Hello,

KDT 웹 개발자 양성 프로젝트

5기!

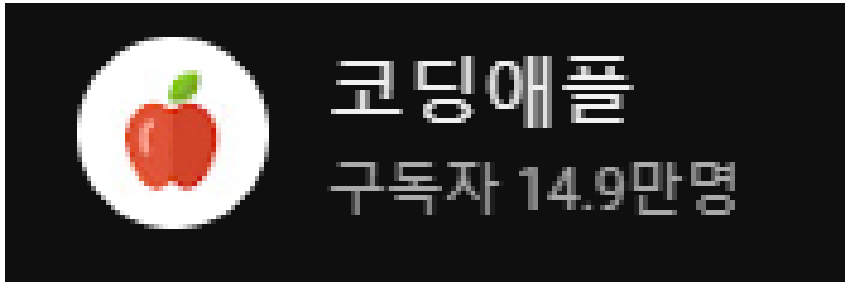
with

정규화

그래서 그거 왜 하나요?

정규화?

- DB 설계에 있어서 중복을 최소화 하기 위해 데이터를 구조화 하는 과정
- 크고 조직화 되지 않은 테이블 → 작고, 잘 조직된 테이블로 변경
- 이렇게 작성을 해야만, 데이터 추가 및 삭제 시에 이상 현상(Abnormal) 을 예방 할 수 있습니다!



https://www.youtube.com/watch?v=Y1FbowQRcmI

제 1 정규형

(First Normal Form)

제 1 정규형(1NF)

- 하나의 컬럼은 반드시 하나의 값(=필드)만을 가져야 하는 법칙

회원번호	회원이름	프로그램
101	강호동	스쿼시초급
102	손흥민	헬스
103	김민수	헬스, 골프초급

회원번호	회원이름	프로그램
101	강호동	스쿼시초급
102	손흥민	헬스
103	김민수	헬스
103	김민수	골프초급

회원번호	회원이름	프로그램
101	강호동	스쿼시초급
102	손흥민	헬스
103	김민수	헬스, 골프초급

테이블의 필드가 1개 이상의
값을 가지고 있습니다!

이런 상황에서 김민수가 헬스를
수강 중인지 여부를 알고 싶다면!?

해당 데이터를 불러왔을 때 바로 알 수 있나요!?

회원번호	회원이름	프로그램
101	강호동	스쿼시초급
102	손흥민	헬스
103	김민수	헬스, 골프초급

이런 상황에서 헬스만 삭제하고 싶다면!?
그리고 프로그램을 추가하고 싶다면!?

뭔가 작업을 추가로 해야합니다!

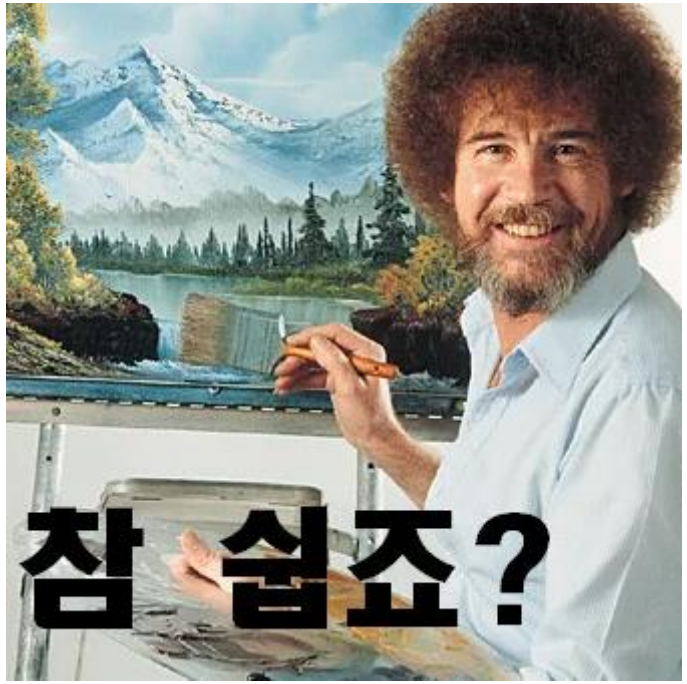
회원번호	회원이름	프로그램
101	강호동	스쿼시초급
102	손흥민	헬스
103	김민수	헬스, 골프초급

회원번호	회원이름	프로그램
101	강호동	스쿼시초급
102	손흥민	헬스
103	김민수	헬스
103	김민수	골프초급

회원번호	회원이름	프로그램
101	강호동	스쿼시초급
102	손흥민	헬스
103	김민수	헬스
103	김민수	골프초급

이런 상황에서 김민수가 헬스를
수강 중인지 여부를 알고 싶다면!?

김민수가 듣는 헬스만 삭제하고 싶다면!?
김민수가 다른 프로그램을 수강한다면!?



제 2 정규형

(Second Normal Form)

제 2 정규형(2NF)

- 모든 컬럼에 대한 부분 종속이 없어야 한다

→ 현 테이블의 목적과 필요 없는 친구는 빼준다!

회원번호	회원이름	프로그램	가격	납부여부
101	강호동	스쿼시초급	5000	0
102	손흥민	헬스	6000	1
103	김민수	헬스	6000	1
103	김민수	골프초급	8000	0

수강등록현황 table

회원번호	회원이름	프로그램	납부여부
101	강호동	스쿼시초급	0
102	손흥민	헬스	1
103	김민수	헬스	1
103	김민수	골프초급	0

프로그램 table

프로그램	가격
스쿼시초급	5000
헬스	6000
골프초급	8000

회원번호	회원이름	프로그램	가격	납부여부
101	강호동	스쿼시초급	5000	0
102	손흥민	헬스	6000	1
103	김민수	헬스	6000	1
103	김민수	골프초급	8000	0

여러분의 데이터가
이게 전부라고 해봅시다!

이 상황에서 헬스 프로그램의
가격이 7000이 된다면!?

- 왜 자꾸 장사가 잘되는데!!?

회원번호	회원이름	프로그램	가격	납부여부
101	강호동	스쿼시초급	5000	0
102	손흥민	헬스	6000	1
103	김민수	헬스	6000	1
103	김민수	골프초급	8000	0
102	손흥민	헬스	6000	1
102	손흥민	헬스	6000	1
102	손흥민	헬스	6000	1
102	손흥민	헬스	6000	1
102	손흥민	헬스	6000	1
102	손흥민	헬스	6000	1

헬스를 듣는 회원이
1180000000명 이라면!?

데이터를 몇 번 수정해야 하죠!?

제2 정규형 부분 종속성 제거를 해봅시다!
(=테이블과 필요 없는 데이터는 따로 뺀다)

회원번호	회원이름	프로그램	가격	납부여부
101	강호동	스쿼시초급	5000	0
102	손흥민	헬스	6000	1
103	김민수	헬스	6000	1
103	김민수	골프초급	8000	0

회원번호	회원이름	프로그램	납부여부
101	강호동	스쿼시초급	0
102	손흥민	헬스	1
103	김민수	헬스	1
103	김민수	골프초급	0

프로그램	가격
스쿼시초급	5000
헬스	6000
골프초급	8000

하지만 부분 종속성을 제거해서 테이블이 아래와 같이 나누어져 있다면!?

회원번호	회원이름	프로그램	납부여부
102	손흥민	헬스	1
103	김민수	헬스	1
103	김민수	골프초급	0
102	손흥민	헬스	1
102	손흥민	헬스	1
102	손흥민	헬스	1
102	손흥민	헬스	1
102	손흥민	헬스	1
102	손흥민	헬스	1
102	손흥민	헬스	1
102	손흥민	헬스	1

프로그램	가격
스쿼시초급	5000
헬스	7000
골프초급	8000

회원번호	회원이름	프로그램	가격	납부여부
101	강호동	스쿼시초급	5000	0
102	손흥민	헬스	6000	1
103	김민수	헬스	6000	1
103	김민수	골프초급	8000	0

다시 이전으로 돌아옵시다!

이런 상황에서
강호동씨가 탈퇴해서 해당 데이터를
삭제했다고 해봅시다!

회원번호	회원이름	프로그램	가격	납부여부
102	손흥민	헬스	6000	1
103	김민수	헬스	6000	1
103	김민수	골프초급	8000	0

이제 데이터에서
스쿼시 프로그램의
가격을 알 수 있나요!?

제2 정규형 부분 종속성 제거를 해봅시다!
(=테이블과 필요 없는 데이터는 따로 뺀다)

회원번호	회원이름	프로그램	가격	납부여부
101	강호동	스쿼시초급	5000	0
102	손흥민	헬스	6000	1
103	김민수	헬스	6000	1
103	김민수	골프초급	8000	0

회원번호	회원이름	프로그램	납부여부
101	강호동	스쿼시초급	0
102	손흥민	헬스	1
103	김민수	헬스	1
103	김민수	골프초급	0

프로그램	가격
스쿼시초급	5000
헬스	6000
골프초급	8000

만약 강호동씨가 탈퇴를 해도!?

스쿼시 초급의 가격을 알 수 있나요!?

회원번호	회원이름	프로그램	납부여부
102	손흥민	헬스	1
103	김민수	헬스	1
103	김민수	골프초급	0

프로그램	가격
스쿼시초급	5000
헬스	6000
골프초급	8000

제 3 정규형

(Second Normal Form)

제 3 정규형(3NF)

- 주요 컬럼(Primary Key)이 아닌 일반 컬럼에만 종속 된 컬럼은 다른 테이블로 빼야 한다!

프로그램	가격	강사	출신대학
스쿼시	5000	김을용	서울대
헬스	6000	박덕팔	연세대
골프	8000	이상구	고려대
골프중급	9000	이상구	고려대
개인피티	6000	박덕팔	연세대

프로그램	가격	강사
스쿼시	5000	김을용
헬스	6000	박덕팔
골프	8000	이상구
골프중급	9000	이상구
개인피티	6000	박덕팔

강사	출신대학
김을용	서울대
박덕팔	연세대
이상구	고려대

프로그램	가격	강사	출신대학
스쿼시	5000	김을용	서울대
헬스	6000	박덕팔	연세대
골프	8000	이상구	고려대
골프중급	9000	이상구	고려대
개인피티	6000	박덕팔	연세대

이상구 강사가 너무 열일해서
골프고급, 골프초고급,
골프프로준비반, 골프에미친사람모임
등등을 운영한다 가정해 봅시다!

프로그램	가격	강사	출신대학
스쿼시	5000	김을용	서울대
헬스	6000	박덕팔	연세대
골프	8000	이상구	고려대
골프중급	9000	이상구	고려대
개인피티	6000	박덕팔	연세대
골프중급	9000	이상구	고려대
골프중급	9000	이상구	고려대
골프중급	9000	이상구	고려대
골프중급	9000	이상구	고려대
골프중급	9000	이상구	고려대

자 그런데 이상구 강사가
학력을 위조한 것이 밝혀졌습니다!

이상구 강사는 사실
하버드를 졸업했다고 한다면!?

데이터를 몇 번 수정해야 하나요!?

제3 정규형을 만족하도록 테이블을 나눠 봅시다!
(=일반 컬럼에 종속적인 부분도 뺀다)

프로그램	가격	강사	출신대학
스쿼시	5000	김을용	서울대
헬스	6000	박덕팔	연세대
골프	8000	이상구	고려대
골프중급	9000	이상구	고려대
개인피티	6000	박덕팔	연세대

프로그램	가격	강사
스쿼시	5000	김을용
헬스	6000	박덕팔
골프	8000	이상구
골프중급	9000	이상구
개인피티	6000	박덕팔

강사	출신대학
김을용	서울대
박덕팔	연세대
이상구	고려대

이젠 어떻게 하면 되죠!?

프로그램	가격	강사
스쿼시	5000	김을용
헬스	6000	박덕팔
골프	8000	이상구
골프중급	9000	이상구
개인피티	6000	박덕팔

강사	출신대학
김을용	서울대
박덕팔	연세대
이상구	하바드

정규화와 JOIN

직접 확인!

인생은 실전이야

정규화가 안된

테이블 만들기!

```sql
-- 데이터베이스 생성
CREATE DATABASE mydb;

-- 데이터베이스 사용
USE mydb;
```

데이터베이스(=스키마) 생성 후
해당 데이터베이스 사용 선언

```sql
-- 통합 테이블 생성
CREATE TABLE total_table (
    order_id INT,
    customer_id INT,
    customer_name VARCHAR(100),
    order_date DATE,
    product_id INT,
    product_name VARCHAR(100),
    product_price INT,
    category_id INT,
    category_name VARCHAR(100),
    quantity INT
);
```

정규화가 안된
물품 주문 목록 테이블 생성

```sql
-- 통합 테이블에 데이터 삽입
INSERT INTO total_taBLE (order_id, customer_id, customer_name, order_DATE, PRODuct_id, produCt_name,
(1, 1, '김시완', '2024-06-01', 1, '맥북에어', 650000, 1, '전자기기', 1),
(1, 1, '김시완', '2024-06-01', 4, '맥북 파우치', 10000, 2, '주변기기', 1),
(2, 2, '이효석', '2024-06-15', 2, '맥북프로', 700000, 1, '전자기기', 1),
(3, 3, '김연비', '2024-05-15', 3, '맥북프로맥스', 750000, 1, '전자기기', 1),
(4, 4, '최호진', '2024-05-20', 5, 'DT-35키보드', 20000, 2, '주변기기', 1),
(5, 5, '최현수', '2024-05-25', 6, '독거미키보드', 50000, 2, '주변기기', 1),
(6, 6, '김동준', '2024-05-25', 6, '독거미키보드', 50000, 2, '주변기기', 1),
(7, 7, '권오현', '2024-05-25', 6, '독거미키보드', 50000, 2, '주변기기', 1);
```

필요 데이터 삽입!

테이블 및 삽입된 데이터 확인!

```
35 •  SELECT * FROM total_table;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_id	customer_name	order_date	product_id	product_name	product_price	category_id	category_name	quantity
1	1	김시완	2024-06-01	1	맥북에어	650000	1	전자기기	1
1	1	김시완	2024-06-01	4	맥북 파우치	10000	2	주변기기	1
2	2	이효석	2024-06-15	2	맥북프로	700000	1	전자기기	1
3	3	김연비	2024-05-15	3	맥북프로맥스	750000	1	전자기기	1
4	4	최호진	2024-05-20	5	DT-35키보드	20000	2	주변기기	1
5	5	최현수	2024-05-25	6	독거미키보드	50000	2	주변기기	1
6	6	김동준	2024-05-25	6	독거미키보드	50000	2	주변기기	1
7	7	권오현	2024-05-25	6	독거미키보드	50000	2	주변기기	1

나이스

그란데말입니다
그것이 알고싶다

이 테이블은 제1 정규형을 만족하나요!?

```
35 •  SELECT * FROM total_table;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_id	customer_name	order_date	product_id	product_name	product_price	category_id	category_name	quantity
1	1	김시완	2024-06-01	1	맥북에어	650000	1	전자기기	1
1	1	김시완	2024-06-01	4	맥북 파우치	10000	2	주변기기	1
2	2	이효석	2024-06-15	2	맥북프로	700000	1	전자기기	1
3	3	김연비	2024-05-15	3	맥북프로맥스	750000	1	전자기기	1
4	4	최호진	2024-05-20	5	DT-35키보드	20000	2	주변기기	1
5	5	최현수	2024-05-25	6	독거미키보드	50000	2	주변기기	1
6	6	김동준	2024-05-25	6	독거미키보드	50000	2	주변기기	1
7	7	권오현	2024-05-25	6	독거미키보드	50000	2	주변기기	1

정규화 작업이 필요한 이유!

독거미 키보드의 product_id 나 product_price, category_id, category_name 이 변경 된다면 작업을 몇 번 해줘야 하나요!?

```
35 •  SELECT * FROM total_table;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_id	customer_name	order_date	product_id	product_name	product_price	category_id	category_name	quantity
1	1	김시완	2024-06-01	1	맥북에어	650000	1	전자기기	1
1	1	김시완	2024-06-01	4	맥북 파우치	10000	2	주변기기	1
2	2	이효석	2024-06-15	2	맥북프로	700000	1	전자기기	1
3	3	김연비	2024-05-15	3	맥북프로맥스	750000	1	전자기기	1
4	4	최호진	2024-05-20	5	DT-35키보드	20000	2	주변기기	1
5	5	최현수	2024-05-25	6	독거미키보드	50000	2	주변기기	1
6	6	김동준	2024-05-25	6	독거미키보드	50000	2	주변기기	1
7	7	권오현	2024-05-25	6	독거미키보드	50000	2	주변기기	1

만약 이효석 Row 가 삭제 된다면!?
맥북 프로의 가격을 나중에 알 수 있나요!?

```
35 •  SELECT * FROM total_table;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_id	customer_name	order_date	product_id	product_name	product_price	category_id	category_name	quantity
1	1	김시완	2024-06-01	1	맥북에어	650000	1	전자기기	1
1	1	김시완	2024-06-01	4	맥북 파우치	10000	2	주변기기	1
3	3	김연비	2024-05-15	3	맥북프로맥스	750000	1	전자기기	1
4	4	최호진	2024-05-20	5	DT-35키보드	20000	2	주변기기	1
5	5	최현수	2024-05-25	6	독거미키보드	50000	2	주변기기	1
6	6	김동준	2024-05-25	6	독거미키보드	50000	2	주변기기	1
7	7	권오현	2024-05-25	6	독거미키보드	50000	2	주변기기	1

잠깐
NEW EPISODE
CN HD

어머!
이건 알아야돼

Foreign Key

(외래 키)

외래 키(Foreign Key)

- 정규화를 하게 되면 테이블은 최소한의 단위로 쪼개지게 됩니다!
- 쪼개진 테이블은 JOIN 을 사용하여 테이블을 합치게 되는데요. 보통 기준이 되는 값을 통해서 테이블을 합쳐 주게 됩니다! → 이 때 기준이 되는 값(서로 공유하고 있는 값)을 외래 키라고 합니다!

기본키(Primary Key)

외래키(Foreign Key)

기본키(Primary Key)

선수번호	이름	팀 코드	포지션	등번호	키
1	김남일	K03	DF	33	177
2	박지성	K07	MF	7	178
3	이영표	K02	MF	22	176

〈선수 테이블〉

팀 코드	팀 명	연고지
K03	스틸러스	포항
K07	드래곤즈	전남
K02	블루윙즈	수원

〈구단 테이블〉

선수번호	이름	팀 코드	포지션	등번호	키	팀 명	연고지
1	김남일	K03	DF	33	177	스틸러스	포항
2	박지성	K07	MF	7	178	드래곤즈	전남
3	이영표	K02	MF	22	176	블루윙즈	수원

잠깐
NEW EPISODE
CN HD

아래의 테이블은 1, 2 정규화를 만족하기 위해서는
어떻게 쪼개야 할지 고민해 봅시다!!

```
35 •  SELECT * FROM total_table;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_id	customer_name	order_date	product_id	product_name	product_price	category_id	category_name	quantity
1	1	김시완	2024-06-01	1	맥북에어	650000	1	전자기기	1
1	1	김시완	2024-06-01	4	맥북 파우치	10000	2	주변기기	1
2	2	이효석	2024-06-15	2	맥북프로	700000	1	전자기기	1
3	3	김연비	2024-05-15	3	맥북프로맥스	750000	1	전자기기	1
4	4	최호진	2024-05-20	5	DT-35키보드	20000	2	주변기기	1
5	5	최현수	2024-05-25	6	독거미키보드	50000	2	주변기기	1
6	6	김동준	2024-05-25	6	독거미키보드	50000	2	주변기기	1
7	7	권오현	2024-05-25	6	독거미키보드	50000	2	주변기기	1

미리보기 방지

미리보기 방지

미리보기 방지

미리보기 방지

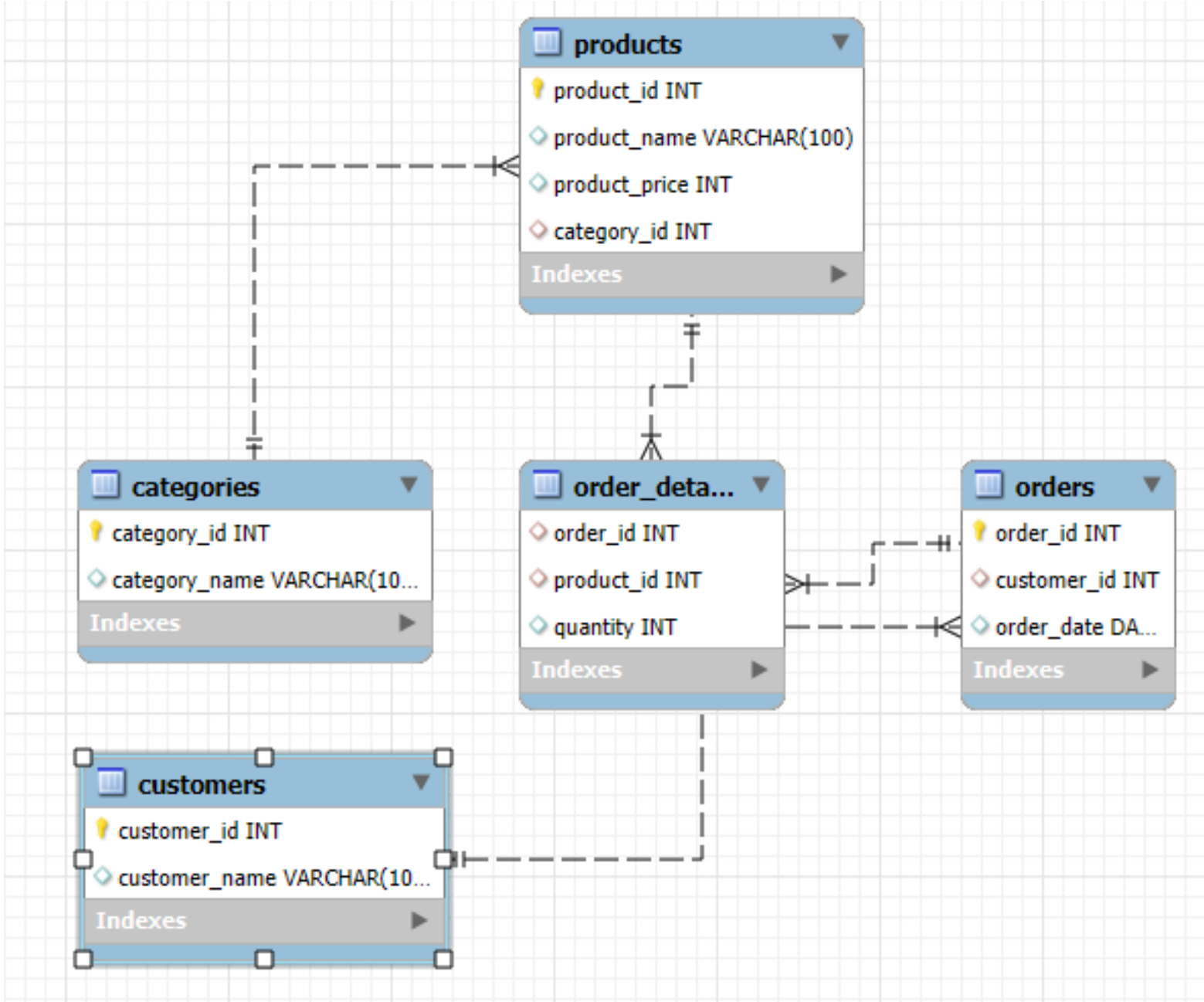
미리보기 방지

미리보기 방지

미리보기 방지

미리보기 방지

테이블 구조


products
product_id INT
product_name VARCHAR(100)
product_price INT
category_id INT
Indexes
categories
category_id INT
category_name VARCHAR(10...
Indexes
order_deta...
order_id INT
product_id INT
quantity INT
Indexes
orders
order_id INT
customer_id INT
order_date DA...
Indexes
customers
customer_id INT
customer_name VARCHAR(10...
Indexes

customers TABLE

customer_id	customer_name
1	김시완
2	이효석
3	김연비
4	최호진
5	김동준
6	최현수
7	권오현
NULL	NULL

products TABLE

Result Grid | Filter Rows: | Edit:

product_id	product_name	product_price	category_id
1	맥북에어	650000	1
2	맥북프로	700000	1
3	맥북프로맥스	750000	1
4	맥북파우치	10000	2
5	DT-35키보드	20000	2
6	독거미키보드	50000	2
NULL	NULL	NULL	NULL

categories TABLE

category_id	category_name
1	전자기기
2	주변기기
NULL	NULL

orders TABLE

Result Grid | Filter Rows:

order_id	customer_id	order_date
1	1	2024-06-01
2	2	2023-06-02
3	3	2023-06-03
4	4	2023-06-04
5	5	2023-06-05
6	6	2023-06-06
7	7	2023-06-07
NULL	NULL	NULL

order_details TABLE

Result Grid | Filter Rows:

order_id	product_id	quantity
1	1	1
1	4	1
2	2	1
3	3	1
4	5	1
5	6	1
6	6	1
7	6	1

테이블 생성

```sql
-- 고객 테이블 생성
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(100)
);

-- 주문 테이블 생성
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

-- 카테고리 테이블 생성
CREATE TABLE categories (
    category_id INT PRIMARY KEY,
    category_name VARCHAR(100)
);
```

```sql
-- 상품 테이블 생성
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100),
    product_price INT,
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES categories(category_id)
);

-- 주문 상세 테이블 생성
CREATE TABLE order_details (
    order_id INT,
    product_id INT,
    quantity INT,
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

데이터 삽입

```sql
-- 고객 테이블에 데이터 삽입
INSERT INTO customers (customer_id, customer_name) VALUES
(1, '김시완'),
(2, '이효석'),
(3, '김연비'),
(4, '최호진'),
(5, '김동준'),
(6, '최현수'),
(7, '권오현');

-- 카테고리 테이블에 데이터 삽입
INSERT INTO categories (category_id, category_name) VALUES
(1, '전자기기'),
(2, '주변기기');


-- 상품 테이블에 데이터 삽입
INSERT INTO products (product_id, product_name, product_price, category_id) VALUES
(1, '맥북에어', 650000, 1),
(2, '맥북프로', 700000, 1),
(3, '맥북프로맥스', 750000, 1),
(4, '맥북파우치', 10000, 2),
(5, 'DT-35키보드', 20000, 2),
(6, '독거미키보드', 50000, 2);
```

```sql
-- 주문 테이블에 데이터 삽입
INSERT INTO orders (order_id, customer_id, order_date) VALUES
(1, 1, '2024-06-01'),
(2, 2, '2023-06-02'),
(3, 3, '2023-06-03'),
(4, 4, '2023-06-04'),
(5, 5, '2023-06-05'),
(6, 6, '2023-06-06'),
(7, 7, '2023-06-07');

-- 주문 상세 테이블에 데이터 삽입
INSERT INTO order_details (order_id, product_id, quantity) VALUES
(1, 1, 1),
(1, 4, 1),
(2, 2, 1),
(3, 3, 1),
(4, 5, 1),
(5, 6, 1),
(6, 6, 1),
(7, 6, 1);
```

완성된 테이블

customers TABLE

customer_id	customer_name
1	김시완
2	이효석
3	김연비
4	최호진
5	김동준
6	최현수
7	권오현
NULL	NULL

products TABLE

Result Grid | Filter Rows: | Edit:

product_id	product_name	product_price	category_id
1	맥북에어	650000	1
2	맥북프로	700000	1
3	맥북프로맥스	750000	1
4	맥북파우치	10000	2
5	DT-35키보드	20000	2
6	독거미키보드	50000	2
NULL	NULL	NULL	NULL

categories TABLE

category_id	category_name
1	전자기기
2	주변기기
NULL	NULL

orders TABLE

Result Grid | Filter Rows:

order_id	customer_id	order_date
1	1	2024-06-01
2	2	2023-06-02
3	3	2023-06-03
4	4	2023-06-04
5	5	2023-06-05
6	6	2023-06-06
7	7	2023-06-07
NULL	NULL	NULL

order_details TABLE

Result Grid | Filter Rows:

order_id	product_id	quantity
1	1	1
1	4	1
2	2	1
3	3	1
4	5	1
5	6	1
6	6	1
7	6	1

JOIN

자~ 이제 시작이야(내꿈을)

테이블은 다 쪼개져 있으니 합쳐 봅시다!

customers TABLE

customer_id	customer_name
1	김시완
2	이효석
3	김연비
4	최호진
5	김동준
6	최현수
7	권오현
NULL	NULL

products TABLE

Result Grid | Filter Rows: | Edit:

product_id	product_name	product_price	category_id
1	맥북에어	650000	1
2	맥북프로	700000	1
3	맥북프로맥스	750000	1
4	맥북파우치	10000	2
5	DT-35키보드	20000	2
6	독거미키보드	50000	2
NULL	NULL	NULL	NULL

categories TABLE

category_id	category_name
1	전자기기
2	주변기기
NULL	NULL

orders TABLE

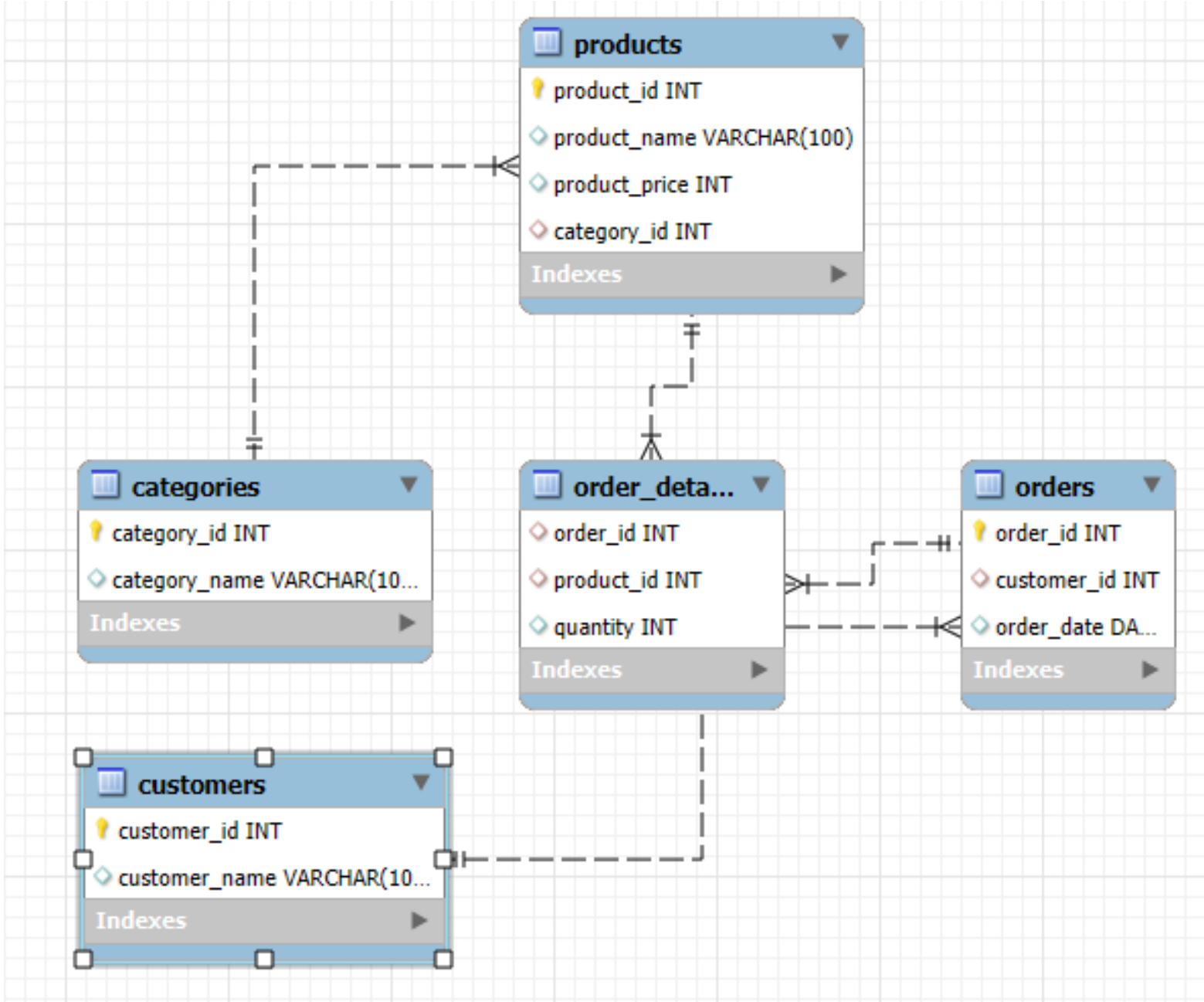
Result Grid | Filter Rows:

order_id	customer_id	order_date
1	1	2024-06-01
2	2	2023-06-02
3	3	2023-06-03
4	4	2023-06-04
5	5	2023-06-05
6	6	2023-06-06
7	7	2023-06-07
NULL	NULL	NULL

order_details TABLE

Result Grid | Filter Rows:

order_id	product_id	quantity
1	1	1
1	4	1
2	2	1
3	3	1
4	5	1
5	6	1
6	6	1
7	6	1


products
product_id INT
product_name VARCHAR(100)
product_price INT
category_id INT
Indexes
categories
category_id INT
category_name VARCHAR(10...
Indexes
order_deta...
order_id INT
product_id INT
quantity INT
Indexes
orders
order_id INT
customer_id INT
order_date DA...
Indexes
customers
customer_id INT
customer_name VARCHAR(10...
Indexes

LEFT JOIN

`customers` 테이블

customer_id	customer_name	customer_email
1	Alice Johnson	alice@example.com
2	Bob Smith	bob@example.com
3	Charlie Brown	charlie@example.com

`orders` 테이블

order_id	customer_id	order_date
1	1	2023-07-01
2	2	2023-07-02
3	1	2023-07-03
4	4	2023-07-04

customer_id 가 3번인
찰리는 주문을
한적이 없습니다!

```sql
SELECT
    c.customer_id,
    c.customer_name,
    o.order_id,
    o.order_date
FROM
    customers c
LEFT JOIN
    orders o ON c.customer_id = o.customer_id;
```

JOIN 의 왼쪽인
customer 를 기준으로
LEFT JOIN 을 하면!?

customer_id	customer_name	order_id	order_date
1	Alice Johnson	1	2023-07-01
1	Alice Johnson	3	2023-07-03
2	Bob Smith	2	2023-07-02
3	Charlie Brown	NULL	NULL

LEFT JOIN 이라서 customer 테이블
데이터는 전부 출력을 해야하지만
찰리는 주문을 한적이 없으므로
NULL 이 들어간 상태로 테이블이 생성

RIGHT JOIN

`customers` 테이블

customer_id	customer_name	customer_email
1	Alice Johnson	alice@example.com
2	Bob Smith	bob@example.com
3	Charlie Brown	charlie@example.com

`orders` 테이블

order_id	customer_id	order_date
1	1	2023-07-01
2	2	2023-07-02
3	1	2023-07-03
4	4	2023-07-04

customer_id 가 4번인
고객은 없는 상태입니다!

```sql
SELECT
    c.customer_id,
    c.customer_name,
    o.order_id,
    o.order_date
FROM
    customers c
RIGHT JOIN
    orders o ON c.customer_id = o.customer_id;
```

JOIN의 오른쪽인
orders 를 기준으로
RIGHT JOIN 을 하면!?

customer_id	customer_name	order_id	order_date
1	Alice Johnson	1	2023-07-01
1	Alice Johnson	3	2023-07-03
2	Bob Smith	2	2023-07-02
NULL	NULL	4	2023-07-04

RIGHT JOIN 이라서 orders 테이블 데이터는 전부 출력을 해야하지만 order_id 4번의 주문자 정보를 찾을 수 없으므로 NULL 표시