

본 자료와 관련 영상 콘텐츠는 저작권법 제25조 2항에 의해 보호를 받습니다.
본 콘텐츠 및 콘텐츠 일부 문구 등을 외부에 공개하거나, 요약해서 게시하지 말아주세요.
Copyright [잔재미코딩](#) Dave Lee

중급 SQL 익히기

한번은 들어보면 좋은 SQL 관련 기술과 일정 난이도가 있는 SQL 기술에 대한 정리

1. 외래키 (FOREIGN KEY)

FOREIGN KEY 생성해보기

```
DROP DATABASE IF EXISTS sqlDB;
CREATE DATABASE sqlDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

USE sqlDB;
DROP TABLE IF EXISTS userTbl;
CREATE TABLE userTbl (
    userID CHAR(8) NOT NULL PRIMARY KEY,
    name VARCHAR(10) UNIQUE NOT NULL,
    birthYear INT NOT NULL,
    addr CHAR(2) NOT NULL,
    mobile1 CHAR(3),
    mobile2 CHAR(8),
    height SMALLINT,
    mDate DATE,
    UNIQUE INDEX idx_userTbl_name (name),
    INDEX idx_userTbl_addr (addr)
) DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

DROP TABLE IF EXISTS buyTbl;
CREATE TABLE buyTbl (
    num INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
    userID CHAR(8) NOT NULL,
    prodName CHAR(4),
    groupName CHAR(4),
    price INT NOT NULL,
    amount SMALLINT NOT NULL,
    FOREIGN KEY (userID) REFERENCES userTbl(userID)
) DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

```
INSERT INTO userTbl VALUES('LSG', '이승기', 1987, '서울', '011', '11111111',
182, '2008-8-8');
INSERT INTO buyTbl (userID, prodName, groupName, price, amount)
VALUES('LSG', '운동화', '의류', 30, 2);
```

buyTbl에 데이터를 추가해보기 (실습)

```
INSERT INTO buyTbl (userID, prodName, groupName, price, amount)
VALUES('STJ', '운동화', '의류', 30, 2);
```

에러가 나면 정상임

- userTbl 에 userID가 STJ인 데이터가 없기 때문에,
 - FOREIGN KEY (userID) REFERENCES userTbl(userID)
 - buyTbl 테이블의 userID 컬럼은 userTbl 테이블의 userID를 참조할 때, userTbl 테이블에 userID가 STJ인 데이터가 없으면, 입력이 안됨
 - 데이터 무결성 (두 테이블간 관계에 있어서, 데이터의 정확성을 보장하는 제약 조건을 넣는 것임)
 - 현업에서는 꼭 필요한 경우만 사용하는 경우가 많음 (비즈니스 로직이 다양하기 때문에, 제약을 걸어놓을 경우, 예외적인 비즈니스 로직 처리가 어렵기 때문)

다음 데이터 넣어보기 (실습)

```
INSERT INTO buyTbl (userID, prodName, groupName, price, amount)
VALUES('LSG', '운동화', '의류', 30, 2);
```

다음 SQL query 실행해보기

- 이번에는 userTbl 에 userID가 STJ 인 데이터를 넣어준 후에, 다시 buyTbl userID에 STJ 관련 데이터를 넣어줍니다.

```
INSERT INTO userTbl VALUES('STJ', '서태지', 1975, '경기', '011', '00000000',
171, '2014-4-4');
INSERT INTO buyTbl (userID, prodName, groupName, price, amount)
VALUES('STJ', '운동화', '의류', 30, 2);
```

다음 SQL query 실행해보기

- 이번에는 userTbl에 userID가 STJ 관련 데이터를 삭제해봅니다.

```
DELETE FROM userTbl WHERE userID = 'STJ'
```

에러나면 정상입니다.

- buyTbl에 해당 userID를 참조하는 데이터가 있기 때문입니다.

2. GROUP BY와 HAVING

환경 구축

bestproducts 데이터베이스 생성 후, bestproducts_items.sql bestproducts_ranking.sql
으로 테이블 만들기

- HAVING 절은 집계함수를 가지고 조건비교를 할 때 사용한다.
- HAVING절은 GROUP BY절과 함께 사용
- 예

```
SELECT provider FROM items GROUP BY provider HAVING COUNT(*) >= 100;
```

HAVING절을 포함한 복합 검색

```
SELECT provider, COUNT(*)  
FROM items  
WHERE provider != '스마일배송' -- 스마일배송은 제외  
GROUP BY provider -- 판매처별로 그룹  
HAVING COUNT(*) > 100 -- 베스트상품이 100개 이상 등록된 경우만 검색  
ORDER BY COUNT(*) DESC; -- 베스트상품 등록갯수 순으로 검색
```

3. JOIN 구문 익히기

- JOIN은 두 개 이상의 테이블로부터 필요한 데이터를 연결해 하나의 포괄적인 구조로 결합시키는 연산
- JOIN은 다음과 같이 세분화될 수 있지만, 보통은 **INNER JOIN**을 많이 사용함
 - INNER JOIN (일반적인 JOIN): 두 테이블에 해당 필드값이 매칭되는 (두 테이블의 모든 필드로 구성된) 레코드만 가져옴

- OUTER JOIN (참고)

- LEFT OUTER JOIN: 왼쪽 테이블에서 모든 레코드와 함께, 오른쪽 테이블에 왼쪽 테이블 레코드와 매칭되는 레코드를 붙여서 가져옴
- RIGHT OUTER JOIN: 오른쪽 테이블에서 모든 레코드와 함께, 왼쪽 테이블에 왼쪽 테이블 레코드와 매칭되는 레코드를 붙여서 가져옴

3.1 JOIN (INNER JOIN)

- INNER JOIN은 조인하는 테이블의 ON 절의 조건이 일치하는 결과만 출력
- 사용법: FROM 테이블1 **INNER JOIN** 테이블2 **ON** 테이블1과 테이블2의 매칭조건

```
SELECT * FROM items INNER JOIN ranking ON ranking.item_code =  
items.item_code WHERE ranking.main_category = "ALL"
```

- 테이블 이름 다음에 한칸 띄고 새로운 이름을 쓰면, SQL구문 안에서 해당 이름으로 해당 테이블을 가리킬 수 있음 (AS 용법과 동일함)
 - 일반적으로 이와 같이 많이 사용됨

```
SELECT * FROM items a INNER JOIN ranking b ON a.item_code = b.item_code  
WHERE b.main_category = "ALL"
```

연습문제

전체 베스트상품(ALL 메인 카테고리)에서 판매자별 베스트상품 갯수 출력해보기

```
SELECT items.provider, COUNT(*)  
FROM ranking  
JOIN items ON items.item_code=ranking.item_code  
WHERE ranking.main_category = "ALL"  
GROUP BY items.provider
```

연습문제

메인 카테고리가 패션의류인 서브 카테고리 포함, 패션의류 전체 베스트상품에서 판매자별 베스트상품 갯수가 5이상인 판매자와 베스트상품 갯수 출력해보기

```
SELECT items.provider, COUNT(*) FROM ranking
JOIN items ON items.item_code=ranking.item_code
WHERE ranking.main_category='패션의류'
GROUP BY items.provider HAVING COUNT(*) >= 5;
```

연습문제

메인 카테고리가 신발/잡화인 서브 카테고리 포함, 전체 베스트상품에서 판매자별 베스트상품 갯수가 10이상인 판매자와 베스트상품 갯수를 베스트상품 갯수 순으로 출력해보기

```
SELECT items.provider, COUNT(*) AS best_product_count FROM ranking
JOIN items ON items.item_code=ranking.item_code
WHERE ranking.main_category='신발/잡화'
GROUP BY items.provider HAVING best_product_count >= 10
ORDER BY best_product_count DESC;
```

연습문제

메인 카테고리가 화장품/헤어인 서브 카테고리 포함, 전체 베스트상품의 평균, 최대, 최소 할인 가격 출력해보기

```
SELECT
    AVG(dis_price), MAX(dis_price), MIN(dis_price)
FROM
    items I
JOIN
    ranking R ON I.item_code = R.item_code
WHERE
    R.main_category = '화장품/헤어'
```

3.2 OUTER JOIN (참고)

- OUTER JOIN은 조인하는 테이블의 ON 절의 조건 중 한쪽의 데이터를 모두 가져옴
- OUTER JOIN은 LEFT OUTER JOIN, RIGHT OUTER JOIN 이 있음

LEFT OUTER JOIN

- 예:

```
SELECT * FROM customer_table C LEFT OUTER JOIN order_table O ON
C.customer_id = O.customer_id
```

customer_table			order_table			
id	customer_id	tomer_name	id	order_id	customer_id	order_date
1	1	Robert	1	100	1	2017.10.22 2:35:50
2	2	Peter	2	200	4	2017.10.23 2:35:50
3	3	Smith	3	300	2	2017.10.24 2:35:50
SELECT * FROM customer_table C LEFT OUTER JOIN order_table O						
id	customer_id	tomer_name	id	order_id	customer_id	order_date
1	1	Robert	1	100	1	2017.10.22 2:35:50
2	2	Peter	3	300	2	2017.10.24 2:35:50
3	3	Smith	NaN	NaN	NaN	NaN

customer_table			order_table			
id	customer_id	tomer_name	id	order_id	customer_id	order_date
1	1	Robert	1	100	1	2017.10.22 2:35:50
2	2	Peter	2	200	4	2017.10.23 2:35:50
3	3	Smith	3	300	2	2017.10.24 2:35:50
SELECT * FROM order_table O LEFT OUTER JOIN customer_table C						
id	order_id	customer_id	order_date	id	customer_id	customer_name
1	100	1	2017.10.22 2:35:50	1	1	Robert
3	300	2	2017.10.24 2:35:50	2	2	Peter
2	200	4	2017.10.23 2:35:50	NaN	NaN	NaN

RIGHT OUTER JOIN

- 예:

```
SELECT * FROM customer_table C RIGHT OUTER JOIN order_table O ON
C.customer_id = O.customer_id
```

customer_table			order_table			
id	customer_id	customer_name	id	order_id	customer_id	order_date
1	1	Robert	1	100	1	2017.10.22 2:35:50
2	2	Peter	2	200	4	2017.10.23 2:35:50
3	3	Smith	3	300	2	2017.10.24 2:35:50
SELECT * FROM customer_table C RIGHT OUTER JOIN order_table O						
id	customer_id	customer_name	id	order_id	customer_id	order_date
1	1	Robert	1	100	1	2017.10.22 2:35:50
2	2	Peter	3	300	2	2017.10.24 2:35:50
NaN	NaN	NaN	2	200	4	2017.10.23 2:35:50

연습문제

sakila 데이터베이스에서 address 테이블에는 address_id 가 있지만, customer 테이블에는 없는 데이터의 갯수 출력하기

```
SELECT COUNT(*) FROM customer C
RIGHT OUTER JOIN address A
ON A.address_id=C.address_id
WHERE customer_id IS NULL
```

4. 서브 쿼리 (MySQL SubQuery)

- SQL문 안에 포함되어 있는 SQL문
 - SQL문 안에서 괄호() 를 사용해서, 서브쿼리문을 추가할 수 있음
- 테이블과 테이블간의 검색시, 검색 범위(테이블 중 필요한 부분만 먼저 가져오도록)를 좁히는 기능에 주로 사용

서브쿼리(Sub Query) 사용법

- JOIN은 출력 결과에 여러 테이블의 열이 필요한 경우 유용
- 대부분의 서브쿼리(Sub Query) 는 JOIN 문으로 처리가 가능

예1: 서브카테고리가 '여성신발'인 상품 타이틀만 가져오기

- JOIN SQL을 사용해서 작성하는 방법

```
SELECT title
FROM items
INNER JOIN ranking ON items.item_code = ranking.item_code
WHERE ranking.sub_category = '여성신발'
```

- 서브쿼리를 사용해서 작성하는 방법
 - 컬럼값 IN 서브쿼리 출력값 -> 컬럼값과 서브쿼리값이 같을 때

```
SELECT title
FROM items
WHERE item_code IN
  (SELECT item_code FROM ranking WHERE sub_category = '여성신발')
```

예2: 서브카테고리가 '여성신발'인 상품중 할인가격이 가장 높은 상품의 할인가격 가져오기

- JOIN SQL을 사용해서 작성하는 방법

```
SELECT MAX(items.dis_price)
FROM items
INNER JOIN ranking ON items.item_code = ranking.item_code
WHERE ranking.sub_category = '여성신발'
```

- 서브쿼리를 사용해서 작성하는 방법

```
SELECT MAX(dis_price)
FROM items
WHERE item_code IN
  (SELECT item_code FROM ranking WHERE sub_category = '여성신발')
```

참고: 다양한 서브 쿼리 삽입 위치

- 비교


```
SELECT category_id, COUNT(*) AS film_count FROM film_category
WHERE film_category.category_id >
      (SELECT category.category_id FROM category WHERE category.name =
'Comedy')
GROUP BY film_category.category_id
```

- FROM 절

```
SELECT
a, b, c
FROM
(SELECT * FROM atoz_table)
```

연습문제

메인 카테고리별로 할인 가격이 10만원 이상인 상품이 몇개 있는지를 출력해보기 (JOIN 활용 SQL 과 서브쿼리 활용 SQL 모두 작성해보기)

JOIN 활용 SQL

```
SELECT ranking.main_category, COUNT(*)
FROM ranking JOIN items ON items.item_code = ranking.item_code
WHERE items.dis_price >= 100000
GROUP BY ranking.main_category;
```

서브쿼리 활용 SQL

```
SELECT ranking.main_category, COUNT(*)
FROM ranking WHERE item_code IN
      (SELECT item_code FROM items WHERE dis_price >= 100000)
GROUP BY main_category;
```

연습문제

'items' 테이블에서 'dis_price'가 200000 이상인 아이템들 중, 각 'sub_category'별 아이템 수

를 구하시오. 이때 서브쿼리를 사용하시오.

JOIN 활용 SQL

```
SELECT ranking.sub_category, COUNT(*)
FROM ranking
JOIN items ON ranking.item_code = items.item_code
WHERE items.dis_price >= 200000
GROUP BY ranking.sub_category;
```

서브쿼리 활용 SQL

```
SELECT ranking.sub_category, COUNT(*)
FROM ranking WHERE item_code IN
    (SELECT item_code FROM items WHERE dis_price >= 200000)
GROUP BY sub_category;
```

파이널 SQL 연습

SQL 연습도 하면 할 수록, 도움이 됩니다
좀더 연습하실 수 있도록 추가 문제로 SQL 기본기를 다져보세요~

연습문제

메인 카테고리, 서브 카테고리에 대해, 평균할인가격과 평균할인율을 출력해보기

```
SELECT main_category, sub_category, AVG(dis_price), AVG(discount_percent)
FROM ranking JOIN items ON items.item_code = ranking.item_code
GROUP BY main_category, sub_category;
```

연습문제

판매자별, 베스트상품 갯수, 평균할인가격, 평균할인율을 베스트상품 갯수가 높은 순으로 출력해

보기

```
SELECT
  provider,
  COUNT(*),
  AVG(dis_price),
  AVG(discount_percent)
FROM items
GROUP BY provider
ORDER BY COUNT(*) DESC
```

연습문제

각 메인 카테고리에서 베스트 상품 갯수가 20개 이상인 판매자의 판매자별 평균할인가격, 평균할인율, 베스트 상품 갯수 출력해보기

```
SELECT ranking.main_category, items.provider, COUNT(*),
AVG(items.dis_price), AVG(items.discount_percent)
FROM items JOIN ranking ON items.item_code = ranking.item_code
GROUP BY ranking.main_category, items.provider HAVING COUNT(*) >= 20;
```

연습문제

'items' 테이블에서 'dis_price'가 50000 이상인 상품들 중, 각 'main_category'별 평균 'dis_price'와 'discount_percent' 출력해보기

```
SELECT ranking.main_category, AVG(items.dis_price),
AVG(items.discount_percent)
FROM ranking
JOIN items ON ranking.item_code = items.item_code
WHERE items.dis_price >= 50000
GROUP BY ranking.main_category;
```

[참고] 7. 인덱스 (MySQL INDEX)

- 데이터베이스 분야에 있어서 테이블에 대한 동작의 속도를 높여주는 자료 구조
- 인덱스를 저장하는 데 필요한 디스크 공간은 보통 테이블을 저장하는 데 필요한 디스크 공간보다 작음
 - 인덱스는 키-필드만 갖고 있고, 나머지 세부 테이블 컬럼 정보는 가지고 있지 않기 때문
- 어떤 데이터를 인덱스로 만드느냐에 따라 방대한 데이터의 경우 성능에 큰 영향을 미칠 수 있음

7.1. 인덱스 종류

- 클러스터형 인덱스: 영어 사전과 같은 형태로 데이터를 재정렬하여 저장한다고 생각하면 됨
 - 테이블의 데이터를 실제로 재정렬하여 디스크에 저장
 - 테이블에 PRIMARY KEY 로 정의한 컬럼이 있을 경우, 자동 생성
 - 한 테이블당 하나의 클러스터형 인덱스만 가질 수 있음
- 보조 인덱스: 데이터는 그대로 두고, 일반 책 뒤에 있는 <찾아보기> 와 같은 형태가 만들어진다고 생각하면 됨
 - 클러스터형 인덱스와는 달리 데이터를 디스크에 재정렬하지 않고, 각 데이터의 위치만 빠르게 찾을 수 있는 구조로 구성

userTbl 테이블 생성

```
CREATE TABLE userTbl (  
  userID CHAR(8) NOT NULL PRIMARY KEY,  
  name VARCHAR(10) NOT NULL,  
  birthYear INT NOT NULL,  
  addr CHAR(2) NOT NULL,  
  mobile1 CHAR(3),  
  mobile2 CHAR(8),  
  height SMALLINT,  
  mDate DATE  
);
```

인덱스 확인

```
SHOW INDEX FROM userTbl
```

- Key_name 이 PRIMARY 로 된 것은 클러스터형 인덱스를 의미
- Column_name 이 userID 임을 확인할 수 있음
- (참고) 주요 인덱스 컬럼

- Table: The name of the table.
- Non_unique: 0 if the index cannot contain duplicates, 1 if it can.
- Key_name: The name of the index. If the index is the primary key, the name is always PRIMARY.
- Seq_in_index: The column sequence number in the index, starting with 1.
- Column_name: The column name.
- Collation: How the column is sorted in the index. This can have values A (ascending) or NULL (not sorted).
- Cardinality: An estimate of the number of unique values in the index.
- Index_type: The index method used (BTREE, FULLTEXT, HASH, RTREE).

buyTbl 테이블 구조

```
CREATE TABLE buyTbl (
  num INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  userID CHAR(8) NOT NULL,
  prodName CHAR(4),
  groupName CHAR(4),
  price INT NOT NULL,
  amount SMALLINT NOT NULL,
  FOREIGN KEY (userID) REFERENCES userTbl(userID)
);
```

- Key_name 이 PRIMARY 가 아닌 것은 보조 인덱스를 의미
- foreign key로 설정된 컬럼이 인덱스가 없다면, 인덱스를 자동 생성

참고: 테이블 변경

```
ALTER TABLE userTbl ADD [CONSTRAINT TESTDate] UNIQUE(mDate);
```

- ALTER TABLE 테이블이름 ADD [CONSTRAINT 제약조건명] UNIQUE(컬럼명)
 - 테이블에 특정 컬럼에 duplicate 값이 나오지 않도록 제약조건을 추가하기

참고: UNIQUE 제약을 넣으면, 보조 테이블이 만들어짐

```
SHOW INDEX FROM userTbl
```

7.2. 인덱스 생성 및 삭제

- 인덱스를 필요에 따라 생성/삭제 가능

생성된 테이블에 인덱스 추가하기

- 기본 문법
 - CREATE INDEX 인덱스명 ON 테이블명 (column 1, column 2, ...);
 - ALTER TABLE 테이블명 ADD INDEX 인덱스명 (column 1, column 2, ...);
- 생성된 테이블에 인덱스 추가 예제 (CREATE INDEX 사용)

```
CREATE INDEX idx_name ON userTbl (name);
```

- 인덱스 확인

```
SHOW INDEX FROM userTbl;
```

- 생성된 테이블에 인덱스 추가 예제 (ALTER TABLE 사용)

```
ALTER TABLE userTbl ADD INDEX idx_addr (addr);
```

- 인덱스 확인

```
SHOW INDEX FROM userTbl;
```

연습문제

groupName 으로 인덱스 추가하고 확인해보기

```
CREATE INDEX idx_prodName ON buyTbl(prodName);
```

연습문제

prodName 으로 인덱스 추가하고 확인해보기

```
CREATE INDEX idx_prodName ON buyTbl(prodName);
```

7.3. 테이블 생성하며 인덱스도 함께 만들기

- 기본 문법
 - INDEX <인덱스명> (컬럼명1, 컬럼명2)
 - UNIQUE INDEX <인덱스명> (컬럼명) --> 항상 유일해야 함.
 - UNIQUE INDEX 의 경우 컬럼명은 유일한 값을 가지고 있어야 함

```
DROP DATABASE IF EXISTS sqlDB;
CREATE DATABASE sqlDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

USE sqlDB;
DROP TABLE IF EXISTS userTbl;
CREATE TABLE userTbl (
    userID CHAR(8) NOT NULL PRIMARY KEY,
    name VARCHAR(10) UNIQUE NOT NULL,
    birthYear INT NOT NULL,
    addr CHAR(2) NOT NULL,
    mobile1 CHAR(3),
    mobile2 CHAR(8),
    height SMALLINT,
    mDate DATE,
    UNIQUE INDEX idx_userTbl_name (name),
    INDEX idx_userTbl_addr (addr)
) DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

DROP TABLE IF EXISTS buyTbl;
CREATE TABLE buyTbl (
    num INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
    userID CHAR(8) NOT NULL,
```

```

    prodName CHAR(4),
    groupName CHAR(4),
    price INT NOT NULL,
    amount SMALLINT NOT NULL
) DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
INSERT INTO userTbl VALUES('LSG', '이승기', 1987, '서울', '011', '11111111',
182, '2008-8-8');
INSERT INTO buyTbl (userID, prodName, groupName, price, amount)
VALUES('KBS', '운동화', '의류', 30, 2);

```

- UNIQUE INDEX idx_uerTbl_name (name) : name 컬럼에 대해 idx_userTbl_name 이름으로 인덱스 생성, name 은 UNIQUE 제약조건 필요
- INDEX idx_userTbl_addr (addr) : addr 컬럼에 대해 idx_userTbl_addr 이름으로 인덱스 생성

7.4. 인덱스 삭제

- 기본 문법
 - ALTER TABLE 테이블명 DROP INDEX 인덱스명
- idx_userTbl_name 인덱스 삭제

```
ALTER TABLE userTbl DROP INDEX idx_userTbl_name
```

- idx_userTbl_addr 인덱스 삭제

```
ALTER TABLE userTbl DROP INDEX idx_userTbl_addr
```

연습문제

PRIMARY KEY 뺀다고 모든 인덱스 삭제하기

```
ALTER TABLE userTbl DROP INDEX name;
```

본 자료와 관련 영상 콘텐츠는 저작권법 제25조 2항에 의해 보호를 받습니다.

본 콘텐츠 및 콘텐츠 일부 문구 등을 외부에 공개하거나, 요약해서 게시하지 말아주세요.

Copyright [잔재미코딩](#) Dave Lee

