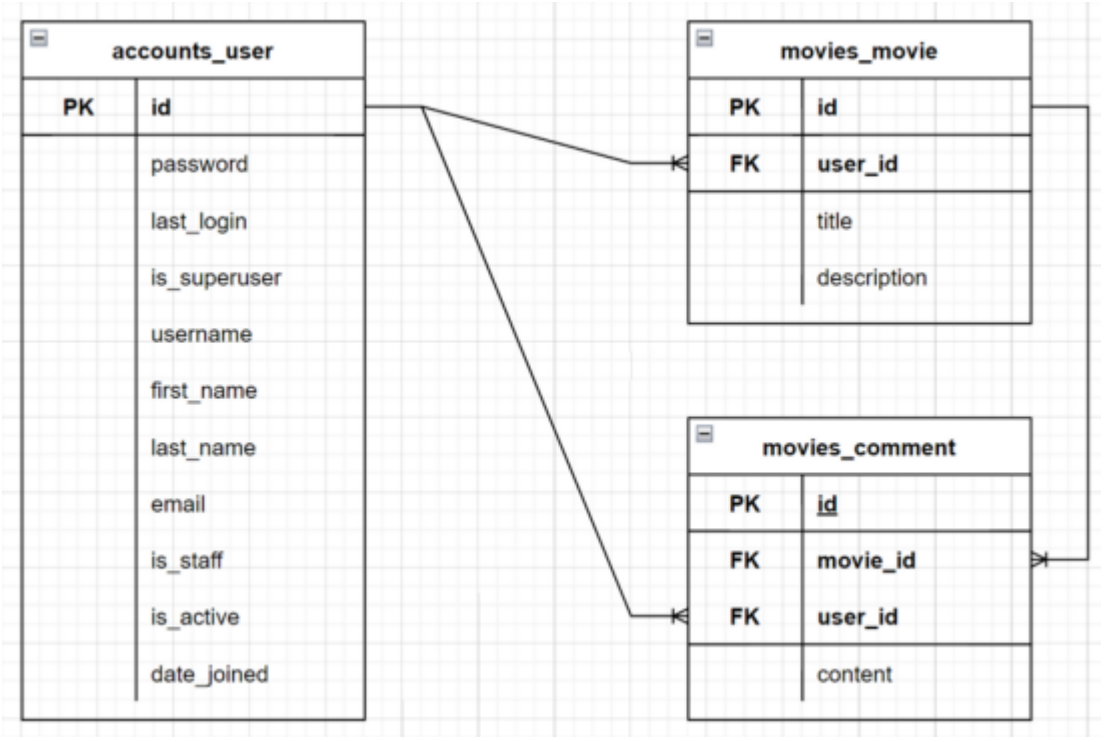
Join

구분	DB

💡 두 개 이상의 Table을 서로 묶어 하나의 테이블 만들기

실재로 한개의 DB에는 여러개의 Table이 존재 한다. 여러개의 Table에서 데이터의 중복을 최소화 하고 Table 간의 관계를 정의하기 좋게 바꾸는 것을 정규화 라고 한다.
이때 두 개 이상의 Table을 서로 묶어서 하나의 결과를 만들어 내는 것을 JOIN 이라고 한다.
두 테이블의 조인을 위해서는 테이블이 1:N (일대 다) 관계로 연결되어야 한다. 예를들면 하나의 영화 게시물(movies_movie) 에는 여러개의 댓글(movies_comment) 을 생성할 수 있다.



즉 영화관련 게시물(게시글) 을 관리하는 DB가 존재 할 것이고 (movies_movie)
사이트 내 댓글들을 관리하는 DB도 존재 할 것이다. (movies_comment)
위 표를 보면 accounts_user에서의 하나의 “유저ID” 는 여러 개의 게시글을(movie) 작성할 수 있고 여러 개의 댓글(comment)도 작성할 수 있다. 그리고
하나의 영화 게시물(movie)에는 여러 개의 댓글(comment)이 달릴 수 있다. 그래서 1:N (일

대 다) 관계가 되는것이다
1:N 관계는 여기까지만 보고 일반적으로 JOIN이라고 부르는 내부조인 (INNER JOIN)을 살펴보자.
내부 조인의 형태는 다음과 같다.

INNER JOIN

- 두 테이블에서 일치하는 데이터만 결과 출력

```
SELECT <열목록>
FROM <첫 번째 테이블>
    INNER JOIN <두 번째 테이블>
    ON <조인될 조건>
```

INNER JOIN 실습

1. JOIN 할 테이블 만들기 - `articles`, `users`

articles 테이블 그리고 users 테이블을 만들어 보자.

```
-- articles 테이블 만들기
CREATE TABLE articles (
    title TEXT NOT NULL,
    content TEXT NOT NULL,
    userid INTEGER NOT NULL
);

INSERT INTO articles
VALUES
    ('대부', '대부내용이다', 1),
    ('행복', '행복내용이다', 2),
    ('극한직업', '극한직업내용이다', 3),
    ('타이타닉', '디카프리오', 7),
    ('엄복동', '자전차왕의 일대기', 2);

-- users 테이블 만들기
CREATE TABLE users (
    name TEXT NOT NULL,
    roleid TEXT NOT NULL
);

INSERT INTO users
VALUES
    ('kevin', 1),
    ('aiden', 3),
    ('jorny', 3),
```

```
    ('bob', 2),
    ('kate', 1);
```

title	content	userid
대부	대부내용이다	1
행복	행복내용이다	2
극한직업	극한직업내용이다	3
타이타닉	디카프리오	7
엄복동	자전차왕의 일대기	2

name	roleid
kevin	1
aiden	3
jorny	3
bob	2
kate	1

articles 라는 테이블 그리고 users 라는 테이블을 생성하였다.
articles 테이블에서 userid 컬럼은 article을 작성한 사람의 아이디를 말하며
users 테이블에서 roleid 컬럼은 유저들의 역할을 의미한다.
편의상 아래 그림과 같이 roles 라는 테이블은 따로 생성은 하지 않았지만 roleid 가 1 이면 관리자 / 2 라면 스탭 / 3 이라면 학생을 의미한다고 생각하자.

roles

id	role
1	admin
2	staff
3	student

2. userid를 기준으로 INNER JOIN 하기
    - articles.userid와 users.rowid를 연결할 수 없는 항목은 누락됨
        - 누락되지 않으려면? LEFT JOIN 사용

```
SELECT * FROM articles INNER JOIN users ON userid=users.rowid;

-- 아래 방법도 가능하지만 권장하지는 않음
```

```
SELECT * FROM articles, users
WHERE articles.userid=users.rowid;
```

title	content	userid	name	roleid
대부	대부내용이다	1	kevin	1
행복	행복내용이다	2	aiden	3
극한직업	극한직업내용이다	3	jorny	3
엄복동	자전차왕의 일대기	2	aiden	3

현재 INNER JOIN을 userid 기준으로 JOIN을 했다.
JOIN 후의 모습을 보면(바로 위에 있는 테이블을 보면)
타이타닉에 관한 내용은 누락되었다.

타이타닉을 작성한 사람의 userid 는 7 이지만
users DB에는 userid 가 7인 사람은 존재하지 않는다.

(뭐.. 아마 userid 가 7번인 사람은 회원가입을 한 후 타이타닉 article을 하나 작성한 다음에 회원탈퇴를 했었을 경우라고 가정하자)

그래서 현재 users DB에는 존재하지 않는 userid 가 7이였던 사람이 작성한 타이타닉은 두 테이블이 합쳐지며 누락이 되었다.

위에 작성했던 Inner Join (내부 조인) 의 형식은 다음과 같다

```
SELECT <열목록>
FROM <첫 번째 테이블>
  INNER JOIN <두 번째 테이블>
  ON <조인될 조건>
```

만약에 '타이타닉'과 같이 article은 누락되지 않으면서 articles 데이터와 users 데이터를 합치고 싶다면? Left join 을 사용해 보자

LEFT (OUTER) JOIN

- 왼쪽에 있는 데이터 articles 로 오른쪽에 있는 데이터 users 를 join 해보자
- 왼쪽 테이블의 데이터를 기준으로 오른쪽 데이터 결합한다는 말이다.
- 일치하는 항목 없을 경우 해당 항목 NULL 값으로 채워진다.

```
SELECT <열목록>
FROM <첫 번째 테이블(LEFT 테이블)>
    LEFT JOIN <두 번째 테이블(RIGHT 테이블)>
    ON <조인될 조건>
[WHERE 검색 조건 추가];
```

LEFT JOIN 실습

- 누락되는 데이터 없이 join 완성 가능
    - JOIN 불가능할경우 해당 항목 NULL 값으로 채움

```
SELECT * FROM articles LEFT JOIN users ON userid=users.rowid;
```

title	content	userid	name	roleid
대부	대부내용이다	1	kevin	1
행복	행복내용이다	2	aiden	3
극한직업	극한직업내용이다	3	jorny	3
타이타닉	디카프리오	7	NULL	NULL
엄복동	자전차왕의 일대기	2	aiden	3

articles 에서 누락되는 데이터 없이 join이 완성 되었으며
해당 name과 roleid 는 NULL 값으로 채워 졌다.
형식을 살펴보면 다음과 같다.

```
SELECT <열목록>
FROM <첫 번째 테이블(LEFT 테이블)>
  LEFT <두 번쨰 테이블(RIGHT 테이블)>
  ON <조인될 조건>
[WHERE 검색 조건 추가];
```

만약에 LEFT 대신 RIGHT 라고 적었다면 오른쪽 테이블 기준으로 왼쪽 테이블의 데이터가 결합이 될 것이며 역시 없는 값은 NULL 값으로 채워질 것이다.
하지만 sqlite 에서는 right join을 지원하지 않는다 ! 그래서 실습을 하지 않는다.

RIGHT JOIN

- sqlite에서는 지원하지 않음

```
SELECT * FROM articles RIGHT JOIN users ON userId=users.rowId;
```

CROSS JOIN

2개의 테이블을 조인하여 데이터를 검색하는 방법 중에
두 테이블의 데이터의 모든 조합을 받아오는 방법이 Cross join 이다

- 모든 조합 출력

```
CREATE TABLE products (
  name TEXT NOT NULL,
  p_id INTEGER NOT NULL
);

INSERT INTO products
VALUES
('책상', 1),
('의자', 2),
('컴퓨터', 3);

CREATE TABLE colors (
  name TEXT NOT NULL,
  c_id INTEGER NOT NULL
);
INSERT INTO colors
VALUES
('red', 1),
('blue', 2),
('white', 3);
```

name	p_id
책상	1
의자	2
컴퓨터	3

name	c_id
red	1
blue	2
white	3

왼쪽 테이블 데이터 1개당 오른쪽 테이블 데이터를 처음부터 끝까지 하나씩 결합한다. 결합 후 모습은 다음과 같다.

name	p_id	name	c_id
책상	1	red	1
책상	1	blue	2
책상	1	white	3
의자	2	red	1
의자	2	blue	2
의자	2	white	3
컴퓨터	3	red	1
컴퓨터	3	blue	2
컴퓨터	3	white	3

```
SELECT * FROM products CROSS JOIN colors;
```

<끝>

지금껏 배운것은

SELECT 문에서의 FORM, ORDER BY (ASC DESC), WHERE, LIMIT
SUM, MAX, MIN, COUNT, DISTINCT, AS(별칭선언) 그리고

GROUP BY 에서의 HAVING SET 그리고

JOIN 에서의 INNER JOIN, LETT JOIN 그 외에도 RIGHT JOIN, OUTER JOIN 도 있지만
sqlite에서는 “RIGHT JOIN”, “OUTER JOIN”을 지원하지 않다고 했다.

싸피에서 따로 다루지 않지만 프로그래머스 SQL 고득점KIT 문제들을
다른 사람들은 어떤 식으로 풀었는지 잘 보면서 공부를 했으면 좋겠습니다.

최소 코딩테스트를 보기 전에는 프로그래머스 문제들을 여러 번 (2~3번) 반복해서 연습해 보기를 바랍니다.

(여력이 된다면 알고리즘 스터디 하면서 한번 정도는 함께 공부하는 것도 나쁘지 않을 듯 싶습니다.)