오픈 소스 데이터베이스 CUBRID 활용을 위한

데이터베이스 개발자 과정

기술본부

CUBRID2008 R3.1 v20110320

본 문서는 나눔 글꼴로 작성되었습니다.

CUBRID

목 차

CUBRID

1. CUBRID 소개

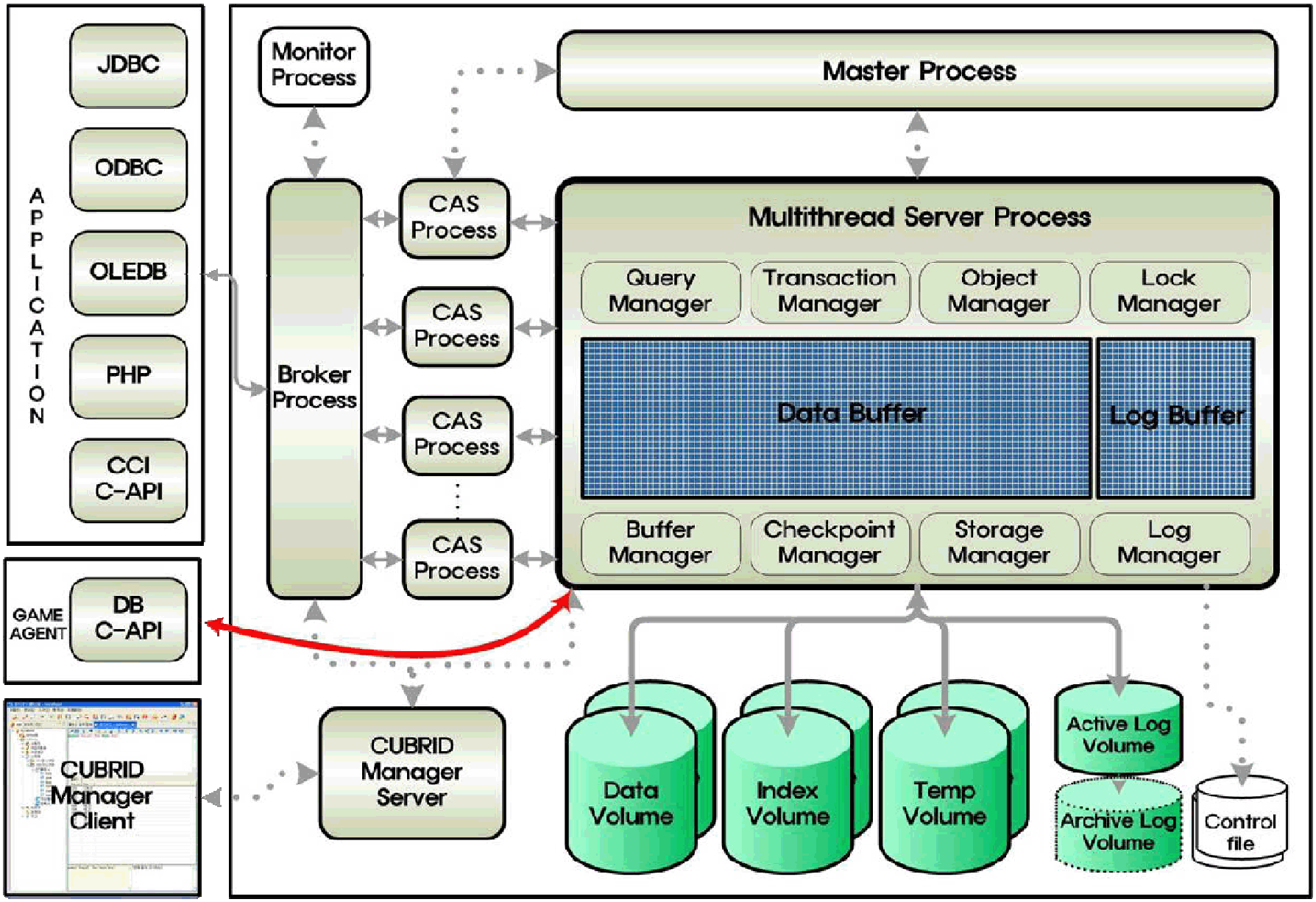
CUBRID

CUBRID 소개

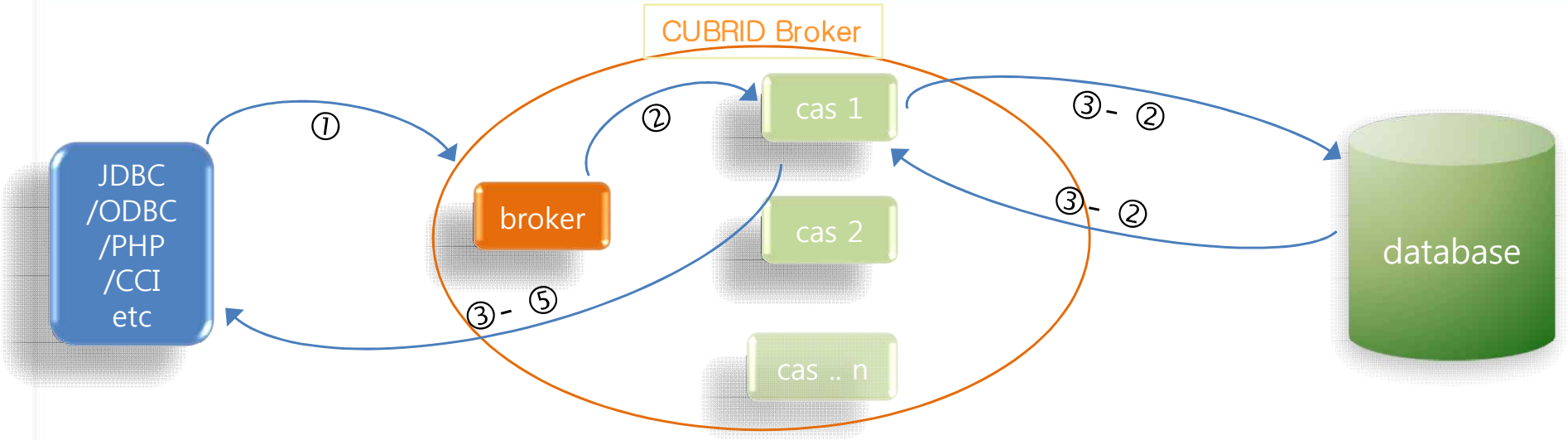
- 인터넷 서비스 최적의 DBMS 지향
- NHN에서 제품 개발을 하는 국내 유일의 오픈 소스 DBMS
  - 2008년 11월 오픈 소스 전환
- 서버(GPL), 인터페이스(BSD)의 유연한 오픈소스 라이선스
- SQL-92 표준 준수
- 트랜잭션 기반의 ACID 충족
- 온/오프라인 백업, 전체/증분 백업, 압축백업
- 시점 복구 지원
- HA

TM
open source

CUBRID 구조



CUBRID 구조



① 응용에서 질의처리를 위해 미들웨어에 작업 요청
  ① 실 작업처리는 cas 가 수행하며, 프로세스 단위 작업
  ② 응용에서 단위 cas에게 직접 요청은 어려우므로, 이들 관리위한 broker 를 두고 broker 에게 작업 요청
② broker 는 사용가능한 cas 에게 작업을 할당
③ cas는 데이터베이스에 연결하여 작업 수행후 결과를 응용에 전달
  ① 데이테베이스는 사용자가 지정하며, 지정한 데이터베이스에 cas 가 연결하는 형식
  ② cas 는 데이터베이스와 연결하여 작업후, 연결해제 요청시 연결해제하지 않고 연결 유지(connection pool)
  ③ 이후 동일한 데이터베이스와 작업 요청시 기존 연결 재사용
  ④ 다른 데이터베이스와 작업 요청시 현재 연결을 끊고, 새로이 연결
    ① 연결 overhead 발생 → broker 별로 데이터베이스를 지정(사용자)하여 사용 권장
  ⑤ 작업결과를 응용에 전달

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

2. 설치 및 기본 환경 구성

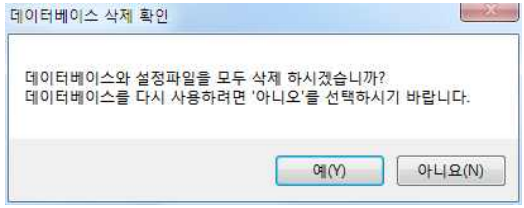
CUBRID

설치 환경

OS	CUBRID 2008 R3.1 Patch 1	용량	호환 OS	호환 CPU	일자
MS Windows 32bit	CUBRID-Windows-x86-8.3.1.1002.exe	61.25M	* Windows XP Home 32bit * Windows XP Pro * Windows 2003 * Windows Vista 32bit	* x86 * Intel EM64T * AMD64	11-03-08
MS Windows 64bit	CUBRID-Windows-x64-8.3.1.1002.exe	61.25M	* Windows Vista 64bit	* Intel EM64T * AMD64	11-03-08
Linux 32bit	CUBRID-8.3.1.1002-linux.i386.sh	83.77M	* CentOS 4이상 * Fedora 4 이상 * Ubuntu 6.10 이상 * openSUSE 11 이상 * Gentoo 2007 이상 * Asianux 2 이상 * Debian 4 이상	* x86 * Intel EM64T * AMD64	11-03-08
	CUBRID-8.3.1.1002-el5.i386.rpm	60.2M	* CentOS5 32bit	상동	11-03-08
Linux 64bit	CUBRID-8.3.1.1002-linux.x86_64.sh	86.63M	* CentOS 4이상 * Fedora 11 * Ubuntu 9.04	* Intel EM64T * AMD64	11-03-08
	CUBRID-8.3.1.1002-el5.x86_64.rpm	61.01M	* CentOS5 64bit	상동	11-03-08
Source RPM	CUBRID-8.3.1.1002-el5.src.rpm	83.93M	* CentOS 4이상 * Fedora 11 * Ubuntu 9.04	* Intel EM64T * AMD64	11-03-08
Source Code	CUBRID-8.3.1.1002.src.tar.gz	83.78M	-	-	11-03-08

설치

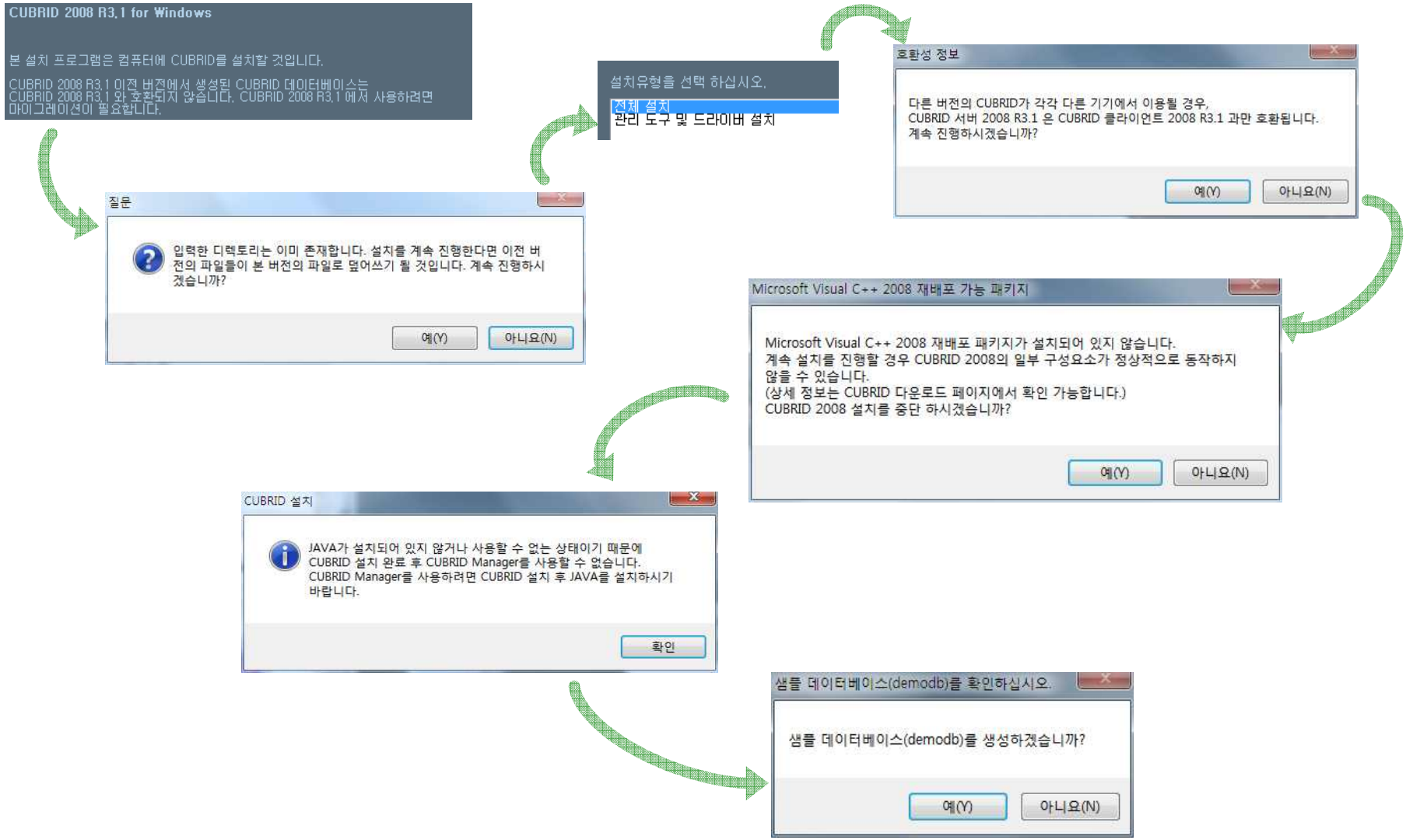
- Windows 버전
  - 업그레이드는 기존 제품 제거(uninstall)후 설치



    - 마이그레이션 도구(migrate_r30.exe) 수행 또는 데이터베이스 재구성
    - 예약어 증가에 따른 사전 점검 (check_reserved.sql)
  - Administrator 권한 설치 권장
  - Microsoft Visual C++ 2008 재배포 패키지 필요
    - ODBC/OLEDB 사용시에도 필요
  - CUBRID Manager client 사용 위하여 JAVA 필요
    - JRE 1.5 이상, 최신 버전 권장

설치 [계속]

- Windows 버전 설치



기본 환경 구성

- 방화벽 설정 (LINUX, Windows 공통)
  - CUBRID Manager : 8001, 8002
    - 질의편집기 : 30000
  - 응용개발 : 33000
- Windows 추가 설정 (기본 설정시)
  - 질의편집기 : 30001 ~ 30040
  - 응용개발 : 33001 ~ 33040
- SELINUX 설정 (/usr/CUBRID 아래 설치 가정)

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

응용 사용 환경

- JDBC
  - class path

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

  - 연결 설정

```
// 서버 IP : 127.0.0.1
// 서버 port : 33000
// 데이터베이스 이름 : demodb

// 일반적인 연결 설정
String url = "jdbc:cubrid:127.0.0.1:33000:demodb:::";

// 사용할 문자셑(UTF-8) 지정 (데이터베이스 입출력시 형변환 아님)
String url = "jdbc:cubrid:127.0.0.1:33000:demodb:::?charset=utf-8";

Class.forName("cubrid.jdbc.driver.CUBRIDDriver");
// 데이터베이스 사용자 이름 : dba
// 데이터베이스 사용자 암호 : dba_pw
Connection conn = DriverManager.getConnection(url, "dba", “dba_pw");
```

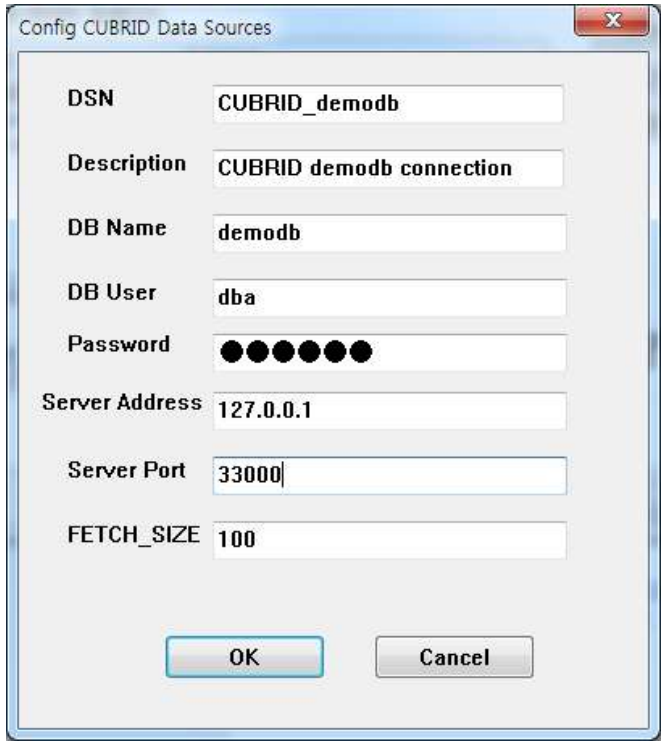
응용 사용 환경 [계속]

- PHP
  - sourceforge.net/projects/cubridinterface
    - CUBRID-PHP5-5.3-WIN32-VC9-TS-8.3.1.0005.bin.zip


```
* php.ini
extension=cubrid_php.dll
```

응용 사용 환경 [계속]

- ODBC

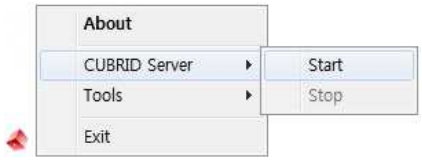



```
sConn =
"driver={CUBRID Driver};server=127.0.0.1;port=33000;uid=dba;pwd=dba_pw;db_name=demodb;
CHARSET=utf-8"
```

3. 기본 사용 방법

CUBRID 서비스

- CUBRID 사용을 위한 기본 프로세스
- 구동
  - windows 의 경우 부팅시 자동 구동 또는 tray 메뉴 사용

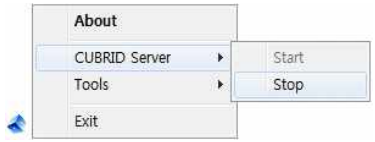


  - 명령어 사용

```
% cubrid service start
@ cubrid master start
** cubrid master start : success
@ cubrid server start

CUBRID 2008 R3.1

** cubrid server start : success
@ cubrid broker start
** cubrid broker start : success
@ cubrid manager server start
** cubrid manager server start : success
```

CUBRID 서비스 [계속]

- 종료
  - windows 의 경우 CUBRID tray 메뉴 사용



  - 명령어 사용

```
% cubrid service stop
@cubrid server stop

Server demodb notified of shutdown.
This may take several monites. Please wait.
** cubrid server stop : success
@cubrid broker stop
** cubrid broker stop : success
@cubrid manager server stop
** cubrid manager server stop : success
@cubrid master stop
** cubrid master stop : success
```

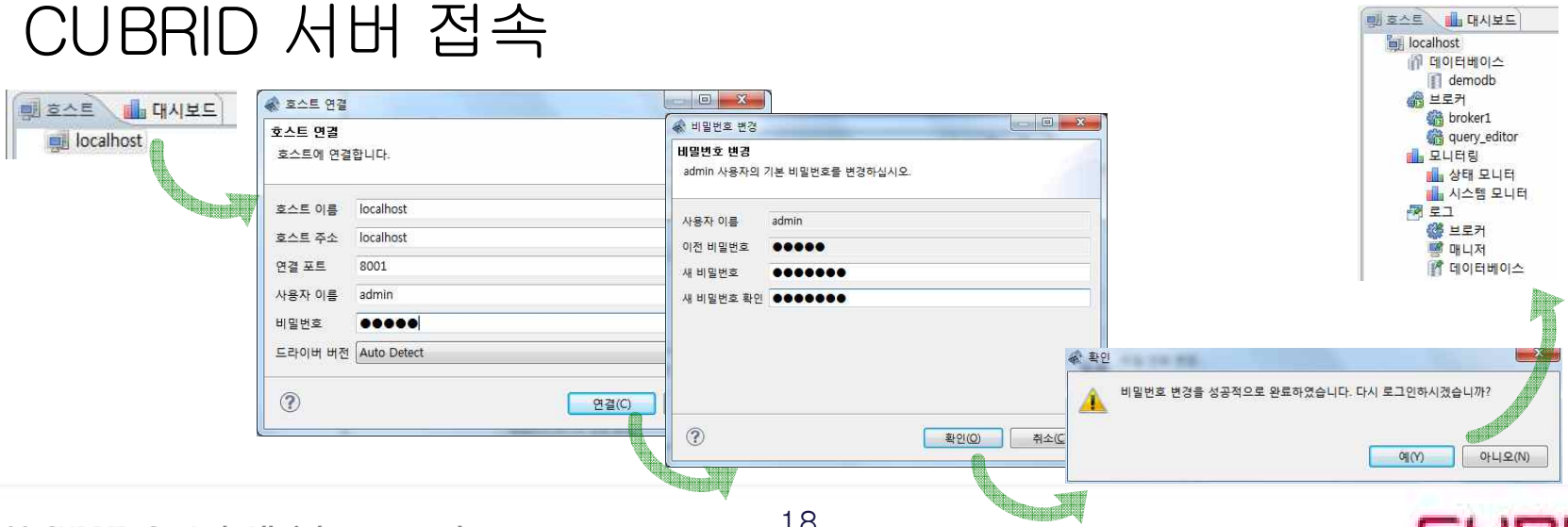
CUBRID Manager Client

- 실행
  - windows

  - LINUX

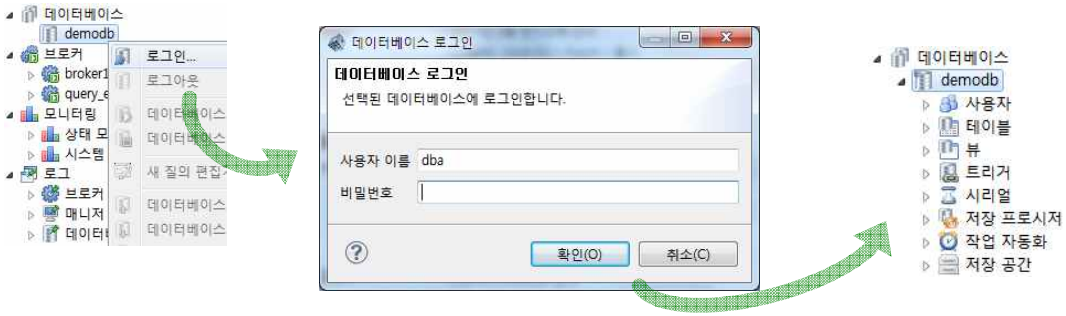
```
% cd $CUBRID/cubridmanager/cmclient
% cubridmanager
```

- CUBRID 서버 접속



CUBRID Manager Client [계속]

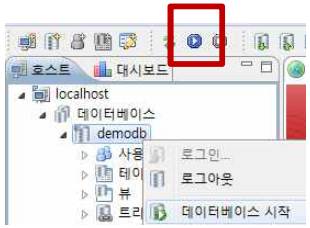
- 데이터베이스 로그인
  - 작업을 원하는 데이터베이스별 데이터베이스 사용자로 로그인
  - 로그인후 사용자 권한에 맞는 작업 가능
    - 서버 구동, 종료, 백업 등 관리 작업 → dba 권한 필요



- 새로고침
  - 명령어, 질의편집기 등의 작업 내용

데이터베이스 서버

- 구동
  - CUBRID manager 사용

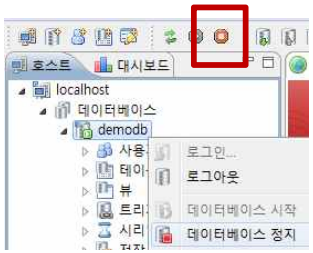


  - 명령어 사용

```
% cubrid server start demodb
** cubrid server start : success
```

데이터베이스 서버 [계속]

- 종료
  - CUBRID manager 사용

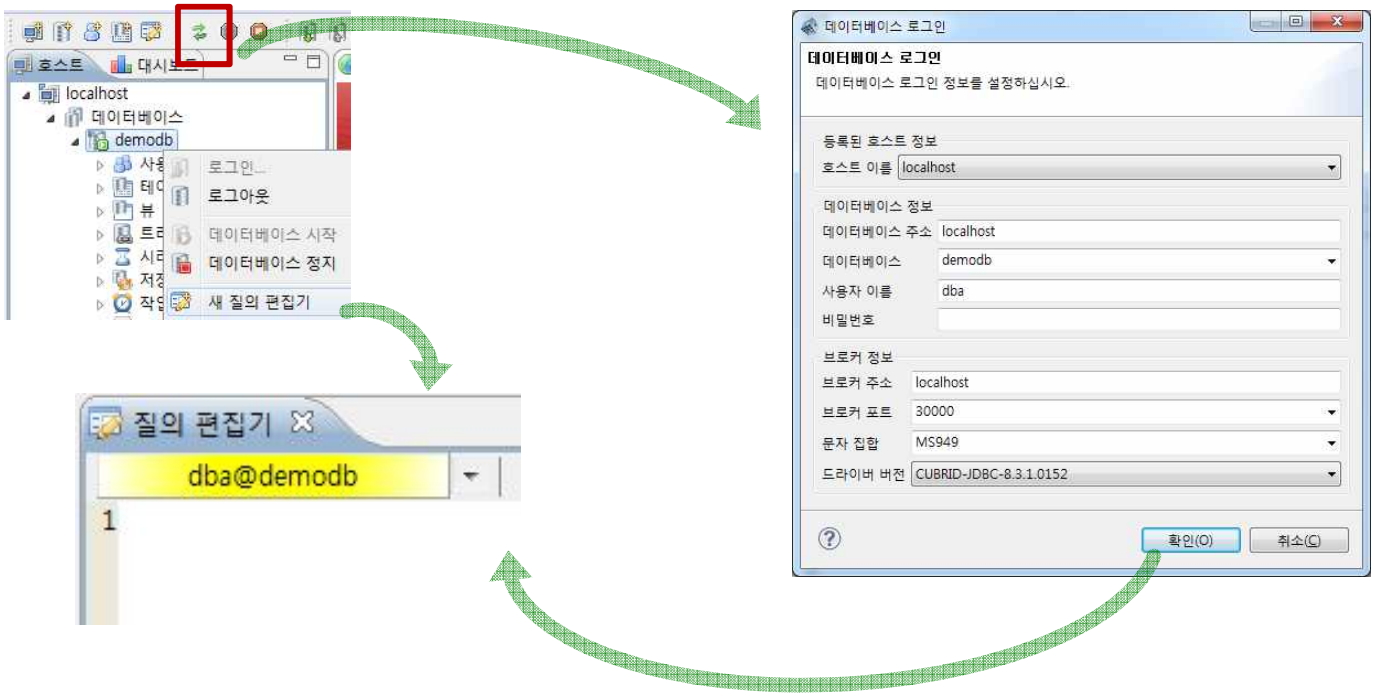


  - 명령어 사용

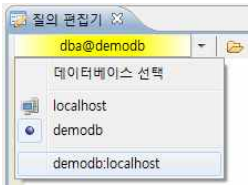
```
% cubrid server stop demodb
@ cubrid server stop : demodb
** cubrid server stop : success
```

질의편집기

- 데이터베이스 서버 구동후 사용 가능



- 데이터베이스 및 데이터베이스 사용자 선택

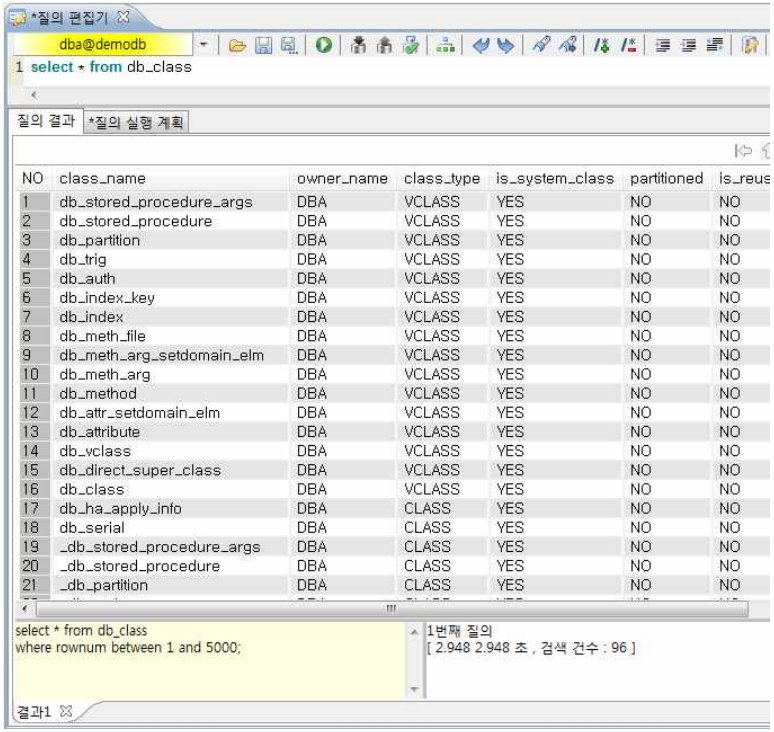


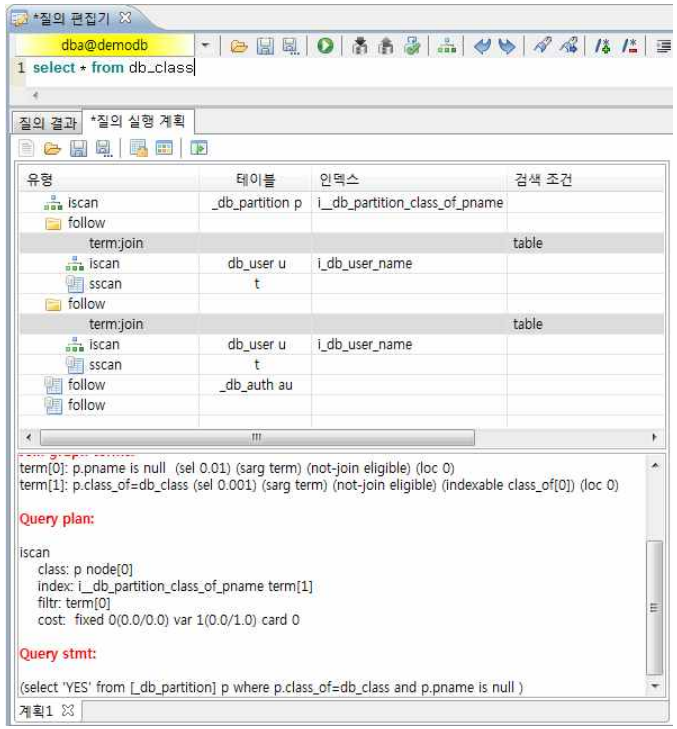
질의편집기 [계속]

- 주요 명령

질의 수행 : F5		질의수행계획보기 : F6	
편집 취소 : ctrl-Z		편집 복구 : ctrl-Y	
자동 커밋 ON		자동 커밋 OFF	
커밋		롤백	

- 질의 수행





CUBRID

질의편집기 [계속]

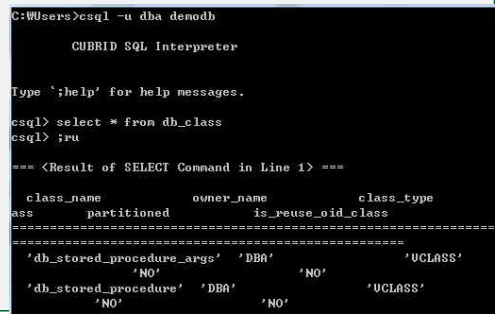
- CUBRID SQL Interpreter
  - 실행
    - csql [-u <데이터베이스 사용자>] [-S] <데이터베이스 이름>
      - 데이터베이스 사용자를 입력하지 않으면 PUBLIC 사용자로 로그인
      - -S : 서버가 구동되지 있지 않은 경우 사용
    - 입력한 SQL 문은 버퍼에 저장되며, 버퍼의 내용을 실행
  - 주요 명령어

;RUn : 버퍼의 내용을 수행 ;Xrun : 버퍼의 내용을 수행 후 버퍼의 내용 지움

;EDit : 버퍼의 내용을 편집 (windows : notepad, Linux : vi)
;List : 버퍼의 내용 보기 ;CLean : 버퍼의 내용 지우기

;SChema <table이름> : 테이블 정보 표시
;AUto_commit ON/OFF : 자동 커밋 수행/취소
;COmmit : 커밋 수행 ;ROllback : 롤백 수행

;EXit : 종료



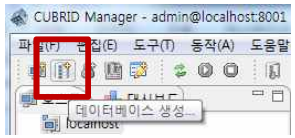
4. 데이터베이스 생성

고려사항

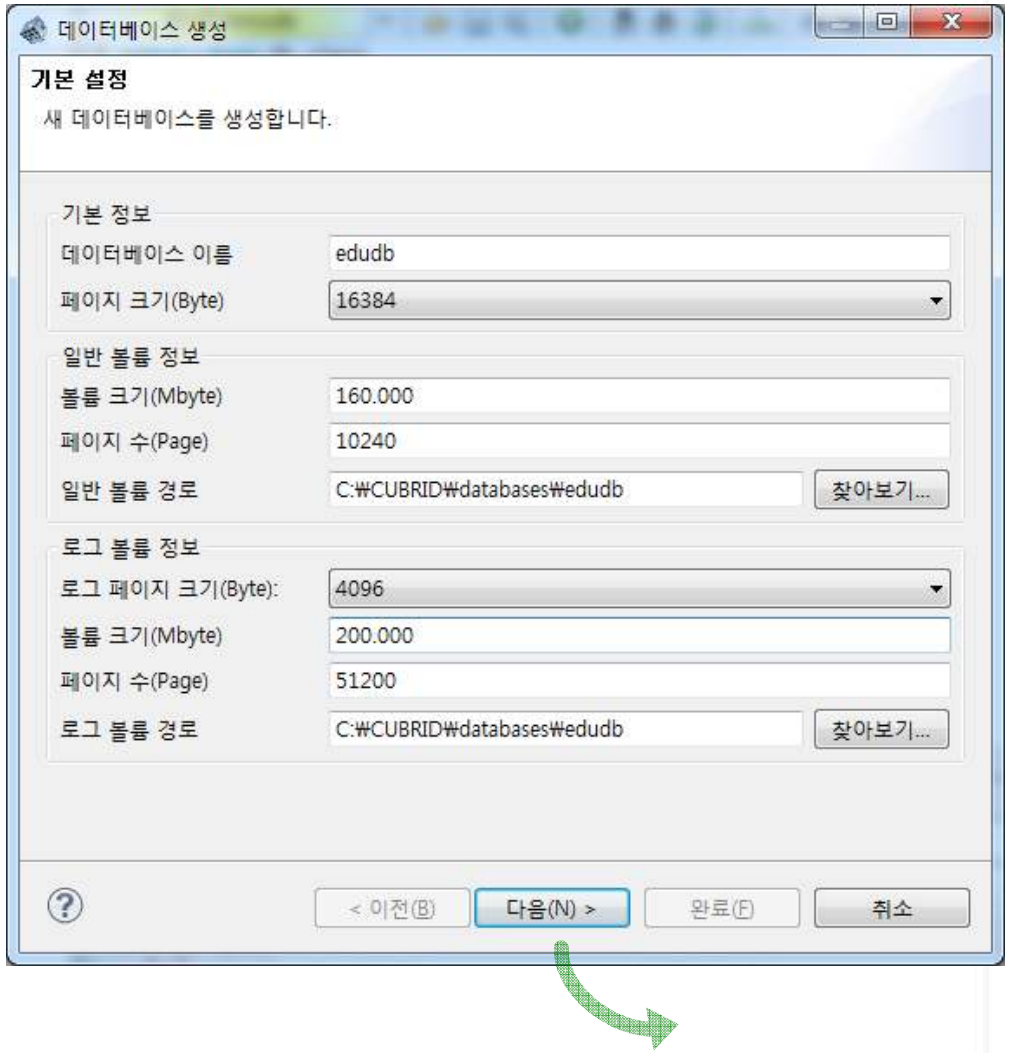
- 데이터베이스 이름
  - 최대 128자, 대소문자 구분
- 데이터베이스 크기
  - 데이터 볼륨 : 레코드 길이, 데이터 보전 연한
  - 인덱스 볼륨 : 데이터의 20~30% 수준
  - 질의 처리 공간(템프 볼륨) : 데이터의 20~30% 수준
  - 로그 : 일반적으로 200M
  - 실 데이터베이스 볼륨 크기 : 계산된 크기의 약 2배(로그 제외)
- 데이터베이스 위치
  - 디스크 I/O bottle-neck 최소화
    - 데이터와 로그 분리 등

데이터베이스 생성

- 데이터베이스 생성 마법사 실행

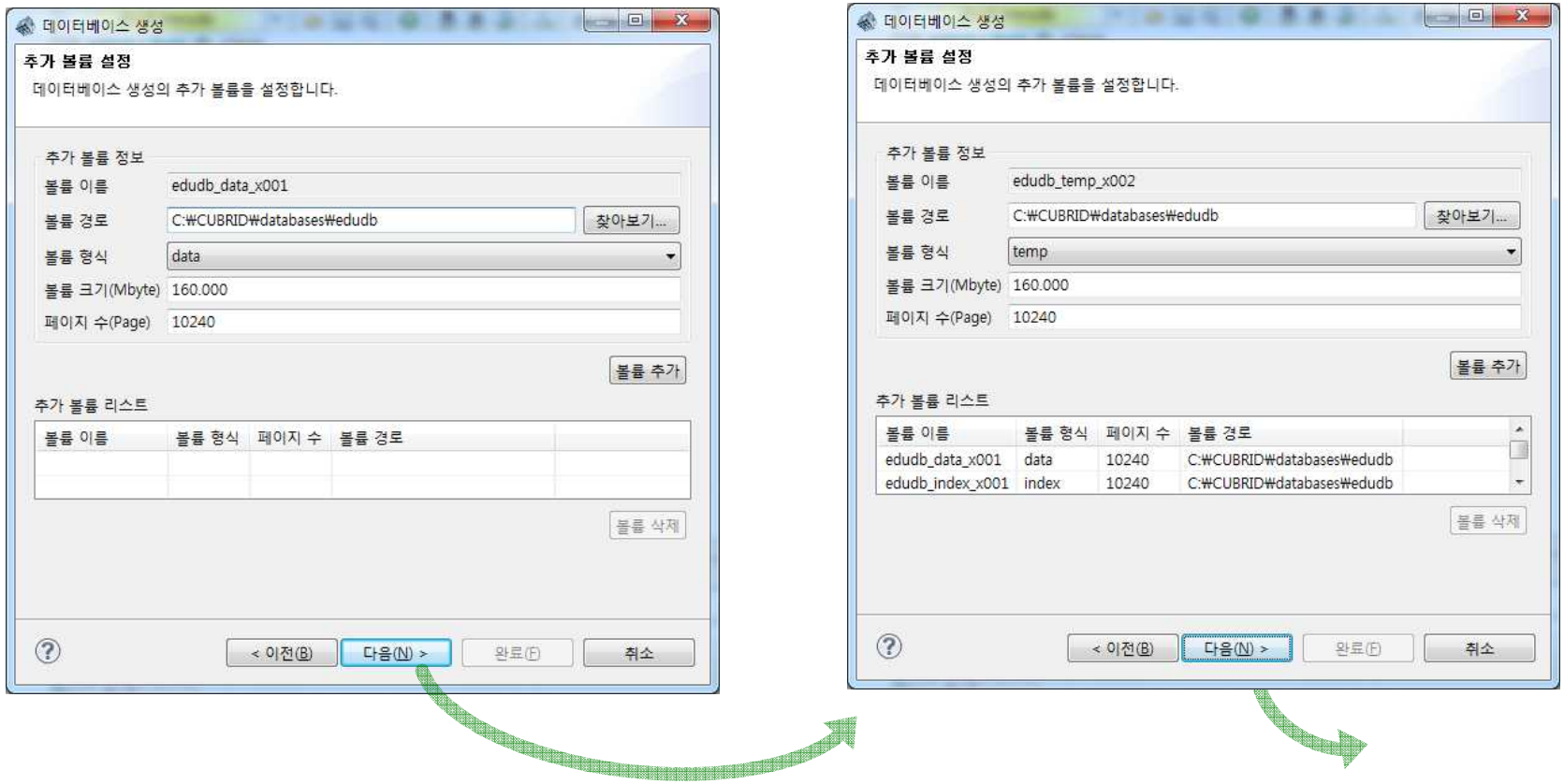


- 데이터베이스 이름 : eudb
- 페이지 크기 : 16384 (성능상 무난함)
- 일반볼륨 정보
  - 볼륨크기 : 160M (기본설정)
  - 볼륨경로 : 생성 위치 지정
- 로그볼륨 정보
  - 볼륨크기 : 200M (성능상 무난함)
  - 볼륨경로 : 로그 위치 지정



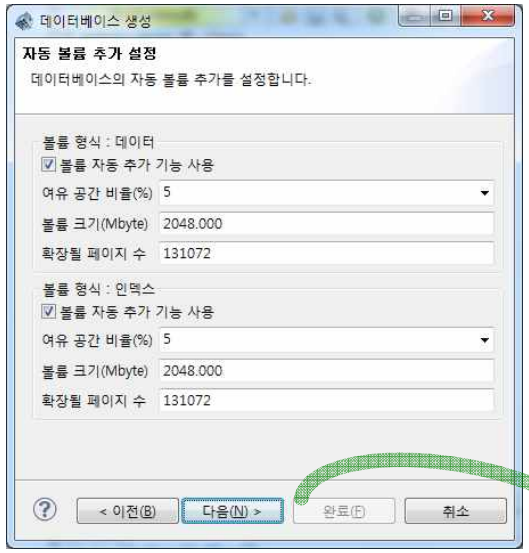
데이터베이스 생성 [계속]

- 볼륨 추가
  - 데이터, 인덱스, 질의처리 볼륨 등 용도별 추가

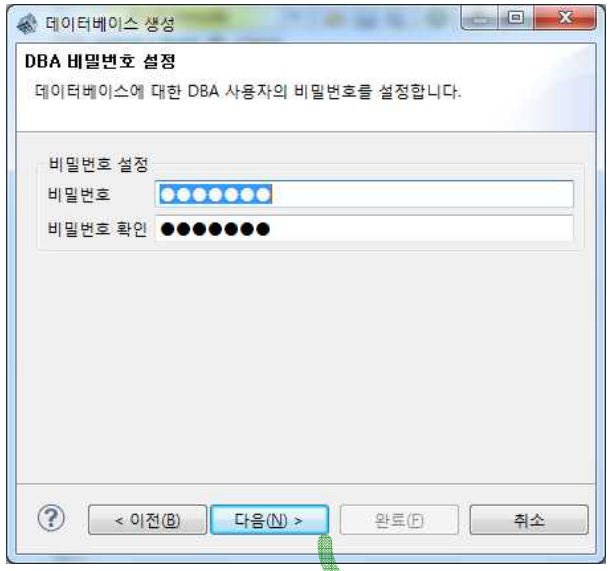


데이터베이스 생성 [계속]

- 볼륨 자동 추가 설정



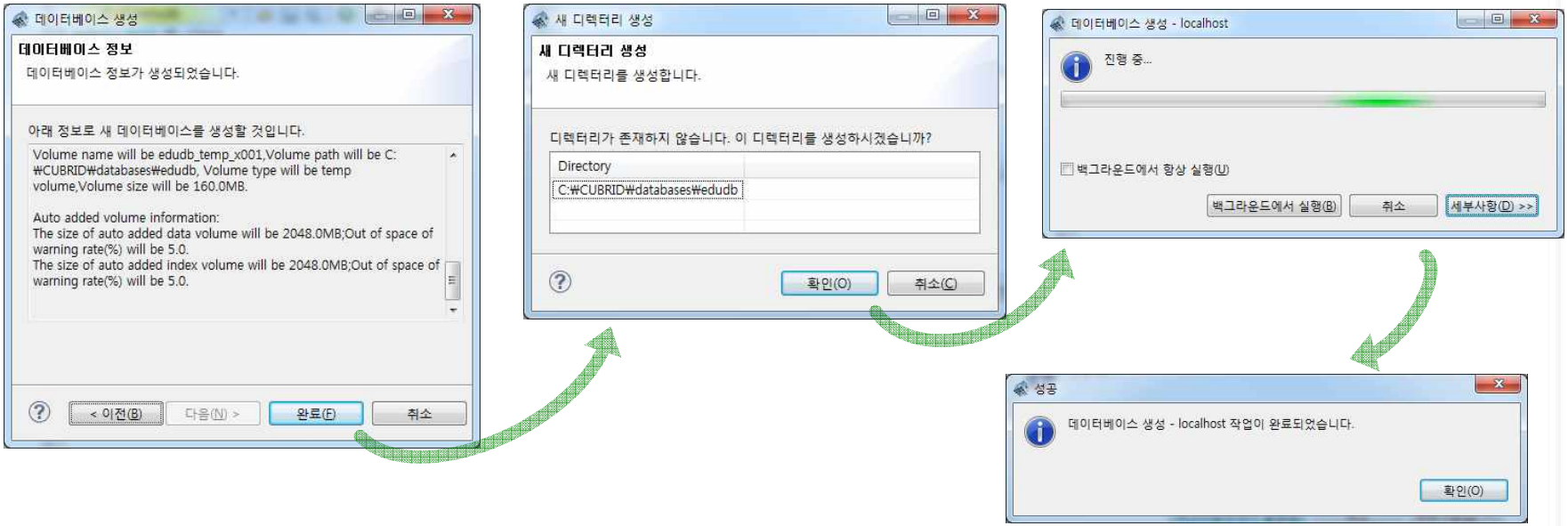
- dba 암호 설정



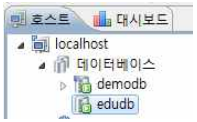
CUBRID

데이터베이스 생성 [계속]

- 내용확인 및 생성



- 생성 확인
  - 생성 완료 후 데이터베이스 서버 자동 구동



5. 스키마 생성 및 조작

고려 사항

- 테이블
  - 개수 무제한
  - 이름에 한글, 영문자, 숫자, _, #, % 사용 가능, 첫글자는 문자(한글, 영문)여야 함.
  - 이름의 최대 길이는 255자
- 컬럼
  - 테이블 당 최대 6,400 개 생성 가능
  - 이름에 한글, 영문자, 숫자, _, #, % 사용 가능, 첫글자는 문자(한글, 영문)여야 함.
  - 이름의 최대 길이는 255자
- index
  - 테이블 당 최대 6,400 개 생성 가능
  - 이름지정 가능
- 제약조건
  - NULL: NULL 값을 허용
  - NOT NULL : NULL 값을 허용하지 않음
  - unique : 중복된 값 허용하지 않음, default 와 같이 선언 불가
  - primary key
  - foreign key

고려 사항 [계속]

- 예약어 사용시 [] 나 “” 로 감싸주어야 함.
  - 예) select [serial] from “test” where …
- db_root
  - 하나의 레코드를 가지는 일종의 가상 테이블
  - 시스템 값(sys_date 등) 을 얻기위해 사용
  - dual
- 스키마 한글 사용
  - 기본 설정은 불가, 한글처리로 인한 성능
  - intl_mbs_support=yes(cubrid.conf) 설정후 사용가능

데이터 타입

- 숫자형
  - smallint: 2bytes, -32768 ~ 32767
  - integer: 4bytes, -2147483648 ~ 2147483647
  - bigint: 8bytes, -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
  - numeric: numeric(10), numeric(10,2), numeric(38)
  - float:-10e38 ~ 10e38, 유효자리수 초과하는 수치 정밀도 보장 불가
  - double:-10e308 ~ 10e308, float 와 마찬가지로 유효자리수 문제
- 문자형
  - char:고정길이형, 다른 컬럼과 같이 저장, 길이가 일정하거나 인덱스로 사용될 때, rtrim()
  - varchar:가변길이형, 최대1G(string), 다른 컬럼과 다른곳에 저장, 데이터 길이와 저장된 곳에 대한 포인터 저장에 대한 overhead, update시 길이 변하는 경우 overhead

데이터 타입 [계속]

- 날자/시간형
  - '' 값 입력시 에러, NULL 또는 표현가능한 값 입력 허용
  - date:날자만 저장, 기본형 mm/dd/yyyy, to_char() 사용시 인덱스 사용 불가
  - time:시간만 저장, 기본형 hh:mi:ss
  - datetime: 1/100초, 'mm/dd[/yyyy] hh:mi[:ss.ff]' 또는 '[yyyy-]mm-dd hh:mi[:ss.ff]'
  - timestamp:1970/1/1 09:00:00 이후 경과한 초, 최대 2038/1/19 03:14:07
- LOB
  - BLOB : binary large data 저장
  - CLOB : character large data 저장

데이터 타입 [계속]

- '' ≠ NULL
  - str = '', str <> '', str is NULL, str is not NULL, str = NULL (wrong result)
- 자동 형변환 제한
  - 문자열 → 숫자, 숫자 → 문자열 지원 않함
    - where '123' = 123 : 타입 불일치 에러

제약조건

- 데이터 무결성 유지를 위한 조건
  - not null, default, unique
  - primary key
    - 테이블당 1개, null 값을 가질 수 없음
  - foreign key
    - 참조대상이 null 이거나 기본키로 존재하여야 함
    - 옵션
      - on update : no action, restrict, set null
      - on delete : no action, restrict, set null, cascade

테이블 생성

- 일반

```
create table comp_tbl (
comp_id int,
comp_name char(50) not null unique,
constraint pk_Id primary key(comp_id)
)

create table user_tbl (
user_id int,
user_name char(10) not null unique,
addr varchar(50) null,
age int not null default 0,
comp_id int,
constraint pk_id primary key(user_id),
constraint fk_comp_id foreign key (comp_id) references comp_tbl(comp_id) on delete cascade
)
```

- 스키마 복제
  - 자동 증가 속성이 없는 경우만 사용 가능
  - 인덱스 이름 동일하게 복사되므로 주의

```
create table tmp_user_tbl like user_tbl
```

테이블 생성 [계속]

- 스키마 , 데이터 복제
  - 타입이 다른 경우 자동 형변환
  - select 절에 없는 필드는 null
  - 중복되는 값이 있는 경우 replace 옵션 사용

```
create table tbl ( id int )

create table tbl2 ( id int ) as select id from tbl

create table tbl3 ( id float) as select id from tbl

create table tbl4 ( id int , name char(10) ) as select id from tbl

create table tbl5 ( id int unique) replace as select id from tbl
```

자동 증가

- 트랜잭션 영향 받지 않음
- serial
  - 일련 번호 생성
  - 현재값 얻기 : [serial name].current_value, .currval
  - 다음값 얻기 : [serial name].next_value, .nextval

```
create serial seq_no [start with 2] [increment by 2] [min value 2] [max value 200] [cycle/nocycle]
select seq_no.current_value from db_root
insert into user_tbl (user_id, user_name) values(seq_no.next_value, 'user_name')
```

- auto increment
  - 레코드 입력시 자동 입력되는 필드 지정 (기존 max + [증가값])
  - 사용자가 임의의 값 입력 가능 ← 자동 증가 인식되지 않음

```
create table my_table (
   id int auto_increment  [ (10, 2) ],
   …
```

인덱스

- 필드별 오름차순, 내림차순 지정 가능
- 필드 길이 지정 가능
- unique, primary key, foreign key 인덱스로 관리

```
create index on tbl4(id, name)              // i_tbl_id_name
create index idx on tbl4(id)
create index idx1 on tbl4(id, name desc)


create unique index on tbl4(name)       // u_tbl_name
create unique index uidx on tbl4(id)

create reverse index r_idx on tbl4(id, name)

create index idx1 on tbl4(name(3))

drop index on tbl4(id, name)                // i_tbl_id_name
drop index idx1  [ on tbl4(name) ]

alter index  idx1  [ on tbl4(name) ] rebuild
```

테이블 변경

- 테이블

```
rename table tbl4 as tbl40
drop table tbl40
```

- 컬럼
  - ■ 특정 위치에 컬럼 삽입

```
alter table tmp_user_tbl add column nick_name char(10)
alter table tmp_user_tbl add column nick_name2 char(10) first
alter table tmp_user_tbl add column nick_name3 char(10) after user_name

alter table tmp_user_tbl alter column nick_name set default 'nick'
alter table tmp_user_tbl change nick_name default ''
alter table tmp_user_tbl rename column nick_name2 as nickname
alter table tmp_user_tbl drop column nickname
```

테이블 변경 [계속]

- 제약조건

```
alter table tmp_user_tbl add constraint unique u_idx(nick_name)
alter table tbl add constraint pk1 primary key(id)
alter table tbl5 add constraint fk1 foreign key(id) references comp_tbl(comp_id)

alter table tbl drop primary key
alter table tbl5 drop foreign key fk1
```

- 인덱스

```
alter table tbl3 add index idx1 (id desc)
alter table tbl3 drop index idx1
```

카달로그

- 스키마 정보를 저장한 시스템 테이블
- SQL 을 통해 스키마 정보 획득
  - 테이블 정보 : db_class
    - 테이블명: class_name, 소유자: owner_name
  - 컬럼 정보 : db_attribute
    - 테이블명: class_name, 컬럼명 : attr_name, 컬럼타입: attr_type

6. 데이터 변경

데이터 입력 및 수정

- insert

```
insert into tbl values(…)
insert into tbl(…) values(…, default)
insert tbl set col1 = val [ , … ]
insert into tbl select …
```

  - ■ unique/PK 위배시 기존값 갱신

```
insert into comp_tbl values(1, 'comp1')
   on duplicate key update comp_name = 'comp_1' [ , … ]

replace into comp_tbl values(1, 'comp1') [ , ( … ) ]
replace comp_tbl set comp_name = 'comp_1' [ , … ]
```

- update

```
update tbl set col0 = val where col1 = val
update tbl set col0 = val where col1 = val limit 1

update tbl set (col0, col1) = (select new_col0, new_col1 where …) where …
```

데이터 삭제

- delete

```
delete from tbl where col1 = val
delete from tbl where col1 = val limit 1
```

- truncate
  - 모든 데이터 삭제
  - on delete trigger 비활성
  - auto increment 초기값부터 생성

```
truncate tbl
truncate table tbl
```

7. 데이터 검색

Group by

- 검색 결과를 특정 필드(들)를 기준으로 그룹화
  - group by 절에 모든 필드 명시하지 않아도 가능
  - 그룹화시 정렬 방법 제공 : ASC/DESC
  - 중간 집계 방법 제공 : WITH ROLLUP
  - 정렬없이 그룹화 방법 제공 : ORDER BY NULL

```
/*
insert into tbl24(id,name) values(1,'name1');
insert into tbl24(id,name) values(1,'name11');
insert into tbl24(id,name) values(2,'name22');
insert into tbl24(id,name) values(2,'name2');
*/

select id,name,count(*) from tbl24 group by id,name

select id,name,count(*) from tbl24 group by id having count(*) > 1

select id,name, sum(id) from tbl24 group by id with ROLLUP

select id,name, sum(id) from tbl24 group by id order by NULL
```

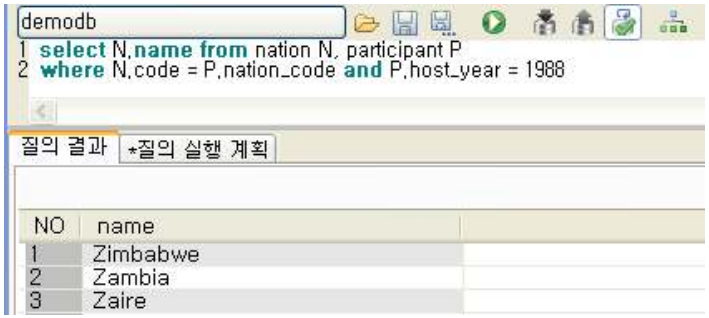
id	name	count(*)
1	name1	1
1	name11	1
2	name2	1
2	name22	1

id	name	count(*)
1	name11	2
2	name2	2

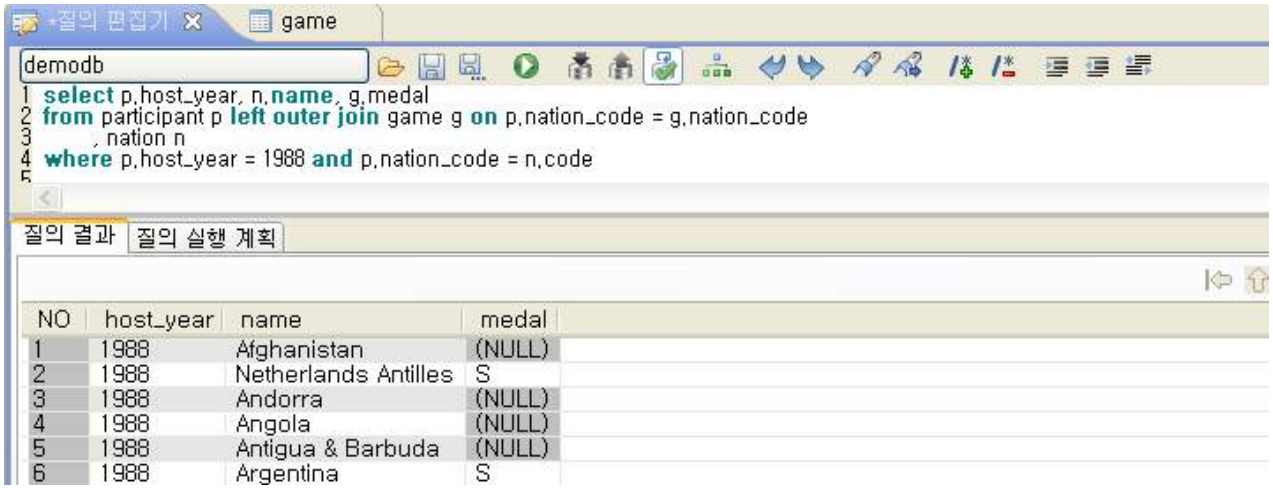
id	name	sum(id)
1	name11	2
2	name2	4
(NULL)	(NULL)	6

조인

- inner join



- outer join



sub query

- 질의 내에 부질의 사용

```
select (select name from athlete where code = athlete_code), medal, score from record where
host_year = 2004

select distinct event_code, (select sports from event where code = event_code)
from record where host_year  = 2004

select (select name from event where code = event_code), medal, score, unit
from record
where host_year = 1988
  and (event_code, athlete_code) in
            (select event_code, athlete_code from game
              where host_year = 1988 and nation_code = 'KOR')
```

limit

- 질의 결과 개수 제한
  - limit 레코드순번 [, 레코드개수]

```
select (select name from athlete where code = athlete_code) athlete, medal, score from record
where host_year = 2004 limit 5

select (select name from athlete where code = athlete_code) athlete, medal, score from record
where host_year = 2004 order by athlete limit 4, 3

select (select name from athlete where code = athlete_code) athlete, medal, score from record
where host_year = 2004 and rownum between 5 and 7

select (select name from athlete where code = athlete_code) athlete, medal, score from record
where host_year = 2004 order by athlete for orderby_num() between 5 and 7
```

Order by

- 질의 결과 정렬
  - select 절에 없이 사용 가능
  - 인덱스와 병행 사용시 성능 개선
    - 인덱스 검색을 통한 정렬 회피 효과

```
select host_nation from olympic order by host_year

// pk (host_year) 를 이용하여 정렬 회피, 인덱스 사용위한 조건 추가
select host_nation from olympic where host_year > 0 order by host_year

// 역순 정렬된 인덱스가 없어 full scan 후 정렬
select host_nation from olympic where host_year > 0 order by host_year desc

create index r_year on olympic(host_year desc)
```

질의 힌트

- 인덱스 지정
  - 지정한 인덱스와 full scan 중 평가
  - 비용 기반의 인덱스 선택
    - 비용 강제 0 설정 → (+) 지정
  - 인덱스 명 중복시 <테이블명>.<인덱스명>
  - join 시 필요한 인덱스 모두 지정

```
create index idx1 on athlete(name)
create index idx1 on record(athlete_code)

select * from athlete where name = 'Lee Eun-Chul' using index none
select * from athlete where name = 'Lee Eun-Chul' using index idx1
select * from athlete where name = 'Lee Eun-Chul' using index idx1(+)
select * from athlete using index idx1(+) // index 사용을 위한 조건이 없어 idx1 index 를 사용하지 못함.
where name > ''

select host_year, medal from athlete a, record b where a.name = 'Lee Eun-Chul' and a.code =
b.athlete_code using index a.idx1, b.idx1
```

질의 힌트

- 인덱스 지정
  - 지정한 인덱스와 full scan 중 평가
  - 비용 기반의 인덱스 선택
    - 비용 강제 0 설정 → (+) 지정
  - 인덱스 명 중복시 <테이블명>.<인덱스명>
  - join 시 필요한 인덱스 모두 지정

```
create index idx1 on athlete(name)
create index idx1 on record(athlete_code)

select * from athlete where name = 'Lee Eun-Chul' using index none
select * from athlete where name = 'Lee Eun-Chul' using index idx1
select * from athlete where name = 'Lee Eun-Chul' using index idx1(+)
select * from athlete using index idx1(+) // index 사용을 위한 조건이 없어 idx1 index 를 사용하지 못함.
where name > ''

select host_year, medal from athlete a, record b where a.name = 'Lee Eun-Chul' and a.code =
b.athlete_code using index a.idx1, b.idx1
```

조인 힌트

- 테이블간 조인 방식 지정
  - select /*+ <조인방식> */ …
    - USE_NL : netsted loop join
    - USE_MERGE : sort merge join
    - USE_IDX : join 시 index 사용
  - 조인 방식 한정
    - USE_NL(a, b) : a, b 테이블이 조인되는 경우에만
  - 조인 순서 지정
    - ORDERED : from 절에 명시된 순서대로 join

click counter

- 게시물 조회시 조회건수 자동 증가
  - select 후 update 형태의 질의 개선
  - 조회건수용 컬럼 값 자동 증가
    - select incr(read_cnt) … from …
      - read_cnt 의 현재값을 넘겨주고, 그 값을 1만큼 증가시킴
  - update 질의 제거를 통하여 트랜잭션 간소화 및 성능 개선

8. 연산자와 함수

연산자

- 논리연산자

AND, &&	OR, ||	XOR	NOT, !

- 비교연산자

=	<=>	!=, <>	<	>	<=	>=	is [not] <true | false | null>

- 산술연산자

+	-	*	/, DIV	*, MOD

  - 숫자 형이 다를 경우 : 범위가 큰 숫자형으로 변환
  - 날자형의 경우
    - time - time = int, time ±int = time
    - date - <date \| timestamp \| datetime > = int, date ±int = date
    - timestamp - <date \| timestamp \| datetime > = int, timestamp ±int = timestamp
    - datetime ±<time \| date \| timestamp \| datetime > = <time \| date \| timestamp \| datetime >, datetime ±*/ int = datetime

연산자

- 집합연산자

UNION	DIFFERENCE	INTERSECTION

- 비트연산자

&	|	^	~	<<	>>

BIT_AND	BIT_OR	BOT_XOR	BIT_COUNT

- 문자열 연산자

+, ||	like

- 기타

between	exists	in

함수

- 문자열 함수

bit_length	bit_length('CUBRID') → 48 bit_length('큐브리드') → 64
length	length('CUBRID') → 6 length('큐브리드') → 8
chr	chr(65) → 'A'
concat	concat('CUBRID', '2008', 'R') → CUBRID2008R
concat_ws	concat_ws('/', '', 'usr', 'CUBRID') → '/usr/CUBRID' length('큐브리드') → 8
field	field('2008', '20', '2009', '2008") → 3 field('2008', '20', '2009', '2010") → 0
instr	instr('CUBRID2008 R3.1', 'R') → 4 instr('CUBRID2008 R3.1', 'R', 5) → 12 instr('CUBRID2008 R3.1', '0', -1) → 9
position	position('R' in 'CUBRID2008 R3.1') → 4

함수 [계속]

- 문자열 함수

lower	lower('CubRid') → 'cubrid'
lcase	lcase('CubRid') → 'cubrid'
upper	upper('CubRid') → 'CUBRID'
ucase	ucase('CubRid') → 'CUBRID'

substring	substring('abcde' from 3) → 'cde' substring('abcde' from 3 for 1) → 'c' substring('abcde' from -3 for 2) → 'ab'
substr	substr('abcde', 3) → 'cde' substr('abcde', 3, 1) → 'c' substr('abcde', -3, 2) → 'cd'
left	left('abcde', 3) → 'abc' left('abcde', -3) → 'abcde'
right	right('abcde', 3) → 'cde' right('abcde', -3) → 'abcde'
mid	mid('abcde', 3, 1) → 'c' mid('abcde', 3, 5) → 'cde'

함수 [계속]

- 문자열 함수

lpad	lpad('CUBRID', 3, '-') → 'CUBRID' lpad('CUBRID', 8) → ' CUBRID'
rpad	rpad('CUBRID', 8, '0') → 'CUBRID00'
ltrim	ltrim(' CUBRID ') → 'CUBRID ' ltrim('aabCUBRID', 'ab') → 'CUBRID'
rtrim	rtrim(' CUBRID ') → ' CUBRID'
trim	trim(' CUBRID ') → 'CUBRID'
reverse	reverse('CUBRID') → 'DIRBUC'
replace	replace('CUBRID', 'RID') → ' CUB' replace('CUBRID, 'RID', '-') → 'CUB-'
translate	translate('CUBRID', 'RD', '') → ' CUBI' translate('CUBRID, 'RD', '-') → 'CUB-I'

함수 [계속]

- 문자열 함수

strcmp	strcmp('CUBRID', 'cubrid') → 0 strcmp('CUBRID', 'CUBRID') → 0 strcmp('a', 'b') → -1 strcmp('a0', 'a') → 1

함수 [계속]

- 산술 함수

sin, cos, cot, tan asin, acos, atan, atans	삼각함수
ln, log2, log10	로그함수
exp	지수함수
ceil	ceil(-1.1), ceil(-1.0), ceil(1.0), ceil(1.1) → -1.0, -1.0, 1.0, 2.0
floor	floor(-1.1), floor(-1.0) → -2.0, -1.0 floor(1.0), floor(1.1) → 1.0, 1.0
greatest	greatest(1, 2, 3, 2) → 3
least	least(1, 2, 3, 2) → 1
mod	mod(11, 4), mod(11, -4), mod(-11, 4), mod(11, 0) → 3, 3, -3, 11
format	format(12.123456, 4), format(12.123, 0) → '12.1235', '12'

함수 [계속]

- 산술 함수

degrees	라디안 값을 각도로 변환
radians	각도 값을 라디안 값으로 변환
pi	3.1415926535897 93
round	round(1.355, 2) → 1.36
trunc, truncate	truncate(1.355, 2) → 1.35
rand	rand(), rand(3) → 임의의 정수값. 레코드 개수에 상관없이 동일 값
random	random(), random(3) → 임의의 정수값, 레코드별 다른 값
drand	drand(), drand(3) → <1 임의의 실수값. 레코드 개수에 상관없이 동일 값
drandom	drandom(), drandom(3) → <1 임의의 실수값, 레코드별 다른 값

함수 [계속]

- 날자/시간 함수

curdate	현재 date : curdate(), current_date, current_date(), sys_date, sysdate
now	현재 datetime : now(), current_datetime, current_datetime(), sys_datetime, sysdatetime
curtime	현재 time : curtime(), current_time, current_time(), sys_time, systime
current_timestamp	현재 timestamp : current_timestamp(), current_timestamp, sys_timestamp, systimestamp, localtime, localtime(), locatimestamp, localtimestamp()

date	date('2010-02-25 14:34:23') → '02/27/2010'
timestamp	timestamp('2010-12-31') → '12:00:00.000 AM 12/31/2010'
unix_timestamp	unix_timestamp() → 1300867571 unix_timestamp('2010-12-25') → 1293202800

함수 [계속]

- 날자/시간 함수

extract	extract(year |month |day |hour |minute |second |millisecond] from date'2010-12-25') → 2010
last_day	last_day(date'2010-2-20' → '02/28/2010'
add_month	add_month(date'2010-3-31', 1) → '04/30/2010'
adddate, date_add	date_add(date'2010-2-25', 3) → '02/28/2010' adddate(date'2010-2-25', interval '1:1' year_month) → '03/25/2011'
datediff	date_diff(date'2010-3-31', date'2010-4-1') → -1
date_sub, subdate	date_sub(date'2010-2-25', 3) → '02/22/2010' subdate(date'2010-2-25', interval '1:1' year_month) → '01/25/2009'

함수 [계속]

- 형변환 함수

cast	cast('1' as int), cast(12 ad char(3)) → 1, '12 '
date_format	date_format('2009-10-04', '%W %m %Y') → 'Sunday 10 2009'
str_to_date	str_to_date('Sunday 1 10 2009', '%W %d %m %Y') → '10/01/2009'
time_format	time_format(systime, '%T') → '20:46:53'
to_date	to_date('081225', 'YYMMDD') → '12/25/2008'
to_datetime	to_datetime('08-Dec-25 13:10:30.999', 'YY-Mon-DD HH24:MI:SS.FF') → '01:10:30.999 PM 12/25/2008'
to_time	to_time('13:10:30', 'HH24:MI:SS') → '01:10:30 PM'
to_timestamp	to_timestamp('08-Dec-25 13:10:30', 'YY-Mon-DD HH24:MI:SS') → '01:10:30 PM 12/25/2008'

함수 [계속]

- 형변환 함수

to_char	to_char(TIMESTAMP'2009-10-04 22:23:00', 'Day Month yyyy') → 'Sunday October 2009' to_char(12.456, '999.99') → ' 12.46'
to_number	to_number('12.456', '999.999') → 12.456

함수 [계속]

- 집계 함수

avg	평균값
count	개수
max	최대값
min	최소값
sum	합계
stddev	표준편차
variance	분산

함수 [계속]

- 조건 함수

case	select case gender when 'M' then '남성' when 'F' then '여성' end from event select case when host_year between 1900 and 1999 then '1900년대' when host_year >= 2000 then '2000년대' else '알수없는연대' end from record
decode	decode(gender, 'M', '남성', 'F', '여성', '중성')
if	if(gender='M', '남성', '여성')
isnull	isnull(NULL), isnull('a') → 1, 0
nullif	nullif('a', 'a'), nullif('a', 'b') → NULL, 'a'
ifnull, coleasce, nvl	ifnull(NULL, ''), ifnull('a', '') → '' , 'a'
nvl2	nvl2(NULL, ' not null value' , 'null value') → 'null value' nvl2('a', ' not null value' , 'null value') → 'not null value'

함수 [계속]

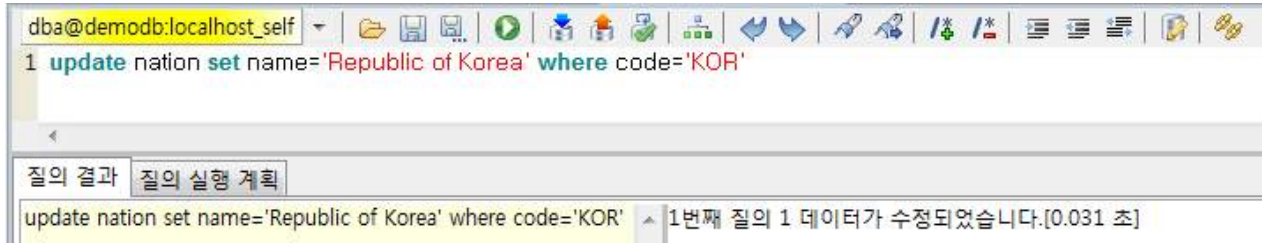
- 정보 함수

current_user, user user(), system_user()	user(), current_user → 'PUBLIC@localhost', 'PUBLIC'
database, schema	database() → 'demodb'
list_dbs	list_dbs() → 'demodb edudb'
default	default(id) from tbl → tbl 테이블의 id 컬럼 default 값
row_count	row_count() → 직전에 수행된 질의(insert,update,delete) 결과 영향을 받은 레코드 수

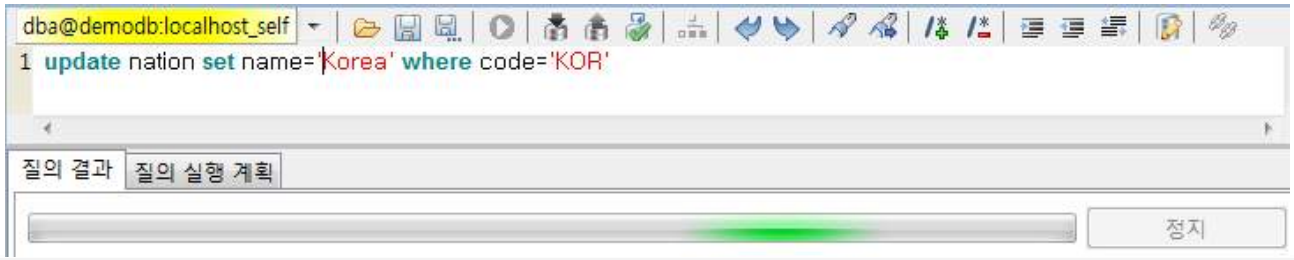
9. 트랜잭션 관리

lock 정보 확인

- 데이터베이스 사용자간 lock 정보 확인
  - dba 만 가능
- 특정 질의 수행중 lock 선점으로 인하여 타 질의 lock 기다림 현상
  - 질의 수행에 장시간 소요로 인하여 응용 반응 느려짐
    - update 질의 수행

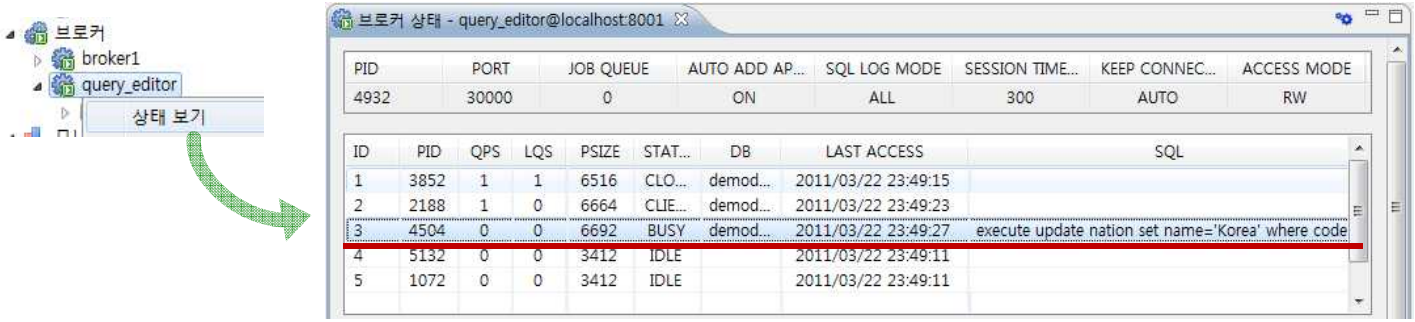


    - 다른 클라이언트에서 동일 레코드에 대하여 update 질의 수행

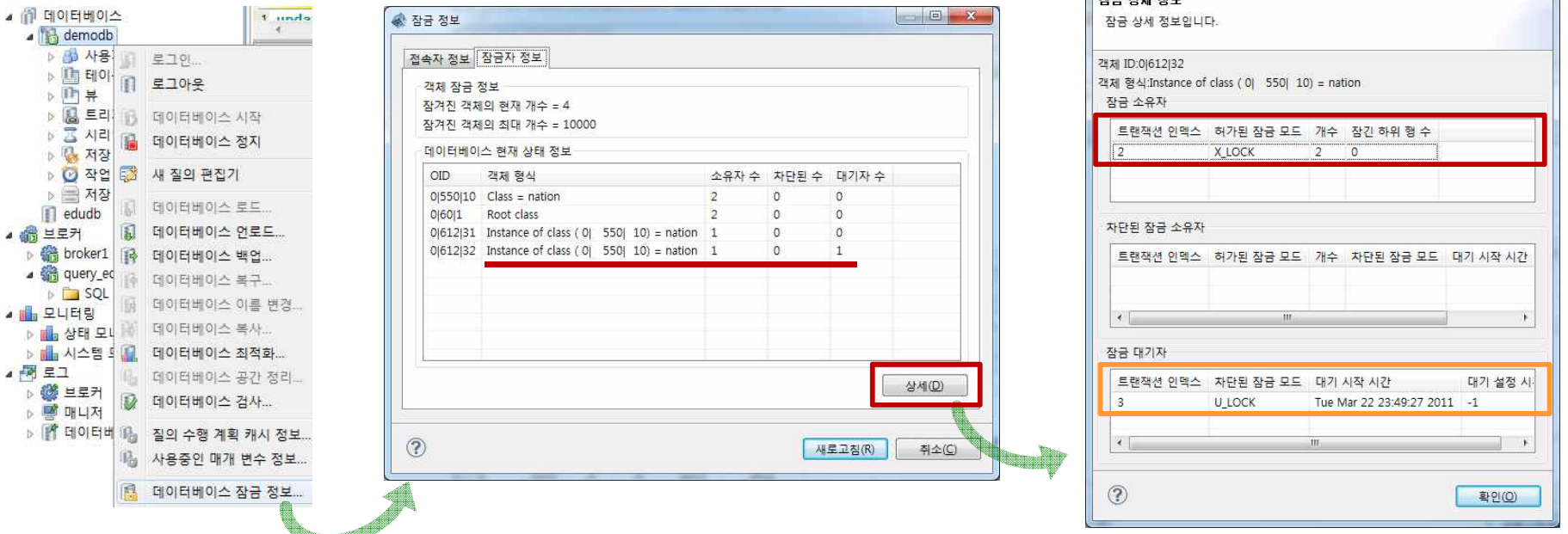


lock 정보 확인 [계속]

- broker 질의 처리 상황 모니터링



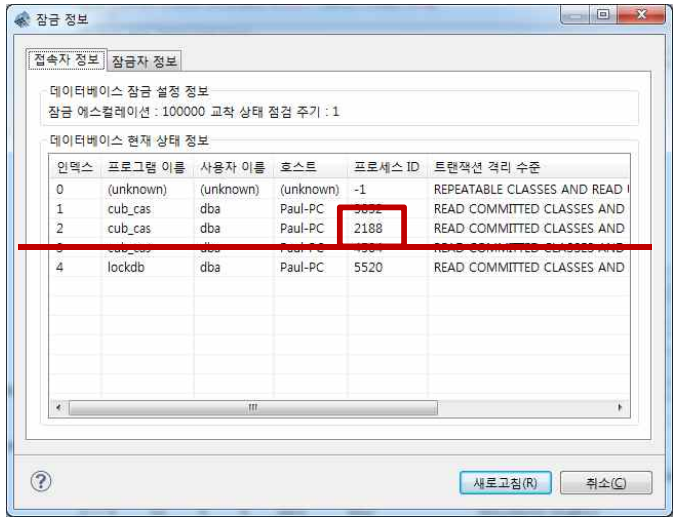
- 데이터베이스 잠금정보 확인



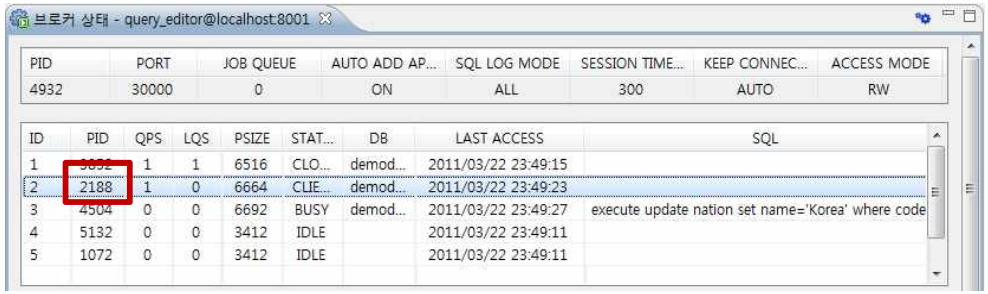
CUBRID

lock 정보 확인

- 데이터베이스 접속자 정보 확인



■ 접속자의 SQL log 확인




```
03/22 23:49:23.921 (0) connect db demodb@localhost user dba url
jdbc:cubrid:localhost:30000:demodb@localhost:dba::?charset=MS949
03/22 23:49:23.921 (0) DEFAULT isolation_level 1, lock_timeout -1
03/22 23:49:23.921 (0) get_db_parameter isolation_level
03/22 23:49:23.921 (1) prepare 0
update nation set name='Republic of Korea' where code='KOR'
03/22 23:49:23.952 (1) prepare srv_h_id 1
03/22 23:49:23.952 (1) execute srv_h_id 1 update nation set name='Republic of Korea' where
code='KOR'
03/22 23:49:23.952 (1) execute 0 tuple 1 time 0.031
```

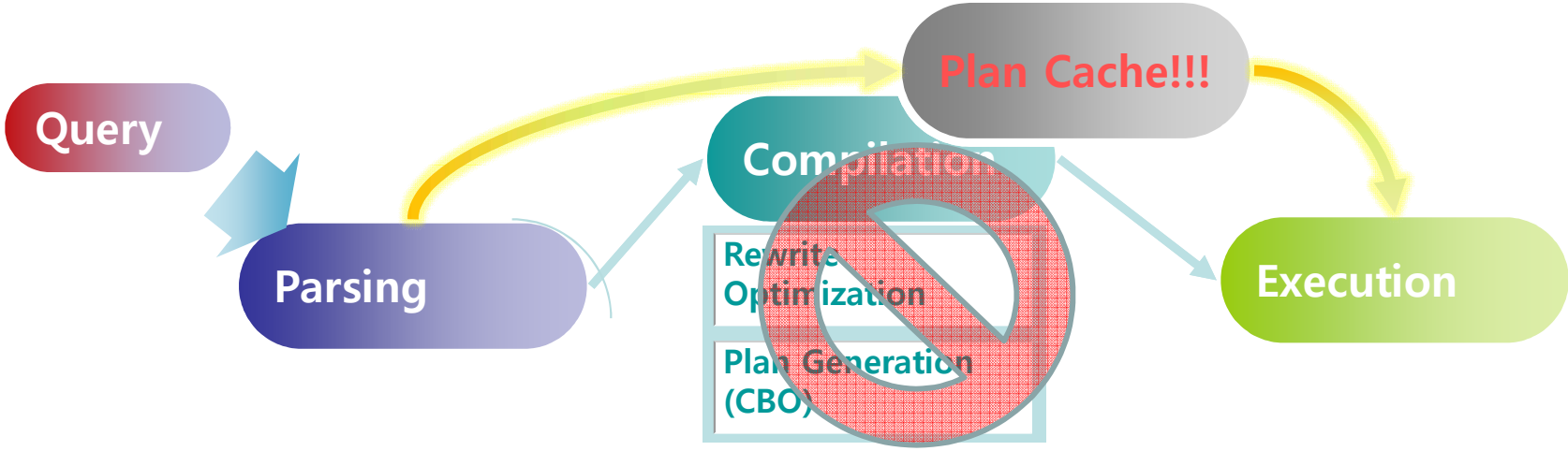
no commit, lock 소유중

CUBRID

10. 성능개선

질의 수행계획 캐쉬

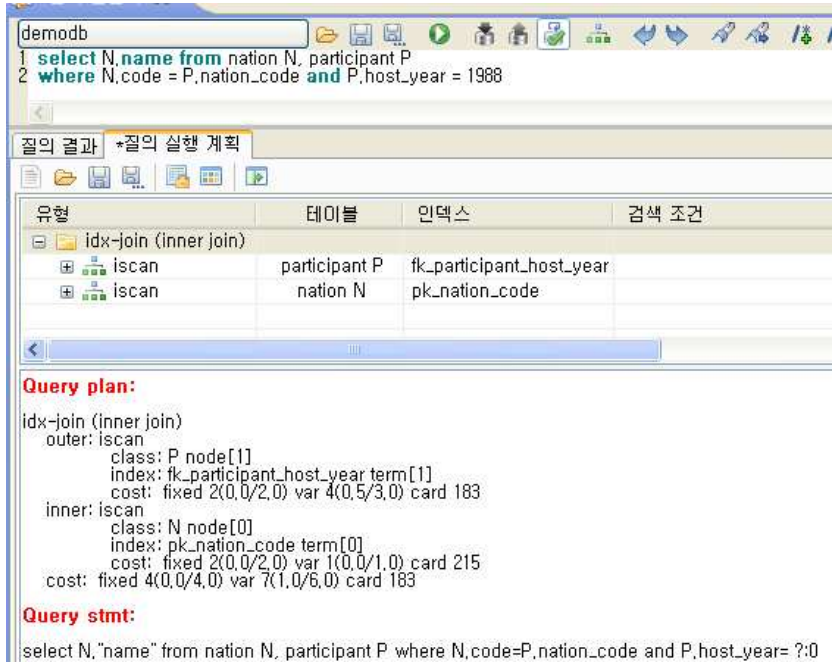
- 질의 수행을 위한 조인 방법, 인덱스 선택 등의 내용 캐쉬
- 동일 질의 수행시 캐쉬 활용
  - 단순 성능 비교시 약 30% 성능 향상
- 기본 설정
  - 1000개의 질의에 대하여 캐쉬



CUBRID

질의 성능 개선

- 질의 조정을 통한 성능 개선
  - loop 내에서 질의 반복 수행 → goup by / in 활용
  - 불필요한 조인 제거 (특히 뷰 사용시)
- 질의 수행계획을 통한 성능 개선
  - 최적의 인덱스 선택
  - 최적의 조인 방식 선택



질의 성능 개선 [계속]

- 통계 정보를 통한 성능 개선 (csql 만 가능)
  - ■ 질의 수행에 필요한 페이지수 최소화

```
csql> ;set optimization_level=257  // csql 에서 통계정보를 보기위한 설정
csql> ;set communication_histogram=yes
csql> ;.h on
csql> select /*+ recompile */ class_name from db_class
csql> ;ru
Query plan:

 Nested loops
    Index scan(participant P, fk_participant_host_year, P.host_year=1988)
    Index scan(nation N, pk_nation_code, N.code=P.nation_code)
...
csql> ;.x

Histogram of client requests:
...
 *** CLIENT EXECUTION STATISTICS ***
...
Elapsed (sec)              =      5449
 *** SERVER EXECUTION STATISTICS ***
...
Num_data_page_fetches      =      1027
Num_data_page_dirties      =        62
Num_data_page_ioreads      =        39
```

질의 성능 개선 [계속]

- 성능 개선 대상 질의 찾기
  - 수행중인 질의

```
% cubrid broker status -f

% broker1  -cub_cas [17083,33000] C:₩CUBRID₩log₩broker₩broker1.access
C:₩CUBRID₩log₩broker₩broker1.err
 JOB QUEUE:0, AUTO_ADD_APPL_SERVER:ON, SQL_LOG_MODE:ALL:100000
 LONG_TRANSACTION_TIME:60.00, LONG_QUERY_TIME:60.00, SESSION_TIMEOUT:300
 KEEP_CONNECTION:AUTO, ACCESS_MODE:RW
------------------------------------------------------------------------------------------------------------
ID   PID   QPS   LQS PSIZE STATUS   LAST ACCESS TIME       DB       HOST   LAST CONNECT TIME   CLIENT IP
------------------------------------------------------------------------------------------------------------
 1 17084   46     0 50068 BUSY     2011/03/02 23:11:38  demodb  localhost 2011/03/02 22:56:20  12.0.0.1
SQL: execute select count(*) from user_tbl …
 2 17085    0     0 46364 IDLE       2011/01/25 11:15:30          -          -          -
```

질의 성능 개선 [계속]

- 성능 개선 대상 질의 찾기
  - 수행된 질의
    - broker_log_top 을 이용하여 SQL log 분석

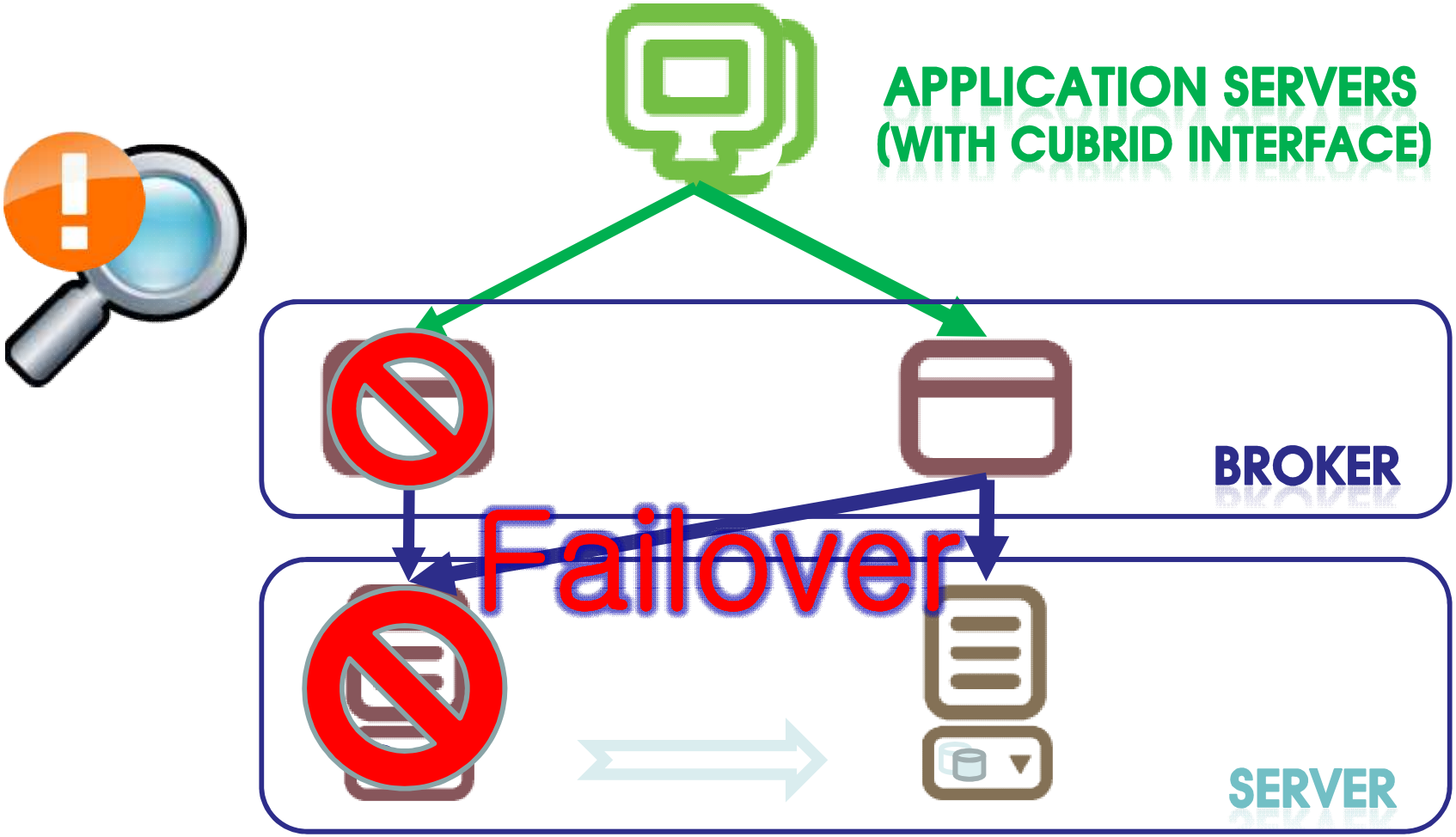
```
 %  cd C:₩CUBRID₩log₩broker₩sql_log
 %  broker_log_top [-f MM/DD] broker1*

 %  more log_top.res
 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)
...

%  more log_top.q
[Q1]-------------------------------
broker1_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]-------------------------------
broker1_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
```

11. HA

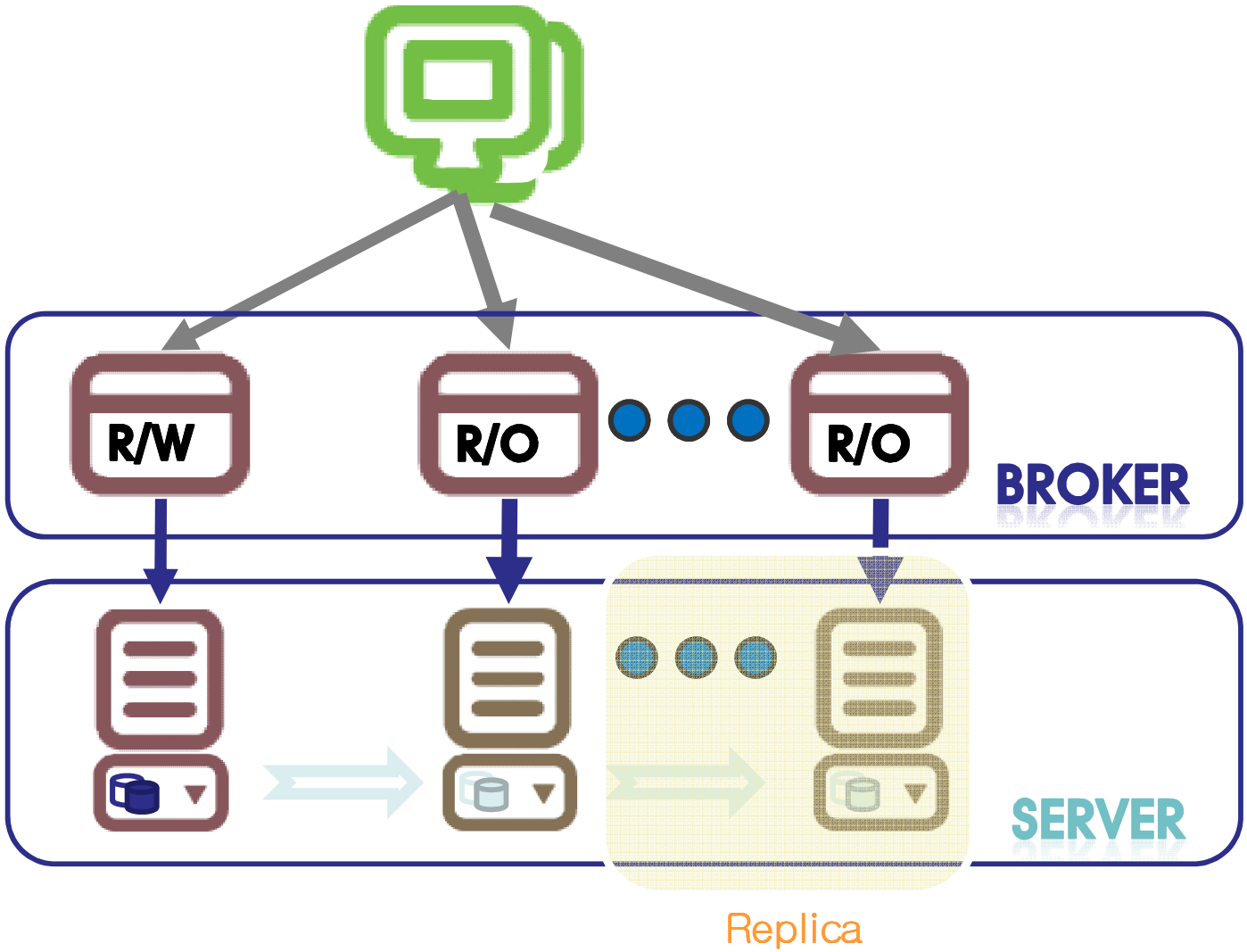
소개

- 장애 발생시 예비장비(stand-by server)로 신속한 서비스 전환을 통한 중단없는 서비스 구현
  - 장애 진단시 자동 절체(failover)
- 저장장치를 공유하지 않는 shared-nothing 방식
  - 로그 복제를 통한 실시간 예비 장비간 데이터베이스 동기화
- 운영서버의 중단없이 예비장비 구성
- 비동기 로그 복제를 통한 원격 IDC로의 백업 서버 구현 가능
  - semi-sync : 예비장비가 살아있는지만 확인
- 부하 분산을 위하여 예비장비 활용 및 replica 추가 가능
  - 추가 장비를 통한 읽기 부하 분산 가능

동작 구조



부하분산 구조



개발 환경

- 응용에서 데이터베이스 연결시 연결 방식 수정
  - database : demodb
  - active → IP : 1.1.0.1, port : 33000
  - stand-by → IP : 1.1.0.2, port : 34000

```
* JAVA
url = “jdbc:cubrid:1.1.0.1:33000:demodb:::?althosts=1.1.0.1:34000”;
DriverManager.getConnection(url, "dba", “dba_pw");

* PHP
url = “cci:cubrid:1.1.0.1:33000:demodb:::?althosts=1.1.0.1:34000”;
cubrid_connect_with_url(url , "dba", “dba_pw");

* CCI
url = “cci:cubrid:1.1.0.1:33000:demodb:::?althosts=1.1.0.1:34000”;
cci_connect_with_url(url , "dba", “dba_pw");

* ODBC : 지원않함
```

감 사 합 니 다.

보다 나은 교육을 위하여 설문 작성을 부탁드립니다.
www.cubrid.com/edu_survey.php

CUBRID