데이터베이스시스템

Project2 보고서

정성원 교수님

01반

20181297

수학과

조다민

1. 프로젝트 개요

이번 프로젝트의 목표는 project1에서 작성한 정보를 바탕으로 mysql을 이용해 실제 데이터베이스를 구축하는 것이다. 이 프로젝트를 통해 판매되는 상품, 매장, 고객 정보, 재고 상태, 고객 주문 내역 등의 내용을 관리할 수 있는 데이터베이스에 대한 기초적인 정보를 담은 logical/physical schema와 그리고 몇 가지 예시 SQL문을 작성하였다.

2. BCNF checking

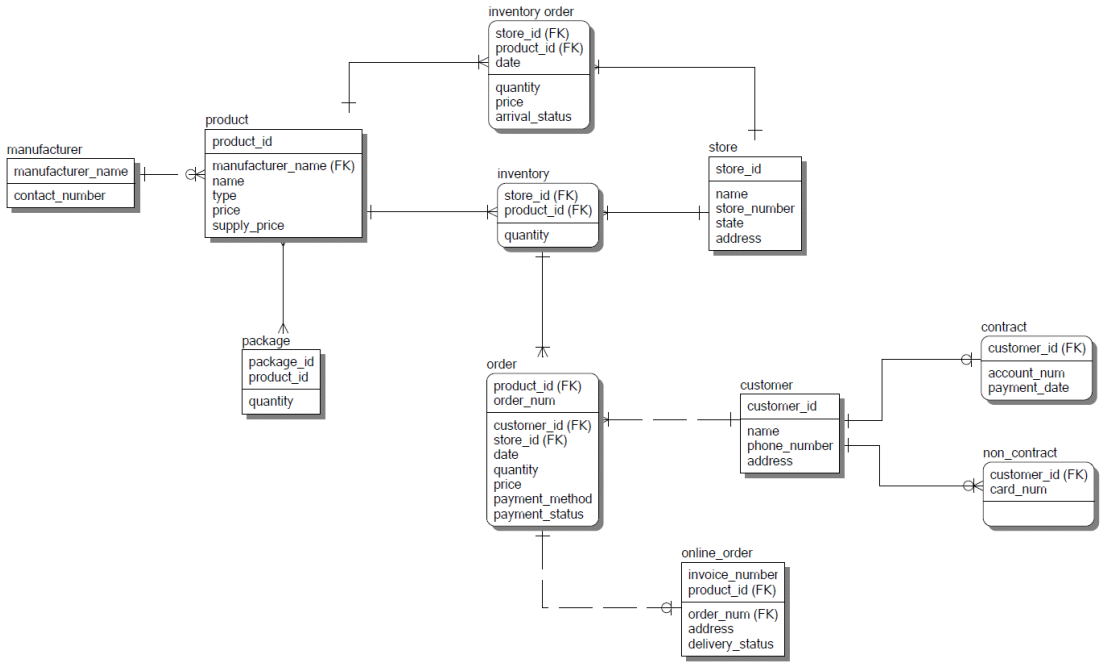


그림 1: project1에서 작성한 relation schema

위의 relation schema 에서 확인했을 때, 다른 table 들은 BCNF 를 만족함을 확인할 수 있었으나, order 의 경우 payment_method, payment_status 의 정보 중복이 심하다는 것을 인지하고 이를 BCNF 형태로 수정하기 위해서 order 를 order_list 와 order_prd 로 나누었다.

이를 수정한 형태는 다음과 같다.

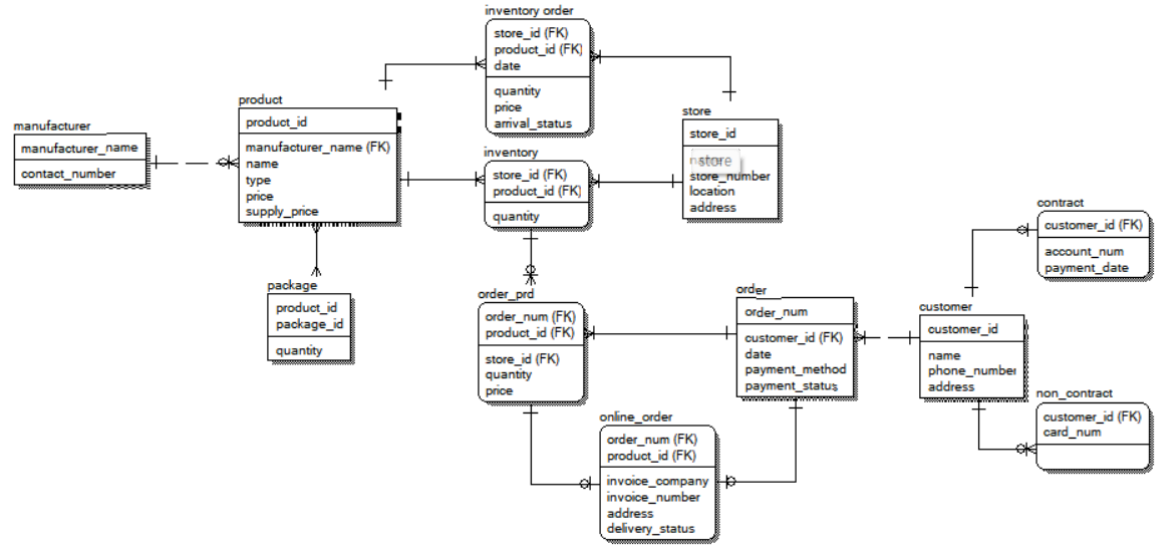
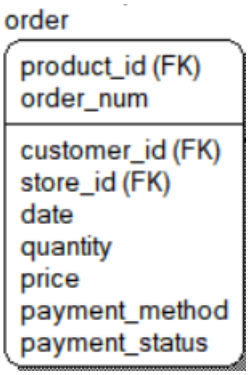
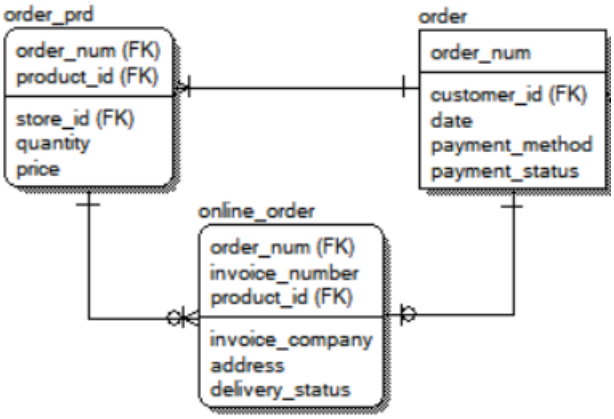


그림 2: 수정된 logical schema

초기의 order 에 대해 각각의 column 에 대하여 A 부터 차례로 별칭을 붙여서 살펴보자.



해당 table 에서 우리는 B->DEHI, AB->CFG 의 relation 을 확인할 수 있다. 이때, A 의 경우 DEHI 와 relation 이 없으므로 우리는 이를 AB->CFG(order_prd)와 B->DEHI(order_list)로 나눌 수 있다.



바뀐 logical schema 는 다음과 같이 나타낼 수 있다.

3. Physical Schema

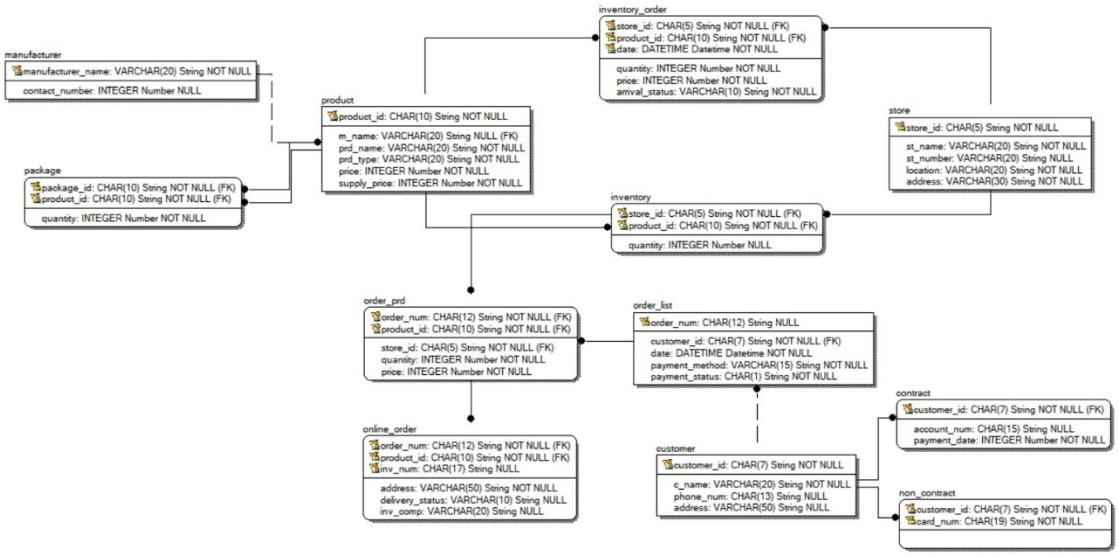


그림 3: logical schema를 바탕으로 한 physical schema. MySQL 예약어와 겹치는 table, column 명을 수정하고 자료형에 대한 정보를 추가하였다.

Table	Column	Key	Data Type	Description
manufacturer	m_name	PK	varchar(20)	Not NULL, 제조사명.
	contact_num		varchar(20)	제조사와의 연락처 저장
product	product_id	PK	char(10)	제품 고유 id 부여, 길이가 10 으로 고정되기 때문에 char 사용. XXXX000000 형태
	m_name	FK	varchar(20)	제조사명.
	prd_name		varchar(20)	제품명.
	prd_type		varchar(20)	제품 분류. ex) monitor, tv, accessory etc.
	price		int	판매가. 단위 $
	supply_price		int	공급가. 단위 $
package	package_id	PK, FK	char(10)	product 중 package 로 묶어서 판매하는 경우.
	product_id	PK, FK	char(10)	package 의 구성요소
	quantity		int	package 구성요소 수량
store	store_id	PK	char(5)	store 고유 id. XX000 형태
	st_name		varchar(20)	도시명으로 store 의 name 을 저장.
	st_number		varchar(20)	매장 번호
	location		varchar(20)	매장의 위치를 state 단위로 분류 저장.
	address		varchar(30)	매장의 상세 주소를 저장.
inventory	store_id	PK, FK	char(5)	
	product_id	PK, FK	char(10)	
	quantity		int	매장의 품목별 수량을 저장
inventory_order	store_id	PK, FK	char(5)	
	product_id	PK, FK	char(10)	
	o_date	PK	datetime	매장의 재고 주문 시간
	quantity		int	재고 주문 수량
	price		int	주문 가격
	arrival_status		varchar(10)	주문 상태 확인
customer	customer_id	PK	char(7)	고객 고유 id
	c_name		varchar(20)	고객 이름
	phone_num		char(13)	고객 전화번호 010-0000-0000 형태
	address		varchar(50)	고객 주소
contract	customer_id	PK,FK	char(7)	

	account_num		char(15)	고객 입금 계좌. 0000-000-000000 형태
	payment_date		int	결제일.
non_contract	customer_id	PK, FK	char(7)	
	card_num	PK	char(19)	온라인 결제 고객 카드 정보 저장. 0000-0000-0000-0000 형태
order_list	order_num	PK	char(12)	고객 주문 번호 000000000000 형태
	customer_id	FK	char(7)	
	o_date		datetime	주문 시간
	payment_method		varchar(15)	결제 방식. account, credit_card, cash 로 나눈다.
	payment_status		char(1)	contract 고객의 경우 결제를 다음달에 한번에 처리하므로 결제 여부를 저장.
order_prd	order_num	PK, FK	char(12)	
	product_id	PK, FK	char(10)	
	store_id	FK	char(5)	
	quantity		int	주문 수량
	price		int	결제가
online_order	ord_num	PK, FK	char(12)	order_num 을 참조
	product_id	PK, FK	char(10)	
	inv_num	PK, FK	char(17)	송장번호. 0000-0000-0000000 형식
	inv_comp		varchar(20)	택배사
	address		varchar(50)	배송지
	delivery_status		varchar(10)	checking, started, done 의 형태.

4. ODBC

1) Connection

먼저 Connection을 위해 mysql server에 연결하기 위한 변수를 설정한다. cpp파일에서 host, user, pw, db를 server 상황에 맞춰 설정한 뒤 프로그램을 실행한다. 먼저 buffer 변수에 sql 내용을 담은 txt 파일의 내용을 저장한 뒤, server와 연결에 성공하면 Connection Succeed가 출력된다. 이후 설정된 db 변수에 따라 사용할 database를 선정한다.

다음으로 buffer에 담긴 내용을 ';'를 단위로 해 끊어가며 sql문을 init에 저장하고 mysql_query를 이용해서 서버에 실행한다. 이를 buffer에서 'drop table'을 만날 때까지 실행 후 break 한다. 해당 while문 실행 시 database에 table과 tuple이 생성된다.

2) Type 별 sql

- Type 1(Assume the package shipped by USPS with tracking number X is reported to have been destroyed in an accident. Find the contact information for the customer.): 이를 처리하기 위해서 다음과 같은 sql문을 사용하였다.

```
"select customer_id, c_name, phone_num
    from customer
        natural join(
            select customer_id from order_list, online_order
            where (order_list.order_num = online_order.ord_num) and
            (inv_comp = ₩"%s₩" and inv_num = ₩"%s₩"))B",
```

입력한 내용이 sprintf를 이용해서 inv_comp, inv_num가 %s에 차례로 들어간다.

예시 입력으로 inv_comp에 USPS, inv_num에 9375-8727-1163137를 넣었을 때 다음과 같은 결과가 출력된다.

```
---- TYPE 1 ----

**Enter the company to be tracked**
USPS
**Enter the invoice number to be tracked.**
ex)0000-0000-0000000
9375-8727-1163137
customer_id: CKO2551
name: Ruby Alvarado
phone: 010-3606-8927
customer_id: CKO2551
name: Ruby Alvarado
phone: 010-3606-8927
----------------------------------
```

그리고 subtype을 실행하면

```
---- TYPE 1-1 ----

execute: insert into online_order values ("225256699248", "bbbb000001", "USPS", "2724-6758-1488622", "P.O. Box 736, 8316
 Ornare, St.","started")
success
execute: insert into online_order values ("225256699248", "JSCL772930", "USPS", "2724-6758-1488622", "P.O. Box 736, 8316
 Ornare, St.","started")
success
```

이렇게 해당하는 주문번호, 제품 id, 새로운 송장번호로 tuple을 추가하도록 해주었다.

- Type2: 지난 해 가장 많은 금액을 주문한 고객을 찾기 위해서 다음과 같은 sql문을 사용하였다.

```
select customer_id, c_name, sp from customer natural join (
    select customer_id, sum(price) as sp from order_list, order_prd
    where (order_list.order_num = order_prd.order_num) and (YEAR(order_list.o_date) = 2021)
    group by customer_id
    order by sp desc LIMIT 1
)a;
```

이를 실행하면 프로그램 내에서 가장 많음 금액을 주문한 고객 정보와 주문 금액의 합계를 알려준다.

```
---- TYPE 2 ----

Find the customer who has bought the most (by price) in the past year.
customer_id: YWJ5767
name: Ivor Bradshaw
 sum: 4916
----------------------------------
```

2-1의 실행에서는 두개의 sql문을 사용한다. 먼저 첫번째 sql문을 사용해서

가장 많은 금액을 사용한 고객의 customer_id를 저장하고, 두번째로 해당 고객이 주문한 제품을 제품별로 묶어 어떤 제품을 가장 많이 구매했는지에 대한 결과를 출력한다. 사용한 두 개의 sql문은 다음과 같다.

```
select customer_id, c_name, sp from customer natural join (
select customer_id, sum(price) as sp from order_list, order_prd
where(order_list.order_num = order_prd.order_num) and (YEAR(order_list.o_date) = 2021)
group by customer_id order by sp desc LIMIT 1)a;

select product_id from order_list natural join order_prd
where (YEAR(order_list.o_date) = 2021)
group by product_id;
```

이를 실행하면 다음과 같이 결과를 출력한다.

- Type3: 작년에 팔린 모든 제품을 출력하기 위해서 다음과 같은 sql문을 사용한다.

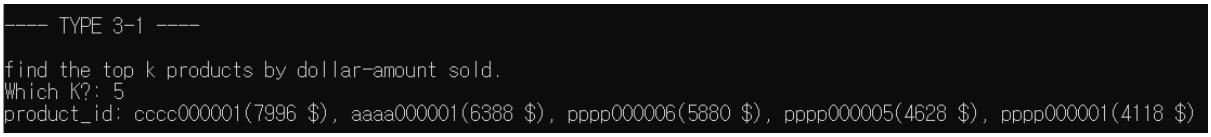
```
select product_id from order_list natural join order_prd
where (YEAR(order_list.o_date) = 2021)
group by product_id;
```

이를 실행한 결과를 받아 프로그램에서는 다음과 같이 출력한다.

그리고 subtype에서는 판매가를 기준으로 각각 상위 k개, 상위 10%의 품목을 출력한다. 3-1에서는 먼저 k를 입력 받고 해당 정보를 sprinf를 이용해서 LIMIT 뒤에 숫자로 들어가게 되고 최종적으로 server에 전달되는 sql문은 다음과 같다.

```
select product_id, sum(price) as sp from order_list natural join order_prd
where (YEAR(order_list.o_date) = 2021)
group by product_id order by sp desc LIMIT 5;
```

이를 실행하면 다음과 같은 결과가 출력된다.



매출량이 많은 순으로 제품 id와 괄호 안에 매출 금액이 출력된다.

Type3-2의 경우 상위 10%가 몇 개인지 구하기 위해서 먼저 2021년에 팔린 product가 몇 종류인지를 구하고 이를 10으로 나눠서 그 값을 k에 저장한다. 그리고 저장한 k를 3-1에서와 마찬가지로 sprintf를 이용해 LIMIT k 형태로 sql문을 실행한다. 실행한 결과는 다음과 같다.

현재 2021년에 판매된 product종류가 33개이므로 상위 10% 즉 3개까지 출력한 것을 볼 수 있다.

- Type4: 이번에는 price가 기준이 아니라 quantity를 기준으로 이를 나열해보자. 작년에 팔린 product와 그 수량을 알기 위해서 다음과 같은 sql문을 실행하였다.

```
select product_id, sum(quantity)
    from order_list natural join order_prd
    where(YEAR(order_list.o_date) = 2021) group by product_id;
```

실행 결과는 다음과 같다.

2021년에 팔린 상품을 상품id*개수 로 출력하고 있다.

Type 4-1의 경우 판매 수량을 기준으로 했을 때, 상위 k개의 상품을 출력하고자 한다. 3-1에서와 비슷하게 order by와 limit를 사용해서 sql문을 작성하였다. sql문과 그 실행결과는 다음과 같다.

```
select product_id, sum(quantity) as sq
    from order_list natural join order_prd
    where(YEAR(order_list.o_date) = 2021)
    group by product_id order by sq desc LIMIT 4;
```

```
---- TYPE 4-1 ----

find the top k products by dollar-amount sold.
Which K?: 4
product_id: aaaa000001*4, bbbb000001*4, cccc000001*4, pppp000005*4
-----------------------------------
```

Type 4-2의 경우 판매 수량을 기준으로 했을 때, 상위 10%의 상품을 출력하고자 한다. 3-2에서와 비슷하게 먼저 판매된 상품의 수를 count를 이용해서 받아와 k로 저장하고 order by와 limit를 사용해서 sql문을 작성하였다. sql문과 그 실행결과는 다음과 같다.

```
select count(product_id) as cnt
    from order_list natural join order_prd
    where(YEAR(order_list.o_date) = 2021);

select product_id, sum(quantity) as sq
    from order_list natural join order_prd
    where(YEAR(order_list.o_date) = 2021)
    group by product_id order by sq desc LIMIT 3;
```

```
---- TYPE 4-2 ----

find the top 10% products by dollar-amount sold.
product_id: aaaa000001*4, bbbb000001*4, cccc000001*4
-----------------------------------
```

- Type5: 캘리포니아 주에 있는 모든 store에서 품절인 상품들을 찾기 위해서 not in 명령어와 subquery를 사용해 다음과 같은 sql문을 작성하였다. 그 실행결과는 다음과 같다.

```
select product_id
from product
where product_id not in
    (select product_id from inventory natural join store
    where store.location = "California"
        and inventory.quantity >0);
```

```
---- TYPE 5 ----

Find those products that are out-of-stock at every store in California.
product_id: pppp000006, aaaa000002, aaaa000003, aaaa000004, pppp000003, BPBA478253, JSCL772930, MRXP326681, pppp000005,
WUDW201145, cccc000002, cccc000003, pppp000002, JIKD383735, TRTG141695, pppp000001, pppp000004, pppp000008, VWYG422647,
pppp000007
-------------------------------
```

- Type6: 아직 도착하지 않은 택배들 중 약속된 날짜(3일)안에 도착하지 않은 택배를 찾기 위해서 datediff를 이용해서 현재 날짜와 해당 주문이 들어온 날짜를 비교해서 그 값이 3 초과인 주문들에 대한 송장번호를 출력하도록 sql문을 작성하였다. sql문과 실행결과는 다음과 같다.

```
with a as(
    select order_num, inv_num, delivery_status, o_date,  datediff(date(now()), date(o_date)) as diff
    from (order_list natural join order_prd), online_order
    where (order_list.order_num = online_order.ord_num)
        and (order_prd.product_id = online_order.product_id))
select inv_num, diff
from a
where (diff>3) and (delivery_status != "done");
```

```
---- TYPE 6 ----

Find those packages that were not delivered within the promised time.
inv_num: 7678-6452-6168568(5 days), 5467-1531-9457171(5 days), 2522-3766-0265874(5 days)
---------------------------------
```

- Type7: contract 고객의 경우 account로 지정된 날짜에 모든 내역을 한번에 결제하는데, 각각 고객에 대한 구매 총합을 계산하고 이를 고객별로 출력하는 sql문을 작성하였다. sql문은 5월달을 기준으로 작성되었으며, sql문과 실행결과는 다음과 같다.

```
select customer_id, sum(price), count(order_num)
from order_prd natural join order_list
where payment_status = "x" and MONTH(order_list.o_date)=5
group by customer_id;
```

```
---- TYPE 7 ----

Generate the bill for each customer for the past month.
customer_id|Number of orders|total price
  MVD7235              3        4817
  QSF9658              1        2099
  HIV7677              1         239
---------------------------------
```

3) Close

사용자가 0을 입력해 query select 단계를 나오면 프로그램을 종료한다. 실험의 편의를 위해 table을 drop해 database를 정리한다. 앞서 1)에서 buffer에 저장해 둔 sql문 중 drop관련 명령어가 아직 buffer 변수에 남아있으므로, 동일하게 ';'를 단위로 끊어가며 sql문을 실행한다.