

CUBRID 전환 가이드

from ORACLE

목차

1. 개요	3
1.1 목적	3
1.2 사용환경	3
2. 특징점	3
3. 데이터베이스 생성	4
3.1 데이터베이스 이름	4
3.2 데이터베이스 생성	4
4. 데이터베이스 이전	4
4.1 사용법	5
5. 스키마	10
5.1 예약어	10
5.2 타입	10
5.3 제약조건	10
6. 질의	11
7. 연산자와 함수	12
7.1 연산자	12
7.2 함수	12
7.3 일련번호	13
8. Stored Procedure(Procedure & Function)	13
8.1 환경설정	14
8.2 작성 방법	14
8.3 프로시저/펄션 작성	14
9. 성능	15
9.1 힌트	15
9.2 rownum	16

9.3	정렬	16
9.4	max, min	16
9.5	covering index	17
9.6	질의 수행 계획	17
10.	응용 interface	17
10.1	연결 포트	17
10.2	JDBC	18
10.3	PHP	18
11.	HA 구성시 고려사항	19

1. 개요

1.1 목적

ORACLE 사용자가 CUBRID 로의 이전을 보다 쉽게 하기위해 만들어졌으며, ORACLE 과 CUBRID 사용시 차이점을 주로 정리하였으며, 일반적으로 사용되는 함수들중에서 차이가 있는 부분에 대하여 정리하여, 개발자들의 불편을 최소화할 수 있도록 하였습니다. 또한 함수의 경우, 모든 부분을 다루기는 어려우므로 관련 함수의 지원 여부에 대하여는 매뉴얼을 참조하시면 됩니다.

이하 문서의 편의를 위해 존칭은 생략한다.

1.2 사용환경

- CUBRID : 9.0 이상 (2008 R4.4 일부 포함. 기능에 대하여 미지원 여부 표기)
- JAVA : 1.6 이상 (Stored Procedure 사용시, 데이터베이스 서버 환경, 응용환경아님)

2. 특징점

CUBRID 는 보편적인 RDBMS 구조를 가지고 있으며, 지원되는 주요 기능들에 대하여 정리하였다.

구분	기능
SQL	ANSI SQL(SQL-92 기준, SQL-99/2003 호환)
용량	테이블 개수, 레코드 건수 제약 없음
data type	char, varchar, int, numeric(number), etc LOB(CLOB, BLOB)
charset	문자단위의 문자 처리(length, substring 등) EUC-KR, UTF8 등 지원 * 9.0 아래 버전은 byte 단위 문자 처리
transaction	record based locking commit/rollback/savepoint 별도의 트랜잭션 시작 구문없이, 임의의 질의 수행시 트랜잭션 시작됨 auto commit 은 언어별 설정에 따름
Backup	on-line(hot)/off-line(cold) backup Full/Incremental backup Time based recovery Full recovery (반드시 백업이 존재하여야, 복구를 할 수 있다)
Miscellaneous	Partition data encryption

	Stored Procedure(JAVA)
HA	자체 지원 별개의 데이터베이스(shared nothing)로 스토리지 장애에 유연함
Sharding	scale out 형태의 데이터 분산(수평분할)
API	JDBC, ODBC, PHP, .NET, C-API, Ruby, Python, etc
미지원	DB link, Temporary Table, Parallel Query, Materialized View, Synonym

3. 데이터베이스 생성

3.1 데이터베이스 이름

- 대소문자 구분하므로, 생성시 대소문자에 신경을 써야한다. 응용에서 연결시에도 대소문자를 구분한다.

3.2 데이터베이스 생성

- 1 개의 데이터베이스에 1 개의 인스턴스 만을 가진다. 따라서 여러 개의 인스턴스가 필요하다면 여러 개의 데이터베이스를 생성하여야 한다.
- 생성후 바로 백업을 하는 것이 좋다. 백업이 있어야만 원하는 시점으로 복구를 할 수가 있기 때문이다.
- 테이블스페이스는 자동 설정되므로, 별도로 지정할 필요없다.
- 필요한 데이터 양을 산정 후, 실제 데이터베이스의 크기는 x1.5 를 해주면 된다.
- 데이터,인덱스,질의처리공간(temp) 별도 생성해주어야 한다. 인덱스 공간도 산정을 통해 계산하는 것이 좋으며, 그렇지 못한 경우 데이터의 20~30%정도로 산정할 수 있다. 질의 처리 공간은 통상적으로 데이터의 10% 수준으로 생성한다.

```

예) 데이터베이스이름: mydb, 데이터 2G, 인덱스 1G 의 경우, /DB 아래 생성
cubrid created -F /DB mydb
cubrid addvoldb -S -p data --db-volume-size=2G mydb
cubrid addvoldb -S -p index --db-volume-size=1G mydb
cubrid addvoldb -S -p temp --db-volume-size=200M mydb
  
```

4. 데이터베이스 이전

데이터베이스 이전은 CMT(CUBRID Migration Toolkit)을 이용하여 이전할 수 있다. 이는 1:1 이전(이전 전후의 스키마와 데이터가 동일)만이 가능하다. 또한 CMT 를 이용하여 전환 가능한 대상 데이터베이스는 ORACLE, MySQL, MS-SQL 이 가능하다.

업무에 맞게 변경하는 것은, 업무를 파악하고 업무에 맞게 조정되어야 하므로 이진후 별도의 과정을 통해 응용단에서 처리하여야 한다.

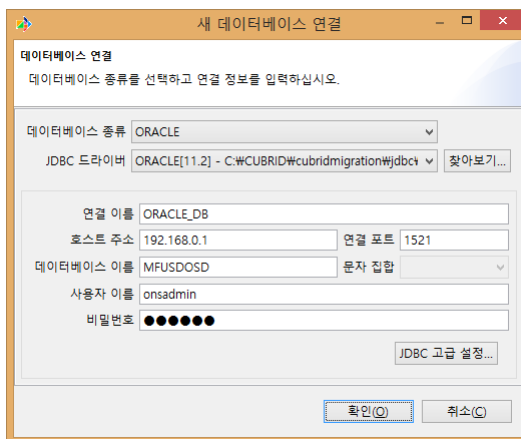
4.1 사용법

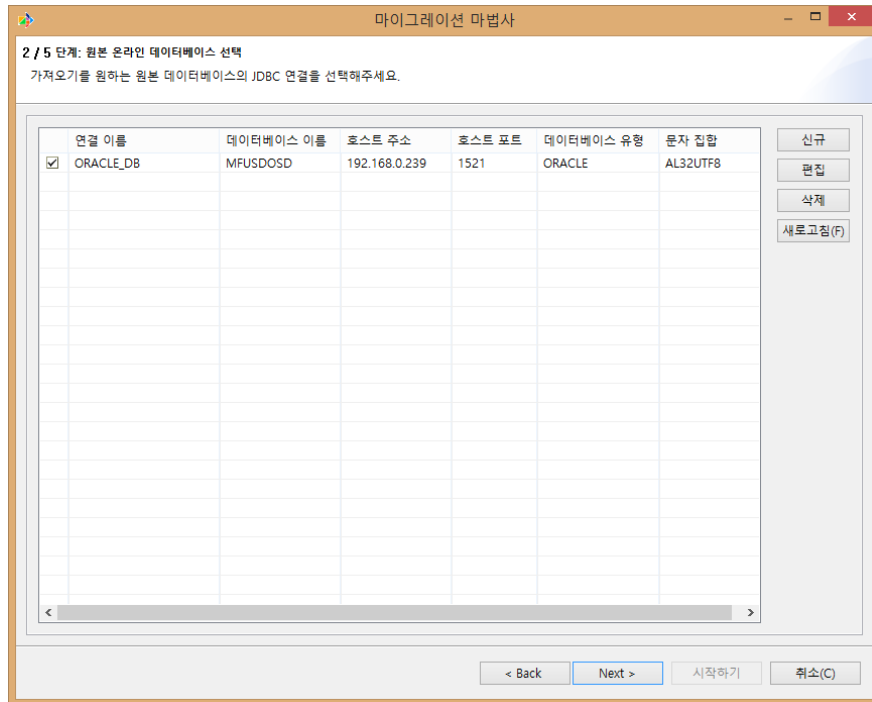
사용법은 비교적 간단하며, ftp.cubrid.org 에서 다운 받을 수 있다. 여기서는 기본 적인 사용법을 제시하며, 보다 자세한 내용은 매뉴얼을 참고하기 바란다.

1. 마이그레이션 유형 선택

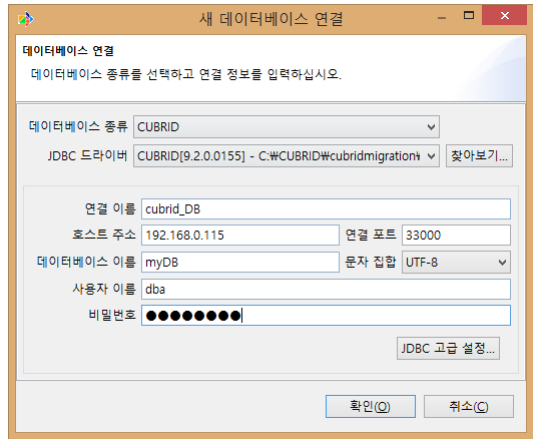


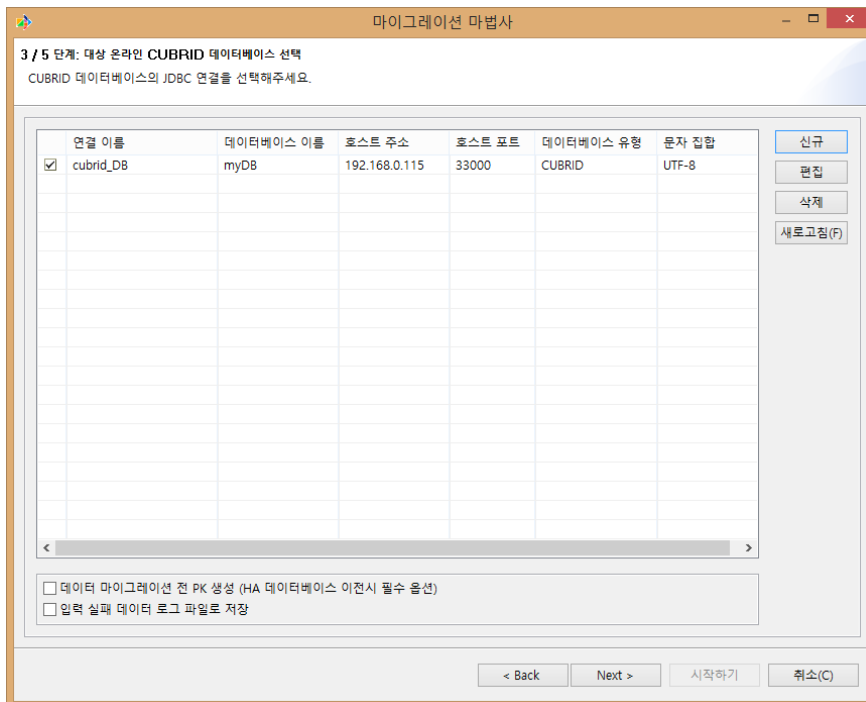
2. 소스 데이터베이스 연결 설정





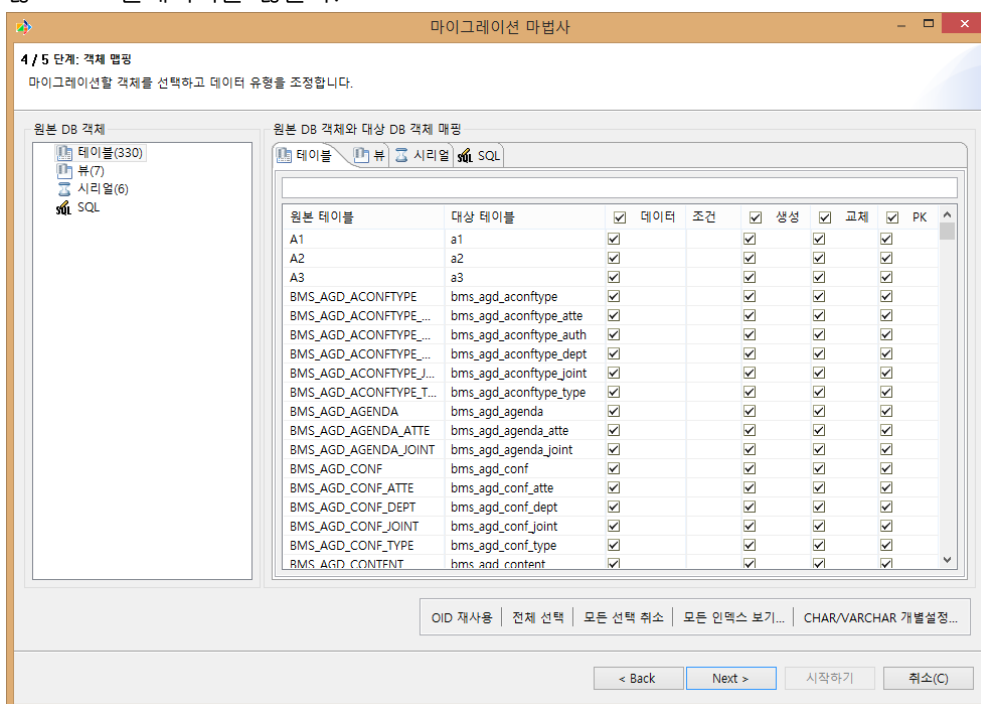
3. 대상 데이터베이스 연결 설정



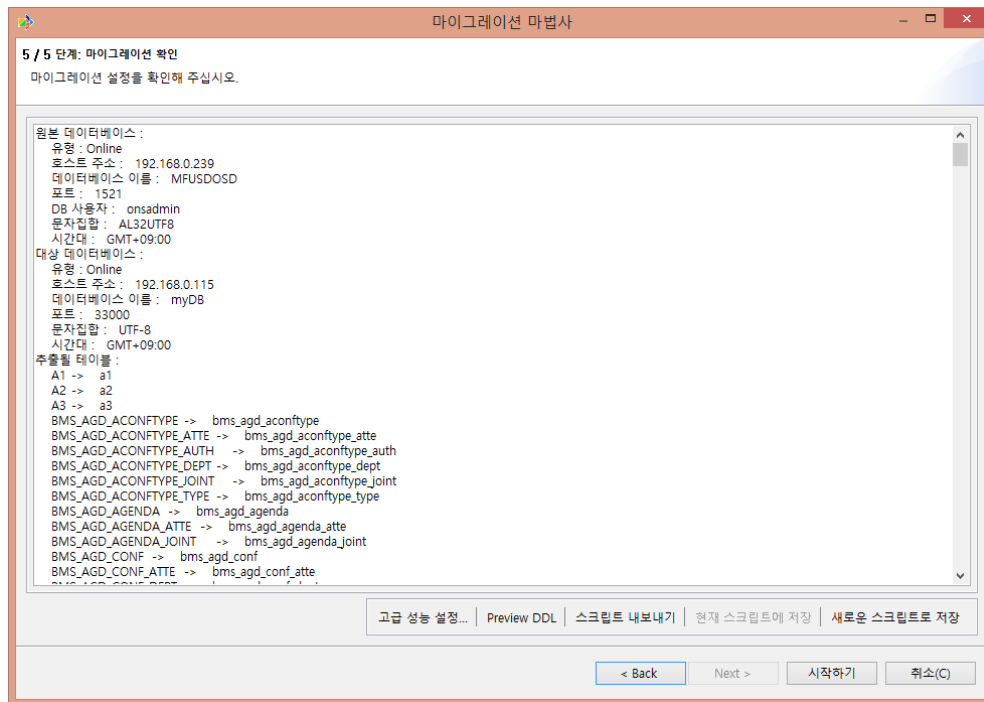


4. 객체 매핑

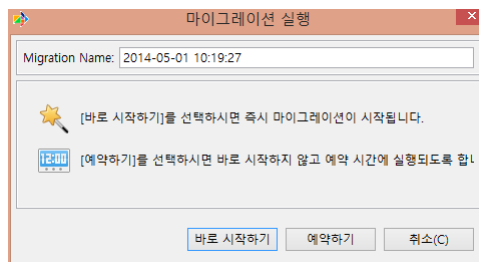
스키마 정보는 기본적으로 모두 소문자로 적용된다. 실제 질의에서 대소문자는 구분하지 않으므로 문제되지는 않는다.

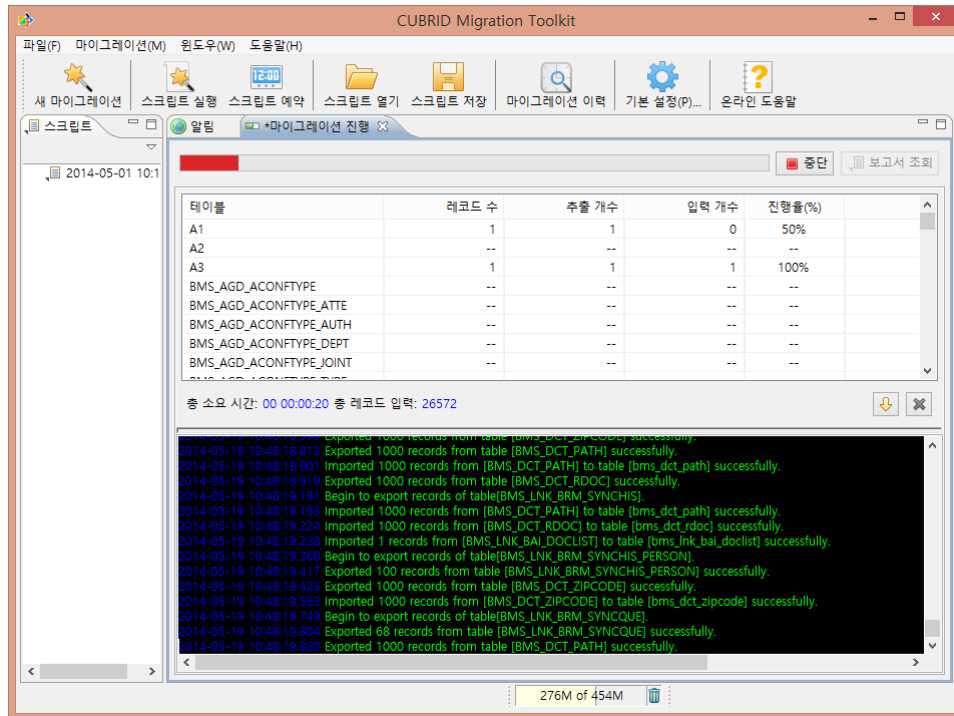


5. 내용 확인



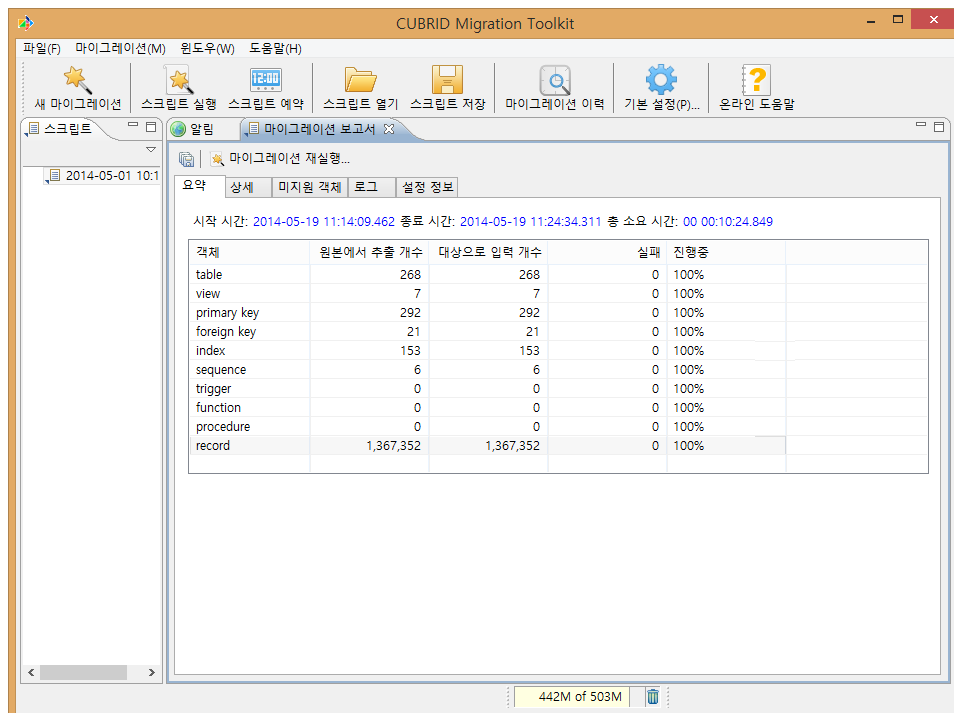
6. 마이그레이션 진행





7. 완료

완료후 보고서를 볼 수 있으며, 만약 에러가 발생한 경우 붉은 색으로 표시되며, 상세를 클릭해보면 에러 내용을 확인해 볼 수 있다.



5. 스키마

5.1 예약어

예약어는 일부 상이하며, 사용시 “” , ``(백틱, 작은따옴표가 아닌 ~키에 있는) 또는 [] 로 감싸서 사용하면 된다. 일반적으로 타입명, 함수명 등이 예약어가 된다. 자세한 예약어 리스트는 매뉴얼을 참고한다.

5.2 타입

대부분 유사하나, 명칭이 다르거나 지원되지 않는 부분에 대하여 사용 가능한 타입을 정리한다.

- NULL 과 " 는 서로 다르게 취급하므로, not null default " 을 이용하거나 하여야 한다.

ORACLE	CUBRID	비고
bit boolean	BIT	BIT 로 사용
number	smallint, int, bigint, float/real, double, numeric	smallint: 2byte int: 4byte bigint: 8byte numeric: 최대 38 자리(소수점포함) * unsigned 미지원
date	date time datetime	date: 날자 부분만 포함 time: 시간 부분만 포함 datetime: 날자, 시간 포함하며, 1/1000 초
lob	lob	외부에 파일로 저장. 내부에 저장하기 위해서는 clob→varchar, blob→bit varying 사용
char	char	고정길이이거나, 인덱스로 사용될 때 사용 고려
varchar, varchar2	varchar, string	string: varchar(1G)와 동일
long	varchar	CUBRID varchar 는 1G 까지 지원

5.3 제약조건

일반적인 제약조건들은 지원되며, 차이가 있는 부분들에 대하여 정리한다.

- Key : PK, FK 모두 지원하며, key 로 지정시 인덱스로도 같이 사용되므로 별도로 인덱스를 생성할 필요없다.
- Primary key : 모든 컬럼의 값이 not null 로 설정된다. 즉, 멀티 컬럼으로 PK 를 구성할 경우 어느 컬럼의 값도 null 이 허용되지 않는다.
- Foreign key : on update 에 대하여 cascade 를 지원하지 않는다.

- Check : 컬럼 속성중 check 속성은 지원하지 않는다.
- Constraint : key 이름 지정을 위한 constraint 는 컬럼 속성 뒤에 명시할 수 없고, 별도로 명시하여야 한다.

```
my_col int constraint pk primary key(my_col) : 지원않됨.
my_col int,
constraint pk primary key(my_col)
```

- 제약조건을 비활성화 할 수 없다.

6. 질의

질의 사용시 사용법이 다르거나, 지원되지 않는 부분에 대하여 정리한다.

ORACLE	CUBRID	비고
dual	db_root 또는 from 절 없이 사용	select sysdatetime
connect by	connect by	
with		지원않함
regexp_replace		지원않함
for update		지원않함. 응용에서 write lock 을 설정하는 형태로 변경
SELECT SUM(COUNT(col_1)) FROM tbl_1 GROUP BY col_1	SELECT SUM(col_2) FROM (SELECT COUNT(col_1) AS col_2 FROM tbl_1 GROUP BY col_1)	집계함수 중복 사용 불가
from a, b where a.i = b.i(+)	from a left outer join b on a.i = b.i	outer join 시 ANSI 표준 사용 권장
from a full outer join b on a.i = b.i		full outer join 지원하지 않음
case when exists (select 1 from ...)	case when 1=(select count(*) from ...	case 문에서 exist 는 지원하지 않음.

order by col nulls first	9.2 부터 nulls first/last 지원	CUBRID 에서 null 은 제일 작은 값이다. 9.1 이하에서는 order by isnull(col), col
--------------------------	----------------------------	--

7. 연산자와 함수

7.1 연산자

ORACLE	CUBRID	비고
=	= <=>	<=>: null 과 비교시 is null 을 사용하지 않아도 가능
!= ◇ ^=	!= ◇	
 concat	 concat +	문자열 합치기
minus	Difference	
Intersect	Intersection	
1/2 = 0.5	1/2 = 0	정수 연산시 둘다 정수이므로, 결과를 정수로 넘겨줌

- datetime 연산시 숫자는 최소 단위인 1/1000 초로 계산

7.2 함수

함수는 종류가 많아 모두 다 열거하기는 어렵다. 일반적으로 많이 사용되는 함수를 기반으로 정리하였으며, 지원되는 모든 함수의 목록은 매뉴얼을 참고한다.

ORACLE	CUBRID	비고
sysdate	sysdate systime sysdatetime	사용되어 지는 데이터 타입에 맞게 사용
to_date	to_date to_time to_datetime	사용되어 지는 데이터 타입에 맞게 사용
instr	instr position	instr() 의 경우 3 개의 아규먼트만 지원한다.

dbms_random.value	rand drand random drandom	d 가 붙은 것은 0~1 사이의 실수를 넘겨준다. rand() 는 질의 기반으로 난수를 만들어주며, random() 은 질의 결과 레코드 기반으로 난수를 만든다.
sys_guid		to_char(random(), '0000000000') + to_char(sys_datetime, 'YYYYMMDDHH24MISSFF') + to_char(random()%10000, '00000')
sys_extract_utc		지원하지 않음
regexp_replace		지원하지 않음
xml 함수		지원하지 않음
over	over 함수	9.1 부터 지원되며, window 절은 현재 지원되지 않음.

7.3 일련번호

ORACLE	CUBRID	비고
sequence	serial	
	auto increment	데이터 입력시 자동증가 속성을 부여하면, 해당 컬럼의 값이 자동으로 주어진 값만큼 증가 create table my_tbl (id int auto_increment, name char(10)) insert into my_tbl(name) values('name1') → id 값은 자동 부여되며, 순차 증가

8. Stored Procedure(Procedure & Function)

Stored procedure 구현을 위해 사용되는 언어는 java 이다. 따라서 PG-SQL 구문을 java 구문으로 변경하여 사용한다.

배치 작업을 위한 procedure 에서의 사용은 무난하지만, function 등의 사용은 권장하지 않는다.

8.1 환경설정

아래 2 개의 환경변수가 설정되어 있어야 하며, 적용을 위해서는 CUBRID 서비스를 재구동해야 반영된다.

```
JAVA_HOME=/usr/java/jdk1.6.0_10
LD_LIBRARY_PATH=$JAVA_HOME/jre/lib/amd64:$JAVA_HOME/jre/lib/amd64/server
```

8.2 작성 방법

- JDK1.6 이상
- 프로시저 내부에서 트랜잭션 명령(commit, rollback)을 사용할 수 없다. 사용하더라도 무시되므로, 트랜잭션 처리는 프로시저 외부에서 하여야 한다.

8.2.1 프로시저/평선 등록

- 컴파일된 java class 를 loadjava 를 이용하여 등록한다.

```
loadjava myDB mySP.class
```

- 프로시저를 등록한다.

```
create procedure myproceure (name varchar)
as language java
name 'mySP.myproceure(java.lang.String)';

create function myfunc(id int) return varchar
as language java
name 'mySP.myfunc(int) return int';
```

8.3 프로시저/평선 작성

java method 작성과 동일한 방법을 사용하면 된다. 굵은 글씨로 표시된 부분만 작성시 유의하면 된다. 프로시저/평선으로 넘어오는 인자값과 리턴값에 대한 처리는 일반 java method 와 동일하게 하면 된다.

다음은 평선 작성 예이다.

```
import java.sql.*;

public class mySP {
    public static void myfunc(int args) throws Exception {
        Connection conn = null;
        String ret_val = null;
        try {
            Class.forName("cubrid.jdbc.driver.CUBRIDDriver");
            conn = DriverManager.getConnection("jdbc:default:connection:");
```

```

        ...
        return ret_val;
    } catch ( SQLException e ) {
        System.err.println(e.getMessage());
    } catch ( Exception e ) {
        System.err.println(e.getMessage());
    } finally {
        if ( conn != null ) conn.close();
    }
}
}
}

```

9. 성능

성능에 관련하여 지원되는 기능 및 질의 작성시 성능을 위해 고려할 부분도 정리한다.

9.1 힌트

9.1.1 조인힌트

ORACLE 과 동일한 형태로 사용하며, 조인 대상 한정기 가능하며, 지원되는 구문은 다음과 같다.

- * USE_IDX : 일반적인 nested loop 조인시, 조인되어 지는 테이블의 조인 조건에 인덱스가 있는 경우 반드시 인덱스를 사용하여 조인을 한다.
- * USE_MERGE : sort merge join 을 사용한다.
- * USE_NL : nested loop join 을 사용한다.
- * ORDERED : from 절에 명시된 순서대로 join 을 수행한다.

9.1.2 인덱스힌트

ORACLE 과 달리 USING INDEX 라는 구문을 사용하며, 사용법은 다음과 같다.

```

Where ...
USING INDEX a.idx1, b.idx2(+)
Order by

```

- * 주어진 인덱스만 있는 것으로 간주하여, 주어진 인덱스와 풀스캔중 비용이 저렴한 것을 선택한다.
- * 조인시 특정 테이블에 대한 인덱스만 지정할 수 있으며, 이 경우 지정하지 않은 테이블은 옵티마이저가 선택한다.
- * 주어진 인덱스 명이 잘못된 경우 에러를 발생한다.

* (+) 를 이용하여 인덱스 사용 비용을 0 으로 만들 수 있다. 이는 해당 인덱스 사용을 강제하는 효과를 얻을 수 있다.

9.2 rownum

ORACLE 과 달리 rownum between 10 and 20 과 같은 형태로 사용할 수 있다. 다만 정렬전에 생성이 되어, 정렬과 함께 사용시 사용법이 좀 복잡해지므로 MySQL 형태의 limit 를 사용하면 훨씬 편리하다. 또한, inline view 를 사용하지 않아도 된다.

- ORACLE

```
select rn, * from (  
  select rownum rn, * from (  
    select * from tbl where ... order by ...  
  )  
  where rownum <= 20  
)  
where rn >= 11
```

- CUBRID

```
select orderby_num() rn, * from tbl  
where ... order by ... limit 11, 20
```

* 정렬후 정렬된 순번을 얻고자 한다면, orderby_num() 을 사용하면 된다.

9.3 정렬

정렬을 하는 경우, 정렬을 위한 비용이 상당하므로 정렬을 피하기 위해 정렬 대상과 검색조건을 인덱스로 생성하게 되면, CUBRID 옵티마이저가 해당 인덱스를 사용하여 정렬을 회피할 수 있다. 이를 통해 검색 성능을 올릴 수 있다.

다만, 모든 경우에 대하여 성능 향상이 이루어 지지 않을 수 있으므로, 반드시 질의 수행 계획 및 성능 개선 여부를 확인하여야 한다.

```
where id = 1 order by name desc  
  
create index idx1 on tbl(id, name desc)
```

9.4 max, min

max, min 대상을 인덱스로 생성시 인덱스를 이용하여 max, min 값을 얻을 수 있으므로 성능 향상을 꾀할 수 있다.

```
Select max(id) from tbl  
Create index idx1 on tbl(id)
```

9.5 covering index

검색에 필요한 정보가 인덱스에 모두 존재하는 경우, 인덱스 정보만으로 데이터를 확인할 수 있으므로 성능 향상을 얻을 수 있다. 특히, 코드에 대한 이름을 얻는 형태의 조인의 경우 많은 도움이 될 수 있다.

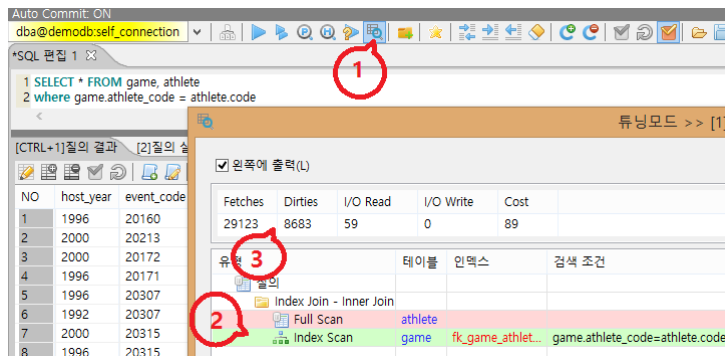
```
select name from tbl where id = 1

create index idx1 on tbl(id, name)
```

9.6 질의 수행 계획

질의 수행 계획을 보는 방법에 대하여 간단히 정리한다. 자세한 내용은 매뉴얼을 참고하기 바란다.

- ① 질의 수행 계획 및 통계 정보를 보기 위한 버튼.
- ② athlete table 에 대하여 full scan 을 하였으며, game table 에 대하여는 fk_game_athlete_code 를 이용하여 인덱스 스캔을 하였다는 의미
- ③ 질의 수행을 위해 발생한 I/O 양이 29123 페이지(CUBRID I/O 단위) 이며, 그중 최종 결과에는 사용되지 않는 페이지는 8683 페이지라는 의미. 결국 이 I/O 량을 줄이는 것이 제일 중요.



NO	host_year	event_code	Fetches	Dirties	I/O Read	I/O Write	Cost
1	1996	20160	29123	8683	59	0	89
2	2000	20213					
3	2000	20172					
4	1996	20171					
5	1996	20307					
6	1992	20307					
7	2000	20315					
8	1996	20315					

10. 응용 interface

응용 개발을 위한 driver 사용시 고려되어야할 부분을 정리한다.

10.1 연결 포트

CUBRID 응용에서는 데이터베이스와 연결을 할 때 반드시 포트를 지정해 주어야 한다. 기본적으로 제공되어 지는 포트는 33000 번이므로 이를 사용하면 된다.

만약 하나의 서버에 여러 개의 데이터베이스를 가지게 되면, 데이터베이스 마다 별개의 포트를 사용하는 것을 권장한다.

CUBRID 는 3-tier 구조를 가지며, broker 라는 미들웨어에서 실제 데이터베이스와 연결하여 작업을 수행한다. broker 에서는 데이터베이스와의 연결에 대하여 pooling 기능을 가지고 있으므로, 이 기능을 제대로 활용하기 위해 broker 별로 데이터베이스 연결을 따로 가지도록 하는 것이다.

응용에서는 이 broker 와 연결을 하여 작업을 처리하게 되므로, 데이터베이스 별로 별개의 broker 를 구성하고 그 broker 에 주어진 포트를 사용하면 된다. broker 를 추가하는 방법은 별도로 매뉴얼을 참고하면 된다.

10.2 JDBC

JAVA 1.6 이상을 지원하며, 일반적인 작성 방법은 JDBC 표준을 따른다.

연결 문자열은 다음과 같으며, charset 을 지정해주어야 문자셋이 깨지지 않고 저장될 수 있다.

- HA 환경에서는 url_ha 형태와 같이, althosts 를 stand-by 서버로 지정해주어야 한다.

```
import java.sql.*;
import cubrid.jdbc.driver.*;
String url = "jdbc:cubrid:192.168.0.100:33000:demodb:::charset=utf8";
String url_ha = "jdbc:cubrid:192.168.0.100:33000:demodb:::?
althosts=192.168.0.101:33000& charset=utf8";

Class.forName("cubrid.jdbc.driver.CUBRIDDriver");
Connection conn = DriverManager.getConnection(url, "db_user", "dbpw");
```

10.3 PHP

PHP 의 경우 함수에 대한 표준이 없다. 다만 대부분의 함수들이 prefix 부분을 제외하고는 유사하므로, 기존 사용함수의 prefix 부분만 cubrid 로 변경하여 검색해보면 된다.

HA 를 사용할 경우, url 형식의 연결 문자열(JDBC 참고)을 사용하는 cubrid_connect_with_url 이라는 함수를 사용하여야 하며, 사용방법은 다음과 같다. HA 를 사용하지 않더라도 해당 함수는 사용 가능(althosts 부분만 제거)하다.

```
$url_ha = "jdbc:cubrid:192.168.0.100:33000:demodb:::?
althosts=192.168.0.101:33000& charset=utf8";

$con = cubrid_connect_with_url(url, "db_user", "dbpw");
```

- 기본적으로 자동 커밋모드로 동작한다. 자동 커밋을 끄기 위해서는 cubrid_set_autocommit(\$con, CUBRID_AUTOCOMMIT_FALSE) 를 이용하거나, 연결문자열 뒷부분에 autocommit=false 를 추가해주면 된다.

11. HA 구성시 고려사항

장애 대비를 위해 자체적으로 지원되고 있는 HA 를 사용시 주의할 사항을 정리한다.

주의사항	비고
플랫폼	LINUX, AIX 에서만 사용가능하며, 동일한 플랫폼끼리만 HA 구성이 가능하다.
Primary key	스토리지 장애에도 대처하기위해, 스토리지를 공유하지 않은 방식을 사용한다. 따라서 서로 다른 별개의 데이터베이스를 복제를 통해 데이터 동기를 유지하므로, 반드시 PK 가 존재하여야 한다.
trigger	9.2 부터 사용 가능
lob	lob 데이터가 별도의 파일로 저장되며, 이 파일이 복제되지 않으므로 사용할 수 없으며, bit varying 을 사용하여야 한다.