MySQL 활용

데이터의 조회 및 JOIN연산

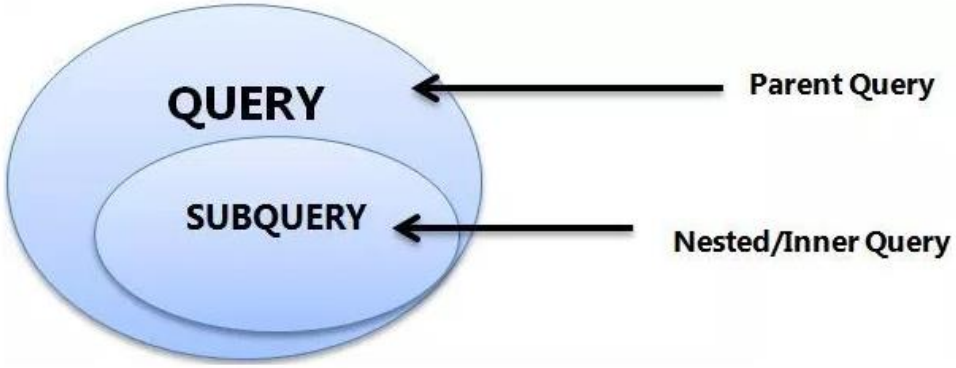
목차

- 서브쿼리
- 조회결과 정렬
- 중복제거
- 그룹핑
- VIEW

서브쿼리

- 서브쿼리
  - 쿼리 안에 있는 쿼리.

- 서브쿼리의 종류
  - Nested Subqueries : 중첩 서브쿼리. WHERE절 안에서 작성
  - Inline view : FROM 절 안에서 작성
  - Scalar Subqueries : SELECT 절 안에서 작성.



서브쿼리

- 서브쿼리 작성
  - 김경호보다 키가 크거나 같은 사람의 이름과 키를 조회

  - 해결방법

    1> 테이블에서 김경호의 키를 찾는다. → 177

    2> SELECT name, height FROM usertbl WHERE height >= 177

  - 즉 177 이라는 값을 얻어오는 쿼리를 부모쿼리에게 넣어줌.

    SELECT name, height FROM usertbl WHERE height >=

    (SELECT height FROM usertbl WHERE Name='김경호');

    177

서브쿼리

- 서브쿼리 작성
  - 지역이 '경남'인 사람의 키보다 키가 크거나 같은 회원 조회

SELECT * FROM usertbl WHERE height >= 경남사람의 키

  - 경남사람의 키

SELECT height FROM usertbl WHERE addr='경남';

height
173
170

  - 즉 이를 그대로 아래와 같이 서브쿼리로 만들면 에러발생 (서브쿼리의 결과가 복수)

SELECT * FROM usertbl WHERE height >=
(SELECT height FROM usertbl WHERE addr='경남'); → ERROR

서브쿼리

- 서브쿼리 작성
  - ANY를 통한 서브쿼리의 완성

```
SELECT * FROM usertbl WHERE height >=
                    ANY(SELECT height FROM usertbl WHERE addr= ' 경남 ' );
```

ANY : OR, >=170 또는 >=173 출력, 결과적으로 >=170 출력
ALL : AND, >= 170 이고 >=173 출력, 결과적으로 >=173 출력

  - ANY를 ALL로 바꾸어 조회해보고 그 결과를 분석해봅시다.

조회결과 정렬

- ORDER BY
  - 조회 결과를 정렬하는 구문, 조회 결과에는 영향을 주지 않음.
  - SELECT 문의 가장 뒤에 위치.

```
SELECT * FROM usertbl;
SELECT * FROM usertbl ORDER BY mDate; -- ASCENDING (오름차순)
SELECT * FROM usertbl ORDER BY mDate DESC; -- DESCENDING (내림차순)
```

  - 둘 이상의 조건 정렬

```
-- 키로 오름차순, 같은 키일때 name으로 오름차순 (단 ASC는 생략가능)
SELECT * FROM usertbl ORDER BY height, name ASC;
```

중복제거

- 중복제거
  - DISTINCT
  - 중복된 row를 제거하여 하나만 출력

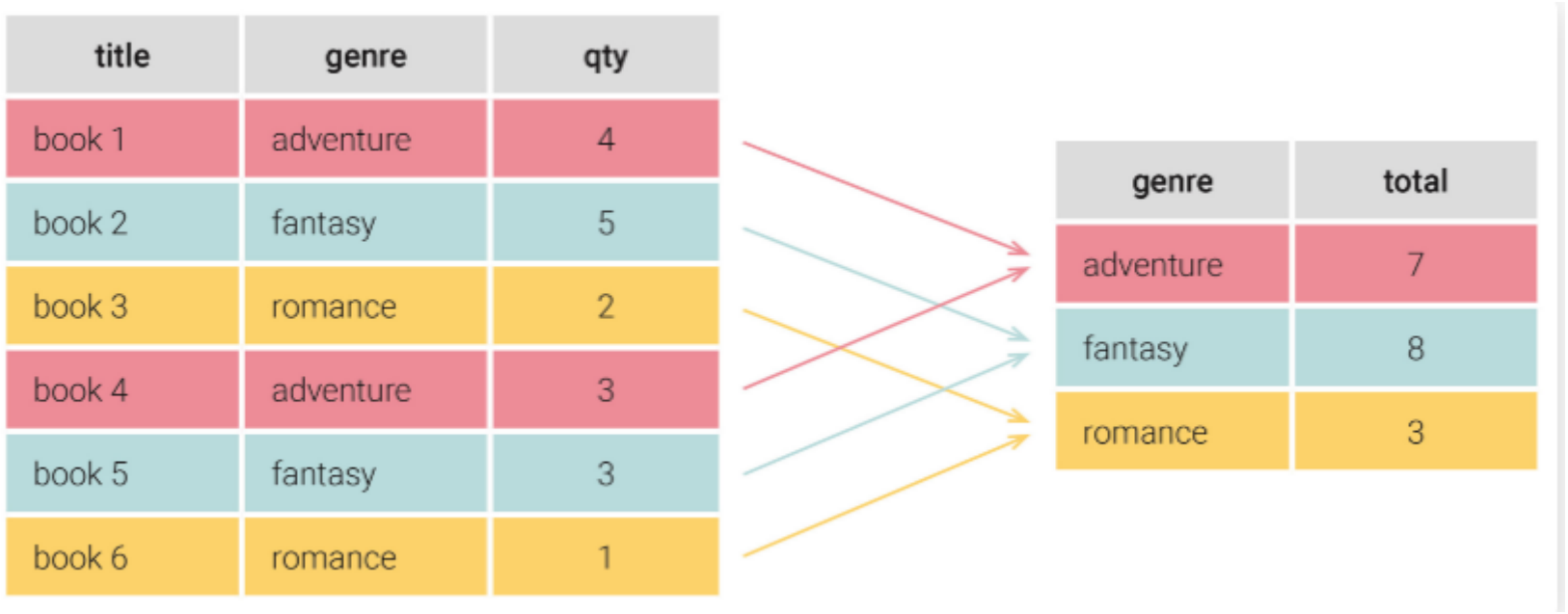
SELECT DISTINCT addr FROM usertbl;

  - 여러 개의 컬럼을 선택시 DISTINCT가 무효화됨.

SELECT DISTINCT addr, name FROM usertbl; → ERROR

그룹핑

- GROUP BY
  - 조회 결과를 특정 기준으로 그룹핑 수행
  - WHERE절 다음에 위치



title	genre	qty
book 1	adventure	4
book 2	fantasy	5
book 3	romance	2
book 4	adventure	3
book 5	fantasy	3
book 6	romance	1

genre	total
adventure	7
fantasy	8
romance	3

그룹핑

- GROUP BY
  - 구매테이블에서 사용자별 구매수량의 총합을 조회.

  SELECT userID, SUM(amount) FROM buytbl; → 문제발생!

  아래와 같이 userID별로 그룹핑하여 처리.

  SELECT userID, SUM(amount) FROM buytbl GROUP BY userID

- 집계함수 (집합함수)의 종류
  - AVG() : 평균
  - MIN() : 최소값
  - MAX() : 최대값
  - COUNT() : 행의 개수

그룹핑

- HAVING
  - 사용자별 총 구매액수를 조회

```
SELECT userID, SUM(price*amount)  FROM buytbl GROUP BY userID;
```

  - 이중 구매액수가 1000이상인 사용자를 조회.
    - WHERE??
    - WHERE는 집계함수의 결과에 적용 할 수 없음.
    - 따라서 HAVING절 필요.

```
SELECT userID, SUM(price*amount)  FROM buytbl GROUP BY userID
                                        HAVING SUM(price*amount) > 1000;
```

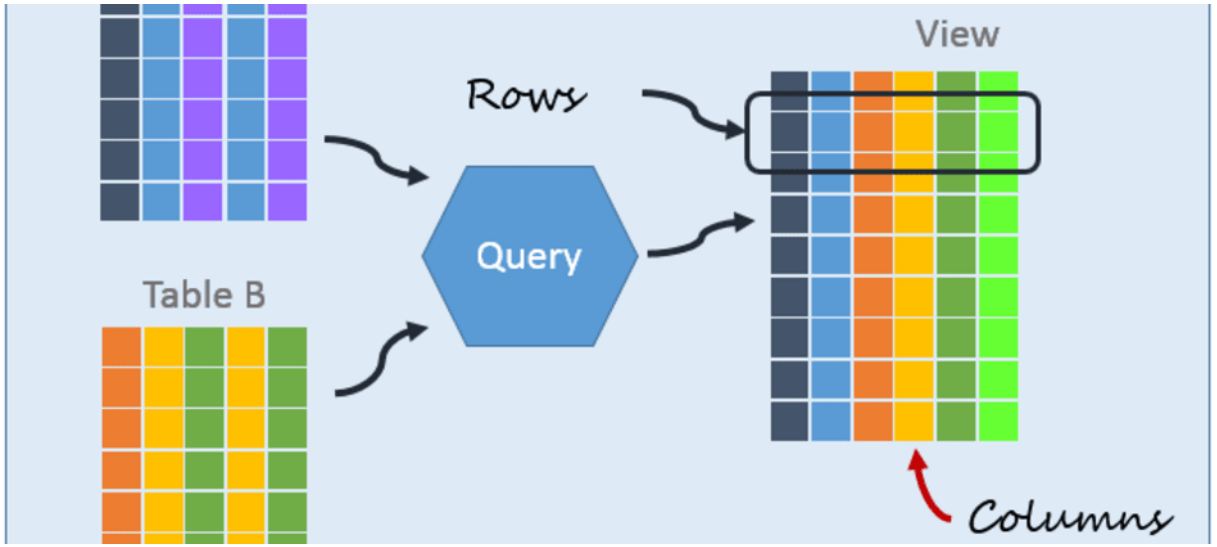
그룹핑

- GROUP BY
  - 회원별로 구매가 몇건이 일어났는지 조회하시오.
  - 회원별로 제품의 평균구매 개수를 조회하시오
  - 가장큰키와 가장 작은키의 회원이름과 키를 조회하시오
  - 휴대폰이 있는 사용자의 수를 조회하시오

VIEW

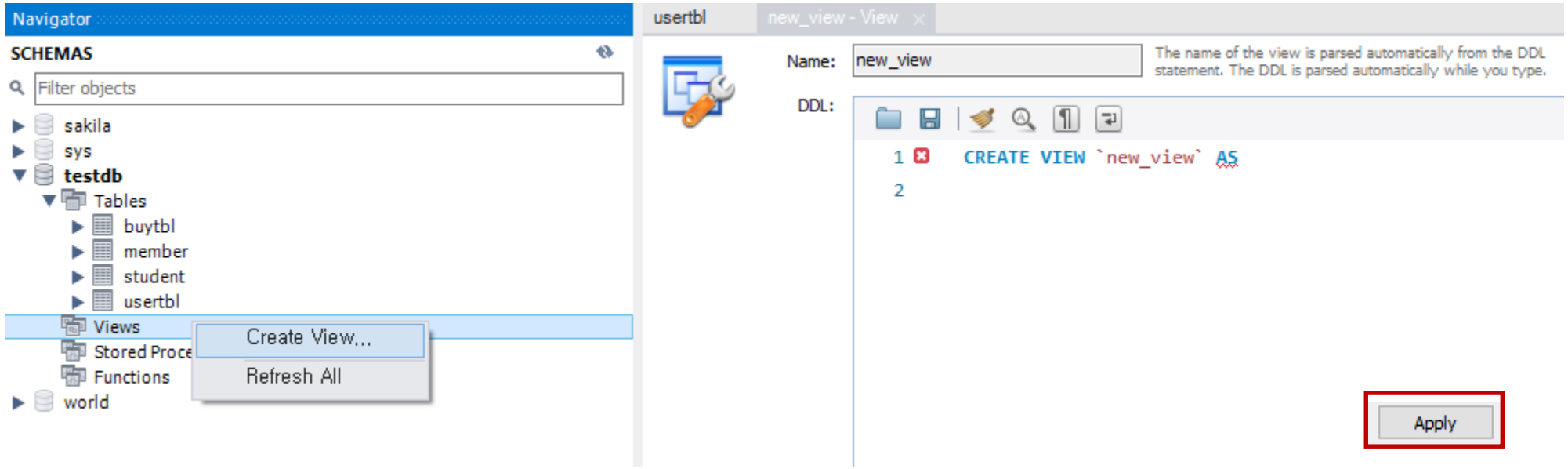
- VIEW의 정의
  - 하나 이상의 테이블에서 원하는데이터를 선택하여 만든 새로운 테이블
  - 사용자에게 접근이 허용된 자료만을 제한적으로 보여주는 가상 테이블

- View의 필요성
  - 논리적 데이터 독립성
  - 다양한 응용
  - 보안성



VIEW의 생성 및 활용

- VIEW의 생성



CREATE VIEW v_usertbl **AS** SELECT userID, name, addr FROM usertbl;