Project 2

BCNF Decomposition & ODBC implementation

20180071 오서영

1. 회사의 정보 및 데이터 관련

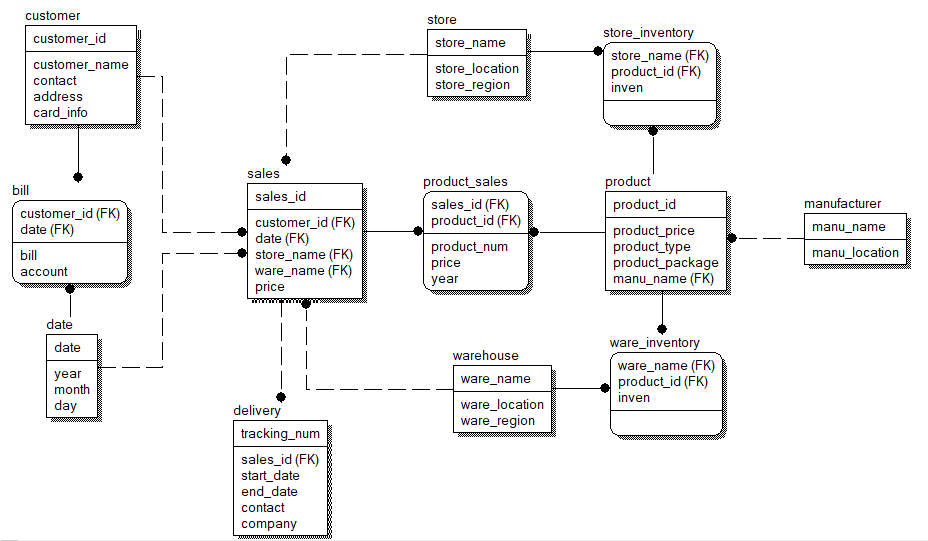
배송: 당일배송 원칙. 만약 출발일과 도착일이 같지 않다면 예정일에 도착하지 못했음을 의미한다. 트래킹 번호는 고유한 값으로 중복되지 않는다.

영수증: 회사와 계약을 맺은 고객에 한해 전달에 사용한 요금은 다음달 1일에 청구된다.

자료 수집 시기: 2021-01-01~2022-02-04

현재 날짜: 2022-02-04

위를 바탕으로 CRUD와 c 프로그램 내의 쿼리문 작성하였습니다.



2. Physical Schema & ODBC implementation

**manufacturer**

manufacturer는 제조사 관련 정보가 담겨 있는 테이블입니다. 속성으로는 manu\_name(제조사 이름), manu\_location(제조사 위치)를 가집니다. manu\_name은 California 01, California 02와 같이 지역명+순번으로 만들어진 속성입니다. 따라서 중복된 값을 가질 수 없고 manu\_location을 결정하므로 primary key가 됩니다.

manufacturer의 functional dependency를 확인해 보면,

-manu\_name->manu\_location

이때 manu\_name이 superkey가 되므로 BCNF의 조건을 만족하는 것을 알 수 있습니다.

**product**

product는 제품 관련 정보가 담겨 있는 테이블입니다. 속성으로는 product\_id(제품 번호), product\_price(제품 가격), product\_type(제품 타입. 선풍기, 냉장고 등), product\_package(회사에서 임의로 묶은 패키징), manu\_name(제조사의 이름.)을 가집니다. product\_id에서 동일한 제품의 품번이 여러 번 발생할 수 없고, product\_id는 튜플을 식별하는 데 사용됩니다. 따라서 product\_id가 product의 primary key가 됩니다. 이때 manu\_name은 manufacturer로부터 가져온 외래키입니다.

product의 functional dependency를 확인해보면,

1. product\_id-> product\_price, product\_type, product\_package, manu\_name

이때 product\_id가 superkey가 되므로 BCNF의 조건을 충족합니다.

**store**

store는 점포 관련 정보가 담겨 있는 테이블입니다. 속성으로는 store\_name(점포 이름), store\_location(점포 위치), store\_region(점포 지역)이 있습니다. store 테이블은 다른 점포라면 동일한 이름을 짓지 못하는 것이 원칙입니다. 따라서 store\_name은 중복된 값이 없고 튜플을 식별하는 데 사용되므로, store의 primary key가 됩니다.

store의 functional dependency를 살펴보면,

1. store\_name->store\_location, store\_region

이때 store\_name이 superkey가 되므로 BCNF의 조건을 충족합니다.

**warehouse**

warehouse는 창고 관련 정보가 담겨 있는 테이블입니다. 속성으로는 ware\_name(창고 이름), ware\_location(창고 위치), ware\_region(창고 지역)이 있습니다. warehouse 테이블은 다른 창고라면 동일한 이름을 짓지 못하는 것이 원칙입니다. 따라서 ware\_name은 중복된 값이 없고 튜플을 식별하는 데 사용되므로, warehouse의 primary key가 됩니다.

warehouse의 functional dependency를 살펴보면,

1. ware\_name->ware\_location, ware\_region

이때 ware\_name이 superkey가 되므로 BCNF의 조건을 충족합니다.

**store\_inventory**

store\_inventory는 점포의 재고 관련 정보가 담겨 있는 테이블입니다. 속성으로는 store\_name(점포 이름), product\_id(제품 품번), inven(현재 재고)를 가집니다. store\_inventory는 추가로 재고가 늘어나거나 줄어들 때마다 튜플을 추가하지 않으며, 원래의 튜플에 재고 수량만 업데이트 되는 형식입니다. 따라서 store\_name, product\_id, inven 셋 모두가 있어야 하나의 튜플을 식별할 수 있으므로 store\_name, product\_id, inven이 primary key가 됩니다.

store\_inventory의 functional dependency를 살펴보면

1. store\_name, product\_id, inven -> store\_name, product\_id, inven

이때 store\_name, product\_id, inven이 superkey가 되므로 BCNF의 조건을 충족합니다.

**ware\_inventory**

ware\_inventory는 창고의 재고 관련 정보가 담겨 있는 테이블입니다. 속성으로는 ware\_name(창고 이름), product\_id(제품 품번), inven(현재 재고)를 가집니다. ware\_inventory는 추가로 재고가 늘어나거나 줄어들 때마다 튜플을 추가하지 않으며, 원래의 튜플에 재고 수량만 업데이트 되는 형식입니다. 따라서 ware\_name, product\_id, inven 셋 모두가 있어야 하나의 튜플을 식별할 수 있으므로 ware\_name, product\_id, inven이 primary key가 됩니다.

ware\_inventory의 functional dependency를 살펴보면

1. ware\_name, product\_id, inven -> ware\_name, product\_id, inven

이때 ware\_name, product\_id, inven이 superkey가 되므로 BCNF의 조건을 충족합니다.

**customer**

customer은 고객 관련 정보를 담고 있는 테이블입니다. 속성으로 customer\_id(고객에게 부여된 임의의 값), customer\_name(고객 이름), contact(고객의 연락처), address(고객의 주소), card\_info(카드번호)를 가집니다. customer 테이블에는 온오프라인 고객이 모두 저장되어 있으며, 온라인 고객의 경우 card\_info가 저장되어 있고 오프라인 고객의 경우 card\_info가 null값으로 비워져 있습니다. customer\_id는 임의로 부여된 값이므로 중복되지 않고 튜플을 식별하는 데 활용되므로 primary key입니다.

customer의 functional dependency를 살펴보면,

1. customer\_id->customer\_name, contact, address, card\_info

이때 customer\_id가 superkey가 되므로 BCNF의 조건을 충족합니다.

**sales**

sales는 판매 관련 정보를 담고 있는 테이블입니다. 속성으로는 sales\_id(판매 시점에 임의로 부여된 값), customer\_id(고객에게 부여된 임의의 값), date(구매 날짜), store\_name(온라인 고객의 구매 점포 이름), ware\_name(온라인 고객의 구매 창고 이름), price(총 구매가격)를 가집니다. 이때 제품의 판매 수량이나 제품별 총 판매 가격과 같은 정보는 저장하지 않았으며 product\_sales라는 테이블을 따로 만들어 해결하였습니다.

sales 테이블에는 세 가지 경우의 판매정보가 모두 포함됩니다. 첫째, 온라인 판매(customer\_id not null, store\_name null, ware\_name not null). 이 판매 케이스에서는 customer\_id가 자동으로 저장되지만 store\_name은 없고 물품이 출고된 ware\_name은 저장됩니다. 둘째, 오프라인 판매인데 고객 정보가 없는 경우(customer\_id null, store\_name not null, ware\_name null). 이 판매 케이스에서는 고객이 정보를 저장하지 않고 물건을 사가기만 한 경우를 의미하는데, customer\_id는 없지만 물건을 판매한 store\_name은 있고 ware\_name은 없습니다. 셋째, 오프라인 판매이고 고객 정보가 있는 경우(customer\_id not null, store\_name not null, ware\_name null). 이 판매 케이스에서는 고객이 정보를 저장하고 물건도 사간 경우를 의미합니다. 이 경우 customer\_id와 store\_name이 모두 저장되고 ware\_name은 비워져 있습니다.

하지만 이 세 가지 경우 모두 sales\_id를 통해 튜플이 식별될 수 있습니다. 따라서 sales\_id가 sales 테이블의 primary key가 됩니다.

sales의 functional dependency를 살펴보면,

1. sales\_id->customer\_id, date, store\_name, ware\_name, price

이때 sales\_id는 superkey이므로 BCNF를 충족합니다. 원래 sales\_id->customer\_id, date, year, month, day, store\_name, ware\_name, price으로 설계하였으나, date->year, month, day라는 추가적인 functional dependency의 발생으로 date->year, month, day를 따로 분해하였습니다. (alpha: date, beta: year, month, day, R-(beta-alpha): sales\_id, customer\_id, store\_name, ware\_name, price)

**product\_sales**

product\_sales는 제품의 판매 정보를 담고 있는 테이블입니다. 속성으로는 sales\_id(판매 시점에 임의로 부여된 값. sales의 외래키), product\_id(제품 품번. product의 외래키), product\_num(제품의 개수), price(제품별 총 판매가격), year(구매년도)를 가집니다. 이때 product\_num은 sales\_id별 product의 개수이며, price는 sales\_id별 product의 총 가격입니다. 따라서 sales의 price와 product\_sales의 price는 다른 값이며 서로의 외래키가 되지 않습니다.

하나의 sales\_id 내에서도 product\_id에 따라 여러 개의 튜플이 존재할 수 있습니다. 따라서 sales\_id와 product\_id 둘이 합쳐져야 고유성을 만족하며 product\_sales의 튜플을 식별할 수 있으므로, sales\_id와 product\_id가 primary key가 됩니다.

product\_sales의 functional dependency를 살펴보면,

1. sales\_id, product\_id -> product\_num, price, year

이때 sales\_id, product\_id가 superkey가 되므로 BCNF를 만족합니다.

**date**

date는 날짜 관련 정보를 담고 있는 테이블입니다. 속성으로는 date(yy-mm-dd로 구성된 date 값), year(년도), month(월), day(일)를 가집니다. date는 중복된 값이 존재할 수 없으며 모든 튜플을 식별하는 데 사용될 수 있으므로 primary key입니다.

date의 functional dependency를 살펴보면

1. date->year, month, day

이때 date가 superkey가 되므로 BCNF를 만족합니다.

**delivery**

delivery는 배송 관련 정보를 담고 있는 테이블입니다. 속성으로는 tracking\_num(트래킹 번호), sales\_id(판매 시점에 임의로 부여된 값. sales의 외래키), start\_date(배송 시작일), end\_date(배송 도착일), contact(고객 연락처), company(배송 회사명)을 가집니다. 만일 배송이 누락되거나 분실될 경우 판매 정보를 찾아 재발송해야 하기 때문에 sales\_id도 저장하였습니다. 또한 tracking\_num은 중복되지 않는다는 전제이고 모든 튜플을 식별할 수 있는 키이므로 primary key가 됩니다.

delivery의 functional dependency를 살펴보면,

1. tracking\_num ->sales\_id, start\_date, end\_date, contact

tracking\_num이 superkey가 되므로 BCNF를 만족합니다.

**bill**

bill은 영수증 관련 정보를 담고 있는 테이블입니다. 영수증은 회사와 계약을 맺은 고객에 한해 전달의 청구요금을 다음달 1일 고객의 계좌로 청구한다는 전제입니다. 속성으로는 customer\_id(고객에게 임의로 부여된 값. customer의 외래키), date(영수증 청구 날짜. 다음달 1일. date의 외래키.), bill(영수증 청구 금액), account(고객 계좌)를 가집니다. 이때 중간에 고객이 account를 변경하는 일이 있을 수도 있으므로 account는 superkey에 포함될 수 없습니다. 대신 customer\_id와 date를 합하면 특정 달 1일의 고객 영수증이 조회되므로, 모든 튜플을 식별할 수 있습니다. 따라서 customer\_id와 date가 primary key가 됩니다.

bill의 BCNF를 살펴보면,

1. customer\_id, date->bill, account

customer\_id와 date가 superkey가 되므로 BCNF를 만족합니다.