8253jang@daum.net
장희정

제1정규형 (1NF, First Normal Form)

정의

- 테이블이 원자값(Atomic Value)을 갖도록 설계해야 한다.
- 즉, **각 컬럼(속성)에 하나의 값만 존재**해야 하며, **반복되는 그룹(중복 컬럼)이 없어야 한다.**

.

어떤 상황이 1NF 를 위반하는가?

1NF 위반 예시

학생(Student) 테이블

학번	이름	수강과목
2023001	장희정	DB, 운영체제
2023002	이가현	네트워크
2023003	이찬범	운영체제, 자료구조

수강 과목 속성에 여러 개의 값(DB, 운영체제)이 포함됨 → **반복 그룹 존재**
즉, 하나의 셀에 여러 개의 값이 들어가 있어 **1NF 위반**

1NF로 변환

학생(Student) 테이블에서 **수강 과목 속성을 분리**
수강(Student_Course) 테이블을 **새롭게 생성**
기존 학생(Student) 테이블과 **1:N 관계 형성**

학생(Student) 테이블

학번	이름
2023001	장희정
2023002	이가현
2023003	이찬범

수강(Student_Course) 테이블 (1:N 관계)

학번	과목명
2023001	DB
2023001	운영체제
2023002	네트워크
2023003	운영체제
2023003	자료구조

8253jang@daum.net
장희정

2. 제2정규형 (2NF, Second Normal Form)

정의

- **1NF 를 만족**하면서,
- **부분 함수 종속(Partial Dependency)이 제거**되어야 한다.
  즉, **기본키(Primary Key)의 일부에만 종속된 속성이 없어야 한다.**

어떤 상황이 2NF 를 위반하는가?

2NF 위반 예시

(학생ID, 강좌ID)를 기본키로 하는 테이블

학생ID	강좌ID	강좌명	교수명
S001	C001	데이터베이스	홍길동
S002	C002	운영체제	이순신

여기서 기본키는 **(학생ID, 강좌ID)** → 즉, 한 학생이 여러 강좌를 들을 수 있다.
하지만 **강좌명과 교수명은 강좌 ID 에만 종속**되므로, **부분 함수 종속**이 발생한다.

2NF 로 변환

해결 방법: 강좌 정보를 별도 테이블로 분리

학생-강좌 등록 테이블

학생ID	강좌ID
S001	C001
S002	C002

강좌 테이블

강좌ID	강좌명	교수명
C001	데이터베이스	홍길동
C002	운영체제	이순신

이제 **강좌명과 교수명은 강좌ID에만 종속**되므로 **부분 종속이 제거됨.**

3. 제3정규형 (3NF, Third Normal Form)

8253jang@daum.net
장희정

정의

- **2NF 를 만족**하면서,
- **이행적 함수 종속(Transitive Dependency)이 제거**되어야 한다.
  즉, **기본키가 아닌 속성이 또 다른 비기본키 속성을 결정하면 안 된다.**

어떤 상황이 3NF 를 위반하는가?

3NF 위반 예시

직원ID	직원명	부서ID	부서명
101	홍길동	D001	개발팀
102	이순신	D002	영업팀

여기서 **기본키는 직원 ID**.
하지만 **부서명은 부서 ID 에 종속**됨 → 즉, 직원 ID 가 부서 ID 를 결정하고, 부서 ID 가 부서명을 결정하는 **이행적 종속 발생**한다.

3NF로 변환

직원 테이블

직원ID	직원명	부서ID
101	홍길동	D001
102	이순신	D002

부서 테이블

부서ID	부서명
D001	개발팀
D002	영업팀

이제 **부서명이 부서ID에만 종속**되어 **이행적 종속이 제거됨**.

정리

정규형	해결하는 문제	예시
1NF	중복된 컬럼 및 다중 값(Atomic Value 위반)	연락처가 한 칸에 여러 개 들어가는 경우
2NF	부분 함수 종속(Partial Dependency 제거)	강좌명이 강좌ID에 종속되지만, 학생-강좌 테이블에 포함된 경우
3NF	이행적 종속(Transitive Dependency 제거)	부서명이 부서ID에 종속되지만, 직원 테이블에 포함된 경우

8253jang@daum.net
장희정

정규화를 잘 수행하면 **데이터 중복이 줄어들고, 데이터 무결성이 보장**되며, **데이터 업데이트 시 이상현상(Anomaly) 발생을 방지**할 수 있다.

1. 관계 차수(Cardinality)란?

관계 차수는 두 개체 간에 **몇 개의 인스턴스가 연결될 수 있는지**를 나타내는 개념이다. 쉽게 말해, "하나의 엔터티가 다른 엔터티와 몇 개의 관계를 가질 수 있는가?"를 정의하는 것이다.

주요 관계 차수 유형

1. **일대일 (1:1, One-to-One)**
2. **일대다 (1:N, One-to-Many)**
3. **다대다 (M:N, Many-to-Many)**

(1) 일대일(1:1) 관계

한 개체(Entity)의 한 인스턴스가 다른 개체의 **하나의 인스턴스**와만 관계를 맺는다.

예시)

주민등록번호	이름	여권번호
123456-7890123	홍길동	P12345
987654-3210987	이순신	P67890

하나의 주민등록번호는 하나의 여권번호와만 연결됨.
여권번호도 마찬가지로 하나의 주민등록번호에만 속함.

(2) 일대다(1:N) 관계

한 개체의 한 인스턴스가 다른 개체의 **여러 개 인스턴스**와 관계를 가짐.

예시)

고객ID	고객명	주문ID
C001	홍길동	O1001
C001	홍길동	O1002
C002	이순신	O1003

한 명의 고객(고객 ID)이 여러 개의 주문(주문 ID)을 할 수 있음.

8253jang@daum.net
장희정

하지만 **각 주문은 단 하나의 고객에게 속함**.

(3) 다 대 다(M:N) 관계

한 개체의 한 인스턴스가 다른 개체의 **여러 개 인스턴스**와 관계를 맺을 수 있으며, 반대도 성립함.

예시)

학생ID	학생명	강좌ID	강좌명
S001	홍길동	C001	데이터베이스
S001	홍길동	C002	운영체제
S002	이순신	C001	데이터베이스

한 명의 학생이 여러 강좌를 수강할 수 있음.
동시에 **하나의 강좌는 여러 학생이 수강할 수 있음**.
이를 해결하기 위해 **중간 테이블(연결 테이블, Bridge Table)을 사용하여 분리**해야 함.

해결 방법 (중간 테이블 추가)

학생-강좌 테이블

학생ID	강좌ID
S001	C001
S001	C002
S002	C001

이제 학생과 강좌 사이의 관계를 다대다에서 **일대다(1:N) + 다대일(N:1) 관계로 변환**하여 관리할 수 있다.

2. 선택성(Mandatory vs. Optional)이란?

선택성은 **개체 간 관계에서 특정 개체의 존재 여부가 필수적인지 아닌지**를 나타낸다.

(1) Mandatory (필수) -> |

- 관계가 **반드시 존재해야 함**.
- 특정 개체가 없으면 다른 개체도 존재할 수 없음.

예시 (Mandatory 관계)

8253jang@daum.net
장희정

직원ID	직원명	부서ID
101	홍길동	D001
102	이순신	D002

모든 직원은 반드시 하나의 부서에 속해야 함 → **부서ID가 NULL이면 안 됨**.

2) Optional (선택적) -> o

- 관계가 **필수가 아니라 선택 가능**.
- 개체가 없어도 다른 개체가 존재할 수 있음.

예시 (Optional 관계)

직원ID	직원명	부서ID
101	홍길동	D001
102	이순신	NULL

어떤 직원은 특정 부서에 소속되지 않을 수도 있음 → **부서ID가 NULL 가능**.

관계 차수(Cardinality)와 선택성(Mandatory vs. Optional) 조합 예시

관계 유형	예시	Mandatory or Optional
1:1	주민등록번호 - 여권	Mandatory
1:N	고객 - 주문	고객(Mandatory), 주문(Optional)
M:N	학생 - 강좌	Optional
1:N	직원 - 부서	직원(Mandatory), 부서(Optional)

결론

관계 차수(Cardinality)

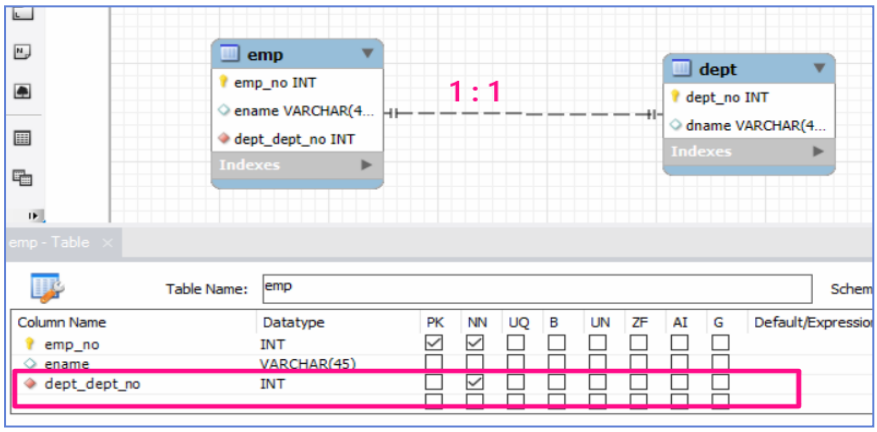
- ✓ **1:1** → 한 개체가 다른 개체의 하나의 인스턴스와만 연결됨.
- ✓ **1:N** → 한 개체가 다른 개체의 여러 개 인스턴스와 연결됨.
- ✓ **M:N** → 한 개체가 다른 개체의 여러 개 인스턴스와 서로 연결될 수 있음 (중간 테이블로 변환 필요).

8253jang@daum.net
장희정

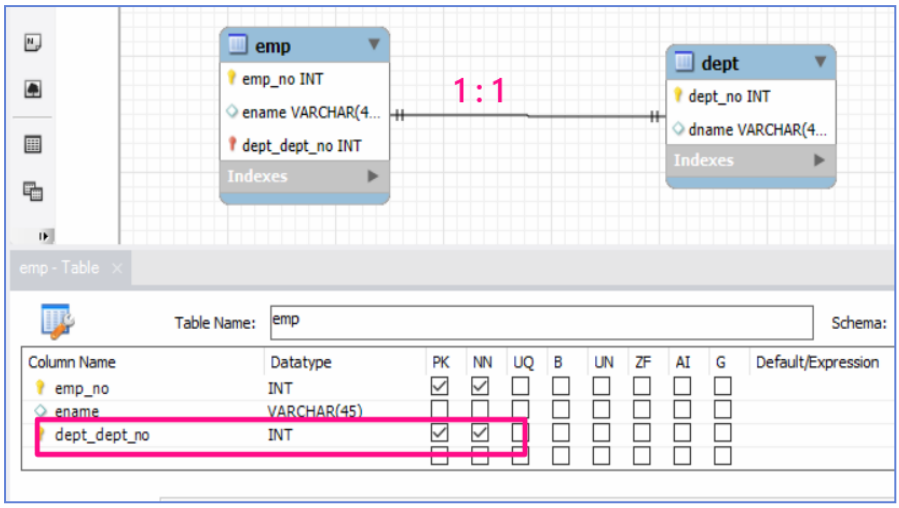
선택성(Mandatory vs. Optional)

- ✓ **Mandatory** → 관계가 반드시 존재해야 함.
- ✓ **Optional** → 관계가 없어도 개체가 독립적으로 존재 가능.

관계 = 1 : 1(비식별관계) – 부모의 pk가 자식의 일반속성으로

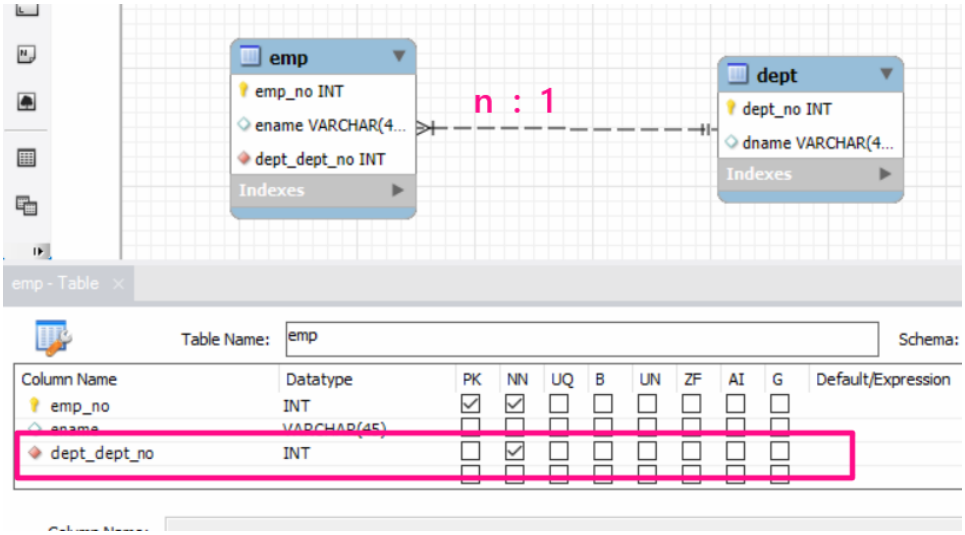


관계 = 1 : 1(식별관계) – 부모의 pk가 자식의 pk로

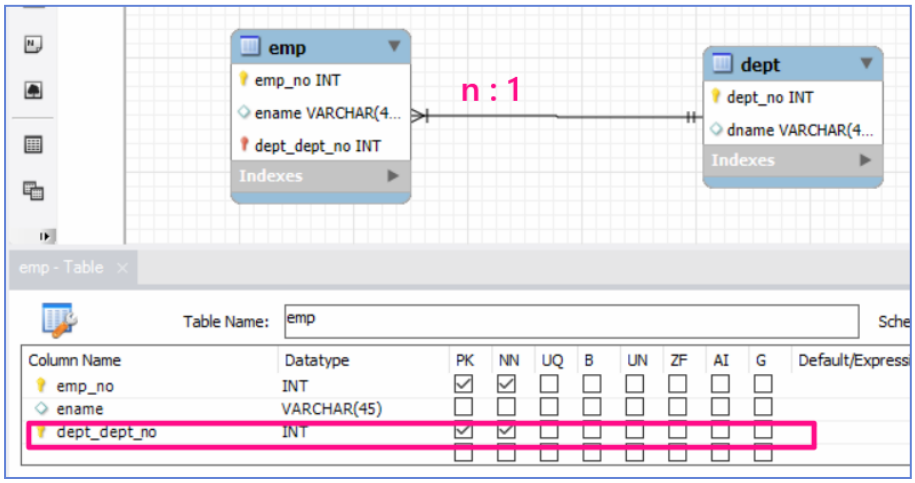


관계 = n : 1(비식별관계)

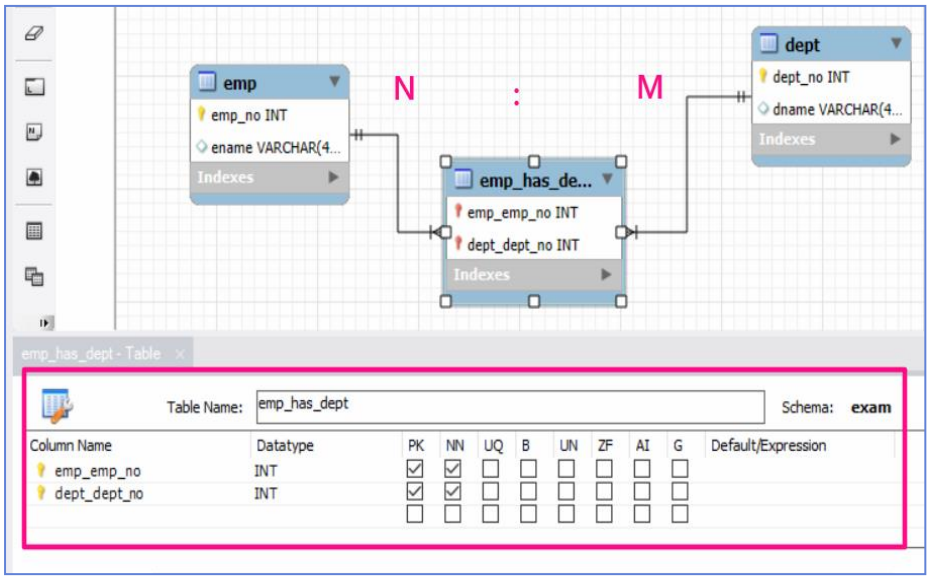
8253jang@daum.net
장희정



관계 = n : 1(식별관계)



관계 = n : m(식별관계) – 새로운테이블에 양쪽 부모의 pk가 자식의 pk로



8253jang@daum.net
장희정

ERD(Entity Relationship Diagram)는 어느 모델링 단계에서 만들까?

ERD는 **데이터베이스 모델링 과정에서 개념적, 논리적, 물리적 단계 모두에서 사용될 수 있다.**
하지만 **각 단계에서 ERD의 디테일과 목적이 달라진다.**

단계	ERD 활용 여부	설명
개념적 모델링	✅ 사용	주요 엔터티(Entity)와 관계(Relationship) 중심으로 개념적 ERD 작성
논리적 모델링	✅ 사용	속성(Attribute), PK & FK, 관계 차수(Cardinality), 식별/비식별 관계 추가
물리적 모델링	✅ 사용	실제 테이블(Table), 데이터 타입(Data Type), 제약 조건(Constraints) 추가

ERD 유형	어느 단계에서?	특징
개념적 ERD	개념적 모델링	엔터티 & 관계 중심, PK & FK 없음, Mandatory & Optional만 결정
논리적 ERD	논리적 모델링	속성 추가, PK & FK 설정, 식별 & 비식별 관계 결정
물리적 ERD	물리적 모델링	실제 테이블과 데이터 타입, 제약 조건 반영

식별 관계 vs. 비식별 관계 & Mandatory vs. Optional 설계 단계

개념	적용 시점	설명
Mandatory & Optional 관계	개념적 모델링 (1단계)	비즈니스 규칙에 따라 "필수"인지 "선택"인지 결정
식별 관계 & 비식별 관계	논리적 모델링 (2단계)	PK와 FK를 설정하면서 "식별" 또는 "비식별" 관계를 결정
PK, FK, 인덱스, 제약 조건	물리적 모델링 (3단계)	DBMS에 맞춰 최적화

Sample 요구사항

1. 고객(Customer) 관리

- ✓ 고객은 회원 가입을 통해 고유한 ID(user_id)를 가짐.
- ✓ 회원 가입 시, **비밀번호(user_pwd), 이름(user_name), 가입일(reg_date)** 정보를 입력해야 함.
- ✓ 고객 정보는 **로그인 및 주문 시 필요**

.

2. 상품(Goods) 관리

8253jang@daum.net
장희정

- ✓ 쇼핑몰에서 판매하는 각 상품(goods)은 고유한 상품 ID(goods_id)를 가짐.
- ✓ 상품 정보에는 상품명(goods_name), 가격(goods_price), 재고(stock), 등록일(regdate)이 포함됨.
- ✓ 상품 등록 후 **가격 및 재고 변경 가능**.
- ✓ **상품이 주문되지 않아도 존재할 수 있음 (Optional 관계)**.

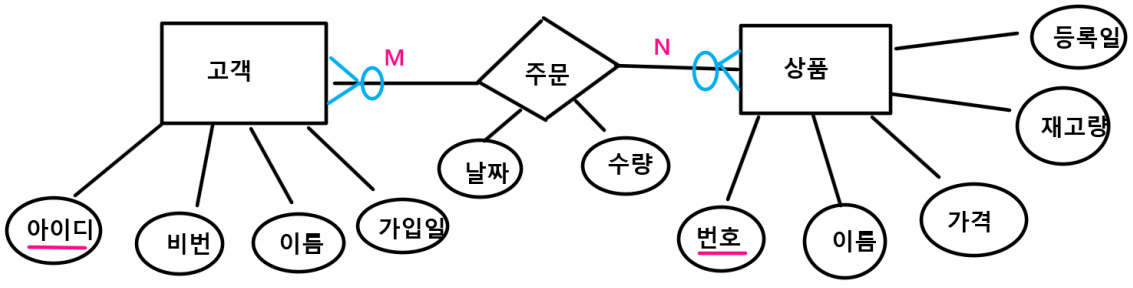
3. 주문(Orders) 관리

- ✓ 고객이 상품을 구매할 때 주문이 생성됨.
- ✓ 주문에는 고유한 주문번호(order_id)가 부여됨.
- ✓ 주문 시, 주문 날짜(order_date), 고객 ID(user_id), 배송지(address), 총 구매 금액(total_amount)가 저장됨.
- ✓ 한 명의 고객이 여러 개의 주문을 할 수 있음 (**1:N 관계**).
- ✓ 주문이 존재하지 않을 경우 **주문 상세(order_line)도 존재할 수 없음** (**Mandatory 관계**).

4. 주문 상세(Order_Line) 관리

- ✓ 주문이 생성될 때, 주문 내역(주문한 상품)이 order_line 에 저장됨.
- ✓ 주문 상세에는 고유한 주문 상세 번호(order_line_id)가 부여됨.
- ✓ 하나의 주문에는 **여러 개의 상품이 포함될 수 있음 (1:N 관계)**.
- ✓ 주문 상세 정보에는 상품(goods_id), 단가(unit_price), 수량(qty), 총 금액(amount)이 저장됨.
- ✓ 주문 상세는 반드시 주문(order_id)과 상품(goods_id)을 참조해야 함 (**Mandatory 관계**).

개념적 설계단계 ERD

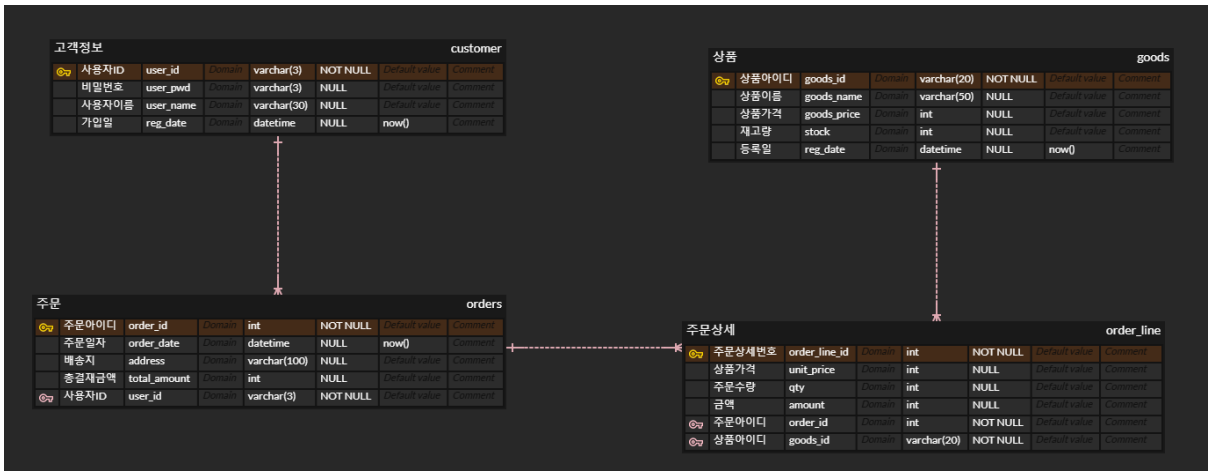


좌 -> 우 : 고객은 하나이상의 상품을 주문 할지도 모른다. (.Optional = '~일지도 모른다.')

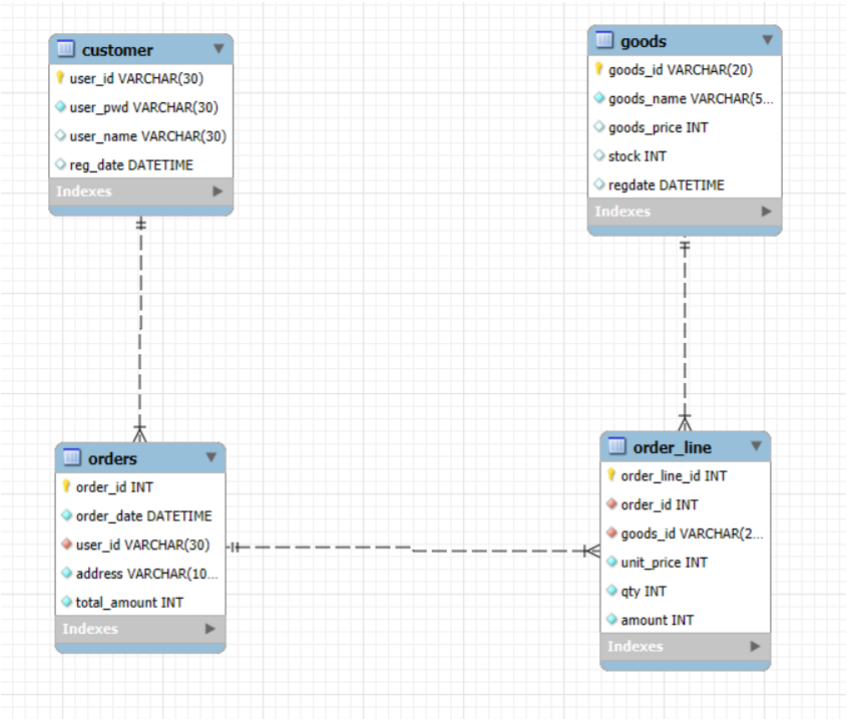
우 -> 좌 : 상품은 한명이상의 고객으로 부터 주문될지도 모른다.(Optional = '~일지도 모른다')

8253jang@daum.net
장희정

erdcloud.com 에서 논리, 물리 ERD설계



MySQL_Workbench에서 ERD 설계



ERD설계로 나온 schema정보

```
create database mygoods;
use mygoods;
```

8253jang@daum.net
장희정

```
create table customer(
   user_id varchar(30) primary key ,
   user_pwd varchar(30) not null,
   user_name varchar(30),
   reg_date datetime
);

create table goods(
   goods_id varchar(20) primary key,
   goods_name varchar(50) not null,
   goods_price int,
   stock int,
   regdate datetime default now()
);



create table orders(
   order_id int primary key auto_increment,
   order_date datetime not null,
   user_id varchar(30) not null ,
   address varchar(100) not null,
   total_amount int not null ,
   foreign key(user_id) references customer(user_id)
);

create table order_line(
   order_line_id int primary key auto_increment,
   order_id int not null,
   goods_id varchar(20) not null,
   unit_price int not null,
   qty int not null,
   amount int not null ,
   foreign key(order_id)   references orders(order_id),
   foreign key(goods_id) references goods(goods_id)
);
```