CUBRID 운영자 교육

CUBRID2008 R4.1 이상

Date: 2012-05

기술본부

목차

CUBRID

1. CUBRID 시스템 구조

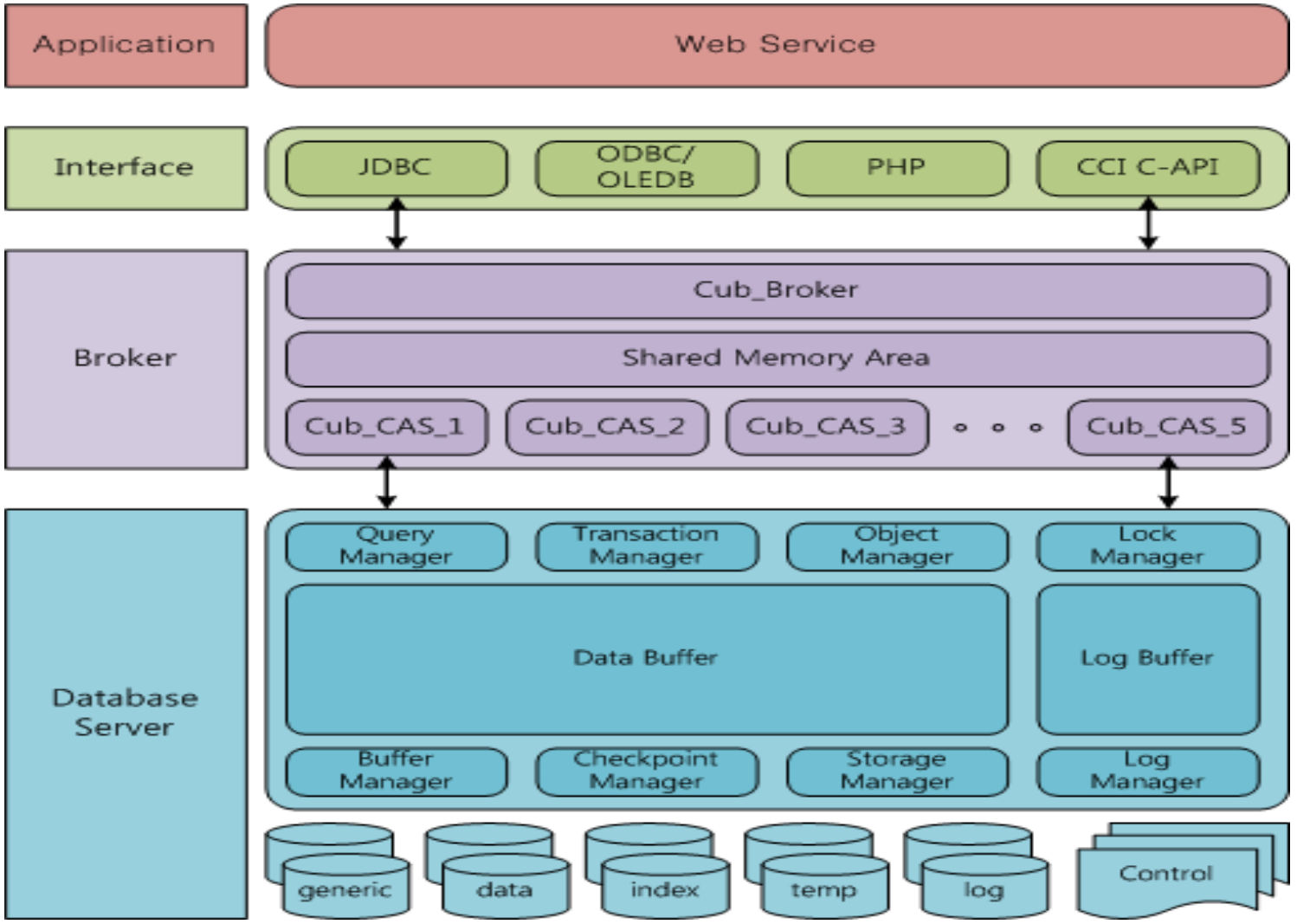
CUBRID 시스템 구조

- Database Server
  - 데이터를 저장 및 관리하는 기능을 수행.
  - Multi thread기반 client/server 방식으로 동작.
  - 사용자가 입력한 질의를 처리하고 DB 내의 객체 관리.
  - 운영에 필요한 백업 및 복구 기능을 지원

- Broker
  - 외부 응용 프로그램 간의 통신을 중계하는 역할.
  - CUBRID 전용 미들웨어.
  - 커넥션 풀링, 모니터링, 로그 추척 및 분석 기능 제공

- CUBRID Manager
  - GUI환경에서 Database Server와 Broker를 관리할 수 있는 tool.
  - CUBRID 설치 시 기본으로 CUBRID Manager Server 모듈이 같이 설치 되고 CUBRID Manager Client를 별도로 받아 사용한다(windows, linux GUI환경 제공).
  - 데이터베이스 관리에 대한 전반적인 기능을 제공하고, SQL문을 수행 시키고 결과를 확인 할 수 있는 질의 편집기를 제공한다.(질의 편집기만 별도로 제공하는 툴인 CUBRID Query Browser도 있다.)

CUBRID 시스템 구조도



CUBRID

CUBRID 프로세스

- CUBRID 설치 후 서비스를 위한 구동이 완료되면 아래의 프로세스를 확인 할 수 있다.
  - “ps -ef | grep cub_*” 명령어를 이용하여 구동 process 확인

- Master Process
  - 프로세스 검색 명령어에 의해 “cub_master” 로 확인
  - 하나의 Engine에서 하나 씩 구동된다. (한대의 서버에는 여러 개의 Engine을 설치 할 수는 있으나 권장하지 않음)
  - Client와 Server프로세스 사이의 연결을 담당하는 프로세스이다.

- Database Server Process
  - 프로세스 검색 명령어에 의해 “cub_server <db_name>”의 형태로 확인
  - 데이터베이스가 구동되어 있을 때 검색되는 프로세스.
  - 구동되는 데이터베이스의 수 만큼 확인된다. (한 개의 Engine에서 여러대의 DB가 구동될 수 있다.)
  - 데이터베이스 파일 및 로그 파일 등에 직접 접근하여 사용자 요청을 처리하는 프로세스.

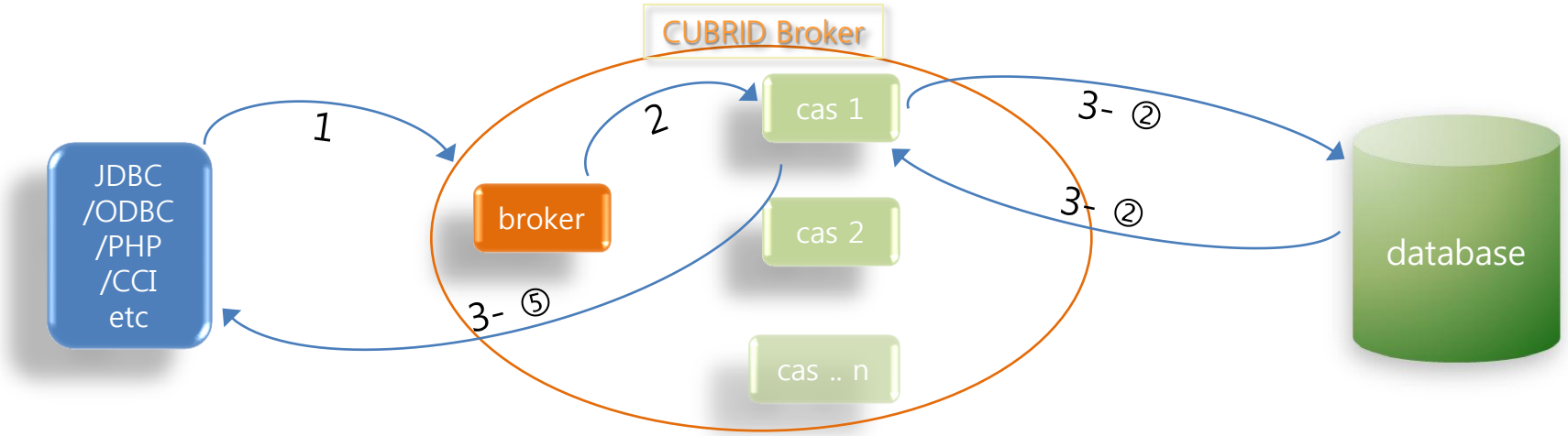
- Broker Process
  - 프로세스 검색 명령어에 의해 “cub_broker”의 형태로 확인
  - Broker가 구동되어 있을 때 검색되는 프로세스.
  - cubrid_broker.conf에 등록되어 Service 상태가 ON인 Broker의 개수 만큼 확인된다.
  - 응용 Client(JDBC, ODBC, PHP 등)와 cub_cas 프로세스 사이의 연결을 중계하는 기능을 수행한다.
  - cub_cas 프로세스의 상태를 모니터링 및 관리한다.

CUBRID 프로세스 - 계속

- Cas Process
  - 프로세스 검색 명령어에 의해 “cub_cas”의 형태로 확인
  - Broker가 구동 되어 있어야만 확인할 수 있는 프로세스로 cub_broker 프로세스가 없이 존재 한다면 비정상 프로세스다.
  - DB에 연결하고자 하는 모든 종류의 응용 Client가 사용하는 공용 응용 서버의 역할을 한다.
  - 환경 설정에 따라 cub_broker 프로세스에 의해 동적으로 조정된다.
  - Query 분석이나 최적화, 실행 계획 생성 등의 작업이 수행된다.

- Manager Process
  - cub_auto Process
    - 프로세스 검색 명령어에 의해 “cub_auto”의 형태로 확인
    - CUBRID Manager Client 사용자의 인증 처리를 및 주기적인 자동화 작업, 진단 정보 수집하는 프로세스.
    - cub_auto는 cm_port를 사용하며 cm_port는 8001번이 기본 설정이다.(사용자 환경에 따라 변경 가능)
  - cub_js Process
    - 프로세스 검색 명령어에 의해 “cub_js”의 형태로 확인
    - CUBRID Manager Client에서 전송되는 사용자 요구를 수행한다.
    - 8002번 port를 사용하며 cm_port보다 1만큼 큰 값을 사용하도록 지정되어 있다.

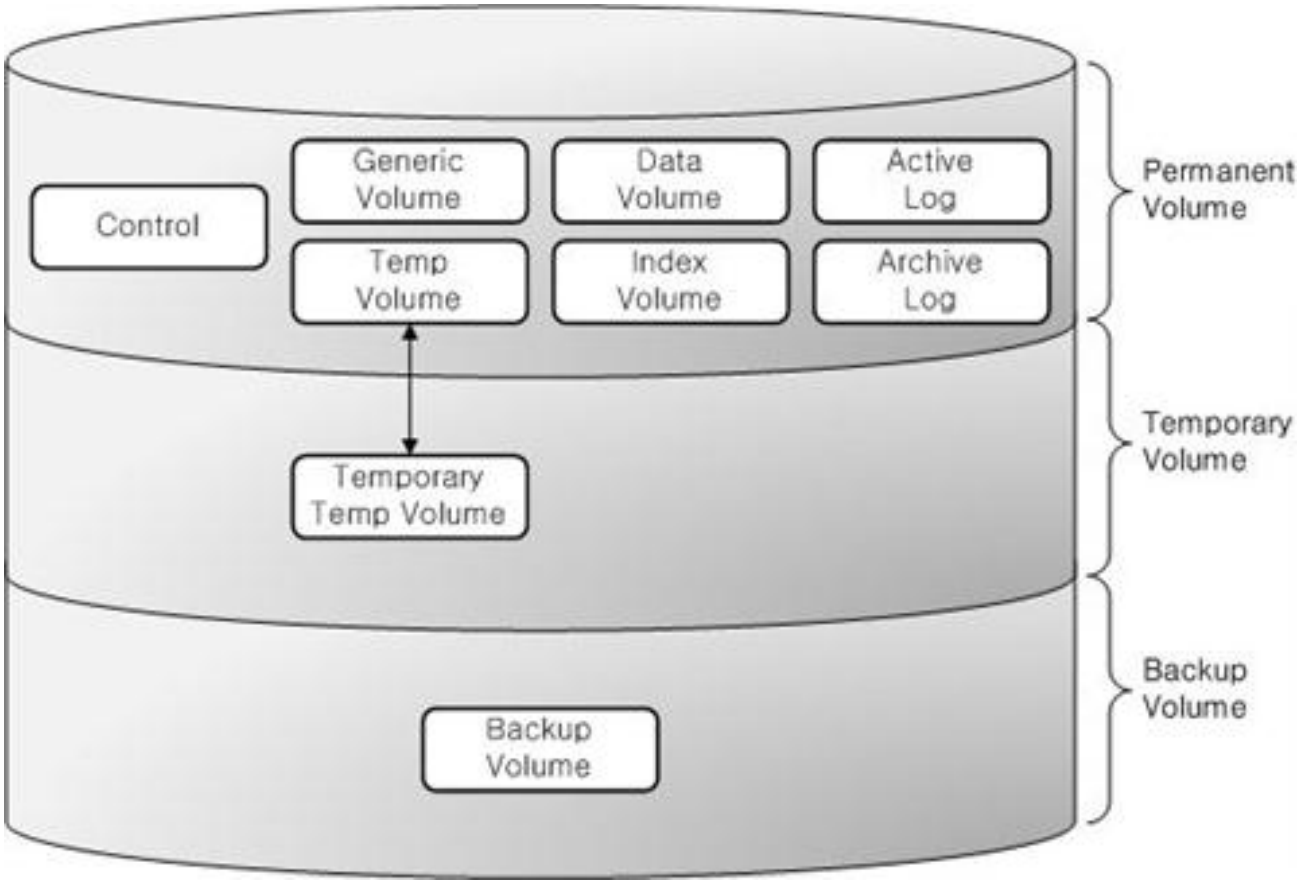
CUBRID 처리 프로세서



1.응용에서 질의처리를 위해 미들웨어에 작업 요청
  ① 실 작업처리는 cas 가 수행하며, 프로세스 단위 작업
  ② 응용에서 단위 cas에게 직접 요청은 어려우므로, 이들을 관리하기 위한 broker 를 두고 broker 에게 작업 요청
2.broker 는 사용가능한 cas 에게 작업을 할당
3.cas는 데이터베이스에 연결하여 작업 수행 후 결과를 응용에 전달
  ① 데이터베이스는 사용자가 지정하며, 지정한 데이터베이스에 cas 가 연결하는 형식
  ② cas 는 데이터베이스와 연결하여 작업 후, 연결 유지(connection pool)
  ③ 이후 동일한 데이터베이스와 작업 요청 시 기존 연결 재사용
  ④ 다른 데이터베이스와 작업 요청 시 현재 연결을 끊고, 새로이 연결
    ➔ 연결 overhead 발생 → broker 별로 데이터베이스를 지정(사용자)하여 사용 권장
  ⑤ 작업결과를 응용에 전달

✓ 트랜잭션 처리 중에는 하나의 cas가 하나의 응용에 종속됨, 트랜잭션 처리 종료 전까지는 다른 요청 받지 않음
  ✓ 제한된 개수의 cas 를 통한 서비스 처리이므로, cas 사용시간 단축을 통한 다중 응용 처리 성능 향상
  ✓ select 도 트랜잭션의 일부

데이터베이스 구성 파일



데이터베이스 구성 파일 - 계속

- Permanent Volumes
  - Information Volumes
    - Information volume은 응용 프로그램의 모든 데이터를 저장하는 영구 볼륨 파일이다.
    - Information volume은 생성할 때 그 사용 목적을 명시 할 수 있다.
      - Data volume
        - » 스키마, 레코드, 멀티미디어 데이터와 같은 응용 프로그램의 데이터를 저장
      - Index volume
        - » 신속한 질의 처리 또는 무결성 제약 조건(integrity constraints)의 신속한 검증을 위하여 인덱스 정보를 유지하는 볼륨이다.
      - Temp volume
        - » 질의 처리(join, group by, order by, sub-query, create index…)나 정렬(sorting)을 위해 사용.
        - » 이것은 permanent 볼륨으로 고정되어 있으며, Temporary Temp volume과는 다르다.
      - Generic volume
        - » DB생성 시 초기 볼륨이며 data, index볼륨으로 사용 가능하다.
  - Log Volumes
    - Active 로그 볼륨은 데이터베이스에 반영된 가장 최근의 갱신 기록을 포함하는 볼륨이다.
    - committed/aborted/active 트랜잭션의 상태를 기록한다.
    - 매체 고장이 발생할 경우 데이터베이스 복구를 위해 사용된다.
    - 각 데이터베이스 마다 하나의 active 로그 볼륨을 가진다.
    - Active 로그로 할당된 스페이스가 모두 사용되면, active 로그의 내용은 새로운 로그(archive 로그)로 복사되어 보관된다

데이터베이스 구성 파일 - 계속

- Control Information Volumes
  - Volume information 파일
    - 데이터베이스 볼륨에 대한 정보를 포함.
  - Log information 파일
    - 로그 볼륨에 관한 정보를 포함.
  - Backup volume information 파일
    - 백업 볼륨에 관한 정보를 포함.
  - databases.txt 파일
    - 데이터베이스의 위치(디렉토리) 정보를 담고 있는 파일
  - cubrid.conf 파일
    - CUBRID server 파라미터의 설정값이 저장되어 있는 파일
  - cubrid_broker.conf 파일
    - CUBRID broker 파라미터의 설정값이 저장되어 있는 파일

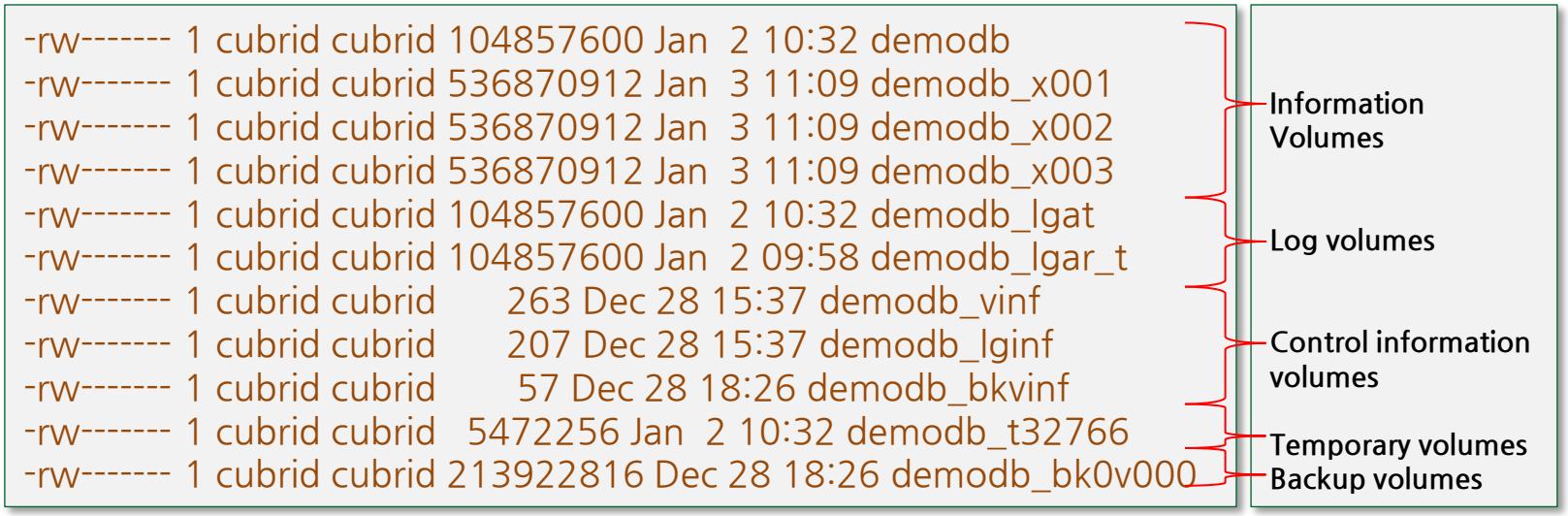
- Temporary Temp Volumes
  - Temporary temp volume은 일시적으로 사용되는 볼륨파일로써 사용자가 영구 볼륨으로 지정한 공간을 초과하여 데이터가 축적되는 경우에만 생성된다.
  - Temporary temp volume은 DB가 재시작 할 때 삭제된다.

- Backup Volumes
  - Backup 작업에 의해 생성되는 파일이다.
  - Backup 볼륨은 백업할 시점 데이터베이스의 스냅샷이다.

데이터베이스 구성 파일 - 계속

- databases.txt파일
  - 데이터베이스의 이름, 경로, 구축된 호스트 이름 정보가 기록되어 있다.
  - 데이터베이스 생성 시, databases.txt 파일에 생성된 데이터베이스에 관한 정보가 기록된다.
  - CUBRID_DATABASES 환경 변수가 명시되어 있는 경로에 저장된다
  - 이 파일에 대한 주의 사항
    - 호스트 이름이 변경되거나, 데이터베이스를 rm과 같은 명령어로 삭제했을 경우 이 파일도 수정해야 한다. DB삭제 시에는 DB명령어를 이용해 삭제할 것을 권장한다.
    - 데이터베이스 생성 및 삭제 시 databases.txt 파일을 수정할 수 있어야 하므로, 데이터베이스를 생성하거나 삭제하는 사용자는 그 파일에 쓰기 접근을 가져야 한다. 이 디렉토리에 쓰기 권한이 없는 사용자가 데이터베이스를 생성한다면 데이터베이스 생성이 실패하게 된다. 따라서 DBA는 그 디렉토리 및 파일에 사용자 쓰기 권한을 허용해야 한다

- 데이터베이스 볼륨파일 정리



2. CUBRID 설치

CUBRID 설치 전 확인 사항

- Linux에서 설치
  - OS버전확인 : OS버전에 상관 없이 glibc 2.3.4버전 이상만 지원
    - 확인방법 : %rpm -q glibc
  - 32/64bit 여부 : CUBRID 2008은 R2.0부터 32bit와 64bit버전을 지원한다.
    - linux OS bit 확인 방법 : uname -a
    - 결과 : Linux host_name 2.6.18-53.1.14.el5xen #1 SMP Wed Mar 5 12:08:17 EST 2008 x86_64 x86_64 x86_64 GNU/Linux → 64bit OS
  - 방화벽 설정(해당 port는 기본적으로 open되어 있어야 한다.)
    - CUBRID Manager : 8001, 8002
    - 질의편집기 : 30000
    - 응용개발 : 33000
  - SELINUX 설정 (/usr/CUBRID 아래 설치 가정)

```
/sbin/restorecon -R -v /usr/CUBRID/lib/libcubridcs.so
/usr/bin/chcon -t texrel_shlib_t /usr/CUBRID/lib/libcubridcs.so
/sbin/restorecon -R -v /usr/CUBRID/lib/libcubridsa.so
/usr/bin/chcon -t texrel_shlib_t /usr/CUBRID/lib/libcubridsa.so
/sbin/restorecon -R -v /usr/CUBRID/lib/libcubrid.so
/usr/bin/chcon -t texrel_shlib_t /usr/CUBRID/lib/libcubrid.so
/sbin/restorecon -R -v /usr/CUBRID/lib/libbrokeradmin.so
/usr/bin/chcon -t texrel_shlib_t /usr/CUBRID/lib/libbrokeradmin.so
```

CUBRID 설치 전 확인 사항 - 계속

- Windows에서 설치
  - 지원 OS : Windows XP 32bit, Windows Vista 32/64bit, Windows 2003 server 32bit
  - 32/64bit 여부 : CUBRID 2008에서는 R2.0부터 32bit, 64bit 제품을 지원한다.
    - OS bit 확인 방법 : 내컴퓨터 → 속성 → 시스템에서 확인 가능.
  - 설치 유형 선택
    - 전체설치 : CUBRID서버와 명령행 도구 및 인터페이스 드라이버(JDBC, ODBC, OLEDB 등)가 설치된다.
    - 인터페이스 드라이버 설치 : 드라이버(JDBC, ODBC, OLEDB)만 설치된다. 원격지 서버 접속 시 설치한다.
  - 방화벽 설정(Windows의 경우 아래의 port 전부에 대해서 접근 port를 열어 주어야 한다)
    - CUBRID Manager : 8001, 8002
    - 질의편집기 : 30000 ~ 30040
    - 응용개발 : 33000 ~ 33040
- JAVA 설치
  - CUBRID Manager client / CUBRID Query Browser 사용시
  - Stored Procedure 사용시
  - JRE 1.6이상 버전이 설치되어 있어야 한다.

설치

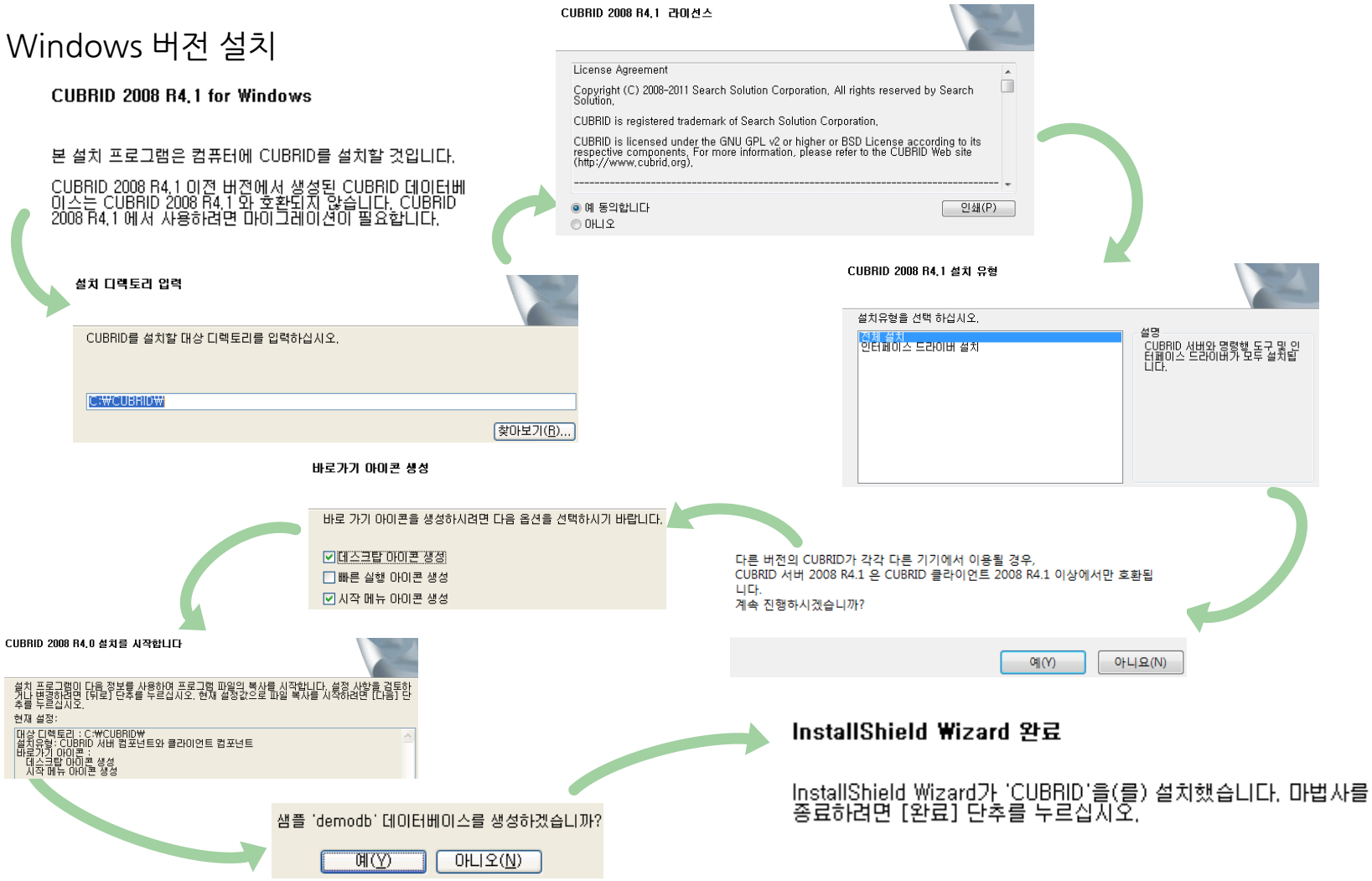
- Linux 버전
  - CUBRID사용 user생성 후 해당 계정에서 수행
  - Default로 생성 계정 $HOME directory에 CUBRID folder에 제품이 설치됨.
  - Server와 Client는 동일 버전 간에만 완전한 호환을 지원한다.

```
$ sh CUBRID-8.4.1.2032-linux.x86_64.sh
Copyright (C) 2008-2011 Search Solution Corporation. All rights reserved.
...
Do you agree to the above license terms? (yes or no) : yes
Do you want to install this software(CUBRID) to the default(/home/cubrid/CUBRID) directory? (yes or no) [Default: yes] : yes
In case a different version of the CUBRID product is being used in other machines, please note that the CUBRID 2008 R4.1 servers are only compatible with the CUBRID 2008 R4.1 clients and vice versa.Do you want to continue? (yes or no) [Default: yes] : yes
...
If you want to use CUBRID, run the following commands
  % . /home/cubrid/.cubrid.sh
  % cubrid service start
```

- Windows 버전
  - Administrator권한 설치 권장
  - CUBRID Manager Client 사용을 위해 JRE 1.6 설치 필요.(전체 설치에 CM이 포함되므로 설치 필요)

설치 - 계속

- Windows 버전 설치



CUBRID

설치 - 계속

- 설치 관련 참고사항
  - CUBRID는 여러 개의 DB instance를 별도로 생성할 수 있다. 각각은 별도의 계정으로 관리 되어야 한다.

```
@ cubrid server status
 Server demodb (rel 8.4, pid 29706)
 Server testdb (rel 8.4, pid 29493)
...
```

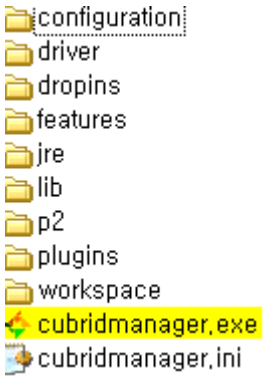
  - 업그레이드 시 현재 설치된 engine을 삭제 후 재 설치 해야 한다.
  - Major업그레이드에 따른 재구성 시에는 unload를 이용하여 데이터를 받아낸 후 engine을 재설치 후 load를 이용하여 작업을 진행해야 한다.

CUBRID Engine 구조

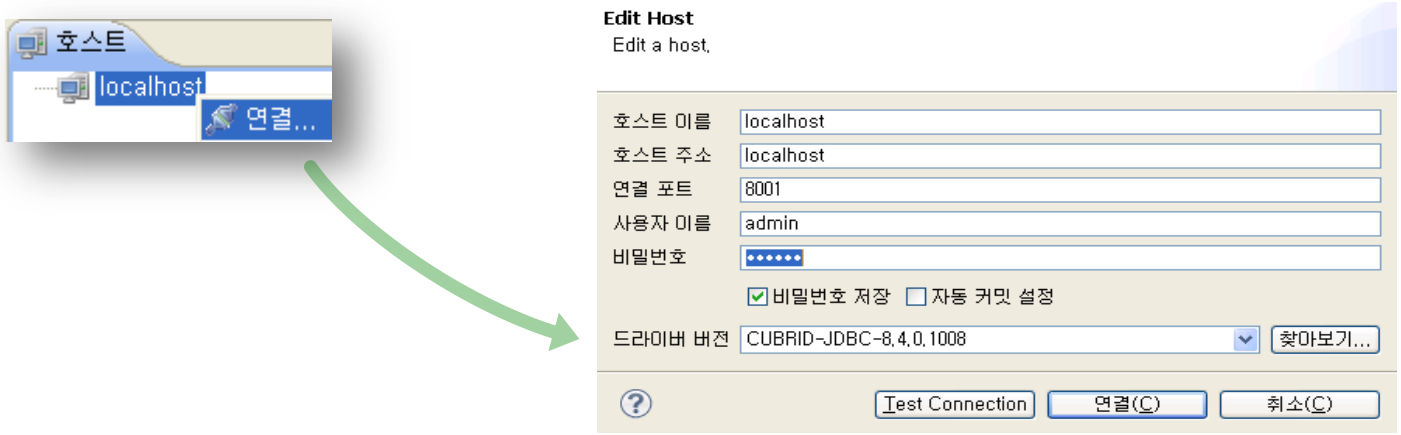
CUBRID/	CUBRID 의 홈 디렉토리
bin/	실행 파일이 위치한 디렉토리
compat/(LINUX)	7.x이하 버전 명령어가 위치한 디렉토리
conf/	환경 설정 파일이 위치한 디렉토리
databases/	databases.txt 와 sample 데이터베이스가 위치한 디렉토리
demo/(LINUX)	demodb생성 script가 저장된 디렉토리
include/	inlude file 이 위치한 디렉토리
java/	JAVA SP 관련 파일이 위치한 디렉토리
jdbc/	CUBRID jdbc driver가 위치한 디렉토리
lib/	library 파일이 위치한 디렉토리
lib32/	32bit OS 를 위한 library 파일이 위치한 디렉토리
log/	각종 log 가 저장되는 디렉토리
msg/	CUBRID 에서 사용되는 각종 메시지 파일이 저장된 디렉토리
share/	HA관련 script가 저장된 디렉토리
tmp/	
var/	

CUBRID Manager Client 설치

- CUBRID Manager Client 를 별도로 설치할 경우 설치 방법
  - CUBRID Manager Client를 다운로드 받아 설치(또는 zip파일로 받아 압축 해제)
  - 구동(cubridmanager.exe파일 클릭)

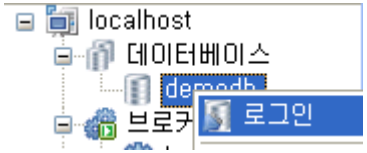


  - 로그인(default 사용자인 admin의 비밀번호는 admin이다. 로그인 후 비밀번호를 바로 변경해야 한다)

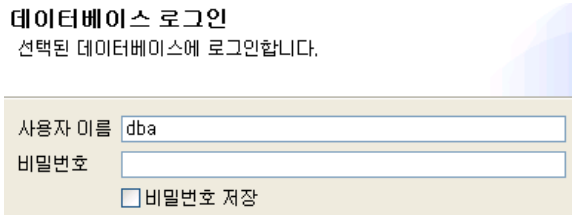


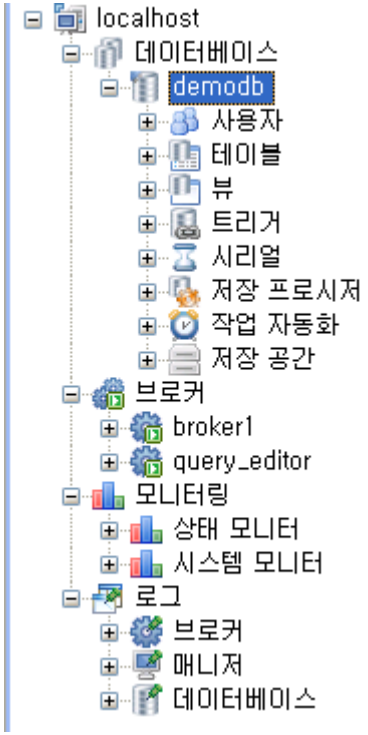
CUBRID Manager Client 접속방법

- CUBRID Manager Client
  - 데이터베이스 로그인
    - 데이터베이스 별 사용자로 로그인 한다.



    - Default 계정: dba, public
    - Default 비밀번호는 없다.(공백으로 두고 접속)





실습 1

제품설치 실습

3. CUBRID 구동 및 종료

CUBRID 서비스 구동

- 서비스 구동
  - CUBRID 운영에 필요한 기본 프로세스 구동
  - CUBRID 사용자 계정으로 로그인 필요
  - master, broker, manager server 구동
  - database server 는 별도 구동, 또는 설정을 통하여 서비스 구동 시 같이 구동 가능

- 명령어

```
$ cubrid service start
@ cubrid master start
++ cubrid master start: success
@ cubrid broker start
++ cubrid broker start: success
@ cubrid manager server start
++ cubrid manager server start: success
```

- Windows
  - service 에 등록되어 부팅 시 구동(default)

CUBRID 서비스 종료

- 서비스 종료
  - CUBRID 관련 모든 프로세스 종료
  - CUBRID 사용자 계정으로 로그인 필요
  - master, broker, manager server 및 database server 종료

- 명령어

```
$ cubrid service stop
@ cubrid broker stop
++ cubrid broker stop: success
@ cubrid manager server stop
++ cubrid manager server stop: success
@ cubrid master stop
++ cubrid master stop: success
```

- Windows
  - Exit 를 선택하면, 서비스 종료 후 tray 응용까지 종료됨.

CUBRID

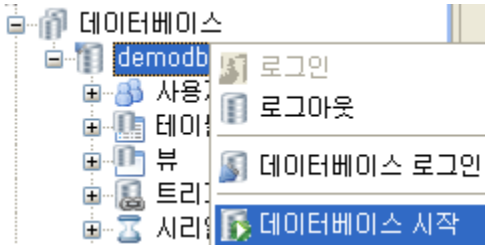
CUBRID Server 구동

- 데이터베이스 구동
  - 사용하는 데이터베이스 별 서버 구동

- 명령어
  - CUBRID 사용자 계정으로 로그인

```
$ cubrid server start demodb
@ cubrid server start: demodb

This may take a long time depending on the amount of recovery works to do.

CUBRID 2008 R4.1

++ cubrid server start: success
```

- Windows
  - CUBRID Manager Client에서 database dba 계정으로 로그인해야만 구동이 가능.



CUBRID Server 종료

- 데이터베이스 종료
  - 사용하는 데이터베이스 별 서버 종료

- 명령어
  - CUBRID 사용자 계정으로 로그인

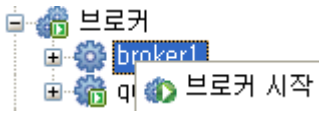
```
$ cubrid server stop demodb
@ cubrid server stop: demodb

Server ons_db notified of shutdown.
This may take several minutes. Please wait.
++ cubrid server stop: success
```

- Windows
  - CUBRID Manager에서 database dba 계정으로 로그인

CUBRID Broker 구동/종료

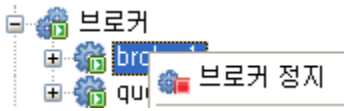
- 구동
  - CUBRID가 설치되어 있는 호스트의 브로커 구동.
  - CUBRID service 구동 시 자동으로 구동된다.

```
% cubrid broker start
@ cubrid broker start
++ cubrid broker start: success
➔ 이미 구동되어 있을 경우  아래와 같이 출력
 ++ cubrid broker is already running.
```



- 종료
  - CUBRID가 설치되어 있는 호스트의 브로커 종료.
  - CUBRID service종료 시 자동으로 종료.

```
% cubrid broker stop
@ cubrid broker stop
++ cubrid broker stop: success
➔ 이미 종료되어 있을 경우
++ cubrid broker is not running.
```



CUBRID Manager Server 구동/종료

- 구동
  - CUBRID가 설치되어 있는 호스트의 Manager Server 구동.
  - CUBRID service 구동 시 자동으로 구동된다.

```
% cubrid manager start
@ cubrid manager server start
++ cubrid manager server start: success
➔ 이미 구동되어 있을 경우  아래와 같이 출력
 ++ cubrid manager server is already running.
```

- 종료
  - CUBRID가 설치되어 있는 호스트의 Manager Server 종료.
  - CUBRID service종료 시 자동으로 종료.

```
% cubrid manager stop
@ cubrid manager server stop
++ cubrid manager server stop: success
➔ 이미 종료되어 있을 경우
++ cubrid manager server is not running.
```

실습 2

제품 구동 및 구동 관련 설정 실습

4. 데이터베이스 생성

데이터베이스 생성

- CUBRID는 engine을 설치한다고 데이터베이스가 같이 설치 되지 않는다. 따라서 CUBRID engine을 설치한 후 데이터베이스를 생성 해 주어야 한다.(demodb는 사용자의 편의를 위해 sample Data를 가지고 있는 데이터베이스이며, CUBRID engine설치 시 추가적으로 생성되는 데이터베이스이다. 서비스를 위해서는 별도의 데이터베이스를 추가하여 작업 할 것을 권장한다.)

- 데이터베이스 생성
  - 데이터 보존 연한간 전체 데이터의 크기를 예측하여 생성
  - 디스크 I/O 를 최소화할 수 있도록 디스크 구성 및 디렉토리의 적절한 분산
  - 생성된 크기를 줄일 수는 없음

- 기본 정보
  - 페이지 크기 : 16384 bytes(디스크 I/O 단위이며, 성능상 무난한 크기임)
  - 볼륨 : 스키마나 데이터정보 등이 저장되는 공간
    - 32bit 사용시 한 개 볼륨의 최대 크기 : 2G
    - 일반 볼륨 : 스키마정보, 카다로그 정보 저장됨.
    - 로그 볼륨 : 데이터 변경에 관련된 정보 저장. 백업 복구 시 사용됨. 200M 무난함
    - 데이터, 인덱스 볼륨 : 각 데이터, 인덱스 저장됨.
    - 템프 볼륨 : 조인, 정렬 등의 질의 처리에 사용되는 공간. 데이터의 10%가 무난하나 질의의 복잡도에 따라 변동.

데이터베이스 생성 - 계속

- 명령어 사용 : createdb
  - '#' 문자로 시작하는 데이터베이스 명을 부여할 수 없다.
  - DB 생성 명령어.

```
% cubrid createdb [option] database_name
```

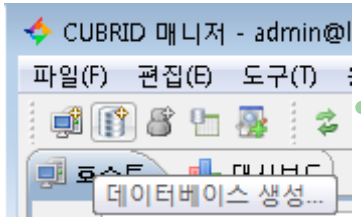
옵션	옵션 천체 이름	설 명	초기값
-F	--file-path	초기 볼륨이 위치할 경로 지정	현재
-L	--log-path	로그 볼륨이 위치할 경로 지정	현재
-B	--lob-base-path	LOB파일이 저장될 위치 경로 지정	<File-path>/lob
-r	--replace	DB가 이미 존재하는 경우 기존 데이터베이스를 삭제하고 재생성함.	존재 시 에러 발생
	--db-volume-size	생성되는 데이터베이스 볼륨의 크기를 바이트 단위로 지정한다.	512M
	--db-page-size	데이터베이스 페이지의 크기를 바이트 단위로 지정한다.	16K
	--log-volume-size	로그 볼륨의 크기를 지정한다.	db_volume_size와 동일
	--log-page-size	로그 볼륨의 페이지 크기를 바이트 단위로 지정한다	db_page_size와 동일

- 사용 예

```
cubrid createdb --db-volume-size=100M -F /data --log-volume-size=200M -L /log edudb
```

데이터베이스 생성 - 계속

- CUBRID Manager를 이용한 DB생성



1. 기본정보
    1. 생성할 데이터베이스 이름
    2. Page크기 (default 16K)

2. 일반 볼륨 정보
    1. 초기 생성 볼륨의 크기
    2. Page 수(생성 볼륨 크기 환산 값)
    3. 초기 볼륨 생성 위치

3. 로그 볼륨 정보
    1. page 크기(default는 DB page값과 동일
    2. 로그 볼륨 크기
    3. 로그 볼륨 page 수
    4. 로그 볼륨 생성 위치

CUBRID

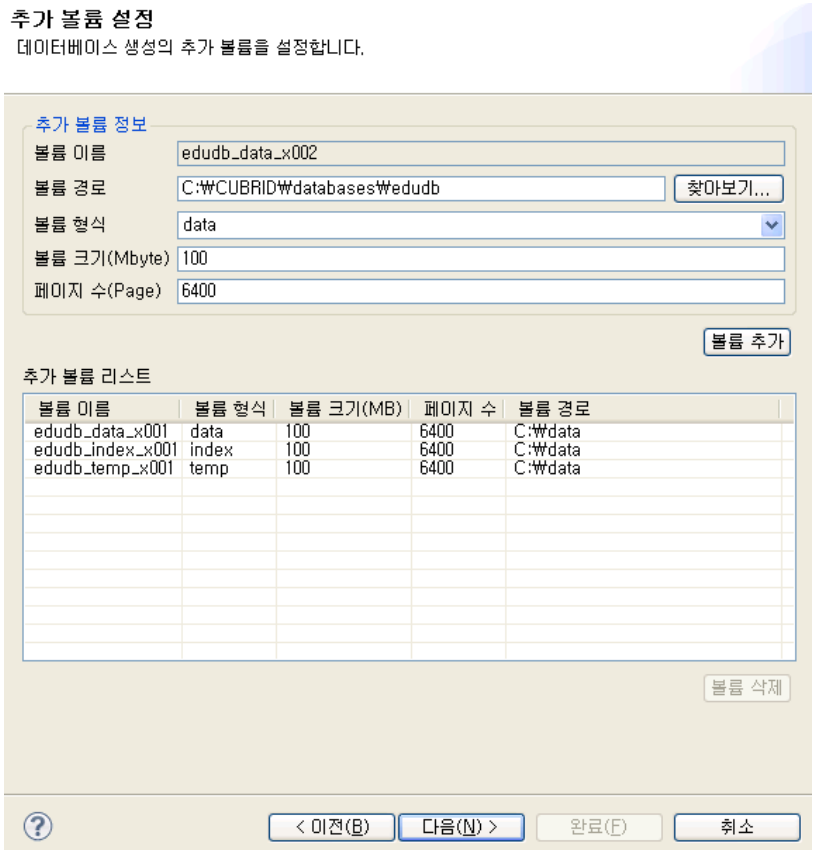
데이터베이스 생성 - 계속

- 볼륨 추가
  - 용도별 볼륨을 분산하는 것이 성능상 효율적.
  - 운영상 적합한 크기를 예측하여 사용용도별 볼륨 확장 필요.

1. 추가 볼륨 정보
    1. 추가할 볼륨의 이름을 기록
    2. 생성될 볼륨 위치
    3. 볼륨의 용도
    4. 볼륨 크기
    5. 볼륨 Page 수(볼륨 크기의 page환산 값)

2. 추가 볼륨 리스트

- Default로 추가되도록 설정된 볼륨으로 삭제하고 다시 추가하는 것이 가능. 리스트를 클릭 후 볼륨 삭제 버튼을 이용하여 제거 가능.



데이터베이스 생성 - 계속

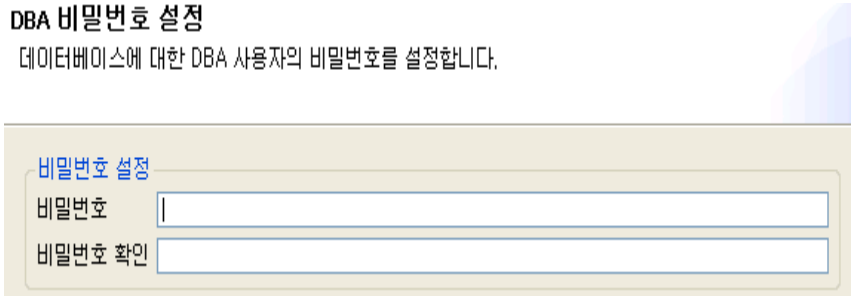
- 볼륨 자동 증가 설정(CUBRID Manager에서만 지원)
  - 각각의 볼륨의 여유 공간 부족 시 볼륨 자동 생성
  - 확장 볼륨 기준(여유공간 비율) 및 자동 확장 볼륨 크기 지정

1. 볼륨 형식

- 데이터, 인덱스 볼륨에 대하여 여유 공간 비율 설정 및 생성될 볼륨 크기를 기록한다. 확장 페이지 수는 볼륨 크기에 따라 자동 변환 된다.
- 데이터와 인덱스 볼륨에 대해서만 자동 볼륨 추가 설정이 가능하다.

- DBA 비밀번호 설정
  - dba의 비밀번호를 설정한다.
  - console작업 시 dba의 default 비밀번호는 없다.

- 데이터베이스 설치 완료
  - 지금까지 설정한 내용들을 보여주고 DB설치 종료.



CUBRID

볼륨 추가

- 명령어 사용 : addvoldb
  - 볼륨 추가 명령어

```
% cubrid addvoldb  [options]  database_name
```

옵션	옵션 전체 이름	설 명	초기값
-F	--file-path	추가되는 데이터베이스 볼륨이 생성될 디렉터리 경로를 지정한다.	DB생성경로
-p	--purpose	추가되는 데이터베이스 볼륨의 용도를 지정한다.	Generic볼륨
-S	--SA-mode	Off-line 상태에서 데이터베이스 볼륨 추가 작업을 실행한다.	On-line
-C	--CS-mode	On-line 상태에서 데이터베이스 볼륨 추가 작업을 실행한다.	On-line
	--db-volume-size	추가되는 데이터베이스 볼륨의 크기를 지정한다.	512M

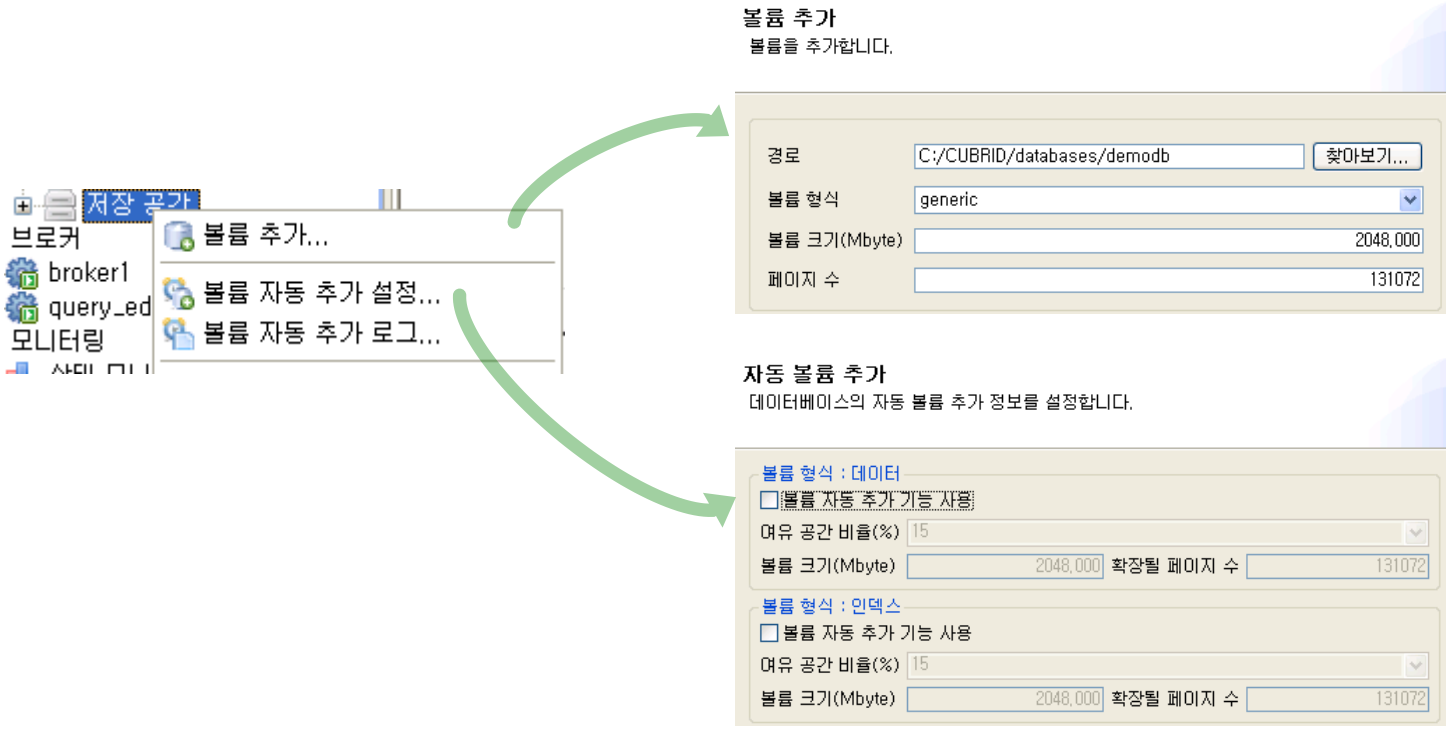
* -F 옵션의 경우 초기 값이 시스템 parameter의 volume_extension_path의 설정 값을 따른다. 기본값은 DB생성 경로이다.

- 사용 예
  - 볼륨
    - 일반위치: /data, 로그위치: /log
    - 데이터 2G, 인덱스 1G, 템프 1G 생성

```
cubrid addvoldb -S -p data -F /data --db-volume-size=2G edudb
cubrid addvoldb -S -p index -F /data --db-volume-size=1G edudb
cubrid addvoldb -S -p temp -F /data --db-volume-size=1G edudb
```

볼륨 추가 - 계속

- 볼륨 추가
  - Data, Index 볼륨의 여유 공간이 부족할 경우 볼륨을 추가
- 확장 자동화
  - Data, Index 볼륨의 여유 공간이 부족할 경우 볼륨 자동 생성 설정
  - 확장될 볼륨의 기준(여유공간 비율) 및 자동 확장이 되는 볼륨의 크기를 지정
  - CUBRID Manager에서 지원



실습 3

DB생성 및 구동 관련 설정 실습

5. 백업 및 복구

데이터베이스 백업 시 참고사항

- 데이터베이스 백업 시 참고사항
  - 데이터베이스 백업은 데이터베이스의 이미지를 파일 등으로 덤프하는 것이다 .
  - OS 유틸리티를 사용한 데이터베이스 백업은 권장하지 않는다.
  - 데이터베이스 백업 시점에 불필요한 로그 아카이브 볼륨들을 정리 할 수 있다.
  - 백업은 매일 받는 것을 권장하며, DBA가 수작업으로 하는 것보다는 자동 백업을 하는 것이 편리하다.
  - 백업 볼륨은 외부 저장장치에 백업하는 것이 안전하다.
  - 데이터베이스를 새로운 버전으로 migration했다면 새로 생성한 데이터베이스를 즉시 백업해야 한다. 이전 버전의 백업 볼륨은 새로운 버전 복구에 사용 불가능하기 때문이다.
  - 백업 볼륨을 이동하거나 이름을 변경하지 않으면 이후의 백업 시에 이전 볼륨을 덮어쓴다.
  - 백업 볼륨은 데이터베이스를 가장 최근의 상태로 복구하기 위해서 로그 볼륨과 함께 사용된다.

- 데이터베이스 백업 정책
  - 백업할 데이터의 선택
    - 데이터베이스의 전체 또는 일부만 백업 할 것인가?
    - 데이터 보존 기간은 얼마로 할 것인가?
    - 데이터베이스와 함께 백업되어야 할 다른 파일은 있는가?
  - 백업 방법
    - 사용 가능한 백업 툴 및 백업 장비
    - FULL/INCREMENTAL백업(0,1,2 level)
    - On-line/Off-line 백업 → on-line백업 시 archive log가 증가하며 백업 본과 같이 보관해야 완전 복구가 가능
  - 백업 시기
    - 데이터베이스의 활동이 가장 적은 시기

CUBRID

백업

- 명령어 이용한 백업: backupdb

```
% cubrid  backupdb  [options]  database_name
```

  - Disk와 tape등의 미디어 지정 가능.
  - 백업 볼륨 파일 및 백업 정보 파일 생성. (예, demodb_bk0v000, demodb_bkvinf)

옵션	인자	설명	기본값
-S | -C		stand-alone, client-server mode 지정(On/Off-line)	-C
-D	filepath/device	볼륨이 저장될 경로 지정	log file 경로와 같음
-l	0,1,2	백업 레벨	0
-r		백업 후 불필요한 archive log 삭제	수행 않음
--no-check		백업 전에 데이터베이스 일관성 점검을 수행하지 않는다.	수행
-z		데이터베이스를 압축하여 백업 볼륨에 저장함	수행 않음
-t	integer	백업을 수행하는 thread 수	0<auto>
-e		백업 시 활성 로그 볼륨을 포함하지 않도록 설정 함	수행 않음

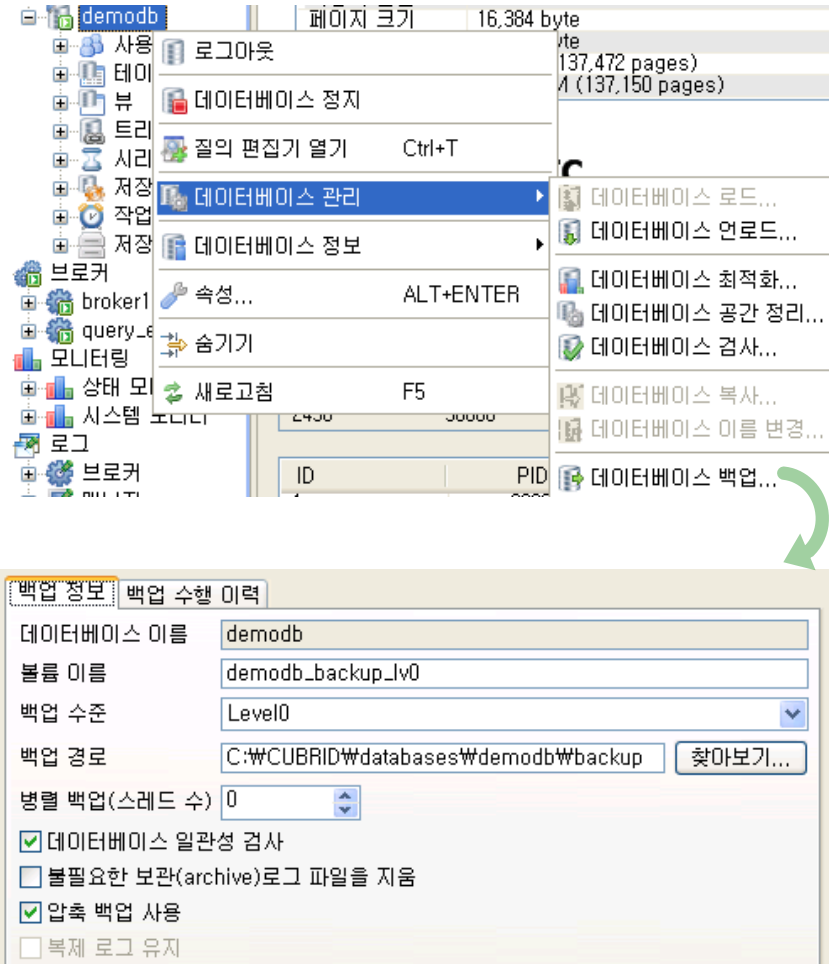
- 사용 예

```
% cubrid  backupdb  -S demodb
➔ % cubrid backupdb -S -D C:₩CUBRID₩databases₩demodb -l 0 demodb
```

백업 - 계속

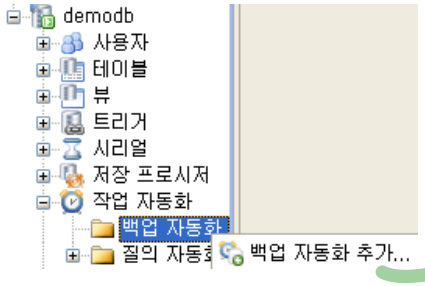
- CUBRID Manager를 이용한 백업 방법

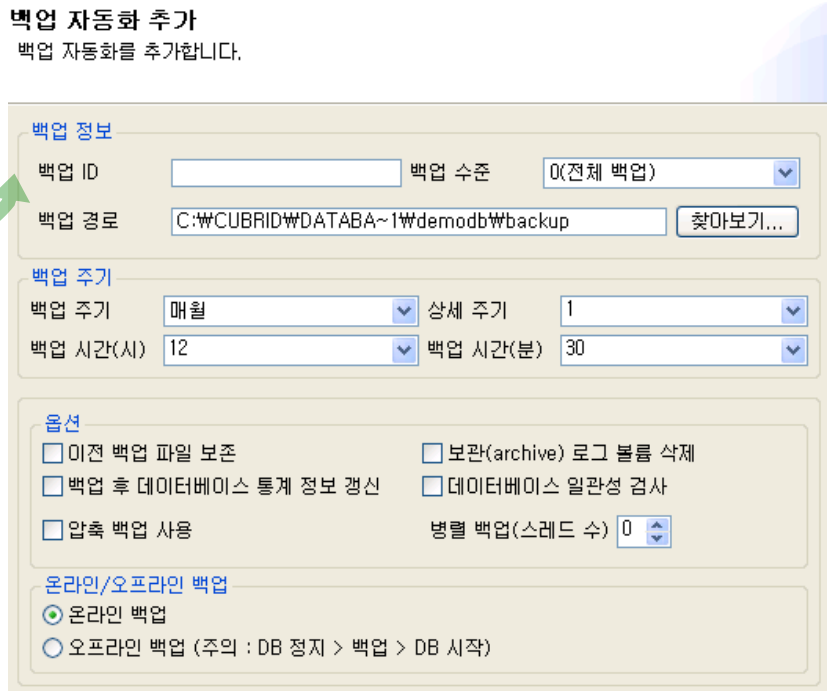
1. 데이터베이스를 선택
2. 백업 정보
   - 백업 볼륨의 이름을 명시한다
   - 백업 레벨을 선택한다. Full backup을 받은 적이 없는 경우 level 0만 표시된다.
   - 백업 경로를 선택한다.
   - 백업 작업을 수행할 thread개수를 설정한다.
   - 데이터베이스 일관성 검사(DB이상 유무 확인)
   - 보관로그 파일 지움(archive log 정리)
   - 압축 백업 사용(백업 파일을 압축하여 저장



백업 - 계속

- CUBRID Manager를 이용한 백업 자동화 설정





1. 백업 정보
   - 백업 ID : 임의 값 설정
   - 백업 level 선택 : 처음 백업 시 level 0만 가능
   - 백업 경로 : 별도의 Disk 또는 media 권장
2. 백업 주기
   - 백업 주기 : 데이터 량에 따라 설정
   - 상세 주기 : 백업주기에 따라 날짜, 요일 등
   - 백업 시간 : DB활동이 가장 적은 시기(시, 분)
3. 옵션
   - 이전 백업 파일 보존 : 이전 단계 복구가 필요할 경우, - 보관 로그 볼륨 로그 삭제: archive log 정리 여부(권장)
   - 백업 후 데이터베이스 통계 정보 갱신 : DB에 갱신 작업이 빈번하게 발생하는 경우 주기적인 작업이 필요
   - 데이터베이스 일관성 검사 : 백업 시 DB 이상 유무를 확인, - 압축 백업 사용 : 사용할 것을 권장
   - 병렬 백업 : 백업 작업에 이용될 thread 수
4. 온/오프라인 백업 : 오프라인 백업 시 DB가 정지 된 후 백업을 진행하고 DB가 구동됨.

복구

- 명령어를 이용한 복구: restoredb

```
% cubrid restoredb   [options]   database_name
```

옵션	인자	설명	기본값
-l	복구 레벨	복구 레벨을 지정한다(0, 1, 2)	0
-d	복구 시점	복구할 시점을 지정 형식) dd-mm-yyyy:hh:mm:ss 예) 21-12-2002:17:00:10. 또는 backuptime : 마지막 백업 시점으로 복구시 사용	가장 최근 시점
-B	파일 패스	복구할 백업 볼륨이 존재하는 디렉토리나 장치명을 지정	초기 로그 볼륨의 위치
-p		archive 로그가 없을 경우 무시하고 수행하여 사용자 인터페이스를 받지 않을 경우	
-u		데이터베이스 위치 파일내에 지정된 경로로 데이터베이스와 로그 볼륨을 복구 (databases.txt)	
--list		백업을 수행하지 않고 백업 볼륨 목록을 출력할 경우	

복구 - 계속

- 데이터 복구
  - 서비스 종료
  - 백업 시점으로의 복구 및 백업 이후 원하는 시점을 복구 가능
    - 원하는 시점으로 복구

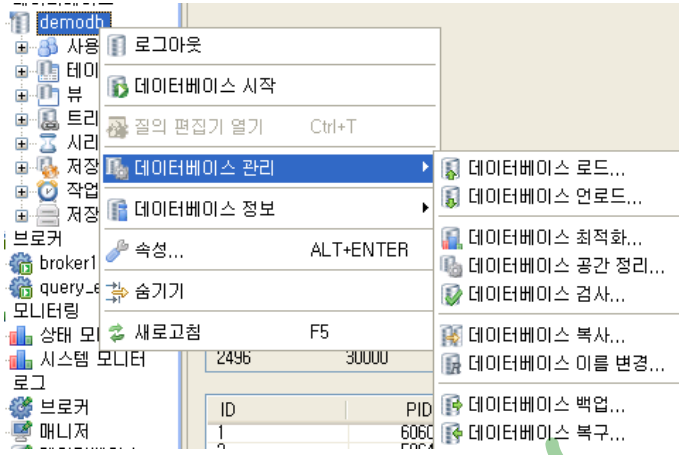
```
$ cubrid restoredb -d 25/12/2011:13:30:00 demodb
```

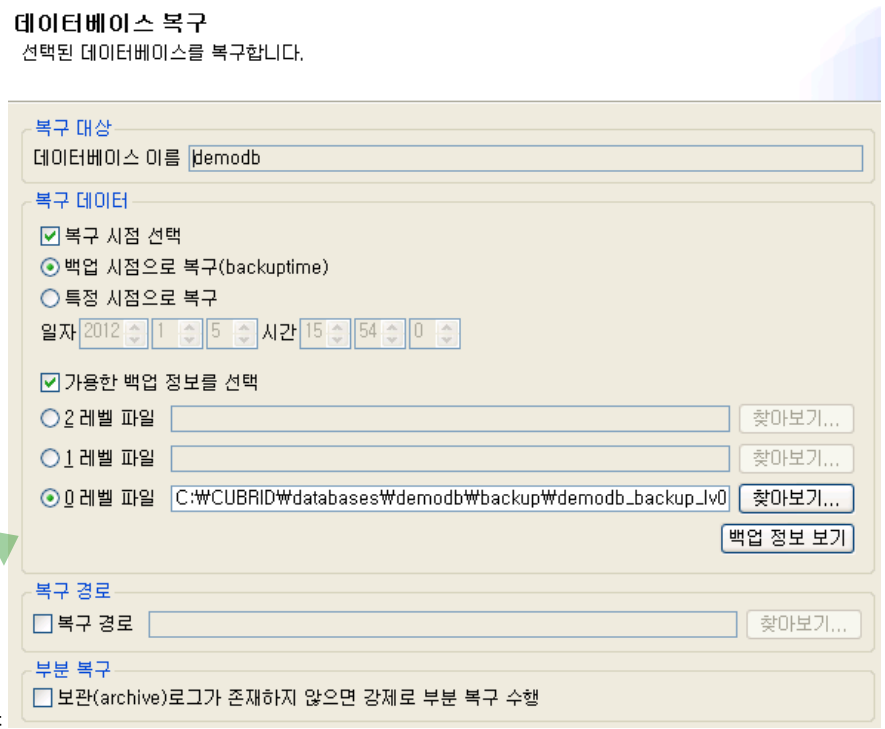
    - 백업시점으로 복구
      - 로그를 다른 곳으로 옮긴 후 복구

```
$ cubrid restoredb demodb
```

복구 - 계속

- CUBRID Manager를 이용한 데이터 복구
  - 서비스 종료
  - 백업 시점으로의 복구 및 백업 이후 원하는 시점을 복구 가능





1. 복구 대상 데이터베이스 이름
2. 복구 데이터
   - 복구 시점 선택 : 백업 시점 또는 특정 시점
   - 가용 가능한 백업 정보 선택 : <db_name>_bkvinf 파일에 백업파일의 경로가 등록되어 있을 경우 자동으로 검색 된다.
3. 복구 경로 : 백업을 복구할 위치를 지정할 수 있다.
4. 부분 복구 : archive log 부재 시 백업 볼륨만 가지고 복구

데이터베이스 복구 사례

- 데이터베이스 복구사례
  - 트랜잭션이 비정상 종료되어 로그가 손상된 경우
    - 비정상트랜잭션 회복하거나 로그복구 → CUBRID Manager에서 지원하지 않는다.
  - 디스크의 media failure가 발생할 경우
  - 데이터베이스를 특정 시점으로 복원하여야 할 경우
    - 백업된 데이터베이스와 로그를 이용해서 복구한다. 이때 필요한 볼륨은 백업 볼륨과 archive log, active log이다.

- 로그 복구
  - 비정상 종료 시 손상된 로그 복구
  - 복구 시 시간이 많이 걸릴 수 있으며, 강제 종료 시 손상이 더 심해질 수 있다.

```
$ csql -S -u dba demodb
```

  - 위의 명령 수행 중 log 관련 에러 발생으로 실패 시 다음 명령 수행

```
$ cubrid emergency_patchlog demodb
```

  - 위 명령어 수행 후 csql 접속 시 이상 없으면 복구 완료, 여전히 에러 발생하면 다음 명령 수행

```
$ cubrid emergency_patchlog -r demodb
```

  - 위 명령어 수행 후 csql 접속 시 이상 없으면 복구 완료, 여전히 에러 발생하면 “백업“ 을 이용한 복구 수행.

실습 4

백업 및 복구 실습

6. 데이터베이스 재구성

데이터베이스 재구성

- 데이터베이스 재구성의 이유
  - 데이터베이스 업그레이드 또는 장애 복구 시 깨끗한 상태로 재구성을 원하는 경우
  - 버전이 달라도 재구성 가능

- 데이터베이스 재구성 방법
  - 언로드 : 원본 데이터베이스를 파일로 내려 받기
  - 백업 : 작업 오류를 대비해 backupdb/copydb/renamedb를 사용한 백업
  - Rebuilding : 기존의 데이터베이스 삭제 및 새로운 데이터베이스 생성
  - 로드 : 언로드 받은 파일을 새로운 데이터베이스에 적재

데이터베이스 재구성 - 계속

- 데이터베이스 재구성을 위한 유틸리티 : 언로드, 로드, 이름변경

- 데이터베이스 언로드
  - 현재의 데이터베이스를 파일로 내려 받는다.
  - 특정 테이블에 대하여 백업을 하고 싶은 경우
  - 데이터베이스의 전체 스키마를 확인할 경우
  - 생성파일
    - 스키마 파일 : 데이터베이스의 스키마 정의 포함하는 파일(demodb_schema)
    - 객체 파일 : 데이터베이스의 데이터를 포함한 파일 (demodb_objects)
    - 인덱스 파일 : 데이터베이스에 정의된 인덱스를 포함하는 파일 (demodb_indexes)
    - 트리거 파일 : 데이터베이스에 정의된 트리거를 포함하는 파일 (demodb_triger)

- 데이터베이스 로드
  - 언로드 받은 파일들을 데이터베이스로 등록
    - 언로드 형식으로 작성된 파일도 수행 가능하며 해당 테이블에 대하여 권한이 있는 계정으로 수행
  - stand-alone 모드에서 DBA 권한으로 수행
    - 언로드한 데이터를 로딩하는 것이므로 각종 시스템 테이블이 포함되어 있어 dba 권한으로 수행
  - 언로드된 데이터량을 확인 하고 적정 크기의 데이터베이스가 존재하여야 함.
  - 기존 데이터가 존재하는 데이터베이스에 로드작업을 할 경우 로드될 데이터가 적합한지 확인 한다.

- 데이터베이스 이름변경
  - 오류 발생 시 앞선 DB를 바로 사용하기 위해 DB를 삭제하지 않고 이름 변경만 해 둔다.(또는 offline상태 백업)
  - 백업/복구 과정보다 빠르게 이전 DB를 백업 백업 해 놓을 수 있다.

CUBRID

데이터베이스 재구성 - 계속

- 데이터베이스 재구성 유틸리티 언로드
- 명령어 사용 : unloaddb

```
% cubrid unloaddb   [options]   database_name
```

옵션	옵션전체	설명	기본값
-i	--input-class-file	인수로 지정된 입력 파일에 지정된 클래스를 대상으로 데이터베이스를 언로드한다.	
-o	--output-path	스키마와 객체 파일이 생성될 디렉터리를 지정한다. 옵션이 지정되지 않으면 현재 디렉터리에 생성된다	현재 위치
-s	--schema-only	데이터 파일은 생성하지 않고, 스키마 파일만 생성한다.	
-d	--data-only	스키마 파일은 생성하지 않고, 데이터 파일만 생성한다.	
-v	--verbose	언로드되는 데이터베이스의 상세 정보를 화면에 출력한다.	수행 않음
-S	--SA-mode	독립 모드에서 데이터베이스를 언로드한다.	
-C	--CS-mode	클라이언트/서버 모드에서 데이터베이스를 언로드한다.	
	--include-reference	-i 옵션과 함께 사용되며, 객체 참조도 함께 생성한다.	
	--input-class-only	-i 옵션과 함께 사용되며, 입력 파일에 포함된 테이블에 관한 스키마 파일만 생성한다.	
	--output-prefix	스키마와 객체 파일명 앞에 붙이는 prefix를 지정한다.	

데이터베이스 재구성 - 계속

- 데이터베이스 재구성 유틸리티 로드
- 명령어 사용 : loaddb

```
% cubrid loaddb  [options]  database_name
```

옵션	옵션전체	설명	기본값
-u	--user	데이터베이스 사용자의 계정을 입력한다.	public
-p	--password	데이터베이스 사용자의 암호를 입력한다.	
-l	--load-only	객체 파일에 포함된 구문과 데이터 타입 검사를 생략하고 레코드를 로드한다.	구문 및 타입 검사 진행
-v	--verbose	데이터 로딩 상태에 관한 상세 정보를 화면에 출력한다.	
-c	--periodic-commit	지정된 개수의 레코드가 입력될 때마다 트랜잭션을 커밋한다.	수행 않음
-s	--schema-file	언로드 작업에 의해 생성된 스키마 파일을 지정하여, 스키마 로딩을 수행한다.	
-i	--index-file	언로드 작업에 의해 생성된 인덱스 파일을 지정하여, 인덱스 로딩을 수행한다.	
-d	--data-file	언로드 작업에 의해 생성된 데이터 파일을 지정하여, 레코드 로딩을 수행한다.	
	--no-statistics	데이터베이스에 관한 통계 정보를 갱신하지 않는다.	
	--ignore-class-file	지정된 파일에 포함된 클래스를 제외하고 로딩 작업을 수행한다.	

데이터베이스 재구성 - 계속

- 데이터베이스 재구성 유틸리티 이름변경
- 명령어 사용 : renamedb

% cubrid renamedb [OPTION] src-database-name dest-database-name

% cubrid renamedb demodb_bk demodb_rename

➔ 원본 edudb를 경로변경 없이 demodb_rename으로 이름 변경

옵션	인자	설명	기본값
-E	volext_path	target database 확장 볼륨 경로 지정	DB home directory
-I	vol_tofrom_path	각 볼륨에 대해 복사할 경로 파일로 지정	
-d	vol_tofrom_path	백업 볼륨과 백업 정보 파일 제거	제거 안함

CUBRID

데이터베이스 재구성 - 계속

- 재구성 절차
  - 서비스 종료
  - 데이터베이스 백업
    - 백업, 데이터베이스 복사, 데이터베이스 이름 변경 중 선택. 이름 변경이 제일 빠름

```
% cubrid renamedb demodb demodb.bak
```

  - 스키마/데이터 내려 받기
    - 스키마 파일 : <데이터베이스 이름>_schema
    - 데이터 파일 : <데이터베이스 이름>_objects
    - 인덱스 파일 : <데이터베이스 이름>_indexes
    - 트리거 파일 : <데이터베이스 이름>_trigger

```
% cubrid unloaddb -S -O /data/unload --output-prefix=demodb demodb.bak
```

  - 데이터베이스 생성
    - 데이터베이스 용량을 재 산정하여 데이터베이스 생성
  - 스키마 / 데이터 올리기
    - 앞서 내려받은 스키마/데이터 등을 생성한 데이터베이스에 올림

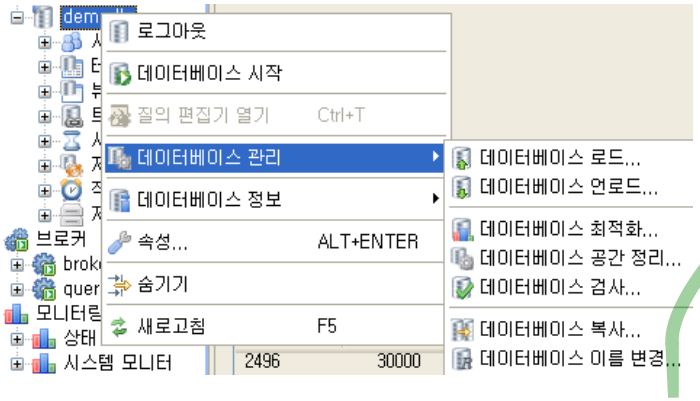
```
% cubrid loaddb -s /data/unload/demodb_schema demodb
# --no-oid : 개체개념을 사용하지 않았거나, foreign key option 에 on cache object 미사용시에만 사용
# 인덱스가 없다면 --no-statistics 옵션 제거
% cubrid loaddb --no-oid --no-statistics -v -c 10000 -d /data/unload/demodb_objects demodb
% cubrid loaddb -v -i /data/unload/demodb_indexes demodb
% cubrid loaddb -v-s /data/unload/demodb_trigger demodb
```

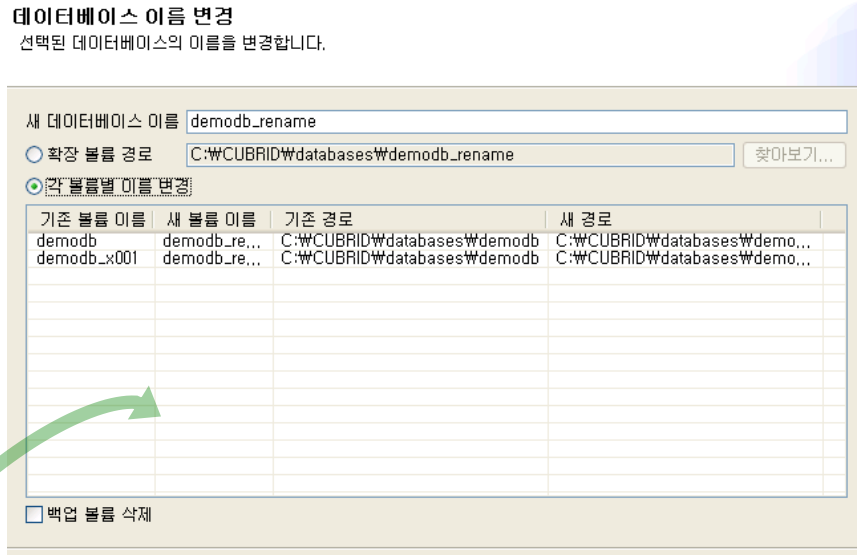
데이터베이스 재구성 - 계속

- CUBRID Manager에서 재구성 하기(이름변경)
  - stand-alone 모드에서 동작

1. 새 데이터베이스 명
   - 변경할 데이터베이스 이름을 명시
2. 옵션
   - 백업볼륨 삭제 : 기존의 백업 삭제 유무
   - 확장볼륨 경로
     추가된 볼륨의 경로를 변경할 경우 명시
   - 각 볼륨 별 이름변경
     볼륨들에 대하여 볼륨의 이름 및 경로를 변경

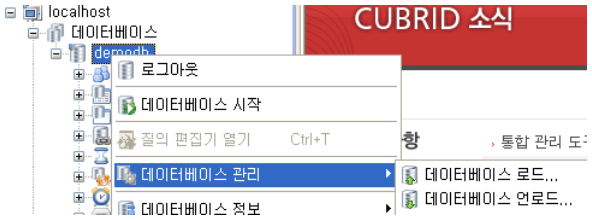
❖ 데이터베이스 이동 및 복사와 다르게 Log볼륨에 대한 경로를 지정 할 수 없다.





데이터베이스 재구성 - 계속

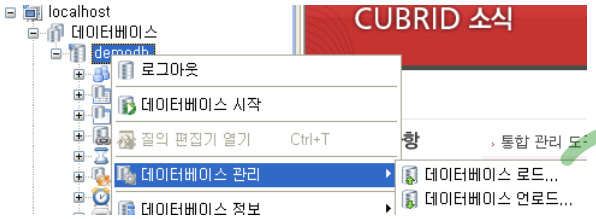
- CUBRID Manager에서 재구성 하기(언로드)

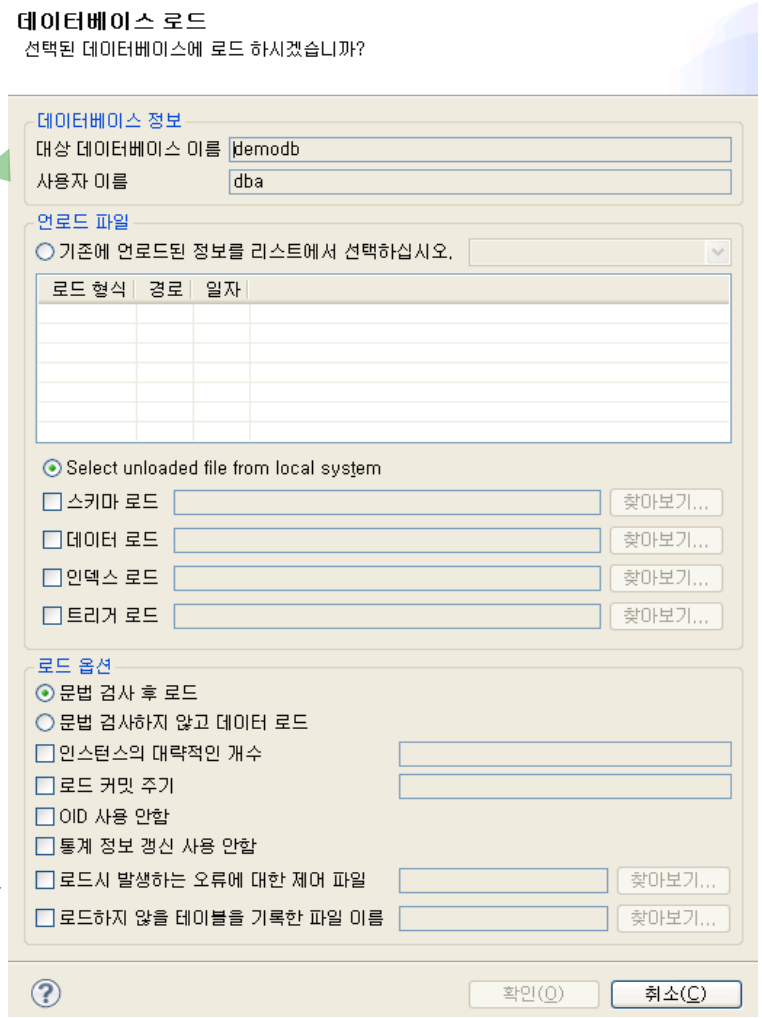


1. 데이터베이스 정보
   - 대상 DB이름
   - 언로드 파일 저장 경로
2. 언로드 대상
   - 스키마 : 전체, 선택, 미선택으로 구분 가능
   - 데이터 : 선택, 미선택으로 구분.
   - 아래 체크 박스로 선택이 가능함.
3. 언로드 옵션
   - 식별자 처음과 끝에 “”사용 : 속성이름을 “”로 묶어서 출력
   - 참조테이블 포함 : 선택된 테이블만 받을 경우 활성화
   - 출력 파일의 접두어 : 결과 파일의 접두어 설정
   - 해쉬 파일 이름 : unload작업 공간에 대한 이름 설정여부
   - 캐시 페이지 개수 : 메모리에 캐시되는 테이블의 수 지정
   - 인스턴스의 대략적인 개수 : 예상되는 레코드 수 지정
   - 하나의 디렉토리에 저장할 LO파일 개수: 개수 명시

데이터베이스 재구성 - 계속

- CUBRID Manager에서 재구성 하기(로드)





1. 데이터베이스 정보
   - 대상 DB이름
   - 사용자 이름
2. 언로드 파일
   - 내부에 받아 놓은 언로드 파일의 경로 선택
3. 로드 옵션
   - 문법 검사 후 로드 : 언로드 받은 파일에 대한 문법 검사
   - 문법 검사하지 않고 데이터 로드 : 검사 시간을 단축한다
   - 인스턴스의 대략적인 개수 : 예상되는 레코드 개수
   - 로드 커밋 주기 : 주기적인 커밋이 성능 향상에 필요하다
   - OID사용 안함 : OID 체크 작업을 생략한다.
   - 통계 정보 갱신 사용 안함 : 작업 시간 단축, 차후 작업 가능
   - 로드시 발생하는 오류에 대한 제어 파일 : 특정 에러를 처리하는 방식에 대한 명세 파일 지정
   - 로드하지 않을 테이블 기록 파일 : 명시된 테이블 작업 생략

데이터베이스 복사 및 이동

- 데이터베이스 재구성 시 백업 방법으로 복사 방법을 활용 가능

- 명령어 사용 : copydb

```
% cubrid copydb   [OPTION]   src-database-name   dest-database-name
% cubrid copydb -F C:₩CUBRID₩databases₩demodb_bk
                -E C:₩CUBRID₩databases₩demodb_bk
                -L C:₩CUBRID₩databases₩demodb_bk
                 demodb demodb_bk
```

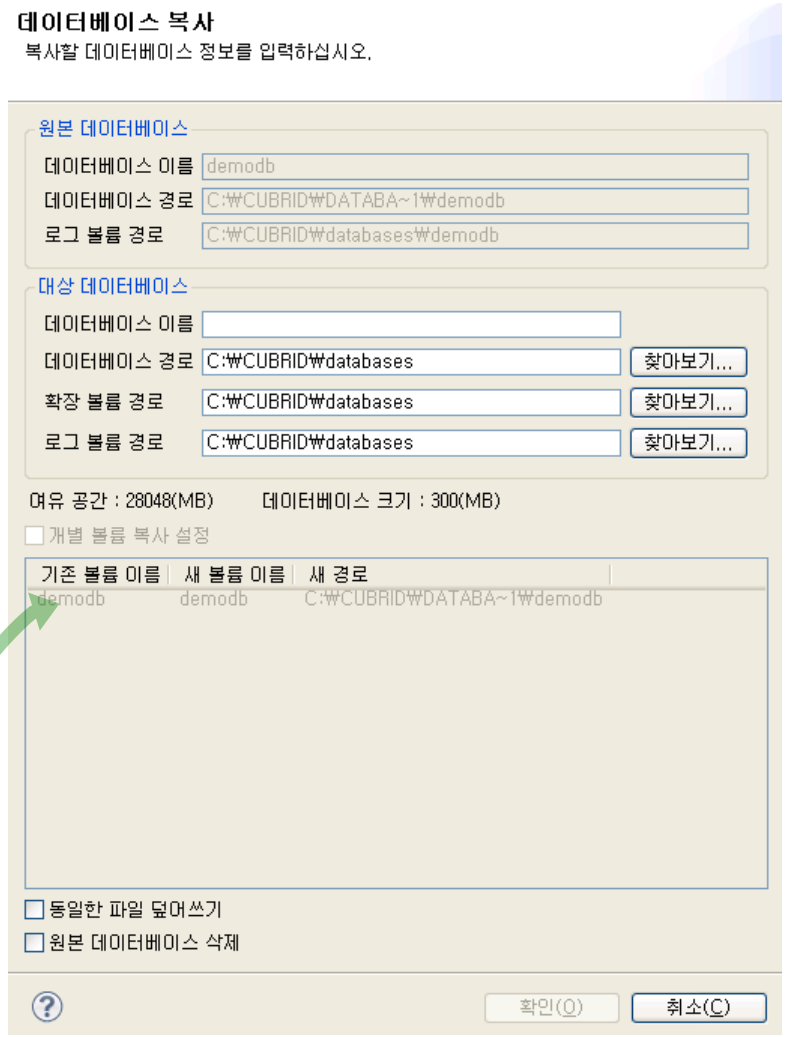
원본 demodb를 "C:₩CUBRID₩databases₩demodb_bk" 경로에 로그 및 확장볼륨 포함하여 demodb_bk로 복사

옵션	인자	설명	기본값
-F	file_path	초기 데이터베이스 볼륨 디렉토리 패스	현재의 디렉토리
-L	log_file_path	데이터베이스 로그 볼륨 디렉토리 패스	초기 볼륨 디렉토리
-E	volext_path	데이터베이스 확장 볼륨 디렉토리 지정	초기 볼륨 디렉토리
-i	to_from_path	각 볼륨에 대해 복사할 경로를 지정한 파일 -E, -F옵션과 같이 사용할 수 없음	없음
-r		target 데이터베이스와 같은 것이 있으면 삭제	수행 않음
-d		복사 후 source 데이터베이스 삭제	수행 않음

데이터베이스 복사 및 이동 - 계속

- 현재의 데이터베이스를 다른 경로에 복사
- stand-alone 모드에서 동작
- 로그볼륨 및 확장볼륨의 경로 지정 가능
- 사용자 계정 및 암호 동일하게 복사

1. 원본 데이터 베이스
   - 복사될 이름 및 경로 표시
2. 대상 데이터베이스
   - 원본으로부터 복사할 데이터베이스 이름/경로 명시
   - 확장된 볼륨에 대한 경로 명시
   - 로그 볼륨의 경로 명시
3. 옵션
   - 개별 볼륨 복사에 대한 설정

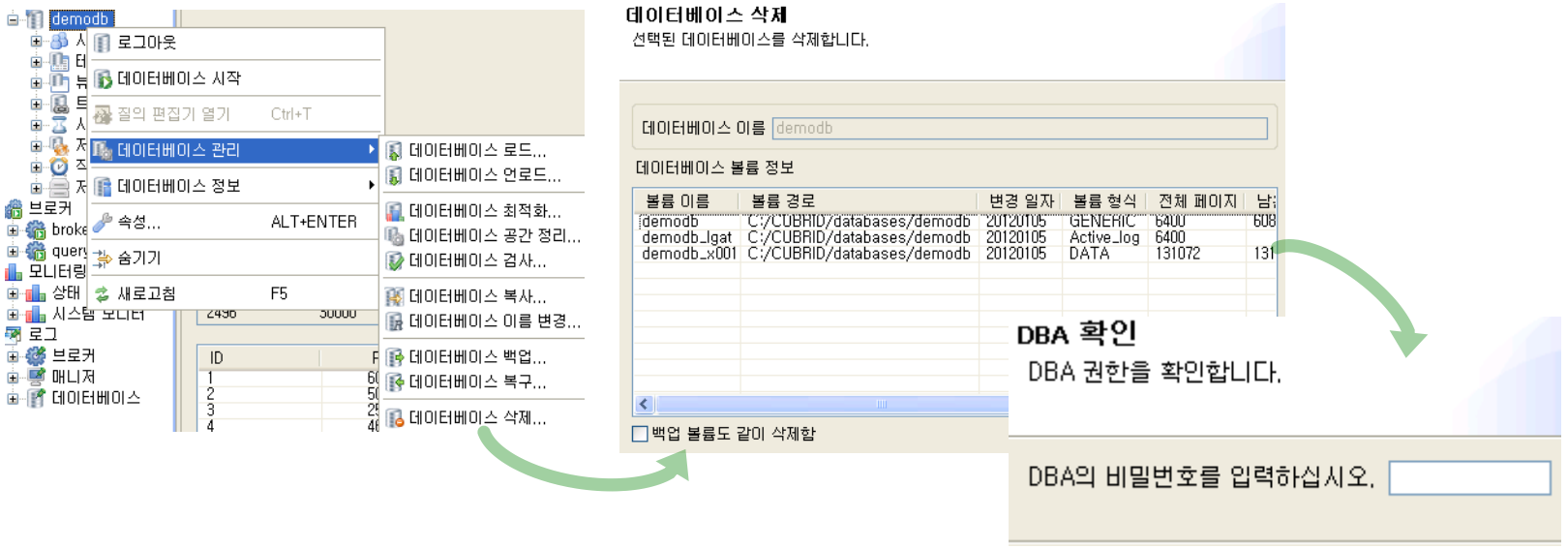


데이터베이스 삭제

- 명령어 사용 : deletedb

```
% cubrid deletedb   [OPTION]   database-name
% cubrid  deletedb -d edudb_rename
➔ 백업볼륨을 포함하여 삭제
```

- stand-alone 모드에서 동작
- 해당 데이터베이스와 관련된 모든 파일을 삭제
  - 백업 볼륨은 옵션을 부여하여 삭제



실습 5

재구성 작업 실습

7. 모니터링

서비스 프로세스 상태 확인

- 서비스 프로세스 상태 확인
  - 구동 중인 서비스(master, server, broker, manager) 상태 출력

```
[cubrid@localhost ~]$ cubrid service status
@ cubrid master status
++ cubrid master is running.
@ cubrid server status
 Server edudb (rel 8.4, pid 25007)
@ cubrid broker status
  NAME          PID  PORT  AS  JQ     REQ  TPS  QPS    LONG-T    LONG-Q  ERR-Q
==========================================================================
* query_editor 25453 40000   5   0       0    0    0    0/60.0    0/60.0     0
* broker1      25463 44000   5   0       0    0    0    0/60.0    0/60.0     0
@ cubrid manager server status
++ cubrid manager server is running.
```

  - Master status : 하나의 engine당 하나의 master process만 존재한다.
  - Server status : 생성되어 구동된 수 만큼의 DB가 보여진다.
  - Broker status : 등록되어 구동된 Broker의 수 만큼 보여진다.
  - Manager server status : Cubrid manager Client가 접속하여 관리가 가능하도록 해주는 Manager Server 구동 여부로써 not running일 경우 CUBRID Manager client는 접속 할 수 없다.(Broker가 구동되어 있을 시 Query Browser는 접속 가능하다)

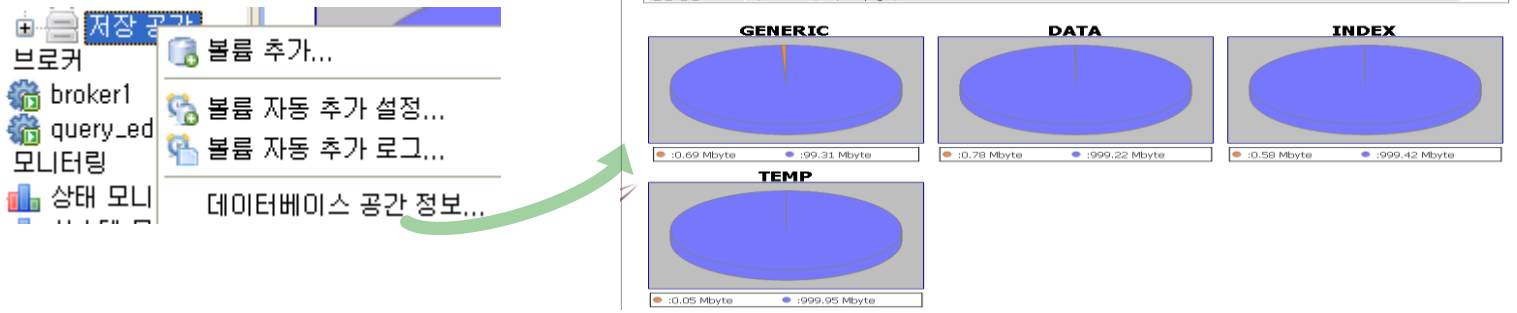
CUBRID

데이터베이스 사용량

- 사용량 확인
  - 각 볼륨별 사용량 확인
- 명령어 확인

```
$ cubrid spacedb edudb ;# 서버가 종료되었을 때는 -S 옵션 추가
Space description for database 'edudb' with pagesize 16.0K. (log pagesize: 16.0K)
Volid  Purpose    total_size    free_size  Vol Name
    0  GENERIC       100.0 M       98.1 M  /data/edudb
    1     DATA         2.0 G         2.0 G  /data/edudb_x001
    2    INDEX         1.0 G      1024.0 M  /data/edudb_x002
    3     TEMP         1.0 G      1024.0 M  /data/edudb_x003
--------------------------------------------------------------------------------
    4                  4.1 G         4.1 G
Space description for temporary volumes for database 'edudb' with pagesize 16.0K.
Volid  Purpose    total_size    free_size  Vol Name
```

- CUBRID Manager를 통한 확인



CUBRID

Broker 상태 확인

- 명령어 사용 : broker status
  - Broker monitoring 명령어
  - 여러 옵션을 같이 사용할 수 있다.

```
% cubrid broker status [options]
```

옵션	설명
broker_name	지정한 브로커에 관한 상태 정보를 출력한다. 옵션이 지정되지 않으면 전체 브로커의 상태 정보를 출력한다
-b	응용 서버에 관한 정보는 포함되지 않고, 브로커에 관한 상태 정보만 출력한다.
-q	작업 큐에 대기 중인 작업을 출력한다
-s secs	브로커에 관한 상태 정보를 지정된 시간마다 주기적으로 출력한다. q를 입력하면 명령 프롬프트로 복귀한다.
-t	화면 출력시 tty mode 로 출력한다. 출력 내용을 리다이렉션하여 파일로 쓸 수 있다.
-f	브로커가 접속한 DB 및 호스트 정보를 출력한다.

Broker 상태 확인 - 계속

- broker 상태 확인
  - % query_editor : 브로커의 이름
  - [4772,30000] : [PID, Broker 접속 포트 번호]
  - JOB QUEUE : 작업 큐에 대기 중인 작업 개수
  - LONG_TRANSACTION_TIME : 장기 실행(Long-duration) 트랜잭션으로 판단하는 트랜잭션의 실행 시간(초단위)
  - LONG_QUERY_TIME : 장기 실행 질의(Long-duration query)으로 판단하는 질의의 실행 시간(초단위)
  - SESSTION_TIMEOUT : 세션 타임 아웃값(초단위)
  - KEEP_CONNECTION : 응용 서버와 클라이언트의 연결(default auto)
  - ACCESS_MODE : 브로커의 동작 모드

```
%cubrid broker status -s 1 query
@ cubrid broker status
% query_editor  - cub_cas [4772,30000] /CUBRID/log/broker//query_editor.access
 /CUBRID/log/broker//query_editor.err
 JOB QUEUE:0, AUTO_ADD_APPL_SERVER:ON, SQL_LOG_MODE:ALL:100000
 LONG_TRANSACTION_TIME:60.00, LONG_QUERY_TIME:60.00, SESSION_TIMEOUT:300
 KEEP_CONNECTION:AUTO, ACCESS_MODE:RW
----------------------------------------
ID   PID   QPS   LQS PSIZE STATUS
----------------------------------------
 1  2685    9    0   52388 IDLE
 2  2686   12    0   56124 BUSY
 3  2687    0    0   48780 CLENT_WAIT
 4  2688    0    0   48780 CLOSE_WAIT
 5  2689    0    0   48780 IDLE
```

Broker 상태 확인 - 계속

- broker 상태 확인
  - ID : 브로커 ID 또는 작업 큐에 대기 중인 작업 ID
  - PID : 브로커의 프로세스 ID
  - QPS : 초당 처리된 질의의 수
  - LQS : 초당 처리되는 장기 실행 질의의 수
  - PSIZE : 응용 서버 프로세스 크기
  - STATUS : 응용 서버의 현재 상태
    - IDLE → 연결되어 있지 않으며, 아무 작업도 수행 중이지 않은 상태
    - BUSY → 질의수행 중이거나, 응용에서 요청한 작업을 처리중인 상태
    - CLIENT_WAIT → 요청한 작업은 완료되었으나 트랜잭션이 끝나지 않은 상태
    - CLOSE_WAIT → 트랜잭션이 완료되었으며 연결은 유지되어 있는 상태

```
%cubrid broker status -s 1 query
@ cubrid broker status
% query_editor  - cub_cas [4772,30000] /CUBRID/log/broker//query_editor.access
 /CUBRID/log/broker//query_editor.err
 JOB QUEUE:0, AUTO_ADD_APPL_SERVER:ON, SQL_LOG_MODE:ALL:100000
 LONG_TRANSACTION_TIME:60.00, LONG_QUERY_TIME:60.00, SESSION_TIMEOUT:300
 KEEP_CONNECTION:AUTO, ACCESS_MODE:RW
------------------------------------------
ID   PID   QPS   LQS PSIZE    STATUS
------------------------------------------
 1  2685    9    0   52388 IDLE
 2  2686   12    0   56124 BUSY
 3  2687    0    0   48780 CLENT_WAIT
 4  2688    0    0   48780 CLOSE_WAIT
 5  2689    0    0   48780 IDLE
```

Broker 상태 확인 - 계속

- broker 단위 별 상태 확인 (cubrid broker status 명령어 사용 시 -b 옵션 추가)
  - NAME : 브로커 이름
  - PID : 브로커의 프로세스 ID
  - PORT : 브로커의 포트 번호
  - AS : 응용 서버 개수
  - JQ : 작업 큐에서 대기 중인 작업 개수
  - REQ : 브로커가 처리한 클라이언트 요청 개수
  - TPS : 초당 처리된 트랜잭션의 수(옵션이 " -b -s < sec > " 일 때만 계산됨)
  - QPS : 초당 처리된 질의의 수(옵션이 " -b -s < sec > " 일 때만 계산됨)
  - LONG-T : LONG_TRANSACTION_TIME 시간을 초과한 트랜잭션 수 / 파라미터 설정 값
  - LONG-Q : LONG_QUERY_TIME 시간을 초과한 질의의 수 / LONG_QUERY_TIME 파라미터의 값
  - ERR-Q : 에러가 발생한 질의의 수

```
%cubrid broker status -b -s 1 query
@ cubrid broker status
 NAME        PID  PORT  AS  JQ    REQ  TPS  QPS   LONG-T   LONG-Q  ERR-Q
==========================================================================
* query_editor  2684 37000   5   0     46   29   21    0/60.0    0/60.0     0
* broker1       2694 38000   5   0      0    0    0    0/60.0    0/60.0     0
```

Broker 상태 확인 - 계속

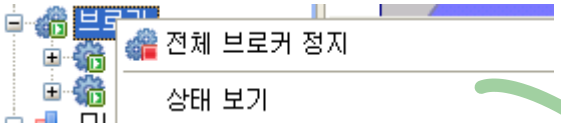
- broker 접속 정보 및 시간 확인 (cubrid broker status 명령어 사용 시 -f 옵션 추가)
  - LAST ACCESS TIME : Broker가 연결한 최근 접속 시간
  - DB : 현재 접속 하는 DB이름
  - HOST : DB가 설치된 HOST
  - LAST CONNECT TIME : Client로 부터 Broker에 접속한 최초 시간
  - CLIENT IP : 클라이언트의 IP주소

```
%cubrid broker status -f -s 1 query
@ cubrid broker status
% query_editor  - cub_cas [2684,37000] /home/vzkim/CUBRID/log/broker//query_editor.access
/home/vzkim/CUBRID/log/broker//query_editor.err
 JOB QUEUE:0, AUTO_ADD_APPL_SERVER:ON, SQL_LOG_MODE:ALL:100000, SLOW_LOG:ON
 LONG_TRANSACTION_TIME:60.00, LONG_QUERY_TIME:60.00, SESSION_TIMEOUT:300
 KEEP_CONNECTION:AUTO, ACCESS_MODE:RW
------------------------------------------------------------------------------------------------------------------------------------
ID   PID   QPS   LQS PSIZE STATUS       LAST ACCESS TIME          DB         HOST     LAST CONNECT TIME
CLIENT IP    SQL_LOG_MODE
------------------------------------------------------------------------------------------------------------------------------------
 1  2685    9    0 52388 IDLE       2011/12/01 18:07:10          edudb      localhost   2011/11/29 13:58:58
10.0.1.101
 2  2686   12    0 56124 IDLE       2011/12/01 18:07:10          edudb      localhost   2011/11/29 13:58:58
-
 3  2687    0    0 48780 IDLE       2011/11/29 13:53:52          -          -           -                   -
 4  2688    0    0 48780 IDLE       2011/11/29 13:53:52          -          -           -                   -
 5  2689    0    0 48780 IDLE       2011/11/29 13:53:52          -          -           -                   -
```


CUBRID

Broker 상태 확인 - 계속

- CUBRID Manager에서 Broker상태 확인
  - CUBRID Manager Client에서는 아래와 같은 형식으로 Broker 상태를 확인 할 수 있다.
  - 각 메뉴 값은 command 명령어에 의해 보여졌던 결과 값과 동일하다.



브로커 상태 - query_editor@localhost:8001

PID	PORT	JOB QUEUE	AUTO ADD A...	SQL LOG MO...	SESSION TIM...	KEEP CONN...	ACCESS MODE
2496	30000	0	ON	ALL	300	AUTO	RW

ID	PID	QPS	LQS	PSIZE	STATUS	DB
1	6060	0	0	2368	IDLE	
2	5064	0	0	2368	IDLE	
3	2524	0	0	2368	IDLE	
4	4624	0	0	2368	IDLE	
5	4416	0	0	2368	IDLE	

작업 큐 상태

ID	PRIORITY	ADDRESS	TIME	REQ

Broker SQL 로그 확인

- 수행된 트랜잭션 기반으로 SQL 별 로그 기록
  - SQL별, 트랜잭션 별 수행시간 기록
  - SQL 로그 분석을 통한 처리 성능 분석
  - $CUBRID/log/broker/sql_log/<broker 이름>_<cas ID>.sql.log

```
02/04 13:45:17.687 (38) prepare 0 insert into unique_tbl values (1)
02/04 13:45:17.687 (38) prepare srv_h_id 1
02/04 13:45:17.687 (38) execute srv_h_id 1 insert into unique_tbl values (1)
02/04 13:45:17.687 (38) execute error:-670 tuple 0 time 0.000, EID = 39
02/04 13:45:17.687 (0) auto_rollback
02/04 13:45:17.687 (0) auto_rollback 0
*** 0.000

02/04 13:45:17.687 (39) prepare 0 select * from unique_tbl
02/04 13:45:17.687 (39) prepare srv_h_id 1 (PC)
02/04 13:45:17.687 (39) execute srv_h_id 1 select * from unique_tbl
02/04 13:45:17.687 (39) execute 0 tuple 1 time 0.000
02/04 13:45:17.687 (0) auto_commit
02/04 13:45:17.687 (0) auto_commit 0
*** 0.000
```

- 응용 클라이언트의 요청 시각
- (39) : SQL 문 그룹의 시퀀스 번호
  prepared statement pooling일 경우
- prepare 0 : prepared statement인지 여부
- (PC) : 플랜 캐시에 저장되어 있는 내용을 사용
- SELECT... : 실행 SQL 문.
  - Statement pooling한 경우,
    WHERE 절의 binding 변수가 ?로 표시된다.
- Execute 0 tuple 1 time 0.000
  - 1개의 row가 실행되고, 소요 시간은 0.000초
- auto_commit/auto_rollback
  - 자동으로 커밋 되거나, 롤백 되는 것을 의미
  - 두 번째 auto_commit/auto_rollback은
    에러 코드이며, 0은 에러가 없이 트랜잭션이 완료

CUBRID

에러 로그

- 데이터베이스 서비스 중 또는 구동/종료 등 DB 작동 관련 발생한 서버 에러 로그 기록
  - 장애 원인 파악
  - $CUBRID/log/server/<db 이름>_<yyyymmdd날짜>_<server ID>.err

```
Time: 11/29/11 13:54:06.263 - ERROR *** ERROR CODE = -116, Tran = 0
Database “edudb" is unknown, or the file "databases.txt" cannot be accessed.
```

- SQL 수행중 발생한 Broker 에러 로그 기록
  - SQL 로그와 같이 사용하면 오류 파악이 용이하다.
  - $CUBRID/log/broker/error_log/<broker 이름>_<cas ID>.err

```
Time: 02/04/10 13:45:17.687 - SYNTAX ERROR *** ERROR CODE = -493, Tran = 1, EID = 38
Syntax: Unknown class "unknown_tbl". select * from unknown_tbl
```

실습 6

수행 환경에 대한 모니터링 실습

8. 환경 설정

환경파일

- 데이터베이스 동작 관련 설정
  - $CUBRID/conf/cubrid.conf,
  - 대/소문자 구분 없음

```
[common]
# 모든 데이터베이스 공통 설정
[@edudb]
# edudb 만을 위한 설정
```

- cubrid.conf
  - CUBRID 시스템 파라미터의 설정 값을 저장하는 파일이다.
  - 위치는 $CUBRID/conf 아래 존재하며, 데이터베이스 별로 다를 수 있는 내용들은 데이터베이스 별로 별도로 지정하는 것이 좋다.

```
[commom]
max_clients=50
[@demodb]
max_clients=100
```

  - 서버 파라미터와 클라이언트 파라미터로 나뉘며 파라미터변경을 할 경우 해당하는 프로세스를 재 구동하여야 한다.
    - SQL을 사용하여 클라이언트 파라미터 변경 시 재 구동이 필요 없다.
  - 파라미터 설정 구문 규칙
    - 대/소문자를 구분 없다.
    - 파라미터 이름과 설정값은 동일한 라인에 입력되어야 한다.
    - 등호 기호(=)를 사용하며, 양 옆에는 공백문자를 사용할 수 있다.
    - 파라미터 값이 문자열인 경우 따옴표 없이 문자열만 입력 만, 해당 문자열에 공백 문자가 포함된 경우에는 따옴표를 사용한다.

환경파일 - 계속

- cubrid.conf의 설정 우선순위 적용
  - 환경변수에 설정할 때는 파라미터 앞에 CUBRID_를 붙여 설정

```
set CUBRID_SORT_BUFFER_PAGE=512
```

  - SQL문을 이용한 설정
    - 클라이언트 파라미터만 설정가능
    - 복수 설정을 할 경우 “;”를 이용

```
SET SYSTEM PARAMETERS 'parameter_name=value [{; name=value}...]'
SET SYSTEM PARAMETERS 'csql_history_num=70'
SET SYSTEM PARAMETERS 'csql_history_num=70; index_scan_in_oid_order=1'
```

  - CSQL 인터프리터를 이용한 설정변경
    - 클라이언트 파라미터만 설정가능

```
csql> ;se block_ddl_statement=1
=== Set Param Input ===
block_ddl_statement=1
```

데이터베이스 환경 설정

- data_buffer_size : 512M
  - 데이터베이스 서버가 메모리 내에 캐시하고 있는 데이터 캐쉬 크기
  - 실제 데이터베이스의 크기 및 실제 메모리의 크기, 기타 다른 프로세스의 개수 및 크기를 고려해서 할당해야 한다.
  - 이 값이 클수록 메모리에 많은 데이터가 캐시되므로 disk I/O는 줄일 수 있지만, 너무 크면 페이지 버퍼 풀 스와핑 발생 한다.
- sort_buffer_size : 2M
  - 정렬을 필요로 하는 질의를 처리할 때 사용되는 버퍼 크기로 generic 볼륨이나 temp 볼륨을 이용한다.
  - Active client request 마다 하나의 sort buffer가 할당된다.
  - 정렬이 끝나면 할당되었던 메모리는 해제된다.
- temp_file_memory_size_in_pages : 4
  - 질의에 관한 임시 결과를 캐시하는 버퍼 페이지 개수를 설정. 최대 20
- lock_timeout_in_secs : -1
  - 잠금 대기 시간을 지정하며, 기본값은 -1로 무제한 대기한다. 0이면 대기하지 않음
  - 시간 이내에 잠금이 허용되지 않으면 해당 트랜잭션이 취소되고 오류가 반환
- single_byte_compare : no
  - 문자열 비교시 byte 단위로 수행하며, UNICODE 사용시 yes 설정. 기본은 no(한글 비교를 위함)
- max_client : 100
  - 데이터베이스 서버에 접속가능한 client의 수. 최대 1024

데이터베이스 환경 설정 - 계속

- isolation_level
  - 트랜잭션의 고립수준을 1에서 6까지의 정수값 또는 문자 스트링으로 설정
  - SERIALIZABLE: 트랜잭션이 끝날 때 까지 접근 불가
  - REPEATABLE: SELECT에서 S_LOCK이 트랜잭션 종료될 때 까지 계속 유지
  - READ UNCOMMITTED: 완료되지 않은 트랜잭션에 Read를 허용

설정값	설 명
TRAN_SERIALIZABLE(6)	SERIALIZABLE
TRAN_REP_CLASS_REP_INSTANCE(5)	REPEATABLE READ CLASS with REPEATABLE READ INSTANCES
TRAN_REP_CLASS_COMMIT_INSTANCE TRAN_READ_COMMITTED(4)	REPEATABLE READ CLASS with READ COMMITTED INSTANCES
TRAN_READ_UNCOMMITTED TRAN_REP_CLASS_UNCOMMIT_INSTANCE(3)	REPEATABLE READ CLASS with READ UNCOMMITTED INSTANCES
TRAN_COMMIT_CLASS_COMMIT_INSTANCE(2)	READ COMMITTED CLASS with READ COMMITTED INSTANCES
TRAN_COMMIT_CLASS_UNCOMMIT_INSTANCE(1)	READ COMMITTED CLASS with READ UNCOMMITTED INSTANCES

데이터베이스 환경 설정 - 계속

- block_ddl_statement
  - 데이터 정의문(Data Definition Language, DDL)을 제한
  - default값은 no로 설정하지 않음
- block_nowhere_statement
  - UPDATE/DELETE문에 조건(Where)이 없는 경우 질의를 수행하지 않음
  - default값은 no로 설정하지 않음
- intl_mbs_support
  - 스키마의 멀티바이트 문자 세트의 지원 여부를 지정(defaut no 설정하지 않음)
  - 연산 비용이 크므로, 테이블 이름이나 칼럼 이름을 영어로 사용할 것을 권장
- oracle_style_empty_string
  - yes로 설정할 경우 빈 문자열(empty string)을 NULL로 처리(default no)
- 통신 서비스 관련
  - cubrid_port_id
    - Master Process Port
    - default 값은 1523
    - 1523이 이미 사용 중일 경우 이 파라미터를 다른 번호로 변경해야 한다.
- 서버 재 구동 설정
  - auto_restart_server
    - 장애로 인하여 서버가 종료된 경우 자동으로 재 구동
    - default는 yes로 자동 재 구동된다

CUBRID

broker 환경 설정

- 환경 설정 변경
  - 설정 파일 : $CUBRID/conf/cubrid_broker.conf
  - 편집기를 통하여 수정가능하며, Broker 재 구동될 경우 설정 적용
  - 재 구동 없이 설정 변경할 경우 명령어를 이용하여 변경

```
% broker_changer  <br-name>  <conf-name>  <conf-value>
% broker_changer broker1 sql_log on
OK
```

- 설정 가능 환경변수
  - APPL_SERVER_MAX_SIZE, SQL_LOG, TIME_TO_KILL
  - KEEP_CONNECTION, ACCESS_MODE, ACCESS_LOG
  - LOG_BACKUP, SQL_LOG_MAX_SIZE, STATEMENT_POOLING
- 환경변수 및 설정 값이 잘못되어 있을 경우 재 구동 시 오류가 발생하며 구동되지 않는다.

CUBRID

broker 환경 설정 - 계속

- MIN_NUM_APPL_SERVER : 5
  - cas의 최소개수. broker 구동시 구동되는 cas 의 수.
  - connection pool 사용시 pool 개수와 동일하게하거나, 근사하게 설정
- MAX_NUM_APPL_SERVER : 40
  - cas의 최대 개수. 동시에 처리가능한 응용 요청의 수와 관련.
  - MIN_NUM_APPL_SERVER 의 수를 넘어 추가된 cas 는 일정시간 요청이 없으면 종료됨.
- TIME_TO_KILL : 120
  - MIN_NUM_APPL_SERVER 의 수를 넘어 추가된 cas 가 처리 요청없이 구동되어 있는 시간 (초).
- SESSION_TIMEOUT : 300
  - 잠금 대기 시간을 지정하며, -1은 무제한 대기, 0이면 대기하지 않음
  - 시간 이내에 잠금이 허용되지 않으면 해당 트랜잭션이 취소되고 오류가 반환
- MAX_STRING_LENGTH : -1
  - 문자형 데이터 타입에 대한 최대 허용 문자길이를 설정한다.100으로 설정되어 있는 상태에서 입력길이가 1,000일 경우 첫 100byte만 허용 하고 이하 버림을 수행한다.
- SELECT_AUTO_COMMIT : OFF
  - PHP(CCI포함)에서 SELECT문에 대한 자동 commit 모드를 수행
- LONG_QUERY_TIME : 60(sec)
  - 장기 실행 질의(Long-duration query)로 판단될 질의 실행 시간을 설정값이 0일 경우 수행하지 않음
- LONG_TRANSACTION_TIME : 60(sec)
  - 장기 실행 트랜잭션(Long-duration transaction)으로 판단될 트랜잭션의 실행 시간을 설정값이 0일 경우 수행하지 않음

CUBRID

broker 추가

- 응용서버그룹(broker) 추가
  - 특정 업무로의 사용을 위해서, 혹은 다른 database 로의 서비스를 위해서 broker 를 추가할 수 있다.
  - 특히 CUBRID BROKER는 connection pooling 개념을 사용하므로 여러 개의 database 를 대상으로 서비스를 한다면 database 별로 별로의 broker 를 사용하여야 connection에 따른 overhead 를 줄일 수 있다.
    - 추가 후 재 구동하여 추가된 브로커를 구동시킬 수 있다.
  - broker 추가를 위해서는 $CUBRID/conf/cubrid_broker.conf 을 직접 편집한다.
    - 기존 내용을 그대로 복사하여 마지막에 추가한다.
    - Broker 이름을 변경한다. BROKER_PORT 와 PPL_SERVER_SHM_ID 를 변경한다

```
# 기존 내용
[%BROKER1]
SERVICE                 =ON
BROKER_PORT             =33000
MIN_NUM_APPL_SERVER     =5
MAX_NUM_APPL_SERVER     =40
APPL_SERVER_SHM_ID      =33000
APPL_SERVER_MAX_SIZE    =20
LOG_DIR                 =log/broker/sql_log
ERROR_LOG_DIR           =log/broker/error_log
SQL_LOG                 =ON
TIME_TO_KILL            =120
SESSION_TIMEOUT         =300
KEEP_CONNECTION         =AUTO
```

```
# 추가할 내용(반드시 수정해야할 부분)
[%MY_BROKER]
SERVICE                 =ON
BROKER_PORT             =40000
MIN_NUM_APPL_SERVER     =5
MAX_NUM_APPL_SERVER     =40
APPL_SERVER_SHM_ID      =40000
APPL_SERVER_MAX_SIZE    =20
LOG_DIR                 =log/broker/sql_log
ERROR_LOG_DIR           =log/broker/error_log
SQL_LOG                 =ON
TIME_TO_KILL            =120
SESSION_TIMEOUT         =300
KEEP_CONNECTION         =AUTO
```

실습 7

데이터베이스 서버 설정

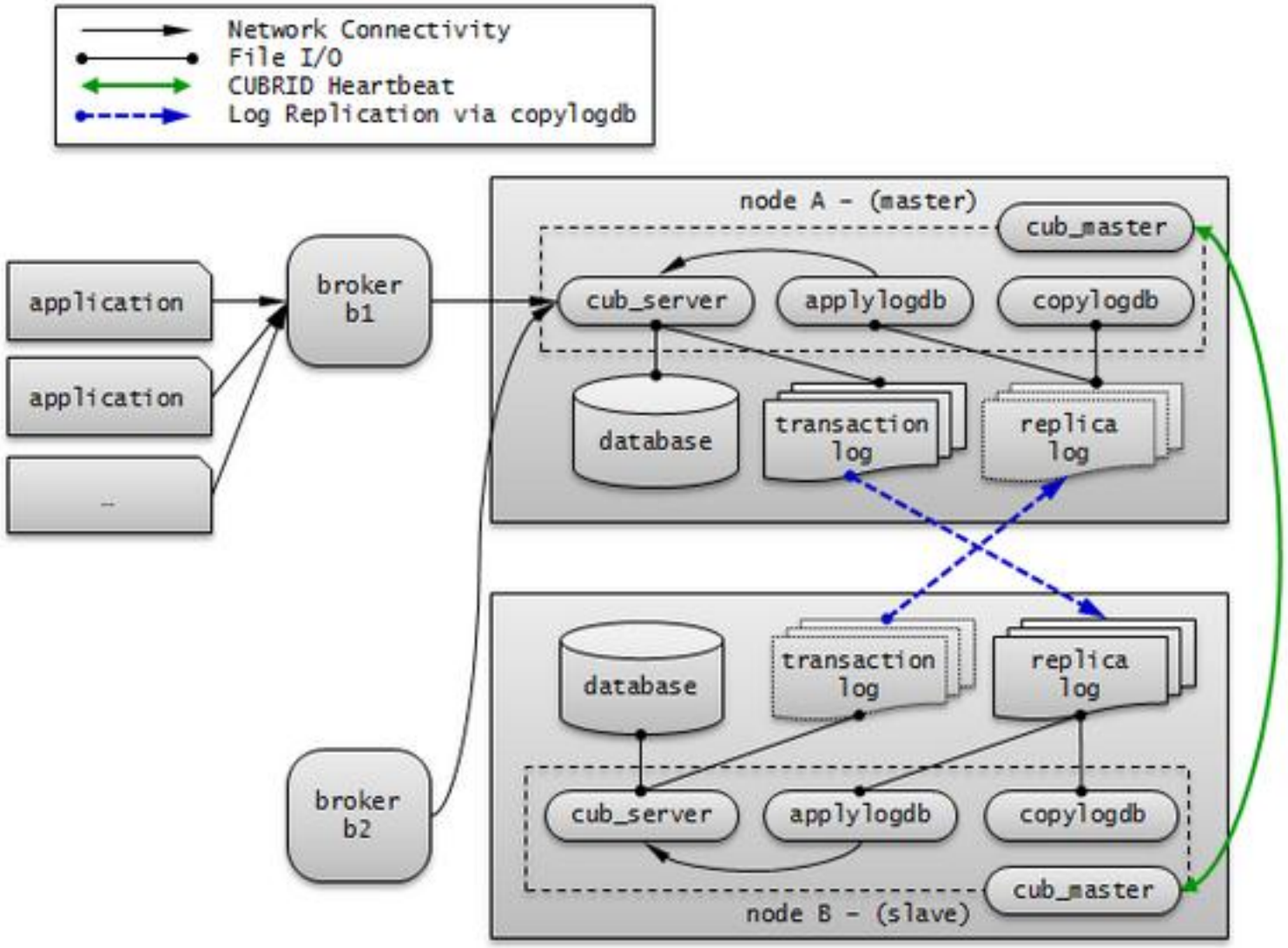
브로커 설정

9. CUBRID HA

CUBRID HA구조

- 개요
  - 여러 서버 시스템에서 데이터베이스를 항상 동기화된 상태로 유지하여 서비스를 제공한다.
  - 서비스를 수행 중인 시스템에 예상치 못한 장애가 발생하면 자동으로 다른 시스템이 서비스를 수행하도록 하여 서비스 중단 시간을 최소화한다.
  - CUBRID의 HA 기능은 shared-nothing 구조이며, 액티브 서버(active server)에서 스탠바이 서버(standby server)로 데이터를 동기화하기 위해 다음 두 단계를 수행한다.
    - 트랜잭션 로그 다중화: 액티브 서버에서 생성되는 트랜잭션 로그를 실시간으로 다른 노드에 복제한다.
    - 트랜잭션 로그 반영: 실시간으로 복제된 트랜잭션 로그를 분석하여 스탠바이 서버에 데이터를 반영한다.
  - 시스템과 CUBRID의 상태를 실시간으로 감시하고 장애가 발생하면 자동 failover를 수행하기 위해 heartbeat 메시지를 사용한다.

CUBRID HA구조도



CUBRID

CUBRID HA 기본개념

- 노드와 그룹
  - 노드는 CUBRID HA를 구성하는 논리적인 단위이다.
    - 마스터 노드: 복제의 대상이 되는 노드로, 액티브 서버를 사용한 읽기, 쓰기 등 모든 서비스를 제공한다.
    - 슬레이브 노드: 마스터 노드와 동일한 내용을 갖는 노드로, 마스터 노드의 변경이 자동으로 반영된다. 스탠바이 서버를 사용한 읽기 서비스를 제공하며 마스터 노드 장애 시 failover가 일어난다.
    - 레플리카 노드: 마스터 노드와 동일한 내용을 갖는 노드로, 마스터 노드의 변경이 자동으로 반영된다. 스탠바이 서버를 사용한 읽기 서비스를 제공하며 마스터 노드 장애 시 failover가 일어나지 않는다.

- 프로세스
  - CUBRID HA 노드는 하나의 마스터 프로세스(cub_master), 하나 이상의 데이터베이스 서버 프로세스(cub_server), 하나 이상의 복제 로그 복사 프로세스(copylogdb), 하나 이상의 복제 로그 반영 프로세스(applylogdb)로 이루어져 있다.
  - 하나의 데이터베이스를 설정하면 데이터베이스 서버 프로세스, 복제 로그 복사 프로세스, 복제 로그 반영 프로세스가 구동된다.
    - 마스터 프로세스(cub_master): heartbeat 메시지를 주고 받으며 CUBRID HA 내부 관리 프로세스들을 제어한다.
    - 데이터베이스 서버 프로세스(cub_server): 사용자에게 읽기, 쓰기 등의 서비스를 제공한다.
    - 복제 로그 복사 프로세스(copylogdb): 그룹 내의 모든 트랜잭션 로그를 복사한다.
    - 복제 로그 반영 프로세스(applylogdb): 복제 로그 복사 프로세스에 의해 복사된 로그를 노드에 반영한다.

- 서버
  - 서버란 데이터베이스 서버 프로세스를 논리적으로 표현하는 단어로, 상태에 따라 액티브 서버(active server), 스탠바이 서버(standby server)로 나눈다.
    - 액티브 서버: 마스터 노드에 속하는 서버로, active 상태이다. 액티브 서버는 사용자에게 읽기, 쓰기 등 모든 서비스를 제공한다.
    - 스탠바이 서버: 마스터 노드 외의 노드에 속하는 서버로, standby 상태이다. 스탠바이 서버는 사용자에게 읽기 서비스만을 제공한다.

CUBRID HA 기본개념 - 계속

- Heartbeat 메시지
  - HA 기능을 제공하기 위한 핵심 구성 요소로, 마스터 노드, 슬레이브 노드, 레플리카 노드가 다른 노드의 상태를 감시하기 위해 주고 받는 메시지이다.

- failover와 failback
  - failover란, 마스터 노드에 장애가 발생하여 서비스를 제공할 수 없는 상태가 되면 우선순위가 가장 높은 슬레이브 노드가 자동으로 마스터 노드가 되는 것이다.
  - failback은 마스터 노드였던 장애 노드가 복구되면 자동으로 다시 마스터 노드가 되는 것이며, 현재 CUBRID HA는 서버의 failback을 지원하지 않는다.

- Broker 모드
  - Broker는 서버에 Read Write, Read Only, Slave Only, Preferred Host Read Only 네 가지 모드 중 한 가지로 접속할 수 있으며, 사용자가 Broker 모드를 설정할 수 있다.
  - databases.txt에 설정된 호스트에 순차적으로 연결을 시도한다.
    - Read Write : 읽기, 쓰기 서비스를 제공하는 Broker이다. 이 Broker는 일반적으로 액티브 서버에 연결하며, 연결 가능한 액티브 서버가 없으면 스탠바이 서버에 연결한다.
    - Read Only : 읽기 서비스를 제공하는 Broker이다. 이 Broker는 가능한 스탠바이 서버에 연결하며, 스탠바이 서버가 없으면 액티브 서버에 연결한다.
    - Slave Only : 읽기 서비스를 제공하는 Broker이다. 이 Broker는 스탠바이 서버에 연결하며, 스탠바이 서버가 없으면 서비스를 제공하지 않는다.
    - Preferred Host Read Only : 읽기 서비스를 제공하는 Broker이다. Read Only Broker와 동일하고, 서버의 접속 순서 및 서버 선정 기준은 PREFERRED_HOSTS로 설정할 수 있다.

CUBRID HA 환경설정

- cubrid.conf
  - $CUBRID/conf 디렉터리에 위치
  - ha_mode : CUBRID HA 기능을 설정하는 파라미터이다. 기본값은 off이다.
    - off : CUBIRD HA 기능을 사용하지 않는다.
    - on : CUBRID HA 기능을 사용하며, 해당 노드는 failover의 대상이 된다. 마스터, 슬레이브 노드에 설정.
    - replica : CUBRID HA 기능을 사용하며, 해당 노드는 failover의 대상이 되지 않는다. 레플리카 노드에 설정.
  - log_max_archives : 보존할 보관 로그 파일의 최소 개수를 설정하는 파라미터이다. CUBRID가 설치된 영역의 가용 디스크 공간과 백업 정책(1 level 백업의 경우 보관 로그 파일 삭제 불가), 1일 보관 로그 생성 개수에 따라 적절하게 적용
  - force_remove_log_archives : log_max_archives로 지정한 개수의 최근 보관 로그(log archive) 파일을 제외한 나머지 파일의 삭제 허용 여부를 지정하는 파라미터로서, ha_mode가 on인 경우 no로 설정하여 HA 관련 프로세스에 필요한 보관 로그 파일이 삭제되지 않도록 설정한다.
  - 설정 예시

```
ha_mode=on
log_max_archives=20
force_remove_log_archives=no
```

CUBRID HA 환경설정 - 계속

- cubrid_ha.conf
  - $CUBRID/conf 디렉터리에 위치, CUBRID의 HA의 전반적인 설정 정보를 담고 있다.
  - ha_node_list : CUBRID HA 그룹 내에서 사용할 그룹 이름과 failover의 대상이 되는 멤버 노드들의 호스트 이름을 명시한다.
    - 그룹 이름과 멤버 노드들의 이름은 @ 구분자로 나눈다. 기본값은 localhost@localhost이다.
    - 이 파라미터에서 명시한 멤버 노드들의 호스트 이름 및 해당 노드의 호스트 이름은 반드시 /etc/hosts에 등록되어 있어야 한다.
  - ha_replica_list : CUBRID HA 그룹 내에서 사용할 그룹 이름과 failover의 대상이 되지 않는 멤버 노드들의 호스트 이름을 명시한다.
    - 그룹 이름과 멤버 노드들의 이름은 @ 구분자로 나눈다. 기본값은 NULL이다.
    - 그룹 이름은 ha_node_list에서 명시한 이름과 같아야 한다. 이 파라미터에서 명시한 멤버 노드들의 호스트 이름 및 해당 노드의 호스트 이름은 반드시 /etc/hosts에 등록되어 있어야 한다
  - 설정 예시

```
[common]
ha_node_list=foo@nodeA:nodeB
ha_replica_list=foo@nodeC:nodeD
ha_db_list=testdb
```

CUBRID HA 환경설정 - 계속

- cubrid_broker.conf
  - $CUBRID/conf 디렉터리에 위치
  - ACCESS_MODE : Broker의 모드를 설정한다. 기본값은 RW이다.
    - RW(Read Write), RO(Read Only), SO(Slave Only), PHRO(Preferred Host Read Only)를 값으로 설정할 수 있다.
  - PREFERRED_HOSTS : Broker 모드를 Preferred Host Read Only로 설정하면 반드시 설정해야 하는 파라미터이다. 기본값은 NULL이다.
    - 여러 노드를 지정할 수 있으며 콜론(:)으로 구분한다.
  - 설정 예시

```
[%broker1]
... ...
ACCESS_MODE             =RW
PREFERRED_HOSTS         =nodeA:nodeB:nodeC
```

- databases.txt
  - $CUBRID_DATABASES 디렉터리에 위치하며, db_hosts 값을 설정하여 Broker가 접속하는 서버의 순서를 결정할 수 있다. 여러 노드를 설정하려면 콜론(:)으로 구분한다.
  - 설정예시

```
#db-name    vol-path                db-host                log-path                        lob-base-path
testdb      /home/cubrid/DB/testdb  nodeA:nodeB:nodeC      /home/cubrid/DB/testdb/log      file:/home/cubrid/DB/testdb/lob
```

CUBRID HA 환경설정 - 계속

- jdbc 설정
  - JDBC에서 CUBRID HA 기능을 사용하려면 Broker에 장애가 발생했을 때 연결할 Broker의 연결 정보를 연결 URL에 추가로 지정해야 한다. CUBRID HA를 위해 지정되는 속성은 장애가 발생했을 때 연결할 하나 이상의 Broker 노드 정보인 althosts와 Broker의 장애 복구 후 재연결을 시도하는 주기인 rctime이다.
  - 설정 예시

```
Connection connection =
DriverManager.getConnection("jdbc:cubrid:primary_broker:33000:testdb:::?charset=utf-8&althosts=secondary_broker:33000&rctime=600", "dba", "");
```

CUBRID HA 구성형태

- HA 기본 구성
  - 하나의 마스터 노드와 하나의 슬레이브 노드로 구성된다.
  - CUBRID HA 고유의 기능인 장애 시 무중단(nonstop) 서비스 기능에 초점을 맞춘 구성.
  - 설정 예시
    - nodeA(마스터 노드)
      - cubrid.conf 파일의 ha_mode를 on으로 설정한다

```
ha_mode=on
log_max_archives=20
force_remove_log_archives=no
```

      - cubrid_ha.conf 파일의 설정 예

```
ha_port_id=59901
ha_node_list=cubrid@nodeA:nodeB
ha_db_list=testdb
```

      - databases.txt 파일의 설정 예

```
#db-name     vol-path                  db-host       log-path
testdb       /home/cubrid/DB/testdb    nodeA:nodeB   /home/cubrid/DB/testdb/log
```

      - nodeB(슬레이브 노드): nodeA와 동일하게 설정한다.
      - JDBC 설정

```
Connection connection =
DriverManager.getConnection("jdbc:cubrid:nodeA:33000:testdb:::?charset=utf-
8&althosts=nodeB:33000&rctime=600", "dba", "");
```

CUBRID HA 구성형태 - 계속

- 부하 분산 구성
  - HA 구성(한 개의 마스터 노드와 한 개의 슬레이브 노드)에 여러 개의 레플리카 노드를 두어 CUBRID 서비스의 가용성을 높이고, 많은 읽기 부하를 분산하여 처리할 수 있는 구성이다.
  - 마스터 노드와 슬레이브 노드의 Broker는 HA 구성에 포함되도록 RW Broker와 RO Broker를 생성하여야 하고, 레플리카 노드에는 RO Broker를 생성한다.
  - 레플리카 노드는 HA 구성에 포함되지 않으므로 HA 구성 내의 모든 노드에 장애가 발생해도 failover되지 않고 읽기 서비스만 제공한다.
  - Read Write 요청은 Master Node에서 처리하고 Read Only요청을 Slave 및 Replica Node에서 처리하도록 응용에서 질의 요청시 선택해주도록 개발하여야 한다.
  - 설정 예시
    - nodeA(마스터 노드)
      - cubrid.conf 파일의 ha_mode를 on으로 설정한다.

```
ha_mode=on
log_max_archives=20
force_remove_log_archives=no
```

      - cubrid_ha.conf 파일의 설정 예

```
ha_port_id=59901
ha_node_list=cubrid@nodeA:nodeB
ha_replica_list=cubrid@nodeC:nodeD
ha_db_list=testdb
```

      - databases.txt 파일의 설정 예

```
#db-name     vol-path                db-host       log-path
testdb       /home/cubrid/DB/testdb  nodeA:nodeB   /home/cubrid/DB/testdb/log
```

CUBRID HA 구성형태 - 계속

- cubrid_broker.conf 파일의 설정 예

```
[%broker1]
BROKER_PORT             =33000
... ...
ACCESS_MODE              =RW
[%broker2]
BROKER_PORT             =34000
... ...
ACCESS_MODE              =RO
```

- nodeB(슬레이브 노드)
  - cubrid.conf, cubrid_ha.conf, cubrid_broker.conf는 nodeA와 동일하게 설정한다
  - Databases.txt 파일의 설정 예

```
#db-name     vol-path                  db-host        log-path
testdb       /home/cubrid/DB/testdb    nodeB:nodeA    /home/cubrid/DB/testdb/log
```

CUBRID HA 구성형태 - 계속

- nodeC(레플리카 노드)
  - cubrid.conf 파일의 ha_mode를 replica로 설정한다.

```
ha_mode=replica
log_max_archives=20
force_remove_log_archives=no
```

  - cubrid_ha.conf 파일은 node A와 동일하게 설정한다.
  - databases.txt 파일의 설정 예

```
#db-name     vol-path                  db-host        log-path
testdb       /home/cubrid/DB/testdb    nodeC:nodeD    /home/cubrid/DB/testdb/log
```

  - cubrid_broker.conf 파일의 설정 예

```
[%broker2]
BROKER_PORT             =34000
... ...
ACCESS_MODE             =RO
```

- nodeD(레플리카 노드)
  - cubrid.conf 파일은 node C와 동일하게 설정한다.
  - cubrid_ha.conf 파일은 node A와 동일하게 설정한다.
  - databases.txt 파일의 설정 예

```
#db-name     vol-path                  db-host        log-path
testdb       /home/cubrid/DB/testdb    nodeD:nodeC    /home/cubrid/DB/testdb/log
```

CUBRID HA 구성형태 - 계속

– cubrid_broker.conf 파일의 설정 예

```
[%broker2]
BROKER_PORT             =34000
... ...
ACCESS_MODE             =RO
```

- JDBC 설정 : Connection URL 값을 부하분산 환경에 맞게 Read Write 요청을 처리하는 Connection과 Read Only 요청을 처리하는 Connection 으로 분리하는데 Read Only Connection은 (Slave node + Replica node) 수 만큼 생성한다. Read Only의 경우 생성되는 각 Connection에 명시되는 각 hostname을 순서대로 반복하여 기입한다.

구분	연결설정 예시
Read Write	jdbc:cubrid:nodeA:33000:testdb:::?charset=utf-8&althosts=nodeB:33000&rctime=600
Read Only 1	jdbc:cubrid:nodeB:34000:testdb:::?charset=utf-8&althosts=nodeC:34000,nodeD:34000&rctime=600
Read Only 2	jdbc:cubrid:nodeC:34000:testdb:::?charset=utf-8&althosts=nodeD:34000,nodeB:34000&rctime=600
Read Only 3	jdbc:cubrid:nodeD:34000:testdb:::?charset=utf-8&althosts=nodeB:34000,nodeC:34000&rctime=600

CUBRID HA 구동 및 모니터링

- 구동
  - 해당 노드의 CUBRID HA 구성 요소(데이터베이스 서버 프로세스, 복제 로그 복사 프로세스, 복제 로그 반영 프로세스)를 모두 구동해야 한다.
  - 구동하는 순서에 따라 마스터 노드와 슬레이브 노드가 결정되므로, 순서를 주의해야 한다.
    - 사용 방법

```
%cubrid heartbeat start
```

  - 위 명령으로 Broker, 매니저 서버는 구동되지 않으므로 구동되어 있지 않은 경우에 별도로 각각 구동해야 한다.
  - CUBRID HA 구성 요소, Broker, 매니저 서버 구동을 함께 수행하고자 하는 경우에는 $CUBRID/conf/cubrid.conf의 service 파라미터 부분에 server를 삭제하고 heartbeat 을 포함해준 후 cubrid service start 명령으로 기동한다.
    - 설정 예시

```
...
# Any combinations are available with server, broker, manager and heartbeat.
service=heartbeat, broker, manager
...
```

    - 사용 방법

```
%cubrid service start
```

CUBRID HA 구동 및 모니터링 - 계속

- 중지
  - 해당 노드의 CUBRID HA 구성 요소(데이터베이스 서버 프로세스, 복제 로그 복사 프로세스, 복제 로그 반영 프로세스)를 모두 종료해야 한다. 이 명령을 실행한 노드는 종료되고 HA 구성에 있는 다음 순위의 슬레이브 노드로 failover가 일어난다.
    - 사용 방법

```
%cubrid heartbeat stop
```

  - 위 명령으로 Broker, 매니저 서버는 종료되지 않으므로 별도로 각각 종료해야 한다.
  - cubrid.conf의 service 파라미터에 heartbeat이 설정되어 있고 CUBRID HA 구성 요소, Broker, 매니저 서버 중지를 함께 수행하고자 하는 경우
    - 사용 방법

```
%cubrid service stop
```

- 모니터링
  - CUBRID HA 그룹 정보와 CUBRID HA 구성 요소의 정보를 확인할 수 있다.
    - 사용 방법

```
%cubrid heartbeat status
```

  - CUBRID HA의 복제 상태를 모니터링.
    - 사용 방법

```
cubrid applyinfo [option] database-name
예) cubrid applyinfo -L /NCIA/engn/cubrid/CUBRID/databases/testdb_master_node -r
master_node -a testdb
```

CUBRID HA 구동 및 모니터링 - 계속

- 옵션설명

옵션	기본값	설명
-r	none	트랜잭션 로그를 복사하는 대상 노드의 이름을 설정한다. 이 옵션을 설정하면 대상 노드의 액티브 로그 정보(Active Info.)를 출력한다.
-a		cubrid applyinfo를 수행한 노드(localhost)의 복제 반영 정보(Applied Info.)를 출력한다. 이 옵션을 사용하기 위해서는 반드시 –L 옵션이 필요하다.
-L	none	상대 노드의 트랜잭션 로그를 복사해 온 위치를 설정한다. 이 옵션이 설정된 경우 상대 노드에서 복사해 온 트랜잭션 로그의 정보(Copied Active Info.)를 출력한다.
-p	0	–L 옵션을 설정한 경우 설정 가능한 것으로 복사해 온 로그의 특정 페이지 정보를 출력한다.
-v		더 자세한 내용을 출력한다.

CUBRID HA 주의사항

- Linux 계열에서만 CUBRID HA 기능을 사용 가능
- 마스터 노드와 슬레이브 노드는 반드시 동일 플랫폼으로 구성
- 기본 키(primary key)가 설정된 테이블에 대해서만 복제 가능
- 다음의 경우 HA 구성에서 마스터 노드 데이터베이스와 슬레이브 노드 데이터베이스간 데이터 불일치 발생 가능
  - 테이블에 trigger 또는 Java Stored Procedure가 설정된 경우 : 슬레이브 노드에서 중복 수행
  - 테이블에 method가 사용된 경우 : 메소드 사용 시 복제 로그가 생성되지 않아 복제되지 않음. 예) add_user()
  - CUBRID 매니저를 이용하여 컬럼에 NOT NULL 옵션을 설정하는 경우 : 복제 로그가 생성되지 않음
  - 독립 실행 모드(standalone)에서 작업이 수행된 경우 : 작업 관련 로그가 슬레이브 노드로 반영되지 않음
  - cubrid backupdb -r 사용 금지 : -r 옵션을 사용하여 로그를 삭제하는 경우 동기화가 완료되지 않은 로그가 삭제되어 불일치 발생할 수 있음.
- INCR/DECR 함수 : 슬레이브 노드에서 사용하지 않도록 주의.
- 분할하여 작업할 수 없는 대량의 데이터 작업은 HA 정지 후 각 노드별로 따로 진행하는 것을 권장. 특히 UPDATE로 특정 컬럼을 모두 변경하는 작업이 있을 경우 HA 정지하고 각 노드에서 데이터 작업을 완료 이후 HA를 구동.
- 스키마 변경(기본키 변경 제외), 인덱스 변경, 권한 변경 작업의 경우 해당 테이블의 레코드 수가 많은 경우 마스터 노드에서 완료 후 슬레이브 노드로 복제로그가 복사되어 슬레이브 노드에 반영됨에 따라 운영 작업이 완료되는데 2배의 시간이 소요됨. 따라서 HA 정지 후 각 노드별로 변경 작업을 진행하는 것을 권장.

실습 8

HA구성 실습

10. Trouble Shooting

Trouble Shooting

- 데이터베이스가 구동이 안 될 경우
  - 환경변수 설정이 제대로 되었는가?
    - env를 실행하여 PATH, CUBRID, CUBRID_DATABASES 등 설정이 정상적인지 확인한다.
  - 볼륨이 손상되지 않았는가?
    - 볼륨이 손상이 되면 백업파일을 이용하여 복구한다.
  - 데이터베이스를 생성한 사용자가 맞는가?
    - $CUBRID_DATABASES/databaes.txt 파일을 열어 해당 데이터베이스 정보가 존재하는 지 확인한다.
  - 다른 사용자가 stand-alone 모드로 데이터베이스를 사용하고 있지 않는가?
    - ps -ef 로 프로세스를 확인한 후 해당 프로세스가 종료되면 DB를 구동한다.
  - 디스크 공간이 부족하지 않는가?
    - df -h 로 CUBRID가 설치된 디스크 및 데이터 볼륨이 존재하는 디스크 공간을 확인하여 부족한 경우 불필요한 파일을 삭제하여 디스크 공간을 확보한다.
  - Log Active 볼륨 파일이 존재하는가?
    - $CUBRID_DATABASES/databases.txt 내용 중 log-path 부분을 확인하고 해당 경로에 <database_name>_lgat 파일이 존재하는 지 확인한다
    - 존재하지 않는 경우 cubrid emergency_patchlog 명령어를 이용해서 임시 로그 생성을 시도하고 실패하는 경우 백업파일을 이용하여 데이터베이스 복구 작업 실행한다.
  - cubrid master start에 실패하는가?
    - cubrid master start: fail 발생하는 경우 %hostname 명령 실행한 결과로 나오는 해당 hostname이 /etc/hosts에 명시되어 있는지 확인한다.

Trouble Shooting - 계속

- 서버를 비정상 종료한 경우
  - 데이터베이스 복구 및 rollback을 위해 stand-alone 모드 명령인 "csql -S <database-name>"을 사용하여 서버 복구 작업을 수행한 이후 "cubrid server start <database-name>" 명령으로 서버를 구동해야 한다. csql을 수행할 때 시간이 오래 걸려도 연결될 때까지 기다려야 한다.

- 데이터베이스 접속이 안 될 경우
  - 서버에 접속되지 않는 경우의 대부분은 데이터베이스 서버나 Broker서비스가 구동 되지 않은 경우가 많다. 이런 경우 서버에서 cubrid server status 명령을 이용하여 데이터베이스 서버가 구동되어있는지를 확인하고 cubrid broker status 명령을 수행하여 Broker 서비스 상태를 확인한다.
  - 서버에 접속되지 않는 다른 경우는 클라이언트가 잘못된 데이터베이스 서버 정보를 가지고 있는 경우이다. 이런 경우는 우선 데이터베이스의 이름, 데이터베이스 서버의 hostname 또는 IP, broker port을 확인하여 클라이언트의 정보를 변경시켜준다.
  - 여기까지 이상이 없는 경우는 WAS서버의 접속 정보를 확인해 본다.

- Log볼륨이 삭제된 경우의 대처 방법은?
  - log active volume 및 log archive volume은 임의로 편집, 삭제 및 이동해서는 안된다. 그러나 이들 로그가 삭제된 경우, 데이터베이스서버를 종료 시킨 후, emergency_patchlog 명령을 이용하여 다음과 같이 임시 로그 생성을 시도한다.
  - cubrid emergency_patchlog -r database-name

Trouble Shooting - 계속

- Log Archive볼륨을 삭제 하여도 괜찮습니까?
  - log archive volume은 log active volume이 일정량 적재되면 생성되는 것으로 데이터베이스에 commit되지 않는 변경 사항까지 기록하고 있기 때문에, media failure가 발생 시 데이터베이스 복구를 가능하게 한다. 정상 수행 중에 발생한 불필요한 archive log는 log 볼륨정보 파일(dbname_lginf)에 기록되지만, 자동으로 삭제되지는 않는다. 이러한 log는 임의로 삭제할 수 없으며 데이터베이스를 백업하는 utility인 backupdb의 -r 옵션을 사용해서 제거해야 한다.
  - 작업도중에 데이터베이스가 다운되었을 경우도 archive log가 생성된다. 이때는 stand-alone 모드로 데이터베이스를 회복(recovery)하면 다운되기 이전의 변경 사항을 rollback하면서 archive log도 삭제된다.
  - CUBRID HA 구성되어 있는 경우에는 HA 관련 프로세스에 필요한 보관 로그 파일이 삭제되지 않도록 하기 위해 cubrid.conf 파일에 force_remove_log_archives=no, log_max_archives=20 과 같이 파라미터를 설정하여 자동 관리되도록 한다.

- 데이터베이스의 Lock을 확인 하는 방법은?
  - cubrid.conf 파일의 isolation_level의 값에 따라 lock의 레벨이 정해진다. 다른 lock과 관련된 파라미터 값들은 lock_escalation, lock_timeout_in_secs, deadlock_detection_interval_in_secs 등 이다. 사용 중인 어플리케이션의 특성에 따라 적절한 locking 정책을 결정해서 각 파라미터 값들을 조정해야 한다.
  - 현재 데이터베이스에 접근하고 있는 클라이언트 어플리케이션들 가운데 lock 충돌과 동시성 제어 문제를 파악하기 위해 lockdb utility를 사용한다. 이 utility의 출력 정보는 서버의 클라이언트 정보, 시스템 클래스, 사용자 정의 클래스, 인스턴스 그리고 페이징 locking이다.
    - cubrid lockdb [option] database_name@localhost

Trouble Shooting - 계속

- 데이터베이스에 실행중인 트랜잭션을 확인하고, 문제를 발생시키는 트랜잭션을 중단하는 방법은?
  - 현재 실행중인 트랜잭션의 확인 및 트랜잭션 중단은 cubrid killtran 유틸리티를 이용합니다. 이 유틸리는 DBA만 수행할 수 있다.

```
$cubrid killtran -u dba testdb@localhost
Tran index      User name      Host name      Process id      Program name
------------------------------------------------------------------------------
     1(+)            dba      myhost             664          cub_cas
     2(+)            dba      myhost            6700            csql
     3(+)            dba      myhost            2188          cub_cas
     4(+)            dba      myhost             696            csql
     5(+)         public      myhost            6944            csql
------------------------------------------------------------------------------
```

  - killtran 유틸리티를 사용해서 testdb의 현재 트랜잭션 정보를 출력한다. 출력된 트랜잭션 중에서 제거하고자 하는 트랙잭션의 index 값을 확인한다.

```
$cubrid killtran -u dba -i 1 testdb
Ready to kill the following transactions:
Tran index      User name      Host name      Process id      Program name
------------------------------------------------------------------------------
     1(+)            dba      myhost            4760            csql
------------------------------------------------------------------------------
Do you wish to proceed ? (Y/N)y
Killing transaction associated with transaction index 1
```

  - 제거할 트랜잭션 1에 -i 옵션을 사용하면 다시 한 번 삭제 여부를 확인한 후에 1번 트랜잭션이 강제로 종료된다.

Trouble Shooting - 계속

- 서비스 지연 현상이 발생하는 경우 확인 방법 및 종료 방법
  - cubrid broker status 를 이용한 현재 서비스 상태 확인한다.

```
$cubrid broker status -f -s 1 broker1
% broker1- cub_cas [4307,33000] /home/cubrid/CUBRID/log/broker//broker1.access /home/cubrid/CUBRID/log/broker//broker1.err
 JOB QUEUE:0, AUTO_ADD_APPL_SERVER:ON, SQL_LOG_MODE:ALL:100000
 LONG_TRANSACTION_TIME:60.00, LONG_QUERY_TIME:60.00, SESSION_TIMEOUT:300
 KEEP_CONNECTION:AUTO, ACCESS_MODE:RW
------------------------------------------------------------------------------------------------------------------------
ID   PID   QPS   LQS PSIZE STATUS       LAST ACCESS TIME            DB          HOST   LAST CONNECT TIME      CLIENT IP
SQL_LOG_MODE
------------------------------------------------------------------------------------------------------------------------
 1  4308     0     0 25956 BUSY   2011/09/25 10:35:02          demodb       localhost 2011/09/25 10:34:56    192.168.32.1
-
SQL:
 2  4309     0     0 24912 IDLE       2011/09/25 10:32:52          -           -           -                  -
 3  4310     0     0 24912 IDLE       2011/09/25 10:32:52          -           -           -                  -
 4  4311     0     0 24912 IDLE       2011/09/25 10:32:52          -           -           -                  -
 5  4312     0     0 24912 IDLE       2011/09/25 10:32:52          -           -           -                  -
```

  - 항목 중 status를 확인하여 현재 상태가 IDLE 또는 CLOSE_WAIT 가 아닌 BUSY, CLIENT_WAIT 이면 현재 사용 중 이라는 뜻이다.
  - busy상태가 길어지고 LAST ACCESS TIME과 현재 시간의 차이가 클 경우 $CUBRID/log/broker/sql_log/에 존재하는 SQL LOG를 확인한다.
  - 해당하는 SQL LOG는 해당 <broker명>_<cas아이디>.sql.log 이며 위 예제로 표현하면 broker1_1.sql.log이다.

Trouble Shooting - 계속

- 서비스 지연 현상이 발생하는 경우 확인 방법 및 종료 방법 -계속

```
$tail -f broker1_1.sql.log
09/25 10:35:02.365 (0) *** elapsed time 0.000
09/25 10:35:02.366 (0) check_cas 0
09/25 10:35:02.367 (0) set_db_parameter lock_timeout 1000
09/25 10:35:02.380 (0) end_tran COMMIT
09/25 10:35:02.381 (0) end_tran 0 time 0.000
09/25 10:35:02.381 (0) *** elapsed time 0.015
09/25 10:35:02.381 (0) END OF LOG
09/25 10:45:05.959 (0) get_db_parameter isolation_level
09/25 10:45:05.998 (1) prepare 2
select * from olympic  where rownum between 1 and 5000;
09/25 10:45:06.324 (1) prepare srv_h_id 1
09/25 10:45:06.408 (1) execute srv_h_id 1 select * from olympic  where rownum between 1 and 5000;
09/25 10:45:06.481 (1) execute 0 tuple 25 time 0.073
09/25 10:45:06.568 (0) end_tran COMMIT
09/25 10:45:06.570 (0) end_tran 0 time 0.002
09/25 10:45:06.571 (0) *** elapsed time 0.614
...
```

  - 위와 같은 진행 상태를 확인 할 수 있으며 어느 부분에서 작업이 지연되고 있는지 파악 가능하다.
  - 해당 작업을 강제 종료하고자 할 때는 cubrid broker status 모니터링 결과 중 STATUS 상태에 따라 작업내용이 달라진다.
    - CLIENT_WAIT : 접속된 응용을 종료 시키고 계속 CLIENT_WAIT이 유지되는 경우 해당 cas의 PID를 확인하여 kill -9 PID 실행으로 종료.
    - BUSY : 해당 cas의 PID를 확인하여 kill -9 PID 실행으로 cas 종료. 이후 cubrid killtran 유틸을 사용하여 killtran 모니터링 결과 중 Process id가 종료 시킨 cas의 PID와 일치하는 트랜잭션의 index를 확인하여 종료 시킨다.

Trouble Shooting - 계속

- 데이터베이스 볼륨 중 임시볼륨의 용도는?
  - 데이터 볼륨의 임시 볼륨은 질의 처리(order by, group by, subquery, join query, create index…) 및 sorting을 위해서 사용된다. 초기 임시 볼륨(permanent temp)이 부족하면, 추가 임시 볼륨(temporary temp)이 생성되는데, 이는 성능 저하를 초래하므로 충분한 양의 임시 볼륨을 미리 할당하는 것이 필요하다. space 제한은 temp_file_max_size_in_pages 파라미터에서 지정한다. 초기 임시 볼륨은 영구적인데 반해, 추가로 생성된 임시 볼륨은 데이터베이스 서버가 종료되거나 시스템의 shutdown, 데이터베이스 서버를 재 구동할 때 제거된다.

- Broker가 비정상 종료한 경우
  - Broker가 비정상 종료할 경우 Broker를 구동한 후 cubrid broker status 를 수행하면 PSIZE가 -1 또는 비정상적일 경우가 발생한다. 이 경우는 cubrid broker restart 명령을 사용하여 Broker를 restart한다.

- 에러로그 확인 방법
  - CUBRID 운영 중 발생하는 에러는$CUBRID/log/server/ 디렉토리에 DB명_년월일_시분.err 파일로 존재한다.

CUBRID

Trouble Shooting - 계속

- Broker가 구동이 안 될 경우
  - Broker 를 구동 시 “Shared memory open error.” 라는 메시지가 출력되면서 Broker가 시작 되지 않는 경우가 발생할 수 있다. 이 경우에는 아래와 같은 사항을 확인 후 각 상황에 맞는 처리가 필요하다.
    - Broker가 이미 시작되어 있을 수 있다. 이는 cubrid broker status를 사용하여 확인할 수 있다.
    - Shared memory가 해제되지 않은 경우가 있을 수 있다. 이는 ipcs로 shared memory 정보를 확인할 수 있으며 CUBRID 사용자가 소유권을 가지고 있는 shared memory를 해제해야 한다. Shared memory 해제는 ipcrm을 사용하여 수행할 수 있다.

```
$ipcs
------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 4587521    root       644        52         2
0x00000000 4620291    root       644        16384      2
0x00000000 4653060    root       644        268        2
0x00030001 5799942    cubrid     644        10064      2
0x00030000 5832711    cubrid     644        122072     6
0x00033000 5865480    cubrid     644        94492      6

$ipcrm -m 5799942
$ipcrm -m 5832711
$ipcrm -m 5865480
```

    - Broker가 다른 사용자로 구동되어 있을 수 있다. 이 경우 해당 사용자로 로그인 하여 Broker를 중지하여야 한다.

Q&A

감사합니다.