데이터베이스시스템(CSE4110)

Project 2 보고서

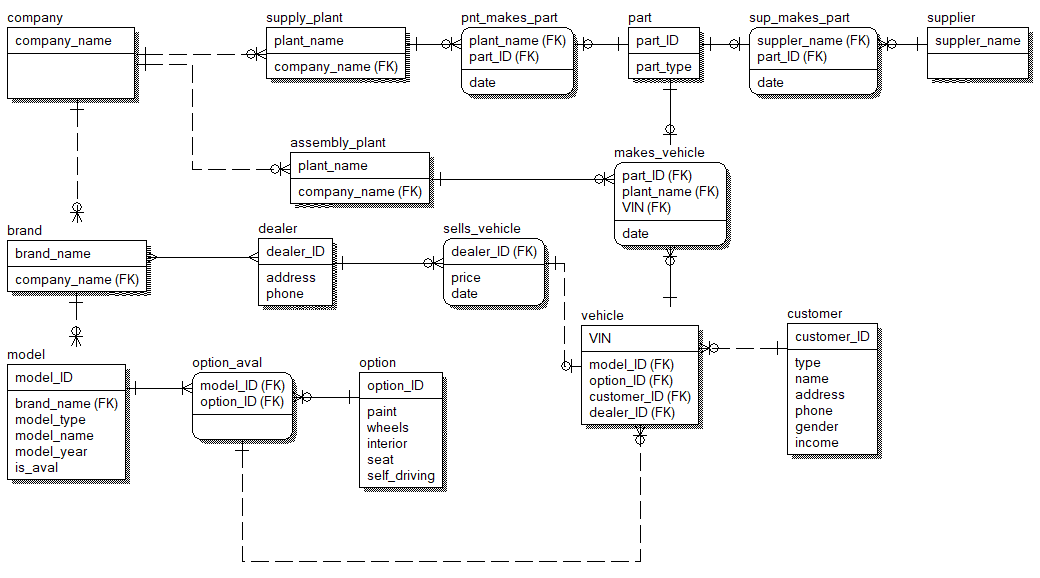
|  |  |  |  |
| --- | --- | --- | --- |
| 전공: 미국문화 | 학년: 4 | 학번: 20170175 | 이름: 김태안 |

1. 프로젝트 목표

GM이나 Ford와 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.

1. Logical Schema
   1. BCNF Decomposition

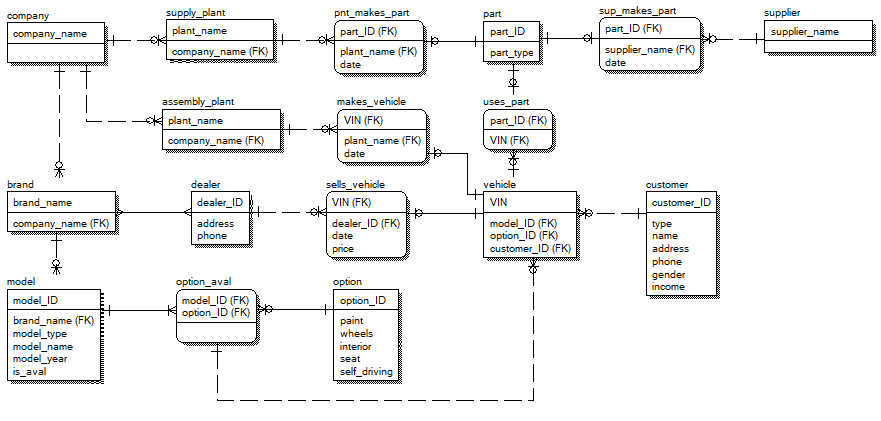
기존 Logical Schema의 이상현상을 막기 위해 모든 Relation을 분석한 결과, 기존의 makes\_vehicle가 BCNF를 만족하지 않는 것을 확인하였다.



*그림 1. 기존의 Logical Schema*

part\_ID는 VIN을 결정한다. 그리고, VIN은 plant\_name을 결정한다. 즉, part\_ID→VIN, VIN→ plant \_name, date이 모두 성립하는 transitive한 관계이다. 이를 해결하기 위해 다음과 같이 makes\_vehicle을 uses\_part와 makes\_vehicle로 분해하였다.

추가로, 최소가 아닌 Primary Key를 수정하였다.



*그림 2. Decomposition을 적용한 Logical Schema*

1. Physical Schema

각 Relation의 Physical Schema를 다음과 같이 구성하였다.

|  |  |  |  |
| --- | --- | --- | --- |
| **company** | | | |
| company\_name | VARCHAR(20) | NOT NULL | 모든 이름은 20자의 가변 문자열로 통일하였다. |

company는 데이터베이스를 사용하는 회사 Tesla Inc. 하나만 저장된다.

|  |  |  |  |
| --- | --- | --- | --- |
| **brand** | | | |
| brand\_name | VARCHAR(20) | NOT NULL |  |
| company\_name (FK) | VARCHAR(20) | NOT NULL |  |

brand는 Tesla Inc.가 소유한 브랜드 Tesla만 저장된다.

|  |  |  |  |
| --- | --- | --- | --- |
| **model** | | | |
| model\_ID | INTEGER | NOT NULL | model\_ID는 임의의 6자리 숫자로 지정했다. |
| brand\_name (FK) | VARCHAR(20) | NOT NULL |  |
| model\_type | VARCHAR(20) | NOT NULL |  |
| model\_name | VARCHAR(50) | NULL | 길이가 긴 이름이 있어 50자로 지정.  아직 모델명을 정하지 않은 모델의 경우를 생각해 NULL을 가능하게 함. |
| model\_year | YEAR | NOT NULL | MySQL에서 제공하는 연도 데이터 타입 |
| is\_aval | boolean | NOT NULL |  |

Tesla의 자동차는 Model S, Model 3, Model X, Model Y, Roadster, Cybertruck의 6가지 model\_type이 존재한다. 각 모델은 엔진에 따라 모델명이 정해진다.

|  |  |  |  |
| --- | --- | --- | --- |
| **car\_option** | | | |
| option\_ID | CHAR(21) | NOT NULL | option\_ID는 21자의 문자로 이루어진다. |
| paint | VARCHAR(25) | NOT NULL |  |
| wheels | VARCHAR(20) | NOT NULL |  |
| seat | INTEGER | NULL |  |
| interior | VARCHAR(20) | NOT NULL |  |
| self\_driving | boolean | NULL |  |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **option\_aval** | | | |
| model\_ID(FK) | INTEGER | NOT NULL |  |
| option\_ID(FK) | CHAR(21) | NOT NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **vehicle** | | | |
| VIN | INTEGER | NOT NULL | VIN은 6자리의 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 | private과 enterprise의 constraint를 가진다. |
| name | VARCHAR(20) | NOT NULL |  |
| address | VARCHAR(100) | NOT NULL | 긴 문자열을 저장할 수 있게 하였다. |
| phone | CHAR(12) | NULL | -를 포함한 12자리의 문자열 |
| gender | VARCHAR(8) | NULL | male, female, company의 constr aint를 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 12자리의 문자열 |

|  |  |  |  |
| --- | --- | --- | --- |
| **sells\_vehicle** | | | |
| VIN(FK) | INTEGER | NOT NULL |  |
| dealer\_ID(FK) | INTEGER | NOT NULL |  |
| date | DATE | NULL | MySQL에서 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 |  |

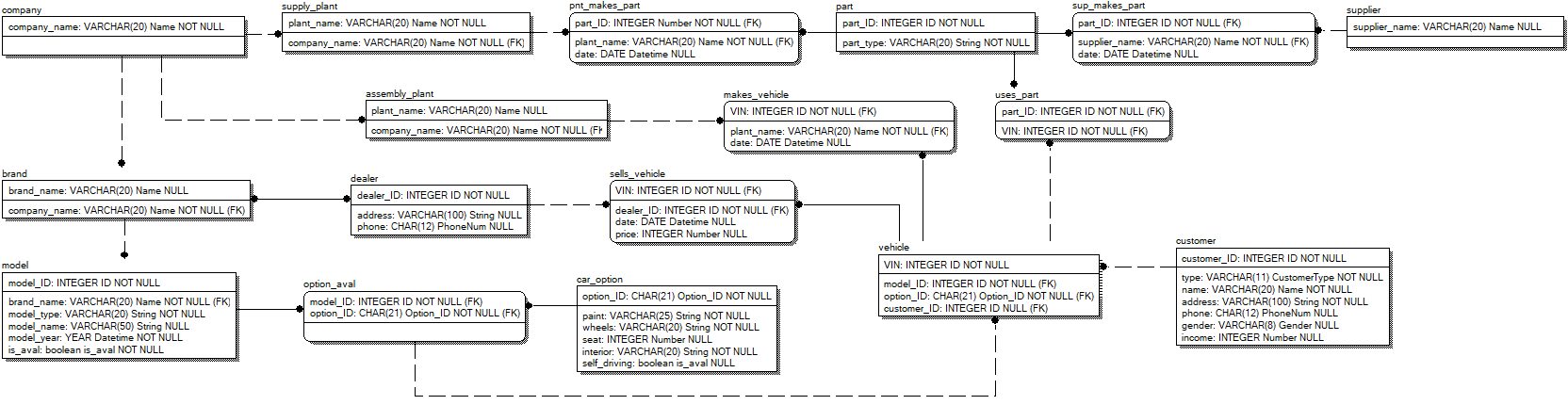
|  |  |  |  |
| --- | --- | --- | --- |
| **sup\_makes\_part** | | | |
| part\_ID(FK) | INTEGER | NOT NULL |  |
| supplier\_name(FK) | VARCHAR(20) | NOT NULL |  |
| date | DATE | NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |  |

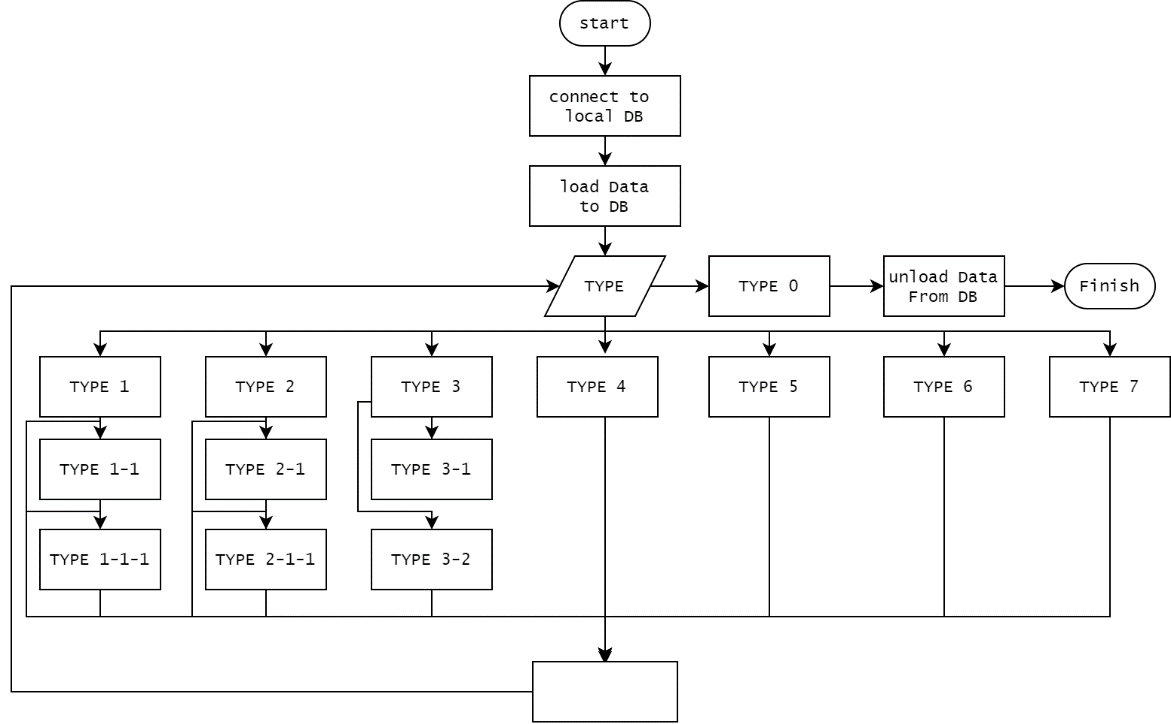
|  |  |  |  |
| --- | --- | --- | --- |
| **uses\_part** | | | |
| part\_ID(FK) | INTEGER | NOT NULL |  |
| VIN(FK) | INTEGER | NOT NULL |  |

erwin을 통해 구현한 Physical Schema는 다음과 같다.



1. CRUD 프로그램

CRUD 프로그램은 DB에 접속해 데이터를 입력, 읽기, 수정, 삭제하는 기능을 가진다. C언어로 작성된 이 프로그램은 입력에 따라 정해진 CRUD 명령을 수행한다. CRUD 프로그램의 구조도는 다음과 같다.



*그림 3. CRUD 프로그램의 Flowchart*

다음의 코드를 통해 C언어를 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 SQL문을 읽어 실행한다. txt 파일은 각 줄이 하나의 명령을 수행하는 SQL문이다. 이를 전역변수 query에 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;  } |

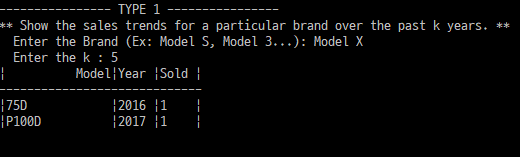
앞의 Logical Schema의 모든 테이블을 생성하고, 테이블에 데이터를 삽입한다. 데이터 종류는 다음과 같다. 많은 양의 데이터 처리로 데이터를 삽입하는데 시간이 걸린다.

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

TYPE 명령어는 각 함수가 처리한다. 변수를 입력 받고 미리 입력한 SQL문에 변수를 넣어 적절한 쿼리문을 만든 뒤, query 변수에 저장한다. 저장한 명령어를 mysql\_query 함수를 통해 쿼리를 실행한다.

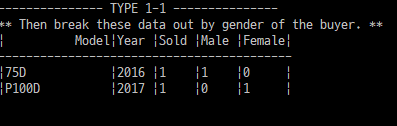
TYPE 1은 지난 k년 동안의 입력 받은 모델의 종류의 판매 경향을 질의해야 한다. 모델명과 출시연도에 따른 판매 차량 수를 출력한다.

|  |
| --- |
| 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; |



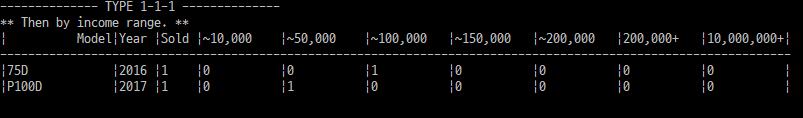
TYPE 1-1은 위 검색 결과를 구매자의 성별에 따라 구분한다.

|  |
| --- |
| 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; |



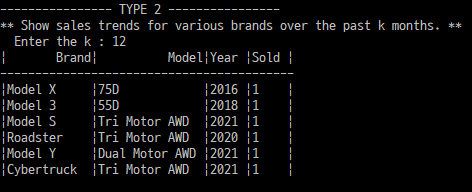
TYPE 1-1-1은 1의 결과를 수입에 따라 구분한다. 10,000 달러 단위로 수입을 나누고, 해당하는 고객의 수를 계산한다. 수입이 10,000,000 달러 이상인 고객은 모두 기업 고객이므로 따로 구분한다.

|  |
| --- |
| 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; |



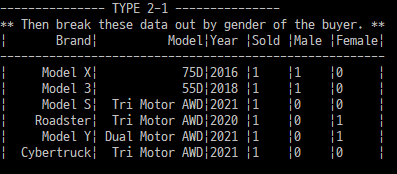
TYPE 2는 지난 k개월 동안의 모든 모델의 판매 경향을 보여준다.

|  |
| --- |
| 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; |



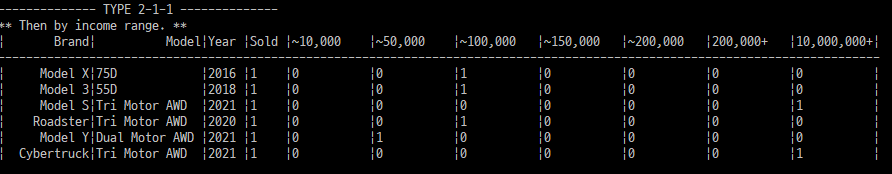
TYPE 2-1은 위 결과를 성별에 따라 분류한다. 기업이 구매한 차량은 표시되지 않는다.

|  |
| --- |
| 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; |



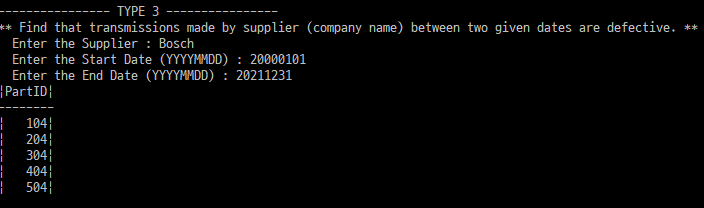
TYPE 2-1-1은 위 결과를 수입에 따라 구분한다.

|  |
| --- |
| 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; |



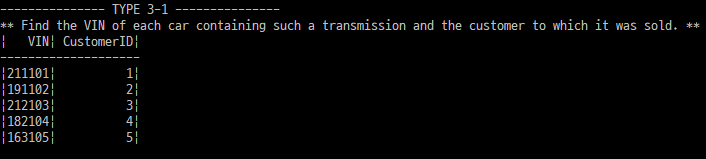
TYPE 3는 납품업체와 기간을 입력 받고, 해당 기간동안 업체가 납품한 불량 변속기의 part\_ID를 출력한다.

|  |
| --- |
| SELECT part\_ID  FROM sup\_makes\_part NATURAL JOIN part  WHERE (date BETWEEN 날짜 AND날짜) AND part\_type = "Transmission"; |



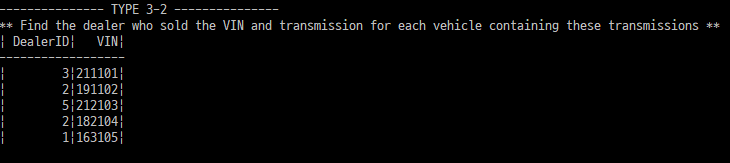
TYPE 3-1은 해당 변속기가 포함된 차량과 고객을 출력한다.

|  |
| --- |
| 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"; |



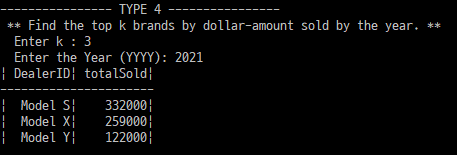
TYPE 3-2는 위에서 찾은 불량 변속기가 포함된 불량 차량을 판매한 판매자와 해당 차량을 출력한다.

|  |
| --- |
| 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 AND날짜) AND part.part\_type = "Transmission"; |



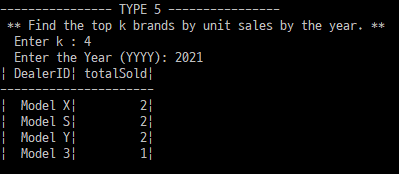
TYPE 4는 해당 연도에 가장 높은 매출을 기록한 모델 종류를 k개까지 출력한다.

|  |
| --- |
| 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; |



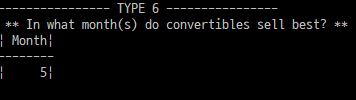
TYPE 5는 해당 연도에 가장 많이 팔린 모델 종류를 k개까지 출력한다.

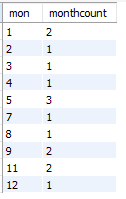
|  |
| --- |
| SELECT model.model\_type, count(sells\_vehicle.VIN) as sold  FROM vehicle JOIN model ON vehicle.model\_ID = model.model\_ID JOIN sells\_vehicle ON vehicle.VIN = sells\_vehicle.VIN  WHERE YEAR(sells\_vehicle.date) = 연도  GROUP BY model.model\_type  ORDER BY count(sells\_vehicle.VIN) DESC LIMIT k; |



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

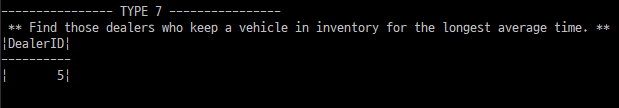
|  |
| --- |
| SELECT MONTH(date) AS Month  FROM sells\_vehicle  GROUP BY MONTH(date)  HAVING COUNT(MONTH(date)) = (SELECT MAX(monthcount)  FROM (SELECT MONTH(date) AS mon, COUNT(MONTH(date)) AS monthcount  FROM sells\_vehicle  GROUP BY MONTH(date)) AS monthdata); |

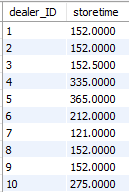




TYPE 7에서는 차의 평균 매장 보관 기간이 가장 긴 판매점을 출력한다. 현재 데이터에서는 5호점이 365일로 가장 길다.

|  |
| --- |
| 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); |





프로그램은 0을 입력 받으면 데이터베이스에 저장된 데이터를 모두 삭제하고 종료된다. 삭제를 위한 SQL문은 20170175\_end.txt에 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;  } |

1. 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

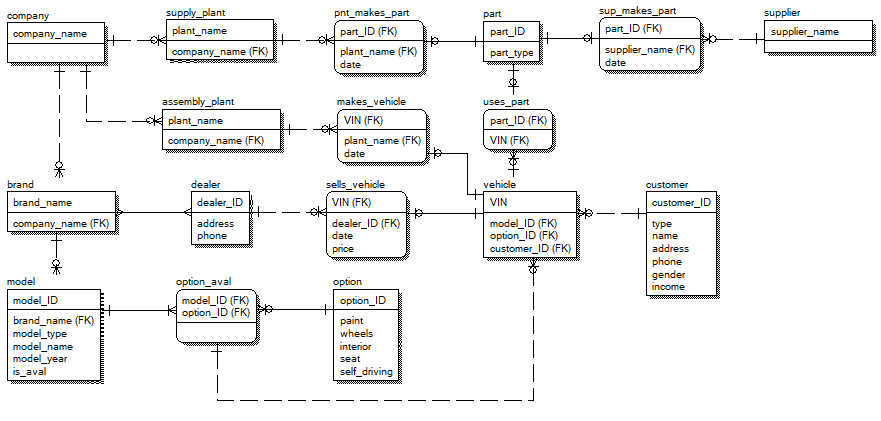
1. 참고 자료

*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

