




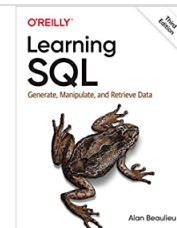
Day 23

🕒 작성일시	@2022년 8월 1일 오전 9:36
▼ 강의 번호	AUS 101
▼ 유형	강의
▼ 강사	Austin
📎 자료	<u>Learning_SQL_Second_Edition.pdf</u>
☑ 복습	<input type="checkbox"/>
📅 날짜	@2022년 8월 1일

Learning SQL: Generate, Manipulate, and Retrieve Data

Amazon.com: Learning SQL: Generate, Manipulate, and Retrieve Data: 9781492057611: Beaulieu, Alan: Books

 https://www.amazon.com/-/ko/dp/1492057614/ref=sr_1_1?crid=80HRM2JMHD9&keywords=Learning+SQL&qid=1659275134&srefix=,aps,279&sr=8-1&language=en_US¤cy=KRW



★★★★★ 330

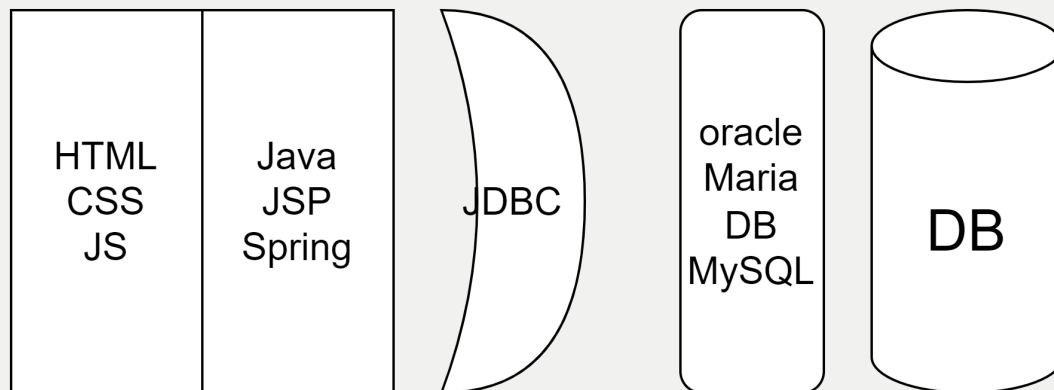
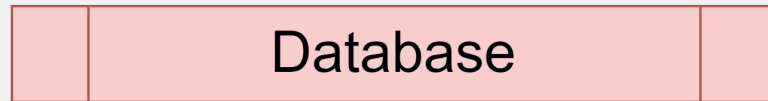


Database



DB

DBMS Database Management System : Oracle, My SQL, MS SQL, Access, Maria DB ...



관계형 데이터 베이스 Relational DBMS

SQL (Structured Query Language) 관계형 데이터베이스들을 위한 표준 데이터베이스 언어이다.

- DDL(Data Definition Language: 데이터 정의어) 데이터베이스와 테이블생성과 변경과 삭제
- DML(Data Management Language: 데이터 조작어) 데이터 삽입, 검색, 갱신, 삭제
- DCL(Data Control Language: 데이터제어어)데이터베이스 접근제어 및 사용 권한 관리

자료형 (My SQL Data Types)

1. 문자형 (Character Data)

char - 고정길이 : 8글자 데이터를 입력할 때 나머지 글자는 모두 공백으로 채워서 저장

varchar - 가변길이 : 8글자 데이터를 입력 할 때 8글자만 저장, 주로 사용하는 문자 데이터

char(20) /* fixed-length /

varchar(20) / variable-length */

2. 텍스트 데이터 (Text Data)

긴 문자열을 저장할 때 사용한다.

text - 주로 사용하는 텍스트 데이터

tinytext (My SQL only)

3. 숫자 데이터 (Numeric Data)

int 주로 사용하는 숫자 자료형

Table 2-3. MySQL integer types

Type	Signed rang	Unsigned range
Tinyint	128 to 127	0 to 255
Smallint	32,768 to 32,767	0 to 65,535
Mediumint	8,388,608 to 8,388,607	0 to 16,777,215
Int	2,147,483,648 to 2,147,483,647	0 to 4,294,967,295
Bigint	9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0 to 18,446,744,073,709,551,615

4. 날짜 데이터 (Temporal Data)

timestamp - 현재 날짜와 시간을 자동입력

datetime - 날짜와 시간

테이블 작성(Table Creation)

1. 설계 (Design)

테이블에 저장할 적절한 항목들과 그 항목들을 저장할 데이터 형과 크기를 설계
이름, 주소, 전화번호, 성별 등

2. 정제 (Refinement)

테이블 작성시 기본키 설정이 중요 (PRIMARY KEY)

이름의 경우 성과 이름으로 분리

주소의 경우도 하나의 필드로 저장하는 것 보다 시 군 구 별로 따로 분리


3. SQL 구문 생성 (Building SQL Schema Statements)

```
CREATE TABLE person
(
  person_id SMALLINT UNSIGNED,
  fname VARCHAR(20),
  lname VARCHAR(20),
  gender CHAR(1),
  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)
);
```

▼ 교재 예제 설치

MySQL :: Other MySQL Documentation

This page provides additional documentation. There's even more available on these extra pages:

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

```
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4f9de4c0-aa29-4a52-b863-97da05d053dd/sakila-db.zip
```

```
mysql> SOURCE C:/temp/sakila-schema.sql;  
mysql> SOURCE C:/temp/sakila-data.sql;
```

테이블 수정

1. 데이터 삽입

- 데이터에 추가할 테이블의 이름
- 데이터를 추가할 테이블의 열 이름
- 열에 넣을 값

```
INSERT INTO person  
(person_id, fname, lname, eye_color, birth_date)  
VALUES (0, 'William', 'Turner', 'BR', '1972-05-27');
```

person 테이블에서 person_id, fname, lname, birth_date 필드값들 전체를 조회

```
SELECT person_id, fname, lname, eye_color, birth_date FROM person;
```

```
SELECT * FROM person; - - 전체 필드를 다 적지 않고 * 로 전체 필드를 표시
```

person 테이블에서 person_id = 1인 조건에 해당하는 person_id, fname, lname, birth_date 필드값들을 조회

```

/* 간주석 */
SELECT person_id, fname, lname, birth_date
FROM person
WHERE person_id = 1; --주석

```

```

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');

```

food열을 favorite_food이라는 테이블로부터 조건 person_id = 1인 값만 순서를 food 열을 오름차순 정렬하여 조회

```

SELECT food
FROM favorite_food
WHERE person_id = 1
ORDER BY food;

```

Susan 데이터 삽입

```

INSERT INTO person
(person_id, fname, lname, birth_date,
street, city, state, country, postal_code)
VALUES (2, 'Susan', 'Smith', '1975-11-02',
'23 Maple St.', 'Arlington', 'VA', 'USA', '20220');

```

2. 데이터 업데이트 (Updating Data)

```

-- person_id = 1 에 추가
UPDATE person
SET street = '1225 Tremont St.',
city = 'Boston',
state = 'MA',
country = 'USA',
postal_code = '02138'
WHERE person_id = 1;

```

3. 데이터 삭제

```
DELETE FROM person
WHERE person_id = 2;
```

자주 등장하는 에러

```
mysql> INSERT INTO person
-> (person_id, fname, lname, gender, birth_date)
-> VALUES (1, 'Charles', 'Fulton', 'M', '1968-01-15');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

고유키값이 중복된 데이터를 입력하려 시도할때 에러 발생

```
mysql> INSERT INTO favorite_food (person_id, food)
-> VALUES (999, 'lasagna');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint
fails ('bank'. 'favorite_food', CONSTRAINT 'fk_fav_food_person_id' FOREIGN KEY
('person_id') REFERENCES 'person' ('person_id'))
```

존재하지 않는 외래키 foreign key를 참조할 때 에러 발생

```
mysql> UPDATE person
-> SET gender = 'Z'
-> WHERE person_id = 1;
ERROR 1265 (01000): Data truncated for column 'gender' at row 1
```

열 값 위반, 선언한 데이터형을 벗어났을때 에러 발생

```
mysql> UPDATE person
-> SET birth_date = 'DEC-21-1980'
-> WHERE person_id = 1;
ERROR 1292 (22007): Incorrect date value: 'DEC-21-1980' for column 'birth_date'
at row 1
```

잘못된 날짜 변환 , 날짜의 기본형인 년-월-일 로 입력 되지 않고, 월-일-년으로 입력되어
에러 발생

테이블제거

```
mysql> DROP TABLE favorite_food;
Query OK, 0 rows affected (0.56 sec)
mysql> DROP TABLE person;
Query OK, 0 rows affected (0.05 sec)
```

DROP TABLE 테이블이름;

쿼리

```
mysql> SELECT emp_id, fname, lname
-> FROM employee
-> WHERE lname = 'Bkadf1';
```



```
mysql> SELECT fname, lname
-> FROM employee;
```

Clause name	Purpose
SELECT	쿼리 결과에 포함 시킬 열들 결정
FROM	결과를 검색할 테이블, 테이블들을 조인하는 방법 등
WHERE	원하지 않는 데이터를 걸러내는 조건 설정
GROUP BY	공통열 값을 기준으로 행들을 그룹화
HAVING	원하지 않는 그룹을 걸러내는 조건 설정
ORDER BY	최종 결과의 행들을 정렬

1. SELECT

```
mysql> SELECT *
-> FROM department;

+-----+-----+
| dept_id | name          |
+-----+-----+
|      1 | Operations    |
|      2 | Loans         |
|      3 | Administration |
+-----+-----+
3 rows in set (0.04 sec)
```

```
SELECT * FROM department;
/* 이 쿼리에서 FROM은 department라는 하나의 테이블의 모든 열을 결과에 포함하는 것을 나타낸다.
*asterisk 문자는 모든열을 지정한다.*/
```

```
mysql> SELECT dept_id, name
-> FROM department;

+-----+-----+
| dept_id | name          |
+-----+-----+
|      1 | Operations    |
|      2 | Loans         |
|      3 | Administration |
+-----+-----+
3 rows in set (0.01 sec)
```

또는 하나의 열만 선택하여 결과를 볼 수도 있다.

```
mysql> SELECT name
-> FROM department;
+-----+
| name          |
+-----+
| Operations    |
| Loans         |
| Administration|
+-----+
3 rows in set (0.00 sec)
```

숫자나 문자를 그냥 출력

기존열의 값을 계산한 결과를 출력

함수를 사용한 결과

```
mysql> SELECT emp_id,
-> 'ACTIVE',
-> emp_id * 3.14159,
-> UPPER(lname)
-> FROM employee;
```

컬럼 별칭

```
mysql> SELECT emp_id,
-> 'ACTIVE' AS status,
-> emp_id * 3.14159 AS empid_x_pi,
-> UPPER(lname) AS last_name_upper
-> FROM employee;
```

```
SELECT emp_id,
'ACTIVE' AS STATUS,
emp_id *3.14159 AS 상태,
UPPER(lname) AS 대문자성
FROM employee;
```

“AS”는 생략 가능

실제 이름을 바꾸는 것이 아니 보이는것 만 바뀌어서 표시

중복 제거 (DISTINCT)

상황에 따라 쿼리가 중복된 행을 반환 할 수 있다. 고유한 하나의 값만 남기고 나머지는 제거한 값을 확인 할 수 있다.

```
SELECT cust_id  
FROM account;
```

중복값 제거 시

```
SELECT DISTINCT cust_id  
FROM account;
```

FROM 절

지금까지는 FROM절에 단 하나의 테이블만 지정하였는데, 대부분의 실제 SQL구문에서는 하나 이상의 테이블을 목록으로 정의하여 사용된다.

FROM절은 쿼리에 사용되는 테이블을 명시할 뿐만아니라 테이블들을 서로 연결하는 수 단도 정의하게 된다.

영구 테이블(Permanent Table) create table 로 생성된 테이블

임시 테이블(Temporary Table) 서브 쿼리로 반환된 행들, 메모리에 임시 저장된 휘 발성 테이블

가상 테이블(Virtual Table) create view로 생성된 테이블

파생 테이블(=임시 테이블)

```
SELECT e.emp_id,e.fname,e.lname  
FROM (SELECT emp_id, fname, lname, start_date, title
```

```
FROM employee) e;
```

view(=가상 테이블)

```
CREATE VIEW employee_vw AS  
SELECT emp_id, fname, lname,  
YEAR(start_date) start_year  
FROM employee;
```

생성 후

```
SELECT emp_id, start_year  
FROM employee_vw;
```

진행

테이블 연결(Table links)

```
SELECT employee.emp_id, employee.fname,  
employee.lname, department.name dept_name  
FROM employee INNER JOIN department  
ON employee.dept_id = department.dept_id;
```

```
SELECT e.emp_id, e.fname,  
e.lname, d.name dept_name  
FROM employee AS e INNER JOIN department AS d  
ON e.dept_id = d.dept_id;
```

별명 붙여서 줄이기

WHERE 절

WHERE 절은 결과에 출력되기를 원하지 않는 행을 걸러내는 방법이다.

```
SELECT emp_id, fname, lname, start_date, title
FROM employee
WHERE title = 'Head Teller';
```

조건 2개를 동시만족하는 데이터 출력

```
SELECT emp_id, fname, lname, start_date, title
FROM employee
WHERE title = 'Head Teller'
AND start_date > '2002_01_01';
```

GROUP BY 절과 HAVING 절

```
SELECT d.name, COUNT(e.emp_id) num_employess
FROM department d INNER JOIN employee e
ON d.dept_id = e.dept_id
GROUP BY d.name
HAVING COUNT(e.emp_id) > 2;
```

GROUP BY 열을 기준으로 행들의 값으로 그룹으로 나누고 그 나뉜 그룹에 조건을 적용하는 것이 HAVING 이다.

ORDER BY 절

일반적으로 쿼리는 반환된 결과셋의 행은 특정한 순서로 정렬되지 않는다. 결과를 원하는 순서로 정렬 하려면 ORDER BY 절을 사용한다.

```
SELECT open_emp_id, product_cd
FROM account;
```

open_emp_id	product_cd
10	CHK
10	SAV
10	CD
10	CHK
10	SAV
13	CHK
13	MM
1	CHK
1	SAV
1	MM
16	CHK
1	CHK
1	CD
10	CD
16	CHK
16	SAV
1	CHK
1	MM
1	CD
16	CHK
16	BUS
10	BUS
16	CHK
13	SBL

```
SELECT open_emp_id, product_cd
FROM account
ORDER BY open_emp_id;
```

open_emp_id	product_cd
1	CD
1	MM
1	SAV
1	CHK
1	CHK
1	CHK
1	MM
1	CD
10	CHK
10	BUS
10	CD
10	CHK
10	SAV
10	CD
10	SAV
13	SBL
13	CHK
13	MM
16	SAV
16	CHK
16	CHK
16	BUS
16	CHK
16	CHK

```
SELECT open_emp_id, product_cd
FROM account
ORDER BY open_emp_id, product_cd;
```

open_emp_id	product_cd
1	CD
1	CD
1	CHK
1	CHK
1	CHK
1	MM
1	MM
1	SAV
10	BUS
10	CD
10	CD
10	CHK
10	CHK
10	SAV
10	SAV
13	CHK
13	MM
13	SBL
16	BUS
16	CHK
16	CHK
16	CHK
16	CHK
16	SAV

정렬의 기준이 여러개인 경우 첫번째 정렬을 마친 후 두번째 정렬 진행

정렬의 기본은 오름차순 이나, **DESC**를 적으면 내림차순 이다.

```
SELECT account_id,product_cd,open_date, avail_balance
FROM account
ORDER BY avail_balance DESC;
```


account_id	product_cd	open_date	avail_balance
29	SBL	2004-02-22	50,000.0
28	CHK	2003-07-30	38,552.05
24	CHK	2002-09-30	23,575.12
15	CD	2004-12-28	10,000.0
27	BUS	2004-03-22	9,345.55
22	MM	2004-10-28	9,345.55
12	MM	2004-09-30	5,487.09
17	CD	2004-01-12	5,000.0
18	CHK	2001-05-23	3,487.19
3	CD	2004-06-30	3,000.0
4	CHK	2001-03-12	2,258.02
13	CHK	2004-01-27	2,237.97
8	MM	2002-12-15	2,212.5
23	CD	2004-06-30	1,500.0
1	CHK	2000-01-15	1,057.75
7	CHK	2002-11-23	1,057.75
11	SAV	2000-01-15	767.77
10	CHK	2003-09-12	534.12
2	SAV	2000-01-15	500.0
19	SAV	2001-05-23	387.99
5	SAV	2001-03-12	200.0
21	CHK	2003-07-30	125.67
14	CHK	2002-08-24	122.37
25	BUS	2002-10-01	0.0

```
/*  
SELECT EMPNO, ENAME, SAL  
FROM EMP;  
  
SELECT empno, ename, sal  
FROM emp;  
  
SELECT empno, ename, sal FROM emp;  
  
SELECT empno, ename, sal  
FROM emp;  
*/  
  
SELECT empno, ename, sal  
FROM emp;
```

고유값 출력 UNIQUE

```
SELECT UNIQUE JOB  
FROM emp;
```