# CBURID운영자과정

고객지원팀

CUBRID
More than open source!

CUBRID2008 R2.0 v. 20090100

# 목차

CUBRID
More than open source!

# CUBRID 살펴보기

CUBRID
More than open source!

# 소개

CUBRID
More than open source!

국내 플랫폼 SW 기술력 확보

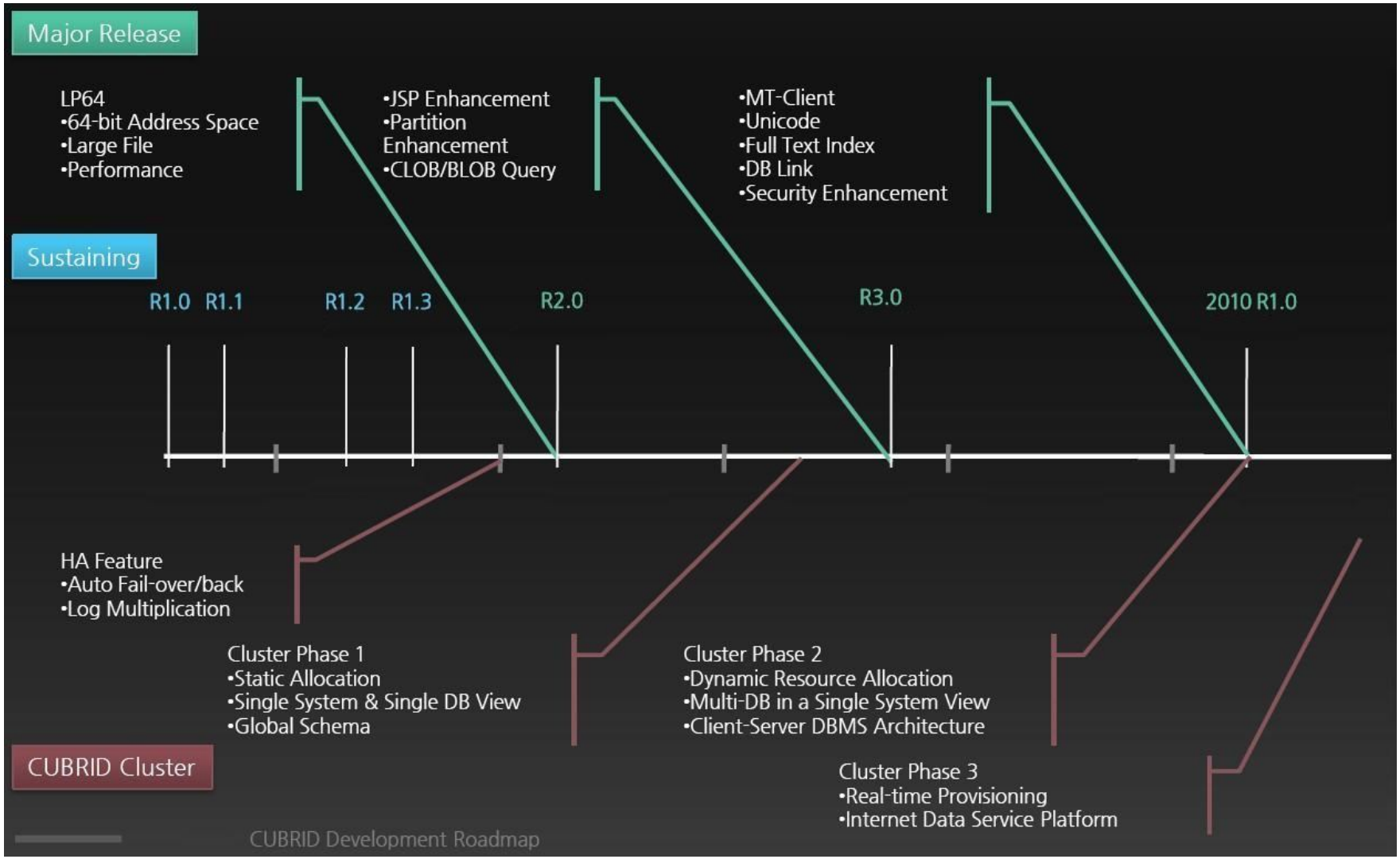
국산 DBMS 성공

웹 서비스 최적의 DBMS

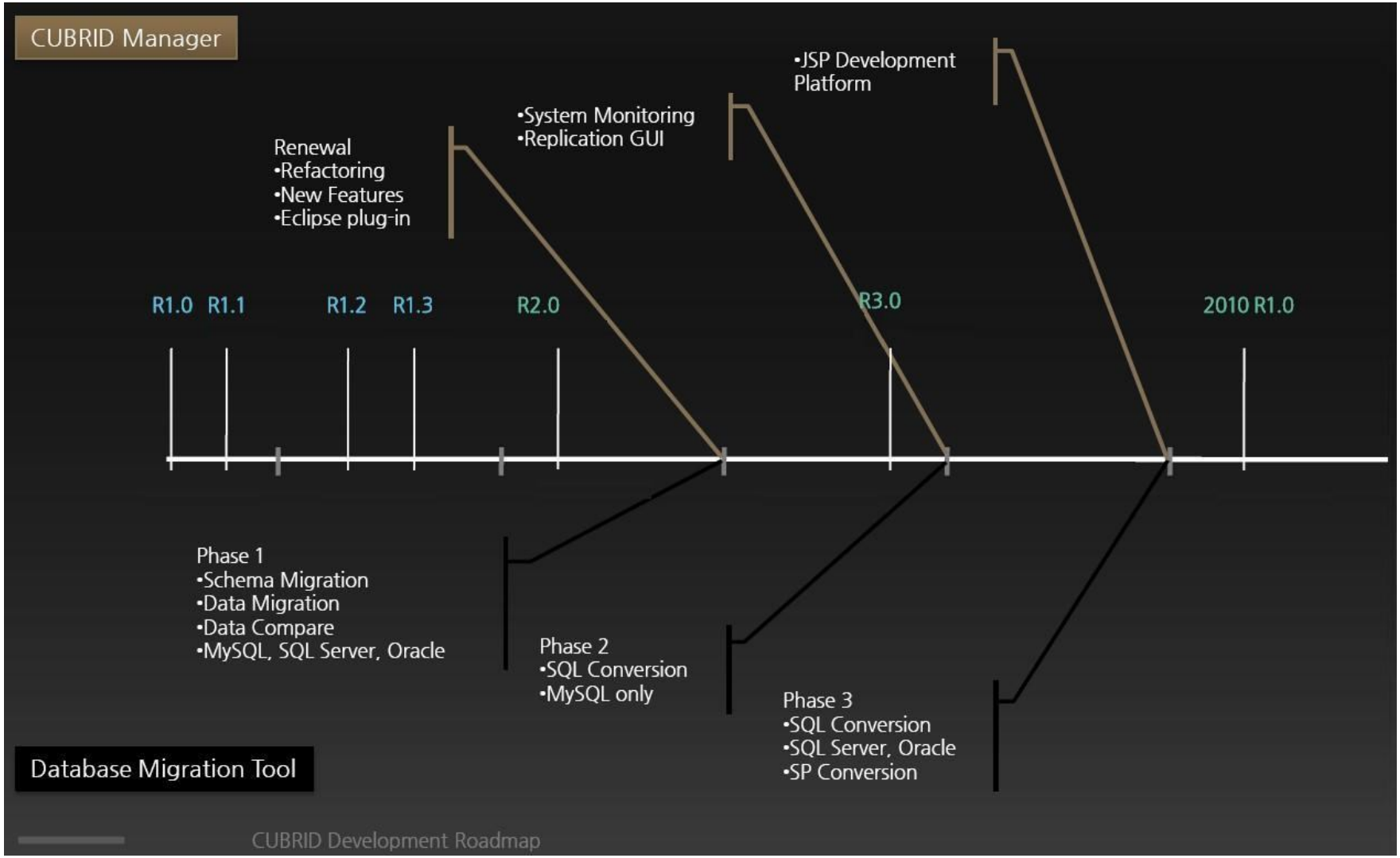
DBMS 카피 점유율: 30%

개발자 확보: 30%

- CUBRID Road Map



- CUBRID Tool Road Map



CUBRID
More than open source!

| 구 분 | 상세설명 |
|---|---|
| 대용량 | DB개수, 멀티볼륨/크기, 테이블 개수/크기 무제한 지원<br>컬럼 개수: 6,400, 컬럼 크기: 2GB, 색인 개수: 6,400 |
| 확장성 | 멀티 쓰레드 서버 구조(멀티 CPU 사용 최적화), 복제, 분할 지원 |
| 고성능 | 고성능 서버구조, 플랜 캐쉬, 질의결과 캐쉬, 최적화기 개선, 분할, 복제 |
| 고가용성 | 완벽한 트랜잭션 지원, 복제간 트랜잭션 일치성 보장 |
| 고성능 Client | Broker 미들웨어(3-tier 구조)<br>• Thread Pool 관리<br>• 부하 최적화 기능 |
| 개발 생산성 | ODBC, JDBC, OLE DB, PHP, Ruby, Python, CCI 등 지원<br>관계형/객체지향형 모델 지원, Java Stored Procedure 지원 |
| 객체지향 | User Definded Data Type, Collection Data Type(set, sequence, multiset)<br>Inheritance, OID(object Identifier) |
| 관리편의성 | JAVA기반의 큐브리드 매니저 지원(Clinet/Server 구조) |

# CUBRID 설치하기

CUBRID
More than open source!

# 지원 플랫폼

CUBRID
More than open source!

| OS | CUBRID 2008 R2.0 | 용량 | 호환 OS | 호환 CPU | 일자 |
|---|---|---|---|---|---|
| MS Windows 32bit | CUBRID-Windows-8.2.0.1150.EXE | 45.14M | * Windows XP Home 32bit<br>* Windows XP Pro<br>* Windows 2003<br>* Windows Vista 32bit | * x86<br>* Intel EM64T<br>* AMD64 | 09-08-14 |
| MS Windows 64bit | CUBRID-Windows-x64-8.2.0.1150-beta.EXE | 46.19M | * Windows Vista 64bit | * Intel EM64T<br>* AMD64 | 09-08-19 |

# 지원 플랫폼 (계속)

CUBRID
More than open source!

| | | | | | |
|---|---|---|---|---|---|
| Linux 32bit | CUBRID-8.2.0.1150-linux.i386.sh | 74.74M | * CentOS 4이상<br>* Fedora 4 이상<br>* Ubuntu 6.10 이상<br>* openSUSE 11 이상<br>* Gentoo 2007 이상<br>* Asianux 2 이상<br>* Debian 4 이상 | * x86<br>* Intel EM64T<br>* AMD64 | 09-08-19 |
| | CUBRID-8.2.0.1150-linux.i386.tar.gz | 74.69M | 상동 | 상동 | 09-08-19 |
| | CUBRID-8.2.0.1150-el5.src.rpm | 54.31M | 상동 | 상동 | 09-08-19 |
| | CUBRID-8.2.0.1150-el5.i386.rpm | 46.26M | * CentOS5 32bit | 상동 | 09-08-19 |
| Linux 64bit | CUBRID-8.2.0.1150-linux.x86_64.sh | 71.79M | * CentOS 4이상<br>* Fedora 11<br>* Ubuntu 9.04 | * Intel EM64T<br>* AMD64 | 09-08-14 |
| | CUBRID-8.2.0.1150-linux.x86_64.tar.gz | 71.73M | 상동 | 상동 | 09-08-14 |
| | CUBRID-8.2.0.1150-el5.x86_64.rpm | 47.09M | * CentOS5 64bit | 상동 | 09-08-19 |

# 확인/준비 사항

CUBRID
More than open source!

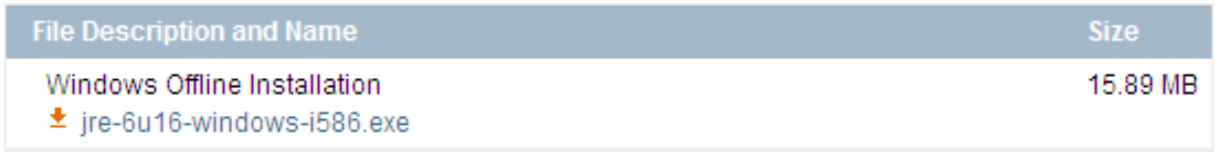
- LINUX  glibc 2.3.4이상
  - rpm -qa glibc
- LINUX/UNIX의 의 경우 root아닌 CUBRID용 계정에 설치 권장
- JAVA 환경(1.5이상, 1.6권장) → 선택사양
  - JAVA버전 확인 ➔ java -version
  - JAVA Stored Procedure ➔ JRE설치
  - CUBRID Manager Client ➔ JRE설치
- Windows설치 시 Visual C++ 재배포 패키지(x86) 설치
  - ODBC, OLEDB 만 사용시에도 설치 필요
- Windows 설치시 Administrator 권한 계정 설치 권장

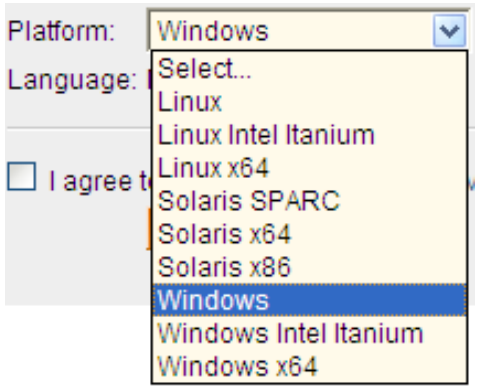
# 설치 - JAVA

CUBRID
More than open source!

- CUBRID Manager Client / JAVA StoredProcedure 사용에 필요
- JAVA다운로드 (java.sun.com → Downloads → JAVA SE)
  - http://java.sun.com/javase/downloads/index.jsp
  - 최신버전의 JAVA 선택 (JAVA SP 실습위해 JDK설치)
    - JDK6 Update16 (운영환경의 경우 JRE6 Update16), 2009년 9월 현재


  - 사용 Platform(OS)선택하여 다운로드
  - Windows offline Installation 선택 설치



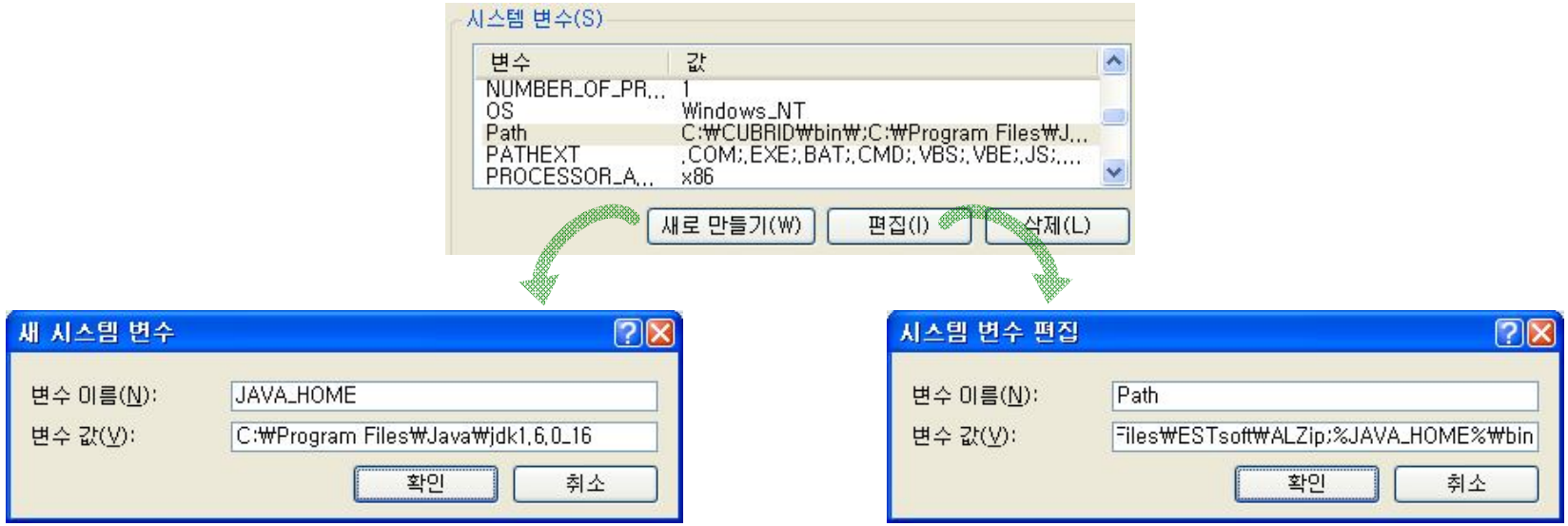



# 설치 - JAVA (계속)

CUBRID
More than open source!

- JAVA 환경 변수 설정
  - Windows 수동 설정
    - 제어판 → 시스템 → 고급탭 → 환경변수 → 시스템변수

```
set JAVA_HOME= C:₩Program Files₩Java₩jdk1.6.0_16
set PATH=%PATH%;%JAVA_HOME%₩bin;%JAVA_HOME%₩jre₩bin₩client
REM JRE 설치시
set JAVA_HOME= C:₩Program Files₩jre6
set PATH=%PATH%;%JAVA_HOME%₩bin;%JAVA_HOME%₩bin₩client
```



# 설치 - JAVA (계속)

CUBRID
More than open source!

- JAVA 환경 변수 설정
  - Linux
    - LD_LIBRARY_PATH를 libjvm.so 가 있는 곳으로 수정

```
export  JAVA_HOME=/usr/local/jdk1.6.16
export  PATH=$PATH;$JAVA_HOME/bin
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$JAVA_HOME/jre/lib/i386:$JAVA_HOME/jre/lib/i386/
client
# JRE 설치시
export  JAVA_HOME=/usr/local/jre1.6.16
export  PATH=$PATH;$JAVA_HOME/bin
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$JAVA_HOME/lib/i386:$JAVA_HOME/lib/i386/client
```

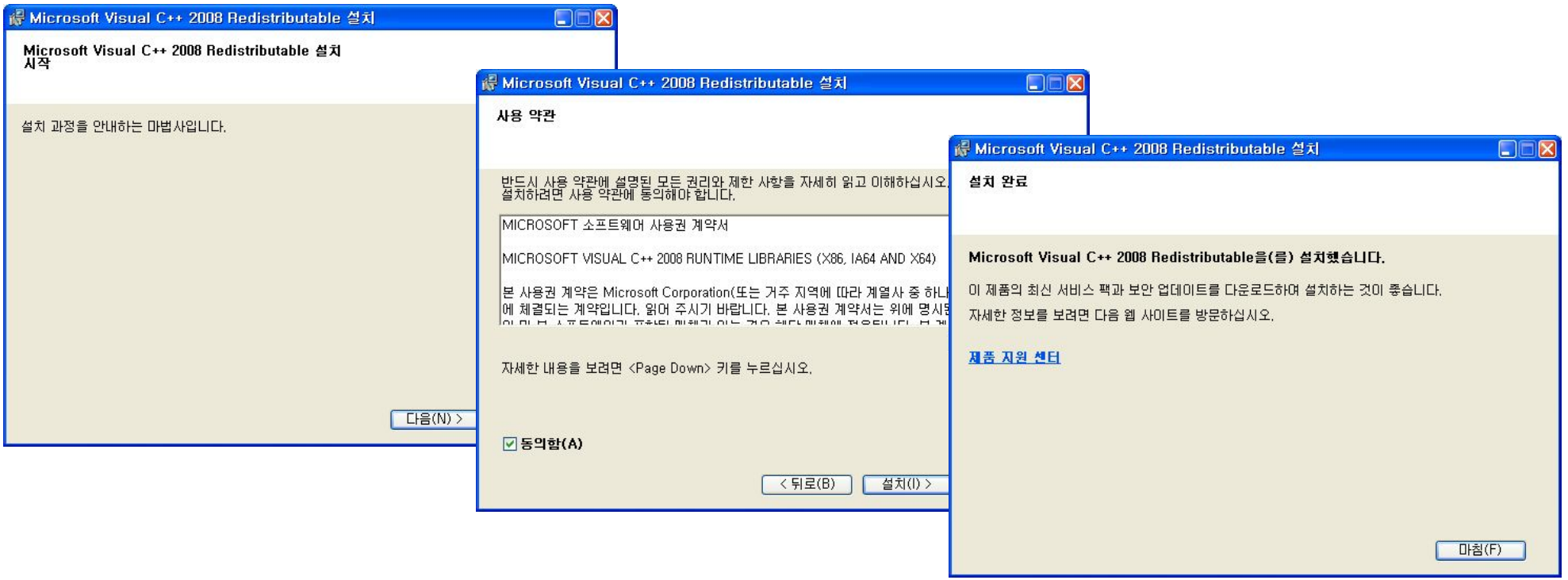
# CUBRID 설치 - Windows

CUBRID
More than open source!

- Microsoft Visual C++ 2008 재배포 패키지를 설치
  - www.cubrid.com 다운로드페이지의 링크 또는 아래 링크에서 설치

**Visual C++ 2008 재배포 패키지 다운로드**

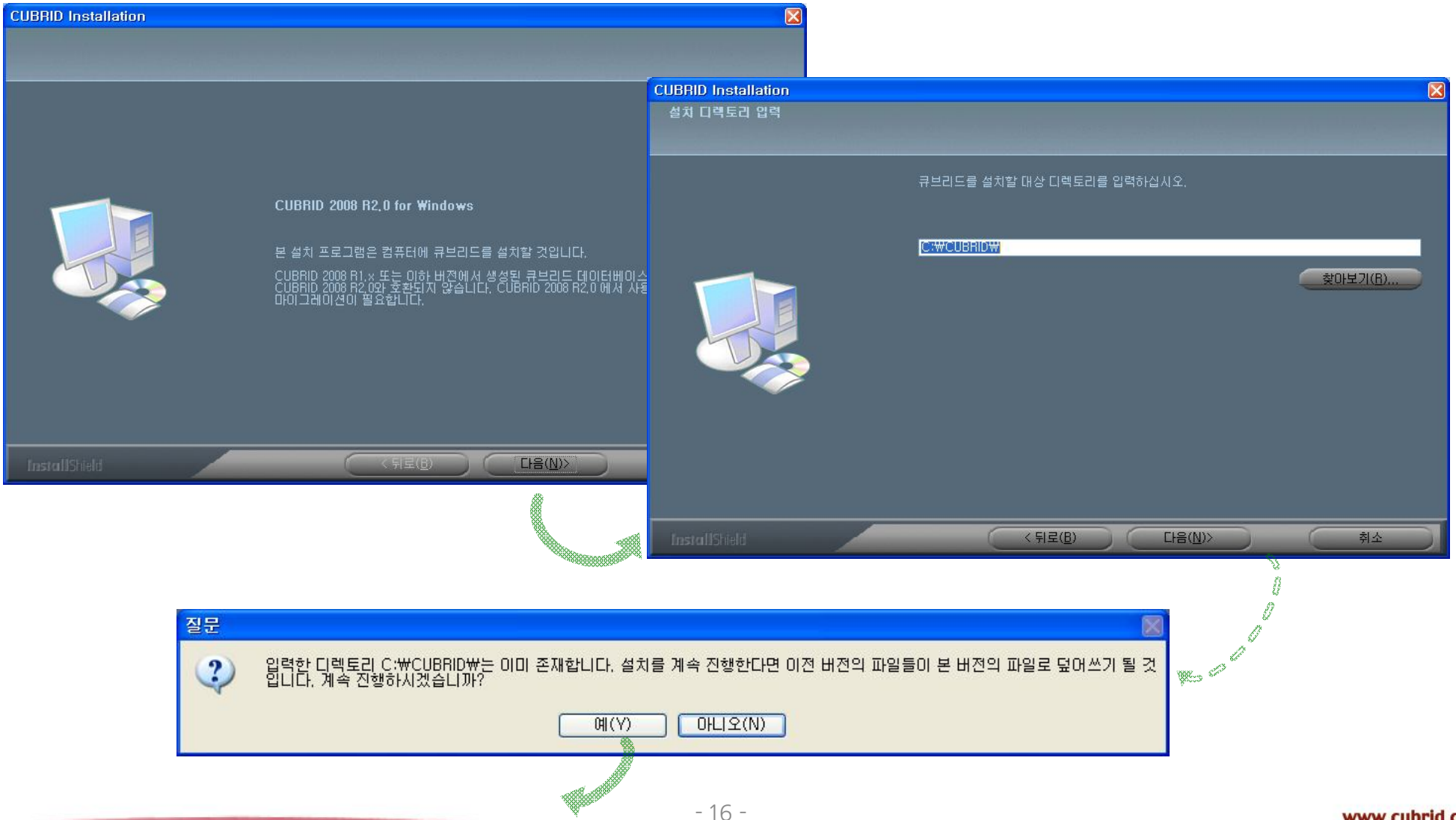
CUBRID 2008 Windows 용 제품을 설치하기 전에 반드시 설치해 주시기 바랍니다.
- http://www.microsoft.com/downloads/details.aspx?displaylang=ko&FamilyID=9b2da534-3e03-4391-8a4d-074b9f2bc1bf



# CUBRID 설치 - Windows (계속)

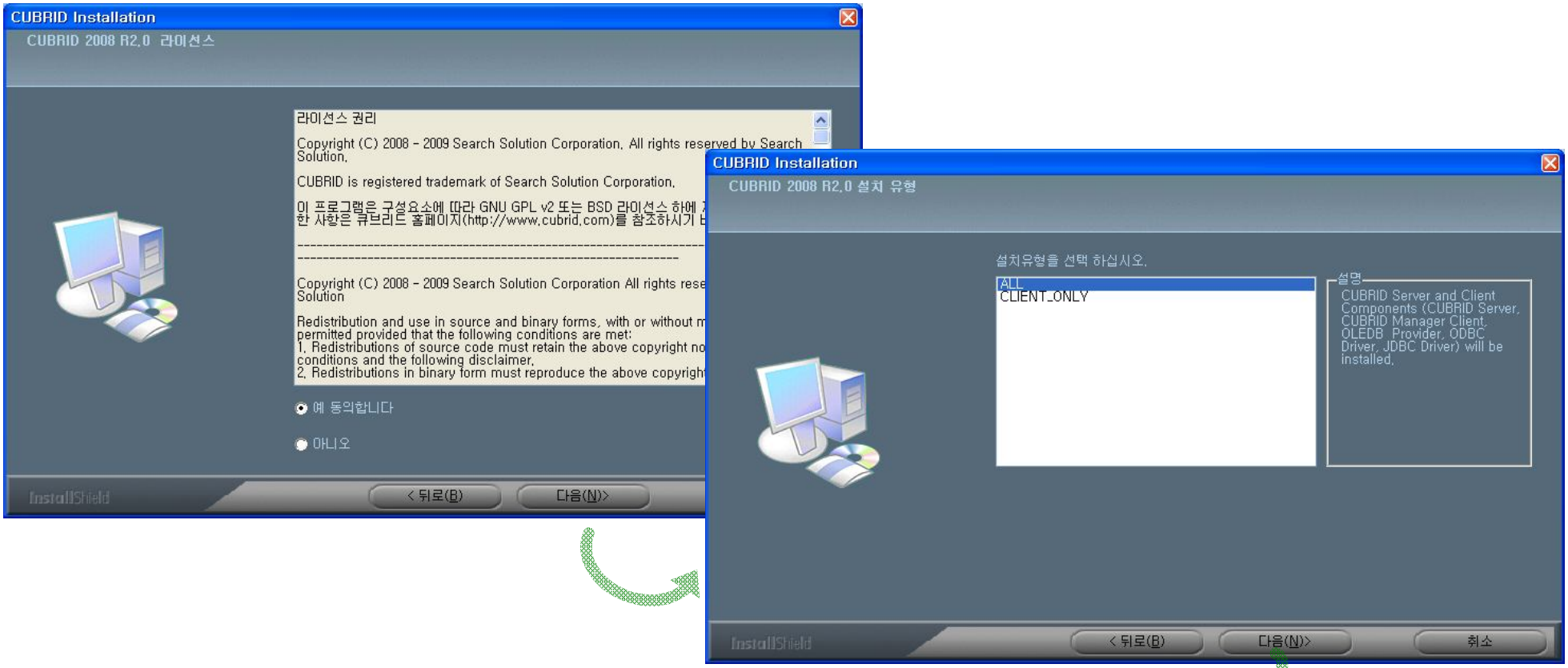
CUBRID
More than open source!

- 다운로드페이지에서 MS Windows(32bit) 제품을 선택



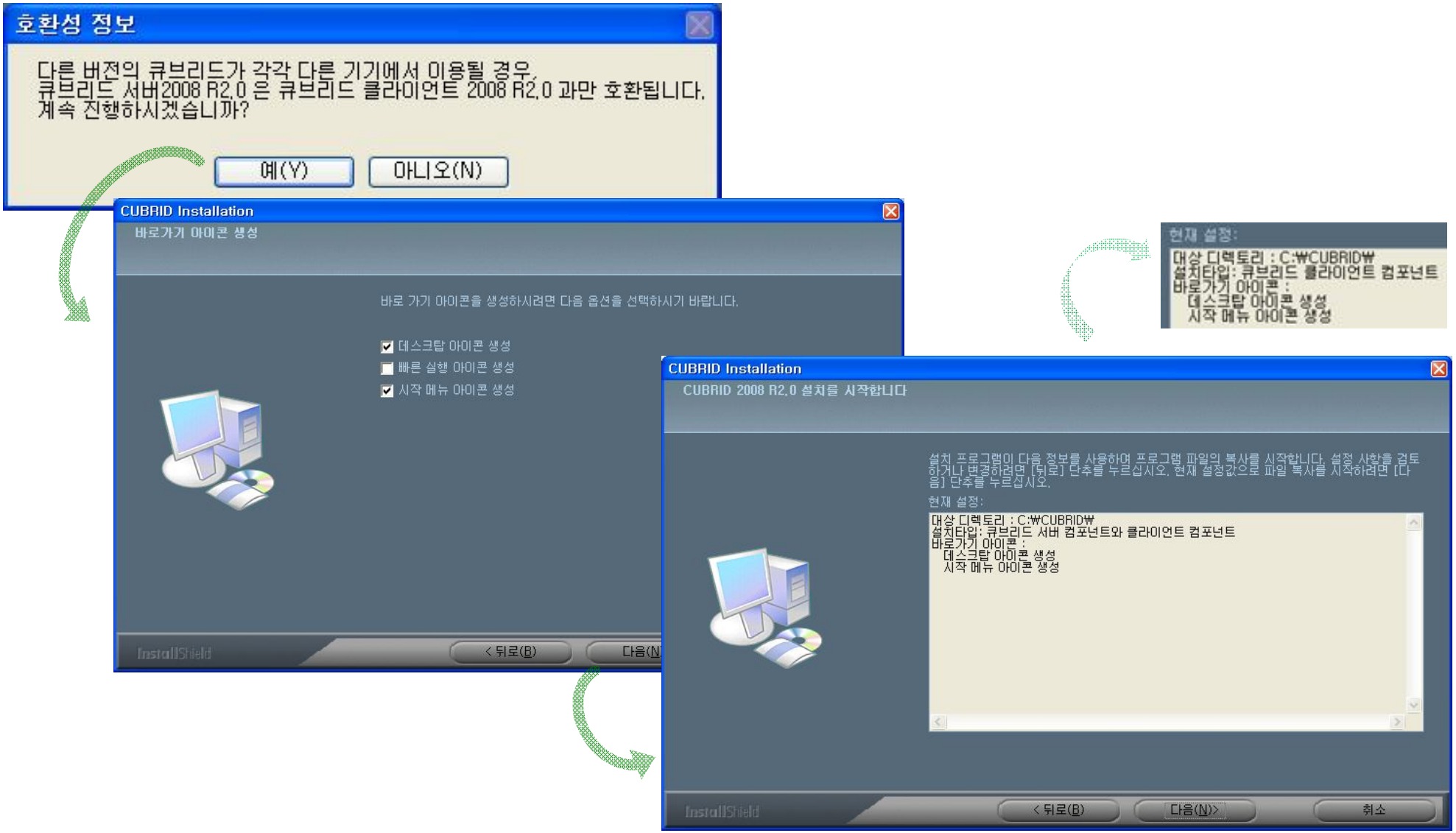
# CUBRID 설치 - Windows (계속)

CUBRID
More than open source!



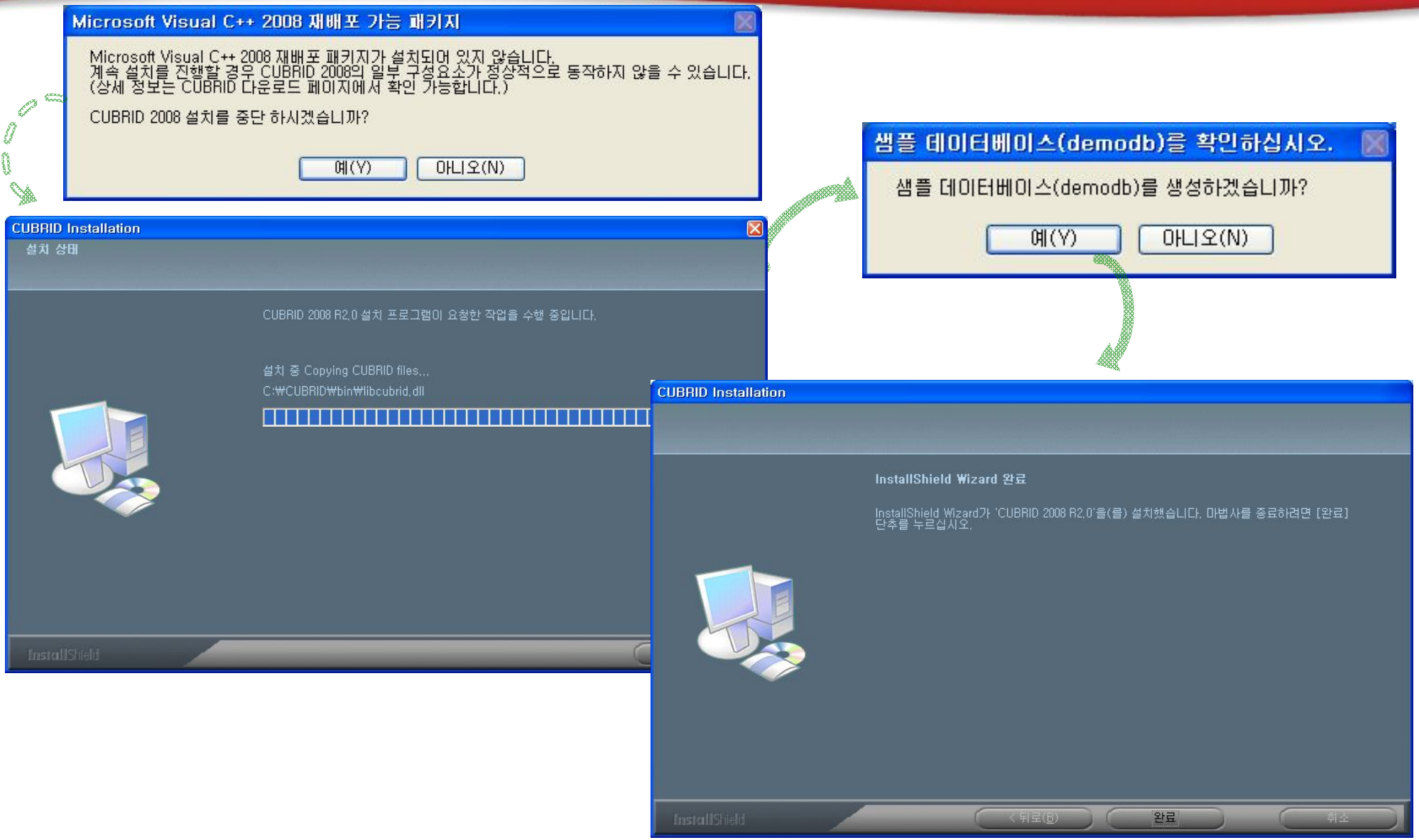

CUBRID는 다른 서버에 설치되어 있고 CUBRID Client 를 설치할 경우 CLIENT_ONLY을 선택

# CUBRID 설치 - Windows (계속)



# CUBRID 설치 - Windows (계속)

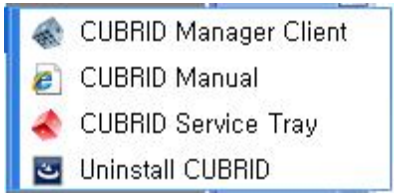
CUBRID
More than open source!



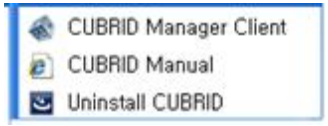
# CUBRID 설치 - Windows (계속)

CUBRID
More than open source!

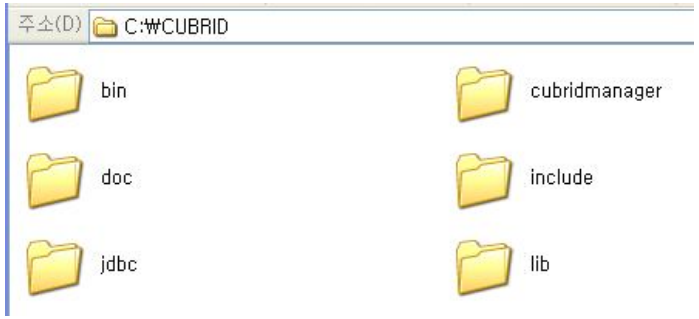
- CUBRID 설치 후 확인사항
  - 전부 설치시




  - client만 설치시

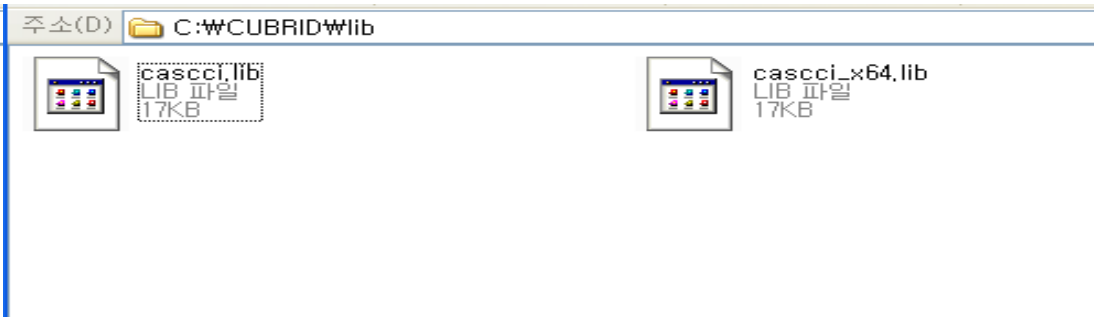



  - 디렉토리 구조





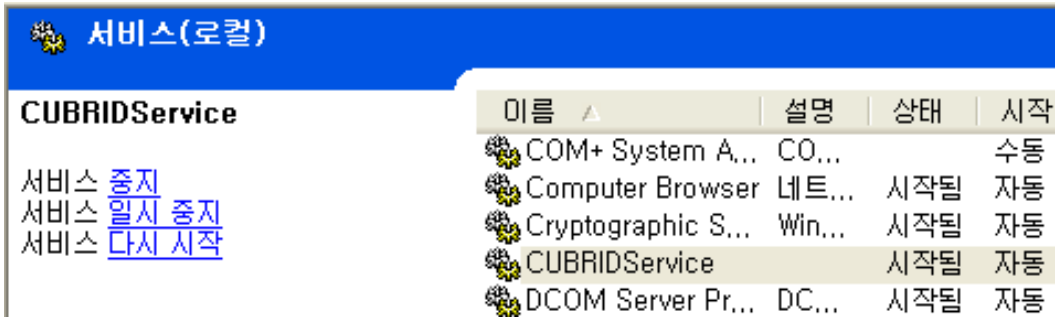


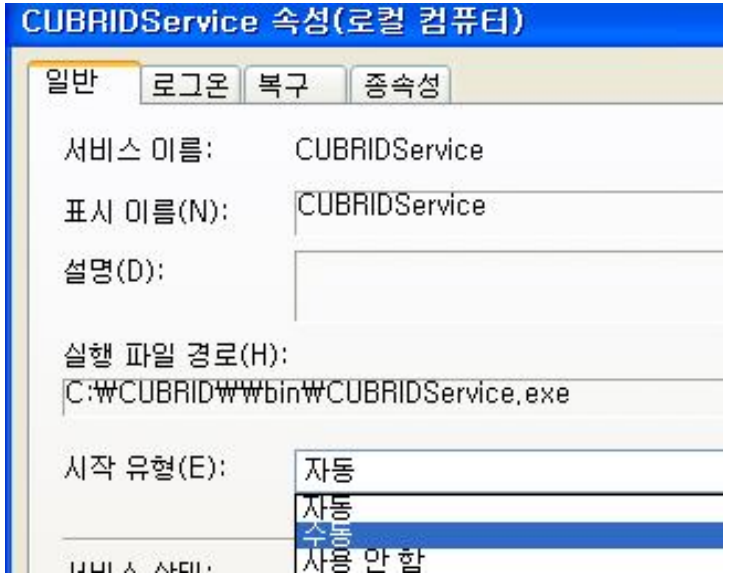


# CUBRID 설치 - Windows (계속)

CUBRID
More than open source!

- CUBRID 설치 후 확인사항
  - 시스템의 서비스에  CUBRIDService 자동 등록된다.
  - Windows 시작 시 서비스 자동구동을 중지하기 위해서는 “시작유형”을 수동으로 변경한다.



# CUBRID 설치 - LINUX

CUBRID
More than open source!

- 다운로드페이지에서 LINUX용 Binary shell 설치 제품을 선택

```
[CUBRID~]$ sh CUBRID-8.2.0.11150-linux.sh
Copyright (C) 2009 Search Solution Corporation. All rights reserved by Search Solution.
CUBRID is registered trademark of Search Solution Corporation.

This Software is released under GNU GPL v2 or BSD according to its components.
For more information, please refer to the CUBRID home page(http://www.cubrid.com).
.
..
.
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) directory? (yes or no) [Default: yes] : yes
Install CUBRID to '/home/CUBRID' ...
In case a different version of the CUBRID product is being used in other machines, please note that the CUBRID
2008 R2.0 servers are only compatible with the CUBRID 2008 R2.0 clients and vice versa.
Do you want to continue? (yes or no) [Default: yes] : yes
Copying old .cubrid.sh to .cubrid.sh.bak ...

CUBRID has been successfully installed.

demodb has been successfully created.

If you want to use CUBRID, run the following commands
  % . /home/CUBRID/.cubrid.sh
  % cubrid service start
```

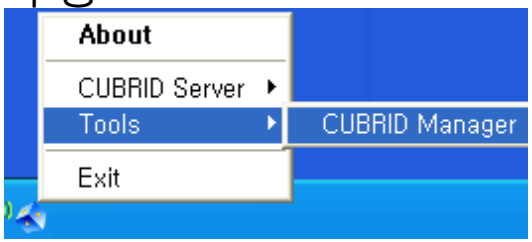
# CUBRID 설치 - LINUX (계속)

CUBRID
More than open source!

- 설치 확인

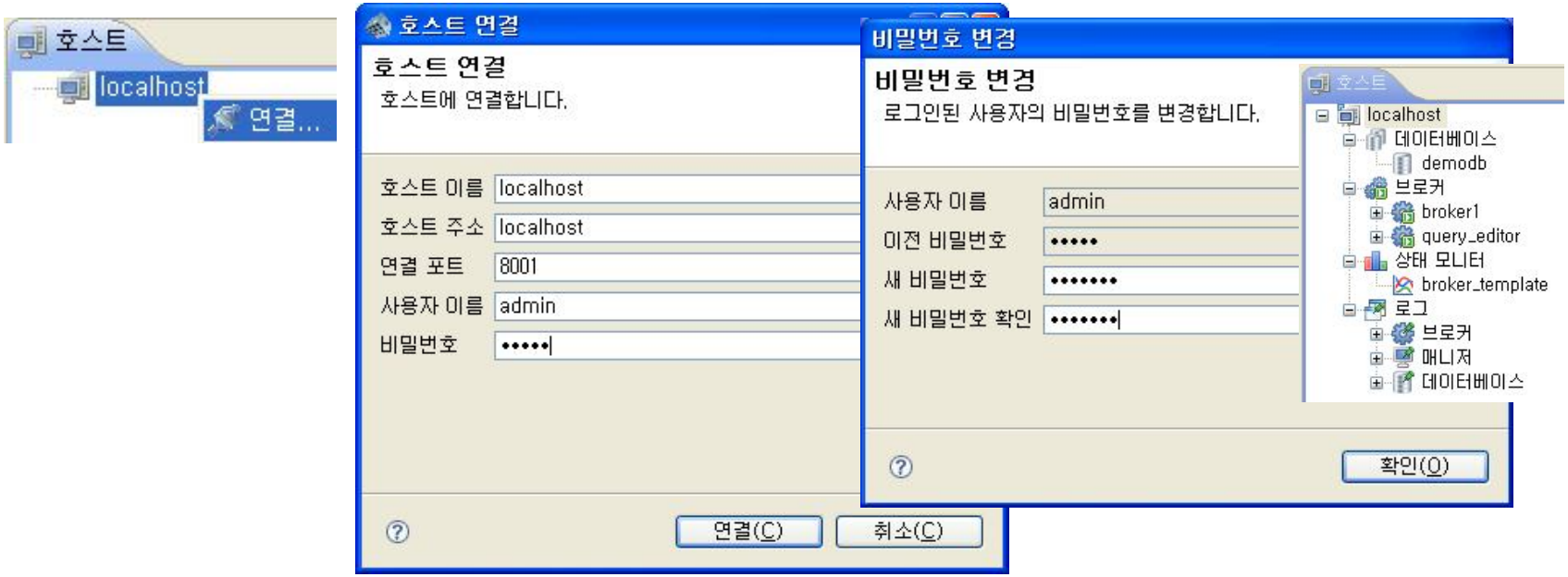
```
[CUBRID~]$. /home/CUBRID/.cubrid.sh  ➔ 버전확인
[CUBRID~]$cubrid_rel

CUBRID 2008 R2.0 (8.2.0.1150)

[CUBRID~]$cubrid service start  ➔ 서비스구동
@ cubrid master start
++ cubrid master is running.

@ cubrid broker start
++ cubrid broker start: success

@ cubrid manager server start
++ cubrid manager server start: success
[CUBRID~]$
```

- CUBRID Manager Client
  - 구동

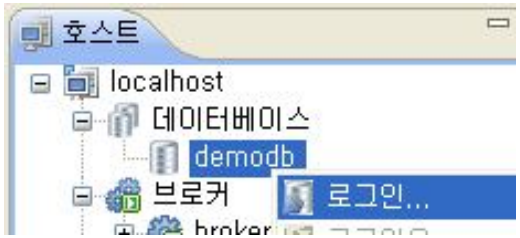


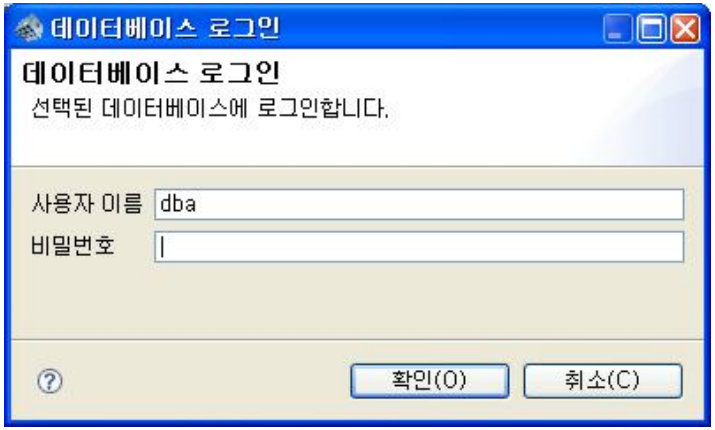


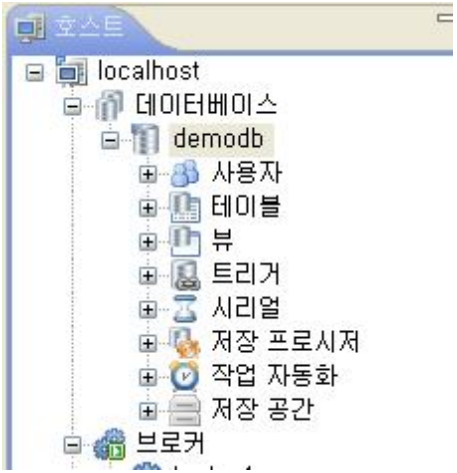
  - 로그인
    - 초기암호는 admin 이며 로그인 후 암호를 변경하여야 한다.



- CUBRID Manager Client
  - 데이터베이스 로그인
    - 데이터베이스별 데이터베이스 사용자로 로그인



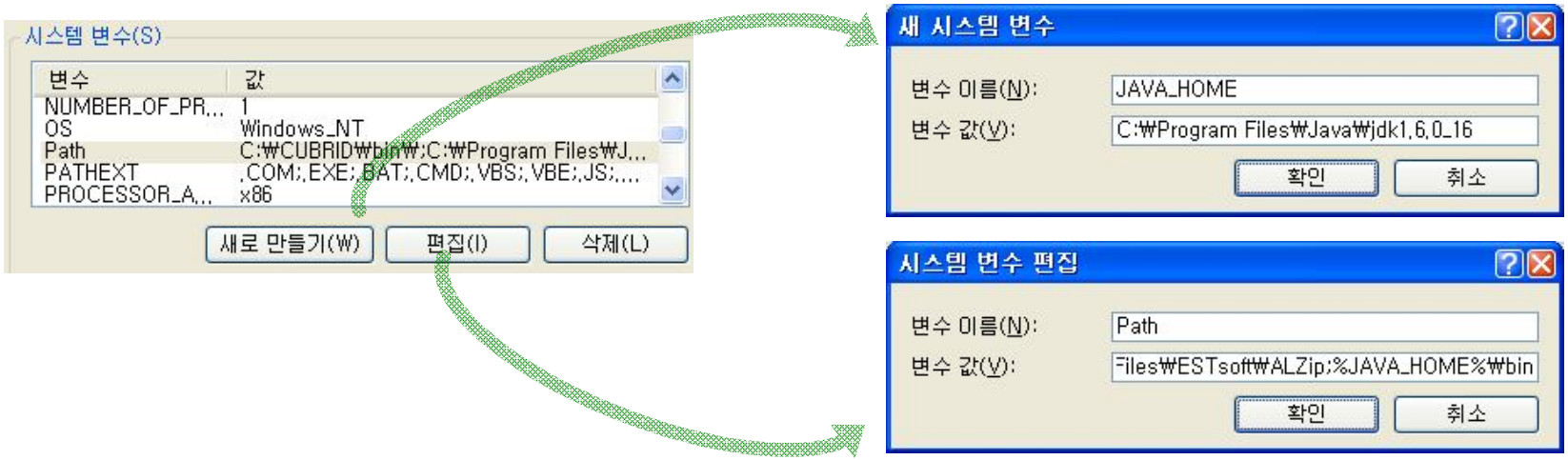
# CUBRID 설정

CUBRID
More than open source!

# JDBC 설정

CUBRID
More than open source!

- JAVA 환경 변수 설정
  - Java 설치 완료 후 JAVA_HOME 환경 변수 추가 (JDK1.5이상)
    - windows
      - 제어판 → 시스템 → 고급탭 → 환경변수 → 시스템변수




    - Linux (예, bash)

```
export JAVA_HOME=/usr/local/jdk1.6.16
export PATH=$JAVA_HOME/bin:$PATH
```

# JDBC 설정 (계속)

CUBRID
More than open source!

- JAVA 환경 변수 설정
  - jdbc driver 에 대한 class path 설정
    - windows

```
CLASSPATH=%CUBRID%₩jdbc₩cubrid_jdbc.jar
```

    - Linux (예, bash)

```
export CLASSPATH=$CUBRID/jdbc/cubrid_jdbc.jar:.
```

# PHP 설정 - Windows

CUBRID
More than open source!

- Apache, PHP 설치
  - Apache, PHP download 및 설치
    - http://httpd.apache.org/download.cgi (예, apache_2.2.10-win32-x86-no_s.msi)
    - http://www.php.net/downloads.php (참조: PHP 5.1이상 5.3이하 사용)
- CUBRID PHP module 설치
  - CUBRID PHP module 설치하기
    - PHP 를 설치한 디렉토리(예, C:₩Program Files₩PHP) 아래 CUBRID 라는 폴더를 만듦.
    - http://dev.naver.com/projects/cubrid-php/download PHP 모듈 다운로드에서 windows 용 PHP 화일들을(cubrid_err.msg, cubrid2008_php5.2.dll) 위에서 생성한 디렉토리에 내려 받는다.
    - 아래 내용을 C:₩windows₩php.ini 에 CUBRID PHP module 정보 추가

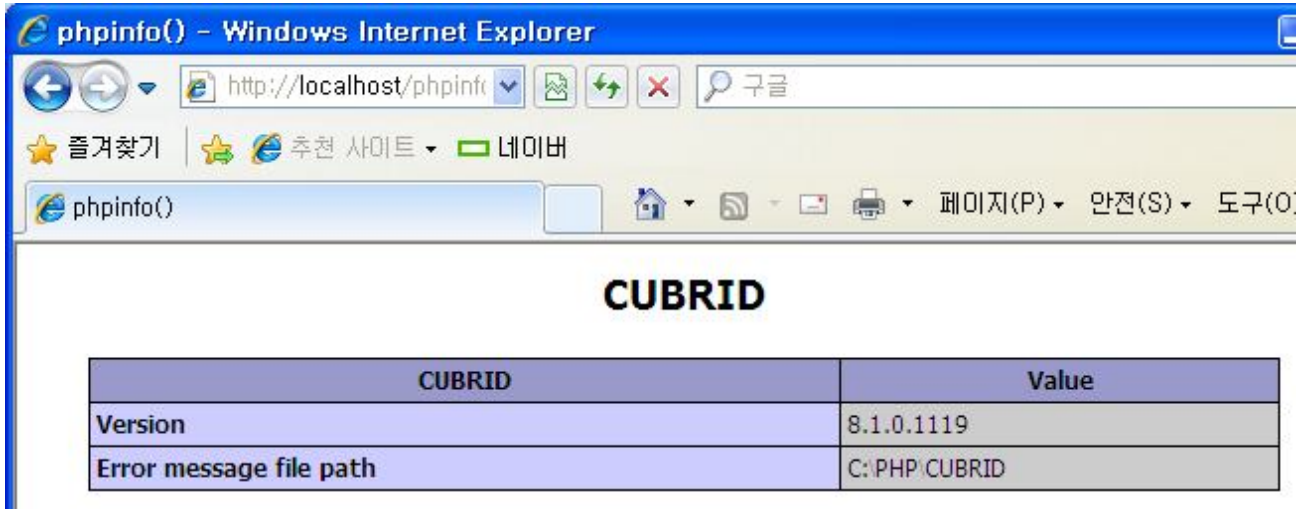
```
extension_dir="C:₩PHP₩CUBRID"
extension= cubrid2008_php5.2.dll

[CUBRID]
cubrid.err_path=C:₩PHP₩CUBRID
```

# PHP 설정 - Windows (계속)

CUBRID
More than open source!

- Apache, PHP, CUBRID 연동 확인
  - Apache web server 재구동하여, phpinfo.php 를 수행하여 아래와 같이 CUBRID 함목이 보이는지 확인한다.

```
phpinfo.php 의 내용: <? phpinfo(); ?>
```




  - CUBRID 내용을 확인할 수 없을 경우 Apache web server의 error log(예, C:₩Program Files₩Apache Software Foundation₩Apache2.2₩logs₩error_log) 의 내용을 참고한다.

# PHP 설정 - Linux

CUBRID
More than open source!

- Apache, PHP, CUBRID 연동 확인
  - Linux 버전은 Windows 와 달리 CUBRID 홈페이지에서 소스코드가 제공 된다.
  - CUBRID PHP module 을 “PHP 모듈 다운로드”에서 Unix/Linux 용(예, cubrid_php_src_8.1.0.1177.tar.gz)을 내려받아 설치 한다.
  - 설정 방법은 큐브리드 홈페이지 > 기술문서 > 튜토리얼 > LINUX에서 PHP 사용하기 - phpize를 이용한 설치 를 참조 한다.
  - php.ini 파일을 아래와 같이 편집하고 저장한 후 웹 서버를 재시작 한다.

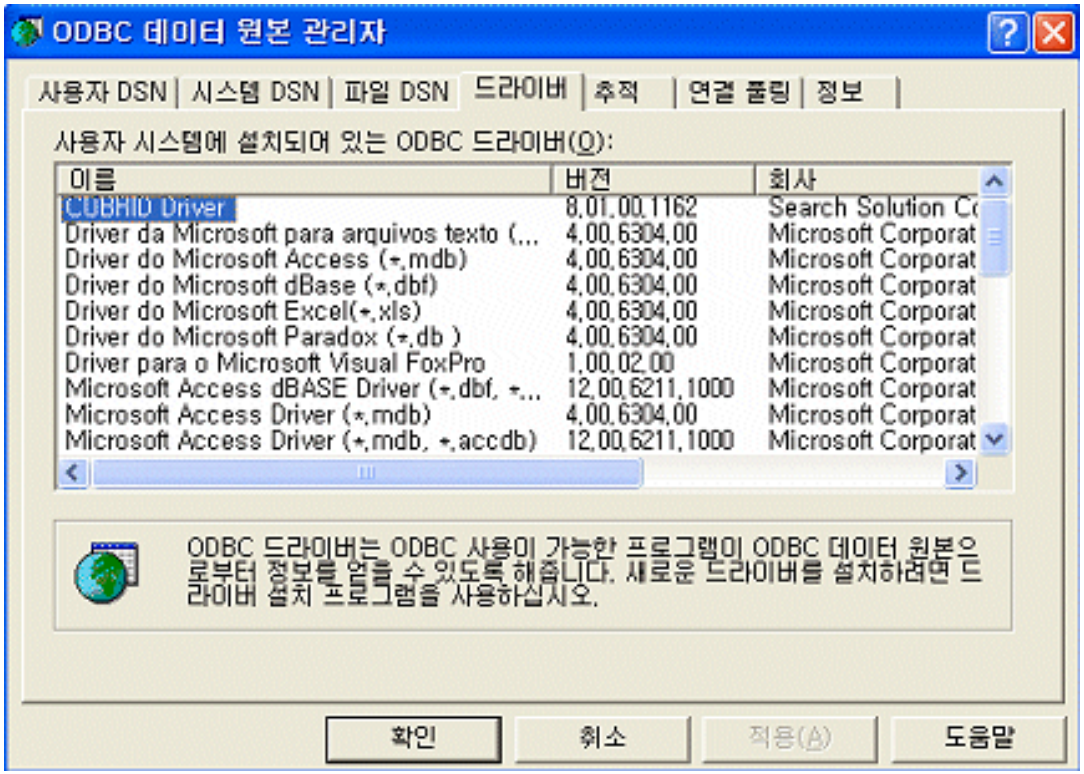
```
% vi /home/web/php/lib/php.ini

extension_dir = "/home/web/php/lib/php/extensions"
extension=cubrid.so
cubrid.err_path="/home/web/php/lib/php/extensions"
```

  - phpinfo()함수를 사용해서 확인했을 때 Windows와 동일하게 웹 브라우저에서 CUBRID 가 보인다면 정상적으로 설치가 완료된 것이다.

# ODBC 설정

CUBRID
More than open source!

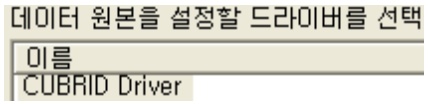
- ODBC 드라이버 확인
  - ODBC Driver는 CUBRID 설치 시 자동으로 설치된다
  - 설치된 CUBRID ODBC Driver를 확인하려면 [제어판]->[관리도구]->[데이터원본(ODBC)]를 실행한다.



# ODBC 설정 (계속)

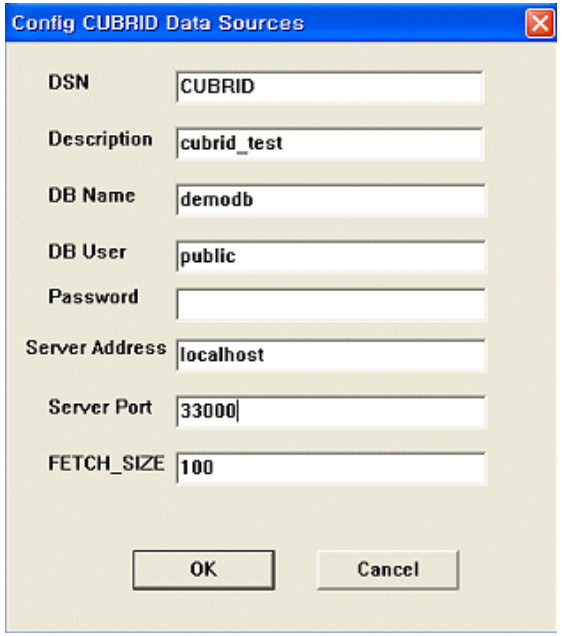
CUBRID
More than open source!

- CUBRID ODBC 설정
  - DSN 설정
    - 제어판→관리도구→데이터원본(ODBC)→ [추가] 클릭하여 CUBRID Driver를 선택 후 [마침] 클릭




    - CUBRID Data Source 설정
      - DSN: 이름
      - Description: 요약 설명
      - DB Name: 접속 대상 데이터베이스 이름
      - DB User: 데이터베이스 사용자 ID
      - Password: 데이터베이스 사용자 암호
      - Server Address: 대상 서버 IP
      - Server Port: CUBRID Broker port 번호

        (Broker port Default: 33000)




  - 연결문자열 지정

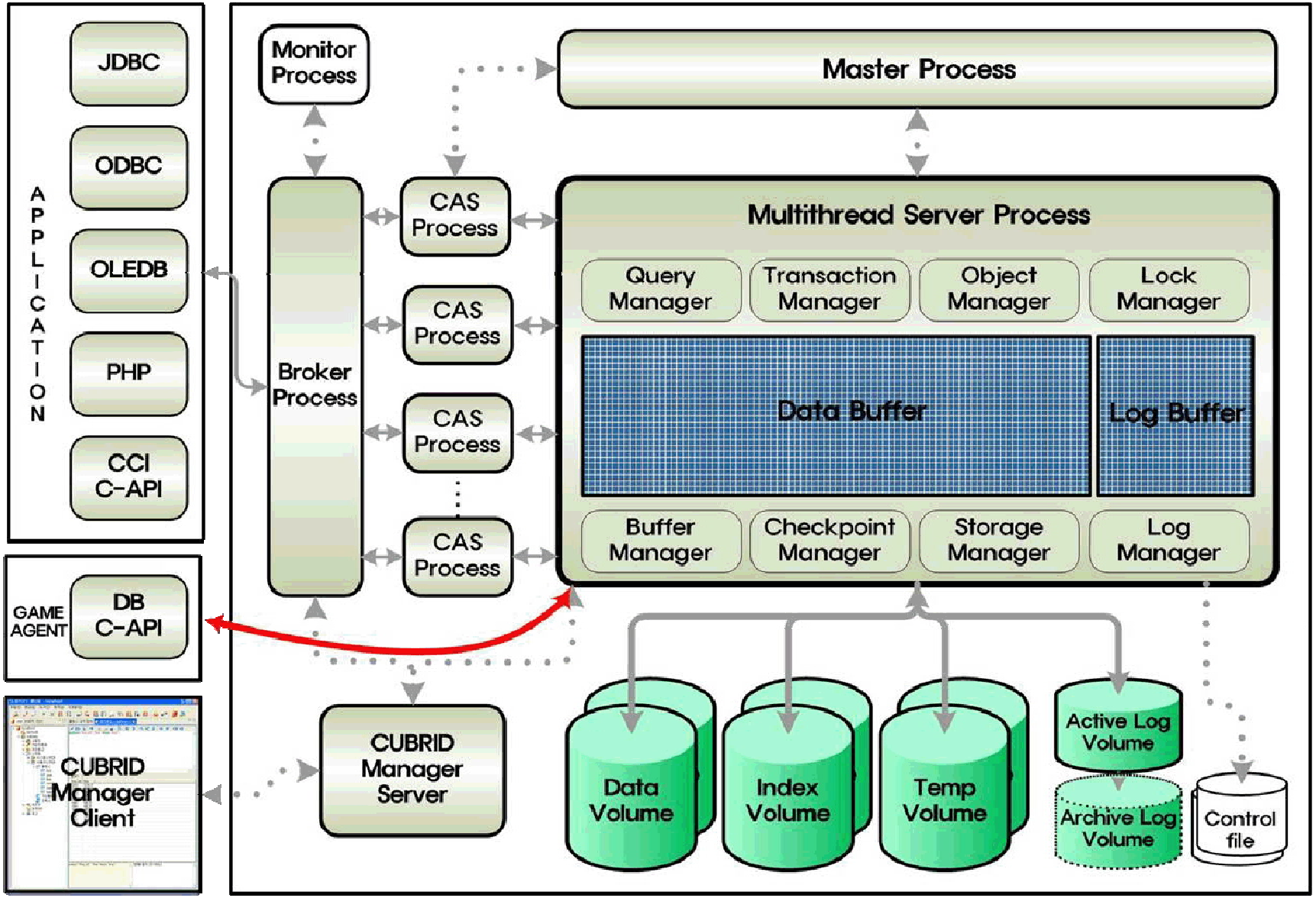
```
sConn = "driver={CUBRID Driver};server=localhost;port=33000;uid=public;pwd=;db_name=demodb;"
```

# 데이터베이스 구조

CUBRID
More than open source!

# CUBRID 구조

CUBRID
More than open source!



- 디렉토리 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 |
| cubridmanager/ | cubrid manager client 가 위치한 디렉토리 |
| databases/ | databases.txt 와 sample 데이터베이스가 위치한 디렉토리 |
| demo/(LINUX) | demodb생성 디렉토리 |
| doc/ | 매뉴얼 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 |
| lib64/ | 64bit OS 를 위한 library 파일이 위치한 디렉토리 |
| log/ | 각족 log 가 저장되는 디렉토리 |
| msg/ | CUBID 에서 사용되는 각종 메시지 파일이 저장된 디렉토리 |

# 데이터베이스 프로세스

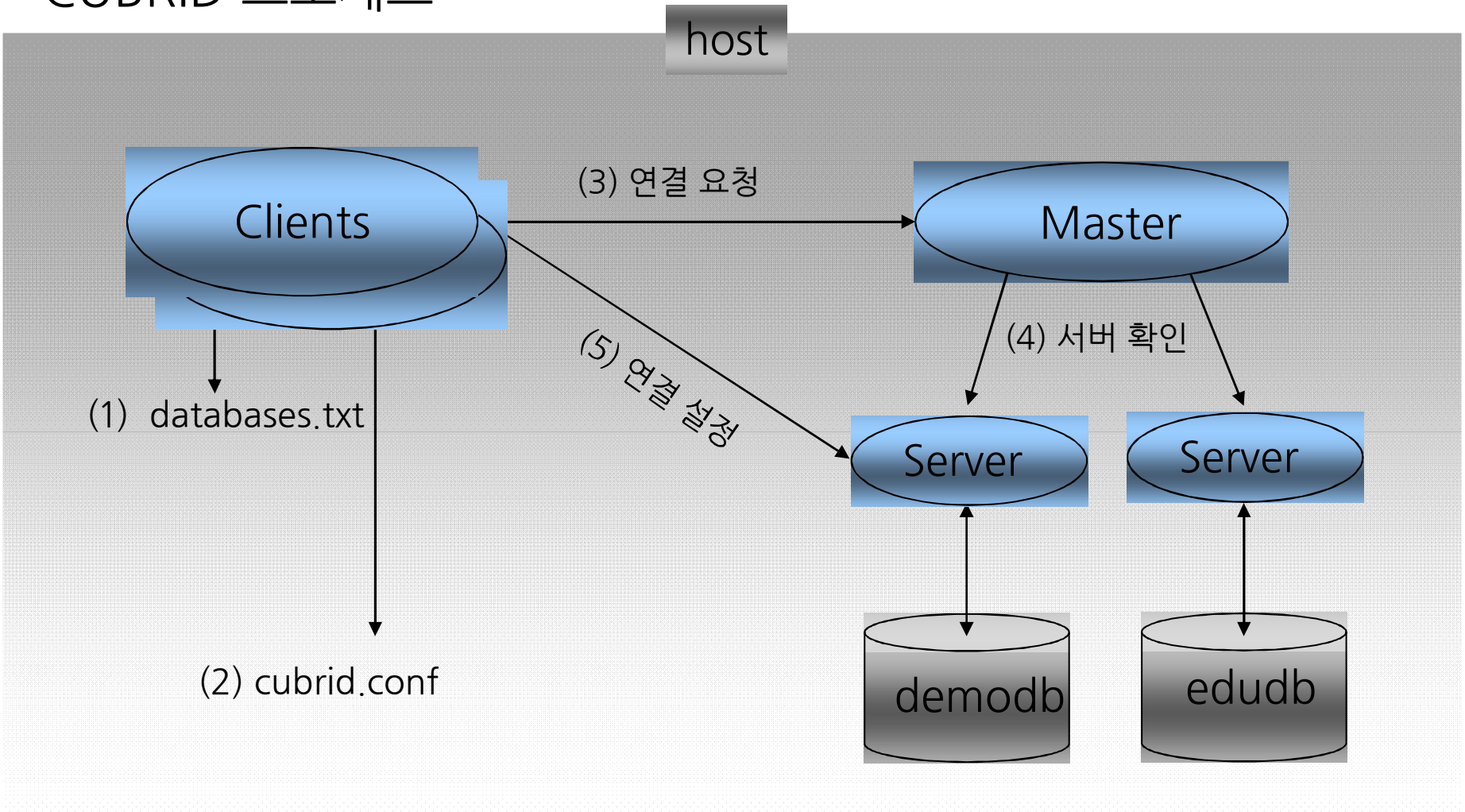
CUBRID
More than open source!

- Master process
  - 클라이언트와 서버 프로세스 사이의 연결을 담당하는 프로세스
  - 서버는 마스터 프로세스에 등록되어 있다.
- Server process
  - 클라이언트의 요청을 처리하는 프로세스
  - 하나의 데이터베이스 마다 하나의 서버 프로세스가 존재한다.
  - 하나 이상의 서버가 하나의 호스트에서 운영될 수 있다.
- Client process
  - 데이터베이스와 통신을 하기 위해 한 명의 사용자에게 허용되는 프로세스
  - 클라이언트는 사용자가 작성한 응용 프로그램 또는 데이터베이스와 관련된 유틸리티이다.
  - 주어진 데이터베이스와 관련된 클라이언트 수는 제한이 없다.
  - 하나의 클라이언트는 한번에 하나의 데이터베이스에만 접속 가능하다.
  - 클라이언트 프로세스는 databases.txt 파일을 이용한다.
  - 대부분의 응용환경에서 CUBRID Broker의 cas프로세스를 의미한다.

# 프로세스간 연결처리 과정

CUBRID
More than open source!

- CUBRID 프로세스

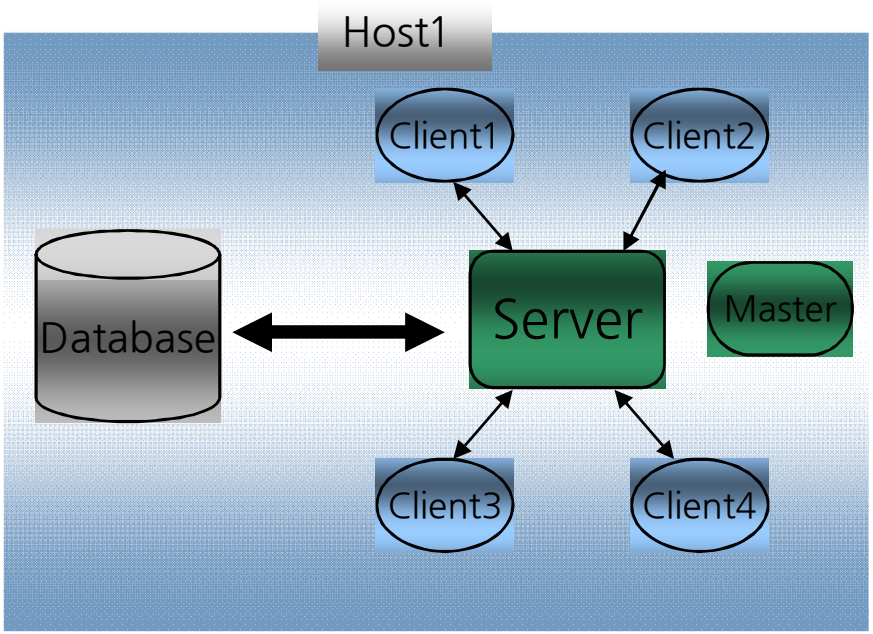


# 서비스 모드

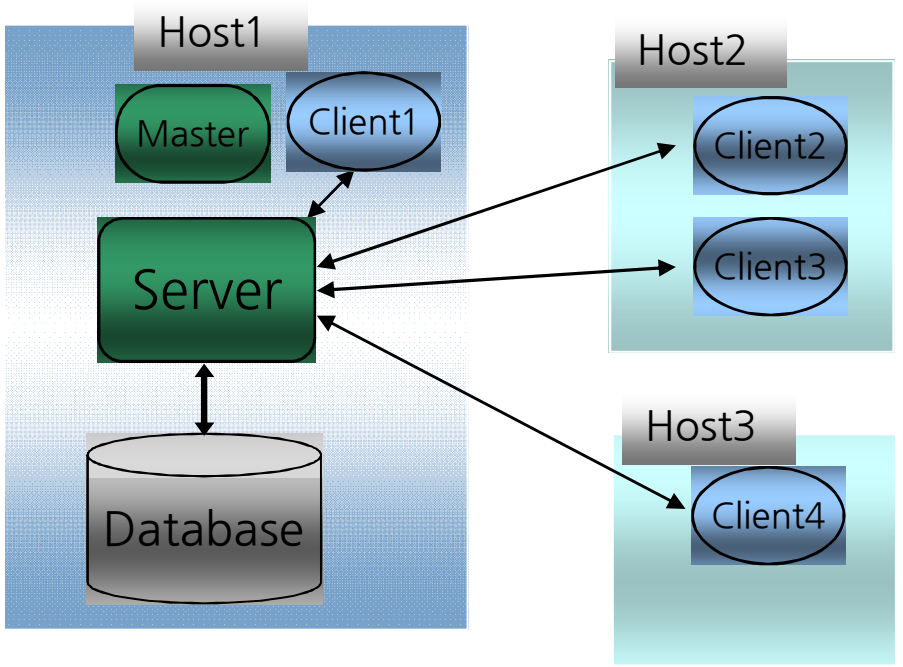
CUBRID
More than open source!

- Client/Server Mode
  - 데이터베이스에 한 개 이상의 클라이언트가 접속하는 구조
  - 호스트에 하나 이상의 데이터베이스를 운영하고자 한다면 서버 프로세스도 하나 이상이 작동해야 한다. 이때, 마스터는 모든 서버가 공유한다.
  - 각 클라이언트의 동시성을 제어하기 위해 LOCK을 사용한다.
  - 클라이언트의 수가 증가함에 따라 성능도 떨어진다.

● A single host Client/Server Mode



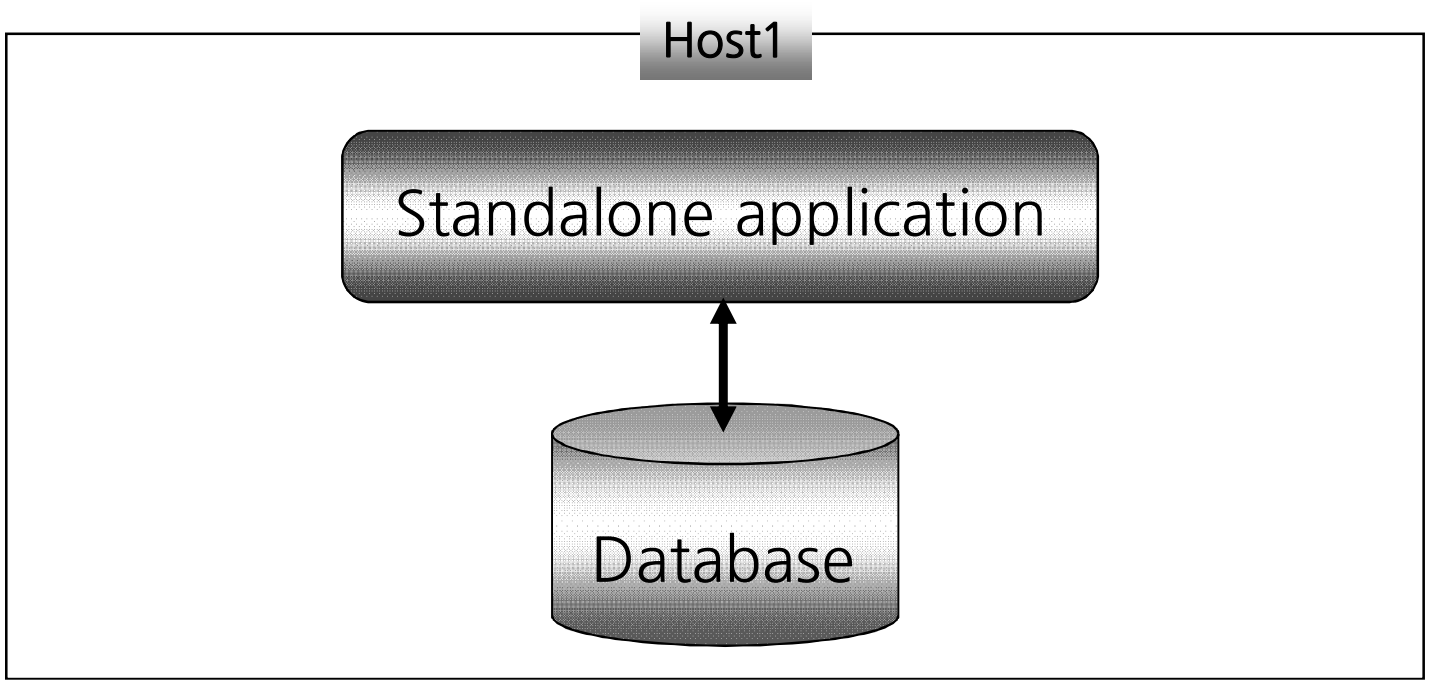

● A multi-system Client/Server Mode



# 서비스 모드 - 계속

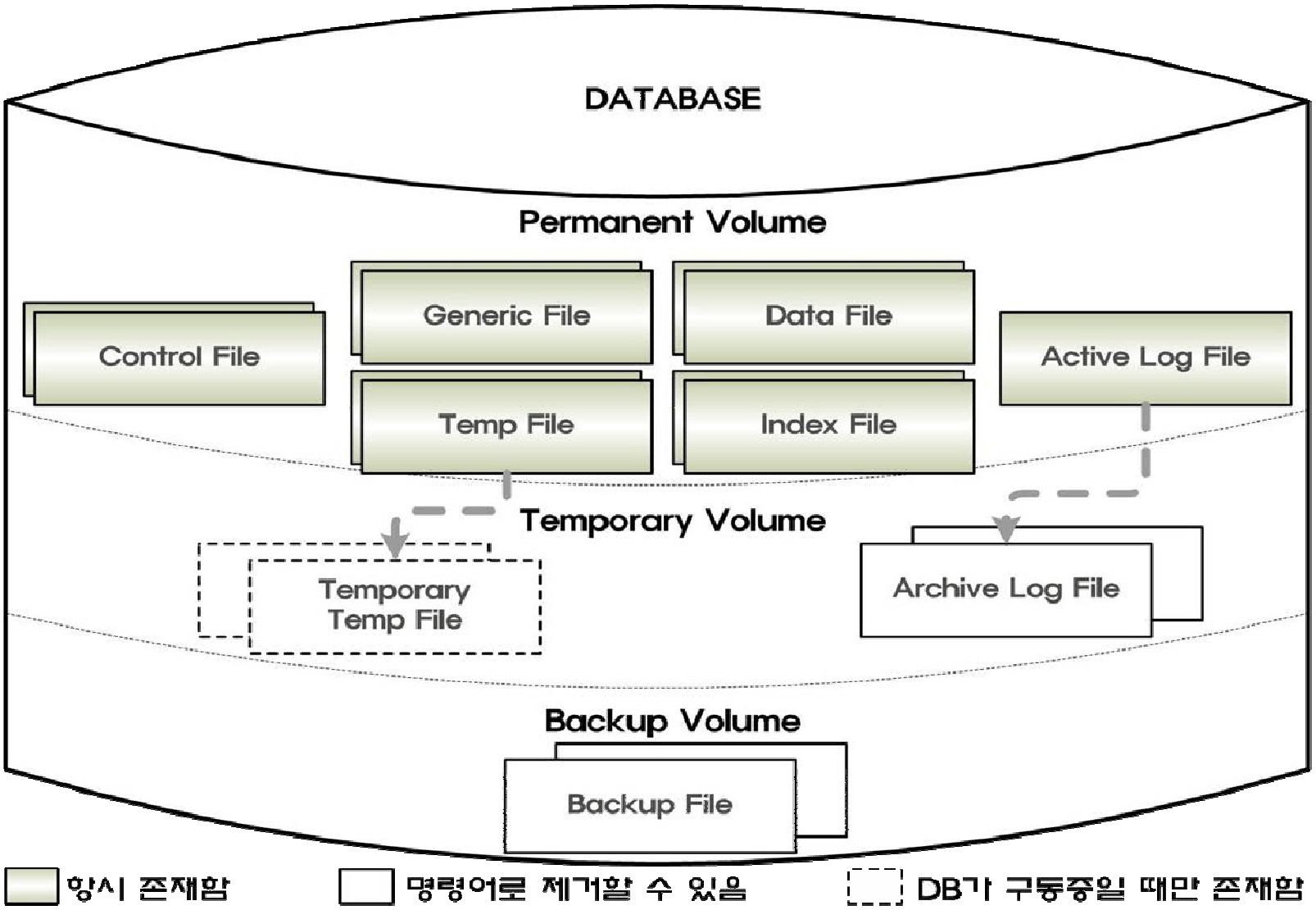
CUBRID
More than open source!

- Standalone Mode
  - 하나의 프로세스가 데이터베이스를 독점적으로 사용하는 구조이다.
  - 데이터베이스를 접근하는 프로세스는 서버와 클라이언트의 역할을 동시에 가진다.
  - 이 모드는 동시성 제어가 필요 없으므로 LOCK을 관리 하지 않으므로 처리 속도가 빠름
  - 테스트 환경 및 standalone모드에서 사용되는 utility를 사용할 경우 주로 이용



# 데이터베이스 구성 파일

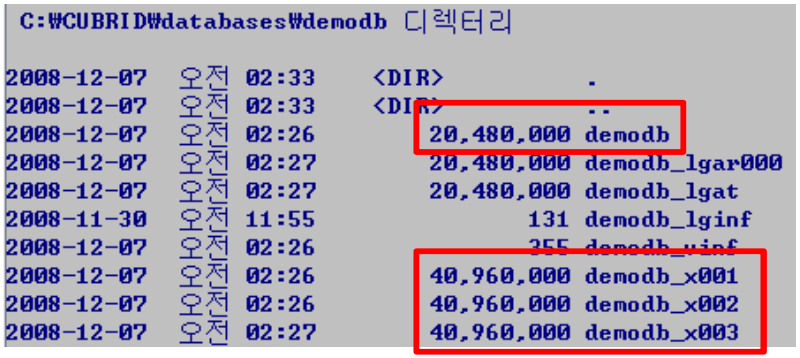
CUBRID
More than open source!



# 데이터베이스 볼륨

CUBRID
More than open source!

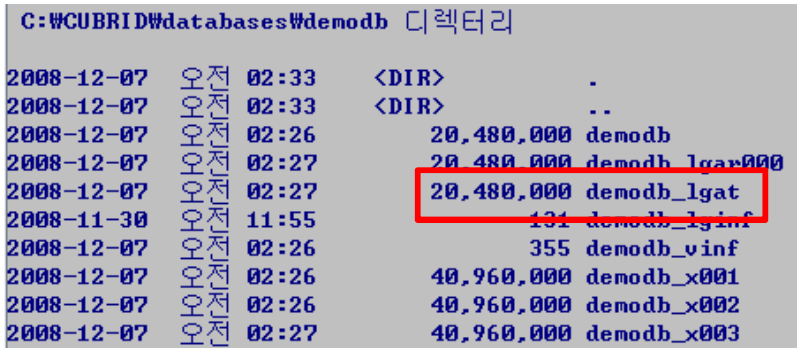
- Information Volumes
  - Information volume은 응용 프로그램의 모든 데이터를 저장한 파일이다.
  - Information volume은 생성할 때 그 사용 목적을 명시할 수 있다.
    - Data volume
      - 스키마, 레코드, 멀티미디어 데이터와 같은 응용 프로그램의 데이터를 저장
    - Index volume
      - 인덱스와 질의 처리 속도를 높이거나 무결성 제약을 보증하는 해쉬(hash) 같은 사용자 데이터를 지원하는 정보
    - Temp volume
      - 질의 처리(join, group by,order by, subquery, create index..)나 정렬(sorting)위해 사용. 이것은 임시 목적으로 사용되는 permanent 볼륨이며, 뒤에서 소개할 임시 볼륨과 혼동하지 않아야 한다.
    - Generic volume
      - DB 생성시 초기 볼륨이며 data, index, temp 볼륨으로도 사용 가능하다.



# 데이터베이스 볼륨 - 계속

CUBRID
More than open source!

- 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.
    - 예 : demodb_lgat(active log), demodb_lgar*(arcive log)



# 데이터베이스 볼륨 - 계속

CUBRID
More than open source!

- Control Information Volumes
  - Volume Information
    - 데이터베이스 볼륨에 관한 정보를 포함
    - 파일명 : demodb_vinf
  - Log Information
    - 로그 볼륨에 관한 정보를 포함
    - 파일명 : demodb_lginf
  - Backup Volume Information
    - 백업 볼륨에 관한 정보를 포함
    - 파일명 : demodb_bkvinf
  - databases.txt
    - 데이터베이스의 위치(디렉토리) 정보를 담고 있는 파일
  - cubrid.conf
    - CUBRID 시스템 파라미터의 설정값을 저장하는 파일

# 데이터베이스 볼륨 - 계속

CUBRID
More than open source!

- Volume Information file
  - 데이터베이스 볼륨에 대한 위치 정보 수록
  - 볼륨이 추가될 때 이 파일에 추가된 볼륨에 대한 정보도 기록
  - 이 파일은 사용자가 임의로 수정, 삭제, 이동할 수 없다.
  - 파일명은 {dbname}_vinf 형태임

```
-5 C:₩CUBRID₩databases₩demodb₩demodb_vinf
-4 C:₩CUBRID₩databases₩demodb₩demodb_lginf
-3 C:₩CUBRID₩databases₩demodb₩demodb_bkvinf
-2 C:₩CUBRID₩databases₩demodb₩demodb_lgat
 0 C:₩CUBRID₩databases₩demodb₩demodb
 1 C:₩CUBRID₩DATABA~1₩demodb₩demodb_x001
 2 C:₩CUBRID₩DATABA~1₩demodb₩demodb_x002
 3 C:₩CUBRID₩DATABA~1₩demodb₩demodb_x003
```

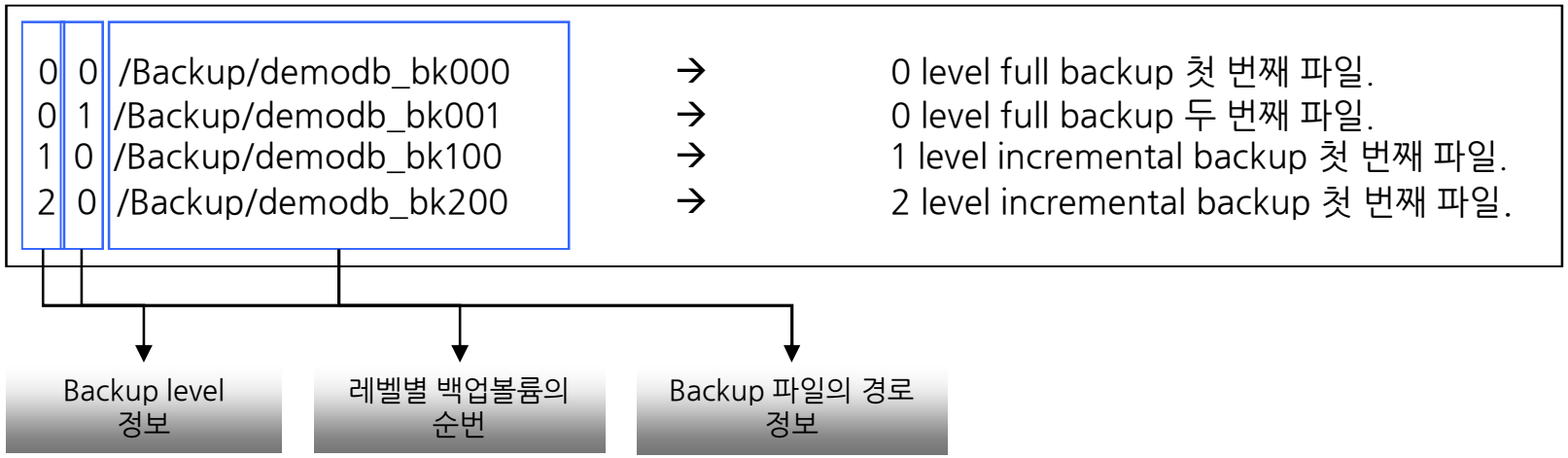
CUBRID
More than open source!

- Log Information file
  - 현재의 로그 및 archive 로그에 대한 정보 기록
  - 새로운 archive 로그파일 및 불필요한 archive 로그파일에 대한 정보를 기록한다.
  - 파일명은 {dbname}_lginf

```
COMMENT: CUBRID/LogInfo for database /CUBRID/databases/demodb
ACTIVE: /CUBRID/databases/demodb_lgat 5000 pages
활성화 되어 있는 로그파일에 대한 내용

ARCHIVE: 0  /CUBRID/databases/demodb_lgar000  0  4997
COMMENT: Log archive /CUBRID/databases/demodb_lgar000 is not needed any longer
unless a database media crash occurs.
로그 백업본인 로그 아카이브에 대한 내용
```

CUBRID
More than open source!

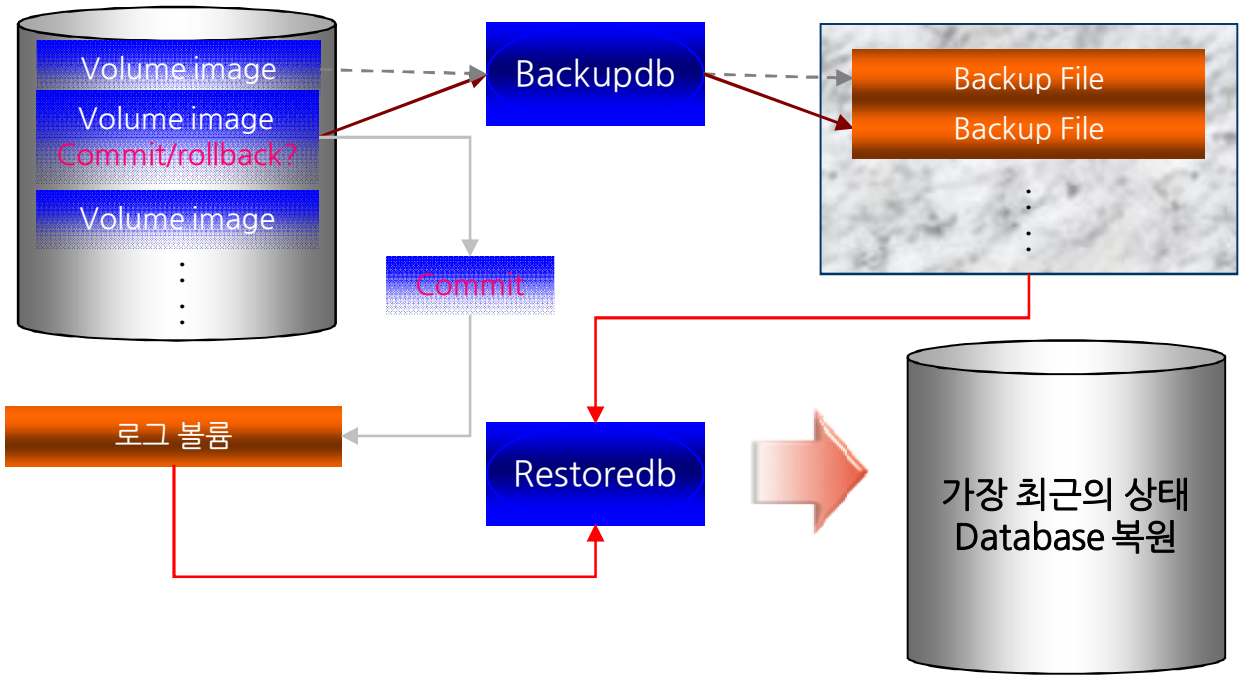
- Backup volume Information file
  - 백업 볼륨에 대한 위치 및 백업 정보 기록
  - 로그 파일과 같은 경로에 위치한다.
  - 파일명은 {dbname}_bkvinf



# 데이터베이스 볼륨 - 계속

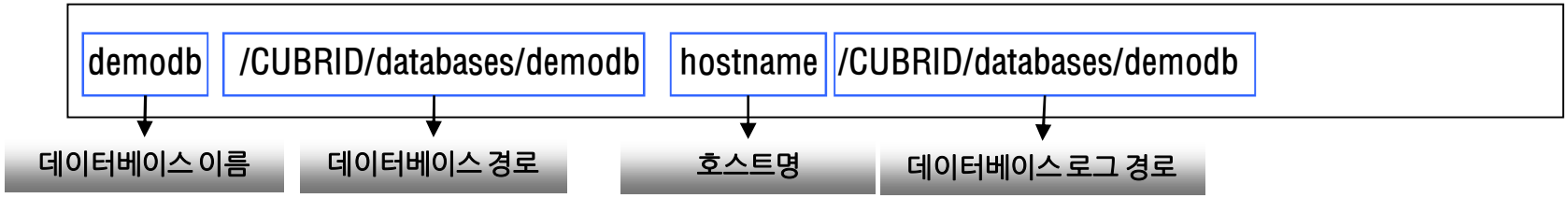
CUBRID
More than open source!

- Backup Volume
  - Backup 볼륨은 백업할 시기의 데이터베이스의 “fuzzy” 스냅샷이다.
  - 이 볼륨은 백업 작업을 하기 전까지는 생성되지 않는다.
  - 백업 볼륨을 이동하거나 이름을 변경하지 않으면 이후의 백업 시에 이전 볼륨을 덮어쓴다.
  - 이 볼륨은 데이터베이스를 가장 최근의 상태로 복구하기 위해서 로그 볼륨과 함께 사용된다.



# 데이터베이스 볼륨 - 계속

- databases.txt
  - 데이터베이스의 이름, 경로, 구축된 호스트이름 정보가 기록되어 있다.
  - 데이터베이스 생성 시, databases.txt 파일에 생성된 데이터베이스에 관한 정보가 기록된다.
  - CUBRID_DATABASES 환경 변수가 명시되어 있는 경로에 저장된다.
  - 환경변수가 지시하는 디렉토리에 존재하지 않는 경우는 현재 디렉토리를 참조한다.
  - 이 파일에 대한 주의 사항
    - 호스트 이름이 변경되거나, 데이터베이스를 rm과 같은 명령어로 삭제했을 경우 이 파일도 수정해야 한다.
    - 데이터베이스 생성 및 삭제 시에 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 디렉토리에 사용자 쓰기 권한을 허용하거나 사용자 각자의 디렉토리에 databases.txt 파일을 생성하고 환경 변수를 설정하는 것이 바람직하다.



# 데이터베이스 볼륨 - 계속

CUBRID
More than open source!

- cubrid.conf
  - cubrid.conf 파일은 CUBRID 시스템 파라미터의 설정값을 저장하는 파일이다.
  - 이 설정 값은 메모리, 파일 크기, 파일 위치 등의 데이터베이스의 기능과 관계된 중요 변수 값을 저장한다.
  - 이 파일은 환경변수 $CUBRID/conf 경로에 존재한다.

# 데이터베이스 프로세스 관리

CUBRID
More than open source!

# 프로세스 제어

CUBRID
More than open source!

- Master Process
  - 클라이언트와 서버간의 연결을 설정한다.
  - 클라이언트는 databases.txt에서 접속할 서버의 호스트를 알아낸 후, 마스터 프로세스에게 연결을 요청한다.
  - 마스터 프로세스는 서버 구동 시 자동으로 구동된다. 서버가 구동될 호스트에는 서버 수행 전에 master process가 구동되어 있어야 하기 때문이다.

# 프로세스 제어 - 계속

CUBRID
More than open source!

- Server Process
  - 서버 프로세스는 데이터베이스당 하나가 동작한다.
  - 서버 프로세스가 정상적으로 동작하기 위해서는 마스터 프로세스가 동작 중이어야 한다.
  - client/server 모드로 동작하기 위해서는 서버 프로세스를 구동해야 한다.
  - 서버 프로세스 구동
    - 마스터가 존재하지 않으면 마스터를 먼저 구동시킨다.
    - 이전에 비정상적으로 서버가 종료된 경우에 restart recovery가 수행된다.

    (경우에 따라 많은 시간이 경과됨)
  - 서버 프로세스 종료
    - 현재 진행중인 트랜잭션 취소 후 종료된다.
    - kill 명령으로 서버를 종료 시키지 말 것.
    - 부득이한 경우에는 csql –S database로 데이터베이스를 회복시킬 것. 이때 회복 작업은 절대 중단시키지 말 것.
  - 서버프로세스가 구동중인 상태에서 장비가 down되기 전 서버프로세스를 종료하여야 한다. 데이터베이스내의 트랜잭션이 정리가 안된 상태에서 종료되기 때문에 로그가 손상될 수 있기 때문이다.

# 큐브리드 서비스 관리

CUBRID
More than open source!

- CUBRID 서비스 구동
  - Windows
    - 자동으로 구동됨
    - 수동구동

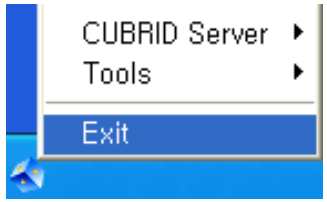

  - 명령어 이용
    - cubrid service 명령어를 이용

```
[cubrid@edu ~]% cubrid service start
@ cubrid master start
++ cubrid master is running.
@ cubrid broker start
++ cubrid broker start: success
@ cubrid manager server start
++ cubrid manager server start: success
```

# 큐브리드 서비스 관리 - 계속

CUBRID
More than open source!

- CUBRID 서비스 중지
  - Windows


  - 명령어 이용
    - cubrid service 명령어를 이용

```
[cubrid@edu ~]% 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
```

# 서버 프로세스 관리 - 계속

CUBRID
More than open source!

- 서버 프로세스 구동
  - 큐브리드 매니저를 이용한 구동


  - 명령어 이용 : % cubrid server start database_name
    - 로그파일 : $CUBRID/log/server/<데이터베이스이름>_yyyymmdd_hhmm.err
- 서버프로세스 종료
  - 큐브리드 매니저를 이용한 구동


  - 명령어 이용 : % cubrid server stop database_name

# 서버 프로세스 관리 - 계속

CUBRID
More than open source!

- 서버프로세스 상태확인
  - 구동 중인 데이터베이스 서버 리스트 출력 및 서버/마스터 종료
  - 사용방법 : cub_commdb [Options]

```
valid options:
  -P, --server-list                print server processes
  -R, --repl-list                  print replication processes
  -O, --all-list                   print all processes
  -S, --shutdown-server=NAME       shutdown NAME database server
  -K, --shutdown-repl-server=NAME  shutdown NAME replication server
  -k, --shutdown-repl-agent=NAME   shutdown NAME replication agent
  -A, --shutdown-all               shutdown all processes
  -h, --host=HOST                  connect to the master server on the given HOST
```

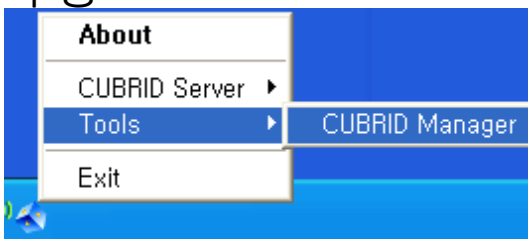
**- 서버 구동확인**

```
[cubrid@edu demodb]% cub_commdb -P
 Server demodb (rel 8.4, pid 8427)
```

**- 마스터 및 모든 서버 종료**

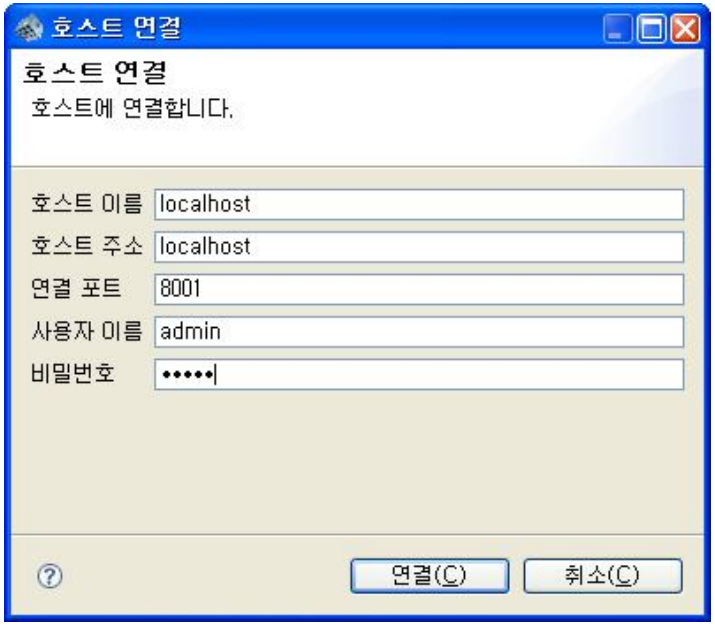
```
[cubrid@edu demodb]% cub_commdb -A
[cubrid@edu demodb]% cub_commdb -P
Could not connect to master server on edu
```

- CUBRID Manager Client
  - 구동

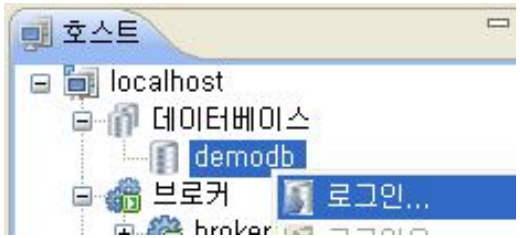


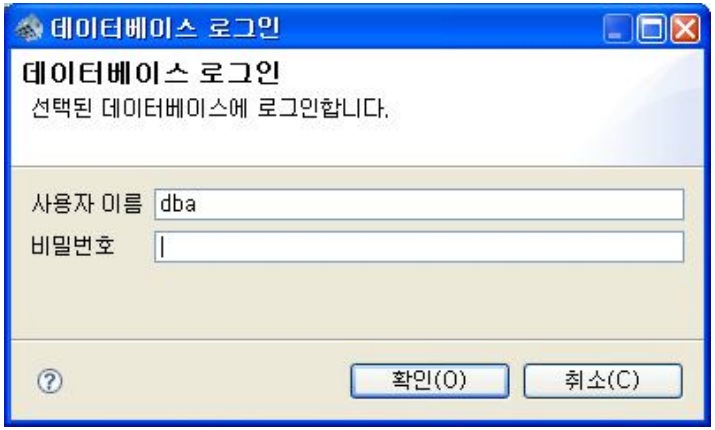


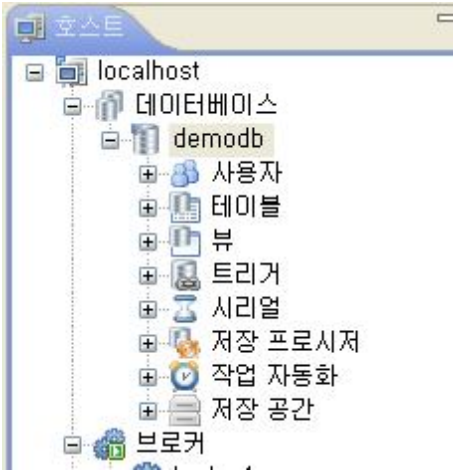
  - 로그인

- CUBRID Manager Client
  - 데이터베이스 로그인
    - 데이터베이스별 데이터베이스 사용자로 로그인

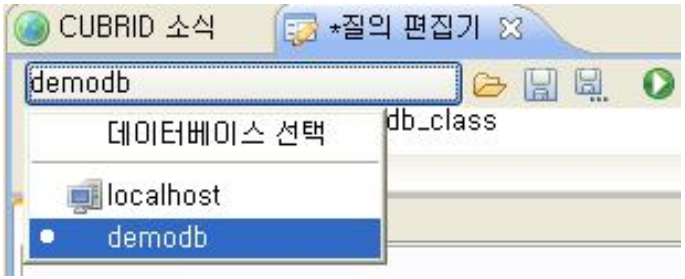


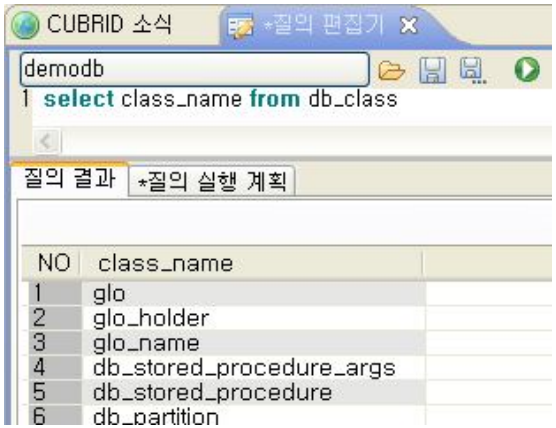



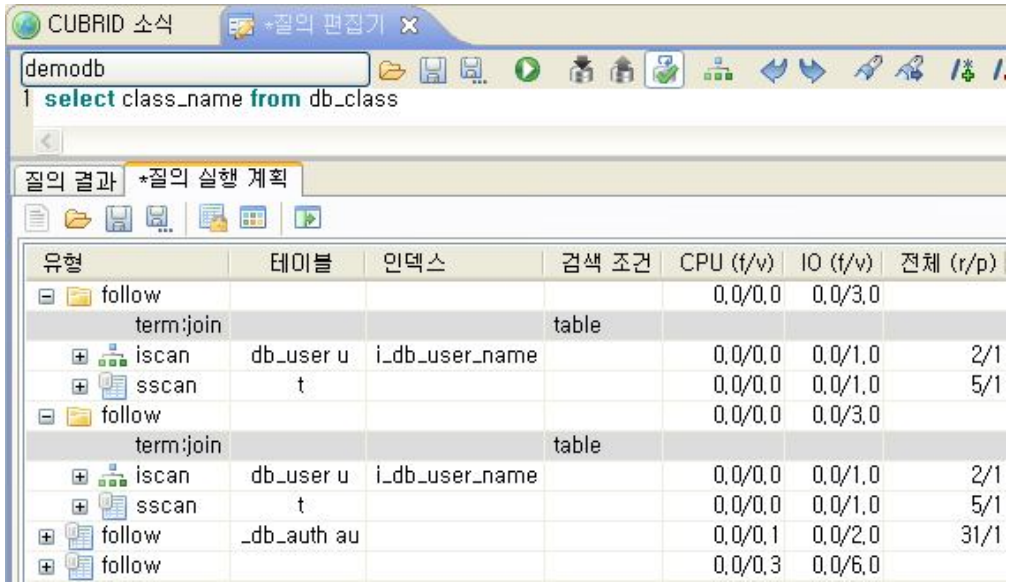
- 질의 편집기 사용
  - 데이터베이스 서버가 구동되어있어야 함
  - 상단의  를 클릭하거나, 데이터베이스 우클릭을 통하여 질의편집기 실행
    - 데이터베이스 로그인 사용자로 로그인되므로 주의(dba로 로그인)


  - 질의편집기에서 데이터베이스 선택 가능



- 질의 편집기 사용
  - 주요 명령
    - 질의수행: F5  질의수행계획보기: F6 
    - 편집취소 : ctrl-Z  편집복구 : ctrl-Y 
    - 자동커밋: ON  OFF 
    - 커밋:  롤백: 



# 데이터베이스 생성

CUBRID
More than open source!

# 데이터베이스 생성

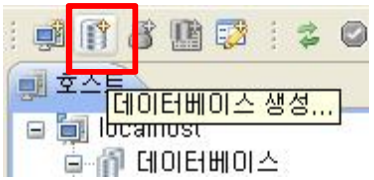
CUBRID
More than open source!

- 데이터베이스 생성시 고려 사항
  - 데이터베이스 이름 (최대 128자)
  - 데이베이스 크기
    - 저장될 데이터의 크기 : 데이터 보전 연한 고려
    - 저장될 인덱스의 크기 : 산정 어려울 경우 데이터의 20 ~ 30% 수준
    - 질의 수행에 필요한 질의 처리용 볼륨(템프 볼륨) : 데이터의 20 ~ 30% 수준
      - 운영중 모니터링을 통한 부족 여부 확인
    - 사용할 로그의 크기 : 일반적으로 10만 페이지의 크기
      - 업무 시간중 확장되지 않는 적정 크기
      - 주기적인 백업 수행을 통한 로그 정리
    - 실 데이터베이스 크기 : 계산된 크기의 2-3배
  - 데이터베이스 위치
    - 디스크 I/O bottle-neck 최소화
      - 데이터와 로그 분리

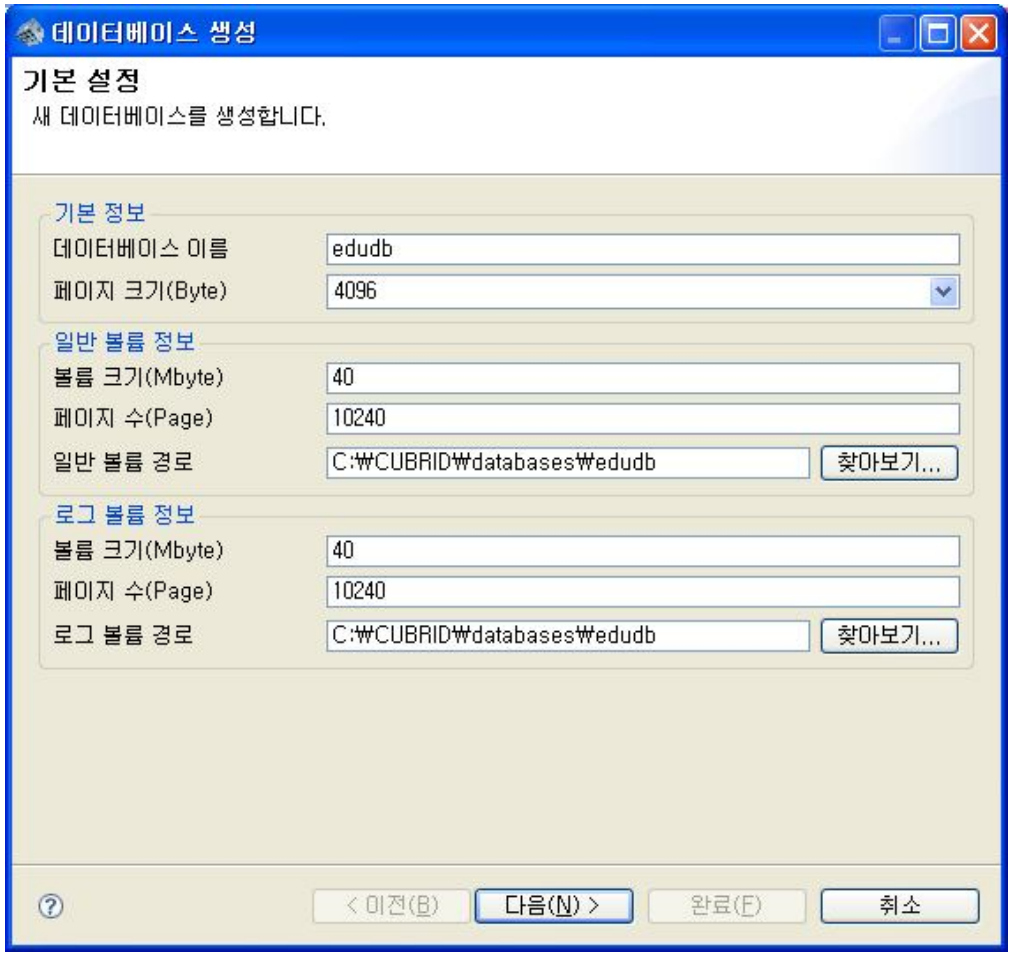
# 데이터베이스 생성 - 계속

CUBRID
More than open source!

- 데이터베이스 생성



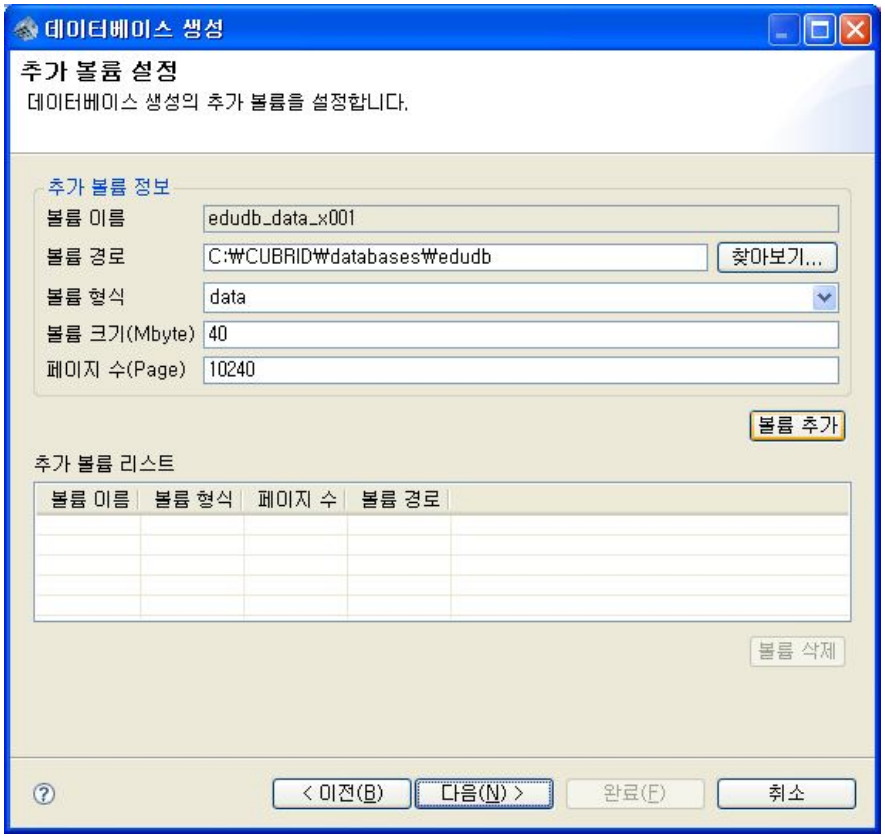

  - 데이터베이스 이름 지정
  - 범용볼륨 정보(20Mbyte = 5,000* 4Kbyte)
    - 기본페이지 수 5,000
    - 페이지 수 4096bytes
    - 초기 데이터베이스 경로
  - 로그볼륨정보(40Mbyte=10,000*4Kbyte)
    - 기본페이지 수 10,000
    - 로그볼륨 경로

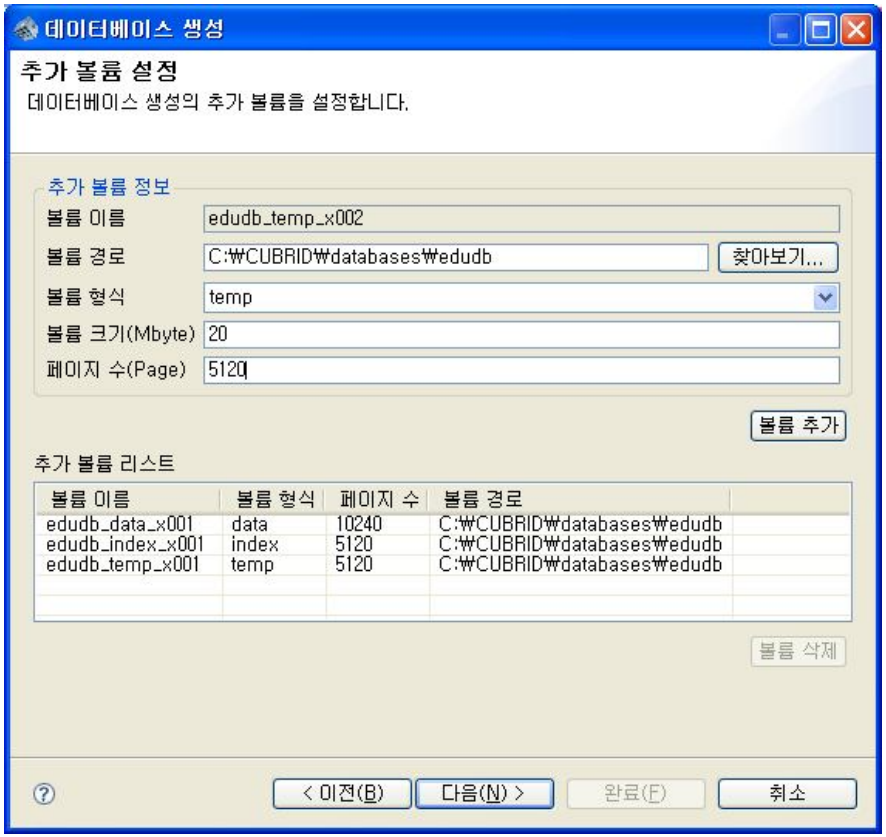


# 데이터베이스 생성 - 계속

CUBRID
More than open source!

- 볼륨 추가
  - 데이터 / 인덱스 / 질의처리공간, 용도별 추가

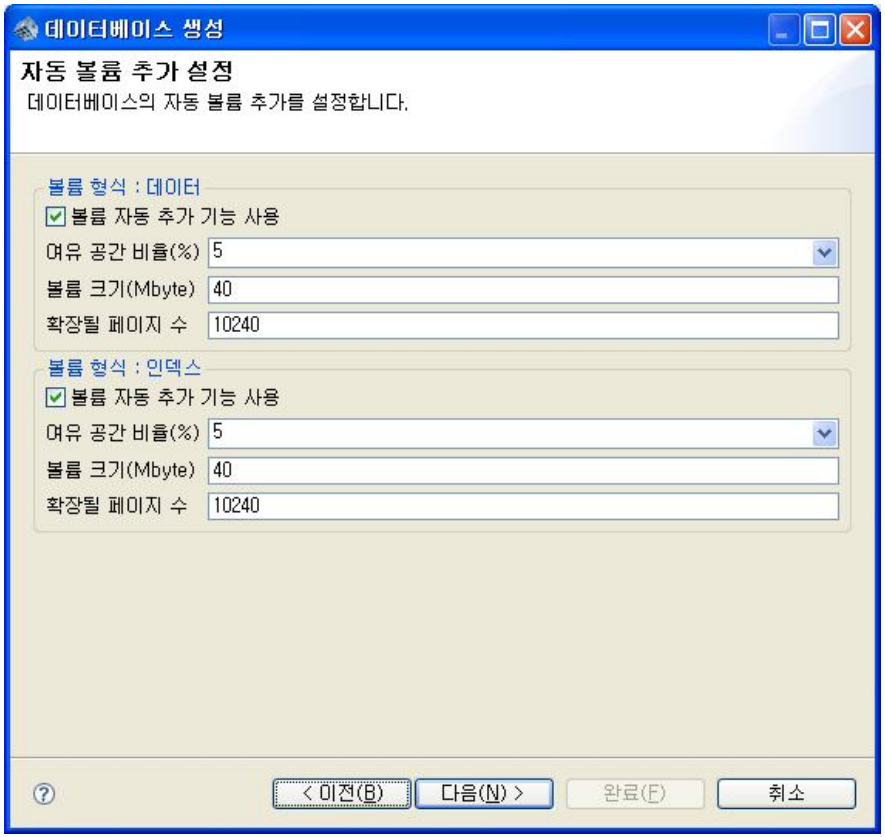


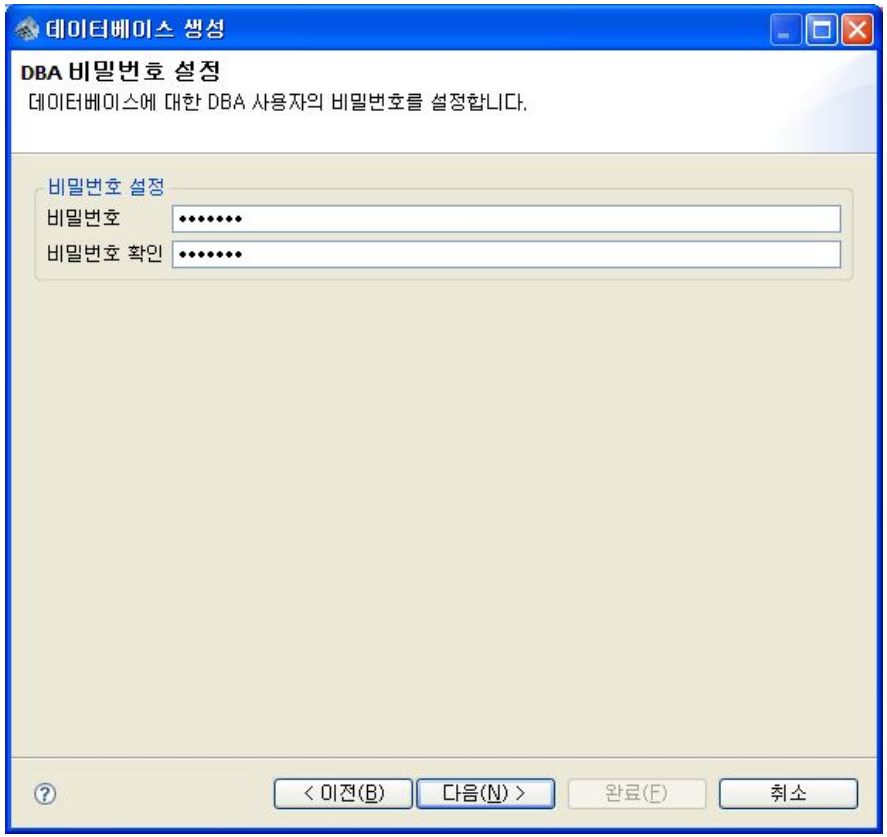



# 데이터베이스 생성 - 계속

CUBRID
More than open source!

- 볼륨 자동 추가 설정
- dba 암호 설정

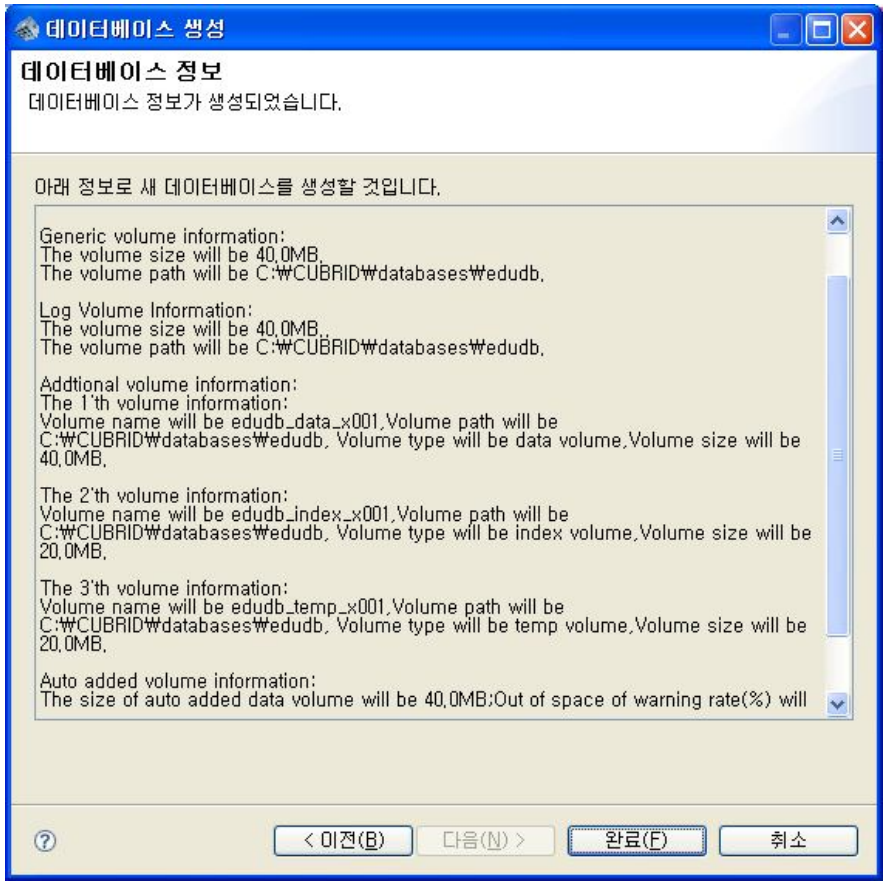


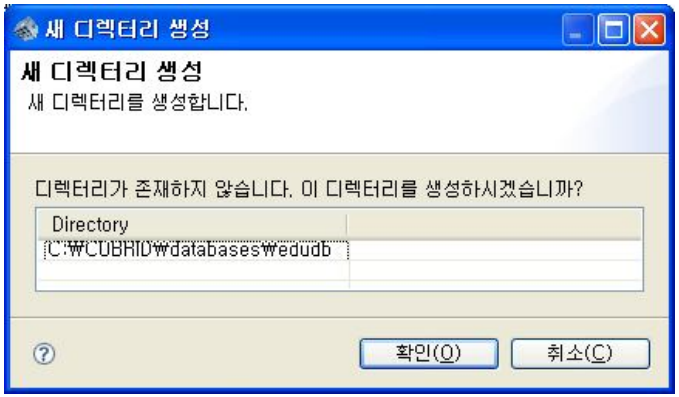



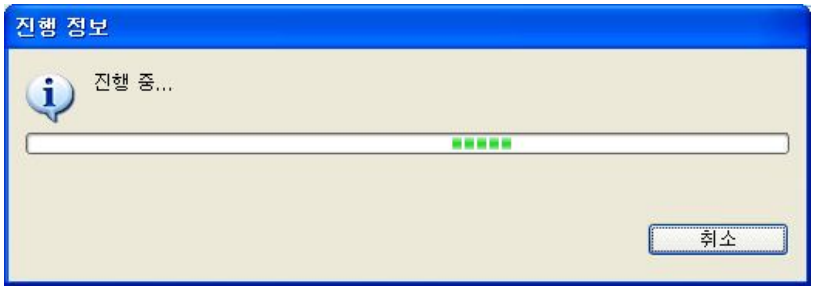
# 데이터베이스 생성 - 계속

CUBRID
More than open source!

- 내용 확인 및 생성



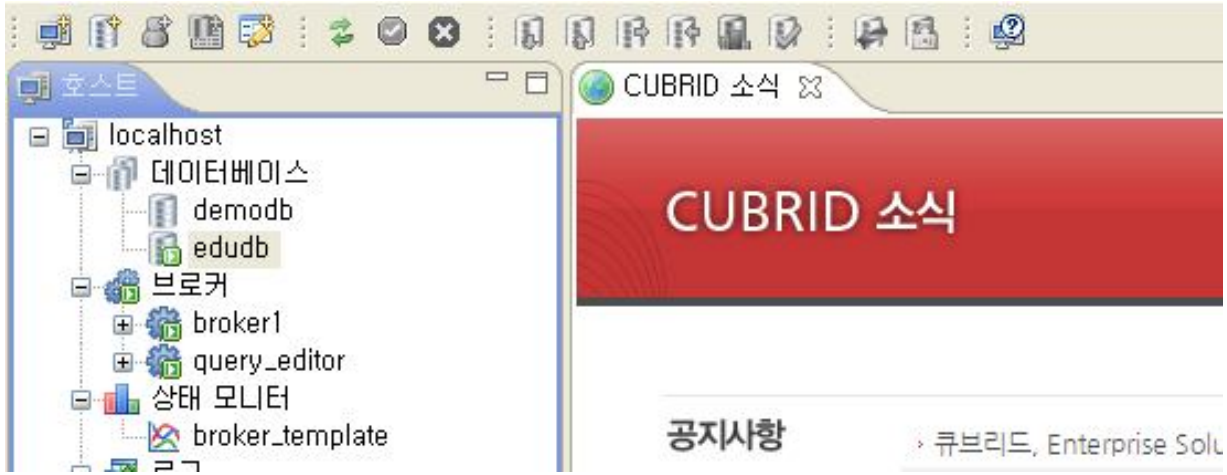



# 데이터베이스 생성 - 계속

CUBRID
More than open source!

- 생성 확인
  - 생성 완료후, 데이터베이스 서버 자동 구동



CUBRID
More than open source!

- 명령어 사용 : createdb
  - '#' 문자로 시작하는 데이터베이스명을 부여할 수 없다.

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

| 옵션 | 인 자 | 설 명 | 초기값 |
|---|---|---|---|
| -F | file-path | 초기 볼륨이 위치할 경로 지정 | 현재 |
| -L | log-file-path | 로그 볼륨이 위치할 경로 지정 | 현재 |
| -p | integer | 초기 볼륨의 페이지 수 지정 | 5000 |
| -l | log-page | 로그 볼륨의 페이지 수 지정 | 5000 |
| -s | Number | 페이지 크기(Byte) | 4096 |
| -r | | 데이터베이스가 이미 존재하는 경우 기존 데이터베이스를 삭제하고 재생성함. | 존재하면 에러 발생 |
| -v | | 작업 내용 출력 | 수행 않음 |

CUBRID
More than open source!

## ● 명령어를 사용한 데이터베이스 생성

– '#' 문자로 시작하는 데이터베이스명을 부여할 수 없다.

```
% cd ₩
% mkdir edudb
% cd edudb

% cubrid createdb edudb
Creating database with 5000 pages.
➔ 디폴트 설정(데이터베이스경로, 로그경로, pagesize, page수, 로그페이지크기)

CUBRID 2008 R1.4
```

- 생성파일
  - 초기생성 되는 generic볼륨 : edudb
  - 로그 정보 파일 : edudb_lginf
  - active로그볼륨 : edudb_lgat
  - 볼륨정보파일 : edudb_vinf

- 변경파일 : databases.txt

```
edudb   C:₩edudb mycom C:₩edudb
demodb C:₩CUBRID₩DATABA~1₩demodb mycom C:₩CUBRID₩DATABA~1₩demodb
```

# 볼륨확장

CUBRID
More than open source!

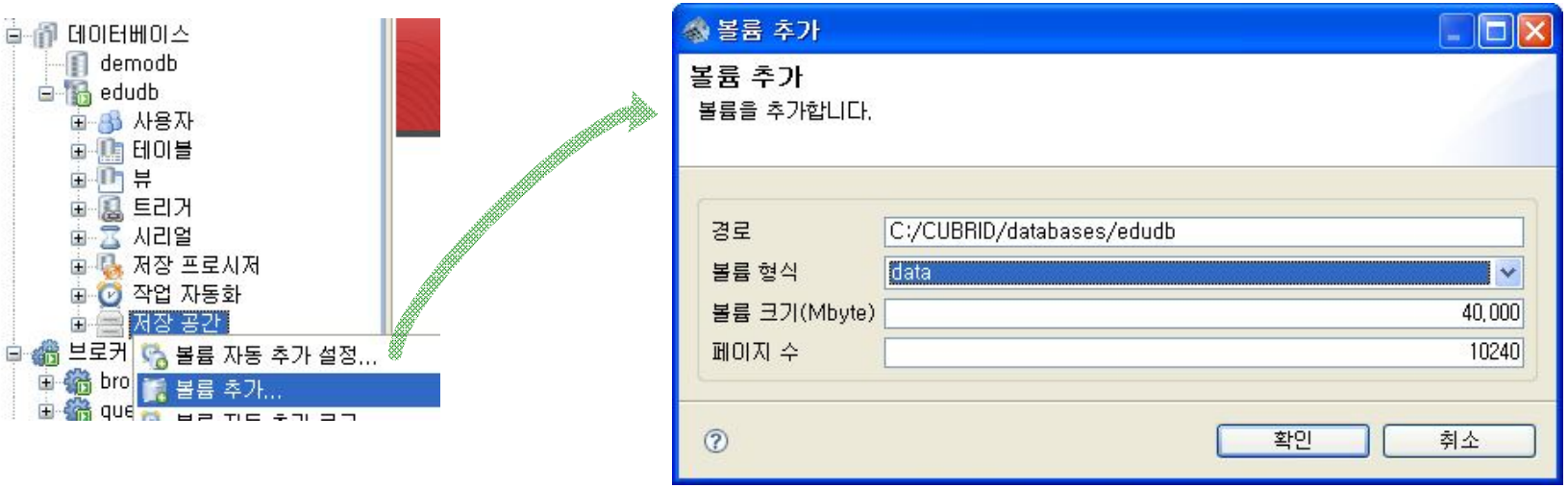
- 왜 추가해서 사용하는가?
  - 데이터베이스가 초기화될 때 생성되는 generic 볼륨 하나로 운영하는 것보다 용도별 볼륨을 분산하는 것이 성능상 효율적이다.
  - 운영상 적합한 크기를 예측하여 사용용도별 볼륨을 확장하는 것이 필요하다.
- 볼륨추가 전 주의사항
  - 동시에 사용될 가능성이 높은 볼륨은 물리적으로 분산배치 하는 것을 권장한다.

    (예를 들면, data와 log, data와 index, 그리고 data와 temp)
  - permanent temp 볼륨의 부족으로 temporary temp 볼륨이 자동 생성되는 것도 성능을 저하시키므로 permanent temp 볼륨의 양을 충분하게 미리 설정해야 한다.

# 볼륨확장 - 계속

CUBRID
More than open source!

- 데이터베이스 트리에서 저장공간의 볼륨에서 볼륨추가를 선택




generic : 어떤 용도로도 사용 가능

data : 테이블, 레코드, 멀티미디어 등의 데이터 저장

index : 인덱스 정보 저장

temp : 질의 처리나 정렬할 때 사용

➢ 일반적으로 generic은 별도 추가하여 사용하지 않는다.

CUBRID
More than open source!

- 명령어를 이용한 볼륨생성 : addvoldb

| 옵션 | 인자 | 설명 | 기본값 |
|---|---|---|---|
| -S | -C | stand-alone, client-server mode 지정 | |
| -F | vol_path | 볼륨이 저장될 경로 지정 | 데이터베이스 경로 |
| -p | vol_purpose | 볼륨 용도 지정(generic, data, index, temp) | generic |
| -n | vol_name | 볼륨의 이름 | DB_x번호 |

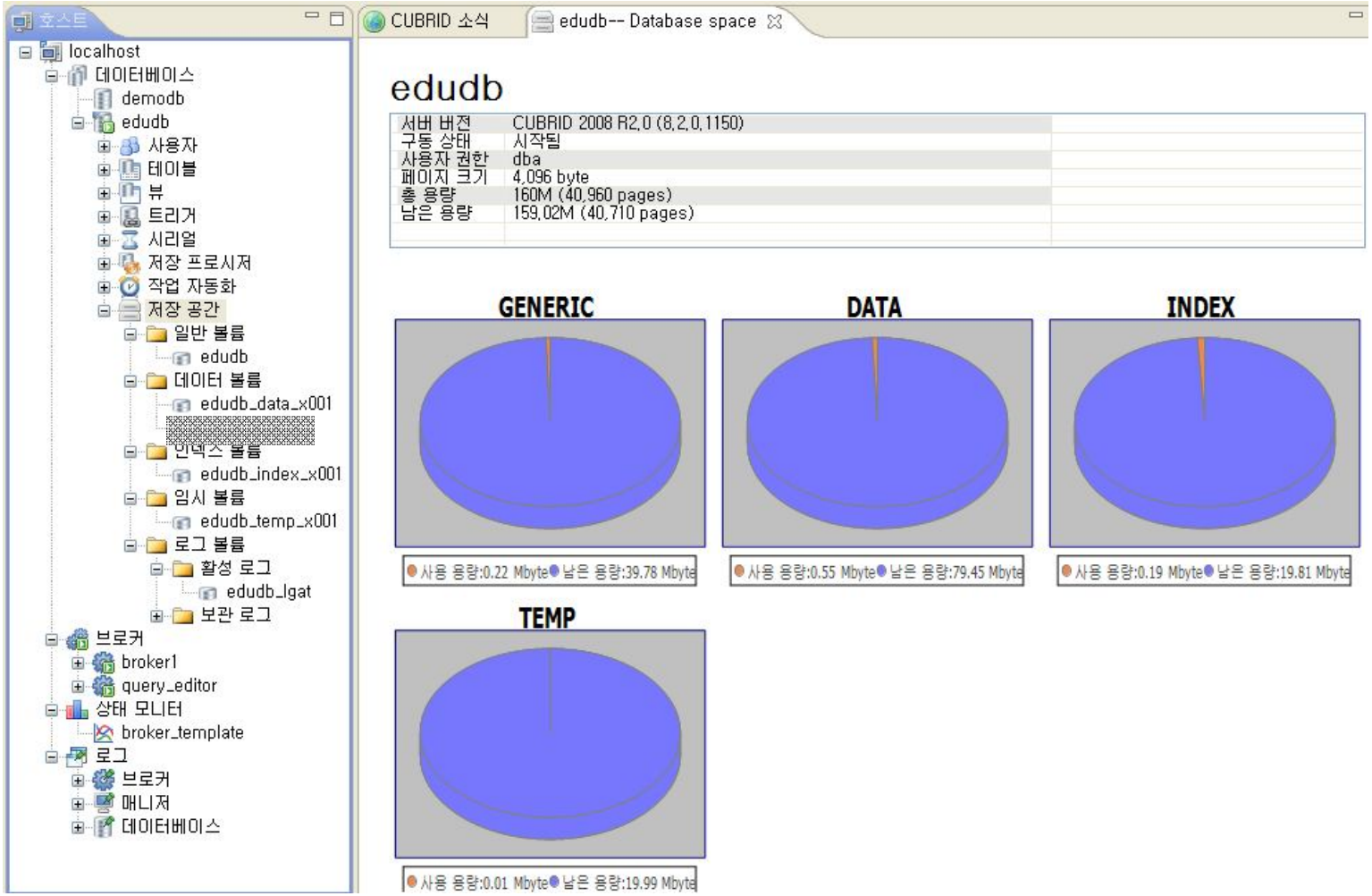
```
% cubrid addvoldb   [options]   database_name   number_of_pages
```

```
% cubrid addvoldb -S -p data edudb 500000

CUBRID 2008 R1.4
```

# 볼륨확장 - 계속

CUBRID
More than open source!

- 볼륨확장 확인



CUBRID
More than open source!

## 명령어를 이용한 볼륨확인 : spacedb

```
% cubrid spacedb  [-S | -C]  database_name
% cubrid spacedb -S edudb
```

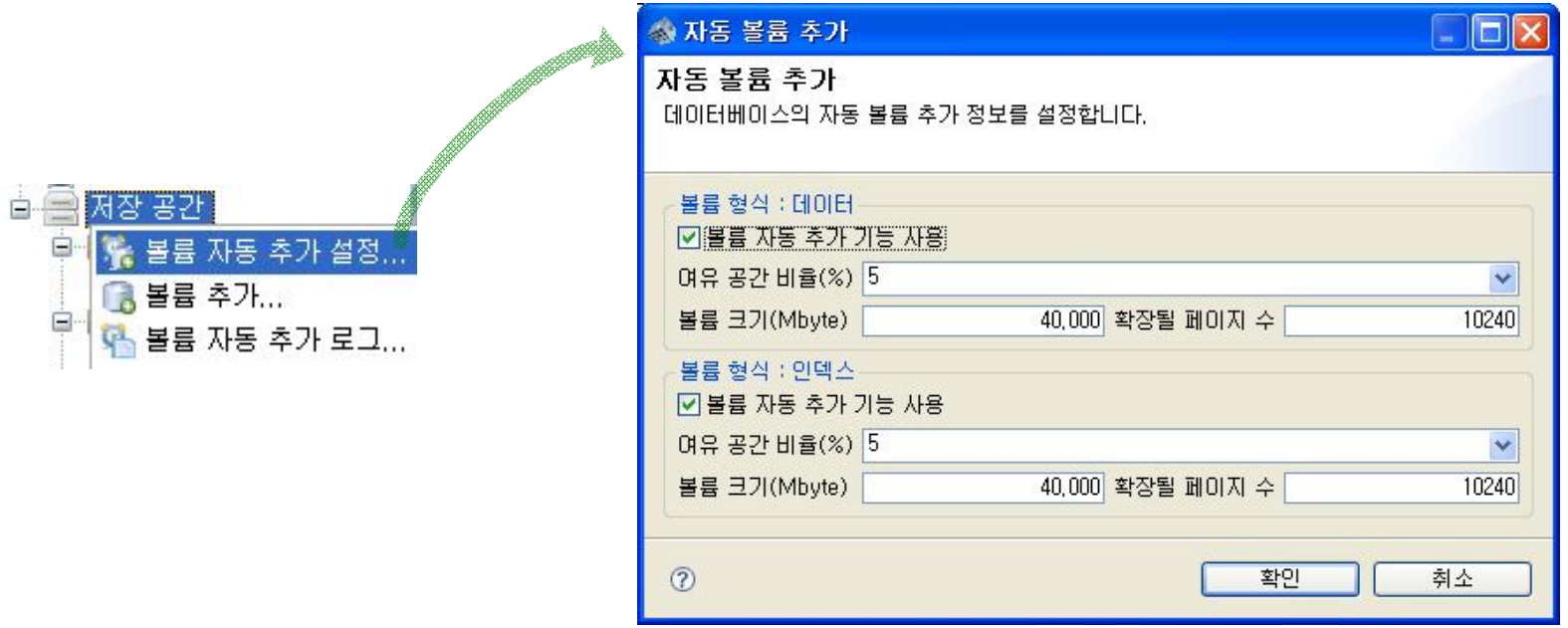
- 각 볼륨의 용도
- 데이터베이스 page 크기
- 각 볼륨의 총 사용량과 여유 공간 크기
- 데이터베이스 총 사용량과 여유 페이지 크기

```
% cubrid spacedb -S edudb

CUBRID 2008 R1.4

Space description for database 'edudb' with pagesize 4096.

Volid  Purpose  total_pages  free_pages  Vol Name

   0  GENERIC        5000        4781  C:₩edudb₩edudb
   1     DATA        5000        4997  C:₩edudb₩edudb_x001
------------------------------------------------------------------------------
   2                10000        9778
```

# 볼륨확장 - 계속

CUBRID
More than open source!

- 확장 자동화
  - Data, Index 볼륨의 여유 공간이 부족할 경우 볼륨 자동생성 설정
  - 확장될 볼륨의 기준(여유공간 비율) 및 자동확장이 되는 볼륨의 크기를 지정
  - Cubrid Manager에서 지원



# 실습 - 데이터베이스 생성

CUBRID
More than open source!

- 아래 조건에 맞는 데이터베이스를 생성
  - 각 볼륨별 생성 위치 및 크기
    - 페이지크기 : 4Kb
    - 데이터베이스명 : edudb2
    - 첫번째 볼륨 : 5,000p, C:₩CUBRID₩databases₩<데이터베이스이름>
    - 로그 볼륨 : 100,000p, C:₩CUBRID₩databases₩<데이터베이스이름>₩log
    - 데이터 볼륨 : 500,000p, C:₩CUBRID₩databases₩<데이터베이스이름>
    - 인덱스 볼륨 : 250,000p, C:₩CUBRID₩databases₩<데이터베이스이름>
    - 템프 볼륨 : 250,000p, C:₩CUBRID₩databases₩<데이터베이스이름>
- 생성된 볼륨 확인
  - databases.txt 내용 확인
  - volume information file 의 내용을 보고 각 디렉토리의 파일 확인
    - control volumes
    - information volumes
    - log volumes

# 데이터베이스 관리

CUBRID
More than open source!

# 데이터베이스 백업

CUBRID
More than open source!

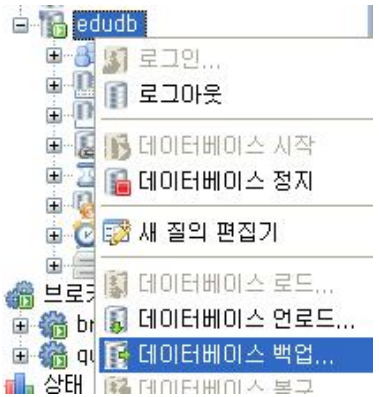
- 데이터베이스 백업 시 참고사항
  - 데이터베이스 백업은 데이터베이스의 이미지를 파일 등으로 덤프하는 것이다.
  - OS 유틸리티를 사용한 데이터베이스 백업은 권장하지 않는다.
  - 데이터베이스 백업 시점에 불필요한 로그 아카이브 볼륨들을 정리 할 수 있다.
  - 백업은 매일 받는 것을 권장하며, DBA가 수작업으로 하는 것보다는 자동 백업을 하는 것이 편리하다.
  - 백업 볼륨은 외부 저장장치에 백업하는 것이 안전하다.
  - 데이터베이스를 새로운 버전으로 migration했다면 새로 생성한 데이터베이스를 즉시 백업해야 한다. 이전 버전의 백업 볼륨은 새로운 버전 복구에 사용 불가능하기 때문이다.

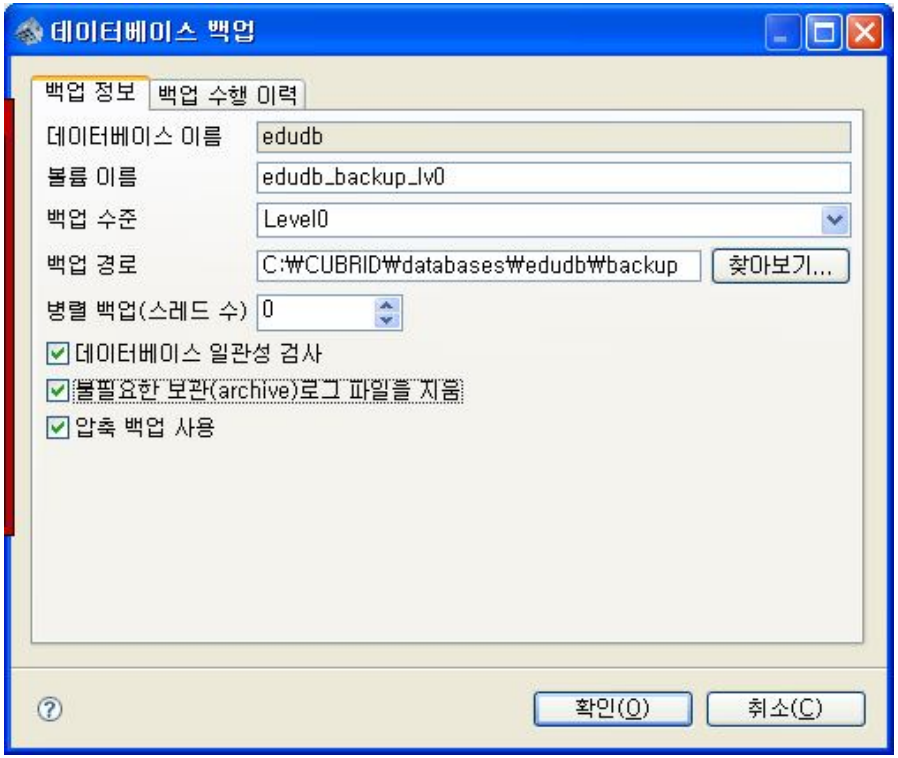
# 데이터베이스 백업 - 계속

CUBRID
More than open source!

- 데이터베이스 백업 정책
  - 백업할 데이터의 선택
    - 전체 데이터베이스 또는 일부만 백업할 것인가?
    - 데이터의 유효 또는 보존 기간은 얼마로 할 것인가?
    - 데이터베이스와 함께 백업되어야 할 다른 파일은 있는가?
  - 백업 방법
    - 사용 가능한 백업 툴 및 백업 장비
    - Full/incremental 백업(0, 1, 2 level)
    - on-line/off-line 백업
      - on-line백업 시 archive log 증가하며 백업 본과 같이 보관해야 완전 복구
  - 백업의 시기
    - 데이터베이스의 활동이 가장 적은 시기

# 데이터베이스 백업 - 계속




- 백업 볼륨의 이름을 명시한다.
- 백업 레벨을 선택한다.현재 상위 백업이 없을 경우 level0만 표시된다.
- 백업 레벨은 level2까지  이름을 있다.
- 백업 파일이 저장될 위치를 기록한다.
- 데이터베이스 일관성(consistency)를 체크한다.
- 백업 후 archive 로그 삭제를 할 경우 체크한다.
- 백업 시 사용할 thread 개수를 정한다. 0일 경우 자동으로 설정 된다.
- 데이터백업 수행 시 압축을 사용한다.

# 데이터베이스 백업 - 계속

CUBRID
More than open source!

- 명령어 이용 : backupdb

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

  - disk와 tape 등의 미디어 지정 가능
  - 백업볼륨파일 및 백업정보파일 생성

| 옵션 | 인자 | 설명 | 기본값 |
|---|---|---|---|
| -S \| -C | | stand-alone, client-server mode 지정 | -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 점검 | 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 |

# 데이터베이스 백업 - 계속

CUBRID
More than open source!

- 명령어를 이용한 데이터베이스 백업

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

```
% cubrid backupdb  -S demodb

CUBRID 2008 R1.4

Backup Volume Label: Level: 0, Unit: 0, Database demodb, Backup Time: Sun Jul 10 23:29:34 2009
```

# 실습 - 데이터베이스 백업

CUBRID
More than open source!

- 아래 조건에 맞는 데이터베이스를 백업
  - 백업의 대상 : demodb
  - 백업의 경로 : C:/CUBRID/databases/demodb/backup
  - 백업의 레벨 : 0 level
  - 보관로그 삭제
  - 압축사용
- 백업 확인
  - backup volume information file 의 내용 보고 디렉토리의 파일 확인
    - backup volume 및 backup information 파일
    - cubrid_manager client를 이용한 백업확인

# 데이터베이스 복구

CUBRID
More than open source!

- 데이터베이스 복구사례
  - 트랜잭션이 비정상 종료되어 로그가 손상된 경우
    - 비정상트랜잭션 회복하거나 로그복구

    ➔ CUBRID Manager에서 지원하지 않는다.
  - 디스크의 media failure가 발생할 경우
  - 데이터베이스를 특정시점으로 복원하여야 할 경우
    - 백업된 데이터베이스와 로그를 이용해서 복구한다. 이때 필요한 볼륨은 백업 볼륨과 archive log, active log이다.

# 데이터베이스 복구 - 계속

CUBRID
More than open source!

- 로그 복구
  - 데이터베이스 서버가 비정상적으로 종료된 경우, 예를 들면, 데이터베이스 서버를 kill 명령어로 중지시켰거나 시스템이 갑자기 셧다운 되거나 리부팅 된 경우에는 데이터베이스를 복구해야 한다. csql 인터프리터를 stand-alone 모드로 실행하면 복구가 된다. 단, 회복 작업 시간이 매우 길더라도 절대 중단해서는 안 된다.

```
% csql  -S  database_name
```

  - 치명적인 오류로 인해 로그를 복구하는 과정에서도 복구가 안될 경우 emergency_patchlog를 이용하여 로그를 복구하거나 재 생성하여 데이터베이스를 구동시킬 수 있다.
  - 위의 방법으로 복구 후에 csql를 수행하여도 정상구동이 되지 않으면 “-r”옵션을 사용하여 복구 한다.

```
% cubrid emergency_patchlog demodb
➔ emergency patch fail.
% cubrid emergency_patchlog -r demodb
```

# 실습 - 데이터베이스 복구

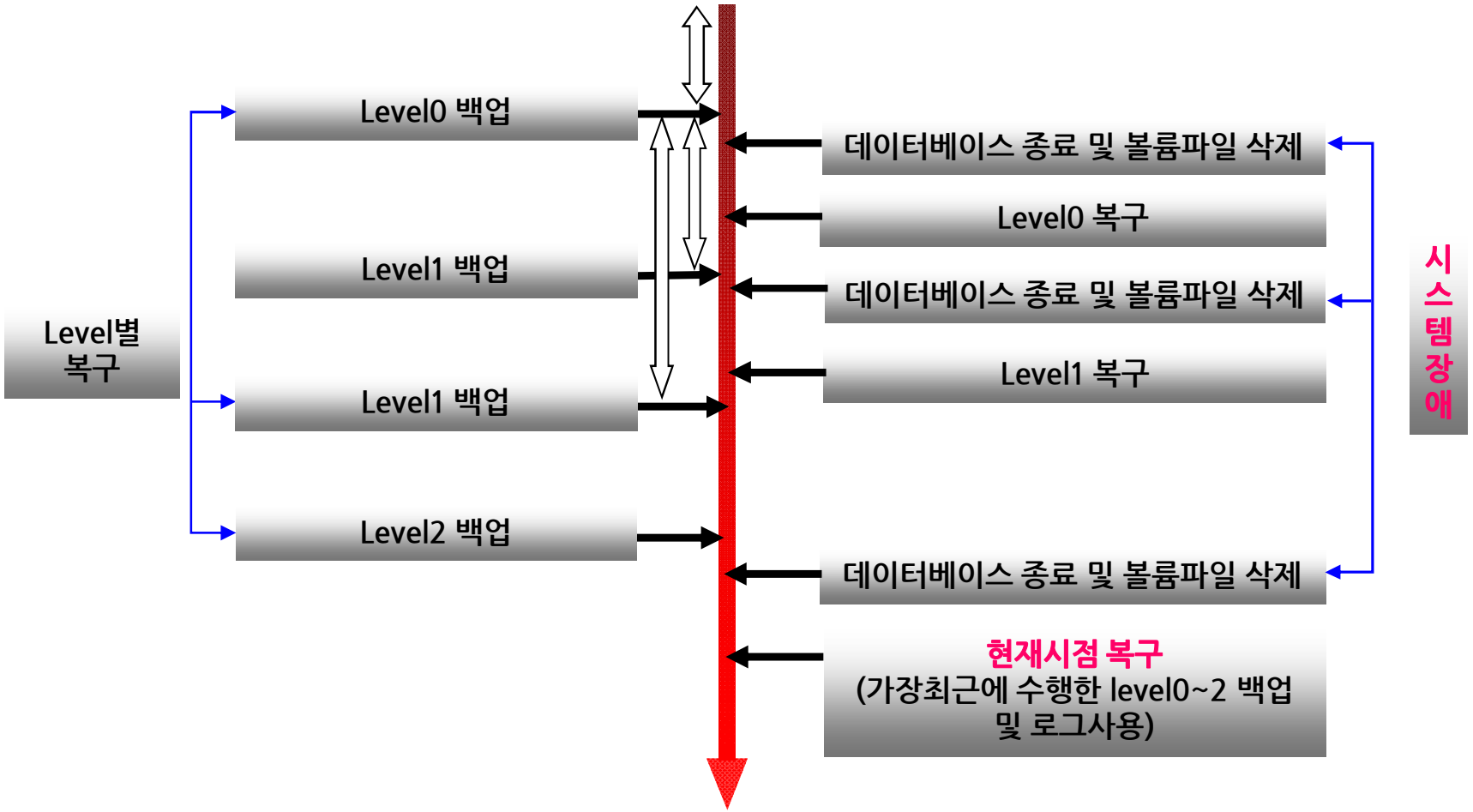
CUBRID
More than open source!

- 다음 순서에 의해 로그를 복구하여 데이터베이스 구동을 확인한다.
  - 대상데이터베이스명 : demodb
  - 데이터베이스 중지
  - 데이터베이스 엑티브로그 이동
    - demodb_lgat
  - 데이터베이스 구동 시도
  - 데이터베이스 엑티브로그 복구
    - emergency_patchlog 사용(-r 옵션사용)
  - 데이터베이스 구동

# 데이터베이스 복원

CUBRID
More than open source!

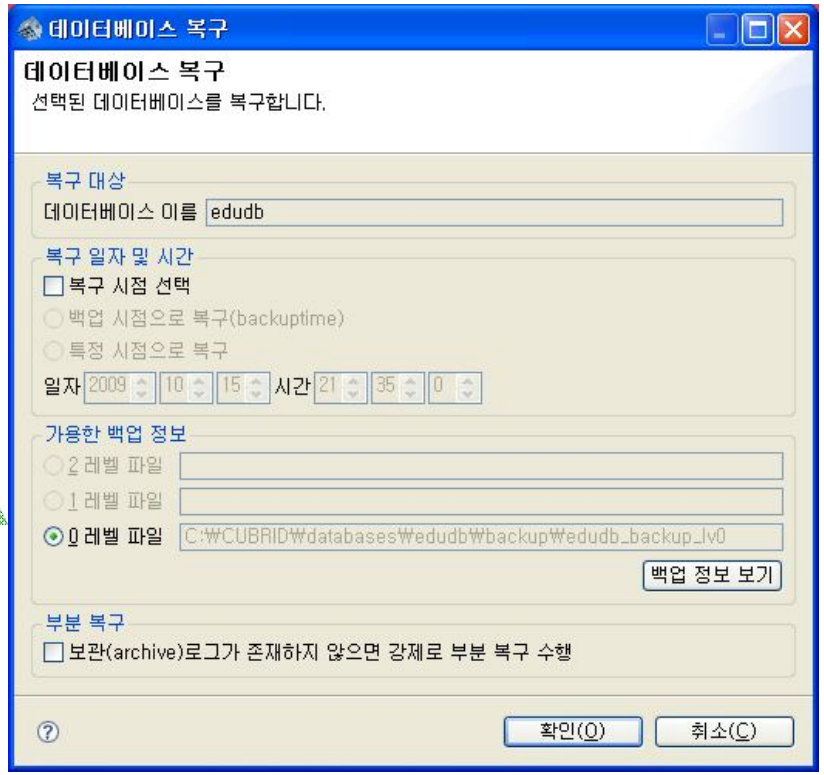
- 백업볼륨을 이용한 데이터베이스 복구시나리오



# 데이터베이스 복원 - 계속

CUBRID
More than open source!

- 데이터베이스 복구 방법
  - 데이터베이스 복구는 서버 종료 후 가능
  - default로 데이터베이스 현재시점 복구

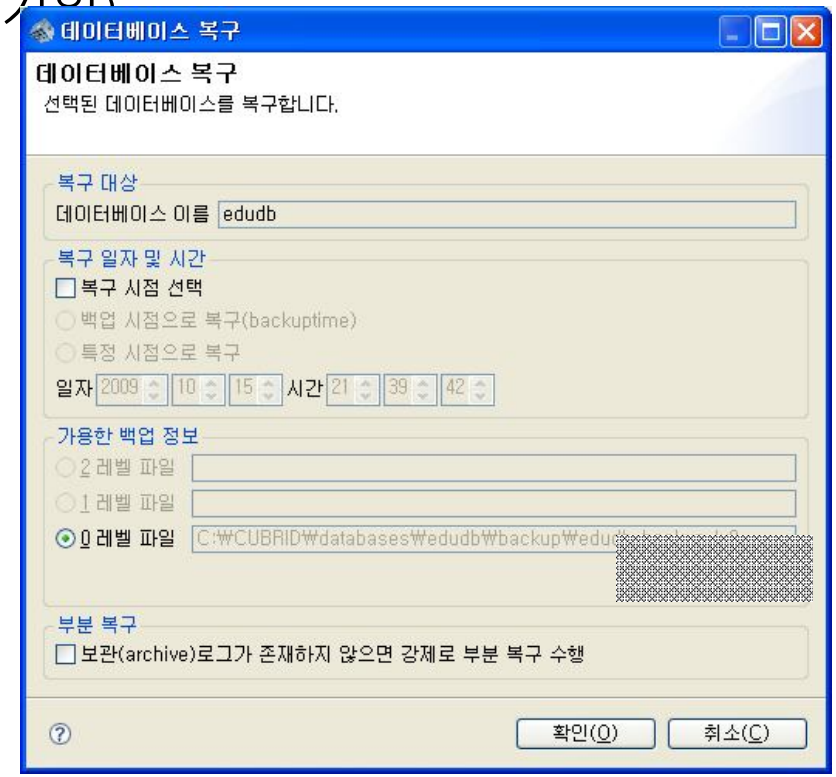
# 데이터베이스 복원 - 계속

CUBRID
More than open source!

- 백업정보
  - 백업을 수행한 정보(stand-alone 모드에서 가능)

Database Name: C:₩CUBRID₩DATABA~1₩demodb₩demodb
DB Creation Time: Tue Mar 31 22:36:32 2009
Pagesize: 4096
**Backup Level: 0 (FULL LEVEL)**
Start_lsa: -1|-1
Last_lsa: 4451|2076
**Backup Time: Sun Apr 10 23:29:34 2009**
**Backup Unit Num: 0**
Release: 8.1.4
Disk Version: 8
Backup Pagesize: 131072
Zip Method: 0 (NONE)
**Zip Level: 0 (NONE)**
**Include Active Log: YES**

Database Volume name: C:₩CUBRID₩DATABA~1₩demodb₩**demodb_vinf**
Volume Identifier: -5, Size: 223 bytes (1 pages) ➔ **볼륨 정보파일**
Database Volume name: C:₩CUBRID₩DATABA~1₩demodb₩**demodb**
Volume Identifier: 0, Size: 20480000 bytes (5000 pages) ➔ **구성 볼륨**
Database Volume name: C:₩CUBRID₩Databases₩demodb₩**demodb_lginf**
Volume Identifier: -4, Size: 131 bytes (1 pages) ➔ **로그 정보파일**
Database Volume name: C:₩CUBRID₩Databases₩demodb₩**demodb_lgat**
Volume Identifier: -2, Size: 20480000 bytes (5000 pages) ➔ **active 로그**



# 데이터베이스 복원 - 계속

- 명령어 사용 : 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<br>예) 21-12-2002:17:00:10.<br>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 | |

# 실습 - 데이터베이스 복구

CUBRID
More than open source!

- 다음 순서와 같이 데이터베이스를 복원한다.
  - 데이터베이스 중지
  - 백업파일을 제외한 데이터베이스 구성파일 이동
    - 데이터베이스 구성파일 : demodb*
  - restoredb명령을 이용하여 수행
    - 구성파일이 이동되므로 archive 로그 파일을 무시고 복원

# 데이터베이스 최적화

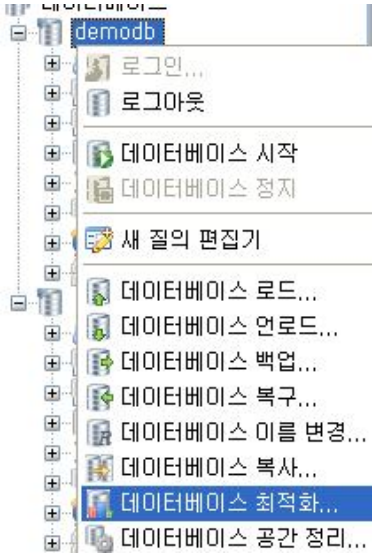
CUBRID
More than open source!

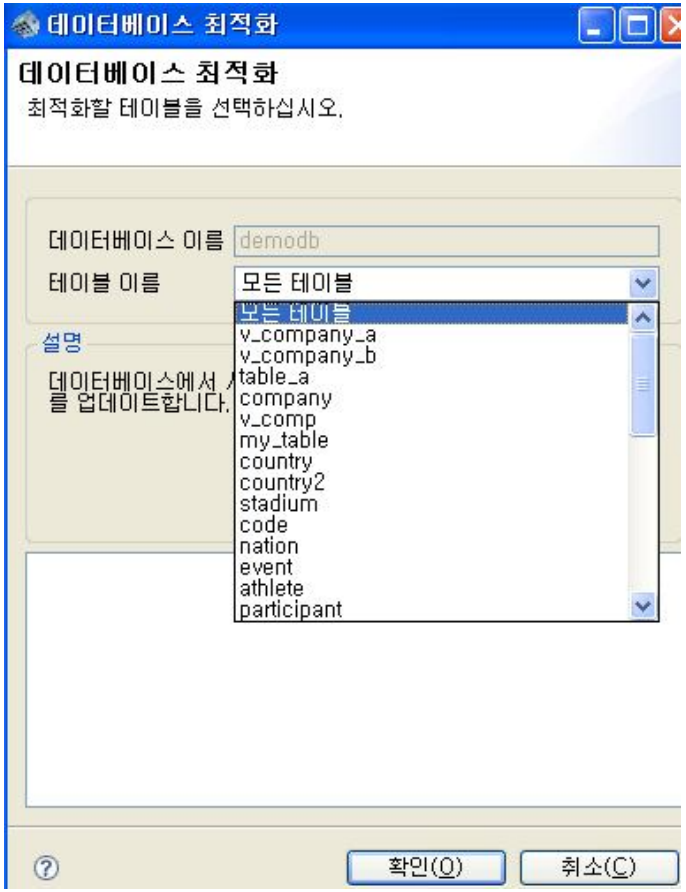
- 통계정보갱신
  - 큐브리드는 질의 실행 계획을 생성하기 위해 질의 최적화기를 사용한다.
  - 질의 최적화기는 테이블에 있는 레코드들의 수, 접근하는 페이지들의 수, 필드 값들의 분산 같은 통계 정보를 사용한다.(예: 필드들의 최소/최대 값)
  - 테이블(또는 데이터베이스)가 광범위하게 수정되었을 때, 질의 프로세스를 최적화하기 위해 optimizedb 명령을 사용할 수 있다.
  - SQL수준인 “UPDATE STATISTICS”문으로 모든 테이블와 특정 테이블에 대하여 통계정보를 갱신할 수 있다.
  - 데이터베이스 재구성을 완료 후 반드시 통계정보갱신을 수행하여야 한다.
  - 주기적 수행을 권장 한다.

# 데이터베이스 최적화 - 계속

CUBRID
More than open source!

- 통계정보갱신 방법




- 데이터베이스 최적화 선택
- “모든테이블”의 경우 모든 테이블의 통계정보를 갱신
- 테이블 선택시 선택한 테이블의 통계정보를 갱신

# 데이터베이스 최적화 - 계속

CUBRID
More than open source!

- 명령어 이용 : optimizedb

```
% cubrid optimizedb   [options]   database_name
```

  - stand-alone 상태에서 수행 가능
  - SQL 을 이용하여 수행 가능

```
update statistics on all classes;
```

# 데이터베이스 정리

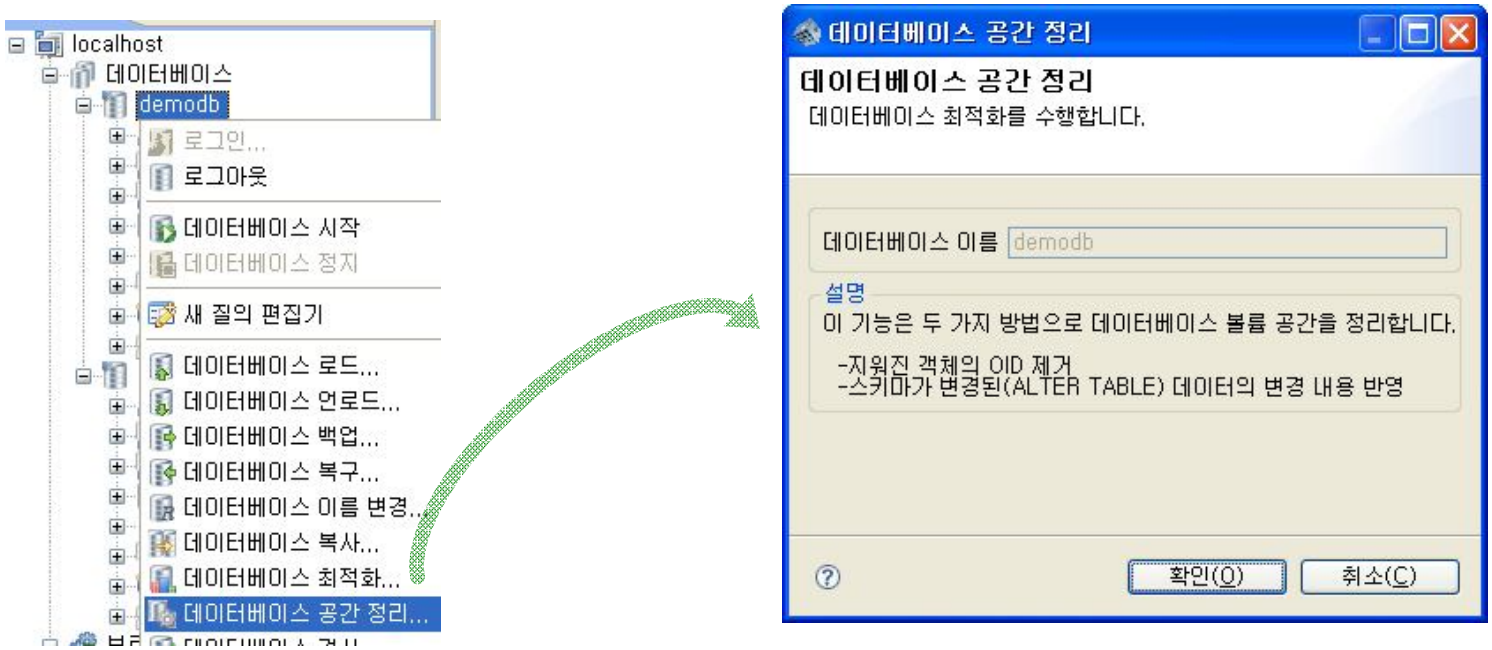
CUBRID
More than open source!

- 데이터베이스 정리
  - 데이터의 삭제 및 변경, 칼럼의 추가 등의 작업을 데이터베이스에 대해 계속적으로 수행한 경우에는 삭제된 객체의 정보와 같은 쓰레기(garbage)가 쌓이게 된다. 이 쓰레기 값을 정리하고 데이터베이스를 효율적으로 사용하기 위해 주기적으로 수행하는 것을 권장한다.
  - 내용
    - Stand-alone 모드에서 실행
    - 삭제된 객체 정보(OID) 정리
    - 가용 공간 확보
    - OID 값 변경 가능

# 데이터베이스 정리 - 계속

CUBRID
More than open source!

- 데이터베이스 정리 방법




- 명령어 이용 : compactdb

```
% cubrid compactdb  [ option ]  database_name
Optoins :
-v : enables status messages during execution
```

# 데이터베이스 검사

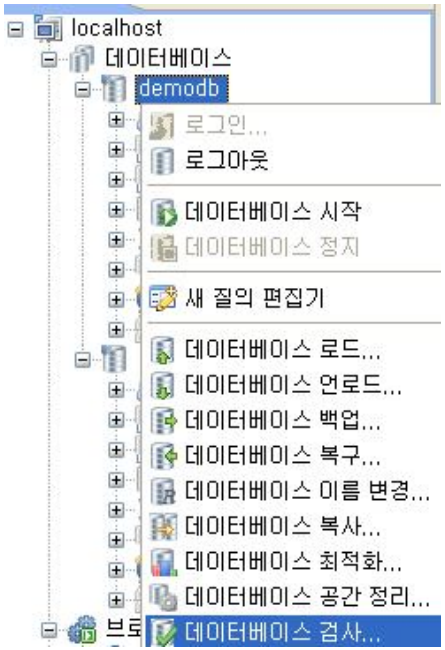
CUBRID
More than open source!

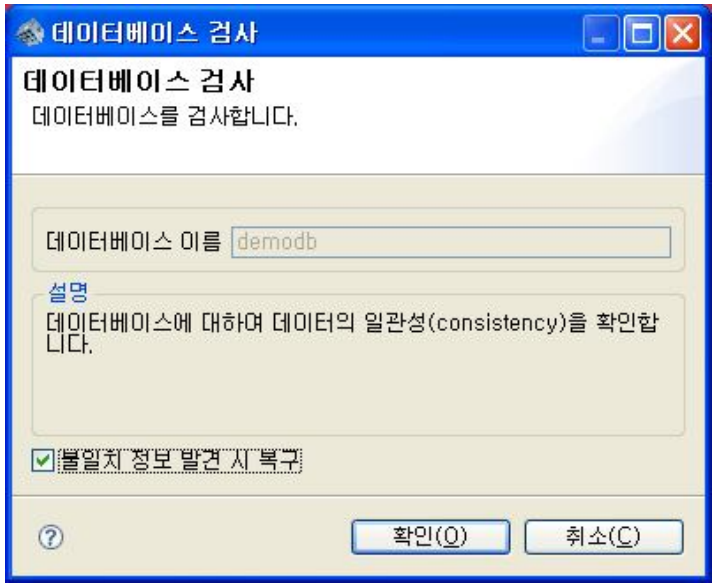
- 주요특징 및 내용
  - 데이터베이스내의 데이터영역과 인덱스영역간의 불일치검사 및 복구기능을 수행
  - 검사중인 대상에 대하여 S_LOCK 선점
    - 서비스가 BUSY인 경우 성능저하 발생할 수 있으므로 가급적 서비스가 없는 시점을 이용하여 수행
  - 주기적인 수행을 권장
  - 불일치에 대한 복구기능을 수행
    - cubrid manger에서는 지원하지 않음
    - 명령어의 옵션을 부여하여 수행

# 데이터베이스 검사 - 계속

CUBRID
More than open source!

- 데이터베이스 검사 방법




- 명령어 이용 : checkdb

```
% cubrid checkdb   [OPTION]   database-name
```

| 옵션 | 설명 | 기본값 |
|---|---|---|
| -S \| -C | stand-alone, client-server mode 지정 | -C |
| -r | 데이터와 인덱스간의 불일치 점검 및 복구 | 수행 안함 |

# 데이터베이스 언로드

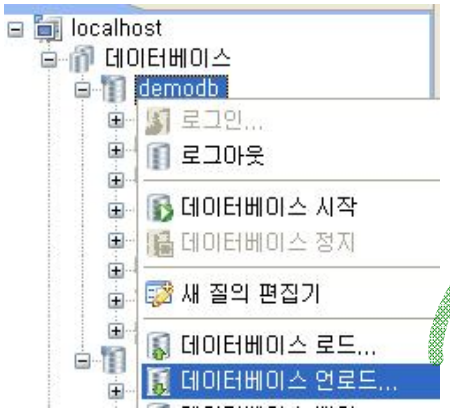
CUBRID
More than open source!

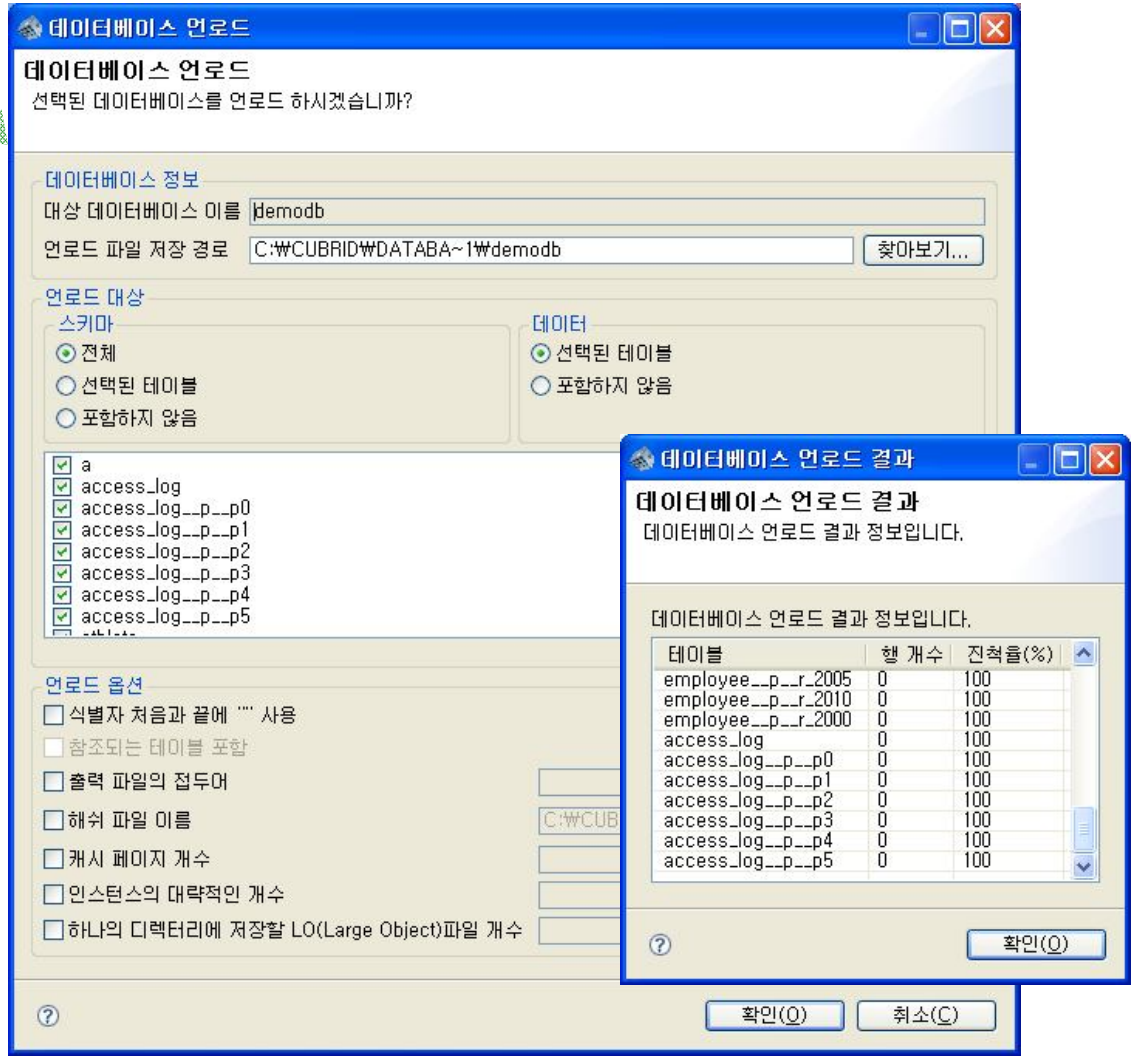
- 현재의 데이터베이스를 파일로 내려 받는다.
- 특정테이블에 대하여 백업을 하고 싶을 경우
- 데이터베이스의 전체 스키마를 확인할 경우
- 생성파일
  - 스키마 파일 :데이터베이스의 스키마 정의 포함하는 파일( demodb_schema)
  - 객체 파일 : 데이터베이스의 데이터를 포함한 파일 (demodb_objects)
  - 인덱스 파일 : 데이터베이스에 정의된 인덱스를 포함하는 파일 (demodb_indexes)
  - 트리거 파일 : 데이터베이스에 정의된 트리거를 포함하는 파일 (demodb_triger)
  - 멀티미디어 데이터(GLO)가 있을 경우 파일(*.lo)

# 데이터베이스 언로드 - 계속

CUBRID
More than open source!

- 데이터베이스 언로드 방법




1. 대상디렉토리
   - unload 받은 파일이 저장될 위치 지정
2. 언로드 대상
   - 스키마 : 전체 혹은 부분 테이블인지 선택
   - 데이터 : 전체 혹은 부분 테이블인지 선택
3. 해시 파일 디렉토리 지정
   - 개체 모델 사용시 사용
   - 테이블 참조 관련 정보 저장파일 생성
4. 기타 옵션을 부여하여 수행

# 데이터베이스 언로드 - 계속

CUBRID
More than open source!

- 명령어 이용 : unloaddb

% cubrid unloaddb [OPTION] database-name
% cubrid unloaddb demodb
➔ demodb의 모든 스키마 및 데이터를 파일로 출력

| 옵션 | 인자 | 설명 | 기본값 |
|---|---|---|---|
| -S \| -C | | stand-alone, client-server mode 지정 | -C |
| -i | input-file | unload 받고자 하는 테이블의 이름을 기술하는 파일. 지정하지 않으면 전체테이블에 해당함 | 없음(모든 테이블) |
| --include-reference | | -i 옵션에서 지정한 테이블의 레코드가 참조하는 레코드를 함께 unload | 실행 않음<br>(지정한 테이블만 해당) |
| --input-class-only | | -I 옵션에서 입력 파일에 지정한 테이블만 출력 | 시스템 테이블 언로드 |
| -O | output_directory | 파일을 출력한 경로 지정 | 현재 디렉토리 |
| -s \| -d | | 스키마만 unload \| 데이터만 unload | 전체 |
| --output-prefix | output_prefix | 실행 결과 파일의 첫 문자열(prefix)를 지정 | 데이터베이스 이름 |
| --hash-file | hash_filename | 해쉬 파일 이름 | 시스템 자동 생성 |
| -v | | 실행 과정 출력 | 수행 않음 |

# 데이터베이스 언로드 - 계속

CUBRID
More than open source!

- 특정 테이블 언로드
  - 언로드 받을 테이블리스트 파일 생성

```
% more table.list
olympic
game
history

➔ 파일의 끝에 엔터 입력(₩n)
```

  - 언로드 수행

```
% cubrid unloaddb -i table.list --input-class-only demodb -v

CUBRID 2008 R1.4

------------------------------+------------------------------
   Class Name                 |        Total Instances
------------------------------+------------------------------
   olympic                    |          25 (100% /   0%)
   game                       |        8653 (100% /  45%)
   history                    |         147 (100% /  45%)
8825 objects dumped.
```

# 데이터베이스 로드

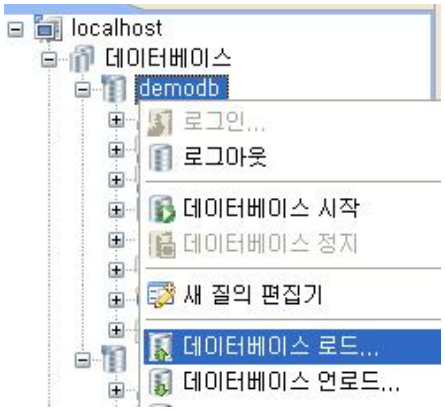
CUBRID
More than open source!

- 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 로드 - 계속

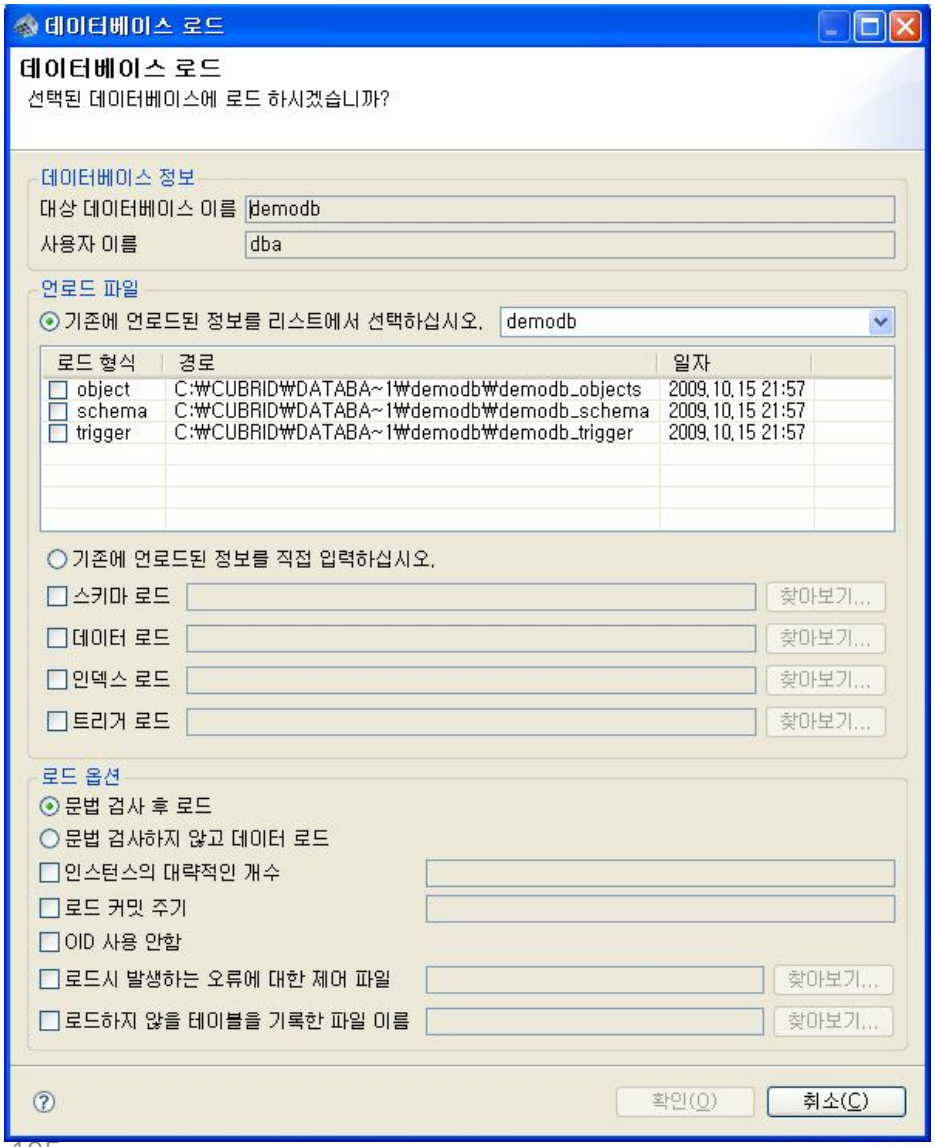
CUBRID
More than open source!

- 데이터베이스 로드 방법




1. 언로드 파일
   - 언로드 받은 이력을 이용하여 선택
   - 이력이 없을 경우 직접 선택
     - 스키마, 데이터, 인덱스, 트리거

3. 로드옵션
   - 문법 검사 유무 확인
   - 커밋주기 : 1만건이 무난
   - OID 사용안함 : 관계형 모델일 경우
   - 로드할 테이블 선택



# 데이터베이스 로드 - 계속

- 명령어 사용 : loaddb

| 옵션 | 인자 | 설명 | 기본값 |
|---|---|---|---|
| -u | user-name | 데이터베이스 사용자 이름. | public |
| -p | password | 사용자 암호 | 없음 |
| -l | | 문법 검사하지 않고 데이터 로드만 수행, loaddb 수행 시간 단축 | 문법검사 및 로드 수행 |
| -v | | 실행 과정 출력, -c 옵션이 있는 경우 commit된 레코드의 수 출력 | 수행 않음 |
| -c | periodic-commit | periodic-commit 값만큼 주기적으로 commit 수행. -i, -s 옵션 사용될 경우에는 수행된 SQL문이 됨 | commit 없음 |
| --no-oid | | oid참조 정보를 만들지 않으며 수행속도 향상, 단 다른 테이블참조 하는 형태의 설계가 없어야 함 | 수행 않음 |
| -s | string[:integer] | 스키마 정의 파일. 특정 라인을 지정하여 로드 가능 | 수행 않음 |
| -i | string[:integer] | 인덱스 정의 파일. 특정 라인을 지정하여 로드 가능 | 수행 않음 |
| -d | string | object input file 이름 | 수행 않음 |

# 데이터베이스 로드 - 계속

CUBRID
More than open source!

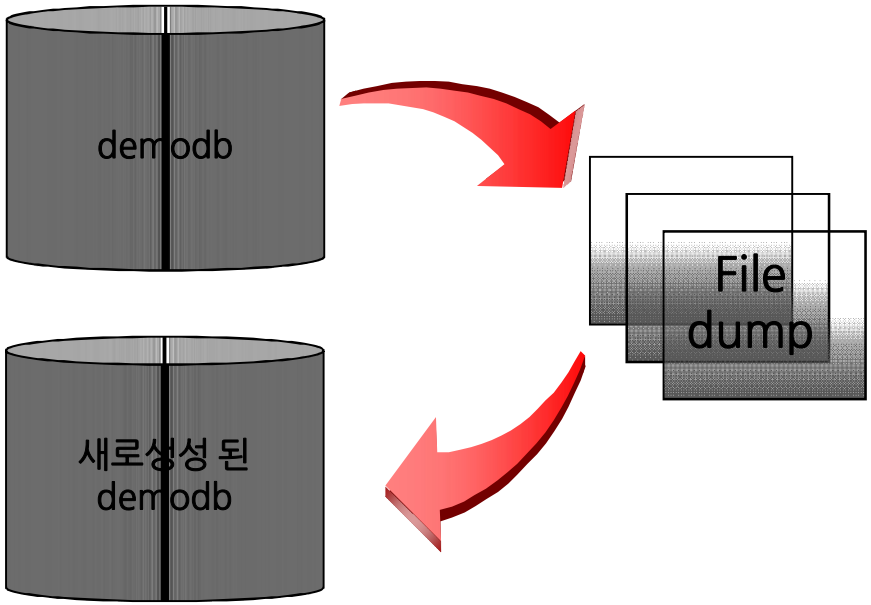
- loaddb 명령어 사용 순서
  - 로드를 수행하는 순서는 스키마 ➔ 데이터 ➔ 인덱스 ➔ 트리거 순이다.
  - 아래 예는 언로드 받은 파일이 스키마, 데이터, 인덱스, 트리거가 있다고 가정

```
% cubrid loaddb -u dba -s demodb_schema edudb
% cubrid loaddb -u dba --no-oid -d demodb_objects -c 10000 edudb
% cubrid loaddb -u dba -i demodb_indexes edudb
% cubrid loaddb -u dba -i demodb_trigger edudb
```

# 데이터베이스 재구성

CUBRID
More than open source!

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



# 실습 - 데이터베이스 재구성

CUBRID
More than open source!

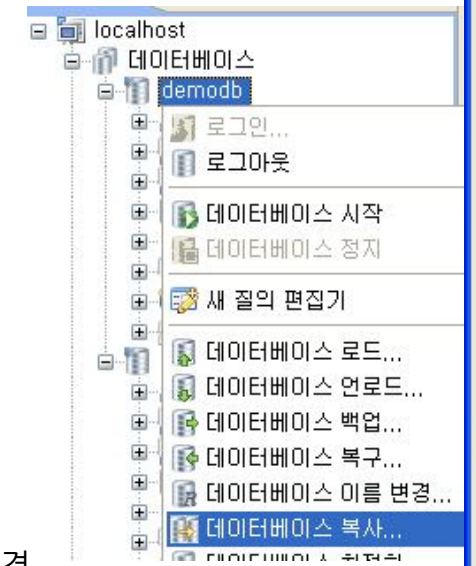
- demodb를 파일로 내려 받은 후 edudb에 적재
  - demodb에서 unload를 수행
  - demodb 백업
  - edudb 구동 종료
  - unload받은 파일 들을 이용하여 edudb에 로드 수행
  - 통계정보 갱신
  - 정상적으로 로드 되었는지 확인

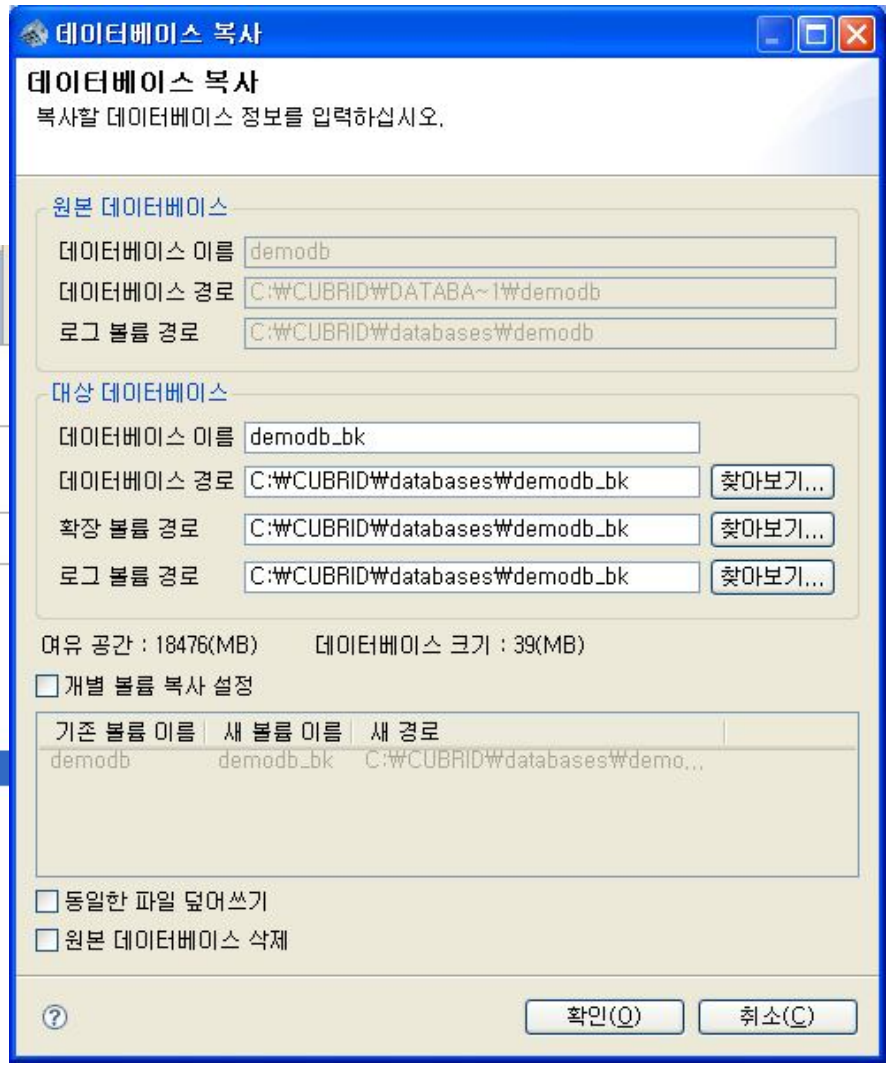
# 데이터베이스 복사 및 이동

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

1. 원본 데이터베이스
   - 복사될 이름 및 경로 표시
2. 대상 데이터베이스
   - 원본으로부터 복사할 데이터베이스
     이름/경로 명시
   - 확장된 볼륨에 대한 경로 명시
   - 로그볼륨의 경로 명시
3. 옵션
   - 개별 볼륨 복사 설정
     복사될 각 볼륨의 이름 변경 및 경로를 변경
   - 덮어쓰기
     기존의 데이터베이스가 있을 경우 덮어쓰기
   - 원본삭제
     복사 후 원본데이터베이스 삭제

❖ 데이터베이스 이동에 사용



# 데이터베이스 복사 및 이동 - 계속

CUBRID
More than open source!

- 명령어 사용 : copydb

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

➔ 원본 edudb를 "C:₩CUBRID₩databases₩edudb_bk" 경로에 로그볼륨 및 확장볼륨 포함하여 edudb_bk로 복사

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

# 실습 - 데이터베이스 복사 및 이동

CUBRID
More than open source!

- 볼륨이 추가된 데이터베이스를 각 볼륨의 경로를 변경하여 데이터베이스를 복사
  - 원본 데이터베이스 : edudb
    - 확장볼륨크기 : data 500,000page, index 250,000page, temp 250,000page
  - 생성할 데이터베이스 : edudb_bk
    - 데이터베이스위치 : C:₩CUBRID₩Databases₩edudb_bk
    - Log볼륨위치 : C:₩CUBRID₩Databases₩edudb_bk₩log
    - Data볼륨위치 : C:₩CUBRID₩Databases₩edudb_bk₩data
    - Index볼륨위치 : C:₩CUBRID₩Databases₩edudb_bk₩index
    - Temp볼륨위치 : C:₩CUBRID₩Databases₩edudb_bk₩temp

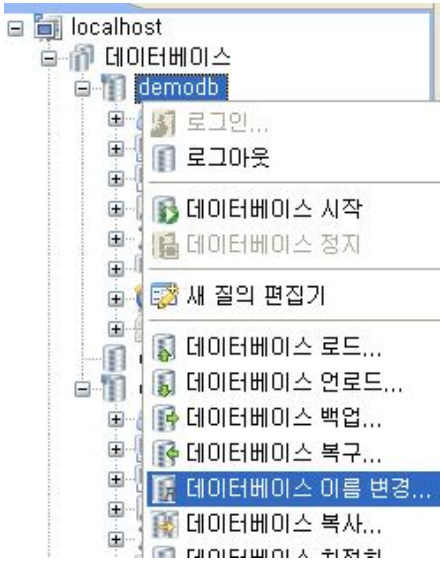
# 데이터베이스 이름변경

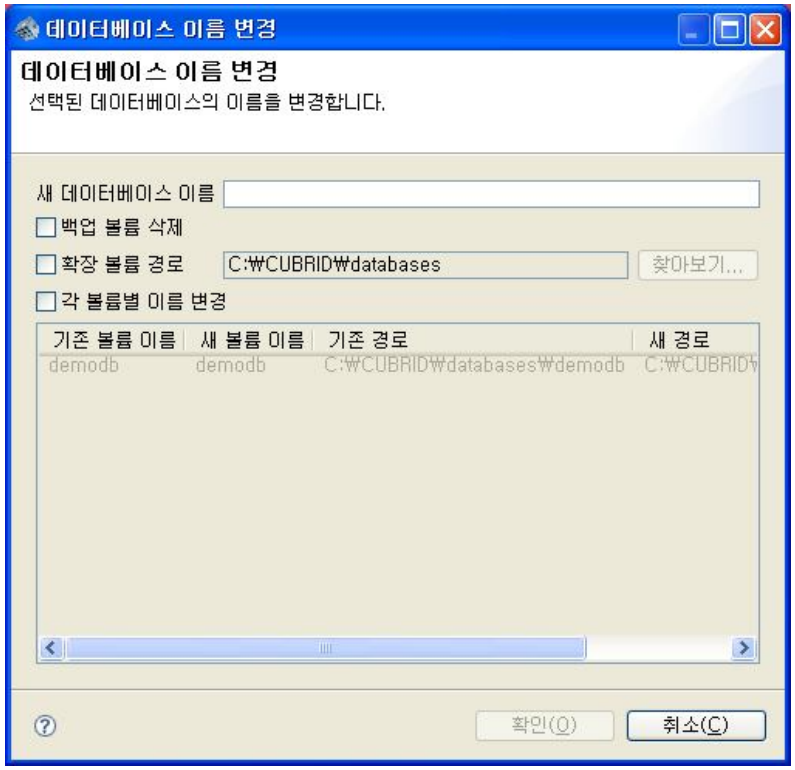
CUBRID
More than open source!

- 이미 작성된 데이터베이스 이름을 변경
- 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 볼륨별 이름변경
     볼륨들에 대하여 볼륨의 이름 및 경로를 변경

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



# 데이터베이스 이름변경 - 계속

CUBRID
More than open source!

- 명령어 사용 : renamedb

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

% cubrid renamedb edudb edudb_rename

➔ 원본 edudb를 경로변경 없이 edudb_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 |

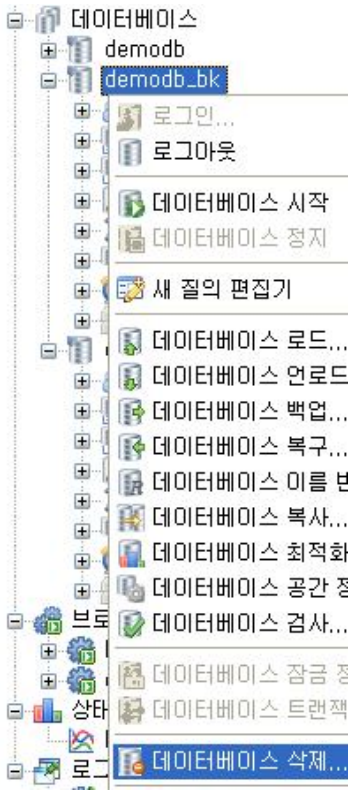
# 실습 - 데이터베이스 이름변경

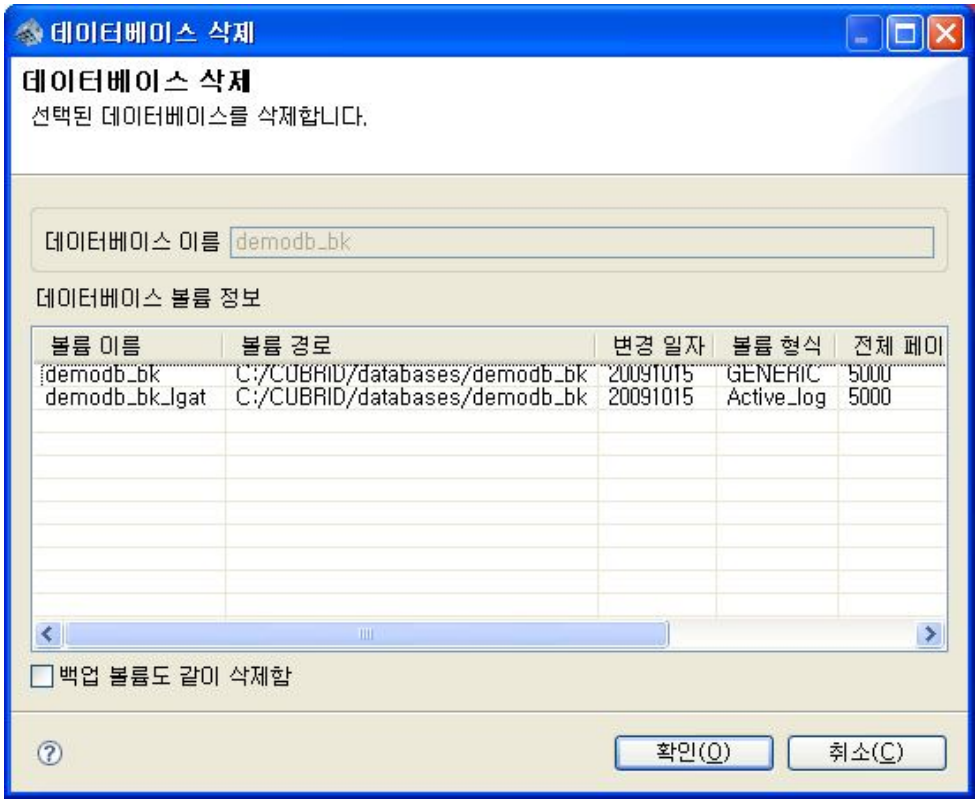
CUBRID
More than open source!

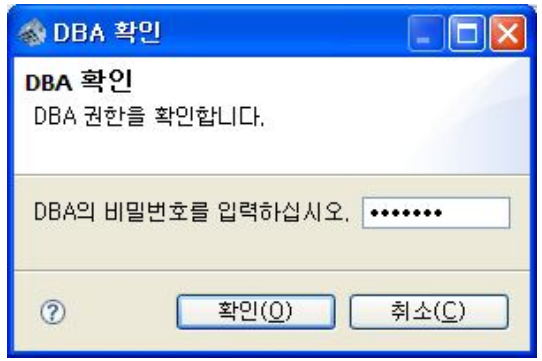
- 데이터베이스 이름변경을 이용하여 각 볼륨의 경로를 변경
  - 데이터베이스 중지
  - 데이터베이스명 변경
    - 대상데이터베이스 : edudb
    - 변경할 이름 : edudb_rename
    - 변경할 위치 : C:₩CUBRID₩databases₩edudb_rename, ./data, ./index, ./temp
  - 데이터베이스 구성파일 확인
    - 로그위치 및 볼륨의 위치

# 데이터베이스 삭제 - 계속

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



# 데이터베이스 삭제 - 계속

CUBRID
More than open source!

- 명령어 사용 : deletedb

```
% cubrid deletedb   [OPTION]   database-name
% cubrid  deletedb -d demodb_new
```

➔ 백업볼륨을 포함하여 삭제

# 실습 - 데이터베이스 삭제

CUBRID
More than open source!

- demodb를 제외한 모든 데이터베이스 삭제
  - 데이터베이스의 경로에서 데이터베이스 구성파일 확인
  - 데이터베이스 삭제 수행
    - 백업 삭제 옵션을 부여
  - 삭제한 데이터베이스의 경로에서 데이터베이스 구성파일 존재 확인

# 데이터베이스 자동화

CUBRID
More than open source!

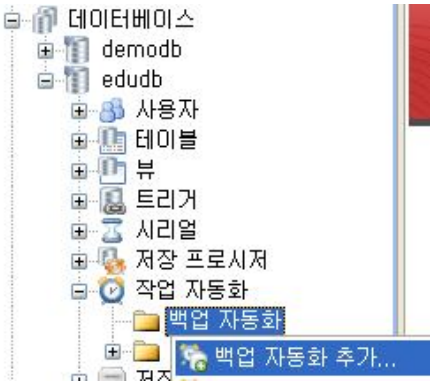
- 데이터베이스 자동시작
  - 큐브리드 서비스구동과 함께 데이터베이스 자동시작 설정
  - 환경 설정 파일(cubrid.conf)에 자동시작할 데이터베이스 이름 등록
    - “server=” 다음에 자동 시작할 데이터베이스를 공백없이 “,”로 구분하여 나열

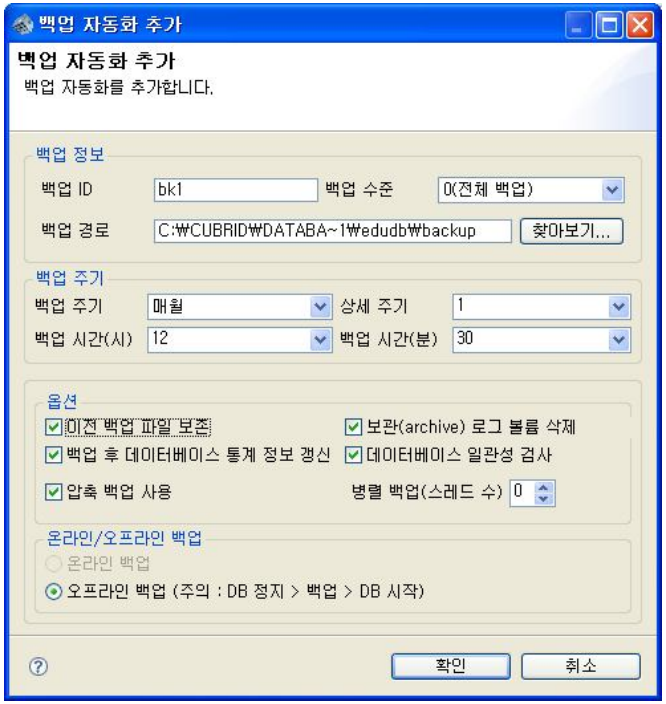
```
# The list of database servers in all by 'cubrid service start' command.
# This property is effective only when the above 'service' property contains 'server' keyword.
server=demodb,edudb
```

# 데이터베이스 자동화 - 계속

CUBRID
More than open source!

- 데이터베이스 백업 자동화
  - 선택한 데이터베이스에 대하여 백업을 설정
  - 주기적인 백업 설정이 가능
  - 기존 백업에서 사용되는 옵션을 부여할 수 있음
  - 자동화 수행에 대한 오류로그를 볼 수 있다.





1. 백업 아이디 : 중복되지 않도록 명시
2. 백업 경로 : 백업을 받을 경로 명시
3. 백업 수준 : 수행할 백업레벨
4. 백업 주기, 상세주기
5. 백업시간 : 백업을 수행할 시간
6. 온라인/오프라인 백업
   - 온라인백업 : 서버 구동 중 백업
   - 오프라인백업 : 서버 중지 후 백업
     백업후 서버 재구동
7. 옵션
   - 기존백업 옵션과 동일

# 실습 - 데이터베이스 자동화

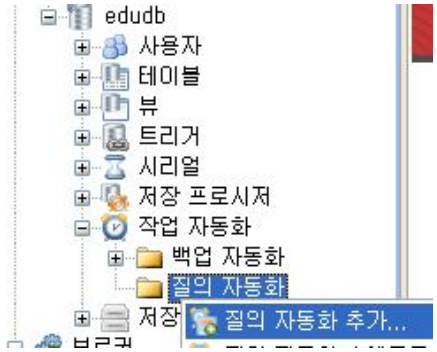
CUBRID
More than open source!

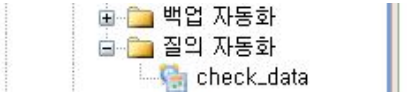
- demodb에 대하여 주간단위 백업 주기를 설정한다.
  - 백업주기 : 매월1일(level 0), 일(level 1), 월~토(level 2)
    - Level0 백업위치 : C:₩CUBRID₩databases₩demodb₩backup0
    - Level1 백업위치 : C:₩CUBRID₩databases₩demodb₩backup1
    - Level2 백업위치 : C:₩CUBRID₩databases₩demodb₩backup2
  - 부여옵션
    - 보관로그삭제
    - 압축백업
    - 온라인 백업
    - 백업시간 00:00
    - Level0 백업에서 통계정보 갱신
    - Level0 백업에서 데이터베이스 일관성 체크

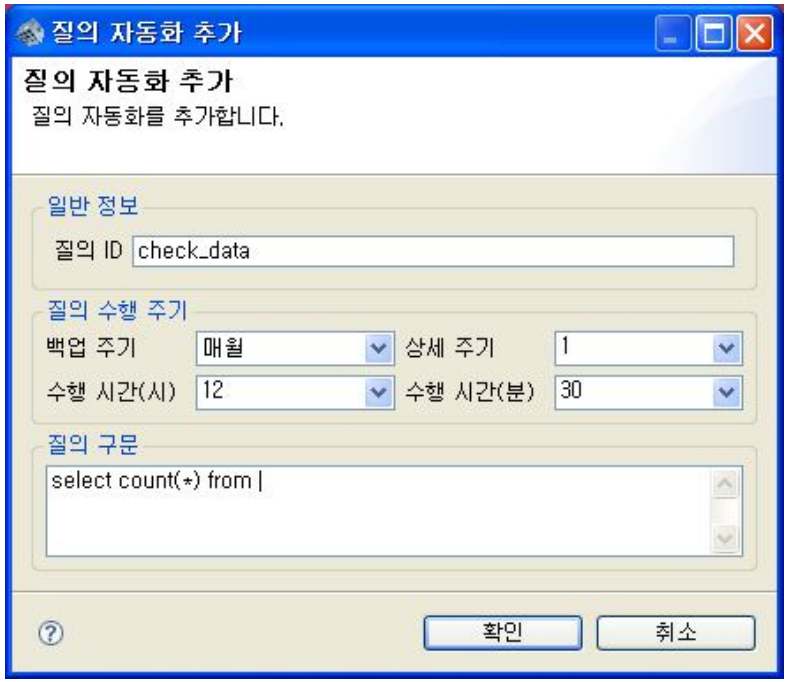
# 데이터베이스 자동화 - 계속

CUBRID
More than open source!

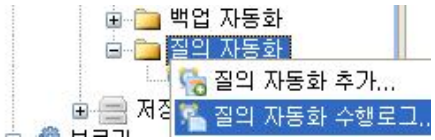
- 데이터베이스 질의 자동화
  - 원하는 주기 및 특정일에 명시된 질의를 수행
  - 자동화 수행에 대한 오류로그를 볼 수 있다.
    - 질의만 수행하며 SELECT결과는 확인 할 수 없다.




1. 질의 실행 계획 아이디 : 중복되지 않도록 명시
2. 질의 실행 주기/시간
   - 원하는 주기를 선택
   - 상세주기 및 수행시간 명시
3. 실행시간 : 수행할 시간 명시
4. 질의구문 : 실행할 질의를 입력



# 실습 - 데이터베이스 자동화

CUBRID
More than open source!

- demodb에 대하여 테이블 생성 질의 자동화를 설정한다.
  - 질의 수행 주기 : 현재일
  - 수행시간 : 실습PC의 현재시간 기준으로 5분 후
  - 테이블 생성 질의문장
    - CREATE TABLE BOARD(ID INT, NAME VARCHAR(20));
  - 테이블 생성 확인

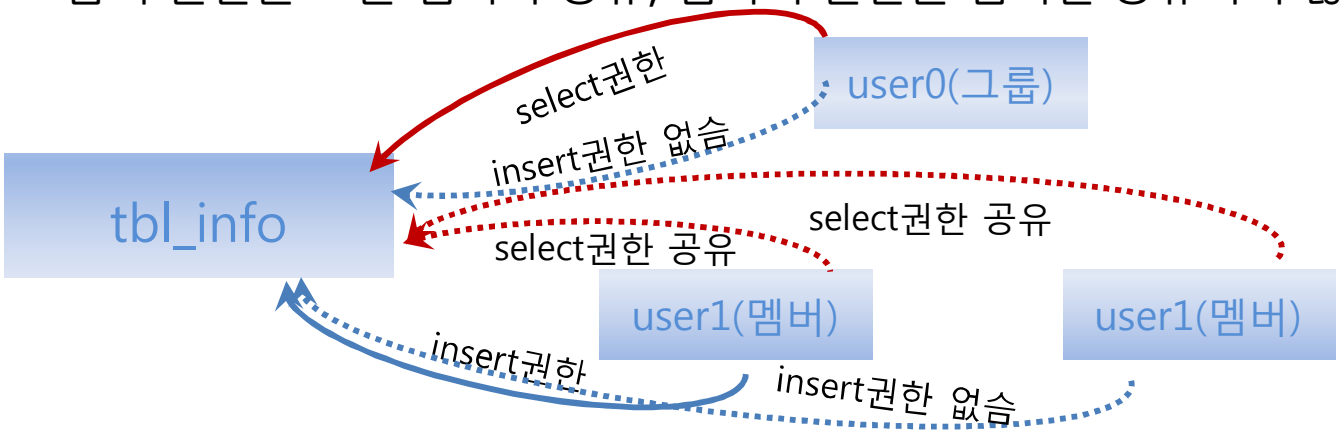
# 보안 및 권한 관리

CUBRID
More than open source!

# 사용자 관리/권한 개요

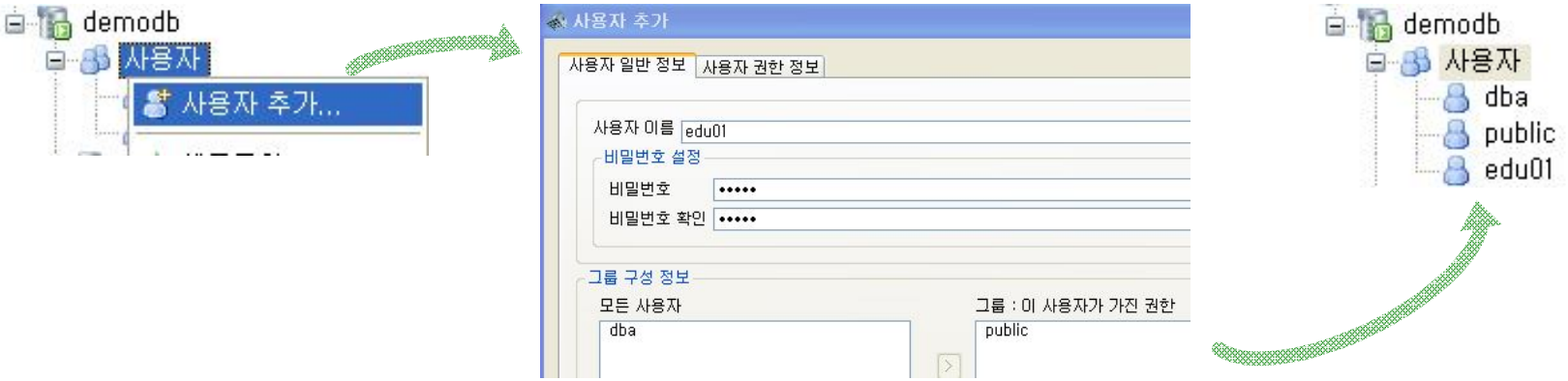
CUBRID
More than open source!

- CUBRID 데이터베이스는 PUBLIC 사용자(그룹)와 DBA 사용자를 기본 제공
- DBA는 모든 사용자의 멤버로 등록, 데이터베이스내의 모든 개체 접근 가능
- DBA는 데이터의 보안을 위하여 사용자를 생성하고 권한을 부여
- DBA 포함, 추가되어지는 모든 사용자들은 PUBLIC 그룹의 멤버로 등록
- 권한 부여 가능자는 테이블의 소유자, DBA, 권한을 가진 그룹의 멤버
- 계층적 사용자 구조를 구성
  - 그룹: 멤버를 가지는 사용자
- 그룹의 권한은 모든 멤버가 공유, 멤버의 권한은 멤버간 공유되지 않음

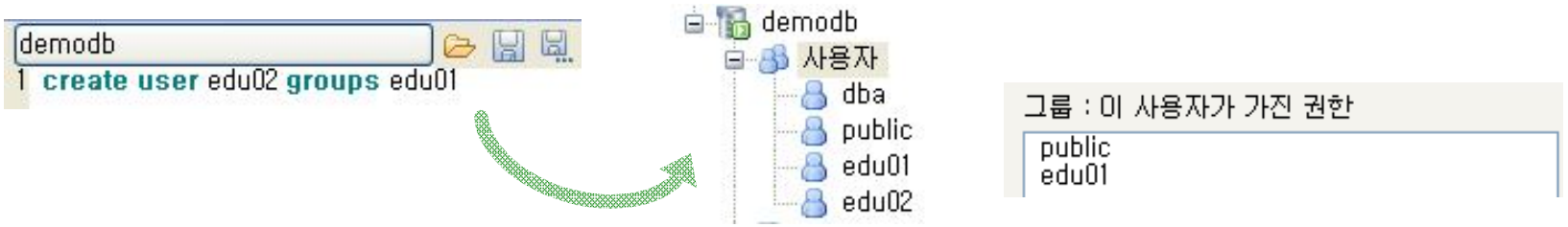


# 사용자 관리

- 사용자 추가
  - DBA와 DBA의 멤버만 가능
  - 패스워드 명시하지 않으면 패스워드는 공백 (CM에서는 반드시 입력)
  - CUBRID Manager client 사용자 트리에서 우클릭후 사용자 추가 선택
    - 새로운 사용자는 PUBLIC의 멤버로 자동 추가됨



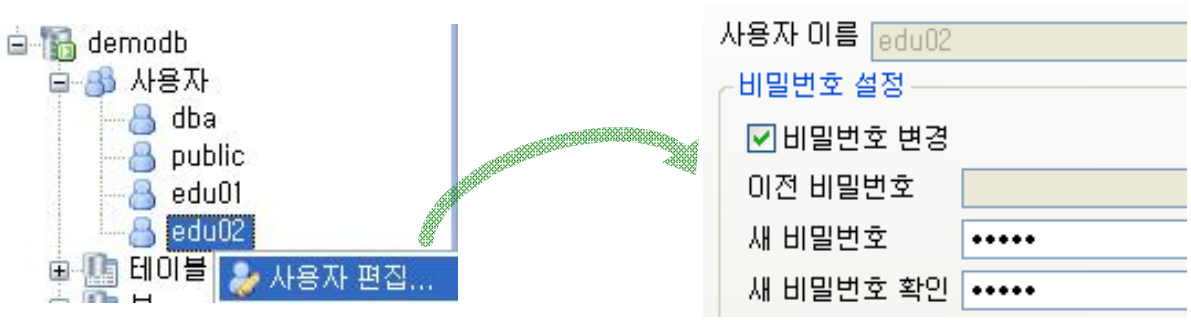

  - SQL



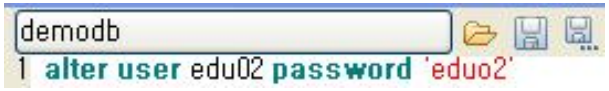
# 사용자 관리 (계속)

CUBRID
More than open source!

- 사용자 정보 변경
  - 암호 변경



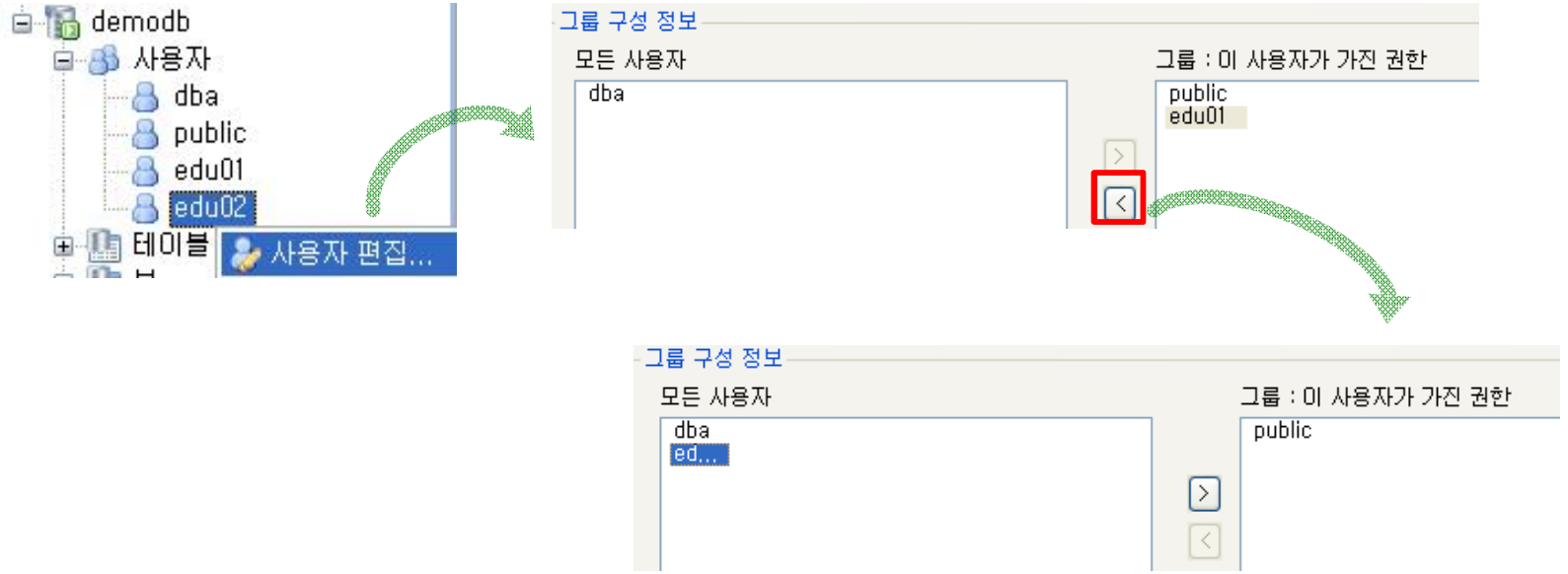

  - SQL



# 사용자 관리 (계속)

CUBRID
More than open source!

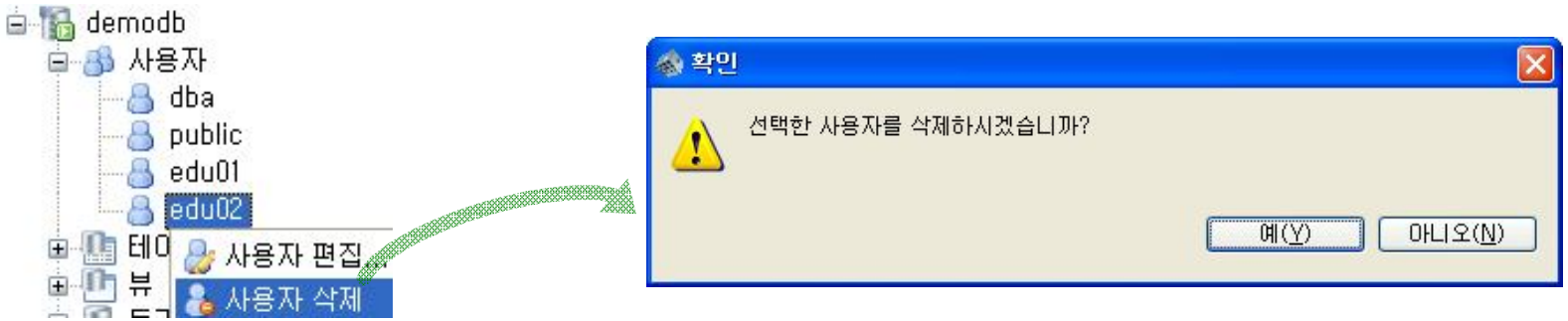
- 사용자 정보 변경
  - 그룹/멤버 설정 변경
    예) edu01 사용자 그룹의 멤버로 변경



# 사용자 추가 및 삭제 (계속)

CUBRID
More than open source!

- 사용자 삭제
  - DBA와 DBA의 멤버만 가능
  - 테이블를 소유하고 있는 사용자는 삭제할 수 없다.
  - 삭제 방법
    - CUBRID Manager client 사용자 트리에서 우클릭후 사용자 삭제 선택



# 접근권한

CUBRID
More than open source!

- 데이터 및 스키마에 대한 접근 제한
  - 권한의 기본 단위는 테이블
  - 사용자와 그룹을 통하여 접근을 제어 관리
  - 시스템 테이블 db_user, db_auto 테이블에서 관리
  - 시스템에서 제공하는 기본 사용자
    - DBA
      - 모든 그룹의 멤버
      - 데이터베이스의 모든 객체 접근 가능
    - PUBLIC
      - 모든 사용자는 PUBLIC 그룹의 멤버
    - 그룹은 멤버를 갖는 사용자로 권한이 그룹에 주어지면 그룹의 모든 멤버들은 같은 권한을 가짐

# 권한부여/제거

CUBRID
More than open source!

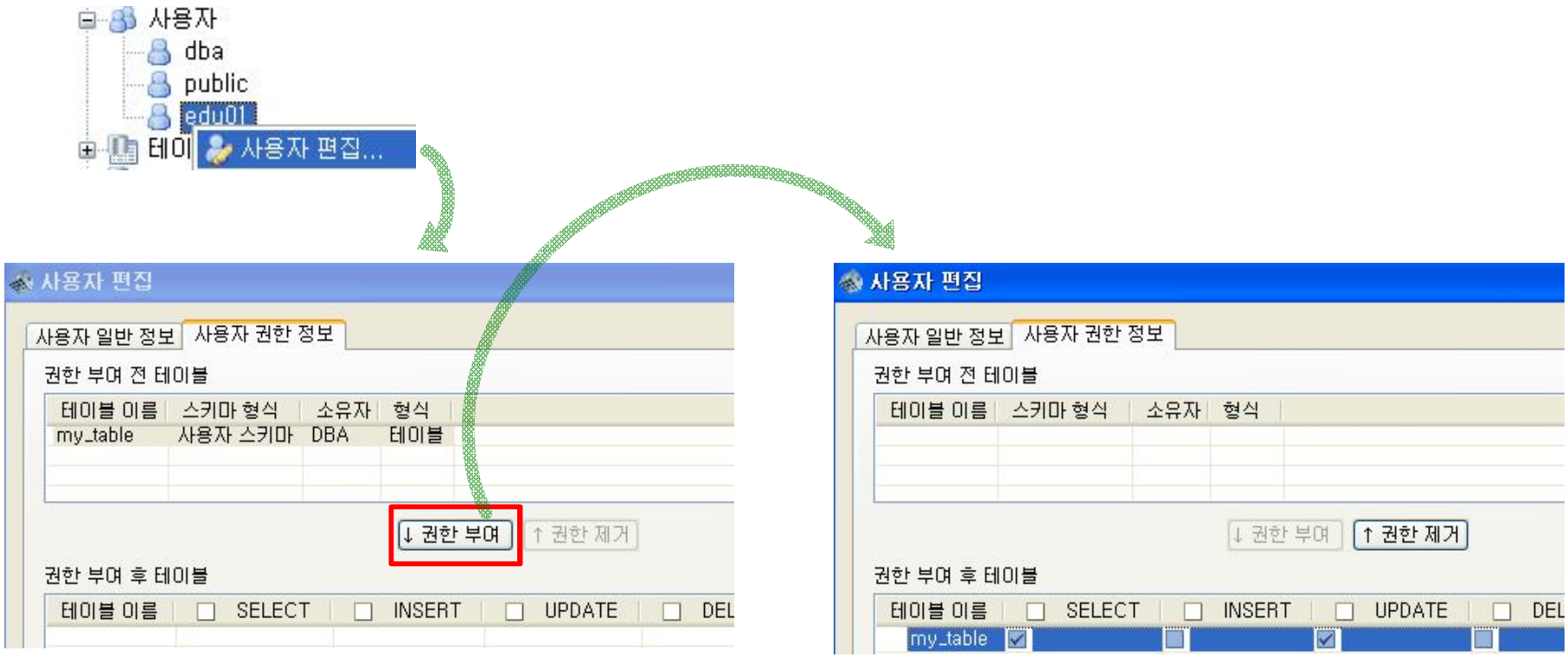
- 임의의 테이블에 대하여 특정 사용자에게 권한 부여 가능
- 테이블의 소유자, DBA, 권한을 가진 그룹의 멤버가 권한 부여 가능
- 권한 부여/제거 방법
  - GRANT/REVOKE SQL문

```
GRANT <privileges> ON <table specification comma list>
    TO <user name comma list> [ WITH GRANT OPTION ]

REVOKE <privileges> ON <table specification comma list>
    FROM <user name comma list>

- PRIVILEGES
  ALL [ PRIVILEGES ] | <privilege comma list>

Ex) GRANT SELECT, INSERT, UPDATE ON employee TO jones;
Ex) GRANT ALL PRIVILEGES ON person, student TO smith, brown;
Ex) REVOKE INSERT, UPDATE ON employee FROM smith;
Ex) REVOKE SELECT ON manufacturing_site FROM jones, brown;
```

# 권한부여/제거 (계속)

CUBRID
More than open source!

- CUBRID Manager를 이용한 권한부여/제거
  - 테이블 소유자로 로그인 후, 권한을 부여할 사용자 선택




체크박스 선택을 통한 권한 부여/제거

# 트랜잭션 관리

CUBRID
More than open source!

# 잠금 정보 확인

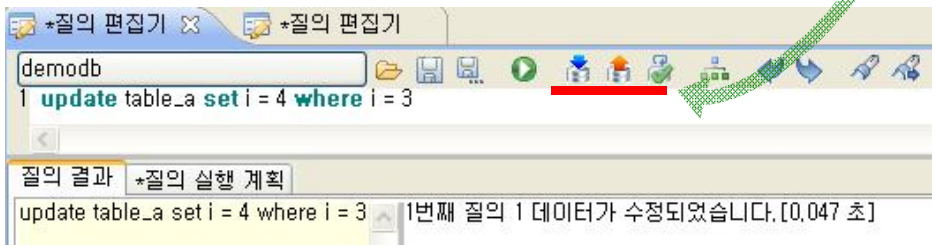
CUBRID
More than open source!

- 데이터베이스에 대한 현재 Locking 상태를 확인할 수 있다.
  - Lock에 대한 객체 생성(객체단위 : 테이블, 테이블의 ROW)
  - 각 객체별 정보를 출력
- 제공되는 정보
  - **서버의 Lock 관련 설정**
  - **서버에 접속중인 클라이언트들의 정보**
  - **객체에 대한 Lock 테이블 정보**

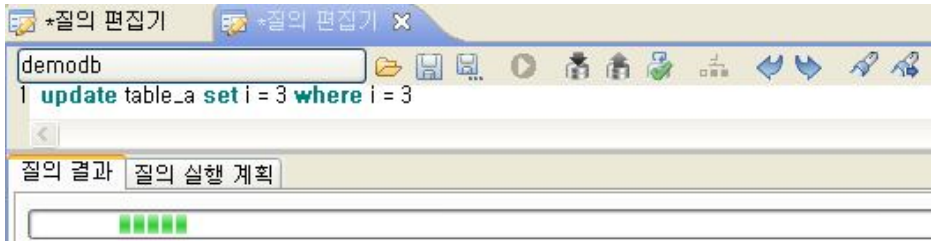
# 잠금 정보 확인

CUBRID
More than open source!

- 내용 확인을 위하여 설정 조정
  - locking 정보 확인위해 autocommit off
  - 임의의 테이블(table_a) 의 한 레코드에 update 수행




  - 새로운 질의 편집기에서 동일한 레코드에 update 수행




  - lock_timeout -1

# 잠금 정보 확인 (계속)

CUBRID
More than open source!

- 명령어 사용 : lockdb
  - 데이터베이스에 대한 Locking 상태의 현재 스냅샷을 보여준다.

**cubrid lockdb [OPTION] database-name**

Options: -o

➔ 출력을 파일로 저장

cubrid lockdb demodb

**서버의 lock관련 설정**

Lock Escalation at = 100000, Run Deadlock interval = 1

➔ Row Lock에서 Table Lock으로 변환될 수 있는 Lock의 수, Deadlock탐지 간격

# 잠금 정보 확인 (계속)

CUBRID
More than open source!

- 개체에 대한 lock 정보

```
OID =  0|  1780|   7
Object type: Instance of class ( 0|   288|   6) = table_a.
Total mode of holders =    X_LOCK, Total mode of waiters =    X_LOCK.
Num holders=  1, Num blocked-holders=  0, Num waiters=  1
LOCK HOLDERS:
   Tran_index =   2, Granted_mode =   X_LOCK, Count =   2
LOCK WAITERS:
   Tran_index =   1, Blocked_mode =   X_LOCK
                Start_waiting_at = Wed Sep 23 12:06:06 2009
                Wait_for_nsecs = -1
```

lock 대상 개체 정보

2번 트랜잭션이 이 개체에 X_LOCK 을 가지고 있음

1번 트랜잭션이 이 개체에 X_LOCK 을 잡기위해 기다리고 있음

# 잠금 정보 확인 (계속)

CUBRID
More than open source!

- 트랜잭션 정보

Transaction (index 1, cub_cas, dba@mycom|2908)
Isolation REPEATABLE CLASSES AND READ UNCOMMITTED INSTANCES
State TRAN_ACTIVE
Timeout_period -1

1번트랜잭션, cub_cas 프로세스, dba로 로그인중, 프로세스ID:2908

lock 레벨: 테이블 읽기보장, 레코드는 더티리드 허용

Transaction (index 2, cub_cas, dba@mycom|2980)
Isolation REPEATABLE CLASSES AND READ UNCOMMITTED INSTANCES
State TRAN_ACTIVE
Timeout_period -1

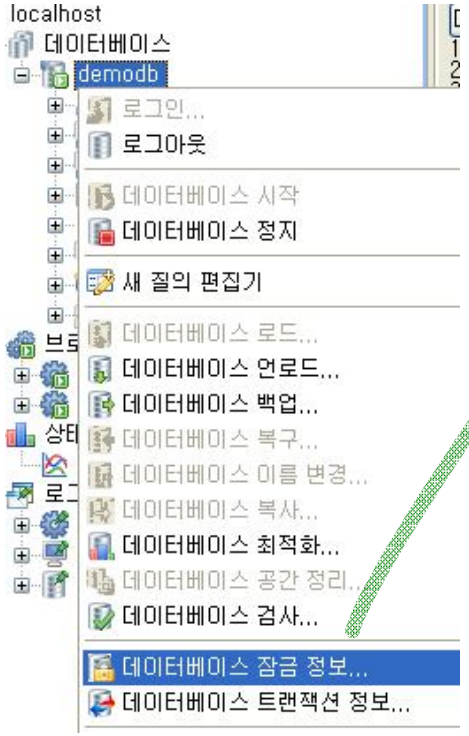
2번트랜잭션, cub_cas 프로세스, dba로 로그인중, 프로세스ID:2980

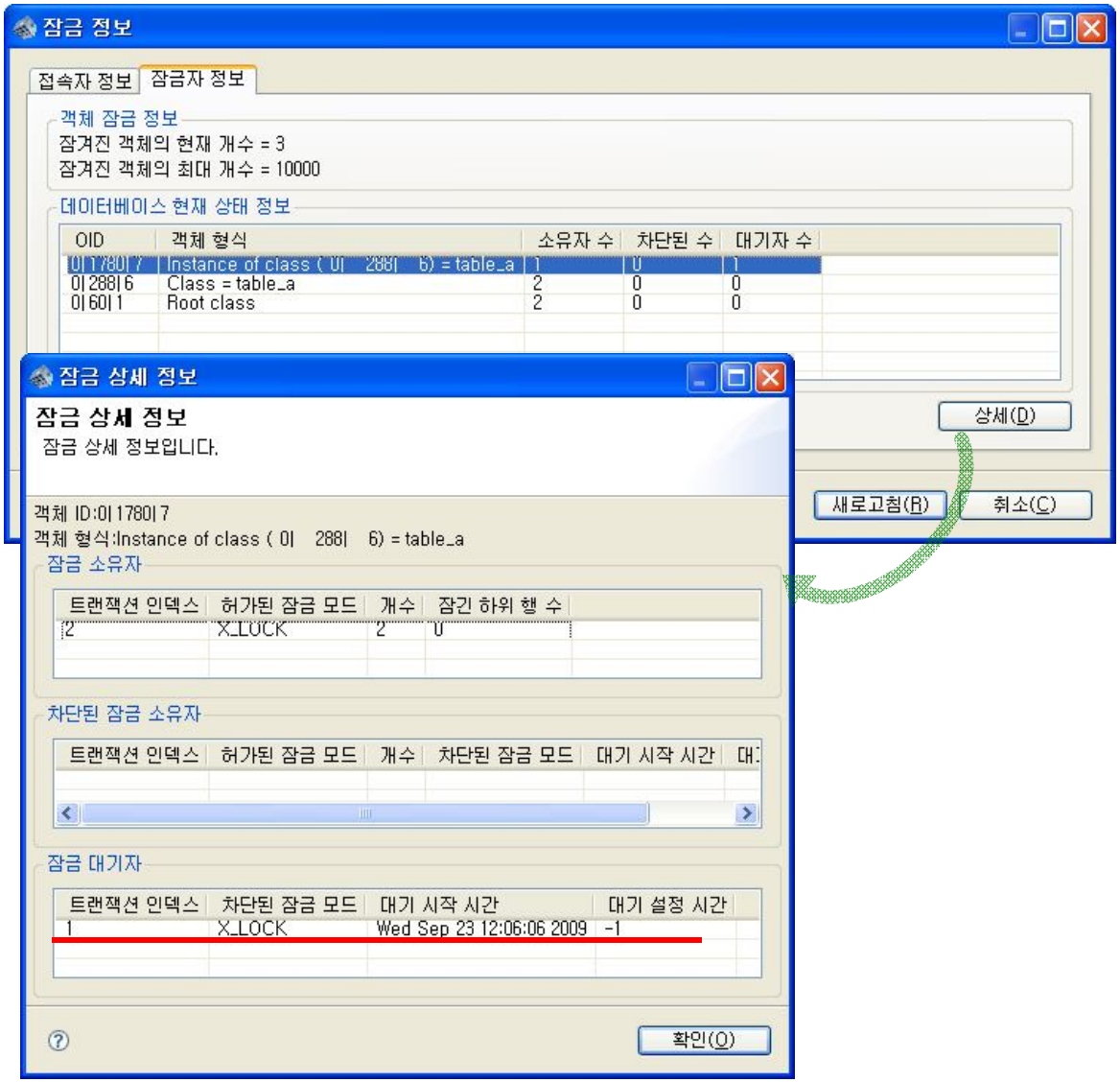
lock을 잡기위해 기다리는 시간, -1은 무제한

# 잠금 정보 확인 (계속)

CUBRID
More than open source!

- CUBRID Manger
  - dba 만 볼 수 있음

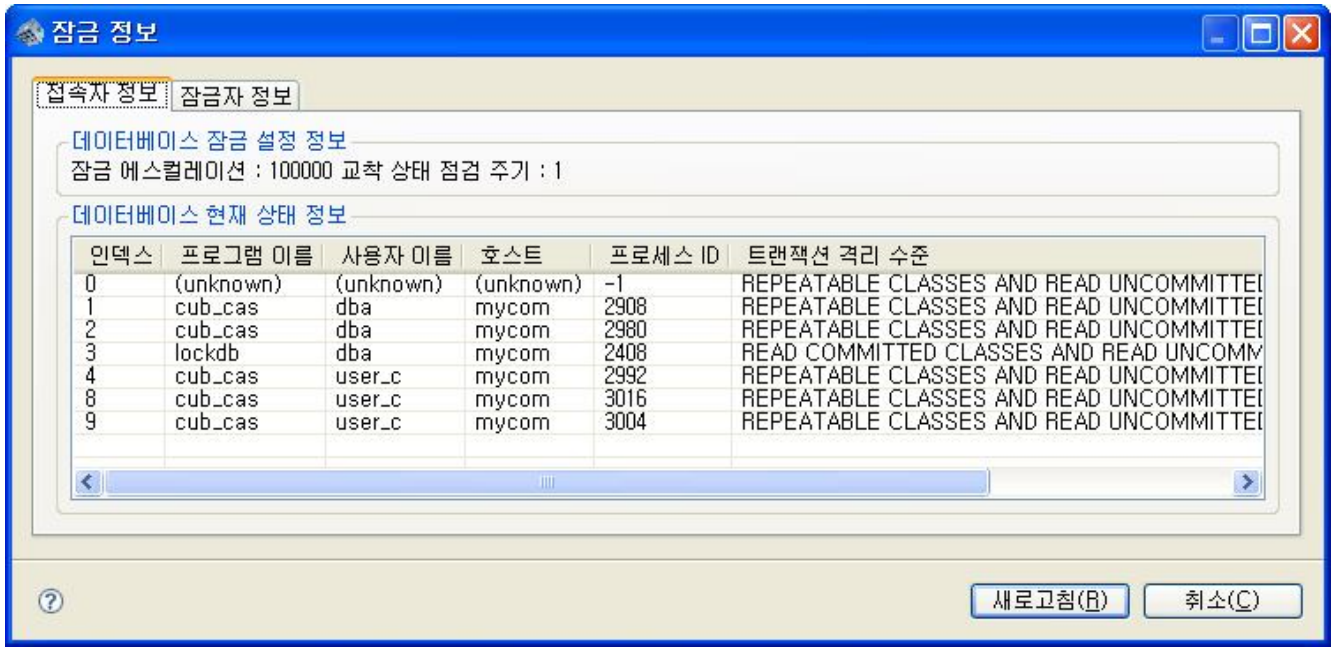


# 잠금 정보 확인 (계속)

CUBRID
More than open source!

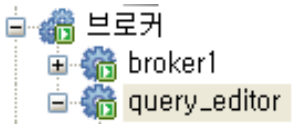
- 트랜잭션 정보



# 잠금 정보 확인 (계속)

CUBRID
More than open source!

- 트랜잭션 소유한 응용 확인
  - cub_cas 의 경우 CUBRID broker에서 정보 확인
  - 프로세스ID를 이용하여 어느 broker의 몇번째 ID인지 확인




브로커 상태 -query_editor

query_editor

| ID | PID | QPS◦ | L... | PSIZE | STATUS | DB | HOST | LAST ACCESS TIME | LAST CONNE... | SQL |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2908 | 42 | 64 | 5796 | BUSY | de... | localhost | 2009/09/23 12:06:06 | 2009/09/23 12:0... | execute update table_a set i = 3 where i = 3 |
| 2 | 2980 | 262 | 63 | 6352 | CLIENT WAIT | de... | localhost | 2009/09/23 12:07:10 | 2009/09/23 12:0... | |
| 3 | 2992 | 8 | 8 | 5540 | IDLE | de... | localhost | 2009/09/23 11:46:21 | 2009/09/23 11:4... | |
| 4 | 3004 | 5 | 5 | 5884 | IDLE | de... | localhost | 2009/09/23 09:05:04 | 2009/09/23 09:0... | |
| 5 | 3016 | 3 | 3 | 5284 | IDLE | de... | localhost | 2009/09/23 09:05:04 | 2009/09/23 09:0... | |

  - 앞서 확인한 프로세스 ID는 2908, 2980 이므로, query_editor broker 의 ID1, 과 ID2에 각각 해당함.
  - 2980이 X_LOCK을 선점하고 있었으므로, 필요하다면 해당 트랜잭션(ID2) 강제 종료 필요

# 잠금 정보 확인 (계속)

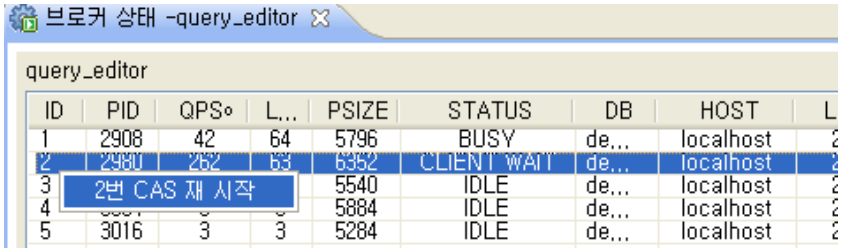
CUBRID
More than open source!

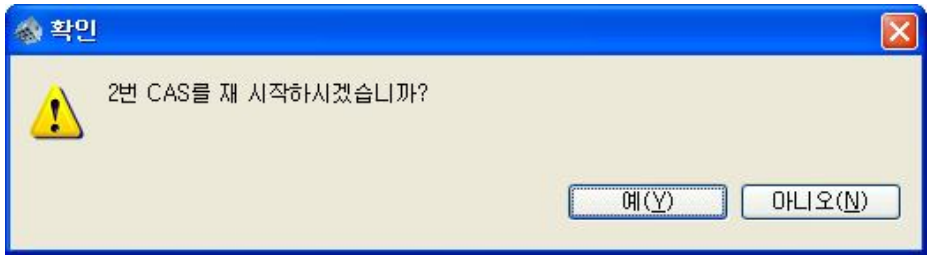
- sql 로그를 통한 작업 내용 확인
  - $CUBRID/log/broker/sql_log 아래 브로커의 cas ID를 이용하여 확인
    - <브로커이름>_<CAS ID>.sql.log : query_editor_2.sql.log

# 트랜잭션 관리

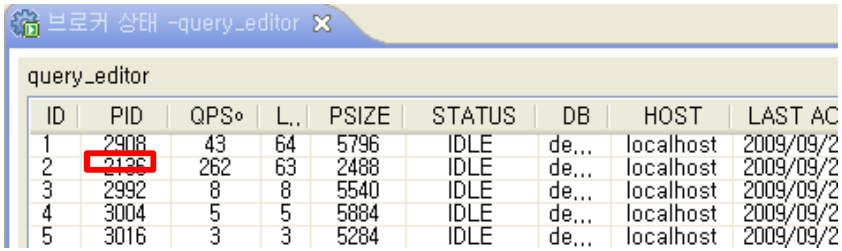
CUBRID
More than open source!

- broker 의 트랜잭션 종료
  - broker의 cas ID를 이용하여 해당 cas 재구동을 통한 트랜잭션 종료
    - 재구동시 해당 트랜잭션 강제 종료(rollback)
  - 앞서 ID2번의 종료




- 프로세스ID가 2136으로 변경되어, 재구동된 것을 알 수 있음.



# 데이터베이스 환경 설정

CUBRID
More than open source!

# 환경파일

CUBRID
More than open source!

- 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
More than open source!

- 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'
```

# 메모리 관련 설정

CUBRID
More than open source!

- data_buffer_pages
  - 데이터베이스 서버가 메모리 내에 캐시하고 있는 데이터 페이지 개수
  - num_data_buffers * database page size(데이터베이스 초기화할 때 지정한 페이지 크기로 default 4KB) 만큼 메모리 필요(default 25,000 설정 시 100MB)
  - 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.
- index_scan_oid_buffer_pages
  - 인덱스 스캔을 수행할 때 OID 리스트의 임시 저장을 위한 버퍼 페이지 개수를 설정
  - 디폴트 값은 4페이지이며, 최소값은 1이며, 최대값은 16이다.
- garbage_collection
  - 클라이언트에서 사용되지 않는 쓰레기(garbage)메모리를 해제할 것인지를 설정하는 파라미터로 디폴트 값은 no이다.

# 메모리 관련 설정 - 계속

CUBRID
More than open source!

- sort_buffer_pages
  - 정렬을 필요로 하는 질의를 처리할 때 사용되는 버퍼 페이지의 개수로 generic 볼륨이나 temp 볼륨을 이용한다.
  - Active client request 마다 하나의 sort buffer가 할당된다.
  - 정렬이 끝나면 할당되었던 메모리는 해제된다.
  - 16 ~ 500 page 범위의 값을 권장.
- temp_file_memory_size_in_pages
  - 질의에 관한 임시 결과를 캐시하는 버퍼 페이지 개수를 설정
  - 디폴트 값은 4이며, 최대값은 20까지 허용된다.
- thread_stacksize
  - 스레드의 스택 크기를 설정
  - 파라미터로 디폴트 값은 100*1024이다.
  - 운영체제가 허용하는 스택 크기를 초과할 수 없다.

# 디스크 저장장치 관련 설정

CUBRID
More than open source!

- temp_file_max_size_in_pages
  - 임시 볼륨을 최대로 할당할 수 있는 페이지 개수를 설정
  - default 값은 -1이며 무제한적으로 임시 볼륨이 생성
  - 0이면 임시 볼륨이 생성되지 않음
- temp_volume_path
  - 임시 볼륨의 생성 디렉토리를 지정하는 파라미터
  - default 값은 데이터베이스 초기 볼륨 위치
- volume_extension_path
  - 자동 확장 볼륨의 위치를 지정하는 파라미터
  - 값을 지정하지 않으면 데이터베이스 초기 볼륨 위치로 확장된다.
- dont_reuse_heap_file
  - 테이블 제거로 인해 생성된 빈 공간을 재할당하지 않기 위하여 설정
  - default값은 0이며 힙 파일 재할당을 수행하고, 1로 설정되면 힙 파일을 재할당하지 않는다.

# 디스크 저장장치 관련 설정 - 계속

CUBRID
More than open source!

- temp_file_max_size_in_pages
  - 임시 볼륨을 최대로 할당할 수 있는 페이지 개수를 설정
  - default 값은 -1이며 무제한적으로 임시 볼륨이 생성
  - 0이면 임시 볼륨이 생성되지 않음
- temp_volume_path
  - 임시 볼륨의 생성 디렉토리를 지정하는 파라미터
  - default 값은 데이터베이스 초기 볼륨 위치
- volume_extension_path
  - 자동 확장 볼륨의 위치를 지정하는 파라미터
  - 값을 지정하지 않으면 데이터베이스 초기 볼륨 위치로 확장된다.
- dont_reuse_heap_file
  - 테이블 제거로 인해 생성된 빈 공간을 재할당하지 않기 위하여 설정
  - default값은 0이며 힙 파일 재할당을 수행하고, 1로 설정되면 힙 파일을 재할당하지 않는다.

# 동시성/잠금 관련 설정

CUBRID
More than open source!

- deadlock_detection_interval_in_secs
  - 중단된 트랜잭션에 대해 교착 상태 여부를 탐지하는 주기를 초 단위로 설정
  - 교착 상태에 있는 트랜잭션 중 하나를 롤백시켜 교착 상태를 해결한다.
  - 디폴트 값은 1초이며, 탐지 주기가 길면 오랜 시간 동안 교착 상태를 탐지할 수 없으므로 주의한다.
- lock_escalation
  - 테이블의 Row Lock의 수가 설정값보다 클 경우 테이블 Lock으로 변환
  - 디폴트 값은 100,000
  - 값이 작으면, 메모리 잠금 관리에 의한 오버헤드가 적은 반면 동시성은 줄어든다.
  - 값이 크면 메모리 잠금 관리에 의한 오버헤드가 큰 반면 동시성이 향상된다.
- lock_timeout_in_secs
  - 잠금 대기 시간을 지정
  - 시간 이내에 잠금이 허용되지 않으면 해당 트랜잭션이 취소되고 오류가 반환된다.
  - 디폴트 값인 -1로 설정하면 대기 시간이 무제한이고, 0이면 잠금대기가 없다.

# 동시성/잠금 관련 설정 - 계속

CUBRID
More than open source!

- isolation_level
  - 트랜잭션의 고립수준을 설정
  - 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
  - READ COMMITTED: 완료된 트랜잭션만 Read허용

| 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<br>TRAN_READ_COMMITTED(4) | REPEATABLE READ CLASS with READ COMMITTED INSTANCES |
| TRAN_READ_UNCOMMITTED<br>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 |

# 로그 관련 설정

CUBRID
More than open source!

- checkpoint_interval_in_mins
  - 검사점이 수행되는 주기를 분 단위로 설정
  - 값이 클수록 데이터베이스를 복구에 시간이 많이 소요
  - 20분 내지 30분의 이내에서 주기를 설정할 것을 권장한다. 디폴트 값은 30이다.
- log_buffer_pages
  - 메모리에 캐시되는 로그 버퍼의 페이지 수를 설정. 디폴트 값은 50이다.
  - 설정값이 크면 데이터베이스 수정 연산이 많고, 길고 큰 트랜잭션이 많은 환경에서는 디스크 I/O가 감소되어 성능이 향상될 수 있다.
- media_failure_support
  - 저장 매체의 장애에 대비하여 보관 로그를 보존할지 여부를 설정
  - 디폴트 값인 yes로 설정하면 활성 로그가 꽉 찬 후 트랜잭션에 변경 사항이 있는 경우 활성 로그 전체를 보관 로그로 복사하여 보존한다.
  - no로 설정하면 활성 로그가 꽉 찬 후 생성된 보관 로그가 자동으로 삭제
  - no로 설정하면 보관 로그가 자동으로 삭제되므로 주의해야 한다.

# 쿼리 캐시 관련 설정

CUBRID
More than open source!

- max_plan_cache_entries
  - 메모리에 캐시하는 쿼리 플랜의 최대 개수를 설정(default 1,000)
  - 1이하 값이면 설정하지 않고, 1이상의 경우 쿼리 캐시(질의결과) 기능 동작
- query_cache_mode
  - 두 가지의 쿼리 캐시 모드 중 하나를 지정(default 0 수행하지 않음)
  - 제 1 쿼리 캐시 모드는 모든 쿼리에 대해 쿼리 캐시 기능을 적용
  - 제 2 쿼리 캐시 모드는 /*+ QUERY_CACHE(1) */이라는 힌트가 주어진 쿼리에 대해서만 쿼리 캐시 기능을 적용
- max_query_cache_entries
  - 쿼리 캐시 기능을 적용할 쿼리의 최대 개수를 지정(default -1)
  - 1이하 값이면 설정하지 않고, 1이상의 경우 쿼리 캐시(질의결과) 기능 동작
  - 조회 데이터가 변화 없고, 반복된 동일 쿼리가 수행될 경우 성능 향상
  - max_plan_chache_entries, query_cache_mode에 종속적

# 구문/타입 관련 설정

CUBRID
More than open source!

- 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
- compat_numeric_division_scale
  - 나눗셈 연산의 결과 값에 대하여 소수점 이하 자리 수 결정(defaut no)
  - default는 no이며 몫의 소수점 이하 자리수를 9로 한다.(반올림 없음)
  - yes로 설정하면 몫의 소수점 이하 자리수가 피 연산자의 소수점 자리 수

# 구문/타입 관련 설정 - 계속

CUBRID
More than open source!

- 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)
- single_byte_compare
  - 문자열 비교 시 1byte 씩 처리하도록 하며, 유니코드 사용 시(default no)

# 기타 설정

CUBRID
More than open source!

- 통신 서비스 관련
  - cubrid_port_id
    - Master Process Port
    - default 값은 1523
    - 1523이 이미 사용 중일 경우 이 파라미터를 다른 번호로 변경해야 한다.
- 클라이언트/서버 요청 관련
  - max_clients
    - 서버에 동시 연결 가능한 클라이언트의 최대 개수로써 동시에 수행중인 트랜잭션 전체의 개수를 의미한다.
    - 따라서, 클라이언트와 연결을 담당하는 server thread의 개수와 같은 값이다.
    - default 값은 50
    - 실제 동시 사용자 수 고려
- 서버 재 구동 설정
  - auto_restart_server
    - 장애로 인하여 서버가 종료된 경우 자동으로 재 구동
    - default는 yes로 자동 재 구동된다.

# 기타 설정 - 계속

CUBRID
More than open source!

- 자바 스토어드 프로시저 관련
  - java_stored_procedure
    - 자바 스토어드 프로시저의 사용여부 조정.
    - default는 no이며, yes로 설정하면 자바 스토어드 프로시저를 사용한다.
- 복제 관련
  - replication
    - 복제 기능을 활성화(default no로 설정하지 않음)
    - yes로 설정하면 복제 로그를 생성하고, 복제 그룹의 마스터가 됨을 의미한다.

# 기타 설정 - 계속

- 트랜잭션 처리 관련 - Log Flush Thread(LFT)
  - async_commit
    - 비동기식 commit 기능을 활성화(default no로 설정하지 않음)
    - 커밋 로그가 디스크에 플러시 되기 이전에 클라이언트에게 커밋을 완료 처리
    - 데이터베이스 서버에 장애가 발생하면 이미 커밋 완료된 트랜잭션을 복구할 수 없다.
  - group_commit_interval_in_msecs
    - 설정값 동안 발생한 커밋을 group으로 취합하여 커밋수행(default 0 설정하지 않음)
    - 커밋 로그가 동시에 디스크에 플러시되도록 하여 커밋에 대한 성능을 향상

# 기타 설정 - 계속

CUBRID
More than open source!

- Index Scan 관련
  - index_scan_in_oid_order
    - 인덱스를 scan한 후 검색 결과 데이터를 가져오는 순서를 OID 순으로 지정
    - ordering 없이 쿼리가 수행되므로 쿼리 성능향상
    - default값인 no이면 데이터 순서대로 결과를 가져온다.
- Insert mode 관련
  - insert_execution_mode
    - 쿼리는 클라이언트에서 실행 계획을 짜서 서버에서 수행을 하는데, 서버 측에서 바로 삽입을 수행 하여 입력 성능향상(default 1이며, 범위는 1~7)
    - 설정방법

```
INSERT_SELECT type    = 1
INSERT_VALUES type    = 2
INSERT_DEFAULT type = 4
```

```
INSERT_SELECT와 INSERT_VALUES   ➔ insert_execution_mode=3설정(1 + 2 = 3)
INSERT_SELECT와 INSERT_DEFAULT ➔ insert_execution_mode=5설정(1 + 4 = 5)
```

# CUBRID 복제

CUBRID
More than open source!

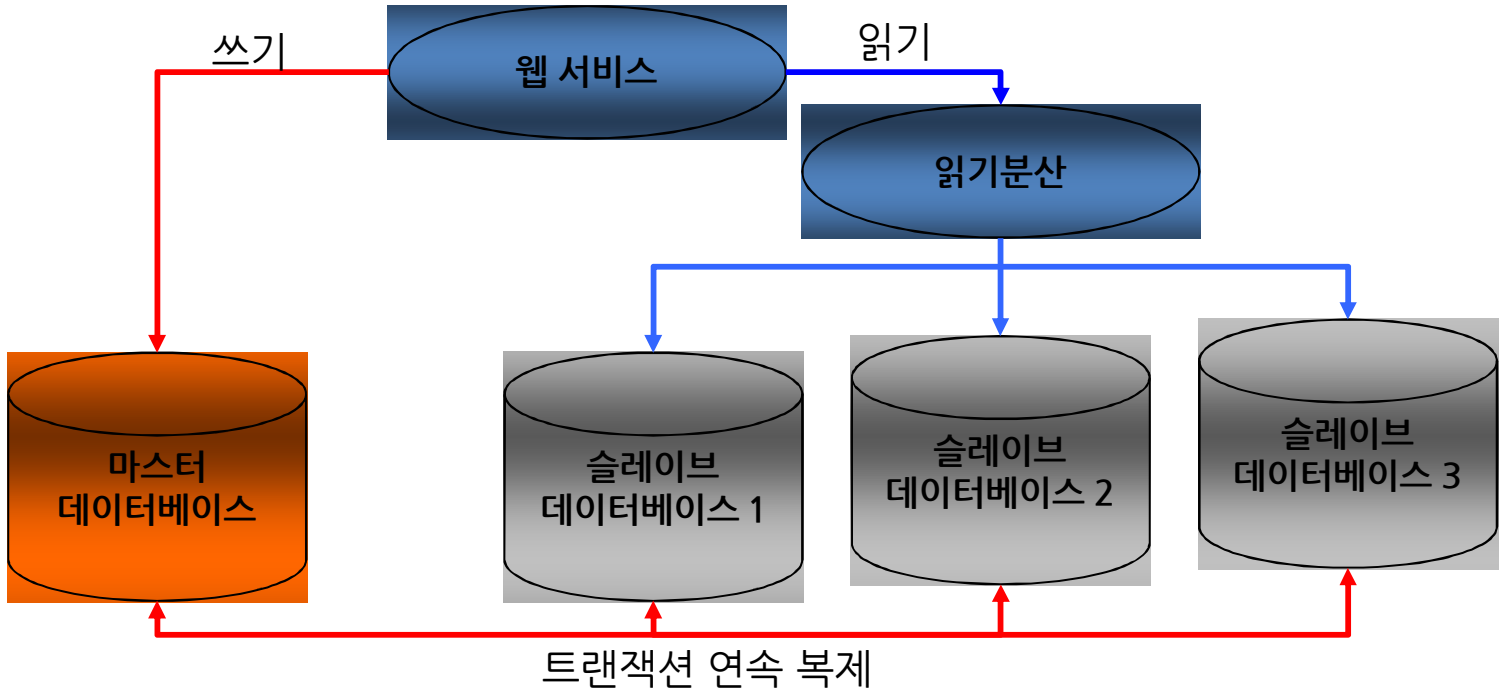
# 소개

CUBRID
More than open source!

- 하나의 데이터베이스에 저장된 객체를 물리적으로 분리된 다른 데이터베이스에 복사하여 두 개 이상의 데이터베이스 서버에서 사용될 수 있도록 하는 분산 데이터베이스 기술 중 하나이다.
- 같은 객체를 이용하는 응용의 접근을 여러 데이터베이스 서버에 분산시킴으로써 성능을 높일 수 있다.
- 복제된 데이터베이스 서버를 다른 용도로 사용할 수 있도록 하여 서로 다른 운영 요구 사항을 만족시킬 수 있다.
- 운영 중인 데이터베이스 서버에 장애가 발생했을 경우 복제 데이터베이스 서버로 대체함으로써 장애 발생 시 긴급하게 대처할 수 있다.

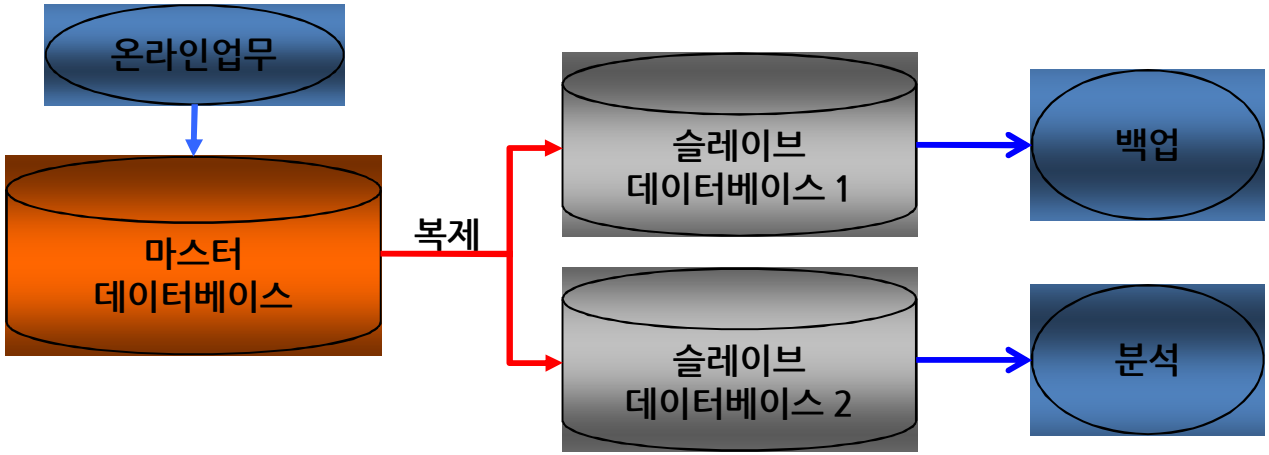
CUBRID
More than open source!

- 부하 분산을 통한 성능 향상
  - 읽기 작업에 대한 부하 분산
    - 마스터 데이터베이스에 대하여 쓰기 작업을 수행하고, 여러 개의 복제된 슬레이브 데이터베이스에 대하여 읽기 작업을 수행함으로서 읽기 작업이 많은 인터넷 서비스 같은 경우 효율적이다.
    - 게시판, 뉴스 등에 사용 가능



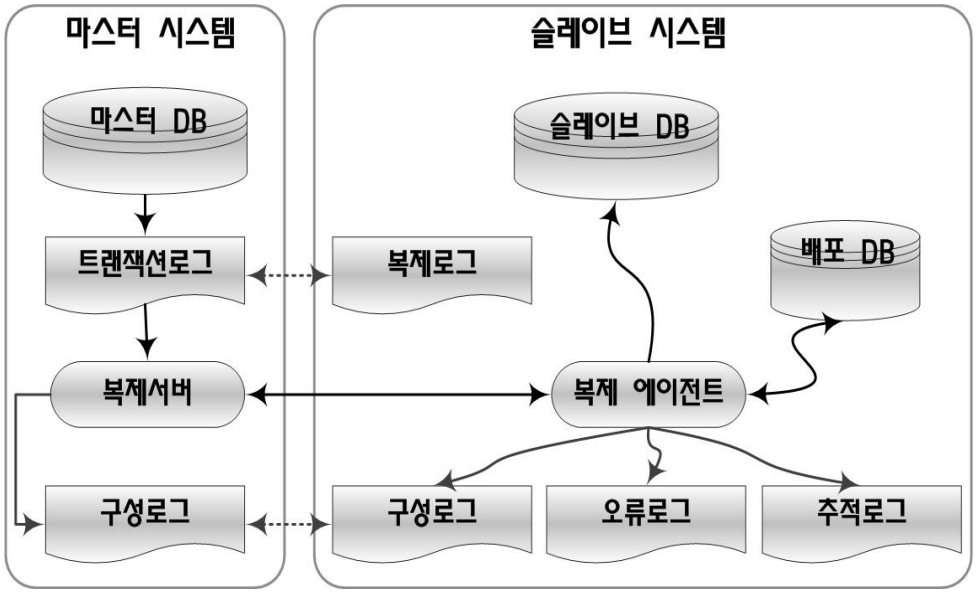
CUBRID
More than open source!

- 장애 대비 가용성 향상
  - 마스터 데이터베이스 장애 대체
    - 마스터 데이터베이스에 장애가 발생 하였을때 슬레이브 데이터베이스를 이용하여 서비스 지속
    - 슬레이브 데이터베이스중의 하나를 마스터 데이터베이스로 설정하고, 응용에서 쓰기 작업을 새로운 마스터 데이터베이스에 대하여 진행할 수 있도록 설정
  - 분리작업을 통한 유연성 향상
    - 별도의 호스트에 마스터 데이터베이스와 같은 내용을 가지는 데이터베이스를 확보할 수 있다.
    - 마스터 데이터버에스에 추가적인 부하 없이 슬레이브 데이터베이스를 통하여 데이터 분석 작업 등을 수행할 수 있다.



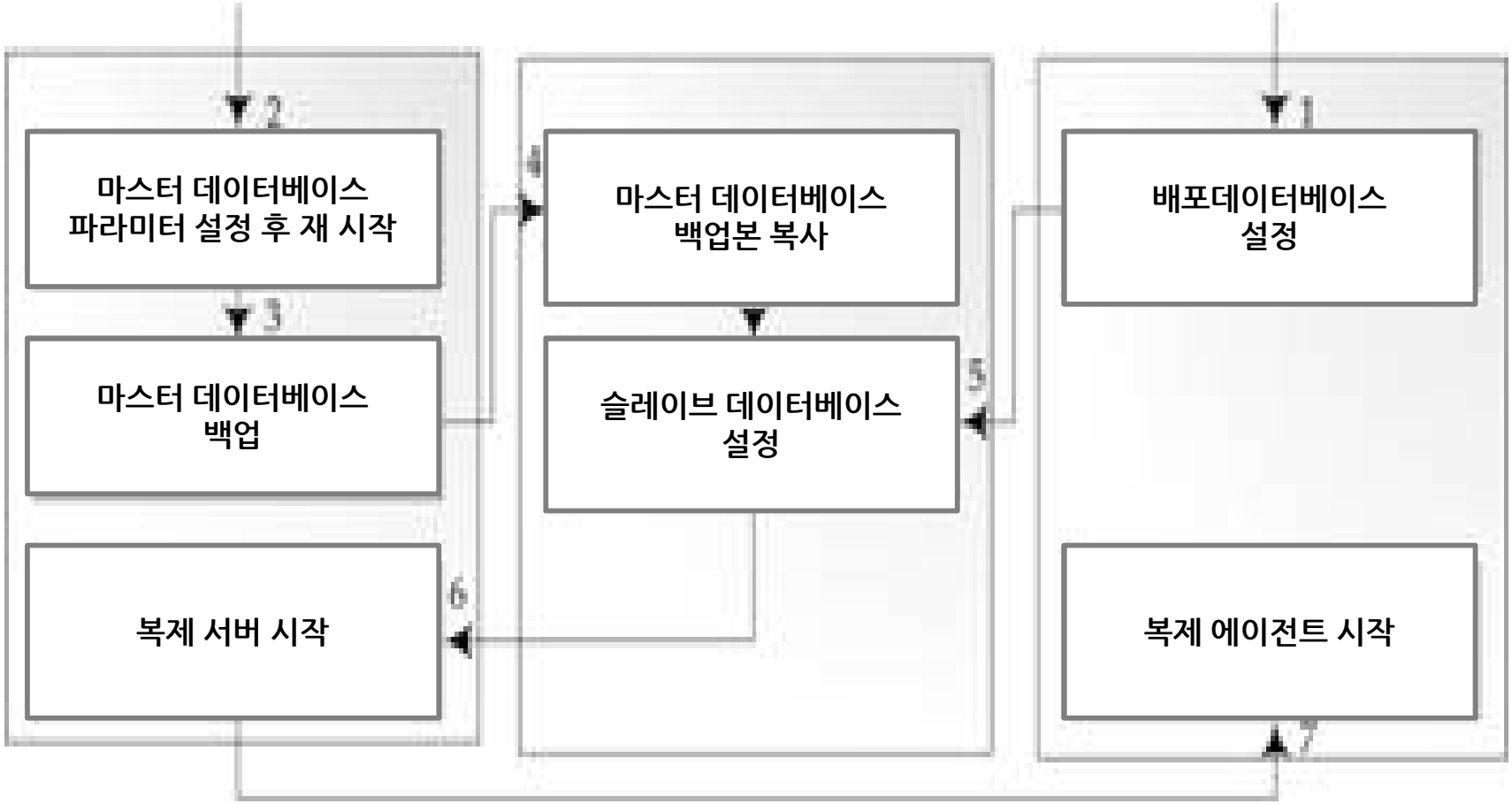
# 특징

- 트랜잭션 로그 기반의 복제
- 단 방향 복제
- 비 동기식 복제
- 중단 없는 온라인 복제
- 기본 키 기반의 복제
- 1:N의 복제 구성 지원
- 그룹 복제 지원
- 스키마 복제 지원
- 스키마의 독립성
- 스냅샷 동기화(on-line복제)
- 반 자동 복제 절체 지원
- 복제 상태 관리 기능 지원
- 복제 성능 모니터링 지원

CUBRID
More than open source!

- 복제 구성



- 마스터 데이터베이스 생성
  - 복제 테스트를 위하여 마스터 데이터베이스 생성
    - 데이터베이스를 생성한 뒤, 복제에 사용할 테이블을 등록한다.
    - primary key 가 존재해야 한다.

```
% cubrid createdb masterdb
% cubrid server start masterdb
% csql masterdb
...
csql> create class info (
csql> id int,
csql> name char(10),
csql> primary key(id)
csql> )
csql> ;run
```

CUBRID
More than open source!

## ● 배포 데이터베이스 설정

- 복제에 필요한 제반 정보 등록
  - 배포 데이터베이스를 생성한다.(보안을 위하여 DBA암호설정을 권장)
  - 배포 데이터베이스를 생성할 경로에서 명령을 수행한다.

```
% repl_make_distdb distdb -p distdb_password
##################################################################
…
##################################################################
STEP 1 : 배포 DB를 생성 중입니다. 잠시만 기다려 주십시오.
STEP 2 : 배포 DB 서버를 구동하고 있습니다. 잠시만 기다려 주십시오.
STEP 3 : 배포 DB 의 DBA 계정을 설정합니다.
STEP 4 : 복제에 필요한 테이블들을 생성합니다.
```

STEP 5 : 복제 대상 마스터 DB 정보를 입력합니다.

1. 마스터 DB의 이름을 입력하시오. >> masterdb
2. 마스터 DB가 위치한 호스트의 IP 주소를 입력하시오.
   - IP 주소가 부정확하게 입력되면 복제가 수행되지 않습니다.

   마스터 DB IP >> 192.168.2.1
3. 복제서버(cubrid repl_server)가 사용하는 TCP/IP 포트번호를 입력 하시오.>> 5627
4. 복제에 필요한 복제로그을 저장할 디렉토리를 입력하시오.
   - /home1/cubrid/CUBRID이면 엔터 키를 입력하시오. >> /home/replication_logs

STEP 6 : 복제 환경 변수를 설정합니다.

1. 추적 로그를 저장할 디렉토리를 입력하시오.
   - /home1/cubrid/CUBRID이면 엔터 키를 입력하시오. >> /home/replication_logs
2. 에러로그를 저장할 디렉토리를 입력하시오.
   - /home1/cubrid/CUBRID이면 엔터 키를 입력하시오. >> /home/replication_logs
3. 복제 에이전트(repl_agent)의 상태 표시를 위한 TCP/IP 포트번호를 입력하시오.>> 33333
4. 복제 지연 시간 log 파일의 크기(line 수)를 입력하시오. >> 200
   ➔ 복제 성능 모니터링 목적으로 복제 지연시간 로그를 남기는데, 이때 기록할 파일의 크기
5. 네트워크 오류시 복제의 재 시작 여부를 입력하시오.(y/n) >> y
   ➔ n로 설정하면 그대로 복제 기능이 종료한다.

CUBRID
More than open source!

- 마스터 데이터베이스 설정 및 백업
  - 마스터 데이터베이스 환경 설정
    - cubrid.conf의 replication설정값을 yes로 수정하여 트랜잭션 로그를 생성
    - 수정한 설정값을 반영시키기 위해 데이터베이스 서버를 재 구동 시킨다.
      - % cubrid server restart masterdb
  - 마스터 데이터베이스 백업
    - 슬레이브 데이터베이스를 생성하기 위해 마스터 데이터베이스를 백업한다.
    - 로그의 정리로 인한 복제에 오류가 발생할 수 있으므로 -r 옵션을 사용하지 않는다.
      - % cubrid backupdb masterdb
    - 백업 후 백업 파일들을 확인 한다.
      - % ls -l masterdb_bk*
      - masterdb_bkvinf, masterdb_bk0v000, …

CUBRID
More than open source!

## 슬레이브 데이터베이스 생성

- 슬레이브 데이터베이스 생성 환경 구성
  - 슬레이브 데이터베이스는 마스터 데이터베이스와 똑 같은 환경(디렉토리 정보) 을 가지고 생성되므로, 사전에 동일한 디렉토리를 생성해야 한다.
- 마스터 데이터베이스 백업 파일 이동
  - 마스터 데이터베이스의 백업화일을 슬레이브 데이터베이스를 생성한 호스트로 옮긴다. 이때 각 파일이 생기는 디렉토리 역시 마스터 데이터베이스가 있던 호스트와 동일하게 위치해야 한다.
    - masterdb_bkvinf 가 있던 디렉토리
    - masterdb_bk0v000, … 가 있던 디렉토리
- 슬레이브 데이터베이스 생성
  - 슬레이브 데이터베이스 생성시 마스터데이터베이스이름_bkvinf 가 있는 디렉토리에서 명령을 수행한다.
    - **repl_make_slavedb <마스터 데이터베이스이름> <슬레이브 데이터베이스 이름> -u <슬레이브 데이터베이스 사용자 아이다> -p <슬레이브 데이터베이스 사용자 암호>**

```
% repl_make_slavedb masterdb slavedb -u repl2 -p repl2
##############################################################
…
##############################################################
```

**STEP 1 : 마스터 DB의 백업본을 복구하기 위한 사전 작업을 진행합니다.**

1. 구축할 슬레이브 DB의 볼륨이 저장될 디렉토리 경로를 입력하시오.
   - 현재 디렉토리이면 엔터 키를 입력하시오. >> /home/slavedb
2. 구축할 슬레이브 DB의 로그 볼륨이 저장될 디렉토리 경로를 입력하시오.
   - 현재 디렉토리이면 엔터 키를 입력하시오. >> /home/slavedb

**STEP 2 : 슬레이브 DB 백업본을 복구합니다.**

- 백업 파일과 백업볼륨 정보 파일이 반드시 현재 디렉토리에 존재해야 합니다.
- 슬레이브 DB를 복구하고 있습니다. 잠시만 기다려 주시기 바랍니다...
- 슬레이브 DB 가 복구되었습니다.

# 복제 구성 및 운영 - 계속

CUBRID
More than open source!

STEP 3 : 복구된 최종 로그 번호와 슬레이브 DB 정보를 배포 DB에 기록합니다.

1. 배포 DB 이름을 입력하시오. >> distdb
2. 배포 DB에 접속하기 위한 DBA 계정의 암호를 입력하시오. >> distdb_password
   - 현재 디렉토리이면 엔터 키를 입력하시오. >> /home/slavedb

STEP 4 : 슬레이브 DB를 생성하고 구동합니다.

STEP 5 : 슬레이브 DB 구축 후 사후 처리 단계

- - 이 작업을 위해서는 마스터 DB의 DBA 계정이 필요합니다.
- - 마스터 DB의 DBA 계정을 입력하시오. >> enter입력

➔ masterdb의 DBA계정 패스워드를 설정하지 않았으므로 enter

# 복제 구성 및 운영 - 계속

CUBRID
More than open source!

STEP 6 : 파라미터 설정 단계

1. perf_poll_interval - 복제 지연시간을 측정하는 단위(초)
2. size_of_log_buffer - 복제에이전트의 로그버퍼 크기(페이지)
3. size_of_cache_buffer - 복제에이전트의 복제로그버퍼 크기(페이지)
4. size_of_copylog - 복제로그의 페이지 수
5. index_replication - 인덱스 복제 여부
6. for_recovery - 마스터 교체용 복제 여부
7. log_apply_interval - 복제 수행 주기(초)
8. restart_interval - 슬레이브 재 접속 주기(초)

-- 변경하고자 하는 파라미터 번호 (q - 중지) q

## 복제 서버 및 에이전트 구동

### 복제 서버 구동

- 마스터 데이터베이스가 있는 호스트에서 복제 서버를 구동한다.
- 다음 명령을 이용하여 복제 서버를 구동한다.
  - **% cubrid repl_server start <마스터데이터베이스이름> <복제서버 포트 번호>**
  - % cubrid repl_server start masterdb 5627
  - 중지할 경우 : % cubrid repl_server stop masterdb

### 복제 에이전트 구동

- 슬레이브 데이터베이스가 있는 호스트에서 복제 에이전트를 구동한다.
- 다음 명령을 이용하여 복제 에이전트를 구동한다.
  - **% cubrid repl_agent start <배포데이터베이스이름> [배포데이터베이스 dba 암호]**
  - % cubrid repl_agent start distdb distdb_password
  - 중지할 경우 : % cubrid repl_agent stop distdb

CUBRID
More than open source!

- 복제 확인
  - 마스터 데이터베이스에 입력/수정/삭제와 동시에 슬레이브 데이터베이스에서 검색을 통해 확인한다.

```
% csql masterdb
…
csql> insert into info values(1, 'name1')
csql> insert into info values(2, 'name2')
csql> ;x
1 rows inserted
Current transaction has been committed.
1 command(s) successfully processed.

csql> update info set name='name' where id = 1
csql> ;x
1 rows updated
Current transaction has been committed.
1 command(s) successfully processed.

csql> delete from info where id = 1
csql> ;x
1 rows deleted
Current transaction has been committed.
1 command(s) successfully processed.
```

```
% csql slavedb
…
# masterdb 에 입력후
csql> select * from info;
csql> ;x
          id  name
================================
          1  'name1     '
1 rows selected.
# masterdb 에 수정후
csql> select * from info;
csql> ;x
          id  name
================================
          1  'name      '
1 rows selected.
# masterdb 에 삭제후
csql> select * from info;
csql> ;x
There are no results.
```

CUBRID
More than open source!

- 복제 서버, 에이전트 모니터링
  - 아래 명령어를 이용하여 복제 서버와 복제 에이전트의 구동여부를 확인한다.

```
% cubrid repl_server status
또는
% cubrid repl_agent status
% cubrid replication status
```

```
repl_agent distdb (rel 8.4, pid 22203)
repl_server masterdb (rel 8.4, pid 12341)
```

CUBRID
More than open source!

- CUBRID의 모든 프로세스의 상태를 출력

```
% cubrid service status
```

```
@ cubrid master status
++ cubrid master is running.

@ cubrid server status

Server slavedb (rel 8.4, pid 11325)
Server distdb (rel 8.4, pid 31440)
Server masterdb (rel 8.4, pid 29191)

@ cubrid broker status
  NAME              PID   PORT  AS  JQ     REQ   TPS   AUTO    SES SQLL   CONN
==========================================================================
* query_editor   12149 30300   5   0       0    ---      ON    OFF ON:A  AUTO
* broker1        12161 33300   5   0       0    ---      ON    OFF ON:A  AUTO

@ cubrid manager server status
++ cubrid manager server is running.

@ cubrid replication status
 repl_agent distdb (rel 8.4, pid 22203)
 repl_server masterdb (rel 8.4, pid 12341)
```

CUBRID
More than open source!

- 복제 성능 모니터링
  - 배포 데이터베이스 생성 시 정의한 추적 로그 이용

```
% tail -f <배포 데이터베이스 이름>.perf
```

```
-------------------------------------------------------------------------------
 No. master_db_name      tran_index        master_time          slave_time      delay
-------------------------------------------------------------------------------
001              mdb              8 2008/10/17 10:03:24 2008/10/17 10:03:24          0
002              mdb             -1 ----/--/-- --:--:-- 2008/10/17 10:04:07          0
003              mdb             10 2008/10/17 10:04:07 2008/10/17 10:04:07          0
004              mdb             11 2008/10/17 10:04:07 2008/10/17 10:04:07          0
005              mdb             -1 ----/--/-- --:--:-- 2008/10/17 10:04:07          0
```

    - tran_index가 -1로 출력되는 부분은 복제가 완료되었다는 표시이다.

CUBRID
More than open source!

- CUBRID의 모든 프로세스의 상태를 출력
  - 복제 서버 중지

```
% cubrid repl_server stop  <마스터 데이터베이스 이름>
% cubrid repl_server stop  masterdb
    cubrid repl_server sample_master notified of shutdown.
    This may take several minutes. Please wait.
```

  - 복제 에이전트 중지

```
% cubrid repl_agent stop  <배포 데이터베이스 이름>
% cubrid repl_agent stop  distdb
    cubrid repl_agent stop sample_dist notified of shutdown.
    This may take several minutes. Please wait.
```

# 복제 주의사항

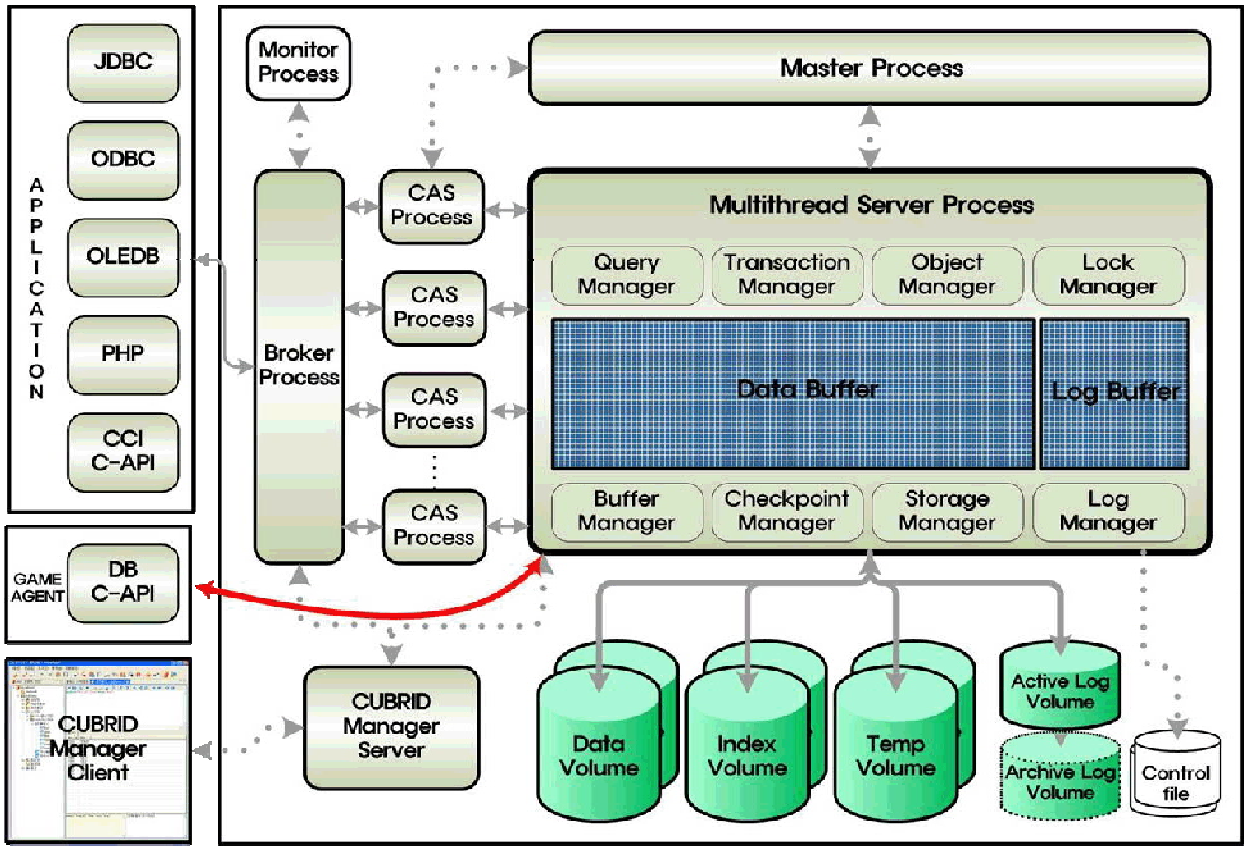
CUBRID
More than open source!

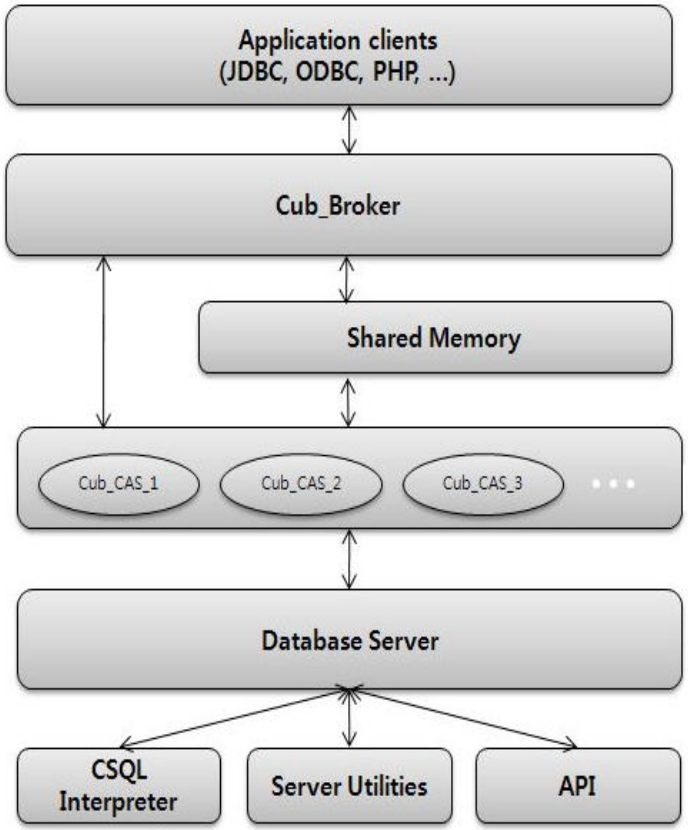
- 제약사항
  - Unix 계열에서만 지원한다.
  - 동일 플랫폼으로 구성에서 복제 구성을 해야 한다.
  - 기본 키가 설정된 테이블에 대해서만 복제할 수 있다.
  - 마스터 데이터베이스의 모든 테이블에 기본키가 설정되어 있어야만 마스터 데이터베이스를 슬레이브 데이터베이스로 교체할 수 있다.
- 비동기 방식
  - 대량의 데이터가 입력/수정/삭제 되는 복제 시간이 지연될 수 있다.
- 슬레이브 데이터베이스 재 성성
  - 재생성 시 이전에 사용되던 추적로그, 복제로그들은 모두 삭제/이동 해야 한다.
- 기타
  - timestamp 값은 마스터 데이터베이스의 값과 동일하게 기록된다.
  - 슬레이브 데이터베이스 장애
    - 슬레이브 데이터베이스 장애로 인하여 서비스 중단 시 복제 에이전트도 같이 중지된다
    - 따라서 이 경우 슬레이브 데이터베이스 구동 후 복제 에이전트도 구동 시켜야 한다.
  - 배포 데이터베이스의 내용은 임의로 수정되어서는 안 된다.

# CUBRID BROKER

CUBRID
More than open source!

# 구조


APPLICATION
JDBC
ODBC
OLEDB
PHP
CCI C-API
GAME AGENT
DB C-API
CUBRID Manager Client
Monitor Process
Master Process
Broker Process
CAS Process
CAS Process
CAS Process
CAS Process
Multithread Server Process
Query Manager
Transaction Manager
Object Manager
Lock Manager
Data Buffer
Log Buffer
Buffer Manager
Checkpoint Manager
Storage Manager
Log Manager
CUBRID Manager Server
Data Volume
Index Volume
Temp Volume
Active Log Volume
Archive Log Volume
Control file


Application clients (JDBC, ODBC, PHP, ...)
Cub_Broker
Shared Memory
Cub_CAS_1
Cub_CAS_2
Cub_CAS_3
Database Server
CSQL Interpreter
Server Utilities
API

CUBRID
More than open source!

- Broker
  - 클라이언트의 요청을 응용서버로 전달하고, 응용서버가 처리한 결과를 클라이언트로 보내준다.
  - ODBC, JDBC등 데이터베이스 연결 시 Broker를 통해 관리
  - Broker
    - 응용서버 CAS 를 관리
  - Shared memory
    - 응용서버와 브로커의 정보(현재상태, 요청처리건수 등)을 관리한다.
- Service pool
  - 클라이언트의 요청을 처리하는 응용서버의 그룹들
  - Broker로부터 요청을 받으면 데이터베이스로 질의를 하거나, 스크립트의 번역을 수행한다.

CUBRID
More than open source!

- 구동
  - CUBRID가 설치되어 있는 호스트의 브로커 구동
  - 큐브리드 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.
```

CUBRID
More than open source!

- 종료
  - CUBRID가 설치되어 있는 호스트의 브로커 종료
  - 큐브리드 service 구동 시 자동으로 종료된다.

```
% cubrid broker stop
@ cubrid broker stop
++ cubrid broker stop: success
```

➔ 이미 구동되어 있을 경우

```
++ cubrid broker is not running.
```

CUBRID
More than open source!

- 특정 응용서버그룹(broker) 구동
  - 특정 broker 만을 구동시킬 수 경우
  - 평소에는 사용되지 않으나 특정시점에 구동을 할 경우

```
% cubrid broker on broker_name
```


- 특정 응용서버그룹(broker) 정지
  - 특정 broker만 종료할 경우

```
% cubrid broker off broker_name
```

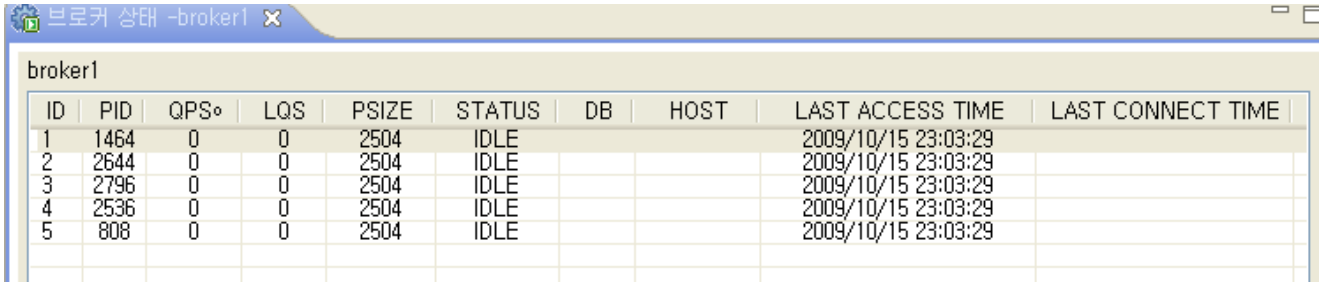
CUBRID
More than open source!

- 응용서버(CAS) 재 시작
  - 서비스 중이거나 개발 중 일때 어느 응용서버가 장시간 BUSY 인 상태로 있는 경우가 있을 수 있다. 이 경우 다른 서비스에 영향을 줄 수도 있으므로 해당 서비스만을 강제로 종료시킨 후 재 구동 시킬 필요가 있다.
    - 재 시작할 응용서버의 번호(응용서버 ID)를 선택한 후, 확인을 click 한다.

# 모니터링

CUBRID
More than open source!

- monitoring
  - CUBRID BROKER의 현재 요청 처리상태를 보여준다.



CUBRID
More than open source!

- 명령어 이용 : broker status

```
% cubrid broker status [option] [broker-name]
% cubrid broker status  -t 0 -s 2 -q broker1
```

- -t <sec> : 지정한 가장 최근의 시간 동안에 처리된 job의 개수를 출력함
- -q : job queue에 대기중인 job의 정보 출력
- -s <sec> : 지정한 간격으로 모니터링 정보를 refresh 함
  - 종료는 'q' 문자로 처리.
- -b : 브로커 정보만 출력
- broker_name : 지정한 브로커에 대한 정보 출력

```
C:₩Documents and Settings₩Administrator>cubrid broker status
@ cubrid broker status
% query_editor  - cub_cas [3288,30000] C:₩CUBRID/log/broker/query_editor.access C:₩CUBRID/
 JOB QUEUE:0, AUTO_ADD_APPL_SERVER:ON, SQL_LOG_MODE:ALL:100000
 LONG_TRANSACTION_TIME:60, LONG_QUERY_TIME:60, SESSION_TIMEOUT:300
 KEEP_CONNECTION:AUTO, ACCESS_MODE:RW
------------------------------------------------
ID   PID   QPS   LQS   PORT PSIZE STATUS         CPU
------------------------------------------------
 1  4028     0     0 30001  2500 IDLE          0.00
 2  2712     0     0 30002  2500 IDLE          0.00
 3  2128     0     0 30003  2500 IDLE          0.00
 4  1392     0     0 30004  2496 IDLE          0.00
 5  3992     0     0 30005  2500 IDLE          0.00
```

CUBRID
More than open source!

- Broker Status

| 값 | 설 명 |
|---|---|
| ID | 응용 서버 ID |
| PID | 응용서버의 process ID |
| QPS | 초당 처리한 질의의 수 |
| LPS | 초당 처리되는 장기수행 질의의 수 |
| PSIZE | 프로세스 크기 |
| STATUS | 응용서버의 현재 상태<br>IDLE : 요청을 받기 위한 대기 상태<br>BUSY : 요청을 처리하고 있음<br>CLIENT WAIT : 처리중은 요청은 없으나, 트랜잭션이 끝나지 않은 상태로 클라이언트의 요청을 기다리고 있는 상태<br>CLOSE WAIT : 트랜잭션이 끝나고 클라이언트의 연결 종료를 기다리는 상태 |
| DB | 접속중인 데이터베이스 이름 |
| HOST | 접속중인 서버 이름 |
| LAST ACCESS TIME | 가장 최근의 요청 처리 시간 |
| LAST CONNECTION TIME | 가장 최근에 데이터베이스와 연결을 맺은 시간 |

CUBRID
More than open source!

- broker 상태 정보 갱신
  - 브로커의 상태를 주어진 시간(초) 간격으로 갱신하여 확인
  - 명령어 이용

```
% cubrid broker status -s 1 broker1
```

➔ broker1에 대한 정보 1초 간격으로 갱신

```
% cubrid broker status -s 1
```

➔ broker이름을 명시하지 않을 경우 모든 broker의 정보를 1초 간격으로 갱신

CUBRID
More than open source!

- 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 APPL_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
[%BROKER_NEW]
SERVICE                 =ON
BROKER_PORT             =34000
MIN_NUM_APPL_SERVER     =5
MAX_NUM_APPL_SERVER     =40
APPL_SERVER_SHM_ID      =34000
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 - BROKER추가

CUBRID
More than open source!

- 아래와 같이 브로커를 생성
  - 브로커이름 : my_broker
  - 브로커포트 : 40000
  - 응용서버의 개수(최소/최대) : 5~10
    - MIN_NUM_APPL_SERVER, MAX_NUM_APPL_SERVER 이용
- 브로커 중지 및 구동
- 추가 브로커 확인

CUBRID
More than open source!

- 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
    - SESSION_TIMEOUT, KEEP_CONNECTION
  - 환경변수 및 설정 값이 잘못되어 있을 경우 재 구동 시 오류가 발생하며 구동되지 않는다.

CUBRID
More than open source!

| Parameter 이름 | 설 명 | 값 |
|---|---|---|
| MASTER_SHM_ID | UniCAS의 전체적인 운영을 위해 필요한 공유 메모리 번호를 지정하는 파라미터<br>%% 문자 뒤에 지정해야 하며 시스템에서 고유한 값이어야 한다. | 30001<br>(int) |
| ADMIN_LOG_FILE | admin log file(CUBRID CAS start/stop/restart 등을 기록) 의 위치와 이름을 지정 | log/admin.log<br>(char) |
| <Broker 이름><br>(Broker name) | 응용서버그룹(Broker)의 이름을 지정하며, 고유한 값으로 지정해야 한다.<br>% 뒤에 지정하며, 대소문자를 구분하지 않음 | BROKER1<br>(char) |
| SERVICE | CUBRID CAS 의 구동시 응용서버그룹의 구동 여부를 지정한다 | ON / OFF |
| BROKER_PORT<br>(Broker Port) | Broker 의 port 번호를 지정한다. 시스템에서 유일한 값이어야 하며, 이 port 를 이용하여 응용프로그램(JDBC,ODBC 등)과 통신한다.<br>방화벽이 구성되어 있을 경우 설정된 포트를 open하여야 한다.<br>Windows에서 Broker의 CAS포트는 “Broker포트+1” 순서로 생성된다. | 30000<br>(int) |

CUBRID
More than open source!

| Parameter 이름 | 설 명 | 값 |
|---|---|---|
| APPL_SERVER_SHM_ID | Broker 와 해당 그룹의 응용서버들이 사용하는 공유메모리의 key 값 | 30000<br>(int) |
| MIN_NUM_APPL_SERVER<br>(AS Minimum) | 응용서버의 초기 개수(처음 구동시 구동되는 개수)를 지정 | 5<br>(int) |
| MAX_NUM_APPL_SERVER<br>(AS Maximum) | 응용서버가 증가할 수 있는 최대 개수를 지정 | 40<br>(int) |
| APPL_SERVER_ MAX_SIZE | 응용서버의 허용가능한 프로세스 크기를 MByte 단위로 지정 | 20<br>(int) |
| LOG_DIR | access log 가 기록되는 디렉토리 지정 | log/broker<br>(char) |
| ERROR_LOG_DIR | error log 가 기록되는 디렉토리 지정 | log/broker/error_log<br>(char) |
| SQL_LOG | 응용서버가 처리하는 transaction 에 대하여 기록을 남길지 여부를 지정 | ON / OFF |

CUBRID
More than open source!

| Parameter 이름 | 설 명 | 값 |
|---|---|---|
| TIME_TO_KILL | 응용서버의 초기 개수를 초과하여 증가된 응용서버가 해당 시간(초) 동안 요청을 받지않는 경우 자동 종료시킨다 | 60<br>(int) |
| SESSION_TIMEOUT | transaction 처리를 위하여 요청을 받게되는 경우 transaction 을 시작하여 transaction 의 끝(commit/rollback) 까지 하나의 응용서버에서 계속 처리하게 되는데 이때 해당 시간(초) 동안 요청이 없는 경우 강제로 transaction 을 종료(rollback) 시킨다<br>-1 일 경우 강제 종료시키지 않고 무한정 기다린다. | 300<br>(int) |
| KEEP_CONNECTION | CAS와 클라인트간의 연결방식을 지정<br>ON은 클라이언트 접속단위로 연결, OFF일 경우 클라이언트의 트랜잭션 단위로 연결, AUTO일 경우 CAS수가 클라언트 수보다 많을 경우 ON과 같이 동작하고, 클라이언트가 많을 경우 OFF와 같이 동작한다. | ON/OFF/ AUTO |
| STATEMENT_POOLING | 아파치 DBCP에서 statement pooling사용에 따른 CAS지원 여부를 결정. ON설정 시 커밋 시점에 CAS의 모든 핸들을 재사용 | ON / OFF |

# 브로커 로그파일

CUBRID
More than open source!

- 접속로그 확인
  - broker 별로 각 용용서버가 요청을 받아 처리한 시간에 대한 기록이며, cubrid_broker.conf 의 ACCESS_LOG 에 명시된 디렉토리에
  - <broker 이름>.access 라는 이름으로 만들어진다.

```
1  192.168.100.201 - - 1158198049.151 1158198049.246 2008/09/14 10:40:49 ~ 2008/09/14 10:40:49 29438 - -1
2  192.168.100.201 - - 1158198049.401 1158198049.406 2008/09/14 10:40:49 ~ 2008/09/14 10:40:49 29438 - -1
```

| 값 | 설 명 |
|---|---|
| 1 | 응용서버 ID |
| 192.168.1.201 | 요청을 보낸 client 의 IP address |
| 1158198049.151 1158198049.246 | 요청을 받은 시간과 요청을 끝낸 시간에 대한 timestamp |
| 2008/09/14 10:40:49 ~ 2008/09/14 10:40:49 | 요청을 받은 시간과 요청을 끝낸 시간 |
| 29438 | 응용서버의 process ID |
| - -1 or ERR 1025 | --1 인 경우 에러 없이 처리<br>에러발생 시 ERR 하며 에러로그의 offset명시 |

# 브로커 로그파일 - 계속

- 에러 로그 확인
  - 응용 클라이언트의 요청을 처리하는 도중에 발생된 에러에 관한 정보를 broker_name_app_server_num.err의 파일에 기록

```
Time: 02/04/09 13:45:17.687 - SYNTAX ERROR *** ERROR CODE = -493, Tran = 1, EID = 38
Syntax: Unknown class "unknown_tbl". select * from unknown_tbl
```

| 값 | 설 명 |
|---|---|
| Time: 02/04/09 13:45:17.687 | 에러 발생 시점 |
| SYNTAX ERROR | 에러의 종류 |
| ERROR CODE = -493 | 에러 코드 |
| Tran = 1 | 트랜잭션 ID |
| EID = 38 | 에러 ID<br>SQL 문 처리 중 에러가 발생한 경우, 서버나 클라이언트 에러 로그와 관련이 있는 SQL 로그를 찾을 때 사용함 |
| Syntax:… | 에러 메세지 |

# 브로커 로그파일 - 계속

- SQL 로그
  - SQL 로그 파일은 응용 클라이언트가 요청하는 SQL을 기록하며, broker_name_app_server_num.sql.log라는 이름으로 저장된다.

```
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

# 브로커 로그파일 - 계속

CUBRID
More than open source!

- SQL 로그 분석
  - Broker_log_top
    - SQL log를 분석하여, 수행된 질의별 또는 트랜잭션별 수행 속도별 정렬
    - 사용법

```
% broker_log_top [-t] [-F MM/DD] [-T MM/DD] <log file> …

# 로그를 분석하여 트랜잭션 수행 속도별로 정렬
% broker_log_top -t broker_1.sql.log   → log_top.t 생성

# 로그를 분석하여 질의 수행 속도별로 정렬
% broker_log_top broker_1.sql.log  → log_top.res, log_top.q 생성
```

    - Log_top.t

```
broker_www_40.sql.log:458
02/09 21:48:44.139 (0) connect 127.0.0.1
02/09 21:48:44.139 (0) connect zbxe zbxe
02/09 21:48:44.139 (0) isolation level : 3, lock timeout : -1, auto_commit : false
02/09 21:48:44.140 (0) get_version
02/09 21:48:44.141 (82) prepare 0 select site_srl from xe_sites
02/09 21:48:44.141 (82) prepare srv_h_id 1
…
02/09 22:18:28.562 (0) SESSION_TIMEOUT
02/09 22:18:28.563 (0) disconnect
*** 1784.424
```

# 브로커 로그파일 - 계속

CUBRID
More than open source!

- log_top.res : 질의별 수행시간이 가장 오래 걸린 순서로 최대, 최소, 평균을 보여줌

```
                max         min         avg  cnt(err)
------------------------------------------------------------------------
[Q1]        2:44.360    1:10.360    1:44.360    12 (0)
[Q2]        1:01.174      33.174      42.174    33 (0)
[Q3]          56.926      32.926      41.926    21 (1)
[Q4]           6.027       4.027       5.027    21 (0)
```

- Log_top.q : 질의 목록

```
[Q1]----------------------------------------
broker_www_40.sql.log:495
02/09 21:48:44.185 (88) execute srv_h_id 1 update "xe_documents" as documents set
      "readed_count" = readed_count+1  where ("document_srl" = 32186)
02/09 22:13:28.545 (88) execute 0 tuple 1 time 1484.360

[Q2]----------------------------------------
broker_www_40.sql.log:876
02/09 22:23:29.570 (155) execute srv_h_id 1 update "xe_documents" as documents set
      "readed_count" = readed_count+1  where ("document_srl" = 32189)
02/09 22:33:30.744 (155) execute 0 tuple 1 time 601.174
```

장시간 교육에 수고 하셨습니다. 교육에 대한 여러분의 의견을 부탁 드립니다.
www.cubrid.com/customer_survey.php

# Thank you.

CUBRID
More than open source!