DB Project2 보고서

20171618 김소흥

정성원 교수님

1. Physical Schema Diagram

<table>
<tr><td>Table Name</td><td>Table information (foreignkey constraints, constraints, data type, attribute name)</td></tr>
<tr><td>Brands</td><td><pre>CREATE TABLE Brands (
    id int PRIMARY KEY,
    name varchar(20),
    established_date timestamp
);</pre></td></tr>
<tr><td colspan="2">Id : Brand id
Name : brand name
Established_date : brand's established date</td></tr>
<tr><td>Models</td><td><pre>CREATE TABLE Models (
    id int PRIMARY KEY,
    brand_id int,
    name varchar(50),
    release_year int,
    vehicle_type varchar(50),
    fuel_type varchar(50),
    door_type smallint,

    FOREIGN KEY (brand_id) REFERENCES brands (id)
);</pre></td></tr>
<tr><td colspan="2">Id : model id
Brand_id : brand id
Name : model name
Release_year : model's release year
Vehicle type : ex) convertible, wega, sedan, suv
Fuel type : ex) gasoline, diesel
Door type : ex) 4 door, 2 door</td></tr>
<tr><td>Options</td><td><pre>CREATE TABLE Options (
    id int PRIMARY KEY,
    color varchar(20),
    engine_type varchar(20),
    transmission_type varchar(20)
);</pre></td></tr>
<tr><td colspan="2">Id : option id
Color : vehicle's color
Engine_type : ex) V6, V8, V12
Transmission_type : ex) AT, DT, CT</td></tr>
</table>

Vehicles	

```
CREATE TABLE Vehicles (
    VIN int PRIMARY KEY,
    option_id int UNIQUE,
    model_id int,

    FOREIGN KEY (option_id) REFERENCES Options (id),
    FOREIGN KEY (model_id) REFERENCES Models (id)
);
```

VIN : Vehicle's identification number
Option_id : option id
Model_id : model id

Customers	

```
CREATE TABLE Customers (
    id int PRIMARY KEY,
    name varchar(20),
    address varchar(20),
    phone int,
    gender varchar(10),
    annual_income int
);
```

Id : customer id
Name : customer name
Address : customer address
Phone: customer phone number
Gender : customer gender
Annual income : customer annual income

Dealers	

```
CREATE TABLE Dealers (
    id int PRIMARY KEY,
    name varchar(20),
    location varchar(50),
    phone int
);
```

Id : dealer id
Name : dealer name
Location : dealer location
Phone : dealer phone number

Sales

```
CREATE TABLE Sales (
    id int PRIMARY KEY,

    customer_id int,
    dealer_id int,
    VIN int UNIQUE,

    price int,
    purchased_at timestamp,
    receipt_at timestamp,

    FOREIGN KEY (customer_id) REFERENCES Customers (id),
    FOREIGN KEY (dealer_id) REFERENCES Dealers (id),
    FOREIGN KEY (VIN) REFERENCES Vehicles (VIN)
);
```

Id: sale id
Customer_id : customer id
Dealer_id : dealer id
VIN : vehicles VIN
Price : Vehicle's selled price
Purchased_at : Vehicle's purchase date (from dealer to customer)
Receipt_at : Vehicle's received date (from Company to dealer)

Suppliers

```
CREATE TABLE Suppliers (
    id int PRIMARY KEY,
    name varchar(20)
);
```

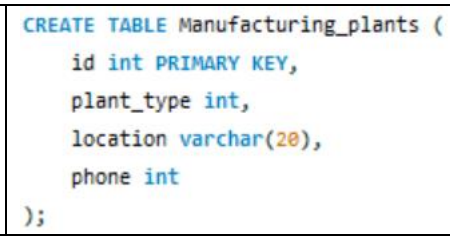
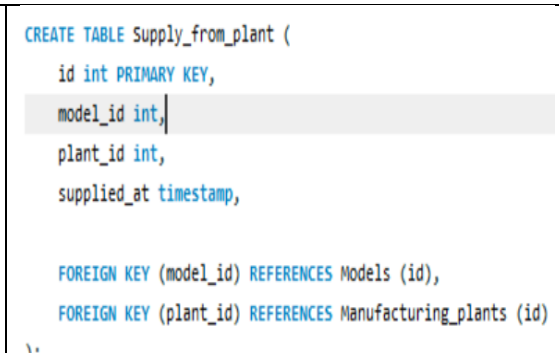
Id : supplier id
Name : supplier name

Supply_from_supplier

```
CREATE TABLE Supply_from_supplier (
    id int PRIMARY KEY,
    model_id int,
    supplier_id int,
    supplied_at timestamp,

    FOREIGN KEY (model_id) REFERENCES Models (id),
    FOREIGN KEY (supplier_id) REFERENCES Suppliers (id)
);
```

Id : supply id
Model_id : model id
Supplier_id : supplier's id
Supplied_at : supplied date

Manufacturing_plants	
Id : plant id Plant_type : supply or assemble Location : plant location Phone : plant phone number	
Supply_from_plant	
Id: supply id Model_id : model_id Plant_id : plant id Supplied_at : supplied date	

2. Check BCNF and decompose

모든 table의 fun. dep.이 오직 primarykey 에서 나머지 attribute로 가는 것만 존재하므로 추가적인 BCNF decomposition은 필요없었다.

Table Name	Func. Dep.	Check BCNF
Brands	Id->name, established_date	id is superkey, so BCNF
Models	Id->brand_id, name, release_year, Vehicle_type, fuel_type, door_type	id is superkey, so BCNF
Options	Id->color, engine_type, transmission_type	id is superkey, so BCNF
Vehicles	VIN->option_id, model_id	id is superkey, so BCNF

Customers	Id->name, address, phone, gender, annual_income	id is superkey, so BCNF
Dealers	Id->name, location, phone	id is superkey, so BCNF
Sales	Id->customer_id, dealer_id, VIN, price, purchased_at, receipt_at	id is superkey, so BCNF
Suppliers	Id->name	id is superkey, so BCNF
Supply_from_supplier	Id->model_id, supplier_id, supplied_at	id is superkey, so BCNF
Manufacturing_plants	Id->plant_type, location, phone	id is superkey, so BCNF
Supply_from_plant	Id->model_id, plant_id, supplied_at	id is superkey, so BCNF

3. Description on physical schema and ODBC C language

1) database connect : localhost에 존재하는 workbench를 통해 미리 만들어 놓은 database에 연결한다. 아직 create table, insert tuple을 수행하지 않았으므로 이 database는 현재 비어있다.

```
connection = mysql_real_connect(&conn, host, user, pw, db, 3306, (const char*)NULL, 0);
if (connection == NULL)
{
    printf("%d ERROR : %s₩n", mysql_errno(&conn), mysql_error(&conn));
    return 1;
}

else
{
    printf("Connection Succeed, wating for creating database₩n");

    if (mysql_select_db(&conn, db))
    {
        printf("%d ERROR : %s₩n", mysql_errno(&conn), mysql_error(&conn));
        return 1;
    }
```

2) create table, insert tuple : 성공적으로 database에 연결한 다음, text 파일을 통해 create table과 insert tuple을 실행해 database를 채운다. 이때 buffer의 최대 용량을 고려하여 이중 while문을 통해 ';' 단위로 text 파일을 읽으며 한 블록 단위 (create table 1개, insert tuple 1개씩)로 buffer에 저장 후 query로 보낸다.

```
while (feof(fp) == 0) {
    while (feof(fp) == 0) {
        buffer[i] = fgetc(fp);
        printf("%c", buffer[i]);
        if (buffer[i] == ';') {
            //sql 전송
            const char* query = buffer;
            int state = 0;

            state = mysql_query(connection, query);

            if (state == 0)
            {
                sql_result = mysql_store_result(connection);
                mysql_free_result(sql_result);
            }
            for (int j = 0; j <= i; j++) {
                buffer[j] = '₩0';
            }
            i = 0;
            break;
        }
        i++;
    }
}
```

3) query execution by user input : 성공적으로 database 구성을 마친 후에는 사용자 입력에 따른 query문 들을 수행한다. 사용자는 미리 정해진 7개의 query 중 하나를 골라 query에 따라 원하는 input을 입력하면 해당 query를 mysql_query()를 통해 server에 전송하고 result를 받아온다. 받아온 result를 printf()를 통해 console 창에 출력한다. 이때 사용자 입력에 따라 query에 들어가는 변수가 항상 바뀌어야 하므로 먼저 sprint()를 통해 buffer에 입력받은 input을 포함한 query를 저장시키고 이 buffer의 내용을 전송하는 방식으로 구현하였다. type 1, type 2의 경우 사용자의 선택에 따라 같은 사용자 입력에 따른 추가적인 결과를 출력해야 하므로, nested if문을 통해 구현했다. 성공적으로 query 하나를 출력하면 또 다른 query를 선택해 실행할 수 있다. 사용자가 0을 입력하면 종료된다.

```
case 5:
    //system("cls");
    printf("---- TYPE 5 ----₩n₩n");
    printf("** Find the top k brands by unit sales by the year**₩n");
    printf(" Which K? : ");
    scanf("%d", &k);
    char temp_buf4[400];
    sprintf(temp_buf4, "SELECT brand_id, name, y, unit_sales FROM( SELECT*, RANK() OVER(PARTITION BY y ORDER BY unit_sales DESC) AS rnk FROM s) AS x WHERE rnk <= '%d'; ", k);
    state = 0;

    state = mysql_query(connection, temp_buf4);
    printf("₩n₩n%s₩t%s₩t₩t%s₩t%s₩n", "brand_id", "name", "year", "unit_sales");
    printf("-------------------------------------------------------------₩n");
    if (state == 0)
    {
        sql_result = mysql_store_result(connection);
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
        {
            printf("%s₩t₩t%s₩t₩t%s₩t%s₩n", sql_row[0], sql_row[1], sql_row[2], sql_row[3]);
        }
        printf("₩n₩n");
        mysql_free_result(sql_result);
    }
    break;
```

4) delete, drop : 사용자가 0을 입력해 성공적으로 query execution이 종료된 경우, 프로그램을 종료하기 전에 database에 있는 정보를 전부 삭제해야 한다. 이를 delete와 drop을 사용해 구현했다. table 별로 우선적으로 table의 tuple 수 만큼 반복문을 돌며 해당 tuple을 삭제하고, 모든 tuple을 삭제 한 후 해당 table을 삭제하는 방식으로 구현했다. 모든 table이 삭제 된 후에는 생성했던 view 까지 삭제함으로 써 database의 모든 정보가 삭제되도록 구현했다.

```
//delete and drop
    //sales
    for (i = 1; i <= 21; i++) {
        char delete_buf[400];
        sprintf(delete_buf, "delete from Sales where id = '%d'", i);
        state = 0;

        state = mysql_query(connection, delete_buf);
        if (state == 0)
        {
            sql_result = mysql_store_result(connection);
            mysql_free_result(sql_result);
        }
    }
    const char* c7 = "drop table Sales";
    state = 0;

    state = mysql_query(connection, c7);
    if (state == 0)
    {
        sql_result = mysql_store_result(connection);
        mysql_free_result(sql_result);
    }
```

4. query explanation

Query type	explanation
Type 1	사용자가 년 수, 브랜드 이름을 입력하면 input에 따른 과거 년도에서 2021년 까지의 해당 브랜드의 모델 별 판매를 출력한다.
Type 1-1	Type 1의 결과에 추가적으로 구매자 정보를, 구매자의 성별에 따라 나열한다.
Type 1-1-1	Type 1-1의 결과에 추가적으로 구매자의 연간 소득에 따라 나열한다.
Type 2	사용자가 개월 수를 입력하면 2021-06-13부터 과거 k 달 동안의 모든 브랜드의 모델 별 판매를 출력한다.
Type 2-1	Type 2의 결과에 추가적으로 구매자 정보를, 구매자의 성별에 따라 나열한다
Type 2-1-1	Type 2-1의 결과에 추가적으로 구매자의 연간 소득에 따라 나열한다.
Type 3-1	사용자가 두 날짜와 공급처 이름을 입력하면 해당 기간 동안의 해당 업체 부품을 공급받아 제작된 model을 찾고, 해당 model을 구매한 고객과 VIN을 출력한다.
Type 3-2	사용자가 두 날짜와 공급처 이름을 입력하면 해당 기간 동안의 해당 업체 부품을 공급받아 제작된 model을 찾고, 해당 model을 판매한 판매처와 VIN을 출력한다.
Type 4	사용자가 k를 입력하면 년도 별로 매출액이 가장 큰 브랜드 상위 k개를 출력한다.
Type 5	사용자가 k를 입력하면 년도 별로 자동차 판매 수가 가장 많은 브랜드 상위 k개를 출력한다. 이때 판매수가 동률이면 동률인 브랜드를 모두 출력한다.
Type 6	Convertible이 가장 많이 팔린 달과 판매량을 출력한다.
Type 7	판매처 중 vehicle의 평균 보유 기간이 가장 긴 판매처를 출력한다. 보유 기간은 purchased date – received date를 의미한다. 여기서 received data는 판매처에 자동차가 입고된 날짜를 의미한다.