Back\_end 홍기욱

2017.06.28

**1. 산술 연산자의 우선 순위 조사**

|  |  |
| --- | --- |
| 우선순위 | 연산자 |
| 1 | INTERVAL |
| 2 | BINARY, COLLATE |
| 3 | ! |
| 4 | * (음수), ~ (BIT 전이) |
| 5 | ^ |
| 6 | \*, /, DIV, %, MOD |
| 7 | -, + |
| 8 | <<, >> |
| 9 | & |
| 10 | | |
| 11 | =, <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN |
| 12 | BETWEEN, CASE, WHEN, THEN, ELSE |
| 13 | NOT |
| 14 | &&, AND |
| 15 | XOR |
| 16 | ||, OR |
| 17 | = (할당), := |

**2. 대소문을 구별하는 방법에 대한 조사**

1. 테이블명이나 컬럼명에서 대소문자 구분

Test와 test를 다른 것으로 간주

2. 데이터 조회 시 대소문자 구분

VARCHAR, TEXT는 대소문자 구분하지 않음

VARBINARY로 컬럼 생성시 대소문자 구분

**3. date\_format 표현 방식에 대한 조사**

Syntax

DATE\_FORMAT(date, format\_mask)

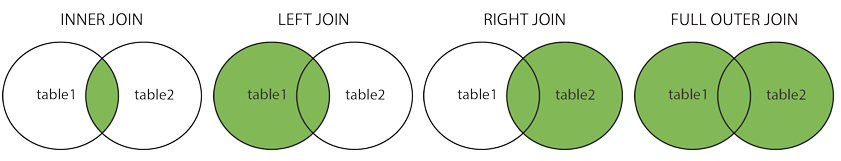
Parameter

date : The date to format

format\_mask : The format to apply to date. Can be one or a combination of the following values:

|  |  |
| --- | --- |
| Format | Description |
| %a | Abbreviated weekday name (Sun to Sat) |
| %b | Abbreviated month name (Jan to Dec) |
| %c | Numeric month name (0 to 12) |
| %D | Day of the month as a numeric value, followed by suffix (1st, 2nd, 3rd, ...) |
| %d | Day of the month as a numeric value (01 to 31) |
| %e | Day of the month as a numeric value (0 to 31) |
| %f | Microseconds (000000 to 999999) |
| %H | Hour (00 to 23) |
| %h | Hour (00 to 12) |
| %I | Hour (00 to 12) |
| %i | Minutes (00 to 59) |
| %j | Day of the year (001 to 366) |
| %k | Hour (00 to 23) |
| %l | Hour (1 to 12) |
| %M | Month name in full (January to December) |
| %m | Month name as a numeric value (00 to 12) |
| %p | AM or PM |
| %r | Time in 12 hour AM or PM format (hh:mm:ss AM/PM) |
| %S | Seconds (00 to 59) |
| %s | Seconds (00 to 59) |
| %T | Time in 24 hour format (hh:mm:ss) |
| %U | Week where Sunday is the first day of the week (00 to 53) |
| %u | Week where Monday is the first day of the week (00 to 53) |
| %V | Week where Sunday is the first day of the week (01 to 53). Used with %X |
| %v | Week where Monday is the first day of the week (01 to 53). Used with %X |
| %W | Weekday name in full (Sunday to Saturday) |
| %w | Day of the week where Sunday=0 and Saturday=6 |
| %X | Year for the week where Sunday is the first day of the week. Used with %V |
| %x | Year for the week where Monday is the first day of the week. Used with %V |
| %Y | Year as a numeric, 4-digit value |
| %y | Year as a numeric, 2-digit value |

**4. JOIN 방식에 대한 조사**



1. 내부 조인 (Inner Join)

내부조인 (Inner Join) 은 단순히 연결만 한 개념이 아니라 결과값이 일치하는 것만 join을 한 것이다. 교집합(intersection)의 개념과도 비슷한데 기준이 되는 테이블과 일치하는 값만 join 하여 return 한다.

select \* from tbl\_name INNER JOIN tbl\_b ON id = no ;

select \* from tbl\_name , tbl\_b where tbl\_name.id = tbl\_b.no ;

2. 외부 조인 (Outer Join)

내부조인 한 결과에 덧붙여 일치하지 않는 데이터도 가져올 수 있다. 외부조인에는 왼쪽, 오른쪽, 전체 외부조인의 3가지로 구분이 된다.

왼쪽 외부 조인 (Left Join)

왼쪽 테이블을 기준으로 외부 조인 한다. 오른쪽 테이블에 인치하는 값이 없어도 왼쪽 테이블의 데이터는 모두 가져 온다. 이때 오른쪽에 없는 데이터는 null 로 대치 된다.

외부조인은 여집합(Complement)과 유사함.

select \* from tbl\_name LEFT JOIN tbl\_b ON id = no ;

오른쪽 외부 조인 (Right Join)

오른쪽 테이블을 기준으로 외부 조인 한다. 왼쪽 테이블에 인치하는 값이 없어도 오른쪽 테이블의 데이터는 모두 가져 온다. 이때 왼쪽에 없는 데이터는 null 로 대치 된다.

외부조인은 여집합(Complement)과 유사함.

select \* from tbl\_name LEFT JOIN tbl\_b ON id = no ;

전체 외부 조인 (Full Join)

오른쪽 외부조인과 왼쪽 외부조인을 합친 개념이다. 왼쪽/오른쪽 테이블의 값이 일치하든 일치 하지 않든 모두 가져오는 조인이다. 다만 Cross join 과 다른 점은 Full join 은 빈 값을 null로 return 하는 것이고 Cross join 은 양쪽 테이블을 그냥 단순히 연결시켜 준다는 점이다.

select \* from tbl\_name FULL JOIN tbl\_b ON id = no ;

3. 교차 조인 (Cross Join)

교차 조인 (Cross Join) 은 말 그대로 두 개의 테이블을 그냥 붙였다는 개념이다. 아무런 조건이나 제한 사항 없이 양쪽은 있는 그대로 연결만 했다는 개념이다.

합집합(Union)과 유사함.

select \* from tbl\_a CROSS JOIN tbl\_b ;

select \* from tbl\_a , tbl\_b ;

4. Self Join

원하는 데이터들이 한 테이블에 있을 때 사용한다. 한 테이블에 필요한 칼럼이 있을 때, 한 테이블을 두 개의 테이블처럼 EQUI JOIN으로 조회할 수 있다.

**5. AUTO\_INCREMENT에 대한 조사**

AUTO\_INCREMENT 속성을 사용하여 새로운 행에 대한 고유 ID를 생성 할 수 있습니다.

AUTO\_INCREMENT 컬럼에 값을 지정하지 않으면, MySQL은 시퀀스 번호를 자동으로 할당합니다. NO\_AUTO\_VALUE\_ON\_ZERO SQL 모드가 활성화되어 있지 않으면, 열에 0을 명시 적으로 할당하여 시퀀스 번호를 생성 할 수도 있습니다.

열이 NOT NULL로 선언되면, 열에 NULL을 할당하여 일련 번호를 생성 할 수도 있습니다. AUTO\_INCREMENT 열에 다른 값을 삽입하면 열은 해당 값으로 설정되고 시퀀스가 ​​재설정되어 다음 자동으로 생성 된 값이 가장 큰 열 값에서 순차적으로옵니다.

**6. SELF UPDATE 방식에 대한 내용 조사**

방법 2가지

1. 복사본에 대해 별명을 지정하지 않는 것

Update table

set date = T2.date

from table T2

where T2.user = table.user

1. 별명을 UPDATE의 item으로 사용하는 것

Update T1

set date = T2.date

from table

inner join table T2 on T2.user = table.user

**7. replace / on duplicate key update 의 동작 방식에 대한 조사**

Replace Into involves:

1. Try insert on the table

2. If 1 fails, delete row and insert new row

Insert on Duplicate Key Update involves:

1. Try insert on table

2. If 1 fails, update row

REPLACE는 내부적으로 삭제를 수행 한 다음 삽입을 수행합니다. 이 행을 가리키는 외래 키 제약 조건이 있으면 문제가 발생할 수 있습니다. 외래 키가 계단식 삭제로 설정된 경우 REPLACE가 다른 테이블의 행을 삭제하기 때문입니다.

INSERT ON DUPLICATE KEY UPDATE를 사용하면 이 문제를 피할 수 있습니다.

**8. data type 조사**

|  |  |
| --- | --- |
| Data type | Description |
| CHARACTER(n) | Character string. Fixed-length n |
| VARCHAR(n) or CHARACTER VARYING(n) | Character string. Variable length. Maximum length n |
| BINARY(n) | Binary string. Fixed-length n |
| BOOLEAN | Stores TRUE or FALSE values |
| VARBINARY(n) or BINARY VARYING(n) | Binary string. Variable length. Maximum length n |
| INTEGER(p) | Integer numerical (no decimal). Precision p |
| SMALLINT | Integer numerical (no decimal). Precision 5 |
| INTEGER | Integer numerical (no decimal). Precision 10 |
| BIGINT | Integer numerical (no decimal). Precision 19 |
| DECIMAL(p, s) | Exact numerical, precision p, scale s. Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after the decimal |
| NUMERIC(p, s) | Exact numerical, precision p, scale s. (Same as DECIMAL) |
| FLOAT(p) | Approximate numerical, mantissa precision p. A floating number in base 10 exponential notation. The size argument for this type consists of a single number specifying the minimum precision |
| REAL | Approximate numerical, mantissa precision 7 |
| FLOAT | Approximate numerical, mantissa precision 16 |
| DOUBLE PRECISION | Approximate numerical, mantissa precision 16 |
| DATE | Stores year, month, and day values |
| TIME | Stores hour, minute, and second values |
| TIMESTAMP | Stores year, month, day, hour, minute, and second values |
| INTERVAL | Composed of a number of integer fields, representing a period of time, depending on the type of interval |
| ARRAY | A set-length and ordered collection of elements |
| MULTISET | A variable-length and unordered collection of elements |
| XML | Stores XML data |

**9. not null 과 multi insert 에 대한 조사**

1. NOT NULL

필드에 NULL값을 포함하지 않겠다는 의미입니다. 항상 값을 포함 할 필드에 적용되며, 아무런 값을 넣지 않으면 에러가 발생합니다.

CREATE TABLE T (id int NOT NULL);

1. multi insert

VALUES 구문을 사용하는 INSERT 문은 여러 행을 삽입 할 수 있습니다. 이를 수행하려면 각각 괄호 안에 쉼표로 구분 된 여러 개의 열 값 목록을 포함시킵니다.

INSERT INTO tbl\_name (a,b,c) VALUES(1,2,3),(4,5,6),(7,8,9);

복수 행 INSERT. 또는 INSERT INTO ... SELECT.의 경우, 데이터 유형의 내재 된 디폴트 값으로 설정됩니다. 숫자 유형의 경우 0, 문자열 유형의 경우 빈 문자열 ('') 및 날짜 및 시간 유형의 경우 "0"값입니다. INSERT INTO ... SELECT 문은 여러 행 삽입과 동일한 방식으로 처리됩니다. 서버가 SELECT에서 결과 집합을 검사하지 않아 단일 행을 반환하는지 확인하기 때문입니다. 단일 행 INSERT의 경우 NOT NULL 열에 NULL을 삽입하면 경고가 발생하지 않고 오류가 발생하여 명령문이 실패합니다.

**10. MySQL에서 PRIMARY KEY를 사용 해야 하는 이유 및 AUTO\_INCREMENT를 권장하는 이유 조사**

1. MySQL에서 PRIMARY KEY를 사용 해야 하는 이유

MySQL에서 기본 키는 레코드를 고유하게 정의하는 단일 필드 또는 필드 조합입니다. 기본 키의 일부인 필드에는 NULL 값을 포함 할 수 없습니다. 테이블에는 기본 키가 하나만 있을 수 있습니다.

기본 키는 빠른 쿼리 성능을 위해 필요한 인덱스와 관련이 있습니다. 기본 키는 NULL 값을 포함 할 수 없으므로 NOT NULL 최적화의 이점을 누립니다. 만약, InnoDB 스토리지 엔진을 사용하면 테이블 데이터가 물리적으로 구성되어 기본 키 열을 기반으로 한 초고속 검색 및 정렬 작업을 수행 할 수 있습니다.

1. AUTO\_INCREMENT를 권장하는 이유 (사용자 이름을 key로 했을 때와 차이점)

* 중복 된 사용자 이름을 허용합니다.
* 사용자 이름을 변경하기 쉽습니다. (사용자 이름이 기본 키일 때 사용자가 사용자 이름을 변경하면 사용자 테이블에 대한 외래 키 참조가 있는 모든 테이블을 업데이트 해야 합니다.)
* 숫자 인덱스를 유지하는 것이 훨씬 빠르기 때문에 선택, 조인 및 삽입이 varchar 기본 키보다 빠릅니다.
* 유효성 검사가 간단합니다. if ((int) $ id> 0) {...}
* 외래 키가 잠재적으로 큰 문자열 값을 복제 할 필요가 없으므로 오버 헤드가 훨씬 낮습니다.

**11. foreign key 추가 옵션에 대한 내용 조사**

외부 키를 지원하는 스토리지 엔진의 경우, MySQL은 부모 테이블에 일치하는 후보 키 값이 없으면 자식 테이블에 외래 키 값을 생성하려고 시도하는 INSERT 또는 UPDATE 연산을 거부합니다.

UPDATE 또는 DELETE 조작이 하위 테이블과 일치하는 행을 갖는 부모 테이블의 키 값에 영향을 주는 경우, FOREIGN KEY 절의 ON UPDATE 및 ON DELETE의 옵션에 의존합니다.

* CASCADE : 부모 테이블에서 행을 삭제하거나 업데이트하고, 자식 테이블에서 일치하는 행을 자동으로 삭제하거나 업데이트합니다.
* SET NULL : 상위 테이블에서 행을 삭제하거나 업데이트하고 하위 테이블의 외래 키 열을 NULL로 설정합니다. ON DELETE SET NULL 및 ON UPDATE SET NULL 절이 모두 지원됩니다.
* RESTRICT : 상위 테이블에 대한 delete와 update를 거부합니다. RESTRICT (또는 NO ACTION)을 지정하는 것은 ON DELETE 또는 ON UPDATE 절을 생략하는 것과 같습니다.
* NO ACTION : MySQL에서는 RESTRICT와 같습니다. MySQL 서버는 참조 된 테이블에 관련 외래 키 값있는 경우 상위 테이블에 대한 삭제 또는 업데이트 작업을 거부합니다. MySQL에서는 외래 키 제약 조건이 즉시 검사되므로 NO ACTION은 RESTRICT와 동일합니다.
* SET DEFAULT : InnoDB와 NDB는 ON DELETE SET DEFAULT 또는 ON UPDATE SET DEFAULT 절이 포함 된 테이블 정의를 거부한다.

**12. foreign key 무효화 방법에 대한 조사**

외래 키는 두 테이블의 데이터 간 연결을 설정하고 강제 적용하는 데 사용되는 열입니다.

외래 키 제약조건 때문에 테이블이 삭제되지 않는 경우가 있습니다.

mysql> SET foreign\_key\_checks = 0;

mysql> drop table TABLENAME

mysql> SET foreign\_key\_checks = 1;

ALTER TABLE tb WITH NOCHECK

ALTER TABLE tb NOCHECK CONSTRAINT

**13. enum 유효성 적용 방법에 대한 조사**

ENUM은 테이블 작성시 열 스펙에 명시 적으로 열거되는 허용 값 목록에서 선택된 값을 갖는 문자열 오브젝트입니다.

ENUM의 값은 특정 상황에서 빈 스트링 ('') 또는 NULL 값이 될 수 있습니다.

잘못된 값을 ENUM에 삽입하면 (즉, 허용되는 값 목록에 없는 문자열) 빈 문자열이 특수 오류 값으로 삽입됩니다. 이 문자열은 숫자 값이 0이라는 사실 때문에 "일반"빈 문자열과 구별 될 수 있습니다.

만일 스트릭트 SQL 모드가 활성화 되어 있다면, 유효하지 않은 ENUM 값을 삽입하면 오류가 발생합니다.

ENUM 컬럼이 NULL을 허용하도록 선언되었다면, NULL 값은 열의 유효한 값이고 기본값은 NULL입니다.

ENUM 컬럼이 NOT NULL로 선언되었다면, 디폴트 값은 허용된 값 리스트의 첫 번째 요소(element)가 됩니다.

**14. 데이터 변경이 가능한 뷰와 변경이 불가능한 뷰에 대한 조사**

뷰는 기본 테이블의 정보를 기본으로 조회 합니다. 뷰로 UPDATE문을 수행하면 뷰를 생성할 때 사용되었던 기본 테이블의 내용이 변경됩니다. 뷰를 조회하면 변경된 결과를 저장한 기본 테이블에서 뷰를 정의할 때 기술한 쿼리문을 수행해서 출력합니다. 뷰를 통해서 보여지는 정보는 기본 테이블에서 얻어지는 것입니다.

뷰를 생성할 때 WHERE 다음에 WITH CHECK OPTION 을 기술하면, 그 조건에 의해서 기본 테이블에서 정보가 추출됩니다. 때문에 조건에 사용된 컬럼 값을 뷰를 통해서 변경 불가능하게 됩니다.

뷰의 WITH READ ONLY 옵션은 뷰를 생성한 후 어떠한 DML문도 수행할 수 없도록 하는 기능을 합니다.

**15. MySQL 권한에 대한 내용 및 부여 방법에 대한 조사**

권한 : SELECT, INSERT, DELETE, UPDATE

-모든 권한 주기

grant all privileges on \*.\* to 사용자명@호스트

identified by '비밀번호' with grant option;

-특정 사용자에게 특정 DB의 모든 테이블에 모든 권한 부여하기

grant all privileges on DB명.\* to 사용자명@호스트 identified by '비밀번호';

-특정 사용자에게 특정 DB의 모든 테이블에 select, insert 권한 부여하기

grant select, insert on DB명.\* to 사용자명@호스트

identified by '비밀번호';

-특정 사용자에게 특정 DB의 특정 테이블의 정해준 컬럼에만 update 권한 부여하기

grant update(컬럼1, 컬럼2 ,컬럼3) on DB명. 테이블명 to 사용자명@호스트

identified by '비밀번호';

-모든 호스트에서 접속가능하고 select 권한만 갖는 사용자를 추가할 땐

grant select on DB명.\* to 사용자명@% identified by '비밀번호';

-ip주소가 192.168.0. 으로 시작하는 컴퓨터에서 접속 가능한 사용자를 추가할 땐

grant all privileges on \*.\* to 사용자명@'192.168.0.%' identified by '비밀번호';

-모든 권한 삭제

REVOKE ALL PRIVILEGES ON \*.\* FROM 사용자명@호스트;

-권한 확인하기

SHOW GRANTS FOR 사용자명;

**16. MySQL partitioning 에 대한 조사**

파티셔닝(데이터베이스 설계 기술)은 성능, 관리 효율성을 향상시키고 유지 보수를 단순화하며 대용량 데이터 저장 비용을 절감합니다. 개별 디스크 드라이브에 테이블을 물리적으로 배치하여 테이블을 분할하지 않고도 파티셔닝을 수행 할 수 있습니다. 파티셔닝을 사용하면 테이블, 인덱스 및 인덱스 구성 테이블을 더 작은 조각으로 세분 할 수 있으므로 스캔 할 데이터가 더 적기 때문에 데이터의 일부만 액세스하는 쿼리가 더 빠르게 실행될 수 있습니다.

수평 파티셔닝 : 수평 파티셔닝은 테이블 행을 여러 개의 분할 영역으로 나눕니다. 테이블에 정의 된 모든 열은 각 파티션에서 찾을 수 있으므로 실제 테이블 속성은 누락되지 않습니다. 모든 파티션은 개별적으로 또는 전체적으로 처리 할 수 ​​있습니다.

수직 파티셔닝 : 수직 파티셔닝은 테이블을 더 적은 수의 컬럼을 포함하는 여러 개의 테이블로 나눕니다. 수평 분할과 마찬가지로 수직 분할에서 쿼리는 더 적은 데이터를 스캔하여 쿼리 성능을 향상시킵니다.

파티셔닝의 6가지 타입

RANGE Partitioning

LIST Partitioning

COLUMNS Partitioning

HASH Partitioning

KEY Partitioning

Subpartitioning

**17. MySQL trigger 에 대한 조사**

MySQL 트리거는 SQL 문에 의한 테이블 변경 사항에 대해서만 활성화됩니다. 트리거는 테이블에 대한 이벤트에 반응해 자동으로 실행되는 작업입니다. 트리거는 DML의 데이터 상태의 관리를 자동화하는 데 사용됩니다. 트리거를 사용하여 데이터 작업 제한, 작업 기록, 변경 작업 감사 등을 할 수 있습니다.

즉, A 테이블에 DML의 변화를 줄 때 B 테이블에도 변화를 주는 경우를 말합니다.

트리거에는 트렌젝션이 사용되어, 트리거를 이용하여 테이블을 변경하다가 오류가 발생하면, Rollback이 됩니다.

**18. MySQL procedure 에 대한 조사**

여러 SQL 문을 하나의 SQL 문처럼 정리하여 'CALL ✕ ✕'라는 명령으로 실행할 수 있게 만든 것을 저장 프로시저(Stored Procedure)라고 합니다. 저장 프로시저는 일련의 절차를 정리해서 저장한 것입니다.

사전에 준비 둔 많은 명령을 자동으로 실행할 수 있기 때문에, 작업의 효율성도 높일 수 있습니다. 단, 중요한 데이터가 축적된 데이터베이스에서 제대로 검증되지 않은 저장 프로시저를 실행하는 것은 매우 위험합니다.

