데이터베이스시스템(CSE4110)

Project 2 보고서

전공: 미국문화 학년: 4 학번: 20170175 이름: 김태안

1. 프로젝트 목표

GM 이나 Ford 와 같은 자동차 회사의 데이터베이스를 설계해본다. 이를 통해 관계형 데이터베이스의 개념적 설계와 논리적 설계, 적용, 작동, 유지보수를 경험한다. 자동차 회사는 미국의 전기 자동차 회사 테슬라 주식회사 (Tesla, Inc.)를 참고해 데이터베이스를 설계했다.

2. Logical Schema

2.1 BCNF Decomposition

기존 Logical Schema 의 이상현상을 막기 위해 모든 Relation 을 분석한 결과, 기존의 makes_vehicle 가 BCNF 를 만족하지 않는 것을 확인하였다.

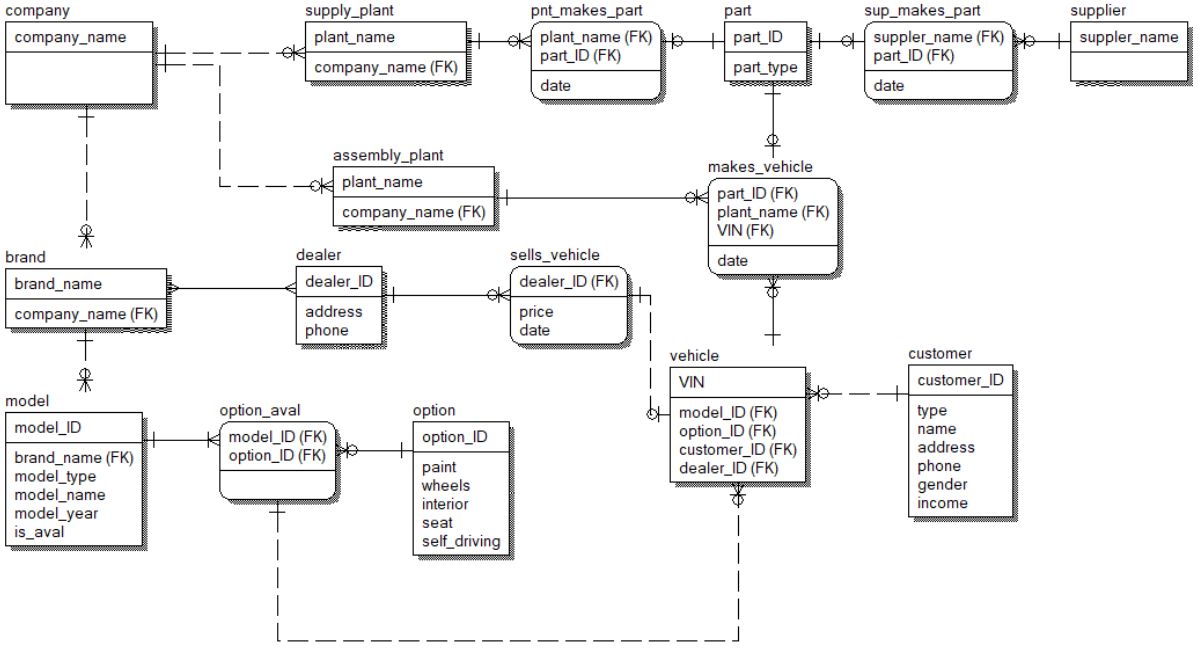


그림 1. 기존의 Logical Schema

part_ID 는 VIN 을 결정한다. 그리고, VIN 은 plant_name 을 결정한다. 즉, part_ID→VIN, VIN→ plant _name, date 이 모두 성립하는 transitive 한 관계이다. 이를 해결하기 위해 다음과 같이 makes_vehicle 을 uses_part 와 makes_vehicle 로 분해하였다.

$$(\alpha \cup \beta) = (\text{VIN} \cup \{\text{plant}_{\text{name}}, \text{date}\}) = (\text{VIN}, \text{plant\_name}, \text{date})$$

$$\big(R - (\beta - \alpha)\big) = (\{\text{part\_ID}, \text{VIN}, \text{plant\_name}, \text{date}\} - (\{\text{plant\_name}, \text{date}\} - \text{VIN}) = (\text{part\_ID}, \text{VIN})$$

추가로, 최소가 아닌 Primary Key 를 수정하였다.

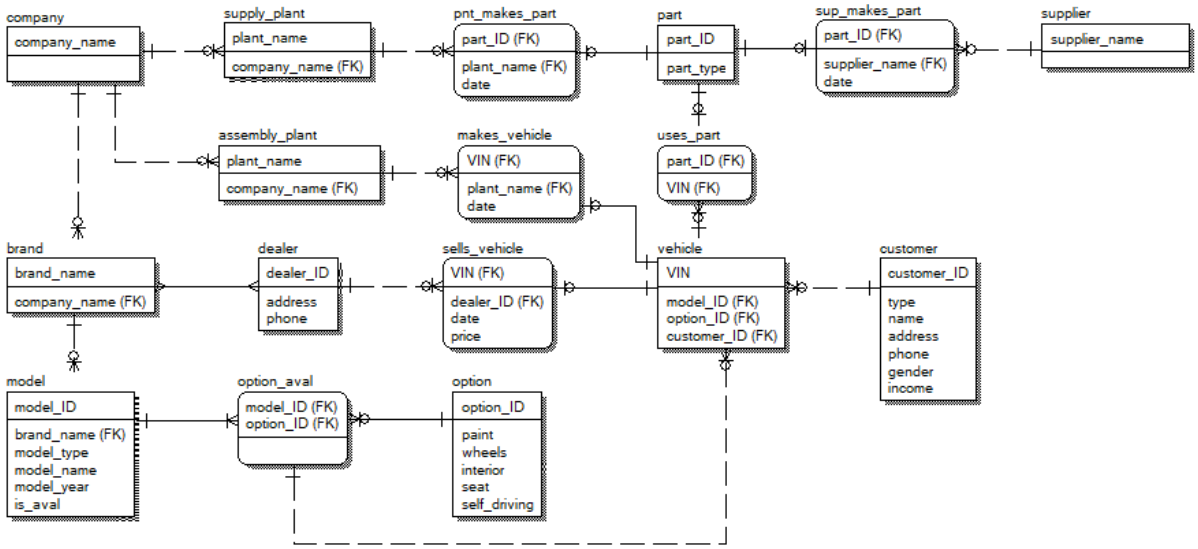


그림 2. Decomposition 을 적용한 Logical Schema

3. Physical Schema

각 Relation 의 Physical Schema 를 다음과 같이 구성하였다.

company			
company_name	VARCHAR(20)	NOT NULL	모든 이름은 20 자의 가변 문자열로 통일하였다.

company 는 데이터베이스를 사용하는 회사 Tesla Inc. 하나만 저장된다.

brand			
brand_name	VARCHAR(20)	NOT NULL	
company_name (FK)	VARCHAR(20)	NOT NULL	

brand 는 Tesla Inc.가 소유한 브랜드 Tesla 만 저장된다.

model			
model_ID	INTEGER	NOT NULL	model_ID 는 임의의 6 자리 숫자로 지정했다.
brand_name (FK)	VARCHAR(20)	NOT NULL	
model_type	VARCHAR(20)	NOT NULL	
model_name	VARCHAR(50)	NULL	길이가 긴 이름이 있어 50 자로 지정.

			아직 모델명을 정하지 않은 모델의 경우를 생각해 NULL 을 가능하게 함.
model_year	YEAR	NOT NULL	MySQL 에서 제공하는 연도 데이터 타입
is_aval	boolean	NOT NULL	

Tesla 의 자동차는 Model S, Model 3, Model X, Model Y, Roadster, Cybertruck 의 6 가지 model_type 이 존재한다. 각 모델은 엔진에 따라 모델명이 정해진다.

car_option			
option_ID	CHAR(21)	NOT NULL	option_ID 는 21 자의 문자로 이루어진다.
paint	VARCHAR(25)	NOT NULL	
wheels	VARCHAR(20)	NOT NULL	
seat	INTEGER	NULL	
interior	VARCHAR(20)	NOT NULL	
self_driving	boolean	NULL	

Tesla 의 모든 차는 5 가지 색을 가진다. 차에 따라 다른 바퀴를 사용하며, 모델 종류에 따라 4, 5, 6, 7 개의 좌석이 있다. 내부 인테리어는 3 가지 색이 존재하며, 색은 모두 긴 이름을 가지므로 20 자의 문자열 데이터 타입을 사용한다.

option_aval			
model_ID(FK)	INTEGER	NOT NULL	
option_ID(FK)	CHAR(21)	NOT NULL	

vehicle			
VIN	INTEGER	NOT NULL	VIN 은 6 자리의 임의의 숫자로 지정했다.
model_ID(FK)	INTEGER	NOT NULL	
option_ID(FK)	CHAR(21)	NOT NULL	
customer_ID(FK)	INTEGER	NULL	아직 팔리지 않은 차량의 경우를 고려.

customer			
customer_ID	INTEGER	NOT NULL	임의의 숫자를 부여하였다.
type	VARCHAR(11)	NOT NULL	private 과 enterprise 의 constraint 를 가진다.
name	VARCHAR(20)	NOT NULL	

address	VARCHAR(100)	NOT NULL	긴 문자열을 저장할 수 있게 하였다.
phone	CHAR(12)	NULL	-를 포함한 12 자리의 문자열
gender	VARCHAR(8)	NULL	male, female, company 의 constraint 를 가진다.
income	INTERGER	NULL	달러 기준.

dealer			
dealer_ID	INTEGER	NOT NULL	임의의 숫자를 부여하였다.
address	VARCHAR(100)	NOT NULL	긴 문자열을 저장할 수 있게 하였다.
phone	CHAR(12)	NULL	-를 포함한 12 자리의 문자열

sells_vehicle			
VIN(FK)	INTEGER	NOT NULL	
dealer_ID(FK)	INTEGER	NOT NULL	
date	DATE	NULL	MySQL 에서 제공하는 Date 자료형
price	INTEGER	NULL	달러 기준

supply_plant			
plant_name	VARCHAR(20)	NOT NULL	
company_name (FK)	VARCHAR(20)	NOT NULL	

supplier			
supplier_name	VARCHAR(20)	NOT NULL	

part			
part_ID	INTEGER	NOT NULL	납품 받은 부품이 가지는 고유번호로 임의의 숫자를 부여하였다.
part_type	VARCHAR(20)	NOT NULL	부품의 이름

pnt_makes_part			
part_ID(FK)	INTEGER	NOT NULL	
plant_name(FK)	VARCHAR(20)	NOT NULL	
date	DATE	NULL	

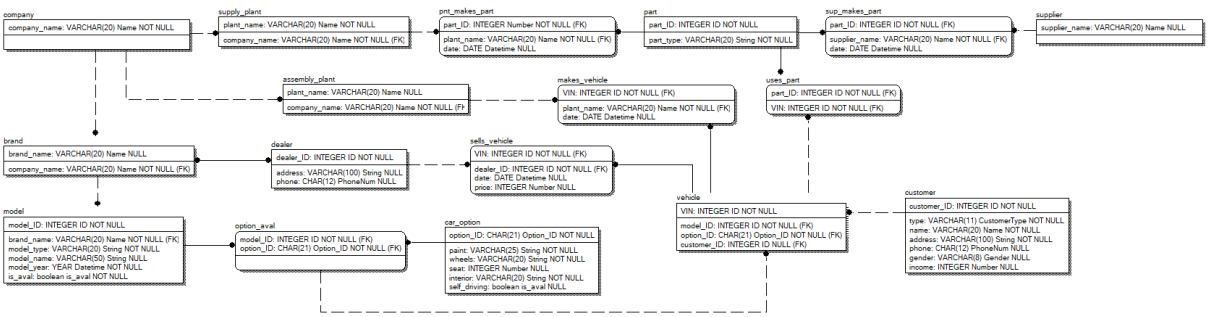
sup_makes_part			
part_ID(FK)	INTEGER	NOT NULL	
supplier_name(FK)	VARCHAR(20)	NOT NULL	
date	DATE	NULL	

assembly_plant			
plant_name	VARCHAR(20)	NOT NULL	
company_name (FK)	VARCHAR(20)	NOT NULL	

makes_vehicle			
VIN(FK)	INTEGER	NOT NULL	
plant_name(FK)	VARCHAR(20)	NOT NULL	
date	DATE	NULL	

uses_part			
part_ID(FK)	INTEGER	NOT NULL	
VIN(FK)	INTEGER	NOT NULL	

erwin 을 통해 구현한 Physical Schema 는 다음과 같다.



4. CRUD 프로그램

CRUD 프로그램은 DB 에 접속해 데이터를 입력, 읽기, 수정, 삭제하는 기능을 가진다. C 언어로 작성된 이 프로그램은 입력에 따라 정해진 CRUD 명령을 수행한다. CRUD 프로그램의 구조도는 다음과 같다.

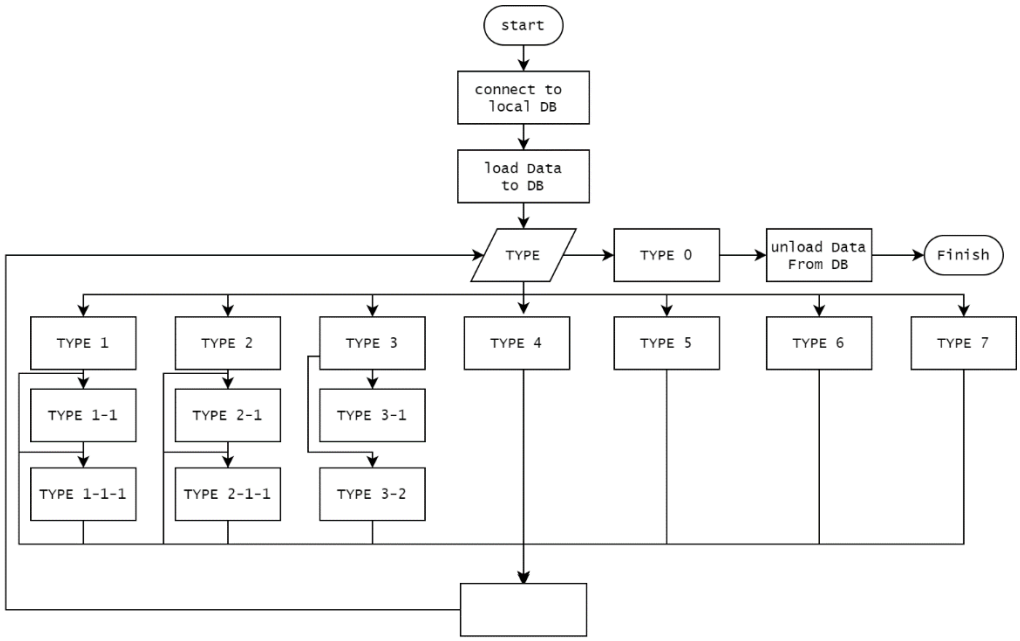


그림 3. CRUD 프로그램의 Flowchart

다음의 코드를 통해 C 언어를 사용해 MySQL 데이터베이스에 접속할 수 있다.

```
// Database Info
const char* host = "localhost";
const char* user = "20170175";
const char* pw = "cs1234";
const char* db = "project2";
int today = 20210614;

// Connect to DB
MYSQL* connection = NULL;
MYSQL conn;
MYSQL_RES* sql_result;
MYSQL_ROW sql_row;

if (mysql_init(&conn) == NULL)
    printf("Error : mysql_init() error\n");
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;
}
```

테이블의 생성과 데이터의 삽입은 20170175_start.txt 파일에 미리 작성한 SQL 문을 읽어 실행한다. txt 파일은 각 줄이 하나의 명령을 수행하는 SQL 문이다. 이를 전역변수 query 에 복사한 뒤, mysql_query 함수를 통해 쿼리를 실행한다.

```
// Generate DB with SQL create insert file
void DBGenerate() {
    int state = 0;
    FILE* fp = NULL;

    fopen_s(&fp, startFile, "r");
    if (fp == NULL) {
        printf("Error: Cannot open 20170175_start.txt");
        exit(1);
    }

    while (!feof(fp)) {
        fgets(&query, sizeof(query), fp);
        state = mysql_query(connection, query);
        if (state == 0)
        {
            sql_result = mysql_store_result(connection);
            mysql_free_result(sql_result);
        }
        memset(query, '\0', sizeof(query));
    }

    fclose(fp);
    return;
}
```

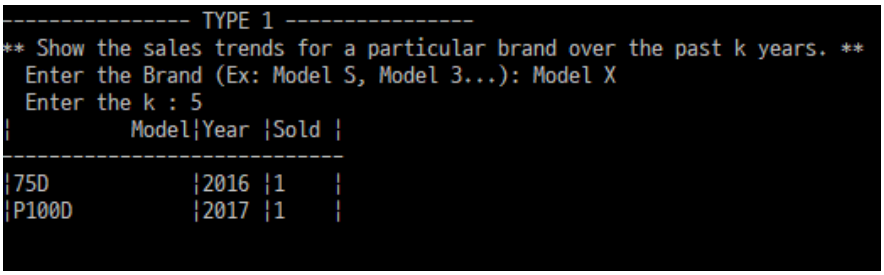
앞의 Logical Schema 의 모든 테이블을 생성하고, 테이블에 데이터를 삽입한다. 데이터 종류는 다음과 같다. 많은 양의 데이터 처리로 데이터를 삽입하는데 시간이 걸린다.

1. company: Tesla Inc.
2. brand: Tesla
3. Tesla 의 모델 총 35 가지 (실제 자료 기반)
4. 가능한 옵션 조합 282 가지 (실제 자료 기반)
5. 가능한 모델-옵션 조합 2730 가지 (실제 자료 기반)
6. 차량 15 대 (임의)
7. 고객 23 명 (임의)
8. 판매점 26 곳 (임의)
9. 판매기록 15 건 (임의)
10. 부품 85 개와 사용 차량 기록 85 건 (임의)
11. Tesla 부품공장 8 곳 (실제 자료 기반)
12. 납품 업체 20 곳 (실제 자료 기반)
13. 납품 기록 34(자회사)+51(납품업체) 건 (임의)
14. Tesla 조립공장 7 곳 (실제 자료 기반)
15. 조립 기록 15 건 (임의)

TYPE 명령어는 각 함수가 처리한다. 변수를 입력 받고 미리 입력한 SQL 문에 변수를 넣어 적절한 쿼리문을 만든 뒤, query 변수에 저장한다. 저장한 명령어를 mysql_query 함수를 통해 쿼리를 실행한다.

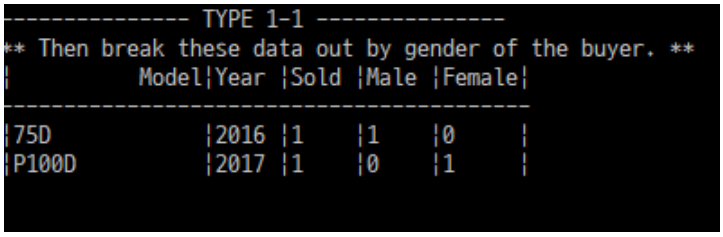
TYPE 1 은 지난 k 년 동안의 입력 받은 모델의 종류의 판매 경향을 질의해야 한다. 모델명과 출시연도에 따른 판매 차량 수를 출력한다.

```
SELECT model.model_name, model.model_year, COUNT(model_name) as total_sold
FROM vehicle JOIN model ON model.model_ID = vehicle.model_ID JOIN sells_vehicle ON sells_vehicle.VIN = vehicle.VIN
WHERE model.model_type = "모델명" and (sells_vehicle.date BETWEEN "날짜" AND "날짜")
GROUP BY model.model_name, model_year;
```



TYPE 1-1 은 위 검색 결과를 구매자의 성별에 따라 구분한다.

```
SELECT model_type, model_year, model_name, COUNT(model_name) as total_sold, count(case when gender = "male" then 1 end) as male, count(case when gender = "female" then 1 end) as female
FROM vehicle JOIN model ON model.model_ID = vehicle.model_ID JOIN sells_vehicle ON sells_vehicle.VIN = vehicle.VIN JOIN customer ON customer.customer_ID = vehicle.customer_ID
WHERE model.model_type = "모델명" and (sells_vehicle.date BETWEEN "날짜" AND "날짜")
GROUP BY model.model_name, model_year;
```



TYPE 1-1-1 은 1 의 결과를 수입에 따라 구분한다. 10,000 달러 단위로 수입을 나누고, 해당하는 고객의 수를 계산한다. 수입이 10,000,000 달러 이상인 고객은 모두 기업 고객이므로 따로 구분한다.

```
SELECT model_type, model_year, model_name, COUNT(model_name) as total_sold,
    count(case when income between 0 and 10000 then 1 end) as "~10,000",
   count(case when income between 10001 and 50000 then 1 end) as "~50,000",
   count(case when income between 50001 and 100000 then 1 end) as "~100,000",
   count(case when income between 100001 and 150000 then 1 end) as "~150,000",
   count(case when income between 150001 and 200000 then 1 end) as "~200,000",
```

```
    count(case when income between 200001 and 10000000 then 1 end) as "200,000+",
    count(case when income > 10000000 then 1 end) as "10,000,000+"
FROM vehicle JOIN model ON model.model_ID = vehicle.model_ID JOIN sells_vehicle ON sells_vehicle.VIN = vehicle.VIN JOIN customer ON customer.customer_ID = vehicle.customer_ID
WHERE model.model_type = "모델명" and (sells_vehicle.date BETWEEN "날짜" AND "날짜")
GROUP BY model.model_name, model_year;
```

```
-------------- TYPE 1-1-1 --------------
** Then by income range. **
|          Model|Year |Sold |~10,000    |~50,000    |~100,000   |~150,000   |~200,000   |200,000+   |10,000,000+|
-------------------------------------------------------------------------------------------------------------
|75D            |2016 |1    |0          |0          |1          |0          |0          |0          |0          |
|P100D          |2017 |1    |0          |1          |0          |0          |0          |0          |0          |
```

TYPE 2 는 지난 k 개월 동안의 모든 모델의 판매 경향을 보여준다.

```
SELECT model.model_type, model.model_year, model.model_name, COUNT(model_name) as total_sold
FROM vehicle JOIN model ON model.model_ID = vehicle.model_ID JOIN sells_vehicle ON sells_vehicle.VIN = vehicle.VIN
WHERE (sells_vehicle.date BETWEEN  BETWEEN date_sub(날짜, interval K month) AND 날짜)
GROUP BY model.model_type, model.model_name, model_year;
```

```
---------------- TYPE 2 ----------------
** Show sales trends for various brands over the past k months. **
  Enter the k : 12
|       Brand|          Model|Year |Sold |
-----------------------------------------
|Model X     |75D            |2016 |1    |
|Model 3     |55D            |2018 |1    |
|Model S     |Tri Motor AWD  |2021 |1    |
|Roadster    |Tri Motor AWD  |2020 |1    |
|Model Y     |Dual Motor AWD |2021 |1    |
|Cybertruck  |Tri Motor AWD  |2021 |1    |
```

TYPE 2-1 은 위 결과를 성별에 따라 분류한다. 기업이 구매한 차량은 표시되지 않는다.

```
SELECT model_type, model_year, model_name, COUNT(model_name) as total_sold, count(case when gender = "male" then 1 end) as male, count(case when gender = "female" then 1 end) as female
FROM vehicle JOIN model ON model.model_ID = vehicle.model_ID JOIN sells_vehicle ON sells_vehicle.VIN = vehicle.VIN JOIN customer ON customer.customer_ID = vehicle.customer_ID
WHERE (sells_vehicle.date BETWEEN date_sub(날짜, interval K month) AND 날짜)
GROUP BY model.model_type, model.model_name, model_year;
```

```
--------------- TYPE 2-1 ---------------
** Then break these data out by gender of the buyer. **
|       Brand|          Model|Year |Sold |Male |Female|
-------------------------------------------------------
|     Model X|            75D|2016 |1    |1    |0     |
|     Model 3|            55D|2018 |1    |1    |0     |
|     Model S|  Tri Motor AWD|2021 |1    |0    |0     |
|    Roadster|  Tri Motor AWD|2020 |1    |0    |1     |
|     Model Y| Dual Motor AWD|2021 |1    |0    |1     |
|  Cybertruck|  Tri Motor AWD|2021 |1    |0    |0     |
```

TYPE 2-1-1 은 위 결과를 수입에 따라 구분한다.

```
SELECT model_type, model_year, model_name, COUNT(model_name) as total_sold,
    count(case when income between 0 and 10000 then 1 end) as "~10,000",
   count(case when income between 10001 and 50000 then 1 end) as "~50,000",
   count(case when income between 50001 and 100000 then 1 end) as "~100,000",
   count(case when income between 100001 and 150000 then 1 end) as "~150,000",
   count(case when income between 150001 and 200000 then 1 end) as "~200,000",
   count(case when income between 200001 and 10000000 then 1 end) as "200,000+",
   count(case when income > 10000000 then 1 end) as "10,000,000+"
FROM vehicle JOIN model ON model.model_ID = vehicle.model_ID JOIN sells_vehicle ON
sells_vehicle.VIN = vehicle.VIN JOIN customer ON customer.customer_ID =
vehicle.customer_ID
WHERE sells_vehicle.date BETWEEN date_sub(날짜, interval K month) AND 날짜)\
GROUP BY model.model_type, model.model_name, model_year;
```

```
------------- TYPE 2-1-1 -------------
** Then by income range. **
|       Brand|          Model|Year |Sold |~10,000    |~50,000    |~100,000   |~150,000   |~200,000   |200,000+   |10,000,000+|
-------------------------------------------------------------------------------------------------------------------------------
|     Model X|75D            |2016 |1    |0          |0          |1          |0          |0          |0          |0          |
|     Model 3|55D            |2018 |1    |0          |0          |1          |0          |0          |0          |0          |
|     Model S|Tri Motor AWD  |2021 |1    |0          |0          |0          |0          |0          |0          |1          |
|    Roadster|Tri Motor AWD  |2020 |1    |0          |0          |1          |0          |0          |0          |0          |
|     Model Y|Dual Motor AWD |2021 |1    |0          |1          |0          |0          |0          |0          |0          |
|  Cybertruck|Tri Motor AWD  |2021 |1    |0          |0          |0          |0          |0          |0          |1          |
```

TYPE 3 는 납품업체와 기간을 입력 받고, 해당 기간동안 업체가 납품한 불량 변속기의 part_ID 를 출력한다.

```
SELECT part_ID
FROM sup_makes_part NATURAL JOIN part
WHERE (date BETWEEN 날짜 AND 날짜) AND part_type = "Transmission";
```

```
--------------- TYPE 3 ---------------
** Find that transmissions made by supplier (company name) between two given dates are defective. **
 Enter the Supplier : Bosch
 Enter the Start Date (YYYYMMDD) : 20000101
 Enter the End Date (YYYYMMDD) : 20211231
|PartID|
--------
|   104|
|   204|
|   304|
|   404|
|   504|
```

TYPE 3-1 은 해당 변속기가 포함된 차량과 고객을 출력한다.

```
SELECT VIN, customer_ID
FROM sup_makes_part NATURAL JOIN part NATURAL JOIN uses_part NATURAL JOIN vehicle
WHERE (date BETWEEN 날짜 AND 날짜) AND part_type = "Transmission";
```

```
--------------- TYPE 3-1 ---------------
** Find the VIN of each car containing such a transmission and the customer to which it was sold. **
|   VIN| CustomerID|
--------------------
|211101|          1|
|191102|          2|
|212103|          3|
|182104|          4|
|163105|          5|
```

TYPE 3-2 는 위에서 찾은 불량 변속기가 포함된 불량 차량을 판매한 판매자와 해당 차량을 출력한다.

```
SELECT dealer_ID
FROM sup_makes_part NATURAL JOIN part NATURAL JOIN uses_part NATURAL JOIN vehicle JOIN
sells_vehicle ON vehicle.VIN = sells_vehicle.VIN
WHERE (sup_makes_part.date BETWEEN 날짜 AND 날짜) AND part.part_type = "Transmission";
```

```
--------------- TYPE 3-2 ---------------
** Find the dealer who sold the VIN and transmission for each vehicle containing these transmissions **
| DealerID|   VIN|
------------------
|        3|211101|
|        2|191102|
|        5|212103|
|        2|182104|
|        1|163105|
```

TYPE 4 는 해당 연도에 가장 높은 매출을 기록한 모델 종류를 k 개까지 출력한다.

```
SELECT model.model_type, sum(price) AS "total_sold($)"
FROM vehicle JOIN model ON vehicle.model_ID = model.model_ID JOIN sells_vehicle ON
vehicle.VIN = sells_vehicle.VIN
WHERE YEAR(sells_vehicle.date) = 연도
GROUP BY model.model_type
ORDER BY SUM(price) DESC LIMIT k;
```

```
--------------- TYPE 4 ---------------
 ** Find the top k brands by dollar-amount sold by the year. **
  Enter k : 3
  Enter the Year (YYYY): 2021
| DealerID| totalSold|
----------------------
|  Model S|    332000|
|  Model X|    259000|
|  Model Y|    122000|
```

TYPE 5 는 해당 연도에 가장 많이 팔린 모델 종류를 k 개까지 출력한다.

```
SELECT model.model_type, count(sells_vehicle.VIN) as sold
FROM vehicle JOIN model ON vehicle.model_ID = model.model_ID JOIN sells_vehicle ON
vehicle.VIN = sells_vehicle.VIN
WHERE YEAR(sells_vehicle.date) = 연도
GROUP BY model.model_type
ORDER BY count(sells_vehicle.VIN) DESC LIMIT k;
```

```
---------------- TYPE 5 ----------------
** Find the top k brands by unit sales by the year. **
 Enter k : 4
 Enter the Year (YYYY): 2021
| DealerID| totalSold|
----------------------
|  Model X|         2|
|  Model S|         2|
|  Model Y|         2|
|  Model 3|         1|
```

TYPE 6 는 차량이 가장 많이 팔린 월을 출력한다. 현재 데이터에선 5 월이 3 대로 가장 많다.

```
SELECT MONTH(date) AS Month
FROM sells_vehicle
GROUP BY MONTH(date)
HAVING COUNT(MONTH(date)) = (SELECT MAX(monthcount)
    FROM (SELECT MONTH(date) AS mon, COUNT(MONTH(date)) AS monthcount
        FROM sells_vehicle
        GROUP BY MONTH(date)) AS monthdata);
```

```
---------------- TYPE 6 ----------------
** In what month(s) do convertibles sell best? **
| Month|
--------
|     5|
```

mon	monthcount
1	2
2	1
3	1
4	1
5	3
7	1
8	1
9	2
11	2
12	1

TYPE 7 에서는 차의 평균 매장 보관 기간이 가장 긴 판매점을 출력한다. 현재 데이터에서는 5 호점이 365 일로 가장 길다.

```
SELECT sells_vehicle.dealer_ID as dealer_ID
FROM sells_vehicle JOIN vehicle ON sells_vehicle.VIN = vehicle.VIN JOIN makes_vehicle ON
vehicle.VIN = makes_vehicle.VIN
GROUP BY sells_vehicle.dealer_ID
HAVING AVG(DATEDIFF(sells_vehicle.date, makes_vehicle.date)) = (SELECT MAX(storetime)
FROM       (SELECT       sells_vehicle.dealer_ID,       AVG(DATEDIFF(sells_vehicle.date,
makes_vehicle.date)) AS storetime
    FROM sells_vehicle JOIN vehicle ON sells_vehicle.VIN = vehicle.VIN JOIN makes_vehicle
ON vehicle.VIN = makes_vehicle.VIN
    GROUP BY sells_vehicle.dealer_ID) AS storedata);
```

```
---------------- TYPE 7 ----------------
 ** Find those dealers who keep a vehicle in inventory for the longest average time. **
¦DealerID¦
----------
¦       5¦
```

dealer_ID	storetime
1	152.0000
2	152.0000
3	152.5000
4	335.0000
5	365.0000
6	212.0000
7	121.0000
8	152.0000
9	152.0000
10	275.0000

프로그램은 0 을 입력 받으면 데이터베이스에 저장된 데이터를 모두 삭제하고 종료된다. 삭제를 위한 SQL 문은 20170175_end.txt 에 저장되어 있으며, 이 파일을 한 줄 씩 읽으며 모든 데이터와 테이블을 삭제한다.

```
// Delete DB with SQL drop delete file
void DBDrop() {
    int state = 0;
    FILE* fp = NULL;

    fopen_s(&fp, endFile, "r");
    if (fp == NULL) {
        printf("Error: Cannot open 20170175_end.txt");
        exit(1);
    }

    while (!feof(fp)) {
        fgets(&query, sizeof(query), fp);

        state = mysql_query(connection, query);
        if (state == 0)
        {
            sql_result = mysql_store_result(connection);
            mysql_free_result(sql_result);
            memset(query, '\0', sizeof(query));
        }
    }

    fclose(fp);
    return 0;
}
```

5. 프로젝트 환경

MySQL 8.0.25

MySQL Connector/OBDC 8.0.25

Microsoft Visual Studio Community 2019 버전 16.10.1

AllFusion ERwin Data Modeler 7.2.7.2110

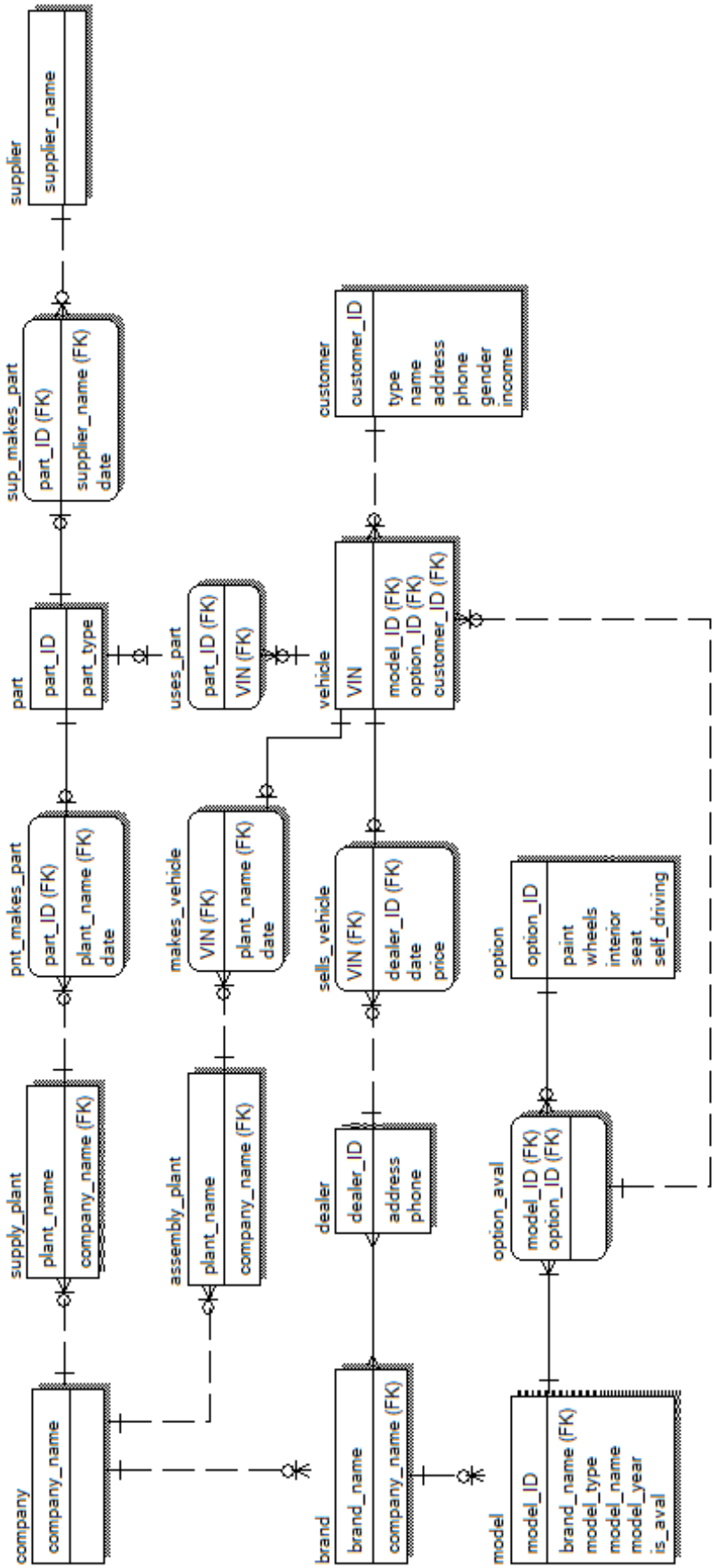
6. 참고 자료

Tesla 대한민국, www.tesla.com/ko_kr.

네이버 자동차, auto.naver.com/.

"Option Codes." *Tesla JSON API (Unofficial)*, tesla-api.timdorr.com/vehicle/optioncodes.

별첨 1. Logical Schema Diagram



별첨 2. Physical Schema Diagram