데이터베이스시스템 Project2 보고서

20200524 김준서

1. Logical Schema Design

- 상품 유통은 (공장생산(factory) -> 중간판매자(vendor) -> 소매업자(store)) 순서로 이루어지게 됩니다.
- 각 손님은 basic, VIP 중 하나의 등급을 부여받습니다
- 테이블은 크게 세 가지로 분류됩니다
- 각 유통 개체에 대한 정보: factory, vendor, store, product
- 재고 및 상품 발주에 대한 정보: ~_storage, ~_request
- 소매 단계에 대한 정보: person, purchase, loyalty

Project1: ERD -> Relational Schema

"테이블명(*): (ERD에 명시된 attribute들) + weak entity 등에 따라 추가된 attribute"

ㄴBCNF 결과에 따라 달라진 내용

Person: (customer_id, address, name, phone_num)

Store: (store_id, address, name, open, close, owner_type, phone_num)

Product: (product_upc, price, type, brand, size)

Vendor: (vendor_id, name, address, email)

Factory: (factory_id, name, address, email)

Customer*: (loyalty, discount_rate) + customer_id, store_id

ㄴLoyalty: (grade, dc_rate)

ㄴCustomer: (customer_id, store_id) (삭제)

ㄴCustomer_Loyalty: (customer_id, loyalty) (삭제)

Store_storage: (inventory, threshold, reorder) + store_id, product_upc

Store_request*: (request_id, date, quantity, shipping_mean, status) + store_id, product_upc, vendor_id

ㄴStore_request: (request_id, date, shipping_mean, status, store_id, product_upc, vendor_id)

ㄴReorder: (quantity, store_id, product_upc) (삭제)

Vendor_storage: (inventory, threshold, reorder) + vendor_id, product_upc

Vendor_request*: (request_id, date, quantity, shipping_mean, status) + vendor_id, product_upc, factory_id

ㄴVendor_request: (request_id, date, shipping_mean, status, vendor_id, product_upc, factory_id)

ㄴReorder: (quantity, vendor_id, product_upc) (삭제)

Produce: () + product_upc, factory_id

ㄴ삭제

Purchase: (purchase_id, date, quantity, method) + customer_id, store_id, product_upc

2. Normalization Analysis (BCNF)

1에서 설명한 Relation 기준으로 검토하겠습니다.
BCNF 과정에서 달라진 부분을 볼드체로 표시했습니다.

Person(customer_id, address, name, level, phone_num) - 고객정보
NotNull: name - 이름은 꼭 필요합니다
Unique: phone_num - 전화번호는 고유하게 부여되어 있습니다
FD(customer_id -> address, name, level, phone_num) - BCNF
PK: customer_id
FK: level -> Loyalty(grade) - Loyalty 테이블이 독립됨에 따라 생긴 외래키입니다. 이것에 대해서는 아래에서 더 자세히 설명하겠습니다.

Store(store_id, address, name, open, close, owner_type, phone_num) - 가게정보
NotNull: name, address - 위치가 꼭 필요합니다
Unique: phone_num
FD((store_id -> address, name, open, close, owner_type, phone_num) - BCNF
PK: store_id

Product(product_upc, **name**, price, type, brand, size) - 상품정보.
ㄴ기존에는 없던 Attribute인 name이 과제의 질의를 실행하기 위해 추가되었습니다.
NotNull: name
Check: price >= 0 - 상품의 가격은 음수일 수 없습니다
Unique: name - 상품의 이름은 고유합니다
FD((product_upc -> **name**, price, type, brand, size),
(name -> product_upc, price, type, brand, size)) - BCNF
PK: product_upc

Vendor(vendor_id, name, address, email) - 중간공급자정보
NotNull: name, address
Unique: email
FD(vendor_id -> name, address, email) - BCNF
PK: vendor_id

Factory(factory_id, name, address, email) - 생산공장정보
NotNull: name, address
Unique: email
FD(factory_id -> name, address, email) - BCNF

PK: factory_id

Customer(customer_id, store_id, loyalty, discount_rate) - 점포고객정보

FD((customer_id -> loyalty), (loyalty -> discount_rate)) - **BCNF 위반**

ㄴ하위테이블1

Loyalty(grade, dc_rate)

Check: grade in ('basic', 'VIP') - 고객등급은 두 가지로 나누어짐

****(이 때문에 해당 테이블에는 레코드가 두 개만 들어있습니다.)****

FD(grade -> dc_rate) - BCNF

PK: grade

ㄴ하위테이블2

Customer_loyalty(customer_id, loyalty) - 고객의 등급 정보

이것은 Person 릴레이션에 완전히 포함됨 --> 삭제

하위테이블3

Customer(customer_id, store_id) - 점포&고객관계정보

이것은 Purchase 릴레이션(추후 서술)에 완전히 포함됨 --> 삭제

Store_storage(store_id, product_upc, inventory, threshold, reorder) - 가게 재고정보

NotNull: inventory, threshold, reorder - 재고가 등록된 상품의 수량 정보는 Null 불가

Check: inventory >=0, threshold >= 0, reorder > threshold - 수량 음수 불가, 발주 이후 재고수량은 항상 임계치보다 높아야합니다.

FD((store_id, product_upc) -> inventory, threshold, reorder) - BCNF

PK: (store_id, product_upc)

FK: store_id -> store(store_id), product_upc -> product(product_upc)

Store_request(request_id, date, quantity, shipping_mean, status, store_id, product_upc, vendor_id) - 가게발주정보

FD((request_id -> date, quantity, shipping_mean, status, store_id, product_upc, vendor_id), ((product_upc, store_id) -> quantity)) - **BCNF 위반**

***각 가게의 발주량은 해당 가게의 reorder 값과 항상 일치한다**

ㄴ하위테이블1

Store_request(request_id, date, shipping_mean, status, store_id, product_upc, vendor_id) - 가게발주정보

NotNull: status

Check: status in ('checking', 'shipping', 'delivered') - 상태는 셋 중 하나만 가능

FD(request_id -> date, shipping_mean, status, store_id, product_upc, vendor_id) - BCNF

PK: request_id

FK: store_id -> store(store_id), product_upc -> product(product_upc), vendor_id -> vendor(vendor_id), 원래는 store_storage에 대한 FK(store_id, product_upc)도 정의하려 했으나 복합키 오류 가능성으로 배제

ㄴ하위테이블2

Reorder(quantity, store_id, product_upc) - 발주 수량 정보

이것은 store_storage 릴레이션에 완전히 포함됨 --> 삭제

Vendor_storage(vendor_id, product_upc, inventory, threshold, reorder) - 중간판매자 재고정보

NotNull: inventory, threshold, reorder - 재고가 등록된 상품의 수량 정보는 Null 불가

Check: inventory >=0, threshold >= 0, reorder > threshold - 수량 음수 불가, 발주 수량은 항상 임계치보다 높아야합니다.

FD((vendor_id, product_upc) -> inventory, threshold, reorder) - BCNF

PK: (vendor, product_upc)

FK: vendor_id -> vendor(vendor_id), product_upc -> product(product_upc)

Vendor_request(request_id, date, quantity, shipping_mean, status, vendor_id, product_upc, factory_id) - 중간판매자발주정보

FD((request_id -> date, quantity, shipping_mean, status, vendor_id, product_upc, factory_id), ((product_upc, vendor_id) -> quantity)) - **BCNF 위반**

***각 중간판매자의의 발주량은 해당 가게의 reorder 값과 항상 일치한다**

ㄴ하위테이블1

Vendor_request(request_id, date, shipping_mean, status, vendor_id, product_upc, factory_id) - 중간판매자발주정보

NotNull: status

Check: status in ('checking', 'shipping', 'delivered') - 상태는 셋 중 하나만 가능

FD(request_id -> date, shipping_mean, status, vendor_id, product_upc, factory_id) - BCNF

PK: request_id

FK: vendor_id -> vendor(vendor_id), product_upc -> product(product_upc), factory_id -> factory(factory_id), 원래는 vendor_storage에 대한 FK(vendot_id, product_upc)도 정의하려 했으나 복합키 오류 가능성으로 배제

ㄴ하위테이블2

Reorder(quantity, vendor_id, product_upc) - 발주 수량 정보

이것은 vendor_storage 릴레이션에 완전히 포함됨 --> 삭제

Produce(product_upc, factory_id) - 공장제품생산정보

이것은 vendor_request 릴레이션에 완전히 포함됨 --> 삭제

Purchase(purchase_id, date, quantity, method, customer_id, store_id, product_upc) - 단건판매정보

***고객은 한 가게를 하루에 최대 한 번만 방문한다고 가정**

NotNull: date, quantity

Check: quantity > 0, method in ('card', 'cash') - 판매수량은 항상 양수. 지불방식은 카

드와 현금만 가능함
FD(purchase_id -> date, quantity, method, customer_id, store_id, product_upc) - BCNF
PK: purchase_id
FK: customer_id -> person(customer_id), store_id -> store(store_id), product_upc -> product(product_upc), Store_storage에 대한 FK(store_id, product_upc)도 정의하려 했으나 복합키 오류 가능성 및 불필요하다 판단되어 생략함

3. Physical Implementation

Sample Data Description:
Loyalty: 등급{varchar}, 할인율{varchar} <-- __% 형식

Person: CXXXXXXX 형태의 ID{varchar}, 성씨{varchar}, 영문주소{varchar}, 등급{varchar}, 전화번호{varchar}

Store: SXXXXXXX 형태의 ID{varchar}, 가게이름{varchar}, 영문주소{varchar}, 오픈/마감{time}, 운영형태{varchar}, 전화번호{varchar}

Product: PXXXXXXX 형태의 ID{varchar}, 상품명(간격없음){varchar}, 가격{numeric}, 분류(snack/ramen/drink/coffee/water){varchar}, 브랜드(lotte/orion/nongshim/starbucks/ottuki){varchar}, 크기{varchar}

Vendor: VXXXXXXX 형태의 ID{varchar}, 이름{varchar}, 주소{varchar}, 이메일{varchar}

Factory: FXXXXXXX 형태의 ID{varchar}, 이름{varchar}, 주소{varchar}, 이메일{varchar}

Store_storage: 보관주체{varchar}, 상품{varchar}, 재고{numeric}, 임계{numeric}, 발주량{numeric}

Store_request: SRXXXXXX 형태의 ID{varchar}, 일자{date}, 운송수단(truck/van){varchar}, 배송상태(checking/shipping/delivered){varchar}, 주문자{varchar}, 상품{varchar}, 판매자{varchar}

Vendor_storage: 보관주체{varchar}, 상품{varchar}, 재고{numeric}, 임계{numeric}, 발주량{numeric}

Store_request: VRXXXXXX 형태의 ID{varchar}, 일자{date}, 운송수단(truck/van){varchar}, 배송상태(checking/shipping/delivered){varchar}, 주문자{varchar}, 상품{varchar}, 판매자{varchar}

Purchase: PCXXXXXX 형태의 ID{varchar}, 일자{date}, 수량{numeric}, 지불방법(card/cash){varchar}, 구매자{varchar}, 가게{varchar}, 상품{varchar}

4. Application Development

main() 함수의 형태는 다음과 같습니다

```
int main() {
 MYSQL *conn;
const char *server ="localhost";
const char *user ="root";
const char *password ="1234"; // 여기에 비밀번호 입력
const char *database ="store"; // 여기에 데이터베이스 이름 입력
// MySQL 초기화
...
// MySQL 서버 연결
...
int 입력값;
while(1){
- 선택화면 출력 -
- 사용자 입력 (몇 번 문제에 대한 답을 출력할지) -
switch (입력값){
(case 1 ~ 7) <-- 만일 범위 외 값이 입력되었을 경우 "Invalid Input" 출력 후 다시 루프 진입
 }
printf("\n");
 }
// 리소스 해제
return 0;
}
```

타입 1-7의 기본 형태는 다음과 같습니다

```
void Top_SellingItems(MYSQL *conn) {
MYSQL_RES *res;
MYSQL_ROW row;
char query[1024]; // 쿼리 문자열
char prdct[10]; // 사용자 입력 문자열 (필요한 경우)
printf("\n---------- TYPE n ----------\n");
printf("문제");
scanf("%s", prdct); // 사용자 입력 문자열 (필요한 경우)
sprintf(query, "SQL쿼리");
// 쿼리 실행
...
// 결과 저장
...
// 리소스 해제
...
}
```

생략된 부분들은 제공된 코드를 활용하였습니다.

다음은 문제에 따른 SQL 쿼리입니다.

1.

```
SELECT st.name, ss.product_upc, ss.inventory
FROM store_storage ss NATURAL JOIN store st
WHERE ss.product_upc = '입력문자열'
```

product_upc의 사용자 입력을 받아 해당 품목을 재고로 보유하는 매장의 정보를 출력

2.

```
WITH monthly_data AS (
SELECT *
FROM purchase
WHERE date >= '2025-05-01' AND date <= '2025-05-31'
),
monthly_total AS (
SELECT store_id, product_upc, SUM(quantity) AS sales
FROM monthly_data
GROUP BY store_id, product_upc
),
store_max_sales AS (
SELECT store_id, MAX(sales) AS m_sales
FROM monthly_total
GROUP BY store_id
)
SELECT st.name, pd.name
FROM monthly_total AS mt NATURAL JOIN store st
JOIN product pd ON mt.product_upc = pd.product_upc
JOIN store_max_sales AS sms
ON mt.store_id = sms.store_id AND mt.sales = sms.m_sales;
```

monthly_data: 지난 5월 간의 판매데이터를 추출

monthly_total: 각 가게 당 상품의 월간 판매량으로 정리

max_sales: 각 가게에서 월간 가장 많이 팔린 상품의 판매량을 추출

-> 각 가게에서 가장 많이 팔린 상품의 판매량과 일치하는 품목 이름 모두 출력

3.

```
WITH quarter_data as (
SELECT store_id, product_upc, quantity
FROM purchase
WHERE date >= '2025-04-01' AND date <= '2025-06-30'),
store_revenue as (
SELECT store_id, SUM(qd.quantity * pd.price) as revenue
FROM quarter_data as qd JOIN product as pd
ON qd.product_upc = pd.product_upc
GROUP BY store_id)
SELECT st.name
FROM store_revenue as sr NATURAL JOIN store st
WHERE sr.revenue = (SELECT MAX(revenue)
FROM store_revenue);
```

quarter_data: 이번분기(4월-6월)의 판매데이터를 추출

store_revenue: 가게별로 판매량*상품가격을 합해서 매출을 계산

-> 가장 높은 매출값과 일치하는 가게 이름 출력

4.

```
WITH vendor_total as (
SELECT vendor_id, store_id, product_upc
FROM store_request
WHERE status = 'delivered'),
vendor_q as (
SELECT vt.vendor_id, SUM(ss.reorder) as total_supply
FROM vendor_total as vt JOIN store_storage as ss
ON vt.store_id = ss.store_id AND vt.product_upc = ss.product_upc
```

```
GROUP BY vt.vendor_id)
SELECT vd.name, vq.total_supply
FROM vendor_q vq NATURAL JOIN vendor vd
WHERE total_supply = (SELECT MAX(total_supply)
FROM vendor_q);
```

vendor_total: 중간판매자 별로 배송완료된 품목을 추출
vendor_q: 중간판매자의 배송완료 품목의 전체 양을 store_id, product_upc를 통해 얻어낸 reorder 값들의 총합을 통해 추출
-> 가장 많은 배송량과 일치하는 가게를 찾아 이름과 배송량을 출력
5.

```
SELECT st.name, pd.name
FROM store_storage ss NATURAL JOIN store st
JOIN product pd ON ss.product_upc = pd.product_upc
WHERE inventory < threshold;
```

-> store_storage에서 가게별로 임계보다 낮은 재고를 가지고 있는 품목 이름을 출력
6.

```
WITH p_code as (
SELECT product_upc
FROM product
WHERE type = '사용자입력문자열'),
accompanied_data as (
SELECT pc.date as date, pc.customer_id as customer_id, pc.store_id as store_id
FROM purchase as pc JOIN p_code as pcd
ON pc.product_upc = pcd.product_upc
JOIN person as ps
ON pc.customer_id = ps.customer_id
WHERE ps.level = 'VIP'),
accompanied_product as (
SELECT pc.product_upc as product_upc, SUM(pc.quantity) as quantity
FROM purchase as pc JOIN accompanied_data as ad
ON pc.date = ad.date
AND pc.customer_id = ad.customer_id
AND pc.store_id = ad.store_id
WHERE pc.product_upc NOT IN (SELECT product_upc FROM p_code)
GROUP BY pc.product_upc)
SELECT pd.name, ap.quantity
FROM accompanied_product as ap JOIN product as pd
ON ap.product_upc = pd.product_upc
ORDER BY quantity DESC
LIMIT 3;
```

p_code: 사용자입력문자열 prdct(=coffee) type 밸류를 가진 제품들의 product_upc 추출
accompanied_data: 커피를 구매한 VIP 고객들의 구매날짜, 고객정보, 구매가게 추출
accompanied_product: 고객들이 커피와 함께 가게에서 구매한 모든 내역을 추출해서 상품별로 판매량을 합산, 판매량에 따라 정렬 (커피 빼고)
-> 상위 세 개의 투플 출력
7-1.

```
WITH franchise_list as (
SELECT store_id
FROM store
WHERE owner_type = 'franchise'),
```

```
franchise_pnum as (
SELECT fl.store_id, COUNT(ss.product_upc) as num
FROM franchise_list as fl JOIN store_storage as ss
ON fl.store_id = ss.store_id
GROUP BY fl.store_id)
SELECT st.name, fp.num
FROM franchise_pnum fp NATURAL JOIN store st
WHERE fp.num = (SELECT MAX(num)
FROM franchise_pnum);
```

franchise_list: 프랜차이즈 가게 목록 추출
franchise_pnum: 프랜차이즈 가게별로 판매하는 품목의 개수 추출
-> 가장 많은 품목 개수와 일치하는 가게 이름을 모두 출력
7-2.

```
WITH corporate_list as (
SELECT store_id
FROM store
WHERE owner_type = 'corporate')
SELECT st.name, COUNT(ss.product_upc) as num
FROM corporate_list as cl NATURAL JOIN store st
JOIN store_storage as ss
ON cl.store_id = ss.store_id
GROUP BY cl.store_id;
```

corporate_list: 직영 가게 목록 추출
-> 직영 가게별로 판매하는 품목의 개수 모두 추출

5. Testing and Validation

각 출력은 요구된 사항에 대해 사용자가 직관적으로 이해하기 쉬운 형태로 출력됩니다.
초기화면
---------- SELECT QUERY TYPE ----------

1. TYPE 1
2. TYPE 2
3. TYPE 3
4. TYPE 4
5. TYPE 5
6. TYPE 6
7. TYPE 7
0. QUIT

SELECT:

TYPE1
입력: product_upc
입력예시: P0000001 / P0000002 / P0000003 / P0000004...

출력예시:
---------- TYPE 1 ----------
Which stores currently carry a certain product (by UPC, name, or brand), and how much inventory do they have?
Enter a product_upc: P0000002
name product_upc inventory
SmileMart P0000002 30
EzMart P0000002 30

TYPE2
출력예시:
---------- TYPE 2 ----------
Which products have the highest sales volume in each store over the past month?
name name
SmileMart ramen_mild
SeoulMinis samdasoo
GreenShop vanila_latte

TYPE3
출력예시:
---------- TYPE 3 ----------
Which store has generated the highest overall revenue this quarter?
name
Happy24

TYPE4
출력예시
---------- TYPE 4 ----------
Which vendor supplies the most products across the chain, and how many total units have been sold?
name total_supply
Hansung Foods 500

TYPE5
출력예시
---------- TYPE 5 ----------
Which products in each store are below the reorder threshold and need restocking?
name name
SmileMart samdasoo
GreenShop choco_pie

EzMart ramen_spicy
EzMart swing_chip
EzMart fresh
TownMart ramen_mild

TYPE6
입력예시: coffee
출력예시
---------- TYPE 6 ----------
List the top 3 items that loyalty program customers typically purchase with coffee.
Enter a product type 'coffee': coffee
name quantity
choco_pie 41
yogurt_jelly 38
pepero 23

TYPE7
출력예시
---------- TYPE 7 ----------
Among franchise-owned stores, which one offers the widest variety of products, and how does that compare to corporate-owned stores?
franchise store with the widest product variety is:
name num
EzMart 5

corporate stores have following variety of products:
name num
GreenShop 2
SeoulMinis 4

Validation:
유통망(가게, 중간판매자, 공장)은 전국에 고르게 퍼져있고 정보의 완성도도 각 개체마다 차이가 존재하도록 데이터를 설계했습니다.
제품의 경우 주류, 라면, 간식 등을 고르게 사용하였습니다.
재고의 경우 어떤 유통자는 여러 품목을 취급하고 어떤 유통자는 소수의 품목을 취급하는 등 다양성을 반영하였습니다.
발주에 대하여 최근의 요청은 배송 혹은 확인 중, 이전의 것들은 배송완료로 나타나는 등 현실성을 반영하였습니다.
구매의 경우 VIP 고객에 대해 더 많은 판매정보가 존재하도록 설계했습니다.