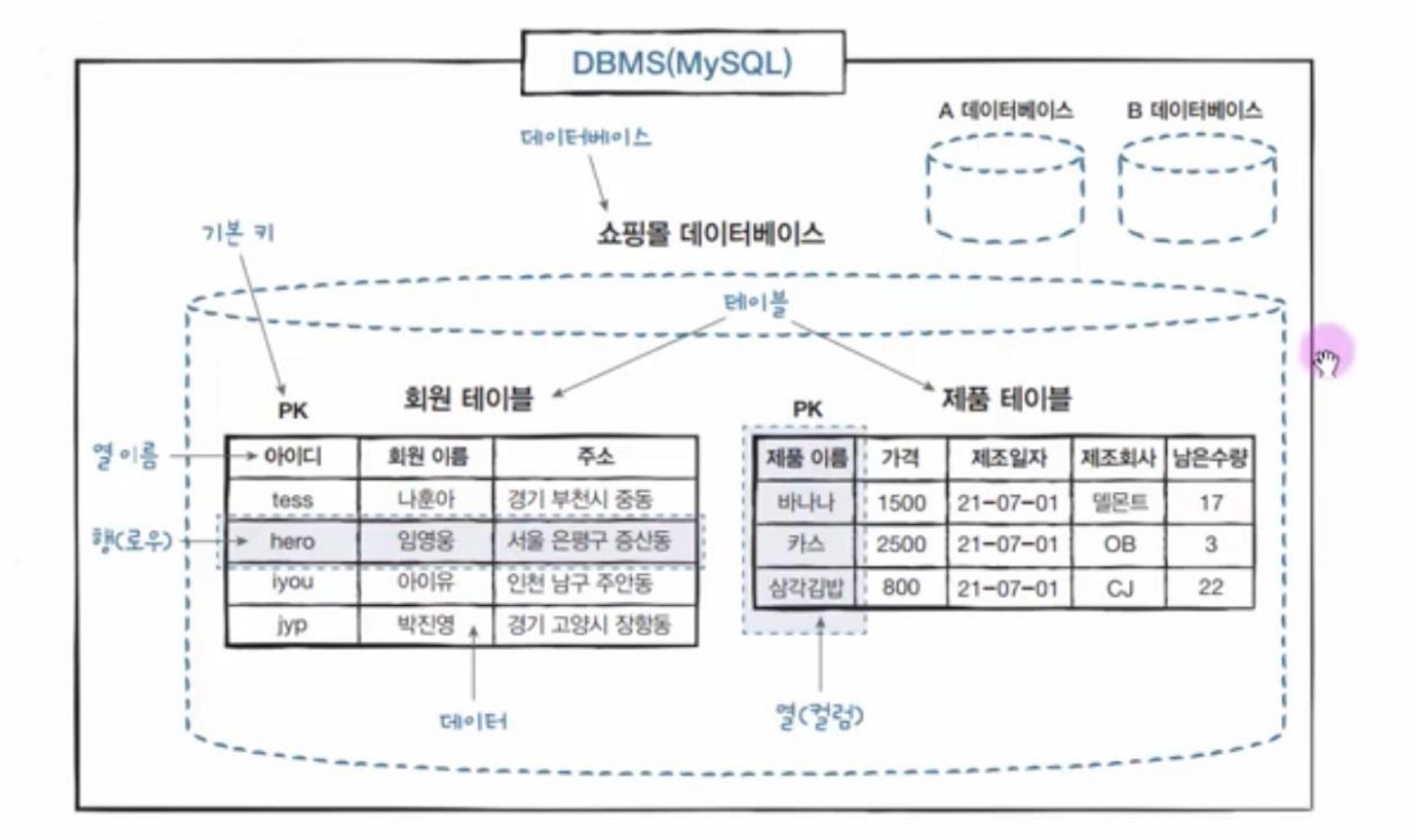
Seoul Software ACademy

웹 개발자 부트캠프 과정

SeSAC x CODINGOn

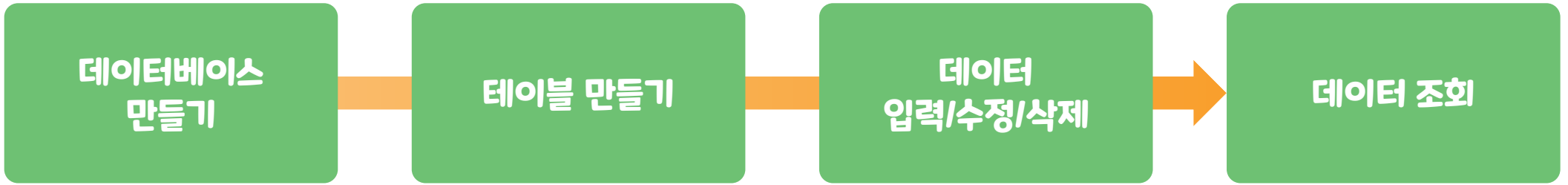
With. 팀 뤼쳐드

전체 데이터베이스 구성도



이미지 출처: https://velog.io/@estell/SQL-3-mySQL-%ED%85%8C%EC%9D%B4%EB%B8%94%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%B2%A0%EC%9D%B4%EC%8A%A4-%EB%AA%A8%EB%8D%B8%EB%A7%81

데이터베이스 구축 단계



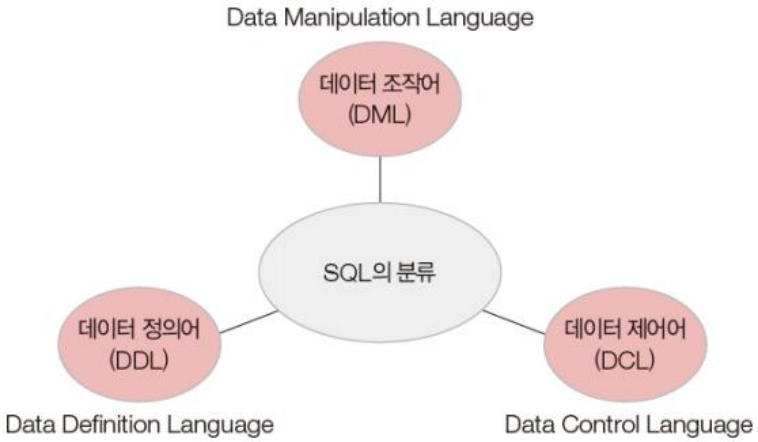
SQL

SQL

- Structured Query Language
- 구조적 쿼리 언어
- 관계형 데이터베이스에 정보를 저장하고 처리하기 위한 프로그래밍 언어
- 실행 순서가 없는 비절차적인 언어

SQL 분류

- 기능에 따른 분류



- 데이터 정의어 (DDL, Data Definition Language)
- 데이터 조작어 (DML, Data Manipulation Language)
- 데이터 제어어 (DCL, Data Control Language)

이미지 출처: https://hoit1302.tistory.com/129

SQL 분류

SQL 분류	명령어	설명
데이터 조작어 (DML)	SELECT	데이터를 조회하거나 검색하기 위한 명령어
	INSERT	데이블의 데이터를 변경하는 명령어. 데이터 삽입, 수정, 삭제
	UPDATE	
	DELETE	
데이터 정의어 (DDL)	CREATE	테이블이나 관계의 구조를 생성하는데 사용하는 명령어
	ALTER	
	DROP	
데이터 제어어 (DCL)	GRANT	데이터베이스에 접근하고 데이터 사용 권한을 주거나 회수하는데 사용하는 명령어
	REVOKE	

데이터 정의어
(DDL, Data Definition Language)

CREATE 문

- 데이터베이스와 테이블을 생성하는 명령어
- 테이블 이름, 열 이름, 데이터 형식 등을 지정
- 기본키, 외래키 정의

```
CREATE DATABASE db_name
DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_general_ci;
-- 이모지 지원에 한계가 있음.


CREATE DATABASE db_name
DEFAULT CHARACTER SET utf8mb4 DEFAULT COLLATE utf8mb4_unicode_ci;
```

CREATE 문

- 데이터베이스 생성 + 한글 인코딩

```
CREATE DATABASE db_name
DEFAULT CHARACTER SET utf8mb4 DEFAULT COLLATE utf8mb4_unicode_ci;
```

- 테이블 생성

```
CREATE TABLE 테이블명 (
  속성이름1 데이터타입 PRIMARY KEY,
  속성이름2 데이터타입,
  [FOREIGN KEY 속성이름 REFERENCES 테이블이름(속성이름)]
);
```

*한글 인코딩: 프로그래밍 언어는 영어 기반이므로 한글을 사용할 수 있도록!!

데이터 베이스 명명 규칙

- 소문자로 작성하는 게 좋다.
- 복합어구에는 _ (언더바)를 사용하는 게 좋다.
- 데이터 타입을 이름으로 정하는 것은 피하는 게 좋다.

데이터 타입

- MySQL에서 제공하는 데이터 타입은 매우 다양
- 자주 사용하는 형식 위주로 알아보자

숫자형

문자형

날짜형

데이터 타입 – 숫자형

타입	바이트 수	범위	설명
TINYINT	1	-128 ~ 127	정수
SMALLINT	2	-32768 ~ 32767	정수
INT	4	약 -21억 ~ 21억	정수
BIGINT	8	약 -900경 ~ 900경	정수
FLOAT	4	-3.40E+38 ~ -1.17E-38	소수점 아래 7자리까지 표현

데이터 타입 – 문자형

타입	바이트 수	설명
CHAR(N)	1 ~ 255	고정길이 문자형 / n을 1부터 255까지 지정
VARCHAR(N)	1 ~ 65535	가변길이 문자형 / n을 1부터 65535까지 지정
TEXT	1 ~ 65535	255 크기의 TEXT 데이터 값
MEDIUMTEXT	1 ~ 16777215	16777215 크기의 TEXT 데이터 값

데이터 타입 – 날짜형

타입	바이트 수	설명
DATE	3	날짜 저장 (YYYY-MM-DD 형식)
TIME	3	시간 저장 (HH:MM:SS 형식)
DATETIME	8	날짜와 시간 저장 (YYYY-MM-DD HH:MM:SS 형식)

ALTER 문

- 생성된 테이블의 속성과 속성에 대한 제약 및 기본키, 외래키를 변경

```
ALTER TABLE 테이블명 ADD 속성이름 데이터타입;      -- 속성 새로 추가하는 경우
ALTER TABLE 테이블명 DROP COLUMN 속성이름;         -- 기존 속성 삭제하는 경우
ALTER TABLE 테이블명 MODIFY 속성이름 데이터타입;   -- 기존 속성 수정하는 경우
```

DROP 문

- 생성된 테이블 삭제
- 주의) **테이블 구조와 데이터 모두 삭제**

```
DROP TABLE 테이블이름;
```

번호	이름	나이
1	이진영	20
2	김찬희	21
3	유지은	22

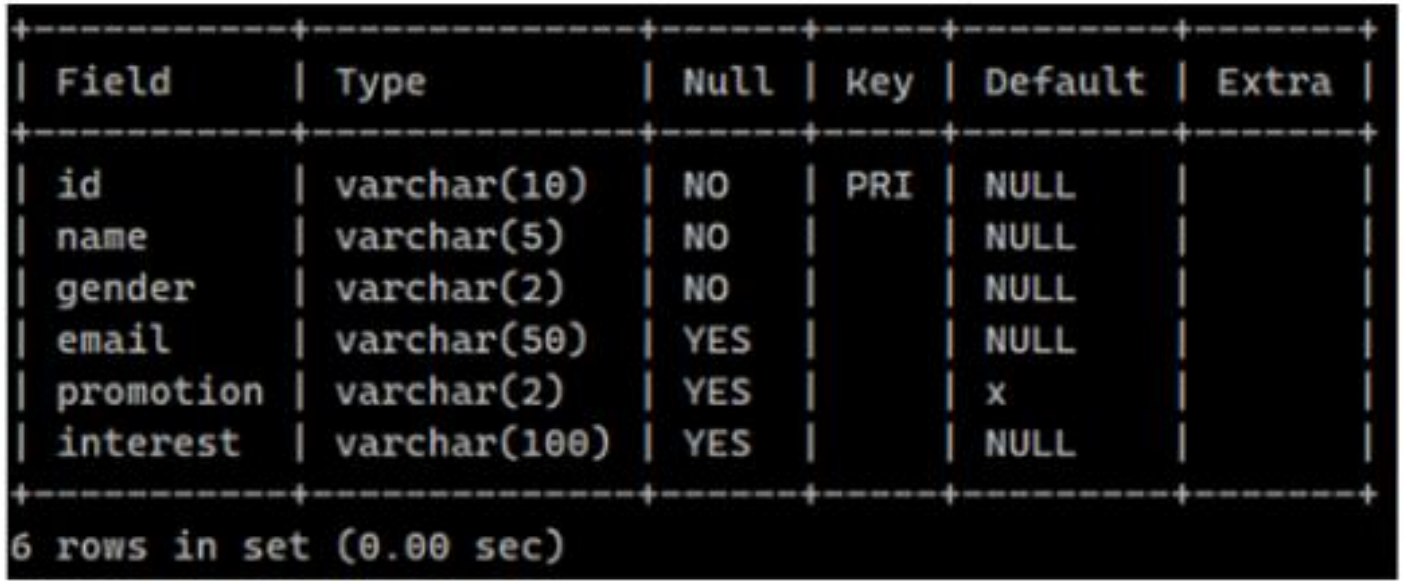
실습. DDL

- DDL을 이용해 아래 테이블 완성하기
- 테이블명: member

```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| id        | varchar(20) | NO   | PRI | NULL    |       |
| name      | varchar(5)  | NO   |     | NULL    |       |
| age       | int         | YES  |     | NULL    |       |
| gender    | varchar(2)  | NO   |     | NULL    |       |
| email     | varchar(50) | YES  |     | NULL    |       |
| promotion | varchar(2)  | YES  |     | x       |       |
+-----------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

실습. DDL

- 이전 실습에서 생성한 테이블을 ALTER 명령어를 이용해 구조 변경



```
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| id        | varchar(10)   | NO   | PRI | NULL    |       |
| name      | varchar(5)    | NO   |     | NULL    |       |
| gender    | varchar(2)    | NO   |     | NULL    |       |
| email     | varchar(50)   | YES  |     | NULL    |       |
| promotion | varchar(2)    | YES  |     | x       |       |
| interest  | varchar(100)  | YES  |     | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

*힌트
- id 컬럼: 값 형식 변경
- age 컬럼: 삭제
- interest 컬럼: 추가

데이터 조작어

(DML, Data Manipulation Language)

DML

- DML (Data **Manipulation** Language)
- 데이터베이스의 **내부 데이터를 관리**하기 위한 언어

종류	역할
SELECT	데이터베이스에서 데이터를 검색(조회)하는 역할을 한다.
INSERT	테이블에 데이터를 추가하는 역할을 한다.
UPDATE	테이블에서 데이터를 수정하는 역할을 한다.
DELETE	테이블에서 데이터를 삭제하는 역할을 한다.

CRUD

- 대부분의 컴퓨터 소프트웨어가 가지는 **기본적인 처리 기능**

- **Create (생성)**
- **Read (읽기)**
- **Update (갱신)**
- **Delete (삭제)**

이름	조작	SQL
Create	생성	INSERT
Read	읽기	SELECT
Update	갱신	UPDATE
Delete	삭제	DELETE

INSERT 문

- 테이블에 새로운 투플을 **추가**

```
INSERT INTO 테이블명 (필드1, 필드2, 필드3, ...) VALUES (값1, 값2, 값3, ...);
```

```
INSERT INTO 테이블명 VALUES (값1, 값2, 값3, ...);
```

필드를 명시하지 않는 경우
테이블의 **모든 컬럼에 값을 순서대로** 추가해야 함

SELECT 문

- **데이터를 검색**하는 기본 문장
- 질의어 (query) 라고도 함
- SQL 문 중 가장 많이 사용되는 문법

```
SELECT 속성이름, ... FROM 테이블이름 [WHERE 검색조건]
```

*대괄호([]) 안의 SQL 예약어는 선택적으로 사용 가능

그 전에 잠깐! CREATE

```sql
CREATE TABLE customer
(
  custid    CHAR(10) NOT NULL PRIMARY KEY,
  custname  VARCHAR(10) NOT NULL,
  addr      CHAR(10) NOT NULL,
  phone     CHAR(11),
  birth     DATE
);
```

```sql
CREATE TABLE orders
(
   orderid   INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
   custid    CHAR(10) NOT NULL,
   prodname  CHAR(6) NOT NULL,
   price     INT  NOT NULL,
   amount    SMALLINT  NOT NULL,
   FOREIGN KEY (custid) REFERENCES customer(custid)
);
```

그 전에 잠깐! INSERT

```
INSERT INTO customer VALUES('bunny', '강해린', '대한민국 서울', '01012341234', '2000-02-23');
INSERT INTO customer VALUES('hello', '이지민', '대한민국 포항', '01022221234', '1999-08-08');
INSERT INTO customer VALUES('kiwi', '최지수', '미국 뉴욕', '01050005000', '1990-12-25');
INSERT INTO customer VALUES('imminji01', '강민지', '영국 런던', '01060001000', '1995-01-11');
INSERT INTO customer VALUES('lalala', '홍지수', '미국 로스앤젤레스', '01010109090', '2007-05-16');
INSERT INTO customer VALUES('jjjeee', '홍은정', '대한민국 서울', '01099991111', '2004-08-17');
INSERT INTO customer VALUES('wow123', '이민혁', '일본 삿포로', '01011223344', '1994-05-31');
INSERT INTO customer VALUES('minjipark', '박민지', '프랑스 파리', '01088776655', '1998-04-08');
INSERT INTO customer VALUES('jy9987', '강지연', '일본 삿포로', '01012312323', '1996-09-01');
```

INSERT 문

```sql
INSERT INTO orders VALUES(NULL, 'jy9987', '프링글스', 3500, 2);
INSERT INTO orders VALUES(NULL, 'kiwi', '새우깡', 1200, 1);
INSERT INTO orders VALUES(NULL, 'hello', '홈런볼', 4200, 2);
INSERT INTO orders VALUES(NULL, 'minjipark', '맛동산', 2400, 1);
INSERT INTO orders VALUES(NULL, 'bunny', '오감자', 1500, 4);
INSERT INTO orders VALUES(NULL, 'jjjeee', '양파링', 2000, 1);
INSERT INTO orders VALUES(NULL, 'hello', '자갈치', 1300, 2);
INSERT INTO orders VALUES(NULL, 'jjjeee', '감자깡', 1200, 4);
INSERT INTO orders VALUES(NULL, 'bunny', '죠리퐁', 1500, 3);
INSERT INTO orders VALUES(NULL, 'kiwi', '꼬깔콘', 1700, 2);
INSERT INTO orders VALUES(NULL, 'hello', '버터링', 4000, 2);
INSERT INTO orders VALUES(NULL, 'minjipark', '칙촉', 4000, 1);
INSERT INTO orders VALUES(NULL, 'wow123', '콘초', 1700, 3);
INSERT INTO orders VALUES(NULL, 'imminji01', '꼬북칩', 2000, 2);
INSERT INTO orders VALUES(NULL, 'bunny', '썬칩', 1800, 5);
INSERT INTO orders VALUES(NULL, 'kiwi', '고구마깡', 1300, 3);
INSERT INTO orders VALUES(NULL, 'jy9987', '오징어집', 1700, 5);
INSERT INTO orders VALUES(NULL, 'jjjeee', '바나나킥', 2000, 4);
INSERT INTO orders VALUES(NULL, 'imminji01', '초코파이', 5000, 2);
```

SQL 문 내부적 실행 순서

- 홍지수 고객의 주소를 찾으시오.

```
SELECT addr FROM customer WHERE custname='홍지수';
```

(1) FROM customer

	custid	custname	addr	phone	birth
▶	bunny	강해린	대한민국 서울	01012341234	2000-02-23
	hello	이지민	대한민국 포항	01022221234	1999-08-08
	imminji01	강민지	영국 런던	01060001000	1995-01-11
	jjjeee	홍은정	대한민국 서울	01099991111	2004-08-17
	jy9987	강지연	일본 삿포로	01012312323	1996-09-01
	kiwi	최지수	미국 뉴욕	01050005000	1990-12-25
	lalala	홍지수	미국 로스앤젤레스	01010109090	2007-05-16
	minjipark	박민지	프랑스 파리	01088776655	1998-04-08
	wow123	이민혁	일본 삿포로	01011223344	1994-05-31

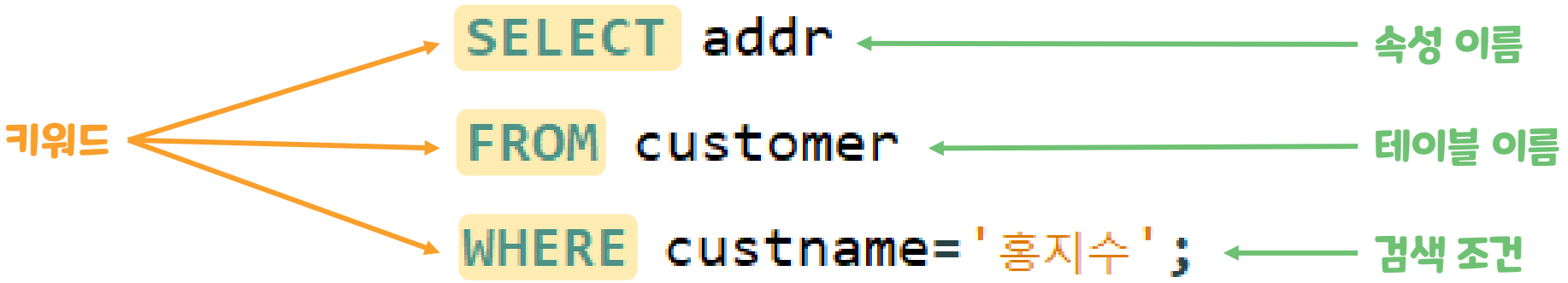
(2) WHERE custname='홍지수'

	custid	custname	addr	phone	birth
▶	lalala	홍지수	미국 로스앤젤레스	01010109090	2007-05-16

(3) SELECT addr

	addr
▶	미국 로스앤젤레스

SELECT 문 구성 요소



WHERE 조건 – 비교 연산자

=	같다
>	보다 크다
>=	보다 크거나 같다
<	보다 작다
<=	보다 작거나 같다

WHERE 조건 – 부정 연산자

!=	같지 않다.
^=	같지 않다.
<>	같지 않다.
NOT 컬럼명 =	~와 같지 않다.

WHERE 조건 – 범위, 집합, 패턴, NULL

BETWEEN a AND b	a와 b의 값 사이에 있으면 참 (a와 b 값도 포함)
IN (list)	리스트에 있는 값 중에서 어느 하나라도 일치하면 참
LIKE '비교문자열'	비교문자열과 형태가 일치하면 사용 (%, _ 사용) • `%` : 0개 이상의 어떤 문자 • `_` : 1개의 단일문자
IS NULL	NULL 값인 경우 true, 아니면 false

WHERE 조건 – 복합 조건

AND	앞에 있는 조건과 뒤에 오는 조건이 참(TRUE)가 되면 결과도 참(TRUE)
OR	앞에 있는 조건과 뒤에 오는 조건 중 하나라도 참(TRUE)면 결과는 참(TRUE)
NOT	뒤에 오는 조건과 반대되는 결과를 돌려준다.

와일드 문자 종류

와일드 문자	의미	예시
%	0개 이상의 문자열과 일치	'%서울%': 서울을 포함하는 문자열
_	특정 위치의 1개의 문자	'_민%': 두번째 글자가 '민'인 문자열

ORDER BY

- 결과가 출력되는 순서 조절
- where 절과 함께 사용 가능
  - 단, where 절 뒤에 나와야 함

```
SELECT 속성이름, ...
  FROM 테이블이름
  [WHERE 검색조건]
  [ORDER BY 속성이름]
```

- **ASC**: Ascending, **오름차순** (기본값)
- **DESC**: Descending, **내림차순**

UPDATE 문

- 테이블에서 특정 속성 값 **수정**

```
UPDATE 테이블명 SET 필드1 = 값1 WHERE 필드2 = 조건2;
```

DELETE 문

- 테이블의 기존 투플을 **삭제**

```
DELETE FROM 테이블명 WHERE 필드1 = 값1;
```

참고) DROP vs. TRUNCATE

```
DROP TABLE 테이블이름;
```

- 테이블 **삭제**하기
- 테이블을 잘못 만들었거나 더 이상 필요 없는 경우

```
TRUNCATE TABLE 테이블명;
```

- 테이블 **초기화**하기
- 테이블의 모든 행(row) 일괄 삭제

Select 심화

DISTINCT

- 중복된 데이터 제거

```
SELECT [DISTINCT] 속성이름, ...
  FROM 테이블이름
  [WHERE 검색조건]
  [ORDER BY 속성이름]
```

LIMIT

- 출력 개수 제한

```
SELECT [DISTINCT] 속성이름, ...
  FROM 테이블이름
  [WHERE 검색조건]
  [ORDER BY 속성이름]
  [LIMIT 개수]
```

집계 함수

SUM()	합계
AVG()	평균
MAX()	최대값
MIN()	최소값
COUNT()	행 개수
COUNT(DISTINCT)	중복 제외한 행 개수

GROUP BY

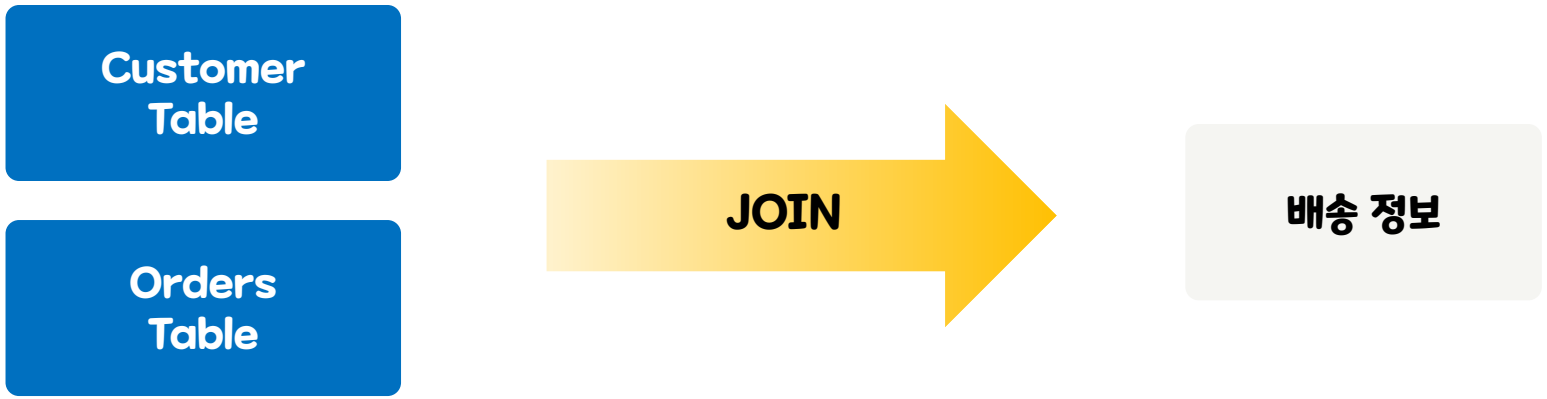
- **group by**
  - 속성이름끼리 그룹으로 묶는 역할
- **having**
  - group by절의 결과를 나타내는 그룹을 제한

```
SELECT [DISTINCT] 속성이름, ...
  FROM 테이블이름
  [WHERE 검색조건]
  [GROUP BY] 속성이름
  [HAVING] 조건식
  [ORDER BY 속성이름]
  [LIMIT 개수]
```

JOIN

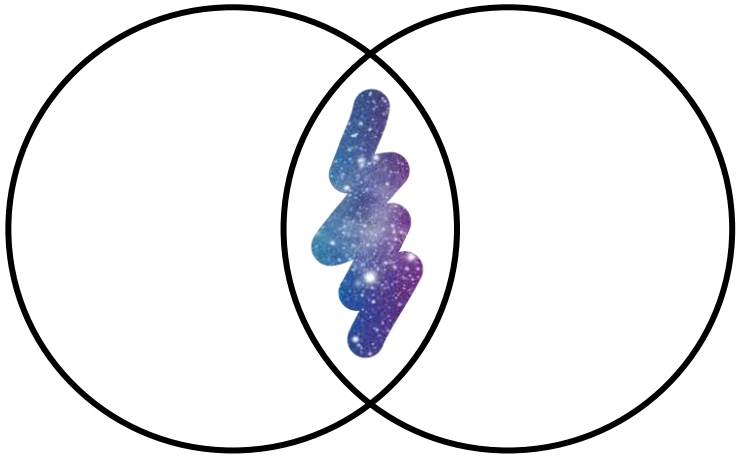
JOIN

- 두 테이블을 묶어서 하나의 테이블을 만듦
- 왜? 두 테이블을 엮어야 원하는 형태가 나오기도 함



Inner Join

```
SELECT 속성이름, ...
  FROM 테이블A, 테이블B
  WHERE 조인조건 AND 검색조건;
```



```
SELECT 속성이름, ...
  FROM 테이블A INNER JOIN 테이블B ON 조인조건
  WHERE 검색조건;
```