:: DataBase\_MySQL ::

Relational Database (RDB – 관계형 DB)

Relational model (DB의 가장 흔한 model)

excel에서의 sheet가 : table (row + column)

rows : records, tuples

excel파일에 있는 여러 개의 sheet==table -> table들의 모임 == DB -> DB를 access하여 외부 요청에 의해 R/W하기 위한 DBMS

column + row -> Table -> DB

DBMS (Data Base Management System) : DB + Instance

SQL (Structured Query Language) : 문법이 한정된 , 질문, 언어

* Domain-specific language in a RDBMS

Create DB : excel file을 만드는 것

Create Table : excel file에서 sheet를 만드는 것

Insert Data : excel sheet에 data를 적는 것 ( Write )

Retrieve Data : data를 display ( Read )

MySQL ->MySQL Connection -> Advanced -> Others -> useSLL=0 (Secure Socket Layer)

DBMS와 user랑 작업할 때 데이터를 암호화할 것인지? 안 하려면 useSLL=0

SCHEMA 만들기 : Create a New schema in the connected server -> name설정, Charset/Collation 설정(utf8(어떤 문자열로 할건지), utf8\_general\_ci(문자열을 정렬할 때 정렬할 순서))

DBMS version check : Administration -> Server Status

대소문자 구별 판별 (Case sensitive? Not Case sensitive?)

Administration -> Status and System Variables -> System Variables-> “lower” 검색 -> lower\_case\_table\_names 의 value 확인

* Value 0 : 대소문자 구분을 한다 (A테이블과 a테이블은 다르다)
* Value 1 : 대소문자 구분을 안 한다 (A테이블과 a테이블이 같다) (보통 Windows에서)

모든 이름을 소문자로 저장

* Value 2 : 1과 마찬가지로 대소문자 구분을 안한다.

저장할 때는 대소문자를 구분, 비교할 때는 대소문자 구분X

TINYINT : 1 Byte

SMALLINT : 2 Bytes

MEDIUMINT : 3 Bytes

INT : 4 Bytes

BIGINT : 8 Bytes

FLOAT : 4 Bytes

DOUBLE : 8 Bytes

TIMESTAMP : and time in UTC (korea의 timezone만 알면 된다. 간단)

DATETIME : timezone에 대한 정보가 없어서 이곳이 어떤 timezone인지 부가적인 정보가 필요

Rid : 고유번호

UN : Unsigned (0 or 양수)

NN : Not NULL (체크하면 누락되지 않게 정보를 채울 수 있다)

AI : Auto Increment (자동으로 증가 : 고유번호 1번째 데이터 설정하면 쭉 자동으로 채워지는)

PK : Primary Key (각각의 row를 unique하게 구분할 수 있게 하는 고유식별자)

ZF : Zero Fill (0으로 채운다 -> INT(3) - 001)

* Create table for temperature from buildings

Building, temperature, time

Building id (int) 🡪 building\_id

Temperature (float) 🡪 temperature

Time(UTC) 🡪 reg\_date

만들어진 SCHEMA의 Tables에 Create Table -> Table Name, Charset/Collation, Engine, Column Nmae 등 설정 후 Apply -> successfully 확인

Building id, temperature, measured date and time

1 , 10 , 2021-12-05 13:00:00

1 , 12 , 2021-12-05 14:00:00

* Data Insertion (Write) : manual insertion

1. 만들어진 Table에 Select Rows 🡪 Insert new row 🡪 데이터 기입 후 Apply
2. 만들어진 Table에 Select Rows 🡪 SQL

INSERT INTO [테이블명](변수1, 변수2, 변수3)

VALUES(값1, 값2, 값3);

Ex) insert into building\_temperature(building\_id,temperature,reg\_date)

values(8,15,'2021-12-06 13:00:00');

(multiple data insertion)

Iinsert into building\_temperature(building\_id,temperature,reg\_date)

values (9,15,'2021-12-06 13:00:00'), (10,15,'2021-12-06 13:00:00'),(11,15,'2021-12-06 13:00:00');

* Data Read (SQL query select)

(전체 data를 Read하고 싶으면 , 전체 : \*)

1. Default SHEMA를 설정 후 -> SELECT \* FROM [Table명]; 입력 후 번개표시
2. SELECT \* FROM [schema명].[table명]; 입력 후 apply

(Set as Default SHEMA가 되어있다면)

SELECT \* FROM [table명];

Ex) select \* from building\_temperature;

(모든 data말고 n개 행만 보고 싶으면)

select \* from [table명] limit [n];

Ex) select \* from building\_temperature limit 4;

(Read하고 싶은 column을 지정하고 싶으면)

Select 변수1,변수2 from [table명];

Ex) select building\_id,temperature from building\_temperature;

또는 select `reg\_date`,building\_id,`temperature`,rid from `building\_temperature`;

(`은 찍어도 안찍어도 상관 X, 명확하게 하기 위해서)

(Sorting)

Select \* from [table명] order by `column명`;

Ex) select \* from building\_temperature order by `building\_id` asc; (asc : 오름차순 , defalt / desc : 내림차순)

Ex) select \* from building\_temperature order by `building\_id` desc limit 1; (가장 큰 building\_id)

Ex) select \* from building\_temperature order by `reg\_date` limit 1; (가장 과거의 reg\_date)

(다중 sorting)

1. Building\_id 오름차순 & rid 내림차순

Ex) select \* from building\_temperature order by `building\_id` asc, `reg\_date` desc;

1. 최신 날짜 & building\_id 오름차순

Ex) select \* from building\_temperature order by `reg\_date` desc, `building\_id` asc;

(조건에 맞는 data만 보여주기)

Select \* from [Table명] where [column명] [비교연산자] [값]

Ex) select \* from building\_temperature where `reg\_date` < '2021-12-06 13:30:00';

Ex) select \* from building\_temperature where `reg\_date` < '2021-12-06 13:30:00' order by `building\_id` desc;

Ex) select \* from building\_temperature order by `building\_id` desc where `reg\_date` < '2021-12-06 13:30:00'; (이건 문법적으로 맞지 않다 🡪 SQL 문법의 특징은 들어가는 곳의 위치가 정해져있다. 🡪 Select Query)

(다중 conditions)

comparison operators : >, <, >=, <=, =, !=

logical operators : and, or

Ex) select \* from building\_temperature where `reg\_date` < '2021-12-06 13:30:00' and `building\_id`=1;

Ex) select \* from building\_temperature where `reg\_date` != '2021-12-06 13:30:00' or `building\_id`=1;

Ex) select \* from building\_temperature where (`reg\_date`<'2021-12-06 13:30:00' and `temperature`>9) or(`building\_id`=1 and `temperature`<20.0);

(in operator)

or 연산을 쭉 쓰는 것과 같음

Ex) select \* from building\_temperature where `building\_id`=1 or `building\_id`=8 or `building\_id`=9;

== select \* from building\_temperature where `building\_id` in(1,8,9);

* Table cleanup (Table 선택 후 마우스 오른쪽)

Drop Table : 테이블 자체를 삭제

Trucate Table : 테이블 안의 내용을 지움

* SQL statements execution

해당라인에 커서 🡪 Ctrl + enter 🡪 해당라인만 실행

* 주석 (Comment)

1. --
2. /\*\*/

* SQL Query

Select

From

Where

Group by

Having

Order by (sorting)

* Subquery (Query 안에 Query)

select \* from building\_temperature where `temperature` >=

( select `temperature` from `building\_temperature` where `building\_id`= 1 );

Subquery의 결과가 2개 이상이 나오면 어떤 것을 반환할지 몰라서 error가 발생할 수도 있음.

따라서 all, any, some을 사용

All -> and 개념

Any, some -> or 개념

Ex) select \* from building\_temperature where `temperature` >=

all ( select `temperature` from `building\_temperature` where `building\_id`= 1 );

Ex) select \* from building\_temperature where `temperature` >= some ( select `temperature` from `building\_temperature` where `building\_id`= 1 );

* Subquery를 이용한 테이블 생성

create table temp\_table (select \* from building\_temperature where `building\_id`>1);

* building\_temperature 테이블에서 building\_id column이 1 초과인 data들을 갖고 새로운 temp\_table명의 테이블을 생성한다
* distinct

중복된 것은 한 번만 찍으시오

Ex) select distinct building\_id from building\_temperature;

* aggregate function

avg, max, min, stddev(표준편차), var\_samp(분산)

ex) select avg(`temperature`),max(temperature),min(temperature) from building\_temperature;

ex) select stddev(`temperature`),var\_samp(temperature) from building\_temperature;

ex) select stddev(`temperature`) as '온도 표준편차',var\_samp(temperature) as '온도 분산'from building\_temperature;

* as 를 이용하여 보여질때 column을 출력을 설정할 수 있음

ex) select count(\*) from building\_temperature;

ex) select count(distinct building\_id) from building\_temperature;

* count를 이용하여 building\_temperture 테이블에 몇 개의 data가 있는지 확인
* QUIZ : 빌딩 1의 평균온도보다 같거나 높았던 빌딩들을 모두 출력하시오

select \* from building\_temperature where temperature >=(select avg(temperature) from building\_temperature where building\_id=1) and building\_id!=1;

* Group by : group 별로 출력하기

Ex) select building\_id, avg(temperature) from building\_temperature group by building\_id;

* 빌딩별로 평균온도 출력
* Having : aggregate function이 들어간 조건을 지정할 때. where에서는 aggregate func 사용이 어렵기 때문

Ex) select building\_id, avg(temperature) from building\_temperature group by building\_id having avg(temperature)>12.0;

* 평균온도가 12보다 높았던 빌딩 번호를 알고 싶을 때
* Rollup : 전체에 대한 평균 (total의 개념)

Ex) select building\_id, avg(temperature) from building\_temperature group by building\_id

with rollup;

* Table creation

Use (schema명)

Create table (만들 table명)

(

(column name) (column data type) (column 속성)

);

Insert into (만들 table명) values(column1 값, …, column?값);

Ex)

use firstdatabase;

create table building\_info

(

rid int auto\_increment primary key,

building\_id smallint unique,

building\_name varchar(100)

);

insert into building\_info values(null,1,'연구동');

insert into building\_info values(null,3,'기숙사');

insert into building\_info values(null,8,'식당');

insert into building\_info values(null,9,'체육관');

* Table 속성 확인하기

Table 오른쪽에 info🡪DDL

* Table 속성 바꾸기

alter table `building\_info` auto\_increment=10;

insert into building\_info values(null,2,'수영장');

* Rid의 auto\_increment가 10부터 시작. 10,2,’수영장’ 으로 insert된다.

Insert into `building\_info` values(null,2,’기초연구동’);

insert into `building\_info` values(null,5,'과학연구동');

* 첫번째 줄에서 Error 발생. Building\_id인 2는 unique한 속성이 있는데, 또 2가 들어와서 error. 두번째 줄은 error가 발생하지 않지만 첫번째 줄에서 exit되기 떄문에 insert가 안됌.
* 이를 극복하기 위해 insert ignore 이용

insert ignore into building\_info values (null,4,'기초연구동');

* Row에 있는 값 수정하기 :

1. Update (table명) set (column명=’바꿀 값’) where (조건);

Ex) update `building\_info` set `building\_name`='첨단연구동' where `building\_id`=1;

1. Insert into ~ values ~ on duplicate key update ~

Ex) insert into building\_info values(null,1,'자연연구동',200)

on duplicate key update building\_name='자연연구동';

* 새로운 column 추가하기 : Alter table (테이블명) add (새로운 column명) (data type);

alter table `building\_info`

add max\_capacity smallint unsigned;

update `building\_info` set `max\_capacity`=200; (max\_capacit를 200으로 초기화)

update building\_info set max\_capacity=200\*2 where building\_name='식당';

* Building\_name이 식당인 곳의 max\_capacity를 2배한다
* Delete

Delete from (테이블명)

* 해당 테이블 삭제됨. 위험하기 때문에 where(조건)이랑 같이 사용한다

delete from building\_info where max\_capacity>200;

* Max\_capacity가 200 초과인 data를 지운다
* Join

Relational database의 특징이다. table끼리 연관성을 따로 분류하고 필요할 때 join하여 원하는 정보를 찾는다 🡺 relational database를 설계한다

Building\_info테이블의 building\_id와 building\_temperature테이블의 building\_id가 같은 것끼리 묶어서 테이블로 보여라

ex) select \* from building\_info join building\_temperature on building\_info.building\_id=building\_temperature.building\_id;

select building\_info.building\_name, building\_temperature.temperature, building\_temperature.reg\_date

from building\_info join building\_temperature on building\_info.building\_id=building\_temperature.building\_id;

* 글자수 줄이는 방법 alias(별명)

select a.building\_name, b.temperature,b.reg\_date

from building\_info a join building\_temperature b on a.building\_id=b.building\_id;

* On 이라는 조건에 맞는 모든 data를 join한다
* Left join, right join

Left join :

select \*

from building\_info a left join building\_temperature b on a.building\_id=b.building\_id;

* 왼쪽에 있는 것은(building\_info a) 짝이 없어도 join하지 않고 data를 보여준다
* Matching 되는 data가 없으면 null로 반환

Right join :

select \*

from building\_temperature a right join building\_info b on a.building\_id=b.building\_id;

* Multiple tables join

select a.building\_name,b.temperature,c.humidity,c.reg\_date

from building\_info a

join building\_temperature b on a.building\_id=b.building\_id

join building\_humidity c on a.building\_id=c.building\_id and b.reg\_date=c.reg\_date;

* building\_info 테이블과 building\_temperature 테이블의 building\_id가 같은 것을 join 하고
* building\_temperatrue 테이블과 building\_humidty 테이블의 building\_id가 같은 것을 join하고
* building\_temperature 테이블과 building\_humidity 테이블의 reg.date가 같은 것을 join
* create table

use [database명];

drop table if exits (만들 테이블명]; -- 만약 같은 테이블명이 있으면 drop table하라(옵션)

create table if not eixts (만들 테이블명] alter

(

col\_name data\_type [not null | null] [default value] [auto\_increment] [unique key|primary key] [comment 'string'],

col\_name2 data\_type [not null | null] [default value] [auto\_increment] [unique key|primary key] [comment 'string'],

...

);

use firstdatabase;

create table if not exists building\_inout

(

rid int auto\_increment primary key comment '데이터 일련번호',

building\_id smallint unsigned comment '건물 고유번호, building\_info table에서 사용되는 값',

reg\_date timestamp default current\_timestamp comment '측정시간',

num\_in smallint unsigned default 0 comment '건물 입장 인원',

num\_out smallint unsigned default 0 comment '건물 퇴장 인원',

sensor\_id varchar(20) null comment '센서 이름',

sensor\_hash varchar(40) unique comment '센서 해쉬값, 데이터 정확성, 변조불가성 (보안강화)'

);

* unique key VS primary key

Unique key : unique key로 설정한 column의 date들은 서로 겹칠 수 없다

Primary key와 다르게 null을 넣을 수 있다

Ex) column명이 도시이름이면 data들 중 ‘서울’을 2개 이상 insert할 수 없다

Primary key : unique key와 똑같지만 null을 넣을 수 없고, 무조건 값이 들어가야 한다.

* 새롭게 익힌 함수들

IF(조건문, 참일 때의 값, 거짓일 때의 값)

* SELECT IF(2 > 1, 'TRUE', 'FALSE') AS result

해당 column이 null일 경우 대체하여 read한다

* SELECT IFNULL(Column명, "Null일 경우 대체 값") FROM 테이블명;

특정 문자를 포함하는 데이터 검색

* SELECT [필드명] FROM [테이블명] WHERE [필드명] LIKE '%특정 문자열%';

여러 개의 특정 문자를 포함하는 데이터 검색

* SELECT [필드명] FROM [테이블명] WHERE [필드명] REGEXP '특정 문자열|특정 문자열2';

Case when

* SELECT

CASE

WHEN permission\_type\_id = 1

THEN '관리자'

WHEN permission\_type\_id = 2

THEN '판매자'

ELSE '일반 유저'

END

Date format 변경하여 Read하기

2021/01/24 12:33:32로 기입되어 있는 항목에서 필요한 값만 뽑아내고 싶은 상황

* SELECT DATE\_FORMAT("2021/01/24 12:33:32", "%Y/%m/%d");