
SESSION #1

# SQL 시작하기

By EDU Team 2nd ( Jaehoon )

© 2018. SNU Growth Hackers all rights reserved

# 목차 소개


## Chapter 01 MySQL 설치하기

## Chapter 02 데이터베이스와 SQL

## Chapter 03 테이블에서 데이터 검색

3.1 검색 조건 지정하기
3.2 패턴 매칭에 의한 검색

## Chapter 04 정렬과 연산

4.1 정렬
4.2 제한
4.3 수치연산

## Chapter 05 실습

5.1 PyMySQL 맛보기
5.2 과제


© 2018. SNU Growth Hackers all rights reserved

# Chapter 1

# Chapter 1.0 MySQL 설치하기

## MySQL 설치하기

Slack에 올린 자료 참고하세요.
MySQL 5.7 버전을 설치하면 됩니다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2

# Chapter 2.0 데이터베이스와 SQL

## 데이터베이스 (DataBase)

데이터란 컴퓨터 안에 기록되어 있는 숫자를 의미하며, 이러한 데이터의 집합을 데이터베이스라고 한다.

데이터베이스를 효율적으로 관리하는 소프트웨어
→ DBMS (데이터베이스 관리 시스템; Database Management System)

데이터베이스는 다양한 종류가 있다. 계층형 데이터베이스, 관계형 데이터베이스, 키-벨류 스토어 등…
그 중 관계형 데이터베이스를 관리하는 소프트웨어 (RDBMS)를 조작할 때 사용하는 언어가 SQL 입니다.

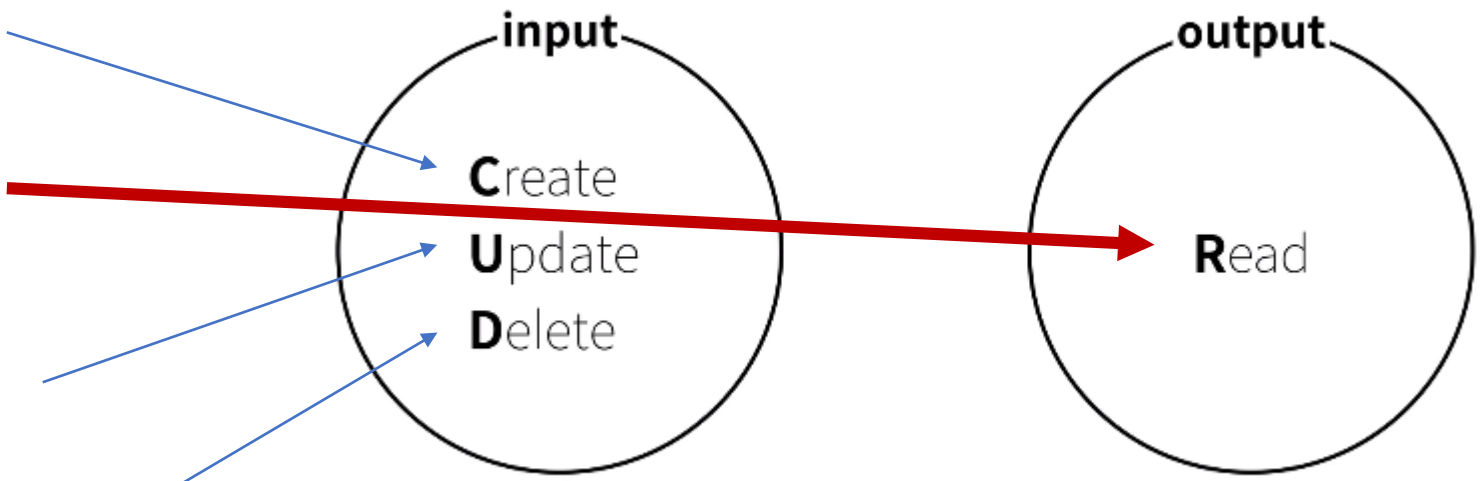
그리고 이 RDBMS 형태의 다양한 소프트웨어 제품이 있습니다. 그 중 하나가 MySQL 입니다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## 데이터베이스의 본질

**C**reate
**R**ead
**U**pdate
**D**elete




© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## 관계형 데이터베이스 구조

[표 Ⅱ-1-2] K-리그 2차 자료 정리

| 선수 | 팀 | 포지션 | 백넘버 | 생년월일 | 키 | 몸무게 | … |
|---|---|---|---|---|---|---|---|
| 박지성 | 서울FC | MF | 7 | 1981/02/25 | 178cm | 73kg | ⋮ |
| 이청용 | 블루윙즈 | MF | 17 | 1988/07/02 | 180cm | 69kg | ⋮ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |


[그림 Ⅱ-1-3] 데이터베이스의 테이블

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

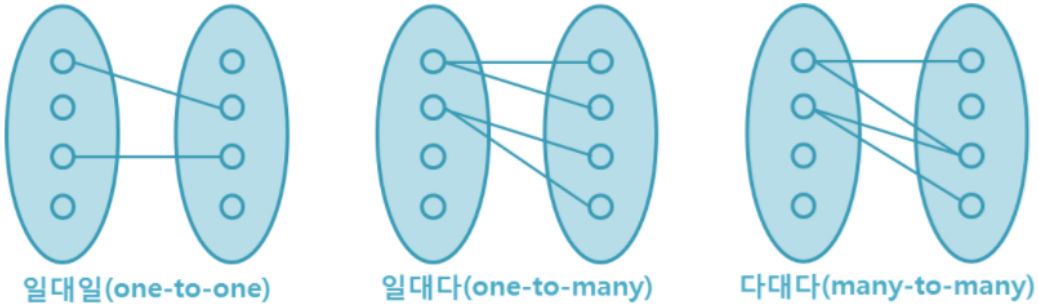
## 관계형 데이터베이스 구조

5. 관계(relationship)
테이블 간의 관계는 관계를 맺는 테이블의 수에 따라 다음과 같이 나눌 수 있습니다.

1. 일대일(one-to-one) 관계
2. 일대다(one-to-many) 관계
3. 다대다(many-to-many) 관계

관계형 데이터베이스에서는 이러한 관계를 나타내기 위해 외래 키(foreign key)라는 것을 사용합니다.
외래 키는 한 테이블의 키 중에서 다른 테이블의 행(row)을 식별할 수 있는 키를 의미합니다.

테이블 간의 관계를 그림으로 표현하면 다음과 같습니다.




© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## 관계형 데이터베이스 구조

| 부대 인원 | | | |
|---|---|---|---|
| 군번 | 이름 | 특기코드 | 소속코드 |
| 13-70000001 | 경영대 | 70110 | 1000 |
| 13-70000002 | 인문계 | 80110 | 2000 |
| 13-70000003 | 컴공과 | 40110 | 3000 |
| 13-70000004 | 회계학 | 70111 | 1001 |
| 13-70000005 | 대차표 | 70110 | 1001 |

기본키

외래키

| 부대 소속 | | |
|---|---|---|
| 소속코드 | 소속 | 부서 |
| 1000 | 본부 | 행정과 |
| 2000 | 헌병대 | 헌병소대 |
| 3000 | 정보통신중대 | 유선반 |
| 1001 | 본부 | 관리과 |

| 부대 특기 | |
|---|---|
| 특기코드 | 특기 |
| 70110 | 총무 |
| 70111 | 회계 |
| 80110 | 헌병 |
| 40110 | 유선정비 |

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## SQL 명령의 종류

DML (Data Manipulation Language)
데이터베이스에 새롭게 데이터를 추가하거나 삭제하거나 내용을 갱신하는 등, 데이터를 조작할 때 사용한다.

DDL (Data Definition Language)
데이터를 정의하는 명령어이다. 데이터베이스는 데이터 베이스 객체라는 데이터 그릇을 이용하여 데이터를 관리하는데,
이 같은 객체를 만들거나 삭제하는 명령어입니다.

DCL (Data Control Language)
데이터를 제어하는 명령어입니다. DCL에는 트랜잭션을 제어하는 명령과 데이터 접근권한을 제어하는 명령이 포함되어 있습니다.

→ SQL 명령어는 이렇게 세 종류로 나뉩니다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## MySQL에 접속하기 >>> 데이터베이스 불러오기

mysql〉 mysql –u [사용자명] –p
Enter password : [패스워드 입력]

mysql〉 SHOW DATABASES;

mysql〉 USE [데이터베이스명];
Database changed

# 세미콜론(;)은 명령이 끝났음을 알린다.
# sample 데이터베이스를 가져와보자!

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sample             |
| sampledb           |
| sys                |
| world              |
+--------------------+
7 rows in set (0.03 sec)
```

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## SELECT 구문 이해하기 : SELECT 명령은 '질의'나 '쿼리'라 부른다.

Mysql〉 SELECT [칼럼명] FROM [테이블명] ;

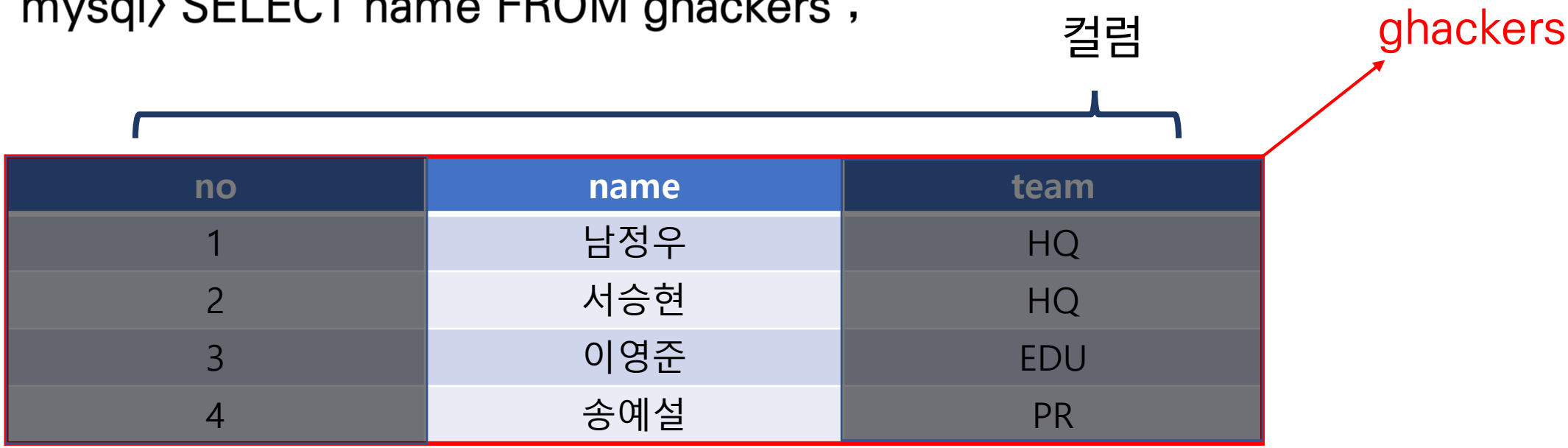

| no | name | team |
|---|---|---|
| 1 | 남정우 | HQ |
| 2 | 서승현 | HQ |
| 3 | 이영준 | EDU |
| 4 | 송예설 | PR |

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## SELECT 구문 이해하기

mysql〉 SELECT name FROM ghackers ;




| no | name | team |
|---|---|---|
| 1 | 남정우 | HQ |
| 2 | 서승현 | HQ |
| 3 | 이영준 | EDU |
| 4 | 송예설 | PR |

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## SELECT 구문 이해하기

```
mysql> SELECT name, team FROM ghackers ;
```

컬럼

ghackers

| no | name | team |
|---|---|---|
| 1 | 남정우 | HQ |
| 2 | 서승현 | HQ |
| 3 | 이영준 | EDU |
| 4 | 송예설 | PR |

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## SELECT 구문 이해하기

mysql〉 SELECT * FROM ghackers ;


| no | name | team |
|---|---|---|
| 1 | 남정우 | HQ |
| 2 | 서승현 | HQ |
| 3 | 이영준 | EDU |
| 4 | 송예설 | PR |

# 애스터리스크(*)는 '모든 열'을 의미하는 메타문자.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## SELECT 구문 이해하기

```
mysql> SELECT * FROM sample21 ;
```

명령의 종류

모든 열

테이블명

명령문의 마지막

```
mysql> SELECT * FROM sample21 ;
+------+--------+------------+--------------------+
| no   | name   | birthday   | address            |
+------+--------+------------+--------------------+
|    1 | 박준용 | 1976-10-18 | 대구광역시 수성구  |
|    2 | 김재진 | NULL       | 대구광역시 동구    |
|    3 | 홍길동 | NULL       | 서울특별시 마포구  |
+------+--------+------------+--------------------+
3 rows in set (0.05 sec)
```

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## 예약어와 데이터베이스 객체명

mysql> SELECT * FROM sample21 ;

예약어 테이블명

데이터베이스 객체명

데이터베이스 객체

# 데이터베이스 객체명에는 예약어와 동일한 이름을 사용할 수 없다.
# 예약어와 데이터베이스 객체명은 대소문자를 구별하지 않는다. (MySQL)

(* 다만 세션에서는 알아보기 쉽게 예약어는 대문자로 데이터베이스 객체명은 소문자로 표기한다.)

# 한 번 대소문자로 바꿔가면서 실행을 해보자.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## 데이터 타입 (예시: SELECT * FROM sample21 ; )

데이터는 자료형으로 분류가 가능하다. 열은 하나의 자료형만 가질 수 있다.

| No | Name | Birthday | Address |
|---|---|---|---|
| 1 | 박준용 | 1976-10-18 | 대구광역시 수성구 |
| 2 | 김재진 | NULL | 대구광역시 동구 |
| 3 | 홍길동 | NULL | 서울특별시 마포구 |

**수치형 데이터 (오른쪽으로 정렬되어 표시된다.)**

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## 데이터 타입

데이터는 자료형으로 분류가 가능하다. 열은 하나의 자료형만 가질 수 있다.

| No | Name | Birthday | Address |
|---|---|---|---|
| 1 | 박준용 | 1976-10-18 | 대구광역시 수성구 |
| 2 | 김재진 | NULL | 대구광역시 동구 |
| 3 | 홍길동 | NULL | 서울특별시 마포구 |

**문자열형 데이터 (왼쪽으로 정렬되어 표시된다.)**

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## 데이터 타입

데이터는 자료형으로 분류가 가능하다. 열은 하나의 자료형만 가질 수 있다.

| No | Name | Birthday | Address |
|---|---|---|---|
| 1 | 박준용 | 1976-10-18 | 대구광역시 수성구 |
| 2 | 김재진 | NULL | 대구광역시 동구 |
| 3 | 홍길동 | NULL | 서울특별시 마포구 |

**날짜형 데이터 (왼쪽으로 정렬되어 표시된다.)**

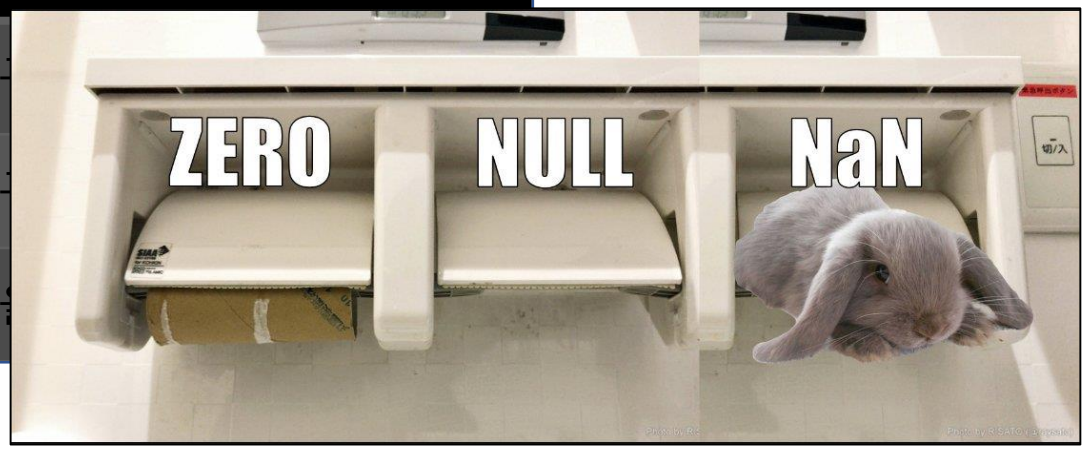
© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## 데이터 타입

데이터는 자료형으로 분류가 가능하다. 열은 하나의 자료형만 가질 수 있다.

| No | Name | Birthday | Address |
|---|---|---|---|
| 1 | 박준용 | 1976-10-18 | 대 |
| 2 | 김재진 | NULL | 대 |
| 3 | 홍길동 | NULL | 서 |




**NULL은 데이터가 들어있지 않은 것을 의미한다.**

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## DESC 명령

테이블에 어떤 열이 정의되어 있는지 알 수 있다.

mysql〉 DESC [테이블명] ;

```
mysql> DESC sample21;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| no       | int(11)     | YES  |     | NULL    |       |
| name     | varchar(20) | YES  |     | NULL    |       |
| birthday | date        | YES  |     | NULL    |       |
| address  | varchar(40) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

각 컬럼의 자료형

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## DESC 명령

테이블에 어떤 열이 정의되어 있는지 알 수 있다.

mysql〉 DESC [테이블명] ;

```
mysql> DESC sample21;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| no       | int(11)     | YES  |     | NULL    |       |
| name     | varchar(20) | YES  |     | NULL    |       |
| birthday | date        | YES  |     | NULL    |       |
| address  | varchar(40) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

NULL 값을 허용할 것인가?

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

## DESC 명령

테이블에 어떤 열이 정의되어 있는지 알 수 있다.

mysql〉 DESC [테이블명] ;

```
mysql> DESC sample21;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| no       | int(11)      | YES  |     | NULL    |       |
| name     | varchar(20)  | YES  |     | NULL    |       |
| birthday | date         | YES  |     | NULL    |       |
| address  | varchar(40)  | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

생략할 경우 적용되는 값

© 2018. SNU Growth Hackers all rights reserved

# Chapter 2.0 데이터베이스와 SQL

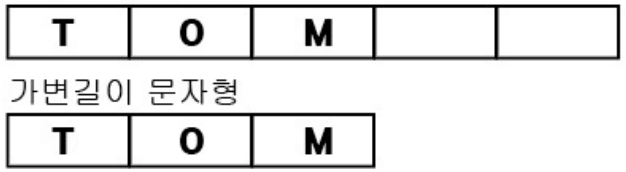
## 자료형

* 다양한 자료형을 참고하기 좋은 페이지
http://www.incodom.kr/DB_-_%EB%8D%B0%EC%9D%B4%ED%84%B0_%ED%83%80%EC%9E%85/MYSQL

INTEGER형 (수치형 자료)
정수값을 저장할 수 있는 자료형. 소수점은 불가.

CHAR형 (문자열형 자료) : 고정 길이 문자열
열의 최대 길이를 지정해야 한다. 언제나 고정된 길이로 데이터가 저장된다.
최대 길이보다 작은 문자열을 저장할 경우 공백문자로 나머지를 채운 후 저장한다.

VARCHAR형 (문자열형 자료) : 가변 길이 문자열
열의 최대 길이를 지정해야 한다. 데이터 크기에 맞춰 저장공간의 크기가 변한다.




DATE형 연, 월, 일의 데이터를 저장한다.

TIME형 시, 분, 초의 데이터를 저장한다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3

# Chapter 3.1 검색 조건 지정하기

## SELECT 구문 이해하기

```
mysql> SELECT * FROM sample21 ;
```

명령의 종류 — SELECT

모든 열 — *

테이블명 — sample21

명령문의 마지막 — ;

```
mysql> SELECT * FROM sample21 ;
+------+--------+------------+--------------------+
| no   | name   | birthday   | address            |
+------+--------+------------+--------------------+
|    1 | 박준용 | 1976-10-18 | 대구광역시 수성구  |
|    2 | 김재진 | NULL       | 대구광역시 동구    |
|    3 | 홍길동 | NULL       | 서울특별시 마포구  |
+------+--------+------------+--------------------+
3 rows in set (0.05 sec)
```

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.1 검색 조건 지정하기

## WHERE 구문 이해하기

Mysql〉 SELECT [컬럼명1], [컬럼명2] FROM [테이블명] WHERE [조건식]

컬럼을 선택할 때에는 SELECT 구문을 사용했다면,
로우를 선택할 때에는 WHERE 구문을 사용한다. (검색 조건을 추가하는 것)

# [조건식]은 열과 연산자, 상수로 구성되는 식이다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.1 검색 조건 지정하기

## WHERE 구문 이해하기

no 컬럼 값이 2인 행만 선택하였다.

Mysql> SELECT * FROM sample21 WHERE no = 2;

```
mysql> SELECT * FROM sample21 WHERE no = 2;
+------+---------+------------+--------------------+
| no   | name    | birthday   | address            |
+------+---------+------------+--------------------+
|    2 | 김재진  | NULL       | 대구광역시 동구    |
+------+---------+------------+--------------------+
1 row in set (0.00 sec)
```

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.1 검색 조건 지정하기

## 조건식에 쓰이는 연산자

# 비교 연산자

| | |
|---|---|
| = | 왼쪽과 오른쪽이 같아요! |
| 〈〉 | 왼쪽과 오른쪽이 달라요! |
| 〉 〈 〉= 〈= | 왼쪽이 오른쪽 보다 이상/이하/초과/미만 … 이에요! |
| IS NULL | NULL 값인가요? |

# 논리 연산자

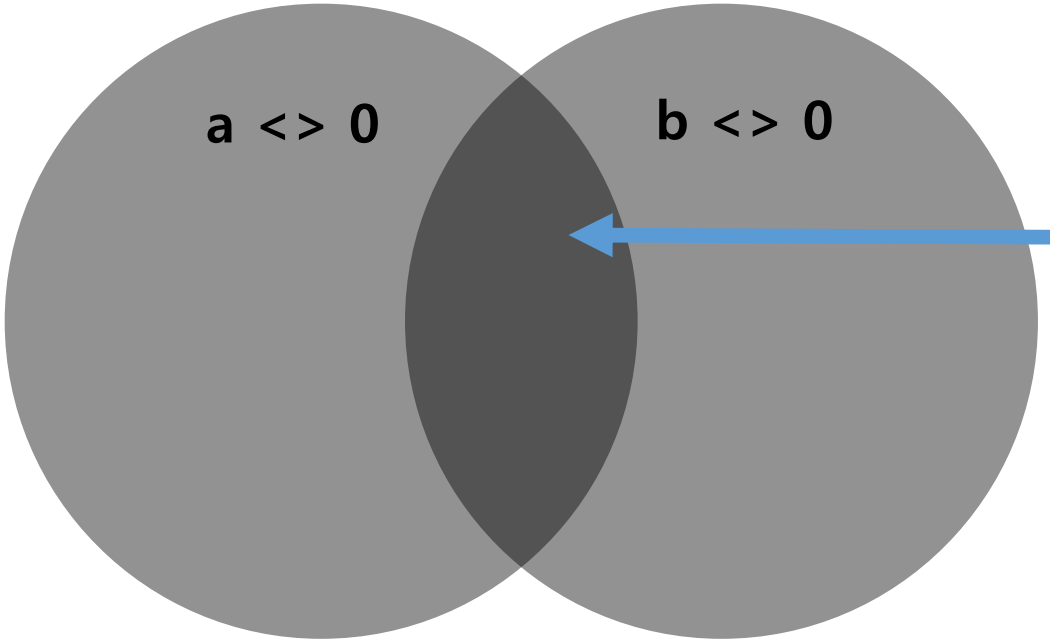
| | |
|---|---|
| AND | 정한 조건에 모두 맞아야해요! |
| OR | 정한 조건 중 맞는게 있으면 되어요! |
| NOT | 저 조건만 아니면 되어요! |

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.1 검색 조건 지정하기

## 조건식에 쓰이는 연산자

- sample24 테이블로 연습을 해보자.




```
mysql> SELECT * FROM sample24;
+------+------+------+------+
| no   | a    | b    | c    |
+------+------+------+------+
|    1 |    1 |    0 |    0 |
|    2 |    0 |    1 |    0 |
|    3 |    0 |    0 |    1 |
|    4 |    2 |    2 |    0 |
|    5 |    0 |    2 |    2 |
+------+------+------+------+
5 rows in set (0.06 sec)
```

3번 행만 제외하고 데이터를 뽑으려면 어떻게 해야할까?

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.1 검색 조건 지정하기

## 조건식에 쓰이는 연산자

정답은 몇 번일까?

1. SELECT * FROM sample24 WHERE a 〈〉 0 OR b 〈〉 0 ;
2. SELECT * FROM sample24 WHERE a 〈〉 0 AND b 〈〉 0 ;
3. SELECT * FROM sample24 WHERE a = 0 OR b = 0 ;
4. SELECT * FROM sample24 WHERE a = 0 AND b = 0 ;

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.1 검색 조건 지정하기

## 연산자의 우선 순위 및 주의할 점

- 연산자를 실행할 때, OR 보다 AND 쪽이 우선순위가 높다.

예시) 어떤 순서로 실행이 될까?

SELECT * FROM sample24 WHERE a = 1 OR a = 2 AND b = 1 OR b = 2

```
mysql> SELECT * FROM sample24 WHERE a=1 OR a=2 AND b=1 OR b=2;
+------+------+------+------+
| no   | a    | b    | c    |
+------+------+------+------+
|    1 |    1 |    0 |    0 |
|    4 |    2 |    2 |    0 |
|    5 |    0 |    2 |    2 |
+------+------+------+------+
3 rows in set (0.00 sec)
```

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.1 검색 조건 지정하기

## 연산자의 우선 순위 및 주의할 점

- 연산자를 실행할 때, OR 보다 AND 쪽이 우선순위가 높다.

예시) 어떤 순서로 실행이 될까? (해답)

SELECT * FROM sample24 WHERE a = 1 OR a = 2 AND b = 1 OR b = 2

```
mysql> SELECT * FROM sample24 WHERE a=1 OR a=2 AND b=1 OR b=2;
+------+------+------+------+
| no   | a    | b    | c    |
+------+------+------+------+
|    1 |    1 |    0 |    0 |
|    4 |    2 |    2 |    0 |
|    5 |    0 |    2 |    2 |
+------+------+------+------+
3 rows in set (0.00 sec)
```


© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.1 검색 조건 지정하기

## 연산자의 우선 순위 및 주의할 점

- 연산자를 실행할 때, OR 보다 AND 쪽이 우선순위가 높다.

예시) OR 부분을 먼저 실행시키고 싶다면? → 괄호를 사용한다!

SELECT * FROM sample24 WHERE (a = 1 OR a = 2) AND (b = 1 OR b = 2)

```
mysql> SELECT * FROM sample24 WHERE (a=1 OR a=2) AND (b=1 OR b=2);
+------+------+------+------+
| no   | a    | b    | c    |
+------+------+------+------+
|    4 |    2 |    2 |    0 |
+------+------+------+------+
1 row in set (0.00 sec)
```

© 2018. SNU Growth Hackers all rights reserved


하지만 NOT가
출동하면 어떨까?
N !
O !
T !

# Chapter 3.1 검색 조건 지정하기

## 연산자의 우선 순위 및 주의할 점

- NOT 구문은 오른쪽에만 항목을 지정하는 '단항 연산자' 입니다.
  오른쪽에 지정한 조건식의 반대 값을 반환합니다.

예시) 하지만 NOT가 출동하면 어떨까?

SELECT * FROM sample24 WHERE NOT ( (a = 1 OR a = 2) AND (b = 1 OR b = 2) ) ;

```
mysql> SELECT * FROM sample24 WHERE NOT ( (a=1 OR a=2) AND (b=1 OR b=2) ) ;
+------+------+------+------+
| no   | a    | b    | c    |
+------+------+------+------+
|    1 |    1 |    0 |    0 |
|    2 |    0 |    1 |    0 |
|    3 |    0 |    0 |    1 |
|    5 |    0 |    2 |    2 |
+------+------+------+------+
4 rows in set (0.00 sec)
```

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.2 패턴 매칭에 의한 검색

## 패턴 매칭 (부분 검색)

= 연산자로 검색하는 경우는 셀의 데이터 값이 완전히 동일한지를 비교합니다.
하지만 '특정문자나 문자열이 포함되어 있는지를 검색하고 싶은' 경우에는
패턴 매칭 방법을 사용한다.

단, 패턴은 문자열로 한정되어 지정할 수 있다. 수치형 상수는 지정할 수 없다.

패턴을 정의할 때는 메타문자를 사용한다. 메타문자(와일드카드) 예시는 아래와 같다.

패턴 매칭 시 '임의의 문자 또는 문자'에
매치하는 부분을 지정하기 위해 쓰이는 특수문자

- % 임의의 문자열
- _ 임의의 문자 하나

단, * 는 LIKE에서는 사용할 수 없다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.2 패턴 매칭에 의한 검색

## LIKE 구문 이해하기

= 연산자로 검색할 경우에는 열 값이 완전히 일치해야 참.
LIKE 구문을 사용하면 열 값이 부분적으로 일치하는 경우에도 참.

```
mysql> SELECT * FROM sample25 WHERE text LIKE 'SQL%' ;
```

text열에 있는 데이터(문자열) 중 SQL로 시작하는 데이터를 가져온다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.2 패턴 매칭에 의한 검색

## LIKE 구문 이해하기

mysql> SELECT * FROM sample25 ;

```
mysql> SELECT * FROM sample25 ;
+------+-----------------------------------------------+
| no   | text                                          |
+------+-----------------------------------------------+
|    1 | SQL은 RDBMS를 조작하기 위한 언어이다.         |
|    2 | LIKE에서는 메타문자 %와 _를 사용할 수 있다.   |
|    3 | LIKE는 SQL에서 사용할 수 있는 술어 중 하나이다. |
+------+-----------------------------------------------+
3 rows in set (0.00 sec)
```

mysql> SELECT * FROM sample25 WHERE text LIKE 'SQL%' ;

```
mysql> SELECT * FROM sample25 WHERE text LIKE 'SQL%' ;
+------+-----------------------------------+
| no   | text                              |
+------+-----------------------------------+
|    1 | SQL은 RDBMS를 조작하기 위한 언어이다. |
+------+-----------------------------------+
1 row in set (0.01 sec)
```

© 2018. SNU Growth Hackers all rights reserved

# Chapter 3.2 패턴 매칭에 의한 검색

## LIKE 구문 이해하기

[전방일치] SQL 은 RDBMS를 ……

SQL %

[중간일치] LIKE는 SQL에서 사용 ……

% SQL %

[후방일치] 입문 SQL

% SQL

```
mysql> SELECT * FROM sample25 WHERE text LIKE '%SQL%' ;
+------+-----------------------------------------------+
| no   | text                                          |
+------+-----------------------------------------------+
|    1 | SQL은 RDBMS를 조작하기 위한 언어이다.         |
|    3 | LIKE는 SQL에서 사용할 수 있는 술어 중 하나이다. |
+------+-----------------------------------------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM sample25 WHERE text LIKE '%₩%%' ;
+------+-----------------------------------------------+
| no   | text                                          |
+------+-----------------------------------------------+
|    2 | LIKE에서는 메타문자 %와 _를 사용할 수 있다.   |
+------+-----------------------------------------------+
1 row in set (0.00 sec)
```

# 문자 '%'를 검색하기 위해서는 \% 로 검색하면 된다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4

# Chapter 4.1 정렬

## ORDER BY 구문 이해하기

ORDER BY 구문을 사용하여 검색결과의 행 순서를 바꿀 수 있습니다.
만일 ORDER BY 구문을 따로 사용하지 않을 경우에는 데이터베이스 내부에 저장된 순서로 반환됩니다.

# 쿼리문을 작성할 때 ORDER BY 구문이 들어가는 위치는 다음과 같다.

mysql> SELECT 열명 FROM 테이블명 WHERE 조건식 ORDER BY 열명
mysql> SELECT 열명 FROM 테이블명 ORDER BY 열명

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.1 정렬

## ORDER BY 구문 이해하기

mysql〉 SELECT * FROM sample31 ;

```
mysql> SELECT * FROM sample31;
+------+------+--------------------+
| name | age  | address            |
+------+------+--------------------+
| A씨  |   36 | 대구광역시 중구    |
| B씨  |   18 | 부산광역시 연제구  |
| C씨  |   25 | 서울특별시 중구    |
+------+------+--------------------+
3 rows in set (0.02 sec)
```

mysql〉 SELECT * FROM sample31 ORDER BY age ;  # 오름차순이 default 값이다.

```
mysql> SELECT * FROM sample31 ORDER BY age ;
+------+------+--------------------+
| name | age  | address            |
+------+------+--------------------+
| B씨  |   18 | 부산광역시 연제구  |
| C씨  |   25 | 서울특별시 중구    |
| A씨  |   36 | 대구광역시 중구    |
+------+------+--------------------+
3 rows in set (0.01 sec)
```

age열의 데이터 정렬 순서에 따라서 행 순서가 바뀐다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.1 정렬

## ORDER BY 구문 이해하기

mysql> SELECT * FROM sample31 ORDER BY age ASC ;  # 오름차순

```
mysql> SELECT * FROM sample31 ORDER BY age ASC ;
+------+------+--------------------+
| name | age  | address            |
+------+------+--------------------+
| B씨  |   18 | 부산광역시 연제구  |
| C씨  |   25 | 서울특별시 중구    |
| A씨  |   36 | 대구광역시 중구    |
+------+------+--------------------+
3 rows in set (0.00 sec)
```

mysql> SELECT * FROM sample31 ORDER BY age DESC ;  # 내림차순

```
mysql> SELECT * FROM sample31 ORDER BY age DESC ;
+------+------+--------------------+
| name | age  | address            |
+------+------+--------------------+
| A씨  |   36 | 대구광역시 중구    |
| C씨  |   25 | 서울특별시 중구    |
| B씨  |   18 | 부산광역시 연제구  |
+------+------+--------------------+
3 rows in set (0.00 sec)
```

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.1 정렬

## ORDER BY 구문 이해하기

mysql> SELECT * FROM sample31 ORDER BY address ;

```
mysql> SELECT * FROM sample31 ORDER BY address ;
+------+------+--------------------+
| name | age  | address            |
+------+------+--------------------+
| A씨  |   36 | 대구광역시 중구    |
| B씨  |   18 | 부산광역시 연제구  |
| C씨  |   25 | 서울특별시 중구    |
+------+------+--------------------+
3 rows in set (0.01 sec)
```

# 문자열 데이터의 대소관계는
사전식 순서에 의해 결정된다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.1 정렬

## 복수의 열을 지정해 정렬하기

mysql> SELECT 열명 FROM 테이블명 ORDER BY 열명1 [ASC|DESC], 열명2 [ASC|DESC] ;

첫 번째 정렬 기준

두 번째 정렬 기준

예시)
SELECT * FROM sample32 ;

```
mysql> SELECT * FROM sample32 ;
+------+------+
| a    | b    |
+------+------+
|    1 |    1 |
|    2 |    1 |
|    2 |    2 |
|    1 |    3 |
|    1 |    2 |
+------+------+
5 rows in set (0.04 sec)
```

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.1 정렬

## 복수의 열을 지정해 정렬하기

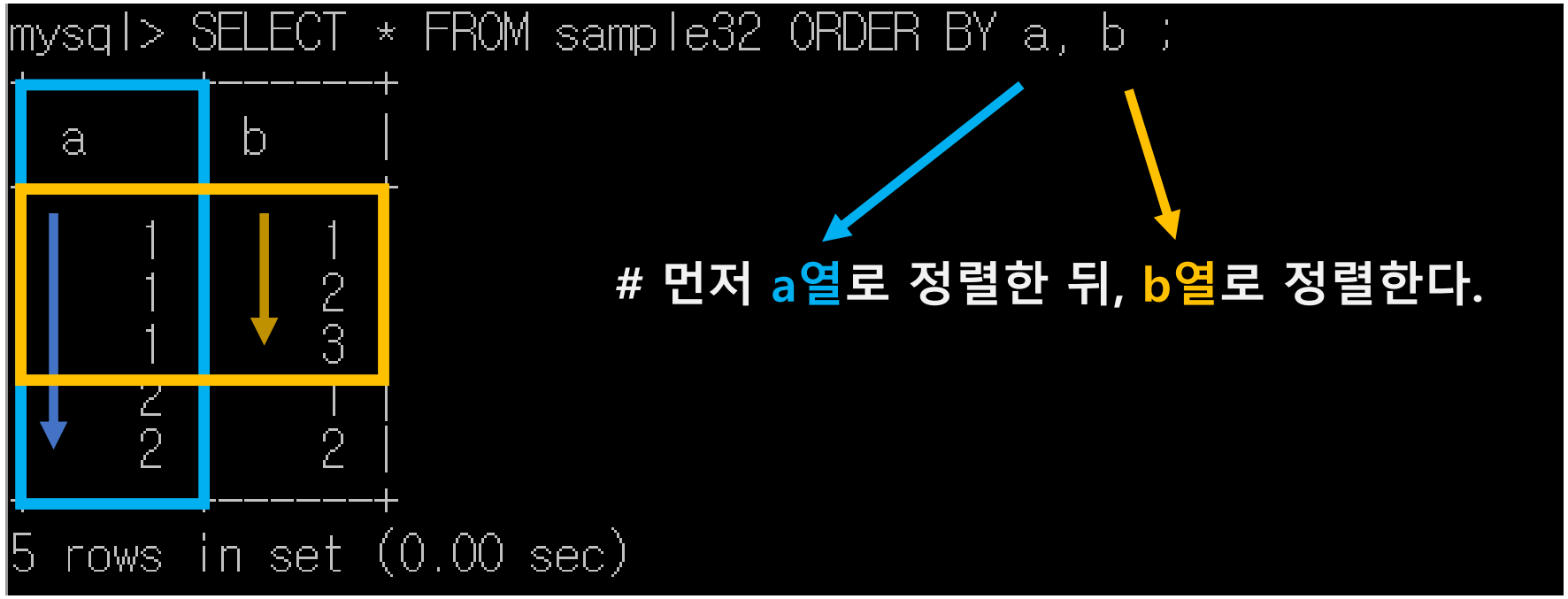
mysql> SELECT * FROM sample32 ORDER BY a ;


© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.1 정렬

## 복수의 열을 지정해 정렬하기

mysql> SELECT * FROM sample32 ORDER BY a, b ;

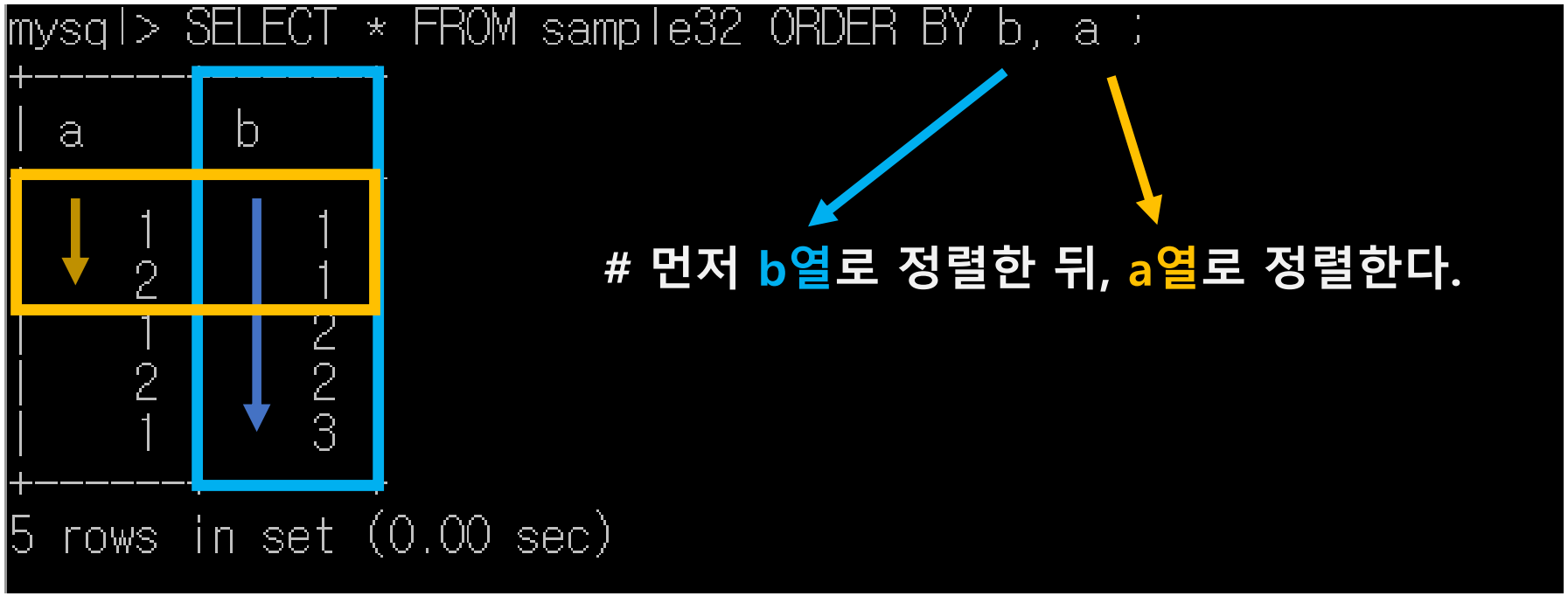



© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.1 정렬

## 복수의 열을 지정해 정렬하기

mysql> SELECT * FROM sample32 ORDER BY b, a ;




© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.1 정렬

## ORDER BY 구문 사용 시 주의 사항

# MySQL에서는 오름차순이 default 값이지만 다른 DBMS에서는 기본값이 다를 수도 있다.
# 문장의 가독성을 높이기 위해서라도 가능한 한 정렬방법을 생략하지 말고 지정하는 것이 좋다.

# ORDER BY로 지정한 열에서 NULL 값을 가지는 행은 가장 먼저 표시되거나 가장 나중에 표시된다.
# MySQL에서는 NULL 값을 가장 작은 값으로 취급하기 때문에 ASC(오름차순)에서는 가정 먼저,
DESC(내림차순)에서는 가장 나중에 표시합니다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.2 제한

## LIMIT 구문 이해하기

mysql> SELECT 열명 FROM 테이블명 LIMIT 행수 [OFFSET 시작행]

mysql> SELECT 열명 FROM 테이블명 WHERE 조건식 ORDER BY 열명 LIMIT 행수

# 2번째 쿼리에서 WHERE구문으로 검색한 후 ORDER BY로 정렬된 뒤 최종적으로
LIMIT구문이 처리된다.

# LIMIT는 정렬과 무관하게 상위 n건의 데이터를 가져오는 기능이다.
# LIMIT로 지정하는 것은 '최대 행수'이므로 LIMIT을 3으로 지정하여도
테이블에 데이터가 하나의 행만 있다면 1개의 행이 반환된다.
# LIMIT 구문은 표준 SQL은 아니며 MySQL과 PostgreSQL에서 사용되는 문법이다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.2 제한

## LIMIT 구문 이해하기

mysql> SELECT * FROM sample33 ORDER BY no DESC LIMIT 3 ;


© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.2 제한

## OFFSET 구문 이해하기

mysql> SELECT 열명 FROM 테이블명 LIMIT 행수 [OFFSET 시작행]

웹에서는 대량의 데이터를 하나의 페이지에 표시하는 것은 기능적으로나 속도적으로나 효율적이지 못하므로 일반적으로 페이지를 나누어 표시합니다. 이는 쿼리 구문을 통해서 데이터를 불러올 때에도 마찬가지 입니다.

LIMIT와 OFFSET 구문은 이러한 페이지 나눔 기능을 구현 가능하게 해줍니다.

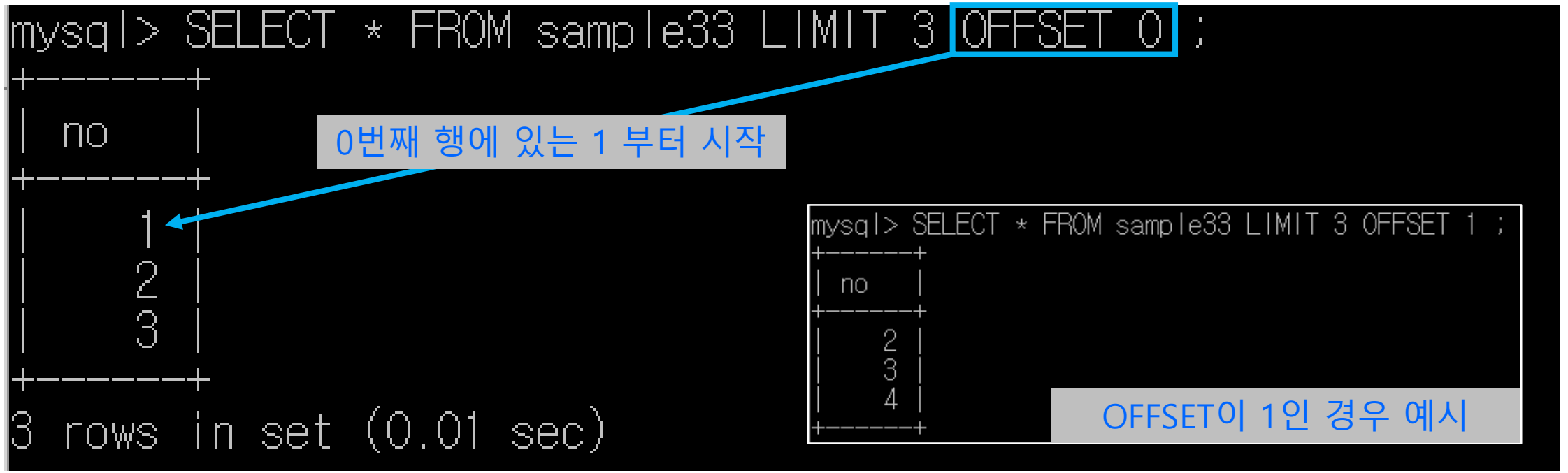
# LIMIT 구문의 OFFSET은 생략 가능하며 기본값은 0 입니다.
# OFFSET의 시작 위치 지정은 0부터 시작합니다. (파이썬의 인덱싱 순서와 같음)

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.2 제한

## OFFSET 구문 이해하기

mysql> SELECT * FROM sample33 LIMIT 3 OFFSET 0 ;




© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.3 수치연산

## 사칙연산

### 산술연산자

| 연산자 | 연산 | 예 |
|---|---|---|
| + | 덧셈 | 1 + 2 → 3 |
| - | 뺄셈 | 1 – 2 → -1 |
| * | 곱셈 | 1 * 2 → 2 |
| / | 나눗셈 | 1 / 2 → 0.5 |
| % | 나머지 | 1 % 2 → 1 |

**# 산술연산자 우선 순위**

**1 순위 : * / %** **2 순위 : + –**

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.3 수치연산

## 사칙연산 (SELECT 구문에서 연산하기)

mysql> SELECT *, price * quantity FROM sample34 ;

```
mysql> SELECT *, price*quantity FROM sample34 ;
+------+-------+----------+----------------+
| no   | price | quantity | price*quantity |
+------+-------+----------+----------------+
|    1 |   100 |       10 |           1000 |
|    2 |   230 |       24 |           5520 |
|    3 |  1980 |        1 |           1980 |
+------+-------+----------+----------------+
3 rows in set (0.02 sec)
```

price * quantity 계산결과

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.3 수치연산

## 사칙연산 (계산결과 열 이름에 별명 붙이기)

mysql> SELECT *, price * quantity AS amount FROM sample34 ;

```
mysql> SELECT *, price*quantity AS amount FROM sample34 ;
+------+-------+----------+--------+
| no   | price | quantity | amount |
+------+-------+----------+--------+
|    1 |   100 |       10 |   1000 |
|    2 |   230 |       24 |   5520 |
|    3 |  1980 |        1 |   1980 |
+------+-------+----------+--------+
3 rows in set (0.00 sec)
```

price * quantity 계산결과가 amount라는 이름으로 반환되었다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.3 수치연산

## 사칙연산 (계산결과 열 이름에 별명 붙이기)

mysql〉 SELECT *, price * quantity AS amount FROM sample34 ;

SELECT 구문에서 콤마(,)로 구분하여 복수의 식에 별명을 지정할 수 있습니다.
MySQL에서는 별명을 중복해서 지정해도 에러는 발생하지 않지만 프로그래밍 언어에서 결과값의 처리 방식에 따라서 문제가 발생할 수 있으므로 기본적으로 중복되지 않도록 지정합니다.

키워드 AS는 생략이 가능합니다.
→ SELECT price * quantity amount ;

에일리어스(alias)라고도 불리는 별명은 영어, 숫자, 한글 등으로 지정할 수 있습니다.
단, 별명을 한글로 지정하는 경우에는 여러 가지로 오작동하는 경우가 많으므로 쌍따옴표로 둘러싸서 지정해야 합니다.
→ SELECT price * quantity "금액" FROM sample34 ;

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.3 수치연산

## 사칙연산 (계산결과 열 이름에 별명 붙이기)

데이터베이스 객체명 (테이블 이름, 열 이름 등…)을 정할 때에는 더블쿼트(“ ”)를 사용한다.
→ “sample21”, “금액”
문자열 상수의 경우에는 싱글쿼트(‘ ’)를 사용한다.
→ ‘ABC’, ‘박준용’

앞서서 예약어와 동일한 이름은 지정할 수 없다고 하였지만, 더블쿼트를 사용한다면 가능합니다.
→ SELECT price * quantity AS SELECT FROM sample34 ; (X)
→ SELECT price * quantity AS “SELECT” FROM sample34 ; (O)

일반적으로 데이터베이스 객체명은 ‘숫자로 시작해서는 안된다’는 제약이 있지만 더블쿼트로 피할 수 있습니다.
MySQL 같은 경우에는 숫자로 시작하는 객체명이 허용되지만 **숫자로만!** 구성되는 객체명은 불가합니다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.3 수치연산

## 사칙연산 (WHERE 구문에서 연산하기)

SELECT *, price * quantity AS amount FROM sample34 WHERE price * quantity >= 2000 ;

```
mysql> SELECT *, price*quantity AS amount FROM sample34 WHERE price*quantity >= 2000;
+------+-------+----------+--------+
| no   | price | quantity | amount |
+------+-------+----------+--------+
|    2 |   230 |       24 |   5520 |
+------+-------+----------+--------+
1 row in set (0.00 sec)
```

SELECT *, price * quantity AS amount FROM sample34 WHERE amount >= 2000 ;

→ 오류가 나는 구문 (amount라는 열은 존재하지 않는다는 에러가 발생한다)

이는 쿼리 구문 처리 순서가 WHERE 구문 → SELECT 구문으로 처리되기 때문이다.

즉, 별명을 붙이는 것은 SELECT 구문에서 이루어지기 때문에 WHERE 구문에서는 아직 별명이 안 지어진 상태이다!

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.3 수치연산

## 사칙연산 (ORDER BY 구문에서 연산하기)

SELECT *, price * quantity AS amount FROM sample34 ORDER BY price * quantity >= 2000 ;

```
mysql> SELECT *, price*quantity AS amount FROM sample34 ORDER BY price*quantity >= 2000;
+------+-------+----------+--------+
| no   | price | quantity | amount |
+------+-------+----------+--------+
|    1 |   100 |       10 |   1000 |
|    3 |  1980 |        1 |   1980 |
|    2 |   230 |       24 |   5520 |
+------+-------+----------+--------+
3 rows in set (0.00 sec)
```

SELECT *, price * quantity AS amount FROM sample34 ORDER BY amount >= 2000 ;

→ ORDER BY 구문에서는 따로 설정한 별명으로도 실행이 된다.

이는 쿼리 구문 처리 순서가 WHERE 구문 → SELECT 구문 → ORDER BY 구문으로 처리되기 때문이다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.3 수치연산

## 사칙연산 (함수)

SELECT amount, ROUND(amount) FROM sample341 ;

```
mysql> SELECT amount, ROUND(amount) from sample341 ;
+---------+---------------+
| amount  | ROUND(amount) |
+---------+---------------+
| 5961.60 |          5962 |
| 2138.40 |          2138 |
| 1080.00 |          1080 |
+---------+---------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT amount, ROUND(amount, 1) from sample341 ;
+---------+------------------+
| amount  | ROUND(amount, 1) |
+---------+------------------+
| 5961.60 |           5961.6 |
| 2138.40 |           2138.4 |
| 1080.00 |           1080.0 |
+---------+------------------+
3 rows in set (0.00 sec)
```

반올림할 자리 수
default 값은 0 이다.

# 함수는 이외에도 TRUNCATE, SIN, COS, SQRT, LOG 등… 이 있습니다.
# MySQL에서 쓸 수 있는 함수는 다음 사이트를 참고하기 바랍니다.
→ http://egloos.zum.com/piccom/v/2866254

© 2018. SNU Growth Hackers all rights reserved

# Chapter 4.3 수치연산

## 사칙연산 (NULL의 처리)

SQL에서는 NULL 값이 0으로 처리되지 않습니다.
즉, NULL + 1의 결과값은 1 이 아닌 NULL 입니다. 이외 모든 사칙연산이 동일한 방식으로 적용됩니다.

통상적인 연산에서는 0으로 나눌 경우 division by zero 에러가 발생합니다.
하지만 1 / NULL을 계산하는 경우에는 NULL이 0으로 처리되지 않기 때문에 에러가 발생하지 않고 결과는 NULL이 됩니다.

© 2018. SNU Growth Hackers all rights reserved

# Chapter 5

# Chapter 5.1 PyMySQL 맛보기

## PyMySQL 설치하기

cmd창을 켠 뒤

conda install PyMySQL 또는 pip install PyMySQL 입력 !

# 세션 진행을 위해서 추가적으로 필요한 라이브러리

conda install sqlalchemy
conda install mysqlclient

© 2018. SNU Growth Hackers all rights reserved

# Chapter 5.1 PyMySQL 맛보기

## PyMySQL 실행하기

Jupyternotebook 파일 참고 (PyMySQL_Tutorial.ipynb)

© 2018. SNU Growth Hackers all rights reserved

# Chapter 5.2 과제

## 과제

# 1. PyMySQL을 통해서 오늘 배운 다른 쿼리문을 사용해보기
# 2. 다음 시간 퀴즈 준비하기. (단답형 주관식 1문제 : 30초)

퀴즈 예시 ) 다음 결과를 만들기 위해서 쿼리문의 빈칸을 채우시오.

```
mysql> SELECT * FROM sample21;
+------+--------+------------+--------------------+
| no   | name   | birthday   | address            |
+------+--------+------------+--------------------+
|    1 | 박준용 | 1976-10-18 | 대구광역시 수성구  |
|    2 | 김재진 | NULL       | 대구광역시 동구    |
|    3 | 홍길동 | NULL       | 서울특별시 마포구  |
+------+--------+------------+--------------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM sample21 [redacted] name;
+------+--------+------------+--------------------+
| no   | name   | birthday   | address            |
+------+--------+------------+--------------------+
|    2 | 김재진 | NULL       | 대구광역시 동구    |
|    1 | 박준용 | 1976-10-18 | 대구광역시 수성구  |
|    3 | 홍길동 | NULL       | 서울특별시 마포구  |
+------+--------+------------+--------------------+
3 rows in set (0.00 sec)
```

© 2018. SNU Growth Hackers all rights reserved


# Thank you

© 2018. SNU Growth Hackers all rights reserved