<Project2>

20150614 어대영

1. BCNF

-Customer

Customer_ID->monthly_account

-Bill

Package_ID->Customer_ID, address, amount, charged_type, date(배송완료 날짜), paid

-Recipient

Package_ID->Customer_ID, address

-Package

Package_ID->type, weight, timeliness, matter

-Time

Package_ID->promised_time, punctual

-Transport

Package_ID, Datetime->type(transport type), name, from, to, delivered

-Place

Package_ID, Datetime->type, name, left

모든 dependency의 왼쪽 속성이 Primary key이므로 BCNF를 만족한다. 하지만 정보의 중복과 이름의 모호성 때문에 수정을 했다.

먼저 정보의 중복은 Bill과 Recipient에서 address 정보가 중복되는 것이다. 그런데 Customer의 주소는 변할 수 있기 때문에 과거에 배송 주소를 따로 기억할 필요가 있다. 따라서 Bill과 Recipient 모두 address 정보를 저장해야 한다. 결국 중복된 저장인 것처럼 보이지만 두 address 모두 필수로 기억해야 한다. 추가로 Customer마다 현재 주소를 기억해야 할 필요도 있으므로 Customer relation에도 address를 추가했다.

이름의 모호성은 Package, Transport, Place의 type에서 나타난다. dependency를 확인할 때 모호할 수 있으므로 Package의 type은 package_type, Transport의 type은 transport_type, Place의 type은 place_type으로 수정했다.

+) query 해결을 위해 customer, package에 name을 포함시켰다.

추가된 dependency

-Customer: Customer_ID->name, address

-Package: Package_ID->name

BCNF만족한다.

+) MySQL에 to, from, leftrk attribute 명으로 들어가지 않기 때문에 to_place, from_place, left_place로 변경했다.

test: 모든 relation에 대해 a->b에서 a는 primary key이고 b는 나머지 attribute들이다. 따라서 a+는 모든 relation의 속성들을 포함하므로 모두 BCNF를 만족한다.

2. Physical Schema Diagram

위의 내용을 수정해서 physical schema를 수정했다. 다음은 각 relation 속성들의 data type, domain, constraints이다. simple함을 위해서 모든 속성은 null 값을 갖지 않도록 했다. constraint가 없다면 '-'로 표시했다. constraint가 있는 경우 이름은 속성명과 같게 했으므로 따로 나타내지 않고 조건을 명시했다.

-Customer

Customer_ID: Integer, Number, -

address: String, Varchar, -

name: String, Varchar, -

-Bill

amount: Integer, Number, -

charged_type: String, Varchar, -

date: Integer, Number, (20000000,21000000)

paid: Integer, Number, -

address: String, Varchar, -

*paid: 지불되지 않았으면 0, 지불되었으면 1 값을 갖는다.

*date: 2000년 00월 00일 -> 20000000

-Recipient

address: String, Varchar, -

-Package

Package_ID: Integer, Number, -

package_type: String, Varchar, (flat envelope, small box, large box)

weight: Integer, Number, -

timeliness: String, Varchar, (overnight, second day, longer)

matter: String, Varchar, (hazardous, international, none)

name: String, Varchar, -

-Time

promised_time: Date, Datetime, (20000000 <= x <= 21000000)

punctual: Integer, Number, -

* punctual: 배송 예정 시간 안에 배송되었다면 1, 늦어졌다면 0 값을 갖는다.

-Transport

Datetime: Integer, Number, (200000000000<= x <=210000000000)

transport_type: String, Varchar, (plane, bus, train, truck)

name: Integer, Number, -

from_place: String, Varchar, -

to_place: String, Varchar, -

delivered: Integer, Number, -

*delivered: to의 배송지까지 배송완료는 1, 배송전은 0의 값을 갖는다.

*datetime: 2000년 00월 00일 00시 00분->200000000000

-Place

Datetime: Integer, Number, (200000000000<= x <=210000000000)

place_type: String, Varchar, -

name: Integer, Number, -

left_place: Integer, Number, -

3. Queries

1-1)

```
create table crushed as select Package_ID
from Transport where transport_type = 'truck' and name = '1721' and delivered = 0;
create table if not exists crushed_customer_id as select distinct B.Customer_ID
from Bill B where B.Package_ID in (select c.Package_ID from crushed as c);
select name from crushed_customer_id natural join Customer;
drop table crushed; drop table crushed_customer_id;
```

Transport에서 truck 1721로 배송했고 아직 배송되지 않은 상품(delivered=0)의 id를 가져와서 테이블로 만든다.

Bill을 활용해서 가져온 상품 id에 맞는 고객 id를 가져온다.

Customer를 활용해서 고객 id에 맞는 이름을 가져온다.

1-2)

```
create table crushed as select Package_ID from Transport where transport_type='truck' and name='1721' and delivered=0;
select name from Recipient R natural join Package P where Package_ID in (select c.Package_ID from crushed as c);
drop table crushed;
```

Transport에서 truck 1721로 배송했고 아직 배송되지 않은 상품(delivered=0)의 id를 가져와서 테이블로 만든다.

Recipient와 Package의 natural join(Package_id)을 활용해서 가져온 상품 id에 해당하는 수령인 이름을 가져온다.

1-3)

create table crushed as select package_id, Datetime from Transport where transport_type='truck' and name='1721' and delivered=1;

create table crushed_package select P1.package_id from Package P1 join crushed C where P1.package_id = c.package_id and Datetime=(select max(C1.Datetime) from crushed C1);

select P.package_id, package_type, weight, timeliness, matter from Package P join crushed_package c where P.package_id=C.package_id;

drop table crushed, crushed_package;

Transport에서 truck 1721로 배송했고 배송된 상품(delivered=1)의 id, datetime(배송시간)을 가져와서 테이블로 만든다.

배송시간이 가장 큰 entity의 상품id를 가져온다.

상품 id로 상품의 정보를 가져온다.

* datetime(배송 시간)은 트럭,버스,비행기로 운송을 시작한 시간으로 운송을 완료한 것 중에 datetime이 가장 큰 것이 가장 최근에 운송된 상품이다.

2) *past year = 2019

create table delivery as select Customer_ID, count(*) num from Bill where date between 20190000 and 20200000 group by Customer_ID;

select name from delivery D1 join Customer C on D1.Customer_ID=C.Customer_ID where num=(select max(D2.num) from delivery D2);

drop table delivery;

Bill을 Customer_id로 group by한 뒤 특정 연도에 한정해서 개수를 센다.

Customer와 join해서 count개수가 가장 많은 customer의 정보를 가져온다.

3)

create table delivery as select Customer_ID, sum(amount) pay from Bill where date between 20190000 and 20200000 group by Customer_ID;

select name from delivery D1 join Customer C on D1.Customer_ID=C.Customer_ID where pay=(select max(D2.pay) from delivery D2);

drop table delivery;

Bill을 Customer_id로 group by한 뒤 특정 연도에 한정해서 amount의 합을 구한다.

Customer와 join해서 합이 가장 큰 customer의 정보를 가져온다.

4)

select Package_ID

from Time

where punctual='False'

5)

*past month=05, customer name='KANGDM'

create table last_month_bill as select C.customer_id, C.address, B.amount, B.package_id, B.charged_type

from Bill B join Customer C

where B.customer_id=C.customer_id and C.name='KANGDM' and B.date between 20200500 and 20200600;

select customer_id, address, SUM(amount)

from last_month_bill;

select package_id, name, amount, package_type, charged_type, timeliness

from last_month_bill B1 natural join Package;

drop table last_month_bill;

Bill과 Customer의 join으로 특정 이름을 갖고 특정 월에 배송을 한 정보를 가져온다.

(Bill은 월 정보 활용, Customer는 name정보 활용)

amount의 합을 계산, Package와 join을 통해 package정보 가져옴

C code에서 query는 table생성, result도출, table삭제 순서로 작성했다. 입력을 받는 경우는 문자열을 합치는 방법으로 query를 완성했는데 숫자는 그냥 합쳤지만 문자열은 ' '를 사용해서 합쳤다.