

데이터베이스와 SQL

2장. 데이터베이스 생성과 데이터 추가

목차

- mysql 명령줄 도구 사용 방법
- MySQL 자료형
- 테이블 생성
- 테이블 수정

mysql 명령줄 도구 사용 방법

■ MySQL 명령줄 사용 방법

- MySQL 8.0 Command Line Client 또는 DBeaver 실행
 - Enter password: 패스워드 입력

■ 사용 가능한 데이터베이스 확인

```
mysql > show databases;
```

```
mysql> show databases;
+-----+
| Database                |
+-----+
| information_schema      |
| mysql                   |
| performance_schema      |
| sakila                   |
| sqlclass_db             |
+-----+
8 rows in set (0.01 sec)

mysql>
```

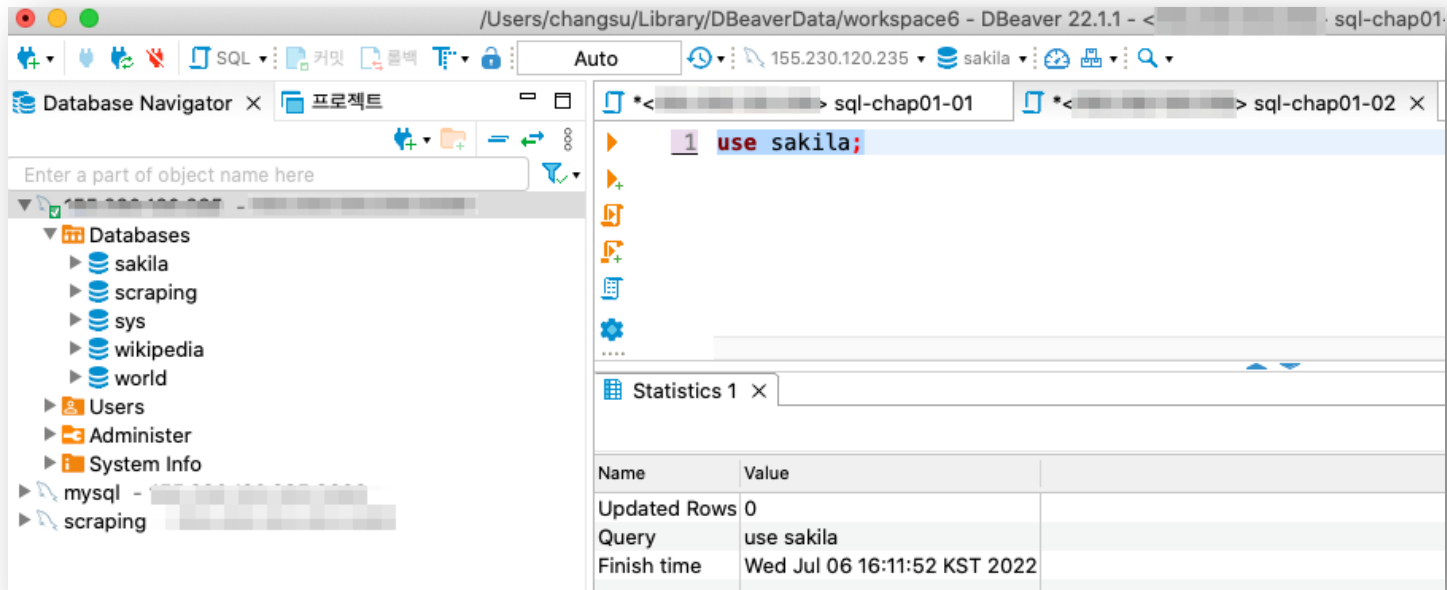
mysql 명령줄 도구 사용 방법

■ 샤키라 데이터베이스 선택

```
mysql> use sakila;  
Database changed
```

■ DBeaver에서 새 SQL 편집기 메뉴 선택 후 직접 입력

- 명령행 한 줄 실행: 명령어를 블록 설정 후 **Ctrl + Enter**키 입력



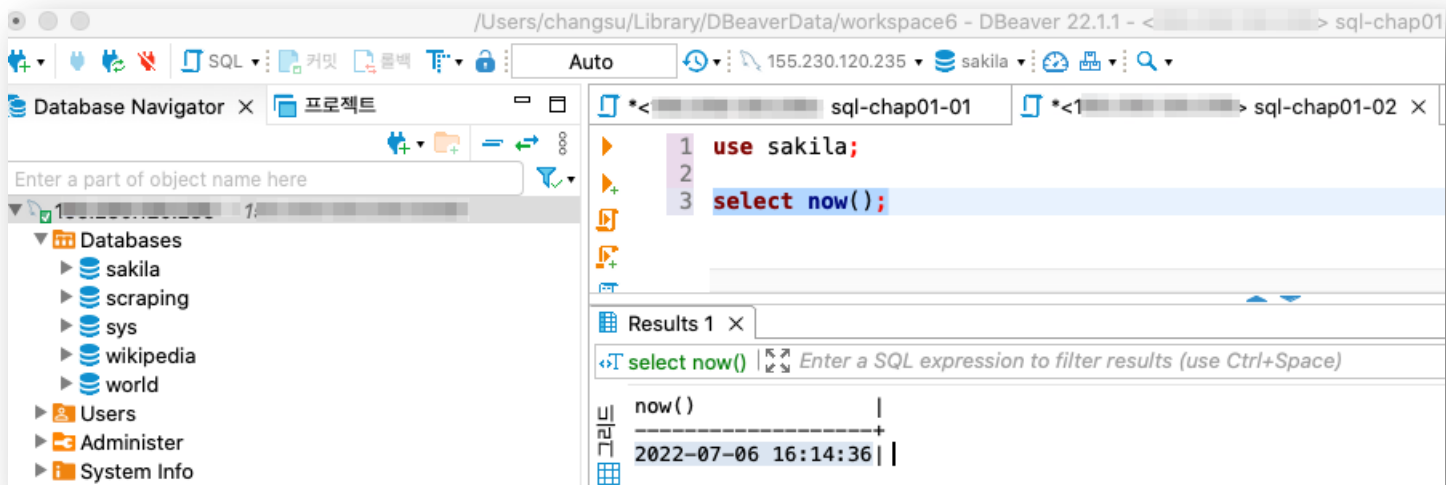
mysql 명령줄 도구 사용 방법

- 현재 날짜와 시간 정보 출력 쿼리

```
select now();
```

```
now() |
-----+
2022-07-06 16:14:36|
```

- DBeaver에서 실행한 화면



MySQL 자료형

■ 문자 데이터

■ 고정 길이 또는 가변 길이 문자열 저장

- char(20) : 고정 길이 문자열 (최대 255 바이트)
- varchar(20): 가변 길이 문자열 (최대 65,535 바이트)

■ 캐릭터셋

- 사용 언어에 따라 크기가 다름

```
show character set;
```

Charset	Description	Default collation	Maxlen
armSCII8	ARMSCII-8 Armenian	armSCII8_general_ci	1
ascii	US ASCII	ascii_general_ci	1
big5	Big5 Traditional Chinese	big5_chinese_ci	2
binary	Binary pseudo charset	binary	1
cp1250	Windows Central European	cp1250_general_ci	1
cp1251	Windows Cyrillic	cp1251_general_ci	1
. . .			
utf16	UTF-16 Unicode	utf16_general_ci	4
utf16le	UTF-16LE Unicode	utf16le_general_ci	4
utf32	UTF-32 Unicode	utf32_general_ci	4
utf8mb3	UTF-8 Unicode	utf8_general_ci	3
utf8mb4	UTF-8 Unicode	utf8mb4_0900_ai_ci	4

utf8mb4가 MySQL 8 버전의 기본 캐릭터셋

■ 텍스트 데이터

- varchar의 제한을 초과하는 데이터를 저장하는 자료형

자료형	최대 바이트 크기
tinytext	255 (1바이트)
text	65,535 (2바이트)
mediumtext	16,777,215 (24바이트)
longtext	4,294,967,295 (32바이트)

- 최대 크기를 초과하는 경우, 데이터가 잘려서 저장
- text 자료형과 varchar 자료형의 크기가 동일함

MySQL 자료형

■ 숫자 데이터

■ 정수 자료형

자료형	부호 있는 정수 저장값의 범위	부호가 없는 정수 저장값의 범위
tinyint	-128 ~ 127	0 ~ 255
smallint	-32768 ~ 32,767	0 ~ 65,535
mediumint	-8,388,608 ~ 8,388,607	0 ~ 16,777,215
int	-2,147,483,648 ~ 2,147,483,647	0 ~ 4,294,967,295
bigint	$-2^{63} \sim 2^{63} - 1$	$0 \sim 2^{64} - 1$

■ 부동 소수점

자료형	숫자 범위
float(p, s)	$-3.402823466 \times 10^{38} \sim 3.402823466 \times 10^{38}$ ex) float(4, 2): 총 4자리 중 소수점 아래 2자리
double(p, s)	$-1.7976931348623157 \times 10^{308} \sim 1.7976931348623157 \times 10^{308}$

■ 시간 데이터

자료형	기본 형식	허용값
date	YYYY-MM-DD	1000-01-01 ~ 9999-12-31
datetime	YYYY-MM-DD HH:MI:SS	1000-01-01 00:00:00.000000 ~ 9999-12-31 23:59:59.999999
timestamp	YYYY-MM-DD HH:MI:SS	1970-01-01 00:00:00.000000 ~ 2038-01-18 22:14:07.999999
year	YYYY	1901 ~ 2155
time	HHH:MI:SS	-838:59:59.000000 ~ 838:59:59.000000

■ 날짜 형식의 구성 요소

자료형	기본 형식	허용값
YYYY	연도	1000 ~ 9999
MM	월	01(1월) ~ 12(12월)
DD	일	01 ~ 31
HH	시간	00 ~ 23
MI	분	00 ~ 59
SS	초	00 ~ 59

datetime과 timestamp 비교

■ datetime과 timestamp 비교 테이블

	datetime	timestamp
데이터 저장 타입	문자형	숫자형
저장 공간	8 bytes	4 bytes
자동 입력 여부	X	O
타임존 영향	X	O (UTC로 변환)

- timestamp는 데이터를 입력하거나 수정하면 자동으로 시간 정보가 입력

테이블 생성

- DBeaver로 MySQL 접속
- 연습용 database 생성: **testdb**

```
create DATABASE testdb;
```

- 새롭게 생성한 testdb 선택

```
use testdb;
```

테이블 생성

■ person 테이블 구성

열	자료형	허용값	비고
person_id	smallint (unsinged)		
first_name	varchar(20)		
last_name	varchar(20)		
eye_color	char(2)	BR, BL, GR	
birth_date	date		
street	varchar(30)		
city	varchar(20)		
state	varchar(20)		
country	varchar(20)		
postal_code	varchar(20)		

■ favorite_food 테이블 구성

열	자료형	비고
person_id	smallint (unsinged)	foreign key
food	varchar(20)	

2.4 테이블 생성

■ person 테이블 생성

```
CREATE TABLE person
(person_id SMALLINT UNSIGNED,
 fname VARCHAR(20),
 lname VARCHAR(20),
 eye_color ENUM('BR', 'BL', 'GR'),
 birth_date DATE,
 street VARCHAR(30),
 city VARCHAR(20),
 state VARCHAR(20),
 country VARCHAR(20),
 postal_code VARCHAR(20),
 CONSTRAINT pk_person PRIMARY KEY (person_id)
);
```

기본 키 제약 조건

- CONSTRAINT [제약 조건 이름] PRIMARY KEY (필드이름)
 - 기본 키(primary key)로 person_id 열을 선정
 - NOT NULL과 UNIQUE 제약 조건의 특징을 가짐
 - 제약 조건(CONSTRAINT)
 - 데이터의 무결성을 지키기 위해, 데이터를 입력 받을 때 실행되는 검사 규칙

2.4 테이블 생성

■ person 테이블 확인

```
desc person;
```

Field	Type	Null	Key	Default	Extra
person_id	smallint unsigned	NO	PRI		
fname	varchar(20)	YES			
lname	varchar(20)	YES			
eye_color	enum('BR','BL','GR')	YES			
birth_date	date	YES			
street	varchar(30)	YES			
city	varchar(20)	YES			
state	varchar(20)	YES			
country	varchar(20)	YES			
postal_code	varchar(20)	YES			

■ Null 항목

- 특정 열의 데이터 생략 여부
- NO: person_id 필드는 **primary key**이므로 반드시 값이 입력되어야 함
- Null값의 의미: 해당 사항 없음, 알 수 없음, 비어 있음

2.4 테이블 생성

■ favorite_food 테이블 생성

```
create table favorite_food
  (person_id smallint unsigned,
   food VARCHAR(20),
   constraint pk_favorite_food primary key (person_id, food),
   constraint fk_fav_food_person_id foreign key (person_id)
   references person(person_id)
);
```

- **primary key(person_id, food)**
 - 2개의 primary key 설정
- **constraint fk_fav_food_person_id foreign key (person_id)**
 - 외래 키(foreign key) 제약 조건
 - favorite_food 테이블에서 person_id의 값에 person 테이블에 있는 값만 포함되도록 제한
- **references** 테이블이름 (필드이름)
 - 현재 테이블에서 참조되는 다른 테이블 이름 및 필드 이름 명시

2.4 테이블 생성

■ favorite_food 테이블 확인

```
desc favorite_food;
```

Field	Type	Null	Key	Default	Extra
person_id	smallint unsigned	NO	PRI		
food	varchar(20)	NO	PRI		

■ DBeaver에서 엔티티 관계도 확인

The screenshot shows the DBeaver interface with the '엔티티 관계도' (Entity Relationship Diagram) tab selected. The diagram illustrates the relationship between the 'person' and 'favorite_food' tables. The 'person' table has a primary key on 'person_id' and attributes: 'fname', 'lname', 'eye_color', 'birth_date', 'street', 'city', 'state', 'country', and 'postal_code'. The 'favorite_food' table has a primary key on 'person_id' and an attribute 'food'. A line with a crow's foot notation connects the 'person_id' primary key in 'favorite_food' to the 'person_id' primary key in 'person', indicating a one-to-one relationship.

Database Navigator: 155.230.120.235 - 155.230.120.235:3306
Databases: sakila, scraping, sys, testdb
Tables: favorite_food (16K), person (16K)
Views, Indexes, Procedures, Triggers, Events, wikipedia, world

Properties: 엔티티 관계도
155.230.120.2

person
123 person_id
ABC fname
ABC lname
ABC eye_color
ABC birth_date
ABC street
ABC city
ABC state
ABC country
ABC postal_code

favorite_food
123 person_id
ABC food

두 테이블의 참조 현황

2.5 테이블 수정 (ALTER)

■ 테이블 수정: ALTER

■ 숫자 키 데이터 생성

- MySQL: 자동 증가(auto-increment) 기능 제공
- foreign key로 설정된 부분은 다른 테이블에서 변경시 에러 발생.
 - 제약 조건 비활성화 → 테이블 수정 → 제약 조건 활성화
- SQL 명령어로 수정

```
set foreign_key_checks=0; # 제약 조건 비활성화
```

```
alter table person modify person_id smallint unsigned auto_increment;
```

```
set foreign_key_checks=1; # 제약 조건 활성화
```

- DBeaver의 Properties에서 변경

	컬럼명	#	Data Type	Not Null	Auto Increment	Key
Columns	person_id	1	smallint unsigned	[v]	[v]	PRI
Constraints	fname	2	varchar(20)	[]	[]	
Foreign Keys	lname	3	varchar(20)	[]	[]	
References	eye_color	4	enum('BR','BL','G')	[]	[]	
Triggers	birth_date	5	date	[]	[]	
Indexes	street	6	varchar(30)	[]	[]	
Partitions	city	7	varchar(20)	[]	[]	
Statistics	state	8	varchar(20)	[]	[]	
	country	9	varchar(20)	[]	[]	
	postal_code	10	varchar(20)	[]	[]	

person_id에 Null값을 주면,
자동으로 1부터 증가됨

2.5 테이블 수정 (ALTER)

- 숫자 키 데이터 생성
 - 표에서 가장 큰 값을 확인하고 값을 추가
 - 데이터베이스 서버가 값을 제공
- person 테이블의 person_id 재정의
 - person_id는 favorite_food에 외래키로 선정, 제약 조건 먼저 해제

```
ALTER TABLE 테이블명 MODIFY 컬럼명 데이터타입 추가할내용;
```

```
set foreign_key_checks=0;  
alter table person modify person_id smallint unsigned auto_increment;  
set foreign_key_checks=1;
```

Field	Type	Null	Key	Default	Extra
person_id	smallint unsigned	NO	PRI		auto_increment
fname	varchar(20)	YES			
lname	varchar(20)	YES			
eye_color	enum('BR', 'BL', 'GR')	YES			
birth_date	date	YES			
street	varchar(30)	YES			
city	varchar(20)	YES			
state	varchar(20)	YES			
country	varchar(20)	YES			
postal_code	varchar(20)	YES			

2.5 데이터 추가(INSERT)

■ 데이터 추가: **INSERT** 문

```
INSERT INTO 테이블이름 (열 이름1, 열 이름2, ...) VALUES (값1, 값2, ...);
```

```
insert into person  
(person_id, fname, lname, eye_color, birth_date)  
values (null, 'William', 'Turner', 'BR', '1972-05-27');
```

■ 데이터 확인: **SELECT** 문

```
SELECT * FROM 테이블이름;
```

■ 해당 테이블에서 모든 데이터(행, 컬럼) 데이터 출력

```
select * from person;
```

person_id	fname	lname	eye_color	birth_date	street	city	state	country	postal_code
1	William	Turner	BR	1972-05-27					

2.5 데이터 추가(INSERT)

■ 데이터 확인: **SELECT** 문

- 테이블의 특정 열의 데이터만 출력

```
SELECT 열 이름1, 열 이름2, ... FROM 테이블이름;
```

```
SELECT 열 이름1, 열 이름2, ... FROM 테이블이름 WHERE 열이름=값;
```

- person_id, fname, lname, birth_date로 출력

```
select person_id, fname, lname, birth_date from person;
```

```
person_id|fname  |lname  |birth_date|
-----+-----+-----+-----+
          1|William|Turner |1972-05-27|
```

- lname의 값이 'Turner'인 데이터 중에서 person_id, fname, lname, birth_date 열만 출력

```
select person_id, fname, lname, birth_date
from person where lname = 'Turner';
```

```
person_id|fname  |lname  |birth_date|
-----+-----+-----+-----+
          1|William|Turner |1972-05-27|
```

2.5 데이터 추가 (INSERT)

- favorite_food 테이블에 데이터 추가

- 한 행씩 추가

```
insert into favorite_food (person_id, food)
values (1, 'pizza');
```

```
insert into favorite_food (person_id, food)
values(1, 'cookies');
```

```
insert into favorite_food (person_id, food)
values (1, 'nachos');
```

- 한 번에 여러 행 추가

- values(값1), (값2), ... ;

```
insert into favorite_food (person_id, food)
values (1, 'pizza'), (1, 'cookie'), (1, 'nachos');
```

2.5 데이터 추가 (INSERT)

- favorite_food 테이블 데이터 확인
 - **order by** 컬럼이름: 컬럼의 값을 알파벳 순서로 정렬

```
select food from favorite_food  
where person_id = 1 order by food;
```

```
food    |  
-----+  
cookies|  
nachos  |  
pizza   |
```

2.5 데이터 추가(INSERT)

- person 테이블에 다른 데이터 추가

```
insert into person
(person_id, fname, lname, eye_color, birth_date,
street, city, state, country, postal_code)
values (null, 'Susan', 'Smith', 'BL', '1975-11-02',
'23 Maple St.', 'Arlington', 'VA', 'USA', '20220');
```

- person 테이블 데이터 확인

```
select person_id, fname, lname, birth_date from person;
```

```
person_id|fname  |lname |birth_date|
-----+-----+-----+-----+
          1|William|Turner|1972-05-27|
          2|Susan  |Smith |1975-11-02|
```

- person_id 필드에 자동으로 2가 저장됨

2.5 데이터 수정 (UPDATE)

■ 데이터 수정: UPDATE 문

```
UPDATE 테이블이름 SET 필드이름1 = 값1, 필드이름2=값2, ...  
WHERE 필드이름=데이터값;
```

■ William Turner의 정보 추가

- William Turner의 자료 입력 과정에서 주소 정보는 입력하지 않았음
- UPDATE 문을 이용하여 주소 정보를 추가

```
update person  
set street = '1225 Tremon St.',  
    city = 'Boston',  
    state = 'MA',  
    country = 'USA',  
    postal_code = '02138'  
where person_id=1;
```

```
select * from person;
```

person_id	fname	lname	eye_color	birth_date	street	city	state	country	postal_code
1	William	Turner	BR	1972-05-27	1225 Tremon St.	Boston	MA	USA	02138
2	Susan	Smith	BL	1975-11-02	23 Maple St.	Arlington	VA	USA	20220

2.5 데이터 삭제(DELETE)

■ 데이터 삭제: DELETE 문

```
DELETE FROM 테이블이름 WHERE 필드이름=데이터값;
```

- WHERE 절을 생략하면 테이블의 모든 데이터 삭제
 - 테이블은 삭제 되지 않음

```
delete from person where person_id=2;
```

person_id	fname	lname	eye_color	birth_date	street	city	state	country	postal_code
1	William	Turner	BR	1972-05-27	1225 Tremon St.	Boston	MA	USA	02138

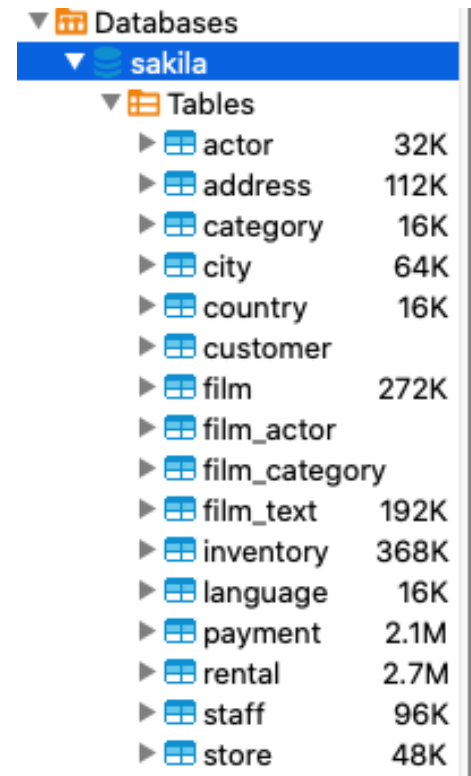
■ 테이블 삭제: DROP TABLE 문

```
DROP TABLE 테이블이름;
```

2.7 샤키라 데이터베이스

- 샤키라 데이터베이스
 - MySQL에서 샘플로 제공하는 데이터베이스
 - DVD 대여점 체인을 설계
- sakila 테이블 이름 및 정의

테이블명	정의
film	출시되어 대여할 수 있는 영화
actor	영화에 출연하는 배우
customer	영화를 보는 고객
category	영화 장르
payment	고객이 지불한 영화 대여료
language	영화배우들이 말하는 언어
film_actor	영화속 배우
inventory	대여 가능한 영화 여부



▼ Databases	
▼ sakila	
▼ Tables	
▶ actor	32K
▶ address	112K
▶ category	16K
▶ city	64K
▶ country	16K
▶ customer	
▶ film	272K
▶ film_actor	
▶ film_category	
▶ film_text	192K
▶ inventory	368K
▶ language	16K
▶ payment	2.1M
▶ rental	2.7M
▶ staff	96K
▶ store	48K

2.7 샤키라 데이터베이스

- sakila 데이터베이스에 포함된 테이블 확인

```
show tables;
```

Tables_in_sakila

```
-----+
actor
actor_info
address
category
city
country
customer
customer_list
film
film_actor
...
```

- customer 테이블 구성 확인

```
desc customer;
```

Field	Type	Null	Key	Default	Extra
customer_id	smallint unsigned	NO	PRI		auto_increment
store_id	tinyint unsigned	NO	MUL		
first_name	varchar(45)	NO			
last_name	varchar(45)	NO	MUL		
email	varchar(50)	YES			
address_id	smallint unsigned	NO	MUL		
active	tinyint(1)	NO		1	
create_date	datetime	NO			
last_update	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP



Questions?