRDB에서 R-tree 인덱스를 위한 NTA 로그
SDR_SDP_1BYTE SDR_SDP_2BYTE SDR_SDP_4BYTE SDR_SDP_8BYTE SDR_SDP_BINARY	물리적 DRDB 로그
SDR_SDP_PAGE_CONSISTENT SDR_SDP_INIT_PHYSICAL_PAGE SDR_SDP_INIT_LOGICAL_HDR SDR_SDP_INIT_SLOT_DIRECTORY SDR_SDP_FREE_SLOT SDR_SDP_FREE_SLOT_FOR_SID SDR_SDP_RESTORE_FREESPACE_CREDIT SDR_SDP_RESET_PAGE SDR_SDP_WRITE_PAGEIMG SDR_SDP_WRITE_DPATH_INS_PAGE	DRDB의 페이지와 슬롯에 관련된 로그

값	설명
SDR_SDPST_INIT_SEGHDR SDR_SDPST_INIT_BMP SDR_SDPST_INIT_LFBMP SDR_SDPST_INIT_EXTDIR SDR_SDPST_ADD_RANGESLOT SDR_SDPST_ADD_SLOTS SDR_SDPST_ADD_EXTDESC SDR_SDPST_ADD_EXT_TO_SEGHDR SDR_SDPST_UPDATE_WM SDR_SDPST_UPDATE_MFNL SDR_SDPST_UPDATE_PBS SDR_SDPST_UPDATE_LFBMP_4DPATH SDR_SDPSC_INIT_SEGHDR SDR_SDPSC_INIT_EXTDIR SDR_SDPSC_ADD_EXTDESC_TO_EXTDIR SDR_SDPTB_INIT_LGHDR_PAGE SDR_SDPTB_ALLOC_IN_LG SDR_SDPTB_FREE_IN_LG	DRDB를 위한 세그먼트와 테이블스페이스에 관련된 로그
SDR_SDC_INSERT_ROW_PIECE SDR_SDC_INSERT_ROW_PIECE_FOR_UPDATE SDR_SDC_INSERT_ROW_PIECE_FOR_DELETEUNDO SDR_SDC_UPDATE_ROW_PIECE SDR_SDC_OVERWRITE_ROW_PIECE SDR_SDC_CHANGE_ROW_PIECE_LINK SDR_SDC_DELETE_FIRST_COLUMN_PIECE SDR_SDC_ADD_FIRST_COLUMN_PIECE SDR_SDC_DELETE_ROW_PIECE_FOR_UPDATE SDR_SDC_DELETE_ROW_PIECE SDR_SDC_LOCK_ROW	DRDB의 테이블 레코드 관리에 관련된 로그
SDR_SDC_UPDATE_LOBDESC SDR_SDC_UPDATE_LOBDESC_KEY SDR_SDC_LOB_WRITE_PIECE SDR_SDC_LOB_WRITE_PIECE4DML SDR_SDC_INIT_LOBPAGE SDR_SDC_LOB_PAGE_TO_AGING_LIST	DRDB에서 LOB 타입 사용에 관련된 로그

값	설명
SDR_SDC_PK_LOG	DRDB에서 이중화를 위한 프라이머리 키 사용에 관련된 로그
SDR_SDC_INIT_CTL SDR_SDC_EXTEND_CTL SDR_SDC_BIND_CTS SDR_SDC_UNBIND_CTS SDR_SDC_BIND_ROW SDR_SDC_UNBIND_ROW SDR_SDC_ROW_TIMESTAMPING SDR_SDC_DATA_SELFAGING	DRDB에서 레코드를 위한 MVCC 관련된 로그
SDR_SDC_BIND_TSS SDR_SDC_UNBIND_TSS SDR_SDC_SET_INITSCN_TO_TSS SDR_SDC_INIT_TSS_PAGE SDR_SDC_INIT_UNDO_PAGE SDR_SDC_INSERT_UNDO_REC	DRDB에서 Trasaction Status Slots (TSS)과 언두 레코드에 관련된 로그
SDR_SDN_INSERT_INDEX_KEY SDR_SDN_FREE_INDEX_KEY SDR_SDN_INSERT_UNIQUE_KEY SDR_SDN_INSERT_DUP_KEY SDR_SDN_DELETE_KEY_WITH_NTA SDR_SDN_FREE_KEYS SDR_SDN_COMPACT_INDEX_PAGE	DRDB에서 B-tree 인덱스 관련된 로그
SDR_SDN_MAKE_CHAINED_KEYS SDR_SDN_MAKE_UNCHAINED_KEYS SDR_SDN_KEY_STAMPING SDR_SDN_INIT_CTL SDR_SDN_EXTEND_CTL SDR_SDN_FREE_CTS	DRDB에서 B-tree 인덱스 키를 위한 MVCC에 관련된 로그
SDR_STNDR_MAKE_CHAINED_KEYS SDR_STNDR_MAKE_UNCHAINED_KEYS SDR_STNDR_KEY_STAMPING	DRDB에서 R-tree 인덱스 관련된 로그
SMR_PHYSICAL	MMDB에서 물리적 로그

값	설명
SMR_SMM_MEMBASE_SET_SYSTEM_SCN SMR_SMM_MEMBASE_ALLOC_PERS_LIST SMR_SMM_MEMBASE_ALLOC_EXPAND_CHUNK SMR_SMM_PERS_UPDATE_LINK SMR_SMM_PERS_UPDATE_NEXT_FREE_PAGE_LINK SMR_SMM_MEMBASE_INFO	MMDB에서 기본 정보와 관련된 로그
SMR_SMC_TABLEHEADER_INIT SMR_SMC_TABLEHEADER_UPDATE_INDEX SMR_SMC_TABLEHEADER_UPDATE_COLUMNS SMR_SMC_TABLEHEADER_UPDATE_INFO SMR_SMC_TABLEHEADER_SET_NULLROW SMR_SMC_TABLEHEADER_UPDATE_ALL SMR_SMC_TABLEHEADER_UPDATE_ALLOCINFO SMR_SMC_TABLEHEADER_UPDATE_FLAG SMR_SMC_TABLEHEADER_SET_SEQUENCE SMR_SMC_TABLEHEADER_UPDATE_TABLE_COLUMN_COUNT SMR_SMC_TABLEHEADER_UPDATE_TABLE_SEGMENT SMR_SMC_TABLEHEADER_UPDATE_FLAG_FOR_MEDIA_RECV SMR_SMC_TABLEHEADER_SET_SEGSTOATTR SMR_SMC_TABLEHEADER_SET_INSERTLIMIT SMR_SMC_INDEX_SET_FLAG SMR_SMC_INDEX_SET_SEGATTR SMR_SMC_INDEX_SET_SEGSTOATTR SMR_SMC_INDEX_SET_DROP_FLAG	MMDB에서 테이블 헤더와 인덱스 헤더에 관련된 로그

값	설명
SMR_SMC_PERS_INIT_FIXED_PAGE SMR_SMC_PERS_INIT_FIXED_ROW SMR_SMC_PERS_UPDATE_FIXED_ROW SMR_SMC_PERS_UPDATE_FIXED_ROW_NEXT_FREE SMR_SMC_PERS_UPDATE_FIXED_ROW_NEXT_VERSION SMR_SMC_PERS_SET_FIX_ROW_DROP_FLAG SMR_SMC_PERS_SET_FIX_ROW_DELETE_BIT SMR_SMC_PERS_INIT_VAR_PAGE SMR_SMC_PERS_UPDATE_VAR_ROW_HEAD SMR_SMC_PERS_UPDATE_VAR_ROW SMR_SMC_PERS_SET_VAR_ROW_FLAG SMR_SMC_PERS_SET_VAR_ROW_NXT_OID SMR_SMC_PERS_WRITE_LOB_PIECE SMR_SMC_PERS_INSERT_ROW SMR_SMC_PERS_UPDATE_INPLACE_ROW SMR_SMC_PERS_UPDATE_VERSION_ROW SMR_SMC_PERS_DELETE_VERSION_ROW	MMDB에서 테이블 레코드와 관련된 로그

MVCC와 관련된 자세한 설명은 *Administrator's Manual* 을 참고하기 바란다.

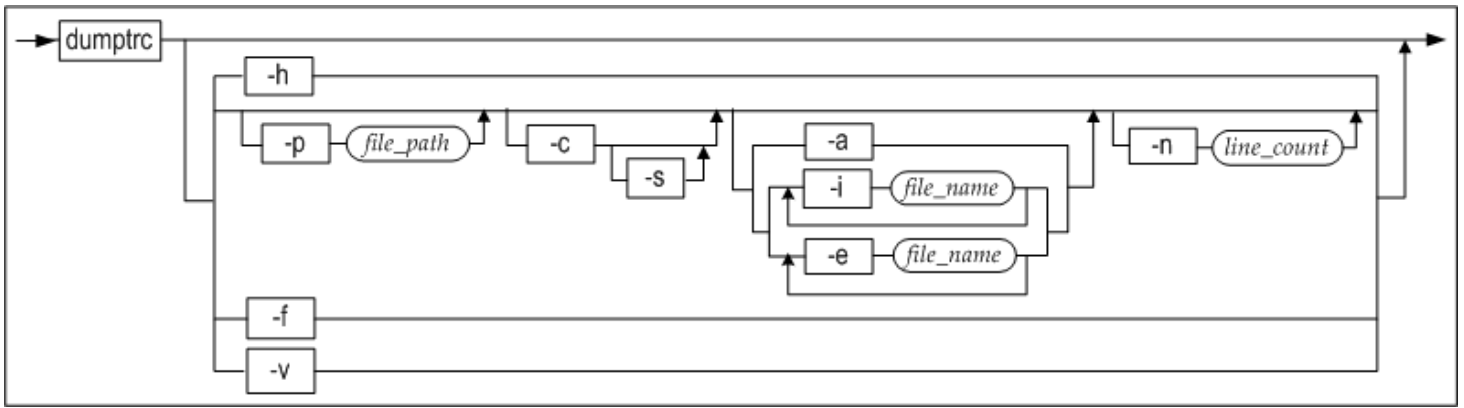
dumptrc

개요

Altibase 서버가 비정상 종료될 때 \$ALTIBASE_HOME/trc 디렉토리에 기록되는 트레이스 로그 파일을 사용자가 식별할 수 있는 형태로 변환하여 출력한다.

```
dumptrc [-h |[-p file_path][-c [-s]]
[-a|-i file_name [-i file_name]..|-e file_name [-e file_name]..] [-n file_count] [x] |-f |-v]
```

구문



파라미터

파라미터	설명
-h	도움말을 출력한다. 파라미터를 생략하거나 다른 파라미터와 중복하여 사용할 경우 도움말이 우선한다.
-p	트레이스 로그 파일을 가져오는 경로를 지정한다. 경로를 지정하지 않으면 기본적으로 \$ALTIBASE_HOME/trc 디렉토리의 로그 파일을 가져온다.
-c	altibase_error.log에 기록된 ALTIBASE 프로세스 콜 스택을 사용자가 식별할 수 있는 함수명으로 변환하여 출력한다. 콜 스택의 주소를 함수명으로 변경하지 않으려면 -s 옵션를 사용한다. 파라미터 -a, -i, -e, -n과 함께 사용하지 않으면 콜 스택만 기록된다.
-s	트레이스 로그에 기록된 콜 스택만을 출력하고 함수명으로 변경하지 않는다.
-a	트레이스 로그 파일의 전부를 정렬하여 출력한다.
-i	지정하는 트레이스 로그 파일을 출력한다. 복수의 로그 파일이 출력 가능하다. '-e' 파라미터와 함께 사용할 수 없다.
-e	지정한 트레이스 로그 파일을 제외한 모든 파일을 출력한다. 반복하여 복수의 로그 파일을 제거할 수 있으며, '-i' 파라미터와 함께 사용할 수 없다.
-n	한 번에 출력할 로그의 개수를 지정한다. 1~127개의 로그를 출력할 수 있으며, 지정하지 않으면 10개의 로그가 출력된다.
-f	트레이스 로그 파일이 기록될 때마다 추가되는 로그 메시지를 출력한다.
-v	dumptrc의 버전을 출력한다.

설명

프로세스 콜 스택에는 Altibase 종료 당시 수행되던 Altibase 내부 모듈에 대한 정보가 기록된다. dumptrc를 이용하여 가져올 수 있는 트레이스 로그 파일은

\$ALTIBASE_HOME/trc 디렉터리에 기록되는 아래의 파일들을 사용할 수 있다.

- ERROR : altibase_error.log
- SERVER : altibase_boot.log
- SM : altibase_sm.log
- RP : altibase_rp.log
- QP : altibase_qp.log
- DK : altibase_dk.log
- DR : altibase_dr.log
- XA : altibase_xa.log
- MM : altibase_mm.log
- RP_CONFLICT : altibase_rp_conflict.log
- DUMP : altibase_dump.log
- TRC : altibase_trace.log
- SNMP : altibase_snmp.log
- CM : altibase_cm.log
- MISC : altibase_misc.log
- SD : altibase_sd.log

만약 \$ALTIBASE_HOME/trc가 아닌 다른 경로에 있는 트레이스 로그 파일을 출력할 때에는 '-p'를 사용하여 경로를 변경할 수 있다.

Altibase 서버가 비정상 종료할 경우, 프로세스 콜 스택을 dumptrc로 변환하여 Altibase 기술서비스 팀에 송부하면, 보다 빨리 문제를 해결할 수 있을 것이다.

주의 사항

Altibase의 실행 파일의 버전과 dumptrc의 버전이 동일해야 정확한 콜 스택 정보를 확인할 수 있다.버전이 틀릴 경우는 잘못된 값이 나올 수 있기 때문에 기본적으로 경고메시지를 보여주고 콜 스택을 출력하지 않는다. 버전이 틀릴때도 강제로 콜스택을 출력하기 위해서는 -x옵션을 사용하면 된다.

사용예

altibase_error.log와 altibase_boot.log를 묶어서 출력한다.

```

$ dumptrc -i server -i error
[2015/10/21 17:29:42 55C][PID:32702][Thread-2]
==> Initialize Disaster Recovery Manager
[2015/10/21 17:29:42 55D][PID:32702][Thread-2]
... [SUCCESS]
[2015/10/21 17:29:42 55E][PID:32702][Thread-2]
==> Initialize MMX Service
[2015/10/21 17:29:42 55F][PID:32702][Thread-2]
... [SUCCESS]
[2015/10/21 17:29:42 560][PID:32702][Thread-2]
==> Initialize Audit Service
[2015/10/21 17:29:42 561][PID:32702][Thread-2]
... [SUCCESS]
[2015/10/21 17:29:42 562][PID:32702][Thread-2]
==> Initialize Job Manager
[2015/10/21 17:29:42 563][PID:32702][Thread-2]
... [SUCCESS]
[2015/10/21 17:29:42 564][PID:32702][Thread-2]
--- STARTUP Process SUCCESS ---
[2015/10/21 17:30:51 57B] Dump of Stack
SIGNAL INFORMATION =====
Signal 6(SIGABRT) caught.
    Sent by process : 2331
    Sent by user    : 1000
BEGIN-DUMP =====
===== SERVER =====
ALTIBASE hdb
    Product version : 6.7.1.0.0
    CPU              : X86
    Operating System : x86_64-unknown-linux-gnu
    Process ID       : 32698
    Thread No        : 0
END-DUMP =====
BEGIN-STACK [CRASH] =====
Caller[0] 00000000011E0EBF
Caller[1] 000000000426CFE
Caller[2] 00007FD306FD4CB0
Caller[3] 00007FD30609F763
Caller[4] 00000000042F66F
Caller[5] 0000000004213CE
Caller[6] 00000000041D1E0
Caller[7] 00007FD305FD376D
Caller[8] 0000000004206BD
END-STACK =====
10 logs printed.

```

error.log만 제외하고 출력한다.


```
$ dumptrc -e error
[2015/10/21 17:29:48 571][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : DROP TABLE T1]
[2015/10/21 17:29:48 572][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
[2015/10/21 17:29:49 573][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : CREATE TABLE T1 ( I1 INTEGER )]
[2015/10/21 17:29:49 574][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
[2015/10/21 17:29:49 575][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : DROP TABLE T1]
[2015/10/21 17:29:49 576][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
[2015/10/21 17:29:53 577][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : CREATE TABLE T1 ( I1 INTEGER )]
[2015/10/21 17:29:53 578][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
[2015/10/21 17:29:53 579][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : DROP TABLE T1]
[2015/10/21 17:29:53 57A][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
10 logs printed.
```

콜 스택과 altibase_boot.log, altibase_sm.log을 함께 출력하되 상위 20개 로그 출력한다.

```

$ dumptrc -c -i error -i server -i sm -n 20
=====
= Callstack Information 0
=====
[2015/10/21 17:29:42 551][PID:32702][Thread-2]
  ==> Initialize Security Module
[2015/10/21 17:29:42 552][PID:32702][Thread-2]
  ... [SUCCESS]
...
[2015/10/21 17:29:42 564][PID:32702][Thread-2]
--- STARTUP Process SUCCESS ---
[2015/10/21 17:30:51 57B] Dump of Stack
SIGNAL INFORMATION =====
Signal 6(SIGABRT) caught.
  Sent by process : 2331
  Sent by user    : 1000
BEGIN-DUMP =====
===== SERVER =====
ALTIBASE hdb
  Product version : 6.7.1.0.0
  CPU              : X86
  Operating System : x86_64-unknown-linux-gnu
  Process ID       : 32698
  Thread No        : 0
END-DUMP =====
BEGIN-STACK [CRASH] =====
Caller[0] 0000000011E0EBF => iduStack::dumpStack(iduSignalDef const*, siginfo*, ucontext*)
Caller[1] 000000000426CFE => mmmSignalHandler
Caller[2] 00007FD306FD4CB0 => not found
Caller[3] 00007FD30609F763 => not found
Caller[4] 00000000042F66F => mmtSessionManager::run()
Caller[5] 0000000004213CE => mmi::serverStart(int, int)
Caller[6] 00000000041D1E0 => main
Caller[7] 00007FD305FD376D => not found
Caller[8] 0000000004206BD => _start
END-STACK =====
20 logs printed.

```

다른 디렉토리의 trc 로그에서 콜 스택을 가져와서 출력한다.

```

$ dumptrc -p /home/djin/work/altidev4/trunk/altibase_home/trc -c -n 20

Path : /home/djin/work/altidev4/trunk/altibase_home/trc
=====
= Callstack Information 0
=====
[2015/10/21 17:29:45 568][PID:32702][Thread-2]
[EXEC_DDL_END : SUCCESS]
[2015/10/21 17:29:45 569][PID:32702][Thread-2]
[EXEC_DDL_BEGIN : DROP TABLE T1]
...
[2015/10/21 17:30:51 57B] Dump of Stack
SIGNAL INFORMATION =====
Signal 6(SIGABRT) caught.
    Sent by process : 2331
    Sent by user    : 1000
BEGIN-DUMP =====
===== SERVER =====
ALTIBASE hdb
    Product version : 6.7.1.0.0
    CPU              : X86
    Operating System : x86_64-unknown-linux-gnu
    Process ID       : 32698
    Thread No        : 0
END-DUMP =====
BEGIN-STACK [CRASH] =====
Caller[0] 00000000011E0EBF => iduStack::dumpStack(iduSignalDef const*, siginfo*, ucontext*)
Caller[1] 0000000000426CFE => mmmSignalHandler
Caller[2] 00007FD306FD4CB0 => not found
...
Caller[8] 00000000004206BD => _start
END-STACK =====
20 logs printed.

```

killCheckServer

개요

killCheckServer는 실행 중인 checkServer를 종료한다.

```
killCheckServer
```

구문



설명

killCheckServer는 실행 중인 checkServer를 종료한다.

server stop 또는 server kill 명령을 사용해서 Altibase 서버를 종료할 경우, checkServer가 동작 중이라면 서버 종료 전에 먼저 server 스크립트 내에서 killCheckServer가 checkServer의 동작을 동작을 멈추도록 되어 있다. 그리고, \$ALTIBASE_HOME/trc 디렉토리의 killCheckServer.log 파일에 killCheckServer 수행 결과가 아래와 같이 기록된다.

- checkServer가 동작 중일 때:

```
checkServer killed.
```

- checkServer가 동작 중이 아닐 때:

```
ERROR CODE : -27
```

그러나, 사용자가 직접 killCheckServer 명령어를 실행할 경우에는 killCheckServer.log에 수행 결과가 기록되지 않는다.

사용예

셸 프롬프트 상에서 다음과 같이 수행한다.

```
$ killCheckServer
```

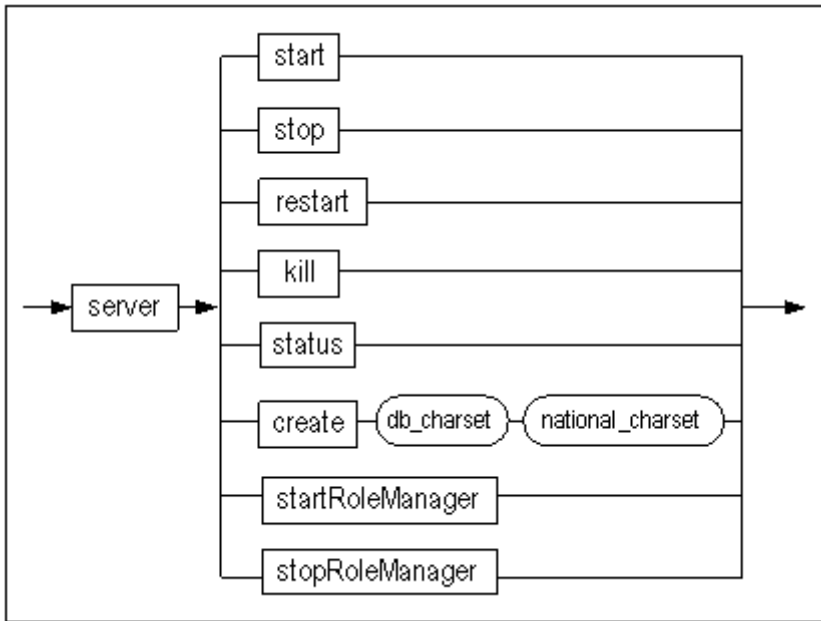
server

개요

Altibase 데이터베이스의 생성, 구동 및 종료시키기 위한 셸 스크립트이다.

```
server { start | stop | restart | kill | status | create db_charset national_charset | startRoleManager | stopRo
```

구문



파라미터

파라미터	설명
start	Altibase 프로세스를 구동한다.
stop	Altibase 프로세스를 종료한다.
restart	Altibase 프로세스를 재구동한다.
kill	Altibase 프로세스를 강제 종료한다.
status	데이터베이스 내의 테이블 목록을 출력한다.
create	크기 10M, 노아카이브 로그로 입력된 캐릭터 셋으로 데이터베이스를 생성한다.
startRoleManager	Altibase 프로세스를 Disaster Recovery의 역할 관리자로서 구동한다.
stopRoleManager	역할 관리자를 종료한다.

설명

Altibase의 구동 및 종료, 데이터베이스 생성은 iSQL상에서 SQL 구문으로 수행해야 한다. Altibase는 사용자 편의를 위해 관리자가 자주 사용하는 이러한 구문들을 묶어서 쉘 스크립트 파일 형태로 제공한다.

server 스크립트가 제공하는 기능은 다음과 같다.

- Altibase 프로세스의 구동
- Altibase 프로세스의 종료

- Altibase 프로세스의 재구동
- Altibase 프로세스의 강제 종료
- “SELECT * FROM TAB;” 결과 출력
- Altibase 데이터베이스 생성
- Altibase 프로세스를 역할 관리자로 구동
- 역할 관리자 종료

Altibase 관리를 위한 SQL 구문에 대한 자세한 내용은 *SQL Reference*를 참조한다.

역할 관리자에 대한 자세한 내용은 Disaster Recovery Manual을 참조한다.

사용예

사용 방법은 다음과 같다.

```
$ server start
$ server restart
$ server stop
$ server status
$ server kill
$ server create ksc5601 utf16
$ server startRoleManager
$ server stopRoleManager
```

참고사항

Administrator’s Manual, *SQL Reference*, *Disaster Recovery Manage* 를 참고하기 바란다.