SQL

- SQL(**S**tructured **Q**uery **L**anguage, 구조적 질의 언어)은 **관계형 데이터베이스 시스템(RDBMS)을 제어하는 컴퓨터 언어이**다.
- 일반적인 프로그래밍 언어(범용 언어)와 달리 **대화식 언어**이기 때문에, **명령문이 짧고 간결하**다.
- SQL 자체는 범용 언어에 비해 한계가 있기 때문에, **단독으로 사용하기 보단** C#, Java, Python, PHP와 같은 **고수준 언어와 함께 쓰는 것이 일반적이다.**

SQL의 장점

- 명확하게 정의된 스키마
- 데이터의 무결성 보장
- 관계는 각 데이터를 중복없이 한번만 저장함
- SQL은 중복된 데이터가 없기 때문에 수정을 해야하는 경우 한번만 수행하면 됨.

SQL의 단점

- 상대적으로 덜 유연함. 데이터베이스 스키마는 사전에 계획하고 알고 사용해야 함.
- 스키마는 나중에 수정하기가 힘들거나 불가능할 수 있다.
- 관계를 맺고 있기 때문에, Join문이 많은 매우 복잡한 쿼리가 만들어 질 수 있다.

SQL 쿼리문은 역할에 따라 3가지로 분류가 가능하다.

DDL(Data Definition Language, 데이터 정의어)

- DB 오브젝트를 생성, 삭제, 변경하는 역할을 하며, DB 설계 단계에서 주로 사용된다.

- ex) CREATE, DROP, ALTER...

DML(Data Manipulation Language, 데이터 조작어)

- DB를 조회, 삽입, 삭제, 변경하는 역할을 하며, 관리 목적의 쿼리문이다.
- ex) SELECT, INSERT, UPDATE...

DCL(Data Control Language, 데이터 제어어)

- 사용자의 권한을 관리하는 역할을 한다.
- ex) GRANT, DENY, REVOKE...

보통 언어의 중요도는 DML > DDL > DCL 순이다.

DB를 조회하고, 관리하는 DML을 가장 많이 사용하고, DB의 테이블의 스키마를 수정하는 DDL을 그다음 으로 많이 사용함

DCL은 DBA가 주로 사용하며 일반 개발자는 사용할 일이 드물다.

SQL 사용 예시

DDL

[CREATE 문] 테이블 생성

```
CREATE TABLE 테이블_이름 (
        ❶ 속성_이름   데이터_타입  [NOT NULL] [DEFAULT 기본_값]
        ❷ [PRIMARY KEY (속성_리스트)]
        ❸ [UNIQUE (속성_리스트)]
        ❹ [FOREIGN KEY (속성_리스트)  REFERENCES  테이블_이름(속성_리스트)]
          [ON DELETE 옵션] [ON UPDATE 옵션]
        ❺ [CONSTRAINT 이름] [CHECK(조건)]
);
```

- [ ] 의 내용은 생략 가능
- SQL 질문의문은 대소문자를 구분하지 않음
- 데이터 무결성 제약조건 설정 가능

속성의 정의

데이터 타입	의미
INT 또는 INTEGER	정수
SMALLINT	INT보다 작은 정수
CHAR(n) 또는 CHARACTER(n)	길이가 n인 고정 길이의 문자열
VARCHAR(n) 또는 CHARACTER VARYING(n)	최대 길이가 n인 가변 길이의 문자열
NUMERIC(p, s) 또는 DECIMAL(p, s)	고정 소수점 실수 p는 소수점을 제외한 전체 숫자의 길이고, s는 소수점 이하 숫자의 길이
FLOAT(n)	길이가 n인 부동 소수점 실수
REAL	부동 소수점 실수
DATE	연, 월, 일로 표현되는 날짜
TIME	시, 분, 초로 표현되는 시간
DATETIME	날짜와 시간

[ALTER 문] 테이블 변경

새로운 속성 추가

```
ALTER TABLE 테이블_이름 ADD 속성_이름 데이터_타입 [NOT NULL] [DEFAULT
```

[DROP 문] 테이블 삭제

```
DROP TABLE 테이블_이름;
```

- 삭제할 테이블에 참조하는 테이블이 존재할 경우 삭제가 수행되지 않음
- 관련된 외래키 제약조건을 먼저 삭제해야 함

DML

[INSERT 문] 데이터 삽입

```
INSERT INTO 테이블_이름 [(속성_리스트)] VALUES (속성값_리스트);
```

- INTO 키워드와 함께 튜플을 삽입할 테이블의 이름과 속성의 이름을 나열
- VALUES 키워드와 함께 삽입할 속성 값들을 나열

[UPDATE 문] 데이터 수정

```
UPDATE 테이블_이름 SET 속성_이름1 = 값1, 속성_이름2 = 값2..[WHERE 조건]
```

- 테이블에 저장된 튜플에서 특정 속성의 값을 수정
- SET 키워드 다음에 속성 값을 어떻게 수정할 것인지를 지정

[DELETE 문] 데이터 삭제

```
DELETE FROM 테이블_이름 [WHERE 조건];
```

- WHERE절에 제시한 조건을 만족하는 튜플만 삭제

[SELECT 문] 데이터 조회

기본 검색

```
SELECT [ALL : DISTINCT] 속성_리스트 FROM 테이블_리스트;
```

- SELECT 키워드와 함께 검색하고 싶은 속성의 이름을 나열
- FROM 키워드와 함께 검색하고 싶은 속성이 있는 테이블의 이름을 나열

조건 검색

```
SELECT [ALL : DISTINCT] 속성_리스트 FROM 테이블_리스트 [WHERE 조건];
```

- 조건을 만족하는 데이터만 검색
- WHERE 키워드와 함께 비교 연산자와 논리 연산자를 이용한 검색 조건 제시

연산자	의미
=	같다.
< >	다르다.
<	작다.
>	크다.
<=	작거나 같다.
>=	크거나 같다.

연산자	의미
AND	모든 조건을 만족해야 검색한다.
OR	여러 조건 중 한 가지만 만족해도 검색한다.
NOT	조건을 만족하지 않는 것만 검색한다.

LIKE를 이용한 검색

- LIKE 키워드를 이용해 부분적으로 일치하는 데이터를 검색
- 문자열을 이용하는 조건에만 LIKE 키워드 사용 가능

사용 예	설명
LIKE '데이터%'	데이터로 시작하는 문자열 (데이터로 시작하기만 하면 길이는 상관 없음)
LIKE '%데이터'	데이터로 끝나는 문자열 (데이터로 끝나기만 하면 길이는 상관 없음)
LIKE '%데이터%'	데이터가 포함된 문자열
LIKE '데이터 _ _ _'	데이터로 시작하는 6자 길이의 문자열
LIKE '_ _ 한%'	세 번째 글자가 '한'인 문자열

정렬 검색

```
SELECT [ALL : DISTINCT] 속성_리스트 FROM 테이블_리스트
[WHERE 조건] [ORDER BY 속성_리스트 [ASC : DESC]];
```

- 오름차순(DEFAULT) : ASC / 내림차순 : DESC
- NULL 값은 오름차순에서 맨 마지막에 출력, 내림차순에서는 맨 먼저 출력

집계 함수를 이용한 검색

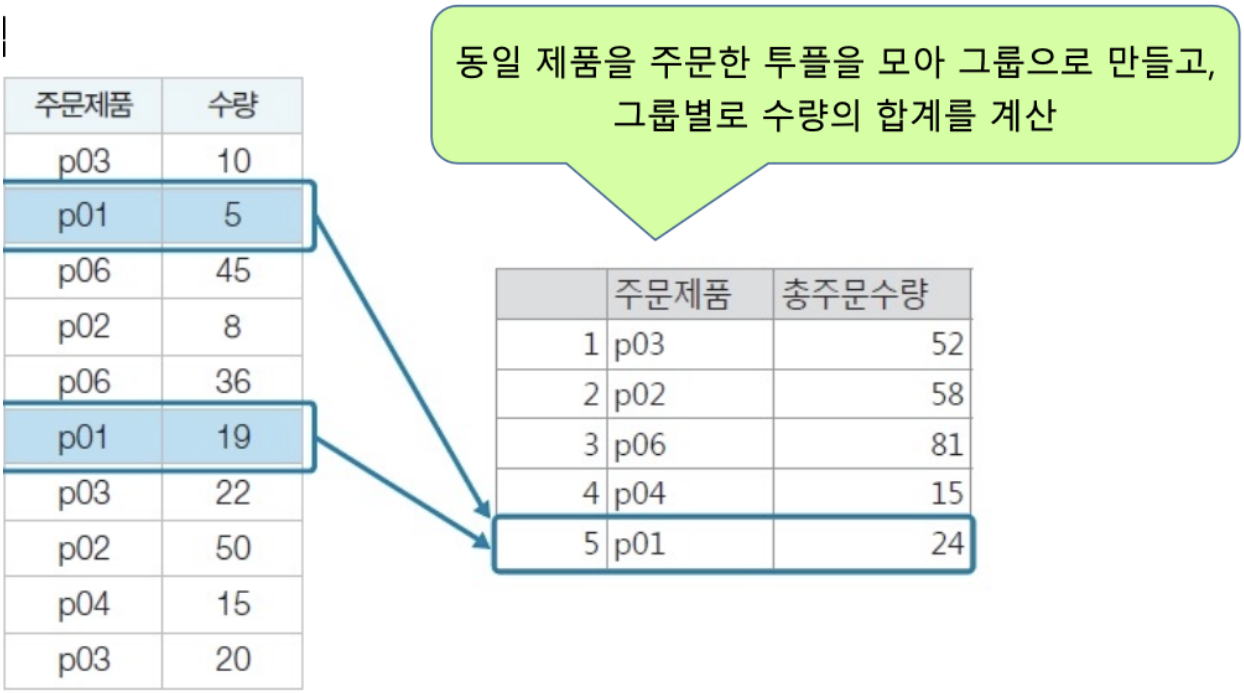
- SELECT, HAVING 에서만 사용 가능
- 집계 함수는 NULL 값은 제외하고 계산함

함수	의미	사용 가능한 속성의 타입
COUNT	속성 값의 개수	모든 데이터
MAX	속성 값의 최댓값	
MIN	속성 값의 최솟값	
SUM	속성 값의 합계	숫자 데이터
AVG	속성 값의 평균	

그룹별 검색

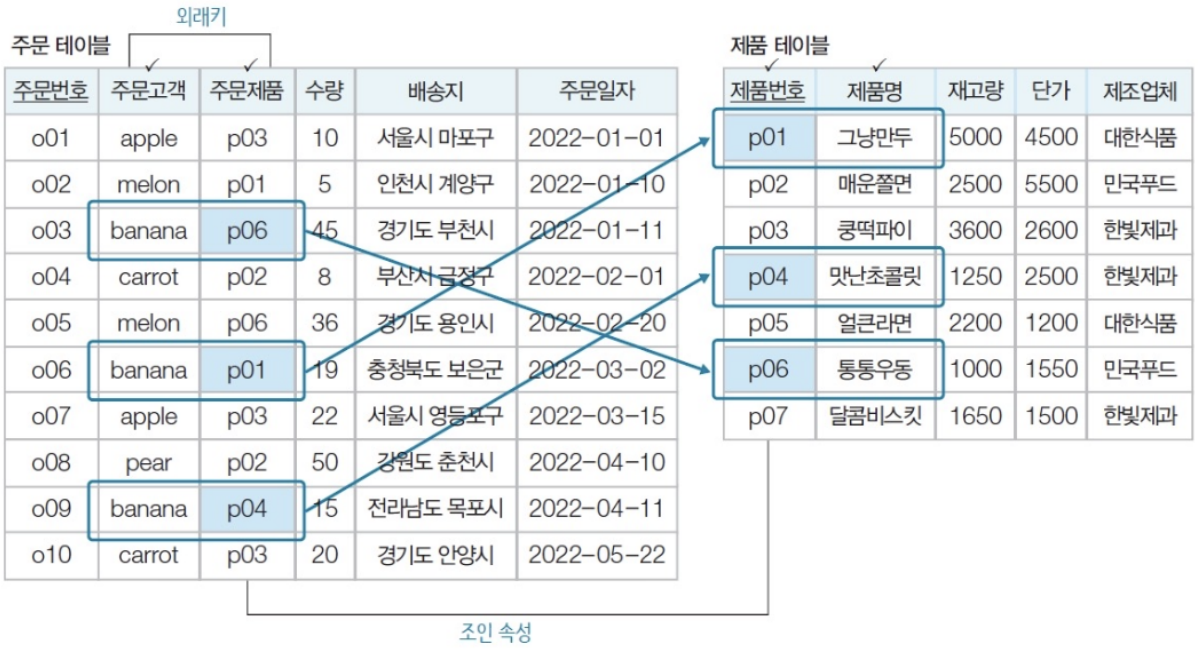
```
SELECT [ALL : DISTINCT] 속성_리스트 FROM 테이블_리스트
[WHERE 조건] [GROUP BY 속성_리스트 [HAVING 조건]]
[ORDER BY 속성_리스트 [ASC : DESC]];
```

- GROUP BY 키워드를 이용해 특정 속성의 값이 같은 투플을 모아 그룹을 만들고, 그룹별로 검색
- HAVING 키워드를 함께 이용해 그룹에 대한 조건을 작성
- 그룹을 나누는 기준이 되는 속성을 SELECT 절에도 작성하는 것이 좋음



여러 테이블에 대한 조인 검색

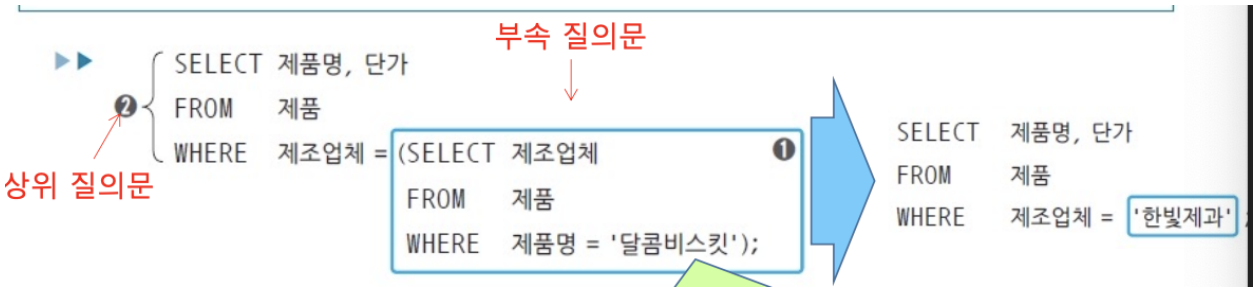
- 조인 검색: 여러 개의 테이블을 연결하여 데이터를 검색하는 것
- 조인 속성: 조인 검색을 위해 테이블을 연결해 주는 속성
- 연결하려는 테이블 간에 조인 속성의 이름은 달라도 되지만 도메인은 같아야 함
- 일반적으로 외래키를 조인 속성으로 이용함
- FROM 절에 검색에 필요한 모든 테이블을 나열
- 같은 이름의 속성이 서로 다른 테이블에 존재할 수 있기 때문에 속성 이름 앞에 해당 속성이 소속된 테이블의 이름을 표시




```
SELECT 속성_리스트 FROM 테이블1 INNER JOIN 테이블2 ON 조인조건
[WHERE 검색조건]
```

부속 질의문을 이용한 검색

- SELECT 문 안에 또 다른 SELECT 문을 포함하는 질의
- 부속 질의문을 먼저 수행하고, 그 결과를 이용해 상위 질의문을 수행
- 부속 질의문과 상위 질의문을 연결하는 연산자가 필요



MYSQL 내장함수

함수	설명
ABS(숫자)	숫자의 절댓값을 계산 ABS(-4.5) => 4.5
CEIL(숫자)	숫자보다 크거나 같은 최소의 정수 CEIL(4.1) => 5
FLOOR(숫자)	숫자보다 작거나 같은 최소의 정수 FLOOR(4.1) => 4
ROUND(숫자, m)	숫자의 반올림, m은 반올림 기준 자릿수 ROUND(5.36, 1) => 5.40
LOG(n, 숫자)	숫자의 자연로그 값을 반환 LOG(10) => 2.30259
POWER(숫자, n)	숫자의 n제곱 값을 계산 POWER(2, 3) => 8
SQRT(숫자)	숫자의 제곱근 값을 계산(숫자는 양수) SQRT(9.0) => 3.0
SIGN(숫자)	숫자가 음수면 -1, 0이면 0, 양수면 1 SIGN(3.45) => 1

반환 구분	함수	설명
문자값 반환 함수 s : 문자열 c : 문자 n : 정수 k : 정수	**CONCAT(s1,s2)**	두 문자열을 연결, CONCAT('마당', ' 서점') => '마당 서점'
	LOWER(s)	대상 문자열을 모두 소문자로 변환, LOWER('MR. SCOTT') => 'mr. scott'
	LPAD(s,n,c)	대상 문자열의 왼쪽부터 지정한 자리수까지 지정한 문자로 채움 LPAD('Page 1', 10, '*') => '****Page 1'
	REPLACE(s1,s2,s3)	대상 문자열의 지정한 문자를 원하는 문자로 변경 REPLACE('JACK & JUE', 'J', 'BL') => 'BLACK & BLUE'
	RPAD(s,n,c)	대상 문자열의 오른쪽부터 지정한 자리수까지 지정한 문자로 채움 RPAD('AbC', 5, '*') => 'AbC**'
	SUBSTR(s,n,k)	대상 문자열의 지정된 자리에서부터 지정된 길이만큼 잘라서 반환 SUBSTR('ABCDEFG', 3, 4) => 'CDEF'
	TRIM(c FROM s)	대상 문자열의 양쪽에서 지정된 문자를 삭제(문자열만 넣으면 기본값으로 공백 제거) TRIM('=' FROM '==BROWNING==') => 'BROWNING'
	UPPER(s)	대상 문자열을 모두 대문자로 변환 UPPER('mr. scott') => 'MR. SCOTT'
숫자값 반환 함수	**ASCII(c)**	대상 알파벳 문자의 아스키 코드 값을 반환, ASCII('D') => 68
	LENGTH(s)	대상 문자열의 Byte 반환, 알파벳 1byte, 한글 3byte (UTF8) LENGTH('CANDIDE') => 7
	CHAR_LENGTH(s)	문자열의 문자 수를 반환, CHAR_LENGTH('데이터') => 3