이전 영상에서 테이블을 생성할 때, 각 컬럼마다 저장될 값에 알맞은 **데이터 타입(Data type)**을 설정했는데요.

각 컬럼에 적절한 데이터 타입을 잘 설정하는 일은 아주 중요합니다.

데이터 타입을 잘 설정해야 저장 용량을 효율적으로 활용할 수 있고, 나중에 row 수가 많아졌을 때는 성능에 영향을 미치기도 하기 때문이죠..

사용할 수 있는 데이터 타입은 DBMS마다 조금씩 차이가 있는데요.

일단 MySQL의 데이터 타입들을 살펴보겠습니다.

MySQL의 데이터 타입 중 일반적으로 쓰이는 것들은 다음 세 가지 카테고리로 분류할 수 있습니다.

**Numeric types(숫자형 타입)**

**Date and Time types(날짜 및 시간 타입)**

**String types(문자열 타입)**

이밖에도 여러 카테고리가 더 있지만, 당장 알 필요는 없습니다. 위의 세 가지 카테고리에 속한 주요 데이터 타입들을 살펴보겠습니다.

**1. Numeric types(숫자형 타입)**

숫자를 나타내기 위해서 사용되는 데이터 타입입니다.

숫자형 타입은 다시 **정수형 타입**과 **실수형 타입**으로 나눌 수 있는데요. 하나씩 살펴볼게요.

**(1) 정수형 타입**

말 그대로 정수값을 저장하는 타입입니다. 여기에 해당하는 타입들은 그것이 나타낼 수 있는 정수값의 범위에 차이가 있습니다.

**1) TINYINT**

: 작은 범위의 정수들을 저장할 때 쓰는 데이터 타입입니다. TINYINT에서 INT는 Integer(정수)의 줄임말인데요. 최소 -128 ~ 최대 127 까지의 정수를 저장할 수 있는 타입입니다. 이때 SINGED와 UNSIGNED라는 개념을 알아야하는데요. SIGNED는 ‘양수, 0, 음수’를 나타내고, UNSIGNED는 ‘0과 양수’를 나타냅니다. TINYINT 뒤에 둘 중 무엇을 붙이느냐에 따라 아래와 같이 나타낼 수 있는 숫자의 범위가 달라집니다.

TINYINT SIGNED : -128 ~ 127

TINYINT UNSIGNED : 0 ~ 255

이런 원리는 다른 정수형 타입에도 똑같이 적용됩니다. 참고로, TINYINT라고만 썼을 때는 SIGNED가 붙은 것으로 자동 해석됩니다.

**2) SMALLINT**

: TINYINT 보다 좀더 큰 범위의 정수를 나타낼 때 쓰는 데이터 타입입니다. SINGED와 UNSIGNED에 따라 저장할 수 있는 수의 범위는 다음과 같습니다.

SMALLINT SIGNED : -32768 ~ 32767

SMALLINT UNSIGNED : 0 ~ 65535

**3) MEDIUMINT**

: 더 넓은 범위를 나타내는 데이터 타입니다.

MEDIUMINT SIGNED : -8388608 ~ 8388607

MEDIUMINT UNSIGNED : 0 ~ 16777215

**4) INT**

: 더 넓은 범위를 나타내는 데이터 타입입니다.

INT SIGNED : -2147483648 ~ 2147483647

INT UNSIGNED : 0 ~ 4294967295

**5) BIGINT**

: 아주 큰 범위의 정수를 저장하는 데이터 타입입니다.

BIGINT SIGNED : -9223372036854775808 ~ 9223372036854775807

BIGINT UNSIGNED : 0 ~ 18446744073709551615

BIGINT 데이터 타입을 사용한다면 아무리 큰 정수라도 잘 저장할 수 있겠죠?

**(2) 실수형 타입**

우리는 정수뿐만 아니라 소수점이 붙어있는 수를 사용하기도 합니다. 이런 수를 저장하기 위한 타입을 실수형 타입이라고 하는데요. 키(180.489cm)나 몸무게(73.45kg), 온도(36.76도)처럼 정밀한 값이 요구되는 컬럼에는 실수형 타입을 지정해야 합니다.

실수형 타입은 그 타입마다

얼마나 넓은 범위의 수를 나타낼 수 있는지 뿐만 아니라

소수점 뒤에 얼마나 많은 개수의 자리수가 존재할 수 있는지, 그러니까 하나의 값을 얼마나 정밀하게 표현할 수 있는지가 다른데요. 하나씩 살펴볼게요.

**1) DECIMAL**

:일반적으로 자주 쓰이는 실수형 타입 중 하나로 보통 DECIMAL(M, D)의 형식으로 나타냅니다. 여기서 M은 최대로 쓸 수 있는 전체 숫자의 자리수이고, D는 최대로 쓸 수 있는 소수점 뒤에 있는 자리의 수를 의미합니다. 예를 들어

DECIMAL (5, 2)라면 -999.99 부터 999.99 까지의 실수를 나타낼 수 있는 거죠.

M은 최대 65, D는 30까지의 값을 가질 수 있습니다.

그리고 DECIMAL이라는 단어 대신 DEC, NUMERIC, FIXED를 써도 됩니다.

**2) FLOAT**

:

-3.402823466E+38 ~ -1.175494351E-38,

0,

1.175494351E-38 ~ 3.402823466E+38

범위의 실수들을 나타낼 수 있는 데이터 타입입니다. 참고로

-3.402823466E+38 은 (-3.402823466) X (10의 38제곱) 을 의미하고

-1.175494351E-38 은 (-1.175494351) X (10의 38제곱 분의 1) 을 의미합니다.

**3) DOUBLE**

:

-1.7976931348623157E+308 ~ -2.2250738585072014E-308,

0,

2.2250738585072014E-308 ~ 1.7976931348623157E+308

범위의 실수들을 나타낼 수 있는 데이터 타입입니다. FLOAT에 비해 더 넓은 범위의 수를 나타낼 수 있을 뿐만 아니라, 그 정밀도 또한 더 높은 타입입니다.(소수점 뒤에 최대로 허용가능한 자리수가 더 많음)

**2. 날짜 및 시간 타입(Date and Time Types)**

데이터베이스에서는 날짜 및 시간 정보를 다뤄야하는 경우가 정말 많습니다. 이런 정보를 다루는 데이터 타입들을 알아볼게요.

**(1) DATE**

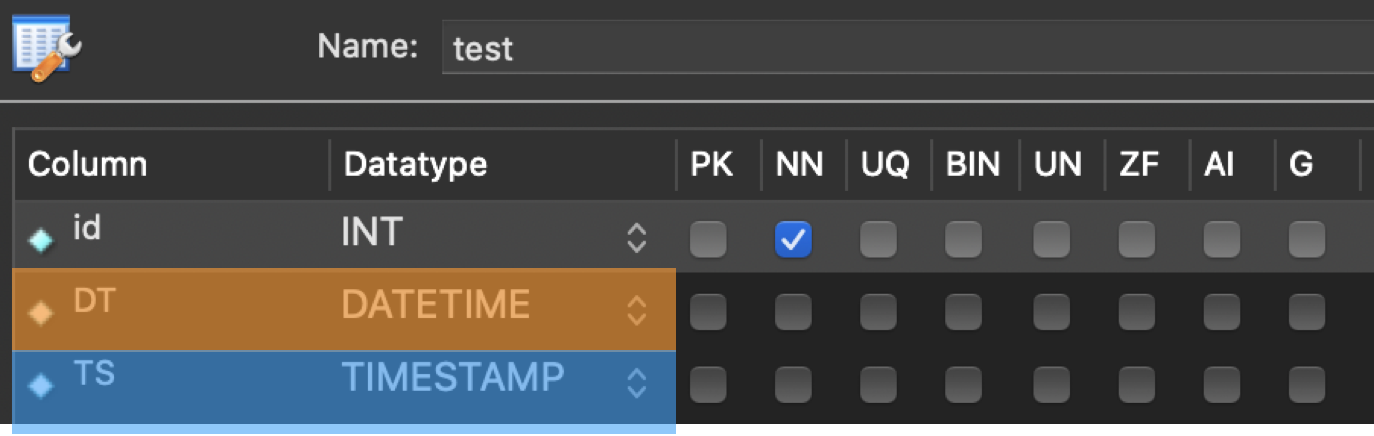
: 날짜를 저장하는 데이터 타입입니다. 날짜는 ’2020-03-26’ 이런 형식의 연, 월, 일 순으로 값을 나타냅니다.

**(2) DATETIME**

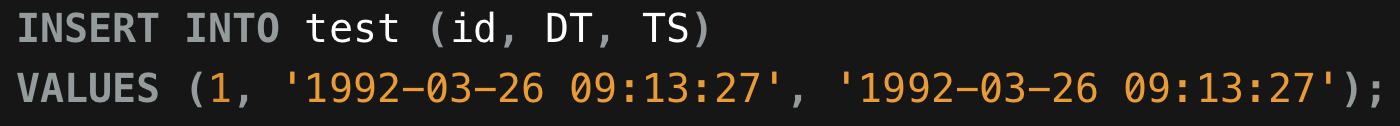
: 날짜와 시간을 저장하는 데이터 타입입니다. ’2020-03-26 09:30:27’ 이런 식으로 연, 월, 일, 시, 분, 초를 나타냅니다.

**(3) TIMESTAMP**

: 날짜와 시간을 저장하는 데이터 타입입니다. ’2020-03-26 09:30:27’ 이런 식으로 연, 월, 일, 시, 분, 초를 나타냅니다. 그럼 DATETIME 타입과는 어떤 점이 다를까요? TIMESTAMP 타입은 타임 존(time\_zone) 정보도 함께 저장한다는 점이 다릅니다. 무슨 말인지 간단한 테스트를 통해 바로 보여드릴게요. 아래와 같이 DATETIME 타입의 컬럼(DT 컬럼) 하나와, TIMESTAMP 타입의 컬럼(TS 컬럼) 하나가 있는 테이블을 만들게요.



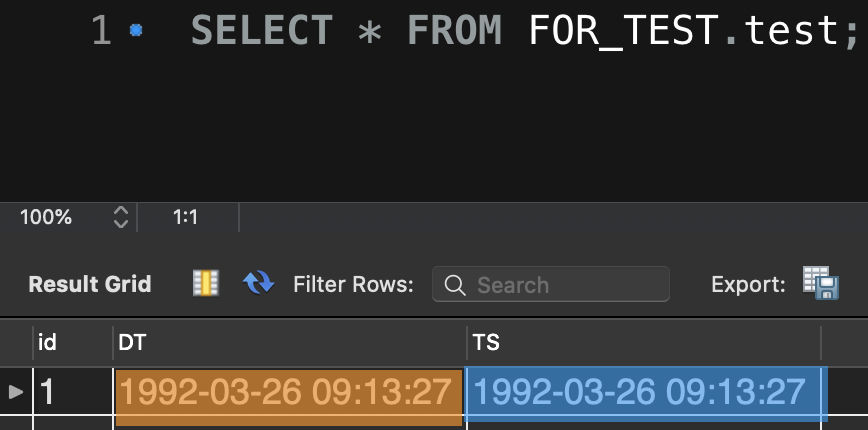
이렇게 test라는 테이블을 만들고,



이 SQL 문을 실행해서 row 하나를 추가할게요. 이 문장은 앞으로 배우게 될 INSERT 문으로 테이블에 row를 추가하는 SQL 문입니다.

저는 두 컬럼에 같은 날짜/시간 값을 넣어줬습니다.

확인해보면,



동일한 날짜/시간 값이 저장된 것을 확인할 수 있습니다.

자, 이제 두 타입의 차이점을 알아볼게요.

그 전에 **UTC**라는 개념을 알아야 합니다. 전세계에 여러 **시간대(타임존, time-zone)**가 존재한다는 사실은 알고 계시죠? 각각의 시간대는 UTC를 기준으로 나타낼 수 있는데요. **UTC(Coordinated Universal Time)**는 국제 사회에서 통용되는 표준 시간 체계로 '국제 표준시'라고도 합니다. UTC는 영국 런던을 기준으로 하는데요. 우리나라는 런던을 기준으로 시간이 9시간 더 빠르기 때문에, 우리나라의 시간을 표현할 때는 **UTC+9**라고 표현합니다.

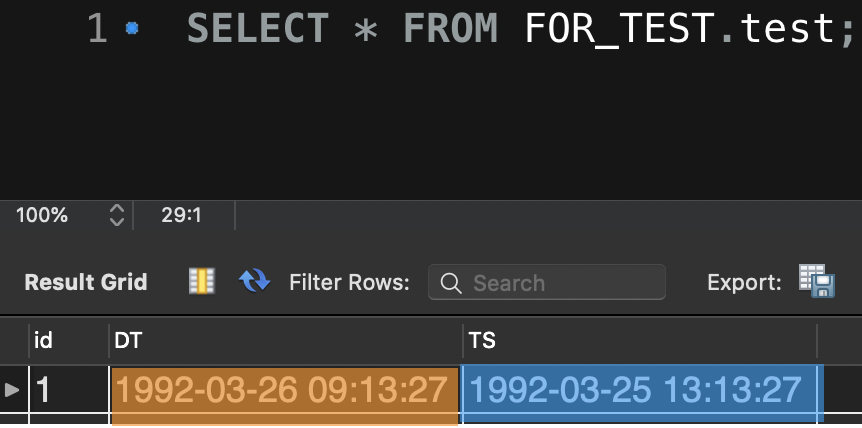
그러니까 위에서 제가 저장한 날짜 시간 값은 이미 **UTC+9**에 해당하는 값인 거죠.

이 상태에서 MySQL 서버의 시간대 설정을 한번 바꿔보겠습니다.

https://bakey-api.codeit.kr/files/3263/CWn5fo?name=d.png

이걸 실행하면, 시간대가 **UTC-11**로 바뀝니다. 지구상에서 이 시간대에는 [니우에](https://ko.wikipedia.org/wiki/%EB%8B%88%EC%9A%B0%EC%97%90)라는 나라가 존재합니다.

다시 test 테이블을 조회해보면



DT 컬럼의 값은 그대로인데, TS 컬럼의 값은 원래 시간보다 **20시간 이전인 값**으로 바뀌어 있는 것을 알 수 있습니다.

원래 **UTC+9**인 시간대에서 **UTC-11**인 시간대로 바뀌었기 때문에 +9와 -11의 차이대로 20시간 차이가 나는 겁니다.

이렇게 Datetime 타입과 Timestamp 타입은 타임 존 정보 저장 여부에서 차이가 있습니다.

만약 타임 존 정보를 굳이 함께 저장할 필요가 없다면 Datetime 타입을, 타임 존 정보도 함께 저장하고 싶다면 Timestamp 타입을 설정하면 되겠죠?

**(4) TIME**

: 시간을 나타내는 데이터 타입입니다. ’09:27:31’ 형식으로 ‘시:분:초’를 나타냅니다.

**3. 문자열 타입(String type)**

문자열을 저장하기 위한 타입입니다. 이름, 댓글, 구매후기 등 문자열 형태의 데이터는 정말 다양하죠? 아래와 같은 타입들이 있습니다.

**(1) CHAR**

: 문자열을 나타내는 기본 타입으로 Character의 줄임말입니다. CHAR(30), 이런 형식으로 나타내는데요. 괄호 안의 숫자는 문자를 최대 몇 자까지 저장할 수 있는지를 나타냅니다. 30이라고 써있으면 최대 30자의 문자열을 저장할 수 있다는 뜻입니다. CHAR 타입의 괄호 안에는 0부터 255까지의 숫자를 적을 수 있습니다.

**(2) VARCHAR**

: VARCHAR도 VARCHAR(30) 이런 식으로 문자열의 최대 길이를 지정할 수 있는 문자열 타입입니다. 괄호 안에 최소 0부터 최대 65,535 (2**16** − 1)를 쓸 수 있습니다. 그런데 VARCHAR는 CHAR보다 허용되는 최대 저장 길이가 더 크다는 점 말고 다른 차이점도 있습니다.

그건 바로 **CHAR는 고정 길이 타입**이고, **VARCHAR는 가변 길이 타입**이라는 점입니다. 사실 VARCHAR라는 단어 자체가 Character Varing의 줄임말로 가변 문자열을 나타냅니다. 쉽게 설명하자면, CHAR(10), VARCHAR(10)이 있을 때,

CHAR(10)은 어떤 길이의 문자열이 저장되더라도 항상 그 값이 10만큼의 저장 용량을 차지합니다. 하지만 VARCHAR(10)의 경우 만약 값이 ‘Hello’ 이런 5자라면 저장 용량도 5만큼 차지합니다. **저장 용량이 설정된 최대 길이에 맞게 고정되는 게 아니라 실제 저장된 값에 맞게 최적화되는 거죠.** 대신 VARCHAR 타입으로 값이 저장될 때는 해당 값의 사이즈를 나타내는 부분(1byte 또는 2byte)이 저장 용량에 추가됩니다.

따라서 값의 길이가 크게 변하지 않을 컬럼에는 CHAR 타입을 사용하고, 길이가 들쑥날쑥할 컬럼에는 VARCHAR 타입을 쓰는 게 좋습니다.

**(3) TEXT**

문자열을 저장하는 데이터 타입으로 최대 65535 자까지 저장할 수 있습니다. 이외에도 16,777,215 (2**24** − 1) 자까지 저장할 수 있는 MEDIUMTEXT, 4,294,967,295(2**32** − 1) 자까지 저장할 수 있는 LONGTEXT 타입이 있습니다.

VARCHAR 타입과 TEXT 계열의 타입은 내부 구현에서 일부 차이가 있는데요. 당장 이해하기에는 어려운 내용이기 때문에 생략하겠습니다. 일단은 정말 길이가 긴 문자열을 저장하려면 TEXT 계열의 타입을 써야하는구나 정도만 기억해주세요.

자, 이때까지 MySQL의 주요 데이터 타입(Data type)들을 살펴봤는데요. 각 컬럼에 적합한 데이터 타입을 설정하기 위해서는 각 DBMS의 매뉴얼 페이지로 각 데이터 타입에 대해 정확하게 공부해야 합니다. 하지만 기초 단계에서는 이 정도만 알아도 충분하니까 위의 내용들을 잘 기억하고 넘어가주세요.