SQL 기본

SELECT 문

❖ 명칭

- 데이터베이스명.테이블명
  - `desc employees.titles;`

- 현재 사용 중인 데이터베이스의 명칭은 생략 가능
  - `use employees`
  - `desc titles;`

- SQL 명령어는 대소문자 구분하지 않음
- 사용자 정의 명칭은 구분하기도 함

❖ 주석

- `--`
  - 한 줄 주석문
- `/*  */`
  - 여러 줄 주석문

SELECT 문

❖ DESC

- 테이블의 구조를 출력

```
> use employees
> DESC titles;
```

```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| emp_no    | int(11)     | NO   | PRI | NULL    |       |
| title     | varchar(50) | NO   | PRI | NULL    |       |
| from_date | date        | NO   | PRI | NULL    |       |
| to_date   | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

SELECT 문

❖ DESC

- 테이블의 구조를 출력

```
> DESC employees;
```

```
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| emp_no     | int(11)       | NO   | PRI | NULL    |       |
| birth_date | date          | NO   |     | NULL    |       |
| first_name | varchar(14)   | NO   |     | NULL    |       |
| last_name  | varchar(16)   | NO   |     | NULL    |       |
| gender     | enum('M','F') | NO   |     | NULL    |       |
| hire_date  | date          | NO   |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
```

SELECT 문

❖ 샘플 데이터베이스 sqlDB

회원 테이블(userTbl)

아이디	이름	생년	지역	국번	전화번호	키	가입일
LSG	이승기	1987	서울	011	1111111	182	2008.8.8
KBS	김범수	1979	경남	011	2222222	173	2012.4.4
KKH	김경호	1971	전남	019	3333333	177	2007.7.7
JYP	조용필	1950	경기	011	4444444	166	2009.4.4
SSK	성시경	1979	서울			186	2013.12.12
LJB	임재범	1963	서울	016	6666666	182	2009.9.9
YJS	윤종신	1969	경남			170	2005.5.5
EJW	은지원	1978	경북	011	8888888	174	2014.3.3
JKW	조관우	1965	경기	018	9999999	172	2010.10.10
BBK	바비킴	1973	서울	010	0000000	176	2013.5.5

PK

구매 테이블(buyTbl)

순번	아이디	물품명	분류	단가	수량
1	KBS	운동화		30	2
2	KBS	노트북	전자	1000	1
3	JYP	모니터	전자	200	1
4	BBK	모니터	전자	200	5
5	KBS	청바지	의류	50	3
6	BBK	메모리	전자	80	10
7	SSK	책	서적	15	5
8	EJW	책	서적	15	2
9	EJW	청바지	의류	50	1
10	BBK	운동화		30	2
11	EJW	책	서적	15	1
12	BBK	운동화		30	2

PK FK

SELECT 문

❖ 샘플 데이터베이스 구축하기

- sqlDB.sql 파일을 c:\temp에 준비
- 시작 > MariaDB > Command Prompt

```
> cd \temp
> mysql -u root -p

MariaDB [sqlDB]> source sqlDB.sql
```

SELECT 문

❖ 샘플 데이터베이스 구축하기

- userTbl

```
SELECT * FROM userTbl;
```

```
+--------+--------+-----------+-------+---------+----------+--------+------------+
| userID | name   | birthYear | addr  | mobile1 | mobile2  | height | mDate      |
+--------+--------+-----------+-------+---------+----------+--------+------------+
| BBK    | 바비킴 |      1973 | 서울  | 010     | 00000000 |    176 | 2013-05-05 |
| EJW    | 은지원 |      1972 | 경북  | 011     | 88888888 |    174 | 2014-03-03 |
| JKW    | 조관우 |      1965 | 경기  | 018     | 99999999 |    172 | 2010-10-10 |
| JYP    | 조용필 |      1950 | 경기  | 011     | 44444444 |    166 | 2009-04-04 |
| KBS    | 김범수 |      1979 | 경남  | 011     | 22222222 |    173 | 2012-04-04 |
| KKH    | 김경호 |      1971 | 전남  | 019     | 33333333 |    177 | 2007-07-07 |
| LJB    | 임재범 |      1963 | 서울  | 016     | 66666666 |    182 | 2009-09-09 |
| LSG    | 이승기 |      1987 | 서울  | 011     | 11111111 |    182 | 2008-08-08 |
| SSK    | 성시경 |      1979 | 서울  | NULL    | NULL     |    186 | 2013-12-12 |
| YJS    | 윤종신 |      1969 | 경남  | NULL    | NULL     |    170 | 2005-05-05 |
+--------+--------+-----------+-------+---------+----------+--------+------------+
```

SELECT 문

❖ 샘플 데이터베이스 구축하기

- buyTbl

```
SELECT * FROM buyTbl;
```

```
+-----+--------+----------+-----------+-------+--------+
| num | userID | prodName | groupName | price | amount |
+-----+--------+----------+-----------+-------+--------+
|   1 | KBS    | 운동화   | NULL      |    30 |      2 |
|   2 | KBS    | 노트북   | 전자      |  1000 |      1 |
|   3 | JYP    | 모니터   | 전자      |   200 |      1 |
|   4 | BBK    | 모니터   | 전자      |   200 |      5 |
|   5 | KBS    | 청바지   | 의류      |    50 |      3 |
|   6 | BBK    | 메모리   | 전자      |    80 |     10 |
|   7 | SSK    | 책       | 서적      |    15 |      5 |
|   8 | EJW    | 책       | 서적      |    15 |      2 |
|   9 | EJW    | 청바지   | 의류      |    50 |      1 |
|  10 | BBK    | 운동화   | NULL      |    30 |      2 |
|  11 | EJW    | 책       | 서적      |    15 |      1 |
|  12 | BBK    | 운동화   | NULL      |    30 |      2 |
+-----+--------+----------+-----------+-------+--------+
```

SELECT 문

❖ SELECT

- 테이블의 내용 출력

```
SELECT 필드 목록
FROM 테이블명
[ WHERE 조건 ]
[ GROUP BY 컬럼명 ]
[ HAVING 조건 ]
[ ORDER BY 컬럼명 ]
```

```
USE sqlDB;
SELECT * FROM userTbl;
```

```
SELECT * FROM userTbl WHERE name='김경호';
```

SELECT 문

❖ SELECT

- ○ WHERE 절 - 관계 연산자(AND, OR, NOT)의 사용

```
SELECT userID, Name
FROM userTbl
WHERE birthYear >= 1970 AND height >= 182;
```

```
SELECT userID, Name
FROM userTbl
WHERE birthYear >= 1970 OR height >= 182;
```

SELECT 문

❖ SELECT

- WHERE 절 - BETWEEN … AND, IN() 그리고 LIKE

```
SELECT userID, Name
FROM userTbl
WHERE height >= 180 AND height <= 183;
```

```
SELECT userID, Name
FROM userTbl
WHERE height BETWEEN 180 AND 183;
```

SELECT 문

❖ SELECT

- WHERE 절 - BETWEEN … AND, IN() 그리고 LIKE

```
SELECT Name, addr
FROM userTbl
WHERE addr = '경남' OR addr = '전남' OR addr = '경북';
```

```
SELECT Name, addr
FROM userTbl
WHERE addr IN('경남', '전남', '경북');
```

SELECT 문

❖ SELECT

- WHERE 절 - BETWEEN … AND, IN() 그리고 LIKE

```
SELECT Name, height
FROM userTbl
WHERE name LIKE '김%';
```

cf) % : 아무 글자가 와도 상관없음 - 개 수에 제한 없음

```
SELECT Name, height
FROM userTbl
WHERE name LIKE '_종신';
```

cf) _ : 한 글자로 아무 글자가 와도 상관없음

SELECT 문

❖ SELECT

- 테이블의 내용 출력

```
USE employees;

SELECT first_name FROM employees;

SELECT first_name, last_name, gender FROM employees;
```

```
SELECT first_name AS 이름, gender AS 성별 FROM employees;

SELECT first_name 이름, gender 성별, hire_date 입사일 FROM employees;
```

SELECT 문

❖ SELECT

- ○ WHERE 절

```
SELECT *
FROM employees
WHERE last_name = 'Lenart';
```

```
SELECT *
FROM employees
WHERE birth_date >= '1960-01-01';
```

```
SELECT *
FROM employees
WHERE birth_date >= '1960-01-01'
  AND birth_date <= '1960-12-31';
```

```
SELECT *
FROM employees
WHERE birth_date  BETWEEN '1960-01-01' AND '1960-12-31';
```

SELECT 문

❖ SELECT

- WHERE 절

```
SELECT *
FROM employees
WHERE last_name = 'Lenart'
   OR last_name = 'Baaz'
   OR last_name = 'Pillow';
```

```
SELECT *
FROM employees
WHERE last_name IN('Lenart', 'Baaz', 'Pillow');
```

SELECT 문

❖ ANY/ALL/SOME 그리고 서브쿼리(SubQuery, 하위쿼리)

- 서브쿼리
  - FROM/WHERE 절에 SELECT 문을 제시
  - 서브 쿼리는 반드시 ()안에 작성

```
SELECT Name, height FROM userTBL WHERE height  > 177;
```

키가 177보다 큰 사람 출력

```
SELECT Name, height FROM userTbl
   WHERE height > (SELECT height FROM userTbl WHERE Name = '김경호');
```

김경호의 키보다 큰 사람 출력

```
SELECT Name, height FROM userTbl
   WHERE height >= (SELECT height FROM userTbl WHERE addr = '경남');
```

??

SELECT 문

❖ ANY/ALL/SOME 그리고 서브쿼리(SubQuery, 하위쿼리)

- ANY/ALL

```
SELECT height FROM userTbl WHERE addr = '경남';
```

```
170
180
```

```
SELECT Name, height FROM userTbl
   WHERE height >= ANY (SELECT height FROM userTbl WHERE addr = '경남');
```

170 이상 또는 180 이상 --> 170 이상

```
SELECT Name, height FROM userTbl
   WHERE height >= ALL (SELECT height FROM userTbl WHERE addr = '경남');
```

170 이상 그리고 180 이상 --> 180 이상

SELECT 문

❖ ANY/ALL/SOME 그리고 서브쿼리(SubQuery, 하위쿼리)

- SOME
  - ANY와 같은 의미
  - = 연산자와 사용 → IN과 동일

```
SELECT Name, height FROM userTbl
   WHERE height = SOME (SELECT height FROM userTbl WHERE addr = '경남');
```

```
SELECT Name, height FROM userTbl
  WHERE height IN (SELECT height FROM userTbl WHERE addr = '경남');
```

SELECT 문

❖ 원하는 순서대로 정렬하여 출력

- 오름차순 : ASC (디폴트, 생략가능)
- 내림차순: DESC

```
SELECT Name, mDate FROM userTbl ORDER BY mDate;
```

mDate의 오름 차순으로 정렬

```
SELECT Name, mDate FROM userTbl ORDER BY mDate DESC;
```

mDate의 내림차순으로 정렬

```
SELECT Name, height FROM userTbl ORDER BY height DESC, name ASC;
```

height로 내림차순 정렬하고,
height가 동률일 때는 name으로 오름차순 정렬

SELECT 문

❖ 중복된 것은 하나만 남기는 DISTINCT

```
SELECT addr FROM userTbl;
```

```
SELECT addr FROM userTbl ORDER BY addr;
```

```
SELECT DISTINCT addr FROM userTbl;
```

SELECT 문

❖ 출력하는 개수를 제한하는 LIMIT

```
USE employees;

SELECT emp_no, hire_date FROM employees
   ORDER BY hire_date ASC;
```

```
SELECT emp_no, hire_date FROM employees
   ORDER BY hire_date ASC
   LIMIT 5 ;
```

```
SELECT emp_no, hire_date FROM employees
   ORDER BY hire_date ASC
   LIMIT 0, 5;  -- LIMIT 5 OFFSET 0 과 동일
```

SELECT 문

❖ 테이블을 복사하는 CREATE TABLE ... SELECT

- 기존 테이블과 동일한 구조로 테이블 생성
- KEY 제약조건은 복사되지 않음
  - 필드의 이름, 타입, 길이, NULL 여부가 동일
- 특정 컬럼(SELECT 절 제시) 또는 특정 행만(WHERE절 제시) 복사 가능

```
형식]
CREATE TABLE 새로운 테이블명(SELECT 복사할열 FROM 기본테이블명)
```

```
USE sqlDB;

CREATE TABLE buyTbl2 (SELECT * FROM buyTbl);
SELECT * FROM buyTbl2;
```

```
CREATE TABLE buyTbl3 (SELECT userID, prodName FROM buyTbl);

SELECT * FROM buyTbl3;
```

특정 컬럼만 복사

SELECT 문

❖ GROUP BY 및 HAVING 그리고 집계 함수

- ○ GROUP BY 절
  - ▪ 특정 컬럼에 대해 동일한 값을 가지는 행들을 하나의 행으로 처리
  - ▪ 통계 작업에 사용

형식 :

```
SELECT select_expr
    [FROM table_references]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}]
```

SELECT 문

❖ GROUP BY 및 HAVING 그리고 집계 함수

- ○ GROUP BY 절

```
SELECT userID, amount
FROM buyTbl
ORDER BY userID;
```

```
SELECT userID, SUM(amount)
FROM buyTbl
GROUP BY userID;
```

```
SELECT userID AS '사용자 아이디', SUM(amount) AS '총 구매 개수'
FROM buyTbl
GROUP BY userID;
```

```
SELECT userID AS '사용자 아이디', SUM(price*amount) AS '총 구매액'
FROM buyTbl
GROUP BY userID;
```

SELECT 문

❖ GROUP BY 및 HAVING 그리고 집계 함수

- 집계 함수

함수명	설명
AVG( )	평균을 구한다.
MIN( )	최소값을 구한다.
MAX( )	최대값을 구한다.
COUNT( )	행의 개수를 센다.
COUNT(DISTINCT)	행의 개수를 센다(중복은 1개만 인정).
STDEV( )	표준편차를 구한다.
VAR_SAMP( )	분산을 구한다.

SELECT 문

❖ GROUP BY 및 HAVING 그리고 집계 함수

- ○ 집계 함수

```
USE sqlDB;

SELECT AVG(amount) AS '평균 구매 개수'
FROM buyTbl ;
```

```
SELECT userID, AVG(amount) AS '평균 구매 개수'
FROM buyTbl
GROUP BY userID;
```

```
SELECT name, MAX(height), MIN(height) FROM userTbl;
```

```
SELECT Name, MAX(height), MIN(height)
FROM userTbl
GROUP BY Name;
```

SELECT 문

❖ GROUP BY 및 HAVING 그리고 집계 함수

- ○ 집계 함수

```
SELECT Name, height
FROM userTbl
WHERE height = (SELECT MAX(height)FROM userTbl)
   OR height = (SELECT MIN(height)FROM userTbl) ;
```

```
SELECT COUNT(*) FROM userTbl;
```

전체 행의 개수

```
SELECT COUNT(mobile1) AS '휴대폰이 있는 사용자'
FROM userTbl;
```

지정한 컬럼에 NULL있는 행은 제외

SELECT 문

❖ GROUP BY 및 HAVING 그리고 집계 함수

- HAVING 절
  - GROUP BY 결과에서 필터링

```
USE sqlDB;

SELECT userID AS '사용자', SUM(price*amount) AS '총구매액'
FROM buyTbl
GROUP BY userID ;
```

```
SELECT userID AS '사용자', SUM(price*amount) AS '총구매액'
FROM buyTbl
WHERE price*amount > 100
GROUP BY userID ;
```

```
SELECT userID AS '사용자', SUM(price*amount) AS '총구매액'
FROM buyTbl
GROUP BY userID
HAVING SUM(price*amount) > 1000 ;
```

SELECT 문

❖ GROUP BY 및 HAVING 그리고 집계 함수

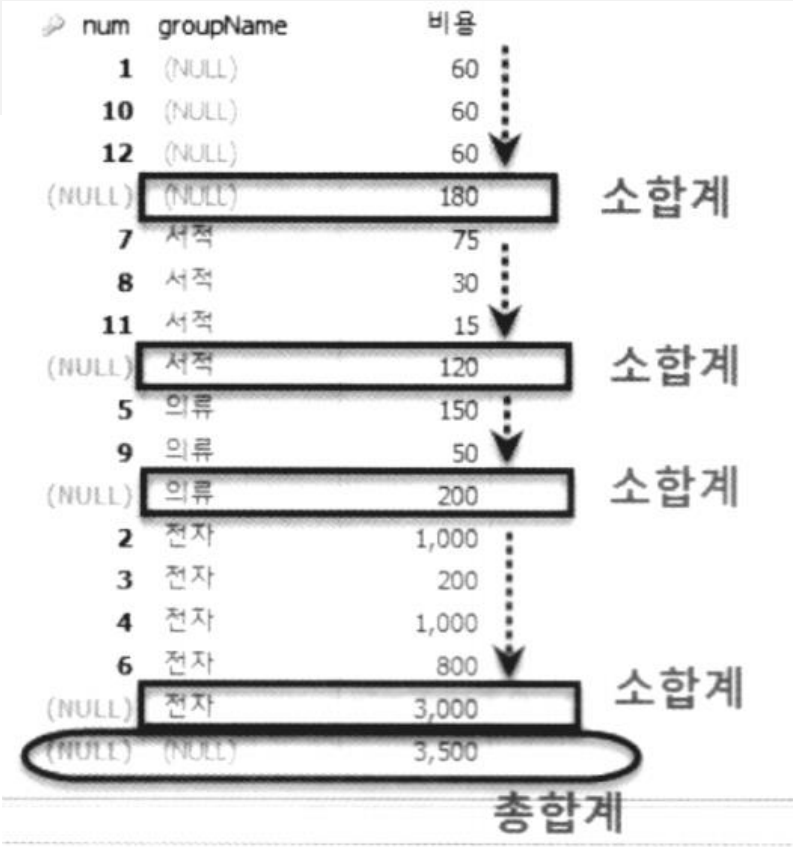
- ○ HAVING 절
  - ▪ GROUP BY 결과에서 필터링

```
SELECT userID AS '사용자', SUM(price*amount) AS '총구매액'
FROM buyTbl
GROUP BY userID
HAVING SUM(price*amount) > 1000
ORDER BY SUM(price*amount) ;
```

❖ GROUP BY 및 HAVING 그리고 집계 함수

- ○ ROLLUP
  - ▪ 중간 합계와 총 합을 출력

```
SELECT num, groupName, SUM(price * amount) AS '비용'
FROM buyTbl
GROUP BY  groupName, num
WITH ROLLUP;
```



SELECT 문

❖ GROUP BY 및 HAVING 그리고 집계 함수

- ROLLUP
  - 중간 합계와 총 합을 출력

```
SELECT groupName, SUM(price * amount) AS '비용'
FROM buyTbl
GROUP BY  groupName
WITH ROLLUP;
```

SELECT 문

❖ SQL 문의 종류

- DDL : Data Definition Language
  - 데이터베이스 객체(테이블, 인덱스, 뷰 등)의 생성, 수정, 삭제 조작

- DML : Data Manipulation Language
  - 데이터의 선택, 삽입, 수정, 삭제 등 데이터 조작
  - SELECT, INSERT, UPDATE, DELETE

- DCL : Data Control Language
  - 권한 부여

데이터 변경을 위한 SQL 문

❖ 데이터의 삽입: INSERT

```
[형식]
INSERT INTO 테이블[(열1, 열2, …)]
VALUES(값1, 값2, …)
```

- 열을 생략하면 테이블에 있는 모든 열에 대해 값을 제시
  이경우 순서는 테이블 생성시 제시한 열의 순서
- 열을 제시하면 순서에 맞춰 값 제시

```
USE sqlDB;

CREATE TABLE testTBL1 (id  int, userName char(3), age int);

INSERT INTO testTBL1 VALUES (1, '홍길동', 25);
INSERT INTO testTBL1(id, userName) VALUES (2, '설현');
INSERT INTO testTBL1(userName, age, id) VALUES ('초아', 26,  3);
```

데이터 변경을 위한 SQL 문

❖ 자동 증가하는 AUTO_INCREMENT

- 값을 제시하지 않은 경우 자동 증가 값으로 추가
- PRIMARY KEY 필드에 주로 사용

```
USE  sqlDB;

CREATE TABLE testTBL2 (
   id  int AUTO_INCREMENT PRIMARY KEY,
   userName char(3),
   age int
);

INSERT INTO testTBL2 VALUES (NULL, '지민', 25);
INSERT INTO testTBL2 VALUES (NULL, '유나', 22);
INSERT INTO testTBL2 VALUES (NULL, '유경', 21);
```

```
SELECT * FROM testTBL2;
```

데이터 변경을 위한 SQL 문

❖ 자동 증가하는 AUTO_INCREMENT

- 시작 값 변경

```
ALTER TABLE testTBL2 AUTO_INCREMENT=100;

INSERT INTO testTBL2 VALUES (NULL, '찬미', 23);
SELECT * FROM testTBL2;
```

데이터 변경을 위한 SQL 문

❖ 자동 증가하는 AUTO_INCREMENT

- 증가 값 변경
  - 테이블 별로 설정 불가
  - @@auto_increment_increment 서버변수로 전역 설정

```
USE  sqlDB;
CREATE TABLE testTBL3
  (id  int AUTO_INCREMENT PRIMARY KEY,
   userName char(3),
   age int );
ALTER TABLE testTBL3 AUTO_INCREMENT=1000;

SET @@auto_increment_increment=3;

INSERT INTO testTBL3 VALUES (NULL, '나연', 20);
INSERT INTO testTBL3 VALUES (NULL, '정연', 18);
INSERT INTO testTBL3 VALUES (NULL, '모모', 19);
SELECT * FROM testTBL3;
```

데이터 변경을 위한 SQL 문

❖ 다중 입력

```
INSERT INTO testTBL3 VALUES (NULL, '나연', 20);
INSERT INTO testTBL3 VALUES (NULL, '정연', 18);
INSERT INTO testTBL3 VALUES (NULL, '모모', 19);
```

```
INSERT INTO testTBL3 VALUES
(NULL, '나연', 20),
(NULL, '정연', 18),
(NULL, '모모', 19);
```

데이터 변경을 위한 SQL 문

❖ 대량의 샘플 데이터 생성

```
형식:
INSERT INTO 테이블이름(열 이름1, 열 이름2, …)
SELECT 문;
```

```
USE sqlDB;
CREATE TABLE testTBL4 (id int, Fname varchar(50), Lname varchar(50));

INSERT INTO testTBL4
SELECT emp_no, first_name, last_name
FROM employees.employees ;
```

```
USE sqlDB;
CREATE TABLE testTBL5
   (SELECT emp_no, first_name, last_name  FROM employees.employees) ;
```

테이블 생성과 서브 쿼리로 데이터 삽입

데이터 변경을 위한 SQL 문

❖ 데이터의 수정 : UPDATE 문

```
형식:
UPDATE 테이블이름
SET
    열1=값1,
    열2=값2,
   …
[WHERE 조건];
```

※ WHERE 조건이 없으면 전체 행이 수정

```
UPDATE testTBL4
SET Lname = '없음'
WHERE Fname = 'Kyoichi';
```

```
USE sqlDB;

UPDATE buyTBL2
SET price = price * 1.5 ;
```

데이터 변경을 위한 SQL 문

❖ 데이터의 삭제: DELETE FROM

```
DELETE FROM 테이블 이름
[WHERE 조건];
```

※ WHERE 조건이 없으면 전체 행이 삭제

```
USE sqlDB;
DELETE FROM testTBL4 WHERE Fname = 'Aamer';

DELETE FROM testTBL4 WHERE Fname = 'Mary' LIMIT 5;
```