DATABASE SYSTEM

Project2. BCNF Decomposition, Physical Schema Diagram and Queries

Electronics Vendor company

CSE4110-01

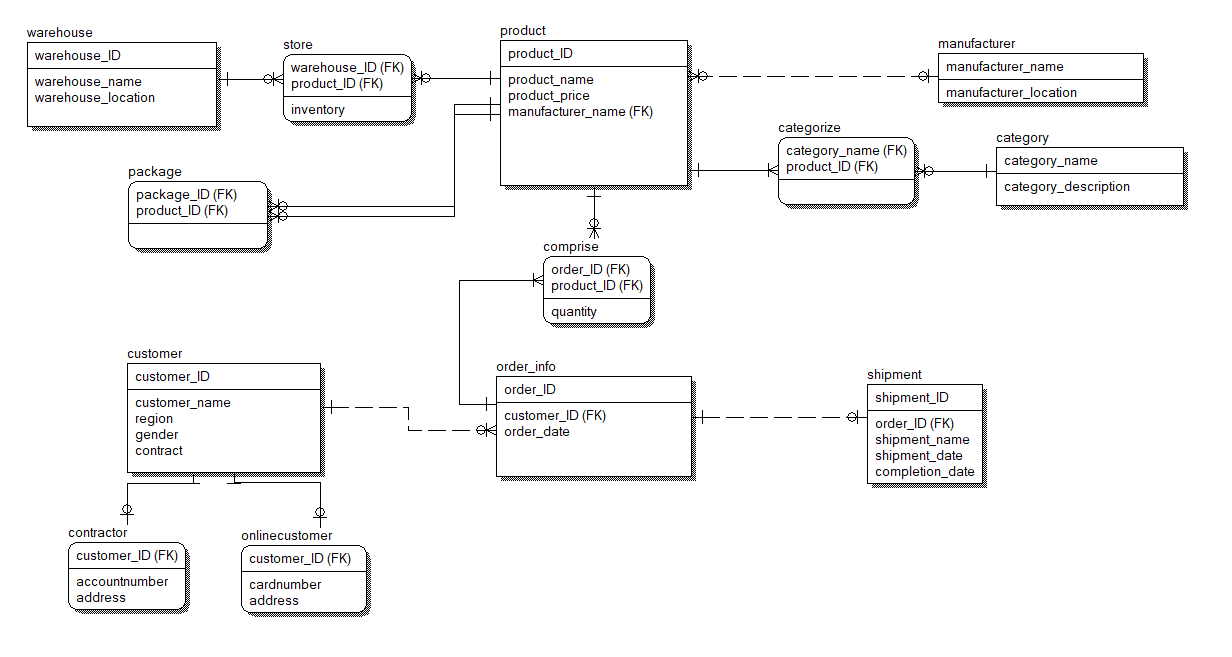
20160530 박상엽

**1. 프로젝트 개요**

가상의 Electronics Vendor company의 DBA라고 가정한다. 이 회사에서는 회사 운영의 기초가 되는 데이터베이스의 주요 부분을 재설하기로 결정했고, 설계 제안을 요청하는 관리자는 컴퓨터에 대해 잘 알지 못하고, 기술 수준에서 매우 상세한 사양을 제공할 수 없다. 때문에 요구사항을 분석하고 주어진 질의에 적절하게 대응할 수 있도록 데이터베이스 설계 제안서를 작성해야 한다.

**2. BCNF Decomposition**

**2.1 Decomposed Logical Schema Diagram**



* 1. **Description**

1. Product {product\_ID -> product\_name, product\_price, manufacturer\_name}

상품의 ID가 같으면 상품명, 상품가격, 제조사에 대한 값도 동일하므로 종속성이 존재한다. Product\_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

1. Warehouse {warehouse\_ID -> warehouse\_name, warehouse\_loaction}

창고의 ID가 같으면, 창고의 이름과 위치가 동일하여 함수적 종속성이 있다. Warehouse\_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

1. Store {warehouse\_ID, product\_ID -> inventory}

창고의 ID와 상품의 ID가 동일하면, 상품의 개수도 동일하다. 따라서 함수적 종속성이 존재하며 warehouse\_ID와 product\_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

1. Manufacturer {manufacturer\_name -> manufacturer\_location}

제조사가 같으면 제조사의 위치가 같으므로 함수적 종속성이 존재한다. 따라서manufacturer\_name이 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

1. Category { category\_name -> category\_description}

카테고리명이 같으면 그에 따른 설명도 동일하므로 함수적 종속성이 존재한다. Category\_name이 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

1. order\_info{ order\_ID -> customer\_ID, order\_date}

주문 ID가 같으면, 고객의 ID와 주문일이 동일하다. 따라서 함수적 종속성이 존재한다. Order\_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

1. customer{customer\_ID -> contract, name, gender, region}

고객의 ID가 같으면, 연락처, 이름, 성별, 지역이 동일하다. 따라서 함수적 종속성이 존재한다. Customer\_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

1. contractor {customer\_ID -> account, address}

specialization을 통해 customer\_ID를 받아왔다. 고객의 ID가 같으면, 계좌정보와 주소가 동일하다. 따라서 함수적 종속성이 존재하고, customer\_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

1. onlinecustomer {customer\_ID -> cardnumber ,address}

specialization을 통해 customer\_ID를 받아왔다. 고객의 ID가 같으면, 계좌정보와 주소가 동일하다. 따라서 함수적 종속성이 존재하고, customer\_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

1. Categorize

다대다 관계에 의해 생성된 테이블로 category\_name과 product\_name이 PK이므로 super key이다. 따라서 BCNF를 만족한다.

1. package

다대다 관계에 의해 생성된 테이블로 package\_ID와 product\_ID가 PK이므로 super key이다. 따라서 BCNF를 만족한다.

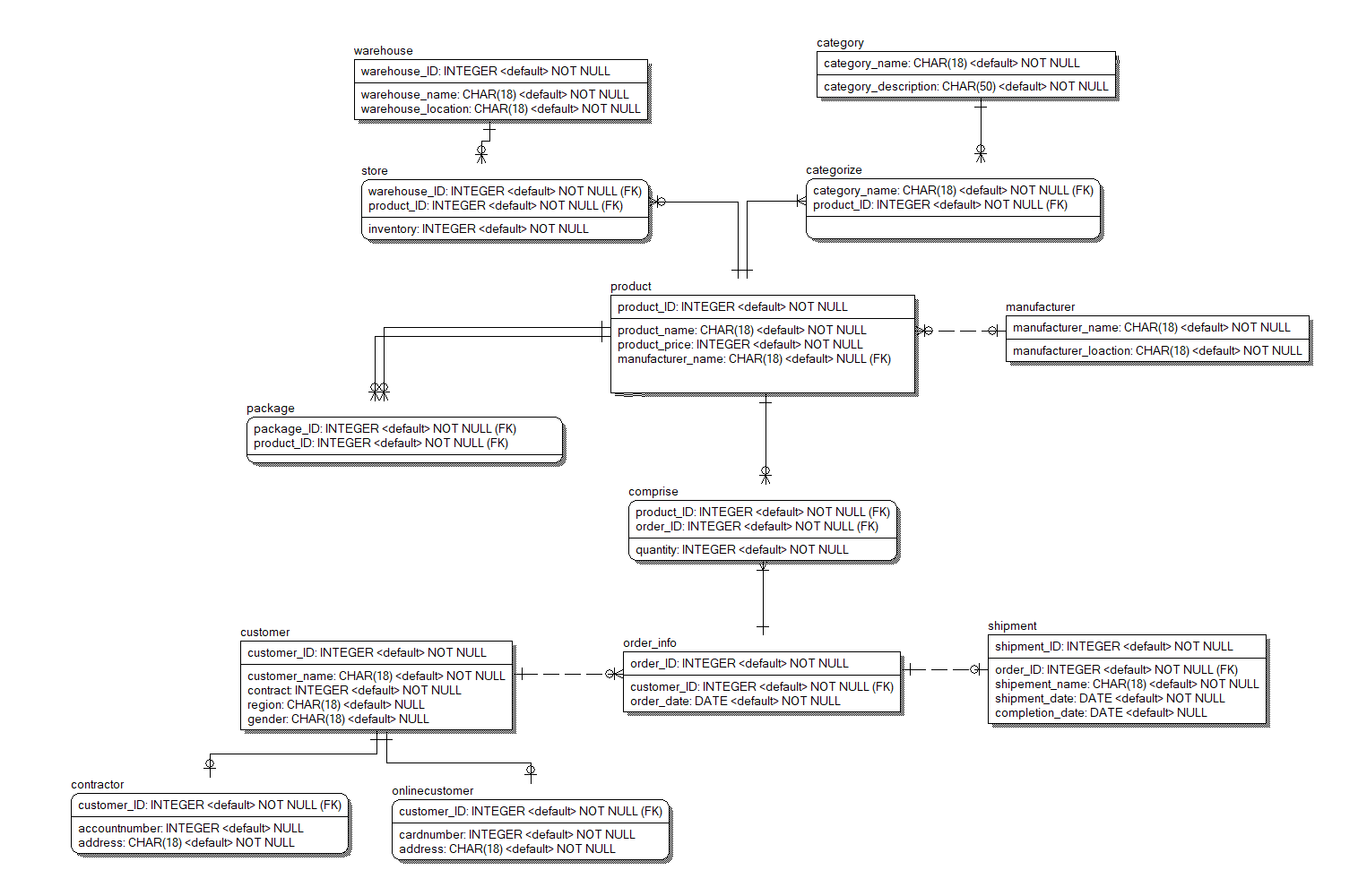
1. comprise {order\_ID, product\_ID -> quantity}

다대다 관계에 의해 생성된 테이블로 order\_ID, product\_ID가 PK 역할을 하며, 둘의 값이 동일하면 수량 또한 동일하여 함수적 종속성이 존재한다. 따라서 BCNF를 만족한다.

1. Shipment {shipment\_ID -> order\_ID, shipment\_name, order\_date, completion\_date}

**3. Physical Scheme diagram**

**3.1 Decomposed Logical Schema Diagram**



**3.2 Description**

|  |  |  |
| --- | --- | --- |
| Entity | Attribute | Description |
| product | product\_ID | 정수형이며 PK이므로 not null이다. |
| product\_name | 문자형이며 not null이다. |
| manufacturer\_name | Manufacturer을 참조하여 FK이며 따라서 문자형이다. 패키지 상품의 경우 제조사가 구분되지 않기 때문에 null을 허용한다. |
| package | package\_ID | product 참조, 다대다 관계에 의한 형성된 PK이자 FK이다. 따라서 product\_ID의 정수형을 가져오며 not null이다. |
| product\_ID |
| manufacturer | manufacturer\_name | 문자형이며 PK이므로 not null이다. |
| manufacturer\_location | 문자형이며 not null이다. |
| warehouse | warehouse\_ID | 정수형이며 PK이므로 not null이다. |
| warehouse\_name | 문자형이며 not null이다. |
| warehouse\_location | 문자형이며 not null이다. |
| store | warehouse\_ID | Warehouse를 참조하며 PK이자 FK이다. 따라서 정수형이며 not null이다. |
| product\_ID | product 참조하며 PK이자 FK이다. 따라서 정수형이며 not null이다. |
| Inventory | 상품의 수량을 측정한다. 정수형이며 not null이다. |
| category | category\_name | 문자형이며 PK이므로 not null이다. |
| category\_description | 문자형이며 not null이다. |
| categorize | category\_name | category PK이자 FK이다. 따라서 문자형이며 not null이다. |
| product\_ID | product 참조하며 PK이자 FK이다. 따라서 정수형이며 not null이다. |
| order\_info | order\_ID | 정수형이며 PK이므로 not null이다. |
| customer\_ID | Customer을 참조하며 FK이다. 따라서 정수형이며 not null이다. |
| order\_date | date형이며 not null이다. |
| shipment | shipment\_ID | 정수형이며 PK이므로 not null이다. |
| order\_ID | Order\_info를 참조하며 FK이다. 따라서 정수형이며 not null이다. |
| shipment\_name | 정수형이고 not null이다. |
| shipment\_date | date형이며 not null이다. |
| completion\_date | date형이며 배송 완료가 된 이후에 업데이트하므로 null을 허용한다. |
| customer | customer\_ID | 정수형이며 PK이므로 not null이다. |
| contact | 정수형이며 not null이다. |
| customer\_name | 고객의 이름으로 문자형이며 not null이다. |
| region | 문자형이며 필수 정보는 아니므로 null을 허용한다. |
| gender | 문자형이며 필수 정보는 아니므로 null을 허용한다. |
| contractor | customer\_ID | Customer을 참조하며 PK이자 FK이다. 따라서 정수형이며 not null이다. |
| accountNumber | 정수형이며 not null이다. |
| address | 문자형이며 not null이다. |
| online  customer | customer\_ID | Customer을 참조하며 PK이자 FK이다. 따라서 정수형이며 not null이다. |
| cardNumber | 정수형이며 not null이다. |
| address | 문자형이며 not null이다. |

**4. ODBC implementation**

**4.1 CRUD**

**1) Create and Insert**

**텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명**

20160530.txt 파일에는 테이블의 생성과 tuple의 삽입을 위한 내용이 적혀있다. 먼저 파일을 읽기모드로 저장하여 모든 파일의 크기를 측정하고 buffer에 모든 내용을 담아둔다. 그 이후 정상적으로 mysql에 접속하면 buffer에서 ‘;’을 기준으로 문장을 읽어들여 sql에 데이터를 저장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**2) Select type**

**텍스트이(가) 표시된 사진

자동 생성된 설명**텍스트이(가) 표시된 사진

자동 생성된 설명

프로그램의 전반적인 형태는 while문과 switch문을 통해 실행된다. 원하는 숫자를 선택하고 1~7을 선택하면 원하는 쿼리문을 실행할 수 있으며, 0은 종료, 그 외 입력은 다시 입력하도록 구성하였다.

**3) Delete and Drop**

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

20160530\_2.txt에는 tuple의 삭제와 테이블의 drop을 위한 내용이 적혀있다. Create and insertion과 마찬가지로 buffer에 내용을 저장한 후에 ‘;’를 기준으로 읽어들여 mysql안에 있는 데이터를 제거한다.

**4.2 Queries**

**1) Type 1**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

추적 번호를 입력하면 추적 번호에 해당하는 고객의 정보를 출력한다.

**MySQL query**

select customer\_ID, customer\_name, contract

from customer where customer\_ID =

(select customer\_Id from order\_info natural join shipment where shipment\_Id = Traking number X)

주문정보와 배송정보를 담고 있는 order\_info와 shipment를 join하여 customer\_ID를 도출한다. 이후에 해당 ID를 customer 테이블에서 찾아 필요한 정보를 출력한다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

* 1. **Type 1-1**

텍스트이(가) 표시된 사진

자동 생성된 설명

Type-1의 서브 타입에서는 파손된 배송에 대한 상품을 다시 구성하여 배송을 시작한다. 새로운 주문 ID와 배송 ID를 구성하고 상품 정보를 받아와 내용을 출력한다.

**MySQL query**

select product\_ID, quantity from comprise where order\_ID =

(select order\_ID from order\_info natural join shipment where shipment\_Id = Traking number X)

주문정보, 배송정보를 담은 테이블을 join하여 추적정보와 일치하는 order\_ID를 선택한다. 이후 상품의 개수를 담고 있는 comprise에서 order\_ID와 일치하는 상품의 ID와 개수를 결과에 담는다.

insert into order\_info values ('new\_order', 'customer\_ID', 'year-month-day')

새로운 order\_ID와 이전에 찾아 놓은 customer\_ID, 오늘 날짜를 order\_info 테이블에 삽입한다.

insert into shipment values ('new\_shipment', 'new\_order', 'shipment\_name', ‘year-month-day', null)

새로운 shipment\_ID와 새로운 order\_ID와 관련 정보를 shipment 테이블에 삽입힌다.

insert into comprise values ('new\_order', 'product\_ID', 'quantity')

담아둔 상품의 ID와 개수를 결과를 comprise 테이블에 삽입한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**2) Type 2**

텍스트이(가) 표시된 사진

자동 생성된 설명

해당 년도를 입력하면 금액 기준으로 그 해 가장 많이 구입한 고객의 ID와 이름, 구매 총액을 출력한다.

**MySQL query**

with tb as(select customer\_ID, sum(quantity\*product\_price) as total\_price

from order\_info natural join comprise natural join product

where date(order\_date) between 'year-01-01' and 'year-12-31'

group by customer\_ID order by total\_price desc limit 1)

select customer\_ID, customer\_name, total\_price from tb natural join customer

먼저 with를 이용하여 테이블을 구성한다. 주문정보(order\_info)와 주문별 상품의 개수(comprise), 상품의 정보(product)를 담고 있는 테이블을 join하고 입력된 년도 사이의 값을 가져온다. 이때 customer\_ID로 묶은 후 상품의 가격과 수량을 곱한 값을 더한 total\_price를 도출하여 내림차순으로 정렬한 후 값을 하나만 취해 최대 값을 가져온다. 이후 이 테이블을 customer 테이블과 join하여 고객에 대한 정보를 출력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**2-1) Type 2-1**

텍스트이(가) 표시된 사진

자동 생성된 설명

그 해의 가장 많이 구입한 상품의 ID, 이름, 총 금액을 출력한다.

**MySQL query**

select product\_ID, product\_name, quantity\*product\_price as total\_price

from order\_info natural join comprise natural join product

where date(order\_date) between 'year-01-01' and 'year-12-31' and customer\_ID = 'customer\_ID'

group by product\_ID order by total\_price desc limit 1

주문정보(order\_info)와 주문별 상품의 개수(comprise), 상품의 정보(product)를 담고 있는 테이블을 join하여 테이블을 구성하는데, 해당 년도 사이의 값을 가져오며, customer\_ID가 최다 구매 고객과 일치하는 값을 찾는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**3) Type 3**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

입력된 년도에 판매된 모든 상품의 ID와 상품명을 출력한다.

**MySQL query**

select distinct (product\_ID), product\_name

from order\_info natural join comprise natural join product

where date(order\_date) between 'year-01-01' and 'year-12-31'

주문정보와 주문된 상품을 담고 있는 테이블을 join하고 이후에 상품의 정보를 담고 있는 테이블을 join한다. 입력된 년도에 해당하는 tuple만을 가져오는데, 중복된 상품을 제거하기 위해 distinct를 이용하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**3-1) Type 3-1**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

K를 입력 받아 금액 기준으로 해당 년도에 가장 많이 팔린 상품의 정보를 k개 만큼 출력한다.

**MySQL query**

select product\_ID, sum(quantity\*product\_price) as total\_price

from order\_info natural join comprise natural join product

where date(order\_date) between 'year-01-01' and 'year-12-31'

group by product\_ID order by total\_price desc limit K

주문정보(order\_info)와 주문된 상품(comprise) join하고 이후에 상품의 정보(product) join한 후 해당 년도에 해당하는 tuple을 가져온다. 이후에 상품의 ID로 묶은 후에 상품의 수량과 가격을 곱한 총 금액 순으로 내림차순 정렬한 후에 K만큼의 개수를 보여준다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**3-2) Type 3-2**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

해당 년도에 판매된 총 금액의 상위 10%안에 드는 상품의 정보를 출력한다.

**MySQL query**

with tb as(select product\_ID, product\_name, sum(quantity\*product\_price) as total\_price

from order\_info natural join comprise natural join product

where date(order\_date) between 'year-01-01' and 'year-12-31'

group by product\_ID order by total\_price desc)

select \* from(select product\_ID, product\_name, total\_price,

percent\_rank() over (order by total\_price desc)\*100 as per from tb) a

where per <=10

먼저 with을 이용하여 테이블을 구성한다. 테이블을 만드는 과정은 Type-3에서 사용한 쿼리와 동일하다. 이후에 total price를 내림차순으로 구성한 테이블에서 percent\_rank()를 활용하여 총 금액의 상위 퍼센트를 가져오고 그 값에 100을 곱하여 100%비율로 나타낸다. where에서 그 값이 10보다 작은 값을 가진 tuple만을 가져와 상품의 정보를 출력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**4) Type 4**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

입력된 년도에 판매된 모든 상품의 ID와 상품명, 판매 수량을 출력한다.

**MySQL query**

select product\_ID, product\_name, sum(quantity)

from order\_info natural join comprise natural join product

where date(order\_date) between 'year-01-01' and 'year-12-31'

group by product\_ID

주문정보와 주문된 상품을 담고 있는 테이블을 join하고 이후에 상품의 정보를 담고 있는 테이블을 join한다. 입력된 년도에 해당하는 tuple만을 가져오는데, product\_ID를 그룹으로 묶어 중복된 상품의 값을 제거하면서 수량의 합계를 가져온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**4-1) Type 4-1**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

K를 입력 받아 개수 기준으로 해당 년도에 가장 많이 팔린 상품의 정보를 k개 만큼 출력한다.

**MySQL query**

select product\_ID, product\_name, sum(quantity) as total\_q

from order\_info natural join comprise natural join product

where date(order\_date) between ‘year-01-01' and 'year-12-31'

group by product\_ID order by total\_q desc limit k

주문정보(order\_info)와 주문된 상품(comprise) join하고 이후에 상품의 정보(product) join한 후 해당 년도에 해당하는 tuple을 가져온다. 이후에 상품의 ID로 묶은 후에 상품의 수량의 총 합을 기준으로 내림차순 정렬한 후에 K만큼의 개수를 보여준다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**4-2) Type 4-2**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

해당 년도에 판매된 총 판매 개수의 상위 10%안에 드는 상품의 정보를 출력한다.

**MySQL query**

with tb as (select product\_ID, product\_name, sum(quantity) as total\_quantity

from order\_info natural join comprise natural join product

where date(order\_date) between '%s-01-01' and '%s-12-31'

group by product\_ID order by total\_quantity desc)

select \* from(select product\_ID, product\_name, total\_quantity,

percent\_rank() over (order by total\_quantity desc)\*100 as per from tb) a

where per <= 10

먼저 with을 이용하여 테이블을 구성한다. 테이블을 만드는 과정은 Type-4에서 사용한 쿼리와 동일하다. 이후에 total\_quantity를 내림차순으로 구성한 테이블에서 percent\_rank()를 활용하여 총 개수의 상위 퍼센트를 가져오고 그 값에 100을 곱하여 100%비율로 나타낸다. where에서 그 값이 10보다 작은 값을 가진 tuple만을 가져와 상품의 정보를 출력한다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**5) Type 5**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

원하는 창고의 위치를 입력하고 해당 창고에서 품절된 상품을 검색하여 결과를 출력한다.

**MySQL query**

select product\_ID, product\_name

from store natural join warehouse natural join product

where inventory = 0 and warehouse\_location = 'California'

창고(warehouse)와 상품의 정보(product), 그리고 상품의 재고가 저장되어 있는 테이블(store)을 각각 join한다. 이후에 재고가 0이면서 창고의 위치가 받아온 위치인 tuple의 product\_ID와 product\_name을 출력한다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**6) Type 6**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

약속된 시간 안에 배달되지 않은 패키지의 정보와 예정 배송일, 배송 완료일을 출력한다.

**MySQL query**

select product\_ID, product\_name, shipment\_date, completion\_date

from order\_info natural join shipment natural join comprise natural join product

where product\_ID > 100 and date(shipment\_date) < date(completion\_date)

주문정보(order\_info)와 배송정보(shipment)를 join하고 주문 상품(comprise)를 join한 이후 상품(product)을 join한다. 이후에 product\_ID가 100이상(일반 상품은 ID가 100미만의 값을 갖고, 패키지 상품은 100이상의 값을 가짐)인 값과, 배송 완료일이 예정 배송일 보다 큰 값을 찾아 출력한다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**7) Type 7**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

해당 월에 대한 정보를 입력하면 해당 월에 거래 내역이 있는 contractor에 대한 bill을 생성한다. 텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

**MySQL query**

select distinct customer\_ID, customer\_name, accountnumber

from order\_info natural join contractor natural join customer

where date(order\_date) between 'year-month-01' and 'year-month-31

주문 정보(order\_info)와 계약자(contractor)를 join하고 고객(customer)을 join한다. 이후에 해당 년도와 해당 입력한 월에 거래된 내역이 있는 customer\_ID, name, accountnumber을 가져오는데 distinct를 이용하여 중복된 값이 제거된 tuple이 결과에 저장한다.

select product\_ID, product\_name, product\_price, quantity, (product\_price\*quantity) as tot\_price

from order\_info natural join contractor natural join comprise

natural join product natural join customer

where date(order\_date) between 'year-month-01' and 'year-month-31’ and customer\_ID = 'customer\_ID’

주문 정보(order\_info)와 계약자(contractor)를 join하고 주문 상품(comprise)에 join, 해당 년, 월 사이의 값을 가져온다. 그리고 customer\_ID가 앞선 쿼리에서 저장된 customer\_ID와 일치하는 tuple을 가져와 저장되어 있는 product\_ID, product\_name, price, quantity, price와 quantity를 곱한 tot\_price를 가져와 bill안에 저장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Query1에서는 해당 월에 거래내역이 있는 contractor의 정보를 가져와 sql\_result에 저장하고 query2에서는 sql\_result에 저장된 row의 값을 하나씩 불러오면서 customer\_ID에 알맞은 상품과 가격을 sql\_result2에 저장하여 값을 차례대로 bill에 저장한다.

**Logical Schema Diagram**

