CUBRID과 MySQL 비교

–

작성년월일: 2009년 3월

목차

1. 일반 기능 비교

1.1 CUBRID 일반 기능 비교

범주	CUBRID	MySQL
테이블 내 속성 수	제한 없음	~3398
레코드 크기	제한 없음	~65534 (BLOB,TEXT별도)
BLOB형식 저장소	8바이트의 OID 및 다른 데이터 페이지 또는 외부객체로 저장된 데이터	다른 데이터 페이지에 저장된 데이터
클러스터되지 않은 테이블 인덱스	제한 없음	~32 (256 byte)
단일 인덱스의 최대 인덱스 속성 수	제한 없음	~16
테이블 명명 규칙	[*소유자*]테이블_이름	테이블_이름
뷰 명명 규칙	[*소유자*]뷰_이름	뷰_이름
인덱스 명명 규칙	[*소유자*]인덱스_이름	인덱스_이름

1.2 CUBRID 일반 기능 비교

범주	CUBRID	MySQL
데이터베이스 이름	도메인 내에서 고유.	도메인 내에서 고유.
데이터베이스 이름 길이	최대 17바이트	64 바이트
테이블 이름	데이터베이스 안에서 고유. 대소문자 구분 X.	데이터베이스 사용자 계정 안에서 고유.
테이블 이름 제한	반드시 문자로 시작. 공백X _, %, #을 포함할 수 있음.	공백X, "₩", "/", "." 문자를 포함할 수 없음
속성 이름	테이블과 뷰 안에서 고유.	테이블과 뷰 안에서 고유.
인덱스 이름	사용자 스키마 안에서 고유.	데이터베이스 테이블 이름 안에서 고유.
예약어/식별자 사용	"" 나 []로 감싸면 사용 가능.	backtick("`")이나 ""으로 감싸면 사용 가능

2. 데이터 타입 비교

2.1 CUBRID 데이터 타입-스트링

데이터 형식		설명	크기
Character Strings	CHAR(n)	고정 길이 문자를 지정. n 디폴트 값은 1.	Fixed
	VARCHAR(n)	가변 길이 문자를 지정. n 디폴트 값은 1,073,741,823.	Variable
	NCHAR(n)	고정 길이 national character string을 지정. NCHAR(n) 데이터 타입에 의해 유지되는 string은 지원되는 Character Set 중 하나임.	Fixed
	NCHAR VARYING(n)	가변 길이 national character string을 지정.	Variable
Bit Strings	BIT(n)	바이너리나 16진수 포맷의 고정길이 bit string을 지정. n 디폴트 값은 1.	Fixed
	BIT VARYING(n)	가변 길이 bit string을 지정. n은 그 string에 허용된 비트의 최대 수를 표현. n 디폴트 값은 1,073,741,823.	Variable

2.2 CUBRID 데이터 타입-수치형

데이터 형식		설명	크기
Numeric	SMALLINT	16 Bit 정수. UNSIGNED 표현은 지원되지 않음	2
	SHORT	SMALLINT와 동일	2
	INT	32 Bit 정수 UNSIGNED 표현은 지원되지 않음	4
	BIGINT	64 Bit 정수 UNSIGNED 표현은 지원되지 않음	8
	NUMERIC(p,[s])	정밀도(p)와 범위(s)는 명세 가능. 1<=정밀도(p)<=38임. p의 디폴트 값은 15, s는 0.	16
	DECIMAL	NUMERIC과 동일	16

2.2 CUBRID 데이터 타입-수치형

데이터 형식		설명	크기
	FLOAT(p)	정밀도(p)<=7이면 단일 정밀도(32 Bit) 부동 소수로서 표현됨. 8<=정밀도(p)<=19이면 DOUBLE 타입으로 취급됨. 범위는 –10e+38 에서 +10e+38까지 지정 가능.	4
	REAL	FLOAT와 동일	4
	DOUBLE(p)	FLOAT 타입에 명시된 정밀도(p)보다 큼. 범위는 –10e+308 에서 +10e+308.	8
	DOUBLE PRECISION	DOUBLE과 동일	8

2.3 CUBRID 데이터 타입-날짜/시간

데이터 형식		설명	크기
Data-time	DATE	day(일), month(월), year(년)를 표현하는 타입. `DATE 'mm/dd[/yyyy]'`	4
	TIME	hour(시), minute(분), second(초)를 표현하는 타입. `TIME 'hh:mm [:ss] [am | pm]'`	4

	TIMESTAMP	date와 time 조합의 값을 표현(32Bit).
		`TIMESTAMP 'hh:mm [:ss] [am|pm] mm/dd [/yyyy]'`
		`TIMESTAMP 'mm/dd [/yyyy] hh:mm [:ss] [am|pm]'`
			4

	DATETIME	date와 time 조합의 값을 표현(64Bit), 밀리세컨드 단위.
		`DATETIME 'mm/dd/yyyy hh:mm:ss[.msec]'`
		`DATETIME 'yyyy-mm-dd hh:mm:ss[.msec]'`
			8

2.4 CUBRID 데이터 타입-집합형

데이터 형식		설명	크기
Collections	SET	중복될 수 없는 값들의 set을 명세	Variable
	MULTISET	중복될 수 있는 값들의 set을 명세.	Variable
	LIST	엔트리에 명세된 순서대로 값들의 set을 유지. 중복 허용.	Variable
	SEQUENCE	LIST와 동일	Variable

2.5 CUBRID vs MySQL 타입 비교 1

MySQL	CUBRID	차이점
BIT	BIT	
BOOLEAN	BIT	
TINYINT	SMALLINT	tinyint ➔ 1byte, smallint ➔ 2byte
SMALLINT	SMALLINT	
MEDIUMINT	INTEGER	midiumint ➔ 3byte, integer ➔ 4byte
INTEGER	INTEGER	
BIGINT	BIGINT	
REAL	REAL	
DOUBLE	DOUBLE	
DOUBLE PRECISION	DOUBLE PRECISION	
FLOAT(n)	FLOAT(n)	MySQL: 1<=n<=53 CUBRID: 1<=n<=24
DECIMAL(p,s) NUMERIC(p,s)	DECIMAL(p,s) NUMERIC(p,s)	MySQL: 1<=p<=65 , 0<=s<=30 CUBRID: 1<=p<=38 , 0<=s<=p

2.5 CUBRID vs MySQL 타입 비교 2

MySQL	CUBRID	비고
TINYBLOB , <=256	BIT VARYING (n) or Large Object(GLO)	1G이하의 데이터는 BIT VARYING으로 대체 권장. 1G이상의 데이터는 GLO로 대체 권장.
BLOB(n) , n<= 65K	BIT VARYING (n) or Large Object(GLO)	1G이하의 데이터는 BIT VARYING으로 대체 권장. 1G이상의 데이터는 GLO로 대체 권장.
MEDIUMBLOB , n<=16M	BIT VARYING (n) or Large Object(GLO)	1G이하의 데이터는 BIT VARYING으로 대체 권장. 1G이상의 데이터는 GLO로 대체 권장.
LONGBLOB , <=4G	BIT VARYING (n) or Large Object(GLO)	1G이하의 데이터는 BIT VARYING으로 대체 권장. 1G이상의 데이터는 GLO로 대체 권장.

➔ ODBC/OLEDB 인터페이스에서는 Large Object사용에 제한이 있다.

2.5 CUBRID vs MySQL 타입 비교 3

MySQL	CUBRID	비고
DATE	DATE	MySQL: date ➔ 3byte CUBRID: date ➔ 4byte
DATETIME	DATETIME	
TIME	TIME	MySQL: time ➔ 3byte CUBRID: time ➔ 4byte
TIMESTAMP	TIMESTAMP	CUBRID: 초단위 연산만 지원
YEAR	-	VARCHAR(4)로 대체.

2.5 CUBRID vs MySQL 타입 비교 4

MySQL	CUBRID	비고
CHAR(n)	CHAR(n)	MySQL: n<=255 CUBRID: n<=1G
VARCHAR(n)	VARCHAR(n)	MySQL: n<=65K CUBRID: n<=1G
MEDIUMTEXT(n)	VARCHAR(n)	MySQL: n<=16M CUBRID: n<=1G
TEXT(n)	VARCHAR(n)	MySQL: n<=2G CUBRID: n<=1G
LONGTEXT(n)	VARCHAR(n)	MySQL: n<=4G CUBRID: n<=1G
SET(n)	SET	MySQL: n<=64, 문자열 타입만 가능. CUBRID:개수 및 데이터 타입 제한 없음.
ENUM(..)	-	대체 기능 없음.

3. 함수 및 연산자 비교

3.1 CUBRID vs MySQL 함수 비교 1

MySQL	CUBRID
NOW	**SYSTIMESTAMP**
CURRENT_DATE	SYSDATE or SYS_DATE or CURRENT_DATE
CURRENT_TIME	SYSTIME or SYS_TIME or CURRENT_TIME

IFNULL
 	
`SELECT IFNULL(NULL,10);`
	NVL
	
	`SELECT NVL(NULL,10)`
	`FROM db_root;`

-	NVL2
	
	`SELECT NVL2(NULL,20,10)`
	`FROM db_root;`

LOCATE `SELECT LOCATE('SOURCE', 'OPENSOURCE');`	POSITION `SELECT POSITION ('SOURCE' IN 'OPENSOURCE');`

3.1 CUBRID vs MySQL 함수 비교 2

MySQL	CUBRID
SUBSTRING SELECT **SUBSTRING**('CUBRID' FROM -4 FOR 2); SELECT **SUBSTRING**('CUBRID', -4, 2); // '20090301' 형태의 문자열로 변경하기 SELECT REPLACE(**SUBSTRING**(CURRENT_DATE, 1,10),'-','')	SUBSTRING SELECT **SUBSTRING**('CUBRID' FROM -4 FOR 2) FROM db_root; SELECT **SUBSTR** ('CUBRID' ,-4 ,2) FROM db_root; // '20090301' 형태의 문자열로 변경하기 SELECT **TO_CHAR**(SYS_DATE,'yyyymmdd') FROM db_root;
LEFT SELECT **LEFT**('CUBRID', 5);	SUBSTR SELECT **SUBSTR**('CUBRID', 1,5) FROM db_root;

3.1 CUBRID vs MySQL 함수 비교 3

MySQL	CUBRID

GROUP_CONCAT
 	
//colum_2의 데이터를 정렬한 후 '^'를 통해 구분하여 출력함.
 	
SELECT col_1, GROUP_CONCAT(col_2 ORDER BY col_2 desc SEPARATOR '^') FROM tbl group by col_1;
	SET or LIST
	
	//int_column인 경우
	SELECT col_1,LIST(SELECT int_column|| '^' FROM tbl ORDER BY int_column desc) FROM tbl group by col_1 ;
	
	//char_column인 경우 trim
	SELECT col_1, LIST(SELECT trim(char_column)|| '^' FROM tbl ORDER BY char_column desc) FROM tbl group by col_1 ;

CONCAT SELECT CONCAT('CU', 'B', 'RID');	+ 연산자 혹은 || 연산자 이용 SELECT 'CU'||'B'||'RID' FROM db_root;

3.2 CUBRID vs MySQL 연산자 비교

MySQL	CUBRID
!= 또는 <>	<>
CONCAT 또는 +	|| 또는 +

4. 쿼리 비교

4.1 표현식의 변환

➔CUBRID 예약어를 사용하기 위해서는 큰 따옴표로 묶어준다.

CUBRID는 MySQL와 예약어가 다르기 때문에 이를 확인하여 큰 따옴표(")처리를 해주어야 한다.

MySQL	CUBRID
`SELECT depth FROM tbl_1;`	`SELECT "depth" FROM tbl_1;`
`SELECT 'key' FROM tbl_1;`	`SELECT "key" FROM tbl_1;`

➔CUBRID에서는 != 대신에 <>를 사용한다.

MySQL	CUBRID
`SELECT * FROM tbl_1 where a != 0;`	`SELECT * FROM tbl_1 where a <> 0;`

4.2 더미 테이블의 명시

➔ SQL 표준을 명확하게 적용하여 쿼리를 작성한다.

CUBRID에서는 FROM절에 더미 테이블(db_root)이 명시되어야 한다.

MySQL	CUBRID
`SELECT 1+1;`	`SELECT 1+1 FROM db_root;`

4.3 데이터 타입의 명시적 정의

➔ 데이터 타입은 명확하게 지정한다.

CUBRID는 묵시적인 데이터 타입 변환을 지원하지 않으므로, CAST AS를 사용하여 명시적으로 데이터 타입을 변환하여야 한다.

MySQL	CUBRID
`SELECT 1+'1';`	`SELECT 1+1 FROM db_root;`

`SELECT * FROM tbl_1`
`WHERE varchar_column = 123;`
 	
`SELECT * FROM tbl_1`
`WHERE int_coulmun = '123';`
	`SELECT * FROM tbl_1`
	`WHERE varchar_column = '123';`
	
	`SELECT * FROM tbl_1`
	`WHERE int_coulmun = 123;`

4.4 GROUP BY를 이용한 데이터 집계

➔SELECT절의 실행으로 검색하고자 하는 컬럼을 GROUP BY절에 명시하고, 집계함수(COUNT, MIN, MAX등)을 이용하여 GROUP BY 구문을 작성한다.

MySQL	CUBRID
SELECT col_1, col_2, col_3 FROM tbl GROUP BY col_1;	SELECT col_1, col_2, col_3 FROM tbl GROUP BY **col_1,col_2, col_3** ; SELECT col_1, col_2, **max(col_3)** FROM tbl GROUP BY col_1,col_2;

4.5 검색 결과의 일부 출력

➔ LIMIT 질의문을 ROWNUM 또는 ORDERBY_NUM으로 대체한다.

CUBRID에서는 ROWNUM 값이 ORDER BY절에 의한 정렬 과정 전에 생성된다. ORDER BY절에 의한 정렬 이후의 값을 얻기 위해서는 ORDERBY_NUM()함수를 사용한다.

MySQL	CUBRID
SELECT col_1, col_2 FROM tbl **LIMIT** 1, 10;	SELECT col_1, col_2 FROM tbl **ROWNUM** between 1 and 10;
SELECT col_1, col_2 FROM tbl group by col_1 **LIMIT** 1,10;	SELECT col_1, col_2 FROM tbl group by col_1 **HAVING GROUPBY_NUM()** between 1 and 10;
SELECT col_1, col_2 FROM tbl ORDER BY col_1 **LIMIT** 1, 10;	SELECT col_1, col_2 FROM tbl ORDER BY col_1 **FOR ORDERBY_NUM()** between 1 and 10;

4.6 검색 결과의 가로 출력(PIVOT 기능)

➔ LIST나 SET 함수를 이용하여 PIVOT 기능을 구현할 수 있다.

1. 정렬이 필요 없이 col_3을 가로 출력:

```
SELECT id
, LIST(SELECT id,col_3 FROM tbl_1 t1 where t2.id = t1.id )
FROM tbl_2 t2;
```

2. col_1, col_2기준으로 정렬한 순서대로 col_3을 가로 출력:

```
SELECT id,
   LIST(SELECT col_3 FROM (SELECT id,col_3 FROM tbl_1 t1
   where t2.id = t1.id ORDER BY col_1, col_2) t3
        on t3.id = t2.id),
FROM tbl_2 t2;
```

4.7 조회수 카운트를 위한 구문 작성

➔INCR함수를 이용하여 SELECT와 UPDATE를 한번에 실행하여 조회수를 증가시킬 수 있다.

MySQL	CUBRID
SELECT content, click_cnt FROM board_1; UPDATE board_1 SET click_cnt = click_cnt +1;	SELECT content, **INCR**(click_cnt) FROM board_1;

4.8 기타 비교

기능	지원 여부	비고
Prepare Statement pooling	O	MySQL의 prepare statement pooling과 유사. 하나의 커넥션 ➔ 하나의 결과값 핸들링 가능 (AUTOCOMMIT이 ON으로 설정, 디폴트). 하나의 커넥션 ➔ 여러 개의 결과값 핸들링 가능 (AUTOCOMMIT이 OFF으로 설정).
Stored Procedure	O	Java Stored Procedure 지원 O
FULL 외부조인	X	Left, right outer Join은 지원.
Character set	X	응용 프로그램에서 사용하는 character set을 사용.
속성 크기 변경	X	ALTER TABLE구문을 통한 속성 크기 변경 불가.
Temp table	X	

4.9 정리

기능	비고
표현식의 차이점	예약어는 큰 따옴표로 감싸서 사용 더미 테이블(db_root)을 반드시 명시 != 대신에 <>의 사용
묵시적 타입 변환	명시적으로 데이터 타입 지정 및 데이터 입력
클릭 카운트(INCR함수)	MySQL의 UPDATE 및 SELECT을 한번에 실행
AUTO_INCREMENT	MySQL의 AUTO_INCREMENT와 유사
검색 결과의 일부 출력	ROWNUM과 ORDERBY_NUM() 사용 MySQL의 LIMIT, OFFSET을 대체
검색 결과의 가로 출력	SET or LIST 및 조인을 이용하여 대체