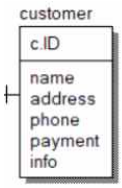
1. logical schema

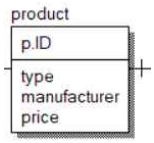
1) relation

- customer(c.ID, name, address, phone, payment, info)



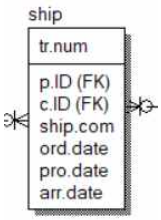
: customer는 고객 정보를 나타내는 relation으로, 각 고객마다 unique한 하나의 c.ID를 갖는다. 따라서 c.ID는 customer의 primary key이고, 그 외 attribute로는 고객의 이름을 나타내는 name, 주소를 나타내는 address, 전화번호를 나타내는 phone, 지불방법을 나타내는 payment, 지불 방법에 따라 계좌번호 또는 카드번호를 나타내는 info가 있다.
phone은 (TYPE 1)에서 고객의 연락처 정보를 찾기 위해 사용된다.

- product(p.ID, type, manufacturer, price)



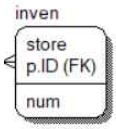
: product는 제품 정보를 나타내는 relation으로, 각 제품마다 unique한 하나의 p.ID를 갖는다. 따라서 p.ID는 product의 primary key이고, 그 외 attribute로는 상품의 종류를 나타내는 type, 제조사를 나타내는 manufacturer, 가격을 나타내는 price가 있다.

- ship(tr.num, ship.com, c.ID, p.ID, ord.date, pro.date, arr.date)



: ship은 고객의 상품 주문 정보를 나타내는 relation으로, 각 주문마다 unique한 하나의 tr.num을 갖는다. 따라서 tr.num은 ship의 primary key이고, 그 외에 몇 개의 attribute들이 있다. ship.com(오프라인 주문은 주문 매장을 표시)은 배송회사, c.ID는 주문 고객의 c.ID(customer의 c.ID), p.ID는 주문 제품의 p.ID(product의 p.ID), ord.date는 주문날짜, pro.date는 도착 약속 날짜, arr.date는 실제 도착 날짜를 의미하는 attribute이다.
ord.date는 (TYPE 4, 4-1, 4-2)에서 작년에 판매된 제품을 찾기 위해 사용된다.
pro.date, arr.date는 (TYPE 6)에서 약속 시간 내에 배달되지 않은 경우를 찾기 위해 사용된다.

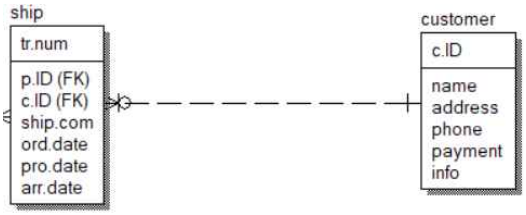
- inven(store, p.ID, num)



: inven은 각 지점별 각 제품의 재고 수를 나타내는 relation이다. 각 지점, 각 제품마다 재고 수를 표시하므로 primary key는 store, pID이다. 그 외 attribute로는 재고 수를 나타내는 num이 있다.
store와 p.ID는 (TYPE 5)에서 사용된다.

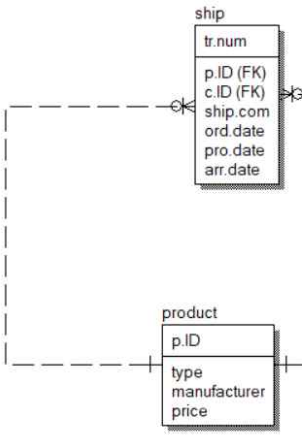
2) mapping cardinality

- customer(one)-shipping(many)



: 한 사람이 여러 개의 주문을 할 수 있고, 각 주문은 한 명의 고객만 해당되므로 one-to-many relation이다. 또한, shipping의 속성인 c.ID가 customer의 primary key에 해당하므로 foreign key가 된다.

- product(one)-shipping(many)



: 한 제품이 여러 번의 주문될 수 있고, 각 주문은 하나의 제품만 해당되므로 one-to-many relation이다. 또한, shipping의 속성인 p.ID가 product의 primary key에 해당하므로 foreign key가 된다.

- product(one)-inven(many)

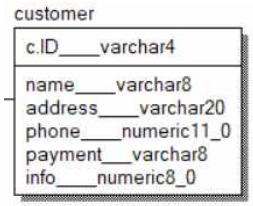
: 각 제품은 여러 매장에 재고가 있을 수 있고, 각 매장에서 제품의 재고는 제품 별로 관리되므로 one-to-many relation이다. 또한, inven의 primary key인 p.ID가 product의 primary key에 해당하므로 foreign key가 된다.

2. physical schema

1) relation, 2) mapping cardinality

logical schema와 동일하다.

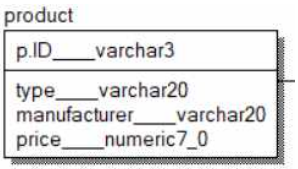
3) domain



- customer(c.ID, name, address, phone, payment, info)

c.ID varchar(4) : 각 tuple마다 unique한 4자리 수로 나타내기 위해 최대 4글자가 가능한 character string으로 설정하였다. primary key이므로 null값은 허용되지 않는다.
name varchar(8) : 고객의 이름은 최대 8글자까지 가능하도록 하였다.
address varchar(20) : 고객의 주소는 최대 20글자까지 가능하도록 하였다.
phone numeric(11,0) : 고객 전화번호는 자연수이므로 소수점이 없는 11자리 숫자로 설정하였다.
payment varchar(8) : 지불방식은 최대 8글자까지 가능하도록 하였다. (account, card 등이 있다.)
info numeric(8,0) : 계좌번호 또는 카드번호는 8자리 자연수로 설정하였다.

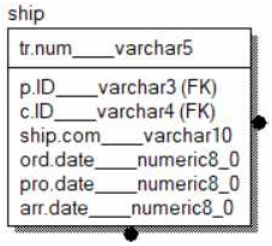
- product(p.ID, type, manufacturer, price)



p.ID varchar(3) : 각 tuple마다 unique한 3자리 수로 나타내기 위해 최대 3글자가 가능한 character string으로 설정하였다. primary key이므로 null값은 허용되지 않는다.
type varchar(20) : 제품 종류는 최대 20글자까지 가능하도록 하였다.
manufacturer varchar(20) : 제품의 제조사명은 최대 20글자까지 가능하도록 하였다.

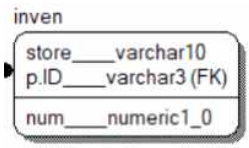
price numeric(5,0) : 제품의 가격은 5자리 자연수로 나타내었고, 단위는 원이다.

- ship(tr.num, ship.com, c.ID, p.ID, ord.date, pro.date, arr.date)



tr.num varchar(5) : 각 tuple마다 unique한 5자리 수로 나타내기 위해 최대 5글자가 가능한 character string으로 설정하였다. primary key이므로 null값은 허용되지 않는다.
ship.com varchar(10) : 배송회사 또는 지점명은 최대 20자리까지 가능하도록 하였다.
c.ID varchar(4) : customer의 c.ID와 마찬가지로 최대 4글자가 가능한 character string으로 설정하였다.
p.ID varchar(3) : prudoct의 p.ID와 마찬가지로 최대 3글자가 가능한 character string으로 설정하였다.
ord.date numeric(8,0) : 날짜를 yyyymmdd로 표현하기 위해 8자리 자연수로 설정하였다.
pro.date numeric(8,0) : 날짜를 yyyymmdd로 표현하기 위해 8자리 자연수로 설정하였다.
arr.date numeric(8,0) : 날짜를 yyyymmdd로 표현하기 위해 8자리 자연수로 설정하였다.

- inven(store, p.ID, num)



store varchar(10) : 지점명은 최대 20자리까지 가능하도록 하였다. primary key이므로 null값은 허용되지 않는다.
p.ID varchar(3) : prudoct의 p.ID와 마찬가지로 최대 3글자가 가능한 character string으로 설정하였다. primary key이므로 null값은 허용되지 않는다.
num numeric(1,0) : 제품의 재고 수는 한자리 자연수로 설정하였다.

3. BNCF & simplified test

- customer(c.ID, name, address, phone, payment, info)
c.ID=A, name=B, address=C, phone=D, payment=E, info=G라 하자.
그러면 R=(A,B,C,D,E,G), F={A->B, A->C, A->D, A->E, A->G}이 되고, union rule에 의해서 F={A->BCDEG}이 된다.
이제 simplified test에 의해 A->BCDEG가 BCNF를 violation하지 않는다는 것만 보이면 된다.
그런데, A는 A->R이므로 superkey이고, 따라서 BCNF를 만족한다.
따라서 R은 BCNF이다.

또한, A는 smallest superkey이므로 candidate key가 된다.

- product(p.ID, type, manufacturer, price)
p.ID=A, type=B, manufacturer=C, price=D라 하자.
그러면 R=(A,B,C,D), F={A->B, A->C, A->D}이 되고, union rule에 의해서 F={A->BCD}이 된다.
이제 simplified test에 의해 A->BCD가 BCNF를 violation하지 않는다는 것만 보이면 된다.
그런데, A는 A->R이므로 superkey이고, 따라서 BCNF를 만족한다.
따라서 R은 BCNF이다.
또한, A는 smallest superkey이므로 candidate key가 된다.

- ship(tr.num, ship.com, c.ID, p.ID, ord.date, pro.date, arr.date)
tr.num=A, ship.com=B, c.ID=C, p.ID=D, ord.date=E, prodate=G, arr.date=H라 하자.
그러면 R=(A,B,C,D,E,G,H), F={A->B, A->C, A->D, A->E, A->G, A->H}이 되고, union rule에 의해서 F={A->BCDEGH}이 된다.
이제 simplified test에 의해 A->BCDEGH가 BCNF를 violation하지 않는다는 것만 보이면 된다.
그런데, A는 A->R이므로 superkey이고, 따라서 BCNF를 만족한다.
따라서 R은 BCNF이다.
또한, A는 smallest superkey이므로 candidate key가 된다.

- inven(store, p.ID, num)
store=A, p.ID=B, num=C라 하자.
그러면 R=(A,B,C), F={AB->C}이 되고, simplified test에 의해 AB->C가 BCNF를 violation하지 않는다는 것만 보이면 된다.
그런데, AB는 AB->R이므로 superkey이고, 따라서 BCNF를 만족한다.
따라서 R은 BCNF이다.
또한, AB의 subset 중에 superkey가 없으므로 AB는 candidate key이다.
(∵A->C이면 =A,C⊉R, B->C이면 B^+=B,C⊉R)

4. ODBC C code

[TYPE 3]

```
case 3:
    printf("₩n₩n");
    printf("** Find all products sold in the past year. **₩n");
    mysql_query(connection, "select distinct pID from ship where orddate>20210100 and orddate<20211232");
    sql_result = mysql_use_result(connection);
    num_fields = mysql_num_fields(sql_result);
    while ((sql_row = mysql_fetch_row(sql_result)))
    {
        for (int i = 0; i < num_fields; i++)
        {
            printf("%s ", sql_row[i]);
        }
        printf(" ");
    }
```

주문정보를 나타내는 ship relation에서 주문 날짜(orddate)가 2021년도인, 즉 orddate 값이

20210100보다 크고 20211232보다 작은 tuple의 상품명(pID)를 distinct하게 꺼낸다.

결과>

```
** Find all products sold in the past year. **
121  122  142  151
```

[TYPE 5]

```
case 5:
    printf("₩n₩n");
    printf("---- TYPE 5 ----₩n₩n");
    printf("** Find those products that are out of stock at every store in California. **₩n");
    mysql_query(connection, "select pID from inven where store='California' and num=0");
    sql_result = mysql_use_result(connection);
    num_fields = mysql_num_fields(sql_result);
    while ((sql_row = mysql_fetch_row(sql_result)))
    {
        for (int i = 0; i < num_fields; i++)
        {
            printf("%s ", sql_row[i]);
        }
        printf(" ");
    }
    break;
```

query문은 “select pID from inven where store='California' and num=0” 이다.
지점명(store)과 제품의 ID(pID)를 primary key로 가지며, 지점별 제품별 재고 수를 num이라는 속성에 담고 있는 inven이라는 relation에서 지점명이 캘리포니아이고 재고 수는 0인 제품의 pID를 꺼낸다.

결과>

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

** Find those products that are out of stock at every store in California. **
121  151  154
```

[TYPE 6]

```
case 6:
    printf("₩n₩n");
    printf("---- TYPE 6 ----₩n₩n");
    printf("** Find those packages that were not delivered within the promised time. **₩n");
    mysql_query(connection, "select trnum from ship where prodate<arrdate" );
    sql_result = mysql_use_result(connection);
    num_fields = mysql_num_fields(sql_result);
    while ((sql_row = mysql_fetch_row(sql_result)))
    {
        for (int i = 0; i < num_fields; i++)
        {
            printf("%s ", sql_row[i]);
        }
        printf(" ");
    }
    break;
```

주문정보를 나타내는 ship relation에서 약속시간(prodate)보다 실제 도착시간(arrdate)이 더 늦은, 즉 prodate<arrdate인 tuple의 주문번호를 꺼낸다.

결과>

```
** Find those packages that were not delivered within the promised time. **
10004  20004
```