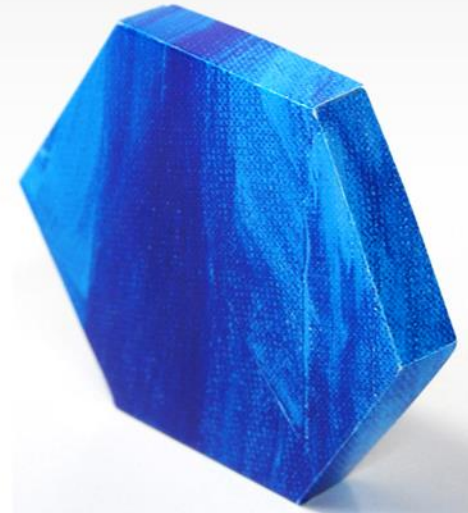


# CUBRID 개념/싱글/HA

공공PS1-2 윤준수 매니저

2022 . 02





# C Contents



I

CUBRID

II

CUBRID 설치

III

CUBRID 싱글

IV

CUBRID HA/복제



# I CUBRID

- 개념
- 특징
- 구조

# CUBRID - 개념

---

## CUBRID

- 국내 오픈소스(open source) RDBMS
- 타 오픈소스 RDBMS와 다르게 별도 기능에 대한 Edition 비용 없음
- Client – Server 모델로 구성되며 온라인 트랜잭션 (OLTP: On-line Transaction Processing) 서비스에 적합



# CUBRID - 특징

---

## CUBRID 엔진 특징 (SQL 제외)

- CUBRID를 업무적으로 사용해야하는 경우 하기와 같은 특징을 가지고 있어, 주의하셔야 합니다.
- **2008.4.4 이하 버전**
  - DB CHARSET 미지원, 응용프로그램의 CHARSET에 따라 데이터 입력
  - 2-PL Locking 지원, 의도 잠금 발생 (ROW/KEY/TABLE/SCHEMA LOCK)
- **9.x 버전**
  - DB CHARSET 지원, DB 생성 시 CHARSET으로 입력
  - 2-PL Locking 지원, 의도 잠금 발생 (ROW/KEY/TABLE/SCHEMA LOCK)
- **10.x 이상 버전**
  - DB CHARSET 지원, DB 생성 시 CHARSET으로 입력
  - MVCC (Multi Version Concurrency Control) 지원 (버전 관리를 통한 ROW/TABLE/SCHEMA LOCK)

# CUBRID - 구조

## CUBRID 브로커

- 3-Tier 구조(응용프로그램 - 브로커 - 데이터베이스)



### Application

- JDBC, ODBC, OLEDB, Python, GO ...등 지원

### Broker

- 응용프로그램과 데이터베이스 간의 연결 및 관리 수행
- 브로커를 통하지 않으면 일반적인 경우 접속 불가
- CCI : CUBRID Native Interface를 사용한 프로그램은 브로커를 통하지 않고 데이터베이스 직접 접근 가능

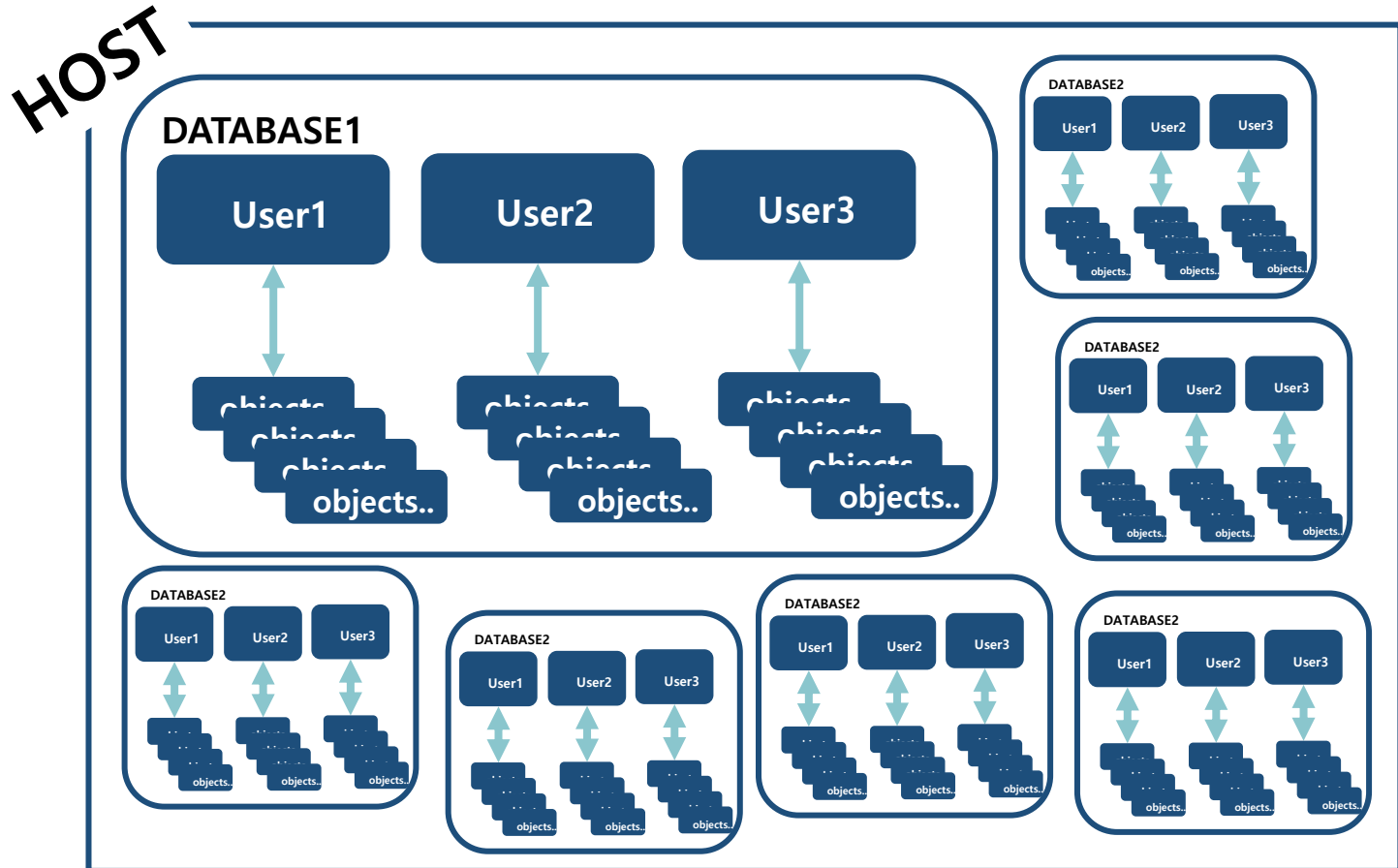
### DATABASE

- RDBMS 기본적인 기능들을 제공
  - ANSI SQL-92 표준 및 특성 SQL 제공
  - ACID (Atomicity | Consistency | Isolation | Durability)
  - Multi-Transaction (버전: 10 ↑ MVCC | 9 ↓ 2PL-LOCK)
  - Multi-Thread (프로세스 내 다중스레드로 로직 처리)
  - Storage(DISK RDBMS, In-memory 미지원)
  - Logging (Archive Log, 정합성 보장)
  - 브로커 다중화, 데이터베이스 다중화(HA) 지원
  - 등등..

# CUBRID - 구조

## CUBRID 데이터베이스

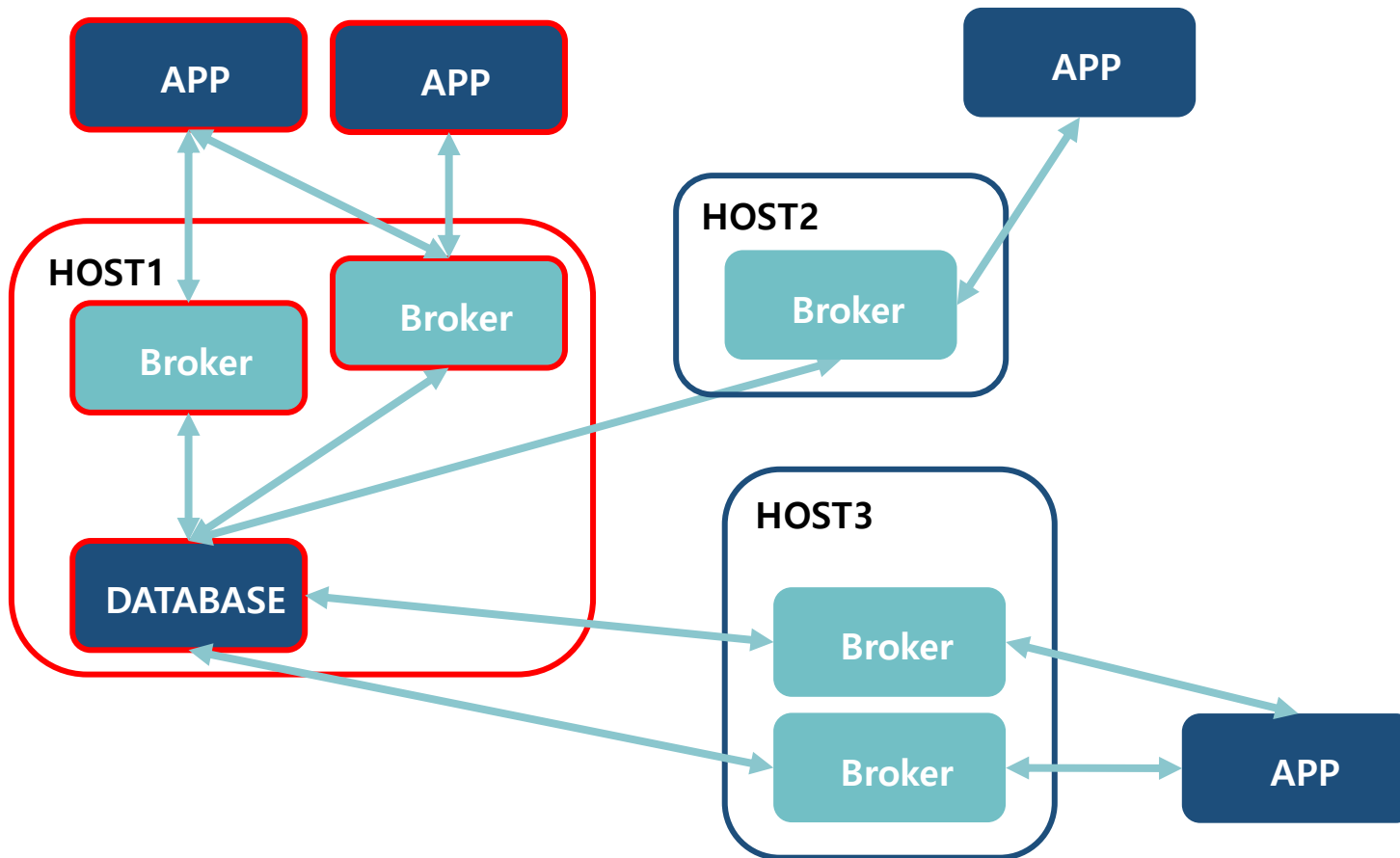
- HOST 1개에 여러 개의 데이터베이스 생성 가능
  - CUBRID 엔진내에 동일한 DATABASE명을 가질 수 없음
  - DATABASE내에 동일한 유저명을 가질 수 없음
  - DATABASE내에 동일한 테이블명 가질 수 없음 (유저가 다르더라도/인덱스 동일한명을 가질 수 있음)



# CUBRID - 구조

## CUBRID 브로커 다중화

- 1개의 호스트에 여러 개의 브로커를 사용할 수 있음
- 타 HOST에 브로커를 사용 할 수 있음
- 타 HOST에 여러 개의 브로커를 사용할 수 있음
- 성능상 이점이 크게 없어 1개 호스트의 여러 개의 브로커 사용
  - 개발용 브로커 / 서비스용 브로커

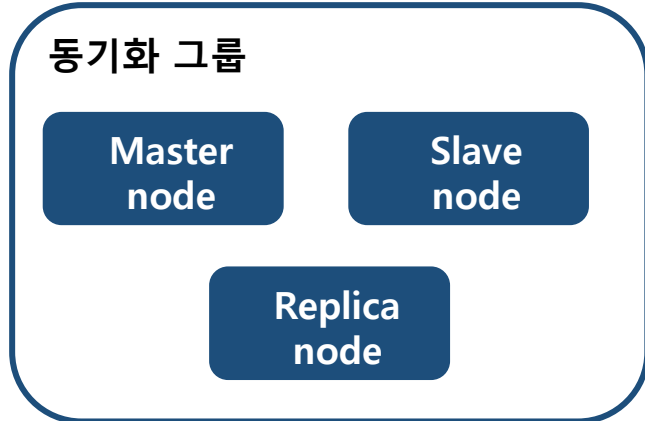




# CUBRID - 구조

## CUBRID 데이터 베이스 이중화/다중화

- 클러스터 개념 없음
  - 동기화 그룹
- 다양한 다중화 구성을 할 수 있음
- **M(Active)-S(Standby)**
  - Active READ/WRITE 가능
  - Standby WRITE 가능
- **M(Active)-S(Standby)-S(Standby)**
  - Active READ/WRITE 가능
  - Standby READ 가능 (트러블 슈팅 시 WRITE 가능)
  - Standby READ 가능 (트러블 슈팅 시 WRITE 가능)
- **M(Active)-S(Standby)-R(Replica)**
  - Active READ/WRITE 가능
  - Standby READ 가능 (트러블 슈팅 시 WRITE 가능)
  - Replica READ/선택적 WRITE 가능
- **M(Active)-S(Standby)-R(Replica)-R(Replica)**
  - Active READ/WRITE 가능
  - Standby READ 가능 (트러블 슈팅 시 WRITE 가능)
  - Replica READ/선택적 WRITE 가능
  - Replica READ/선택적 WRITE 가능





## II

# CUBRID 설치

- 준비
- 설치
- 확인

# CUBRID 설치 - 준비

---

## CUBRID 브로커 다중화


















1. FTP 웹 사이트 접속 후 엔진 파일 다운로드 및 wget 링크로 내려 받기
2. CUBRID OS 유저 생성
3. OS Resource Limit, /etc/hosts 설정
4. CUBRID 엔진 압축해제
5. CUBRID 환경 변수 설정
6. CUBRID 명령 확인

# CUBRID 설치 - 준비

## CUBRID 설치 파일 준비

- CUBRID 지원 플랫폼 : Windows, Linux
- 테스트 환경 : Linux (CentOS 7.9.2009)
- 다운로드 : [http://ftp.cubrid.org/CUBRID\\_Engine/11.0\\_latest/](http://ftp.cubrid.org/CUBRID_Engine/11.0_latest/)

## Index of /CUBRID\_Engine/11.0\_latest

Name	Last modified	Size	Description
 <a href="#">Parent Directory</a>	-	-	-
 <a href="#">CUBRID-11.0-latest-Linux.x86_64.rpm</a>	2022-02-17 12:39	61M	
 <a href="#">CUBRID-11.0-latest-Linux.x86_64.sh</a>	2022-02-17 12:39	88M	
 <a href="#">CUBRID-11.0-latest-Linux.x86_64.tar.gz</a>	2022-02-17 12:39	88M	
 <a href="#">CUBRID-11.0.6.0313-e1160bb-Linux.x86_64.rpm</a>	2022-02-17 12:39	61M	
 <a href="#">CUBRID-11.0.6.0313-e1160bb-Linux.x86_64.sh</a>	2022-02-17 12:39	88M	
 <a href="#">CUBRID-11.0.6.0313-e1160bb-Linux.x86_64.tar.gz</a>	2022-02-17 12:39	88M	
 <a href="#">CUBRID-Windows-x64-11.0-latest.msi</a>	2022-02-17 12:39	28M	
 <a href="#">CUBRID-Windows-x64-11.0-latest.zip</a>	2022-02-17 12:39	22M	
 <a href="#">CUBRID-Windows-x64-11.0.6.0313-e1160bb.msi</a>	2022-02-17 12:39	28M	
 <a href="#">CUBRID-Windows-x64-11.0.6.0313-e1160bb.zip</a>	2022-02-17 12:39	22M	
 <a href="#">check_reserved.sql</a>	2022-02-17 12:39	1.8K	
 <a href="#">cubrid-11.0-latest.tar.gz</a>	2022-02-17 12:39	65M	
 <a href="#">cubrid-11.0-latest.zip</a>	2022-02-17 12:39	48M	
 <a href="#">cubrid-11.0.6.0313-e1160bb.tar.gz</a>	2022-02-17 12:39	65M	
 <a href="#">cubrid-11.0.6.0313-e1160bb.zip</a>	2022-02-17 12:39	48M	
 <a href="#">md5sum-11.0.6.0313-e1160bb.txt</a>	2022-02-17 12:39	523	

# CUBRID 설치 - 준비

## CUBRID OS 유저 생성

```
[root@DB01 ~]# useradd cubrid
[root@DB01 ~]# su - cubrid
[cubrid@DB01 ~]$
```

## /etc/hosts 수정

```
[root@DB01 ~]# cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.152.101 DB01
```

## Resource Limit 수정

```
[root@DB01 ~]# cat /etc/security/limits.conf
cubrid      soft          nfile       100000
cubrid      hard           nfile       100000

[root@DB01 ~]# cat /etc/security/limits.conf
cubrid soft   nproc   100000
cubrid hard   nproc   100000
```

## CUBRID 엔진 파일 다운로드 및 구성 (SFTP 업로드 또는 wget 링크 다운로드)

```
[cubrid@DB01 ~]$ wget http://ftp.cubrid.org/CUBRID_Engine/11.0_latest/CUBRID-11.0-latest-Linux.x86_64.tar.gz
[cubrid@DB01 ~]$ ls
CUBRID-11.0-latest-Linux.x86_64.tar.gz
[cubrid@DB01 ~]$ tar -xvzf CUBRID-11.0-latest-Linux.x86_64.tar.gz
[cubrid@DB01 ~]$ ls
CUBRID  CUBRID-11.0-latest-Linux.x86_64.tar.gz
```

# CUBRID 설치 - 설치

## CUBRID 엔진 환경 설정

```
[cubrid@DB01 ~]$ vi .cubrid.sh
CUBRID=/home/cubrid/CUBRID
CUBRID_DATABASES=$CUBRID/databases
if [ "x${LD_LIBRARY_PATH}x" = xx ]; then
    LD_LIBRARY_PATH=$CUBRID/lib
else
    LD_LIBRARY_PATH=$CUBRID/lib:$LD_LIBRARY_PATH
fi
SHLIB_PATH=$LD_LIBRARY_PATH
LIBPATH=$LD_LIBRARY_PATH
PATH=$CUBRID/bin:$PATH
export CUBRID
export CUBRID_DATABASES
export LD_LIBRARY_PATH
export SHLIB_PATH
export LIBPATH
export PATH

LIB=$CUBRID/lib

if [ -f /etc/redhat-release ];then
    OS=$(cat /etc/system-release-cpe | cut -d':' -f3-3')
elif [ -f /etc/os-release ];then
    OS=$(cat /etc/os-release | egrep "^ID=" | cut -d'=' -f2-2)
fi

case $OS in
    fedoraproject)
        if [ ! -h /lib64/libncurses.so.5 ] && [ ! -h $LIB/libncurses.so.5 ];then
            ln -s /lib64/libncurses.so.6 $LIB/libncurses.so.5
            ln -s /lib64/libform.so.6 $LIB/libform.so.5
            ln -s /lib64/libtinfo.so.6 $LIB/libtinfo.so.5
        fi
        ;;
    centos)
        if [ ! -h /lib64/libncurses.so.5 ] && [ ! -h $LIB/libncurses.so.5 ];then
            ln -s /lib64/libncurses.so.6 $LIB/libncurses.so.5
            ln -s /lib64/libform.so.6 $LIB/libform.so.5
            ln -s /lib64/libtinfo.so.6 $LIB/libtinfo.so.5
        fi
        ;;
    ubuntu)
        if [ ! -h /lib/x86_64-linux-gnu/libncurses.so.5 ] && [ ! -h $LIB/libncurses.so.5 ];then
            ln -s /lib/x86_64-linux-gnu/libncurses.so.6 $LIB/libncurses.so.5
            ln -s /lib/x86_64-linux-gnu/libform.so.6 $LIB/libform.so.5
            ln -s /lib/x86_64-linux-gnu/libtinfo.so.6 $LIB/libtinfo.so.5
        fi
        ;;
    debian)
        if [ ! -h /lib/x86_64-linux-gnu/libncurses.so.5 ] && [ ! -h $LIB/libncurses.so.5 ];then
            ln -s /lib/x86_64-linux-gnu/libncurses.so.6 $LIB/libncurses.so.5
            ln -s /lib/x86_64-linux-gnu/libtinfo.so.6 $LIB/libtinfo.so.5
            ln -s /usr/lib/x86_64-linux-gnu/libform.so.6 $LIB/libform.so.5
        fi
        ;;
    esac
```

# CUBRID 설치 - 설치

---

## CUBRID 엔진 환경 설정

```
[cubrid@DB01 ~]$ vi $HOME/.bash_profile
# 마지막 라인에 추가
#-----
# set CUBRID environment variables
#-----
if [ -f $HOME/.cubrid.sh ];then
. $HOME/.cubrid.sh
fi
[cubrid@DB01 ~]$ source .bash_profile
```

# CUBRID 설치 - 확인

---

## CUBRID 엔진 설치 확인

```
[cubrid@DB01 ~]$ cubrid
cubrid utility, version 11.0
usage: cubrid <utility-name> [args]
Type 'cubrid <utility-name>' for help on a specific utility.
```

Available service's utilities:

- service
- server
- broker
- manager

Available administrator's utilities:

- addvoldb
- alterdbhost
- ... 생략



# CUBRID 설치 - 확인

## CUBRID 엔진 구조

CUBRID	설명
<b>bin</b>	CUBRID 유틸 바이너리
<b>compat</b>	CUBRID 관리 유틸 바이너리
<b>conf</b>	CUBRID 구성 설정
<b>database</b>	CUBRID 데이터베이스 구성 정보 및 데이터 저장 기본 경로 (기동 시 자동 생성)
<b>demo</b>	CUBRID 데모 데이터
<b>include</b>	CUBRID 라이브러리 헤더
<b>java</b>	CUBRID jasp 프로세스 구성 설정
<b>jdbc</b>	CUBRID JDBC 드라이버
<b>lib</b>	CUBRID 라이브러리
<b>locales</b>	CUBRID 로케일 정보
<b>log</b>	CUBRID 에러/메시지 로그 기본 경로
<b>msg</b>	CUBRID 에러 코드 정보
<b>share</b>	CUBRID 유지관리 스크립트
<b>timezones</b>	CUBRID 타임 존 정보
<b>tmp</b>	CUBRID 임시 경로 (기동 시 자동 생성)
<b>var</b>	CUBRID 소켓 및 PID 파일 (기동 시 자동 생성)

```
$ tree -L 1 -d CUBRID
CUBRID
├── bin
├── compat
├── conf
├── database
├── demo
├── include
├── java
├── jdbc
├── lib
├── locales
├── log
├── msg
├── share
├── timezones
├── tmp
└── var
```



### III

## CUBRID 싱글

- 데이터 베이스 생성
- 데이터 베이스 접속

# CUBRID 싱글 – 생성

## CUBRID 데이터 베이스 생성 유틸

```
[cubrid@DB01 testdb]$ cubrid createdb
createdb: Create a database.
usage: cubrid createdb [OPTION] database-name database-locale
... 생략
```

**CREATEDB Manual** : [https://www.cubrid.org/manual/ko/11.0/admin/admin\\_utils.html#createdb](https://www.cubrid.org/manual/ko/11.0/admin/admin_utils.html#createdb)

## CUBRID 데이터 베이스 생성

```
[cubrid@DB01 ~]$ cd $CUBRID_DATABASES
[cubrid@DB01 databases]$ mkdir testdb
[cubrid@DB01 databases]$ cd testdb
[cubrid@DB01 testdb]$ cubrid createdb testdb ko_KR.utf8
Creating database with 512.0M size using locale ko_KR.utf8. The total amount of disk space needed is 1.5G.
```

CUBRID 11.0

```
[cubrid@DB01 testdb]$ ls
lob  testdb  testdb_keys  testdb_lgar_t  testdb_lgat  testdb_lginf  testdb_vinf
```

## CUBRID 데이터 베이스 기동

```
[cubrid@DB01 testdb]$ cubrid server start testdb
@ cubrid server start: testdb
```

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

CUBRID 11.0

```
++ cubrid server start: success
```

# CUBRID 싱글 – 접속

## CUBRID csql 접속

```
[cubrid@DB01 ~]$ csql
A database-name is missing.
interactive SQL utility, version 11.0
usage: csql [OPTION] database-name[@host]
... 생략
```

**csql Manual :** <https://www.cubrid.org/manual/ko/11.0/csql.html?highlight=csql#csql>

```
[cubrid@DB01 ~]$ csql -u dba testdb

          CUBRID SQL Interpreter

Type `;help' for help messages.

csql> SELECT version();

=== <Result of SELECT Command in Line 1> ===

      version()
=====
      '11.0.6.0313'

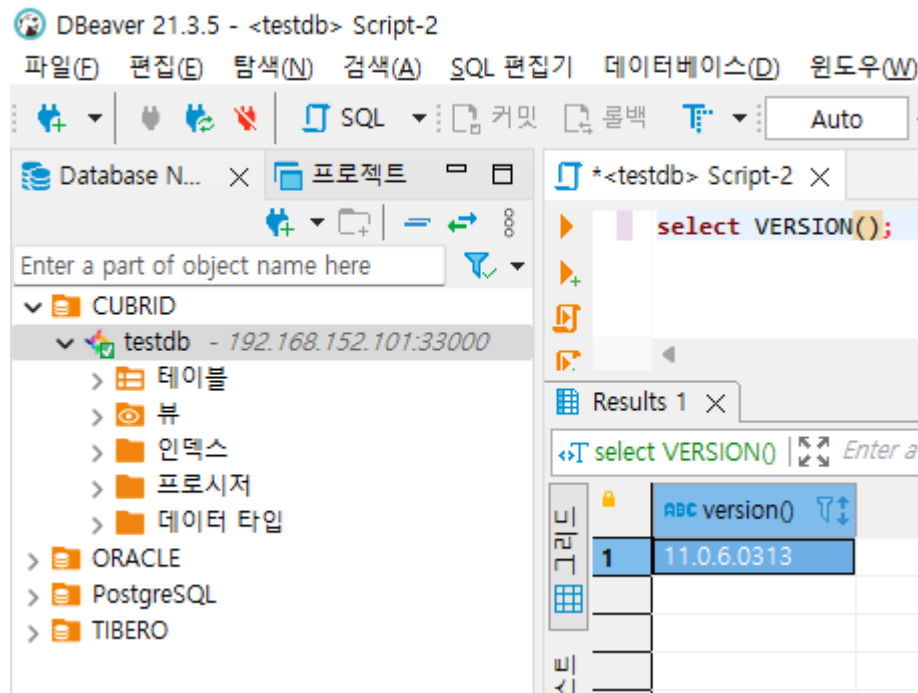
1 row selected. (0.002907 sec) Committed.

1 command(s) successfully processed.
csql> ;ex
[cubrid@DB01 ~]$
```

# CUBRID 싱글 – 접속

## CUBRID 브로커 기동

```
[cubrid@DB01 testdb]$ cubrid broker start
@ cubrid broker start
++ cubrid broker start: success
```



# CUBRID 싱글 – 접속

## 1 CUBRID USER 정보

기본적으로 dba / public 유저가 생성되며 기본 암호는 설정되어 있지 않음

- dba : 데이터 베이스의 전반적인 관리자 기능 모두 가능
- public : 자신이 소유한 객체에만 접근 가능
- MANUAL : <https://www.cubrid.org/manual/ko/11.0/sql/authorization.html?highlight=dba#id2>

## 2 CUBRID 서비스 포트

CUBRID 3-tier 구조로 중간에 브로커가 존재 합니다.

기본적으로 2개의 브로커로 설정 되어 있습니다.

BROKER NAME : query\_editor / PORT : TCP\_30000 / SQL를 사용하는 개발자 툴에서 사용 되는 포트

BROKER NAME : broker1 / PORT : TCP\_33000 / WAS 또는 Application에서 사용 하는 서비스 포트

## 3 CUBRID Connection URL

- 싱글 : jdbc:CUBRID:<IP>:<브로커 PORT >:<dbname>:<user>:<password>:?charset=<charset>
- HA : jdbc:CUBRID:<Master node IP>:<브로커 PORT>:<dbname>:<user>:<password>:  
?altHosts=<Slave node IP>:<브로커 PORT >& charset=< charset>
- 복제 : jdbc:CUBRID:<Replica node IP>:<브로커 PORT >:<dbname>:<user>:<password>:?charset=< charset>
- MANUAL : <https://www.cubrid.org/manual/ko/11.0/api/jdbc.html#id2>



## IV

# CUBRID HA/복제

- HA
- HA 구성
- HA Fail-Over / Fail-Back
- 복제
- 복제 구성
- 복제 선택적 쓰기 구성

# CUBRID HA/복제 – HA

CUBRID HA 구성은 1개의 Master node(active)와 복수의 Slave node(standby)로 구성할 수 있습니다.

Master node에서 발생된 트랜잭션 로그를 SQL 형태로 변형한 후 Slave node에서 수행하기 때문에 KEY 제약 조건 존재 HA 제약 사항 MANUAL : <https://www.cubrid.org/manual/ko/11.0/ha.html?highlight=replication#id37>

Fail-Over가 발생할 경우 Slave node 중 우선 순위가 높은 Slave node가 Active 상태가 됩니다.

Active 상태가 된 Slave node는 Fail-Over 전의 Master node와 같은 상태가 됩니다.

CUBRID HA - Master node : Slave node 동기화 구성

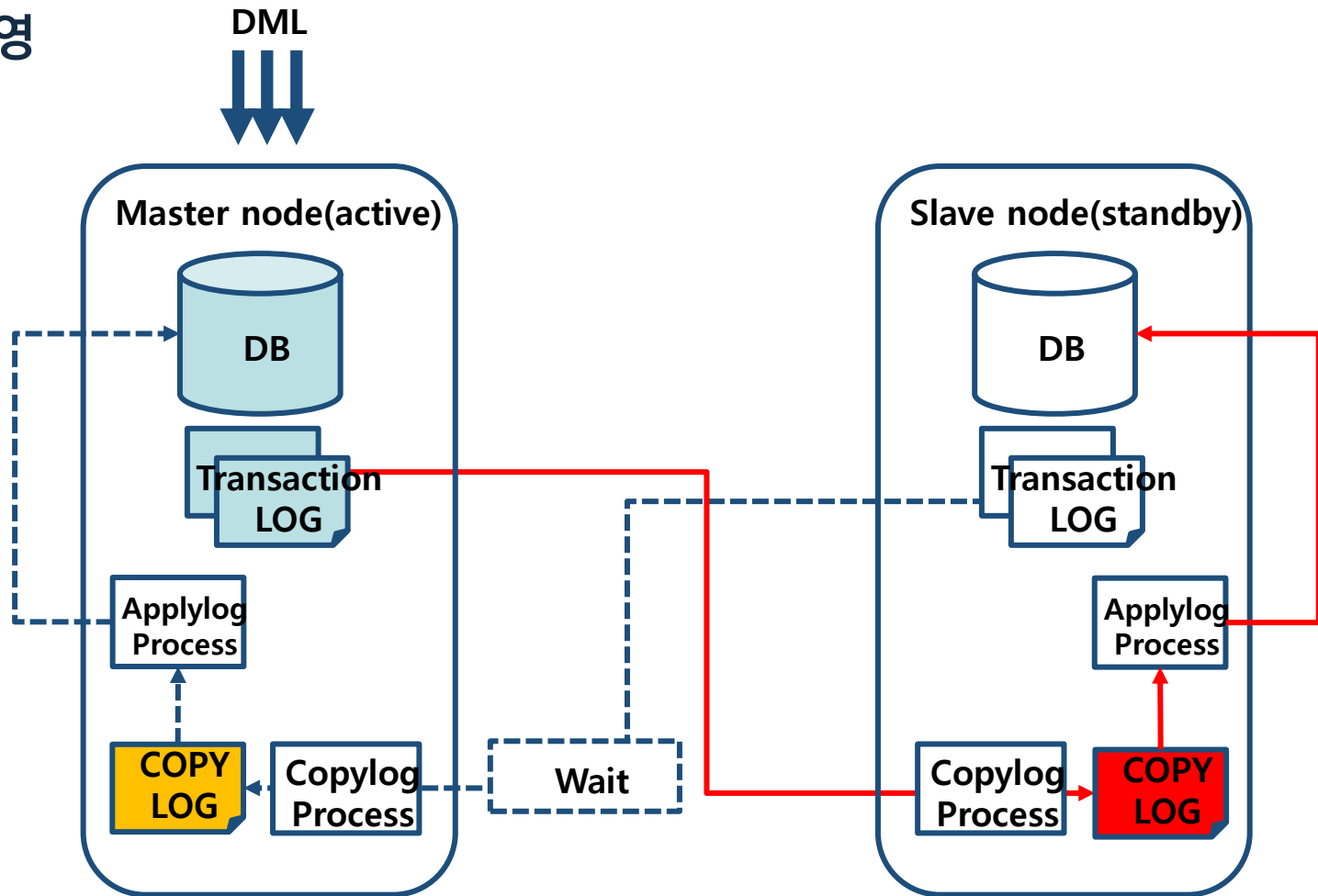
1. Master node DML(SELECT 제외)이 발생하여 트랜잭션 로그에 기록 합니다.
2. Slave node의 Copylogdb 프로세스가 Master node의 트랜잭션 로그를 Slave node로 가져 옵니다.
3. Slave node의 applylogdb 프로세스는 Copylogdb 프로세스가 가지고 온 트랜잭션 로그를 SQL 형태로 변경하여, Slave node의 DB에 반영 합니다.

- Fail-Over 발생 시 위 절차의 Master node와 Slave node가 반대로 동작한다고 보면 됩니다.



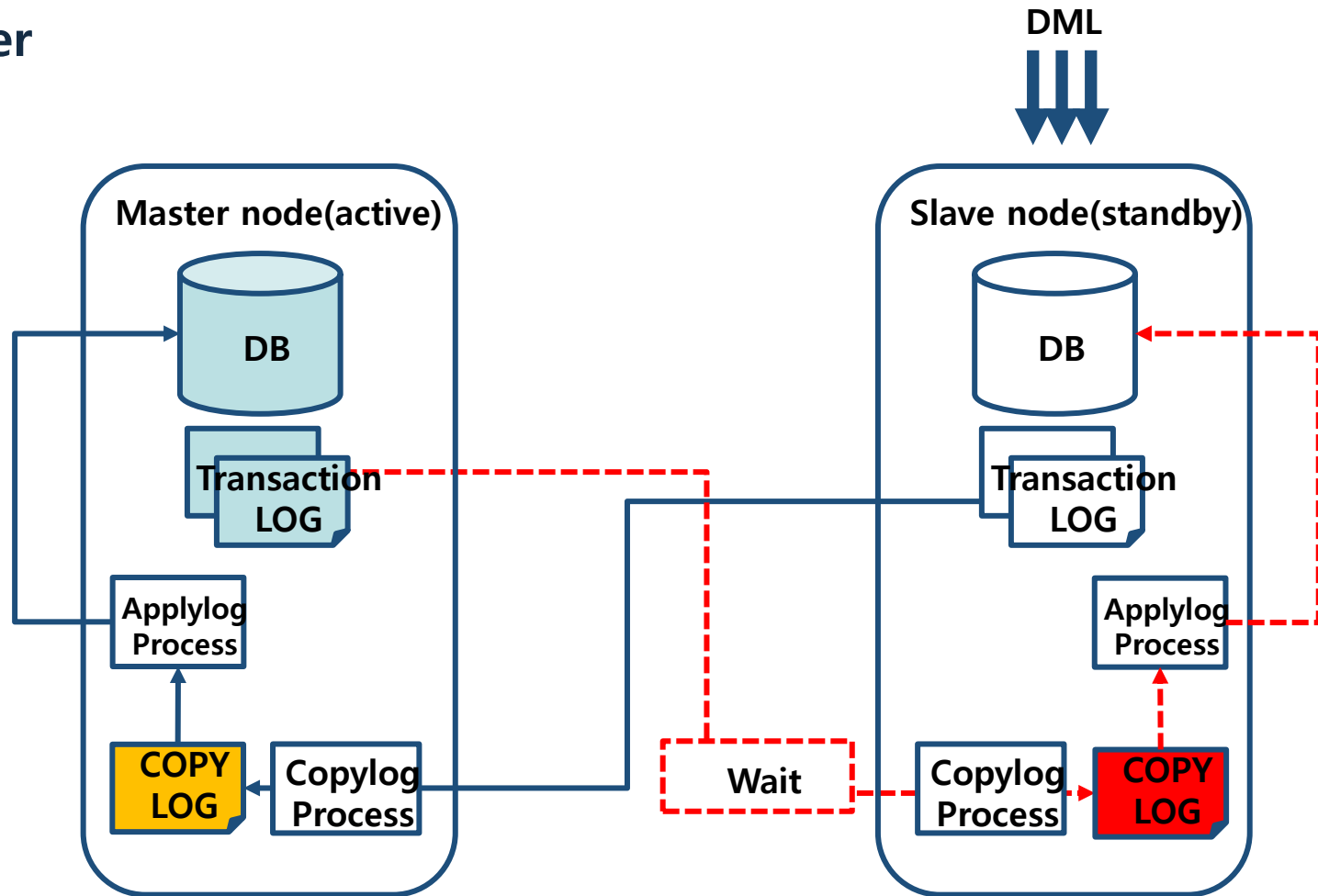
# CUBRID HA/복제 – HA

정상 운영



# CUBRID HA/복제 - HA

## Fail-Over



# CUBRID HA/복제 – HA 구성

## Slave node 데이터 파일 구성

```
$ cubrid service stop
```

```
[cubrid@DB01 ~]$ cd $CUBRID_DATABASE  
[cubrid@DB01 databases]$ tar -cvzf testdb.tar.gz testdb
```

```
# sftp 또는 scp로 DB02으로 testdb.tar.gz 전송  
# sftp 또는 scp로 DB02으로 database.txt 전송
```

```
[cubrid@DB02 ~]$ cd $CUBRID_DATABASE  
[cubrid@DB02 databases]$ ls  
databases.txt testdb.tar.gz  
[cubrid@DB02 databases]$ tar -xvzf testdb.tar.gz  
[cubrid@DB02 databases]$ ls  
databases.txt testdb.tar.gz testdb
```

# CUBRID HA/복제 – HA 구성

## Master / Slave node 동일하게 설정

```
$ cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.152.101 DB01 # Master node
192.168.152.102 DB02 # Slave node
```

```
$ cat $CUBRID/conf/cubrid.conf
service=heartbeat,broker,manager
... 생략
## HA
ha_mode=on
force_remove_log_archives=no
log_max_archives=20
```

```
$ cat $CUBRID/conf/cubrid_ha.conf
... 생략
#[common]
ha_port_id=59901
ha_node_list=cubrid@DB01:DB02
ha_db_list=testdb
ha_apply_max_mem_size=300
ha_copy_sync_mode=sync:sync
ha_copy_log_max_archives=1
```

```
$ cat $CUBRID_DATABASES/databases.txt
##db-name      vol-path      db-host      log-path      lob-base-path
testdb         /home/cubrid/CUBRID/databases/testdb  DB01:DB02    /home/cubrid/CUBRID/databases/testdb
```

# CUBRID HA/복제 – HA 구성

## Master node 기동/확인

```
[cubrid@DB01 ~]$ cubrid hb start
@ cubrid heartbeat start
@ cubrid master start
++ cubrid master start: success
@ HA processes start
@ cubrid server start: testdb
```

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

CUBRID 11.0

```
++ cubrid server start: success
@ copylogdb start
++ copylogdb start: success
@ applylogdb start
++ applylogdb start: success
++ HA processes start: success
++ cubrid heartbeat start: success
```

```
[cubrid@DB01 ~]$ cubrid hb status
@ cubrid heartbeat status
```

```
HA-Node Info (current DB01, state master)
Node DB02 (priority 2, state unknown)
Node DB01 (priority 1, state master)
```

```
HA-Process Info (master 59941, state master)
Applylogdb testdb@localhost:/home/cubrid/CUBRID/databases/testdb_DB02 (pid 60182, state registered)
Copylogdb testdb@DB02:/home/cubrid/CUBRID/databases/testdb_DB02 (pid 60180, state registered)
Server testdb (pid 59949, state registered_and_active)
```

# CUBRID HA/복제 – HA 구성

## Slave node 기동/확인

```
[cubrid@DB02 databases]$ cubrid hb start
@ cubrid heartbeat start
@ cubrid master start
++ cubrid master start: success
@ HA processes start
@ cubrid server start: testdb
```

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

CUBRID 11.0

```
++ cubrid server start: success
@ copylogdb start
++ copylogdb start: success
@ applylogdb start
++ applylogdb start: success
++ HA processes start: success
++ cubrid heartbeat start: success
```

```
[cubrid@DB02 databases]$ cubrid hb status
@ cubrid heartbeat status
```

```
HA-Node Info (current DB02, state slave)
  Node DB02 (priority 2, state slave)
  Node DB01 (priority 1, state master)
```

```
HA-Process Info (master 7580, state slave)
  Applylogdb testdb@localhost:/home/cubrid/CUBRID/databases/testdb_DB01 (pid 7819, state registered)
  Copylogdb testdb@DB01:/home/cubrid/CUBRID/databases/testdb_DB01 (pid 7817, state registered)
  Server testdb (pid 7588, state registered_and_standby)
```

# CUBRID HA/복제 – HA 구성

## Master node에서 DDL 수행

```
[cubrid@DB01 ~]$ csql -u dba testdb@DB01 -c "CREATE TABLE junsu(x int primary key);"  
Execute OK. (0.004821 sec) Committed.
```

## Master node 데이터 입력 후 각 node에서 데이터 확인

```
[cubrid@DB01 ~]$ csql -u dba testdb@DB01 -c "INSERT INTO junsu(x) VALUES(1);"  
1 row affected. (0.002610 sec) Committed.
```

```
[cubrid@DB01 ~]$ csql -u dba testdb@DB01 -c "SELECT * FROM junsu;"
```

```
=== <Result of SELECT Command in Line 1> ===
```

```
      x  
=====
```

1
---

```
1 row selected. (0.003112 sec) Committed.
```

```
[cubrid@DB01 ~]$ csql -u dba testdb@DB02 -c "SELECT * FROM junsu;"
```

```
=== <Result of SELECT Command in Line 1> ===
```

```
      x  
=====
```

1
---

```
1 row selected. (0.002763 sec) Committed.
```

# CUBRID HA/복제 – HA 구성

Slave node에서 동기화 확인 명령어를 수행하여 동기화 확인

```
[cubrid@DB02 databases]$ cubrid applyinfo -a -r DB01 -L $CUBRID_DATABASES/testdb_DB01 testdb
```

```
*** Applied Info. ***
```

```
Insert count      : 1
Update count      : 0
Delete count      : 0
Schema count      : 1
Commit count      : 319
Fail count        : 0
```

```
... 중략 ...
```

```
*** Delay in Copying Active Log ***
```

```
Delayed log page count : 0
Estimated Delay        : - second(s)
```

```
*** Delay in Applying Copied Log ***
```

```
Delayed log page count : 0
Estimated Delay        : - second(s)
```

**applyinfo MANUAL : <https://www.cubrid.org/manual/ko/11.0/ha.html#applyinfo>**



# CUBRID HA/복제 – HA Fail-Over

## Master node 중지

```
[cubrid@DB01 ~]$ cubrid hb stop (Fail-Over 수행)
@ cubrid heartbeat stop
++ cubrid heartbeat stop: success
```

## Slave node 확인

```
[cubrid@DB02 databases]$ cubrid hb status
@ cubrid heartbeat status

HA-Node Info (current DB02, state master)
  Node DB02 (priority 2, state master)
  Node DB01 (priority 1, state unknown)

HA-Process Info (master 7854, state master)
  Applylogdb testdb@localhost:/home/cubrid/CUBRID/databases/testdb_DB01 (pid 8093, state registered)
  Copylogdb testdb@DB01:/home/cubrid/CUBRID/databases/testdb_DB01 (pid 8091, state registered)
  Server testdb (pid 7862, state registered_and_active)
```

# CUBRID HA/복제 – HA Fail-Over

## Master node 기동

```
[cubrid@DB01 ~]$ cubrid hb start
@ cubrid heartbeat start
@ HA processes start
@ cubrid server start: testdb
```

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

CUBRID 11.0

```
++ cubrid server start: success
@ copylogdb start
++ copylogdb start: success
@ applylogdb start
++ applylogdb start: success
++ HA processes start: success
++ cubrid heartbeat start: success
```

## Slave node 확인 (Fail-Over 확인)

```
[cubrid@DB02 databases]$ cubrid hb status
@ cubrid heartbeat status
```

```
HA-Node Info (current DB02, state master)
  Node DB02 (priority 2, state master)
  Node DB01 (priority 1, state slave)
```

```
HA-Process Info (master 7854, state master)
  Applylogdb testdb@localhost:/home/cubrid/CUBRID/databases/testdb_DB01 (pid 8093, state registered)
  Copylogdb testdb@DB01:/home/cubrid/CUBRID/databases/testdb_DB01 (pid 8091, state registered)
  Server testdb (pid 7862, state registered_and_active)
```

# CUBRID HA/복제 – HA Fail-Back

## Slave node 중지 (Fail-Back 수행)

```
[cubrid@DB02 databases]$ cubrid hb stop
@ cubrid heartbeat stop
++ cubrid heartbeat stop: success
```

## Master node 확인

```
[cubrid@DB01 ~]$ cubrid hb status
@ cubrid heartbeat status

HA-Node Info (current DB01, state master)
Node DB02 (priority 2, state unknown)
Node DB01 (priority 1, state master)

HA-Process Info (master 60606, state master)
Applylogdb testdb@localhost:/home/cubrid/CUBRID/databases/testdb_DB02 (pid 61105, state registered)
Copylogdb testdb@DB02:/home/cubrid/CUBRID/databases/testdb_DB02 (pid 61103, state registered)
Server testdb (pid 60874, state registered_and_active)
```

# CUBRID HA/복제 – HA Fail-Back

## Slave node 기동

```
[cubrid@DB02 databases]$ cubrid hb start
@ cubrid heartbeat start
@ HA processes start
@ cubrid server start: testdb
```

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

CUBRID 11.0

```
++ cubrid server start: success
@ copylogdb start
++ copylogdb start: success
@ applylogdb start
++ applylogdb start: success
++ HA processes start: success
++ cubrid heartbeat start: success
```

## Master node 확인 (Fail-Back 완료)

```
[cubrid@DB01 ~]$ cubrid hb status
@ cubrid heartbeat status
```

```
HA-Node Info (current DB01, state master)
  Node DB02 (priority 2, state slave)
  Node DB01 (priority 1, state master)
```

```
HA-Process Info (master 60606, state master)
  Applylogdb testdb@localhost:/home/cubrid/CUBRID/databases/testdb_DB02 (pid 61105, state registered)
  Copylogdb testdb@DB02:/home/cubrid/CUBRID/databases/testdb_DB02 (pid 61103, state registered)
  Server testdb (pid 60874, state registered_and_active)
```

# CUBRID HA/복제 – 복제

CUBRID 복제(Replica node)는 Master / Slave node 의 데이터를 동기화 하여 동작 합니다.

Replica node는 어떠한 상황에서도 자신의 Transaction log를 Master / Slave node에 전달 하지 않습니다.

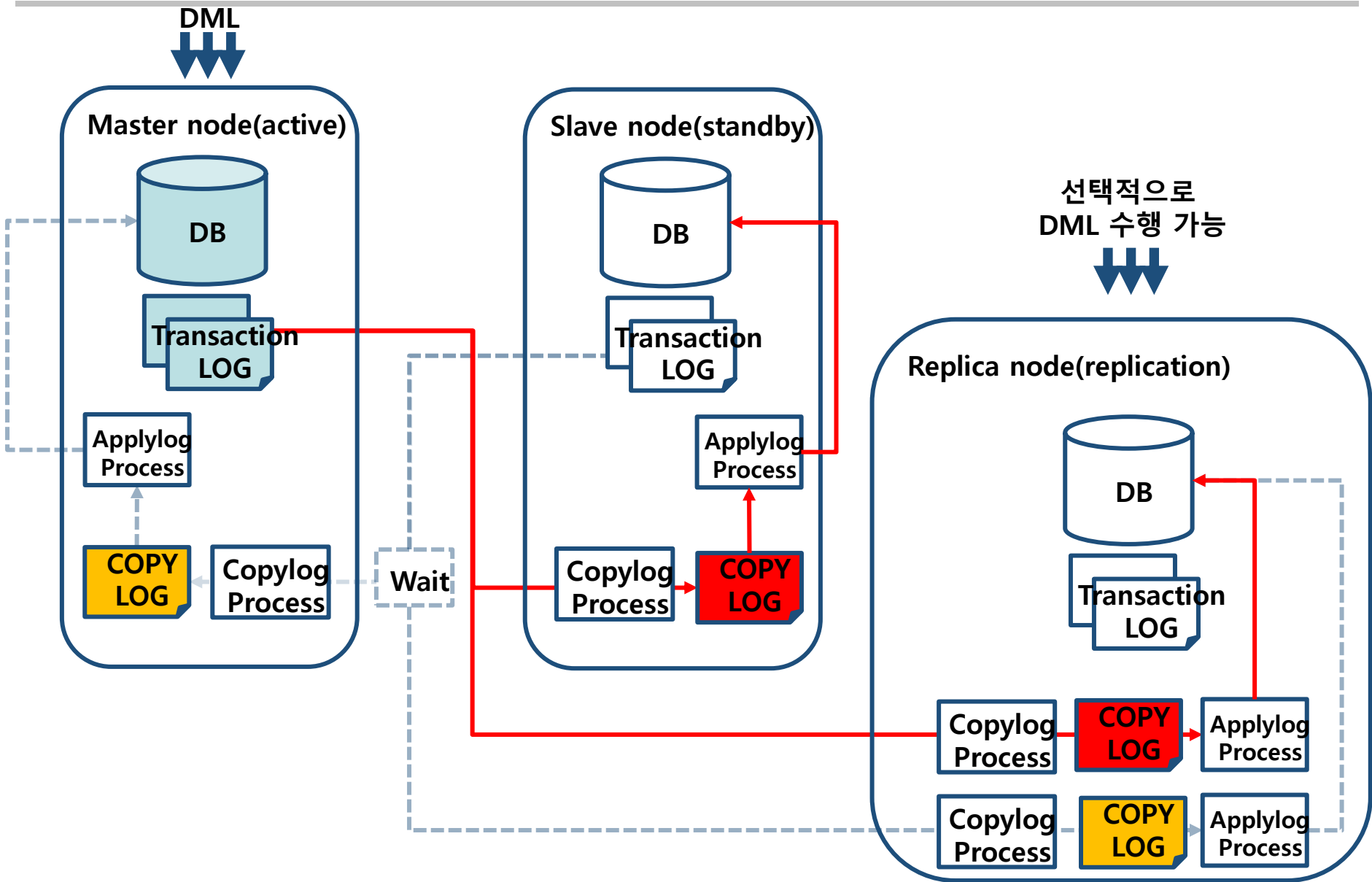
Replica node는 CUBRID 브로커 설정 값을 조정하여 선택적으로 Write를 수행하게 할 수 있습니다.

CUBRID HA - Master node : Slave node : Replica node 동기화 구성

1. Master node DML(SELECT 제외)이 발생하여 트랜잭션 로그에 기록 합니다.
2. Slave node의 Copylogdb 프로세스가 Master node의 트랜잭션 로그를 Slave node로 가져 옵니다.
3. Slave node의 applylogdb 프로세스는 Copylogdb 프로세스가 가지고 온 트랜잭션 로그를 SQL 형태로 변경하여, Slave node의 DB에 반영 합니다.
4. Replica node의 Copylogdb 프로세스가 Master node의 트랜잭션 로그를 Slave node로 가져 옵니다.
5. Replica node의 applylogdb 프로세스는 Copylogdb 프로세스가 가지고 온 트랜잭션 로그를 SQL 형태로 변경하여, Slave node의 DB에 반영 합니다.

- Fail-Over 발생 시 위 절차의 Master node와 Slave node가 반대로 동작한다고 보면 됩니다.
- Replica node는 Master node를 데이터 반영 node에서 Slave node를 데이터 반영 node로 변경합니다.

# CUBRID HA/복제 - 복제



# CUBRID HA/복제 – 복제 구성

## Slave node 데이터 파일 구성

```
[cubrid@DB01 ~]$ cubrid service stop
[cubrid@DB01 ~]$ rm -rf $CUBRID_DATABASE/testdb_DB02
[cubrid@DB01 ~]$ csql -u dba testdb -S --sysadm -c "DELETE FROM db_ha_apply_info;"
```

```
[cubrid@DB01 ~]$ cd $CUBRID_DATABASE
[cubrid@DB01 databases]$ tar -cvzf testdb.tar.gz testdb
```

```
# sftp 또는 scp로 DB02으로 testdb.tar.gz 전송
# sftp 또는 scp로 DB02으로 database.txt 전송
```

```
[cubrid@DB02 ~]$ cd $CUBRID_DATABASE
[cubrid@DB02 databases]$ ls
databases.txt testdb.tar.gz
[cubrid@DB02 databases]$ tar -xvzf testdb.tar.gz
[cubrid@DB02 databases]$ ls
databases.txt testdb.tar.gz testdb
```

# CUBRID HA/복제 – 복제 구성

## Replica node 데이터 파일 구성

```
[cubrid@DB02 ~]$ cubrid service stop
[cubrid@DB02 ~]$ rm -rf $CUBRID_DATABASE/testdb_DB01
```

```
[cubrid@DB01 ~]$ cd $CUBRID_DATABASE
[cubrid@DB01 databases]$ tar -cvzf testdb.tar.gz testdb
```

```
# sftp 또는 scp로 DB03으로 testdb.tar.gz 전송
# sftp 또는 scp로 DB03으로 database.txt 전송
```

```
[cubrid@DB03 ~]$ cd $CUBRID_DATABASE
[cubrid@DB03 databases]$ ls
databases.txt testdb.tar.gz
[cubrid@DB03 databases]$ tar -xvzf testdb.tar.gz
[cubrid@DB03 databases]$ ls
databases.txt testdb.tar.gz testdb
```



# CUBRID HA/복제 – 복제 구성

## Master / Slave / Replica node 설정

```
$ cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.152.101 DB01 # Master   node
192.168.152.102 DB02 # Slave    node
192.168.152.103 DB02 # Replica  node
```

```
$ cat $CUBRID/conf/cubrid_ha.conf
... 생략
#[common]
ha_port_id=59901
ha_node_list=cubrid@DB01:DB02
ha_replica_list=cubrid@DB03
ha_db_list=testdb
ha_apply_max_mem_size=300
ha_copy_sync_mode=sync:sync
ha_copy_log_max_archives=1
```

## Master / Slave node 설정

```
$ cat $CUBRID/conf/cubrid.conf
service=heartbeat,broker,manager
... 생략
## HA
ha_mode=on
force_remove_log_archives=no
log_max_archives=20
```

# CUBRID HA/복제 – 복제 구성

## Replica node 설정

```
$ cat $CUBRID/conf/cubrid.conf
service=heartbeat,broker,manager
... 생략
## HA
ha_mode=replica
force_remove_log_archives=no
log_max_archives=20
```

```
$ cat $CUBRID_DATABASES/databases.txt
##db-name      vol-path      db-host      log-path      lob-base-path
testdb         /home/cubrid/CUBRID/databases/testdb  DB03         /home/cubrid/CUBRID/databases/testdb
```

file:/h

# CUBRID HA/복제 – 복제 구성

## Master node 기동/확인

```
[cubrid@DB01 databases]$ cubrid hb start
@ cubrid heartbeat start
@ cubrid master start
++ cubrid master start: success
@ HA processes start
@ cubrid server start: testdb
```

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

CUBRID 11.0

```
++ cubrid server start: success
@ copylogdb start
++ copylogdb start: success
@ applylogdb start
++ applylogdb start: success
++ HA processes start: success
++ cubrid heartbeat start: success
```

```
[cubrid@DB01 ~]$ cubrid hb status
@ cubrid heartbeat status
```

**HA-Node Info (current DB01, state master)**

Node DB03 (priority 32767, state replica)

Node DB02 (priority 2, state slave)

**Node DB01 (priority 1, state master)**

**HA-Process Info (master 61216, state master)**

Applylogdb testdb@localhost:/home/cubrid/CUBRID/databases/testdb\_DB02 (pid 61455, state registered)

Copylogdb testdb@DB02:/home/cubrid/CUBRID/databases/testdb\_DB02 (pid 61453, state registered)

Server testdb (pid 61224, state registered\_and\_active)

# CUBRID HA/복제 – 복제 구성

## Slave node 기동/확인

```
[cubrid@DB02 databases]$ cubrid hb start
@ cubrid heartbeat start
@ cubrid master start
++ cubrid master start: success
@ HA processes start
@ cubrid server start: testdb
```

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

CUBRID 11.0

```
++ cubrid server start: success
@ copylogdb start
++ copylogdb start: success
@ applylogdb start
++ applylogdb start: success
++ HA processes start: success
++ cubrid heartbeat start: success
```

```
[cubrid@DB02 conf]$ cubrid hb status
@ cubrid heartbeat status
```

**HA-Node Info (current DB02, state slave)**

Node DB03 (priority 32767, state replica)

**Node DB02 (priority 2, state slave)**

Node DB01 (priority 1, state master)

**HA-Process Info (master 1787, state slave)**

Applylogdb testdb@localhost:/home/cubrid/CUBRID/databases/testdb\_DB01 (pid 2030, state registered)

Copylogdb testdb@DB01:/home/cubrid/CUBRID/databases/testdb\_DB01 (pid 2028, state registered)

Server testdb (pid 1795, state registered\_and\_**standby**)

# CUBRID HA/복제 – 복제 구성

## Replica node 기동/확인

```
[cubrid@DB03 conf]$ cubrid hb start
@ cubrid heartbeat start
@ cubrid master start
++ cubrid master start: success
@ HA processes start
@ cubrid server start: testdb
```

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

CUBRID 11.0

```
++ cubrid server start: success
@ copylogdb start
++ copylogdb start: success
@ applylogdb start
++ applylogdb start: success
++ HA processes start: success
++ cubrid heartbeat start: success
```

```
[cubrid@DB03 conf]$ cubrid hb status
@ cubrid heartbeat status
```

```
HA-Node Info (current DB03, state replica)
Node DB03 (priority 32767, state replica)
Node DB02 (priority 2, state slave)
Node DB01 (priority 1, state master)
```

```
HA-Process Info (master 1746, state slave)
Applylogdb testdb@localhost:/home/cubrid/CUBRID/databases/testdb_DB02 (pid 1996, state registered)
Applylogdb testdb@localhost:/home/cubrid/CUBRID/databases/testdb_DB01 (pid 1994, state registered)
Copylogdb testdb@DB02:/home/cubrid/CUBRID/databases/testdb_DB02 (pid 1992, state registered)
Copylogdb testdb@DB01:/home/cubrid/CUBRID/databases/testdb_DB01 (pid 1990, state registered)
Server testdb (pid 1754, state registered_and_standby)
```

# CUBRID HA/복제 – 구성

## Replica node 동기화 확인

```
[cubrid@DB01 ~]$ csql -u dba testdb@DB01 -c "INSERT INTO junsu(x) VALUES(100);"  
1 row affected. (0.002552 sec) Committed.
```

```
[cubrid@DB01 ~]$ csql -u dba testdb@DB01 -c "SELECT * FROM junsu;"
```

```
=== <Result of SELECT Command in Line 1> ===
```

```
      x  
=====
```

1
100

```
2 rows selected. (0.002503 sec) Committed.
```

```
[cubrid@DB01 ~]$ csql -u dba testdb@DB02 -c "SELECT * FROM junsu;"
```

```
=== <Result of SELECT Command in Line 1> ===
```

```
      x  
=====
```

1
100

```
2 rows selected. (0.003211 sec) Committed.
```

```
[cubrid@DB01 ~]$ csql -u dba testdb@DB03 -c "SELECT * FROM junsu;"
```

```
=== <Result of SELECT Command in Line 1> ===
```

```
      x  
=====
```

1
100

```
2 rows selected. (0.007048 sec) Committed.
```

# CUBRID HA/복제 – 선택적 쓰기 구성

## Replica node 접속 브로커 설정 변경

```
[cubrid@DB03 ~]$ cat $CUBRID/conf/cubrid_broker.conf
... 생략
[%query_editor]
SERVICE                =ON
SSL                      =OFF
BROKER_PORT              =30000
MIN_NUM_APPL_SERVER     =5
MAX_NUM_APPL_SERVER     =40
APPL_SERVER_SHM_ID      =30000
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
CCI_DEFAULT_AUTOCOMMIT  =ON
REPLICA_ONLY=ON
ACCESS_MODE=RW
... 생략 ...

[cubrid@DB03 conf]$ cubrid broker start
@ cubrid broker start
++ cubrid broker start: success
```

# CUBRID HA/복제 – 선택적 쓰기 구성

Replica node의 선택적 쓰기 구성은 브로커를 통해서만 가능

- DDL 및 DML 수행

DBeaver 21.3.5 - <testdb\_Replica> Script-6

파일(F) 편집(E) 탐색(N) 검색(A) SQL 편집기 데이터베이스(D) 윈도우(W) 도움말(H)

SQL 커밋 롤백 Auto testdb\_Replica < N/A >

Database N... 프로젝트

Enter a part of object name here

▼ CUBRID

- > testdb - 192.168.152.101:33000
- ▼ testdb\_Replica - 192.168.152.103:3000
  - > 테이블
  - > 뷰
  - > 인덱스
  - > 프로시저
  - > 데이터 타입
- > ORACLE
- > PostgreSQL
- > TIBERO

```
create table junsu_replication(x int primary key);
insert into junsu_replication(x) values(200);
commit;
```

Statistics 1

commit Enter a SQL expression to filter results (use Ctrl+Space)

Name	Value
Updated Rows	0
Query	commit
Finish time	Mon Feb 28 16:14:50 KST 2022



# CUBRID HA/복제 – 선택적 쓰기 구성

Replica node에 수행 된 DDL 및 DML Master / Slave node에서 확인

- Replica node 모든 Transaction Log는 Master / Slave node에서 반영 하지 않습니다.

```
[cubrid@DB01 ~]$ csql -u dba testdb@DB01 -c "SELECT * FROM junsu_replication;"

In line 1, column 1,

ERROR: before ' ;'
Unknown class "junsu_replication".

[cubrid@DB01 ~]$ csql -u dba testdb@DB02 -c "SELECT * FROM junsu_replication;"

In line 1, column 1,

ERROR: before ' ;'
Unknown class "junsu_replication".

[cubrid@DB01 ~]$ csql -u dba testdb@DB03 -c "SELECT * FROM junsu_replication;"

=== <Result of SELECT Command in Line 1> ===

          X
=====
        200

1 row selected. (0.002798 sec) Committed.
```



 Thank you!