데이터베이스

데이터베이스, DBMS

- Database : 컴퓨터시스템에 저장된 구조화된 데이터
  데이터의 체계적인 집합
- DBMS(Database Management System) :DataBase(DB)를 관리하는 시스템

- SQL(Structured Query Language) : 데이터베이스에서 데이터를 정의, 조작, 제어하기 위해 사용하는 언어

  DDL : Data Definition Language

  create, alter, drop

  DML : Data Manipulation Language

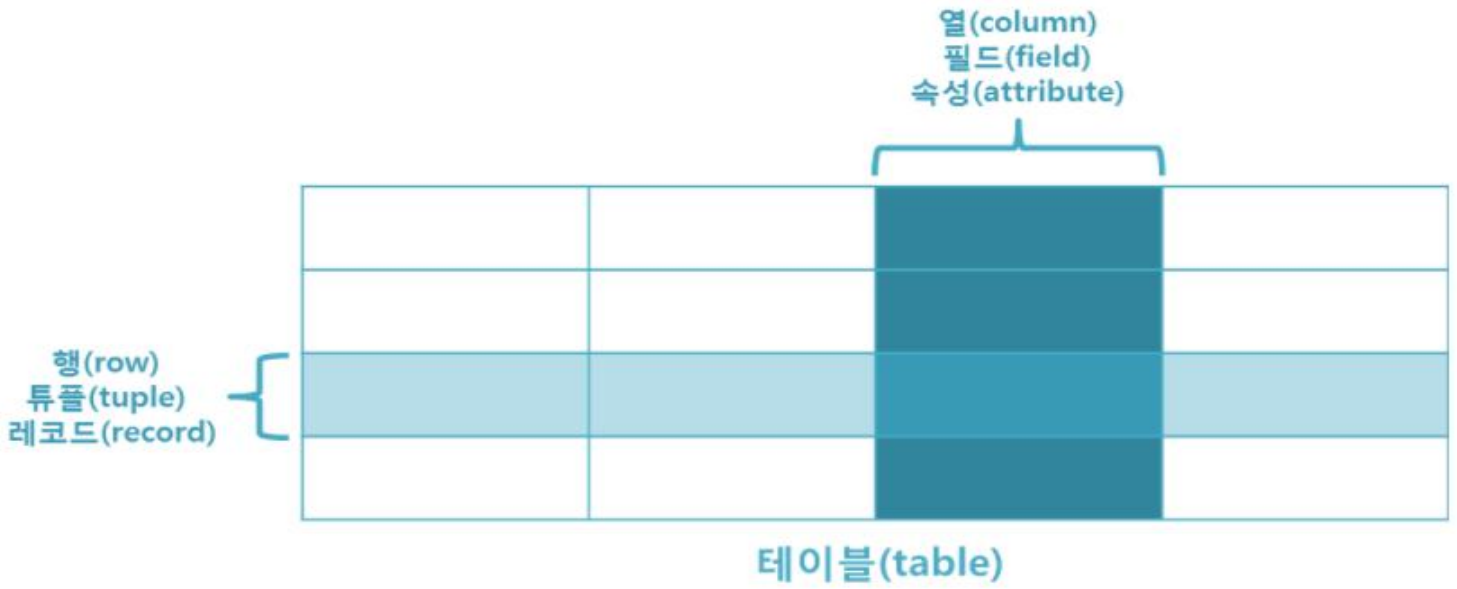
  insert update delete select

  DCL: Data Control Language

  grant revoke

RDBMS

- RDB (relational Database) : 상호 관련된 데이터포인트에 대한 엑세스를 저장하고 제공하는 데이터베이스 형태
- 관계형 데이터베이스는 데이터를 테이블에 직관적으로 나타내는 관계형 모델을 기반으로 함- 여러 개 테이블에 걸쳐있는 데이터 사이의 관계
- RDBMS : ORACLE, MySQL, MSSQL, MariaDB, SQLLite



Mysql 설치

◆ mysql.com

=> mysql community server

=> window x86, 32&64bit insaller msi 설치

mysql server, workbench, samples and examples 설치

https://dev.mysql.com/doc/index-other.html =>sampledb

https://github.com/datacharmer/test_db

다운받은 zip 파일을 c:₩employees폴더로 사용한다.

압축풀은 폴더에서

mysql 접속

mysql –u root –p

패스워드

source c:/employees.sql 경로 ₩->/로 변경 사용

◆ **데이터베이스 목록**

show databases;

◆ **테이블 확인**

show tables;

◆ **테이블 구조 확인**

desc 테이블명

◆ **desc employees;**

```
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| emp_no     | int           | NO   | PRI | NULL    |       |
| birth_date | date          | NO   |     | NULL    |       |
| first_name | varchar(14)   | NO   |     | NULL    |       |
| last_name  | varchar(16)   | NO   |     | NULL    |       |
| gender     | enum('M','F') | NO   |     | NULL    |       |
| hire_date  | date          | NO   |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
```

데이터검색

◆ SELECT select_expr --필수
[FROM table_references] --필수
[WHERE where_condition]
[GROUP BY {col_name |expr |position}]
[HAVING where_condition]
[ORDER BY {col_name |expr |position}]

◆ select... from

1. 모든 열 검색

```
select * from employees
```

2. 컬럼 중복 제거

```
select distinct gender from employees
```

3. 컬럼명 별칭 부여

```
select emp_no, first_name as fname, last_name as lname
from employees;
```

4. 다양한 함수 사용

```
select emp_no, concat(first_name,' ', last_name) as name, birth_date
from employees;
```

https://dev.mysql.com/doc/refman/8.0/en/sql-function-reference.html

select... from.. where

◆ selct 컬럼명
from 테이블명
where 조건

술어	연산자	예
비교	=, <>, <, <=, >, >=	salary < 20000
범위	BETWEEN	salary BETWEEN 70000 AND 75000
집합	IN, NOT IN	price IN (74333, 75286, 75994)
패턴	LIKE	first_name LIKE ‘G%'
NULL	IS NULL, IS NOT NULL	salary IS NULL
복합조건	AND, OR, NOT	(salary< 75000) AND (first_name LIKE ‘G%')

select... from.. where

◆ 조건이 숫자일때

```
select emp_no, first_name
from employees
where emp_no=10001;
```

```
select emp_no, first_name as fname, last_name as lname
from employees
 where emp_no<=10010;
```

◆ 조건이 문자나 날짜일때

```
select emp_no, concat(first_name,' ', last_name) as name, birth_date
from employees
where birth_date='1961-12-04';
```

```
select emp_no, first_name
from employees
where first_name='Georgi'; --조건문의 값은 대소문자 구분
```

https://dev.mysql.com/doc/refman/8.0/en/sql-function-reference.html

select... from.. where

◆ and연산, between 연산

```
select emp_no, first_name
from employees
where emp_no>=10010 and emp_no<=10020;

select emp_no, first_name
from employees
where emp_no between 10010 and 10020;

 select emp_no, first_name,hire_date
 from employees
 where hire_date>='1985-02-02' and hire_date<='1985-02-05'
 order by hire_date;

 select emp_no, first_name,hire_date
 from employees
 where hire_date between '1985-02-02' and '1985-02-05'
 order by hire_date;
```

select... from.. where

◆ or(또는) , in 연산

```
select emp_no, first_name
from employees
 where emp_no=10010 or emp_no=10020 or emp_no=10030;

select emp_no, first_name
from employees
where emp_no in (10010, 10020, 10030);

select emp_no, last_name
from employees
where last_name='Facello' or last_name='Simmel';

select emp_no, last_name
from employees
where last_name in ('Facello', 'Simmel');
```

select... from.. where

◆ like 연산

와일드 문자	의미	사용 예
+	문자열을 연결	' ' + '바이블' : '골프 바이블'
%	0개 이상의 문자열과 일치	'%축구%' : 축구를 포함하는 문자열
[ ]	1개의 문자와 일치	'[0-5]%' : 0-5 사이 숫자로 시작하는 문자열
[^]	1개의 문자와 불일치	'[^0-5]%' : 0-5 사이 숫자로 시작하지 않는 문자열
_	특정 위치의 1개의 문자와 일치	'_구%' : 두 번째 위치에 '구'가 들어가는 문자열

◆
```
select emp_no, last_name
from employees
where last_name like 'C%'; -- C로 시작하는

select emp_no, last_name
from employees
where last_name like '%v';  --v로 끝나는

select emp_no, last_name
from employees
where last_name like '%v%'; -- v가 포함되는
```

select... from.. where

◆ select emp_no, first_name
from employees
where first_name like '_o__'; 전체 4글자 중 두번째 글자가 o인

◆ select emp_no, first_name
from employees
where first_name like '%o%'; -- o가 포함된

◆ is null 연산 : 컬럼안에 null인 자료만 (반대 경우 is not null)

select... from.. where

실습) is null과 null에 대한 실습

현재 table에는 null값이 없음
따라서 새로운 테이블을 만들고 자료를 입력하여 is null test

```
  create table sungjuck(
     id int primary key auto_increment
     ,name varchar(10)
     , kor int
     , hak int
      ,ban int
   );
insert into sungjuck(name, kor, hak,ban) values('a',100,1,1);
 insert into sungjuck(name, kor,hak,ban)  values('b', null,1,1);
 insert into sungjuck(name,hak, ban) values('c',1,1);
 insert into sungjuck(name,kor, hak,ban) values('d', 50,1,2);
 insert into sungjuck(name,kor, hak,ban) values('e', 30,1,3);
 insert into sungjuck(name,kor,hak,ban) values('f', 60,2,1);
 insert into sungjuck(name,kor,hak,ban) values('g', 70,2,1);
 insert into sungjuck(name,kor,hak,ban) values('h', 30,2,2);
```

select... from.. where , limit

- select * from sungjuck;
  select * from sungjuck where kor is null;
  select * from sungjuck where kor is not null;

- select emp_no, first_name, last_name
  from employees
  limit 0,5;

- select emp_no, first_name, last_name
  from employees
  limit 10,20;

- select emp_no, first_name, last_name
  from employees
  order by first_name desc
  limit 10,20;

정렬

◆ SELECT select_expr
[FROM table_references]
[WHERE where_condition]
[ORDER BY {col_name |expr |position}]

◆ select emp_no, last_name
from employees
order by emp_no; --오름차 정렬

◆ select emp_no, last_name
from employees
order by emp_no desc ; --내림차 정렬

◆ select emp_no, last_name, hire_date
from employees
where emp_no<=100000
order by last_name desc, hire_date asc ; --(asc 생략가능)

다양한 함수사용

◆ 숫자관련함수

ceil(x) : 내림

select ceil(10.2), ceil(10.7), ceil(-10.2), ceil(-10.7) from dual;

abs(x):절대값 abs(-10), abs(10)

sin(x) : 싸인, cos(x):코사인, tan(x) :탄젠트 x는 라디안값

div()

mod() pi() pow() round()...

https://dev.mysql.com/doc/refman/8.0/en/sql-function-reference.html

집계함수

집계 함수	문법	사용 예
SUM	SUM([ALL | DISTINCT] 속성이름)	SUM(salary)
AVG	AVG([ALL | DISTINCT] 속성이름)	AVG(salary)
COUNT	COUNT({[[ALL | DISTINCT] 속성이름] | *})	COUNT(*)
MAX	MAX([ALL | DISTINCT] 속성이름)	MAX(salary)
MIN	MIN([ALL | DISTINCT] 속성이름)	MIN(salary)

◆ **select sum(kor), avg(kor), min(kor)**
, max(kor), count(kor), count(*)
from sungjuck;
-- sungjuck테이블의 국어총점, 평균, 최소값, 최대값, 국어인원, 전체인원을 출력

◆ **select sum(kor) , avg(kor) , sum(kor)/count(*), sum(kor)/count(kor)**
from sungjuck;

집계함수

◆ group by : 그룹별 집계를 구함
having : 그룹에 관한 조건 구할때

◆ select hak, avg(kor)
from sungjuck
group by hak;

◆ select hak,ban, avg(kor)
from sungjuck
group by hak,ban;

```
select hak, ban, avg(kor)
from sungjuck
group by hak, ban
having hak=1;
```

```
select hak, ban, avg(kor)
from sungjuck
where hak=1
group by hak, ban;
```

집계함수

◆ select hak, ban, avg(kor)
from sungjuck
group by hak, ban
having avg(kor)>=60;

◆ select hak, ban,avg(kor)
from sungjuck
group by hak, ban
order by ban, hak;

조인

Field	Type	Null	Key	Default	Extra
emp_no	int	NO	PRI	NULL	
birth_date	date	NO		NULL	
first_name	varchar(14)	NO		NULL	
last_name	varchar(16)	NO		NULL	
gender	enum('M','F')	NO		NULL	
hire_date	date	NO		NULL	

employees

Field	Type	Null	Key	Default	Extra
dept_no	char(4)	NO	PRI	NULL	
dept_name	varchar(40)	NO	UNI	NULL	

departments

Field	Type	Null	Key	Default	Extra
emp_no	int	NO	PRI	NULL	
dept_no	char(4)	NO	PRI	NULL	
from_date	date	NO		NULL	
to_date	date	NO		NULL	

dept_emp

join-내부조인
left outer join –왼쪽 조인
right outer join-오른쪽 조인

명령	문법	설명
일반 조인	SELECT <속성들> FROM 테이블1, 테이블2 WHERE <조인조건> AND <검색조건>	SQL 문에서는 주로 동등조인을 사용함. 두 가지 문법 중 하나를 사용할 수 있음.
	SELECT <속성들> FROM 테이블1 INNER JOIN 테이블2 ON <조인조건> WHERE <검색조건>	
외부조인	SELECT <속성들> FROM 테이블1 {LEFT |RIGHT |FULL [OUTER]} JOIN 테이블2 ON <조인조건> WHERE <검색조건>	외부조인은 FROM 절에 조인 종류를 적고 ON을 이용하여 조인조건을 명시함.

조인

◆ select employees.emp_no, first_name, dept_no, dept_emp.emp_no
from employees join dept_emp
on employees.emp_no=dept_emp.emp_no; --(ansi표준)

◆ select employees.emp_no, first_name, dept_no, dept_emp.emp_no
from employees,dept_emp
where employees.emp_no=dept_emp.emp_no;

◆ select e.emp_no, first_name, d.dept_no,de.dept_no,dept_name, de.emp_no
from employees e join dept_emp de
on e.emp_no=de.emp_no
join departments d
on de.dept_no=d.dept_no; -- ansi표준

◆ select employees.emp_no, first_name, dept_emp.dept_no
, departments.dept_no, dept_name, dept_emp.emp_no
from employees,dept_emp,departments
where employees.emp_no=dept_emp.emp_no
and dept_emp.dept_no=departments.dept_no;

서브쿼리

◆ select emp_no, first_name, hire_date
from employeeswhere emp_no =(select emp_no
from salaries
where salary=60098 and
from_date='2000-08-02' and to_date='2001-08-02')

◆ select emp_no, first_name, hire_date
from employees
where emp_no in (select emp_no
from dept_emp
where dept_no='d003');

◆ select e1.emp_no, first_name, (select avg(s2.salary)
from salaries s2
where e1.emp_no=s2.emp_no)
, salary,from_date, to_date
from employees e1 join salaries s
on e1.emp_no=s.emp_no;

서브쿼리

◆ select empno, fname, sal
from (
select employees.emp_no as empno
, first_name as fname, salary as sal
from employees join salaries
on employees.emp_no=salaries.emp_no
) t;

데이터 추가하기

◆ insert into 테이블명 values(컬럼값1, 컬럼값2, 컬럼값3)
-- 테이블에 있는 모든 컬럼의 값을 다 입력해야 한다.

◆ Insert into 테이블명 values(컬럼명1, null, 컬럼값3)
-- 2번째 컬럼의 값을 null 처리

◆ insert into 테이블명(컬럼명1, 컬럼명2)
values(컬럼값1, 컬럼값2)
-- 테이블 안에 컬럼명1, 컬럼명2개만 입력할때 사용

```
insert into sungjuck(name, kor, hak,ban) values('t1',100,3,1);
insert into sungjuck(name, kor,hak,ban)  values('t2', null,3,1);
insert into sungjuck(name,hak, ban) values('t3',3,1);
```

데이터변경과 삭제

◆ update 테이블명
  set 변경컬럼명1=변경값1
    ,변경컬럼명2= 변경값2
 where 조건 – select 조건과 동일

Sungjuck

	id	name	kor	hak	ban
	1	a	100	1	1
	2	b	NULL	1	1
	3	c	NULL	1	1
	4	d	50	1	2
	5	e	30	1	3
	6	f	60	2	1
	7	g	70	2	1
	8	h	30	2	2
	NULL	NULL	NULL	NULL	NULL

```
update sungjuck
set name='hong'
    , kor=50
where id=8;
```

◆ delete
  from 테이블명
  where 조건

테이블 생성

```
CREATE TABLE 테이블이름
 ( { 속성이름 데이터타입
     [NOT NULL | UNIQUE | DEFAULT 기본값 | CHECK 체크조건]
    }
     [PRIMARY KEY 속성이름(들)]
     {[FOREIGN KEY 속성이름 REFERENCES 테이블이름(속성이름)]
    [ON DELETE [CASCADE | SET NULL]
     }
 )
```

auto_increment : 초기값1 부터 1씩 자동증가

테이블생성

◆ 데이터타입

데이터 타입	설명	ANSI SQL 표준 타입
INTEGER INT	4바이트 정수형	INTEGER, INT SMALLINT
NUMERIC(m,d) DECIMAL(m,d)	전체자리수 m, 소수점이하 자리수 d를 가진 숫자형	DECIMAL(p, s) NUMERIC[(p,s)]
CHAR(n)	문자형 고정길이, 문자를 저장하고 남은 공간은 공백으로 채운다.	CHARACTER(n) CHAR(n)
VARCHAR(n)	문자형 가변길이	CHARACTER VARYING(n) CHAR VARYING(n)
DATE	날짜형, 연도, 월, 날, 시간을 저장한다.	

https://dev.mysql.com/doc/refman/8.0/en/numeric-type-syntax.html

테이블 생성

```
create table t1(
  tno int primary key auto_increment
  ,tname varchar(10)
   , age int    default 0 check(age>=0 and age<=150)
  , addr varchar(20)   not null default 'seoul'
  , phone varchar(12)  unique
);
```

제약조건 test

```
insert into t1(tname, age) values('a',10);
insert into t1(tname, age) values('a',200);
insert into t1(tname, phone) values('a','1111');
insert into t1(tname, phone) values('t','1111');
```

테이블 구조 수정, 삭제

◆ ALTER TABLE 테이블이름

[ADD 속성이름 데이터타입]

[DROP COLUMN 속성이름]

[MODIFY 속성이름 데이터타입]

[MODIFY 속성이름 [NULL | NOT NULL]]

[ADD PRIMARY KEY(속성이름)]

[[ADD | DROP] 제약이름]

◆ 테이블 삭제

DROP TABLE 테이블이름

aws 클라우드 접속하기

◆ Aws 가입,

◆ Mysql 서버 셋팅

=>휴대폰 google authenticator설치

=> 내보안 자격증명 MFA 셋팅

◆ 파라미터 그룹

characterset 셋팅

characer_set_* : utf8

collation_* : utf8_general_ci, utf8mb4

default_collation_for_utf8mb4 :utf8mb4_general_ci

◆ aws : RDS 셋팅

파라미터 그룹 셋팅 포함하여

◆ Myql 클라이언트에서 RDS접속하기