Database System Project 2

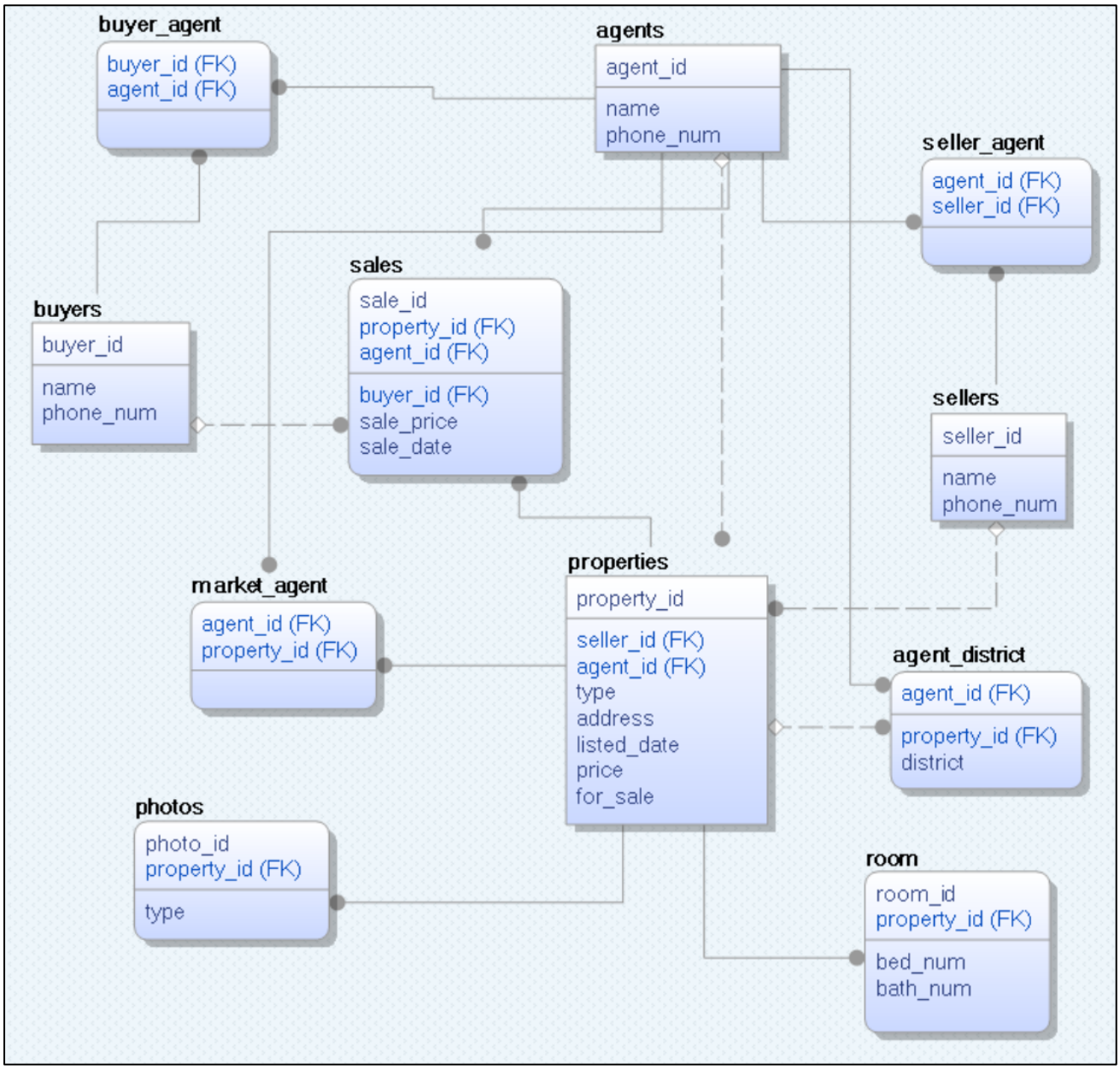
서강대학교
SOGANG UNIVERSITY

분반 : 1 분반

학번 : 20212020

이름 : 박민준

1. BCNF Decomposition



- Properties 테이블의 초기 상태

속성: PropertyID (PK), Address, District, Price, PropertyType, AgentID, SellerID, ListedDate, ForSale

- 함수적 종속성

PropertyID → Address, District, Price, PropertyType, AgentID, SellerID, ListedDate, ForSale

District, AgentID → PropertyID (이것은 비정규 함수 종속이다.)

- 이 함수적 종속성 중에서 District, AgentID → PropertyID 는 부분 함수 종속성을 나타내며 BCNF 위반이다. BCNF 조건은 모든 비자명 함수적 종속성 X → Y 에 대해 X 가 슈퍼 키여야

한다는 것이다. 하지만 District, AgentID → PropertyID 에서 District 와 AgentID 는 슈퍼 키가 아니므로 BCNF 를 위반한다.

- 즉, District 와 AgentID 가 부분적으로 결정자 역할을 하고 있으므로 BCNF 위반이다. 부분 함수 종속성이 존재하므로 PropertyID 에 대해 모든 속성이 결정되지 않는다. 비정규 함수 종속성을 제거하기 위해 District 와 AgentID 를 분리한다. 그리고, District, AgentID → PropertyID 의 종속성을 제거하기 위해 새로운 테이블을 생성한다.

- AgentDistrict 테이블

- 속성: AgentID, District (PK)

- 함수적 종속성: District, AgentID → PropertyID (PK)

- BCNF 로 분해하는 과정에서 District 와 AgentID 를 사용하여 새로운 테이블을 만들어야 한다. 이를 통해 부분 함수 종속성을 제거하고 테이블을 BCNF 형태로 변환한다. 정규화 후 테이블은 다음과 같다.

- Properties 테이블

속성: PropertyID (PK), Address, Price, PropertyType, SellerID, ListedDate, ForSale

- AgentDistrict 테이블

속성: AgentID (FK), District (PK)

- 해당 테이블은 특정 에이전트가 담당하는 지역 정보를 저장하는 테이블이다. 각 에이전트의 고유 ID 와 해당 에이전트가 담당하는 지역을 포함한다.

- 이 과정은 테이블이 BCNF 형태로 변환되었음을 나타낸다. 결과적으로, 모든 함수 종속성이 후보 키에 의해 결정되므로 BCNF 를 만족한다.

- 이 정규화 과정은 데이터 무결성을 유지하고 데이터베이스의 효율성을 높이는 데 기여한다. BCNF 정규화는 이러한 이상 현상을 제거하고 데이터베이스의 구조를 개선하는 중요한 과정이다. 전체 테이블을 요약하면 다음과 같다.

1) Agents 테이블

속성: AgentID (PK), AgentName, PhoneNumber

2) Buyers 테이블

속성: BuyerID (PK), BuyerName, ContactInfo

3) Sellers 테이블

속성: SellerID (PK), SellerName, ContactInfo

4) Properties 테이블

속성: PropertyID (PK), Address, Price, PropertyType, SellerID, ListedDate, ForSale

5) Photos 테이블

속성: PhotoID (PK), PropertyID (FK), PhotoType

6) Rooms 테이블

속성: RoomID (PK), PropertyID (FK), BedroomCount, BathroomCount

7) Sales 테이블

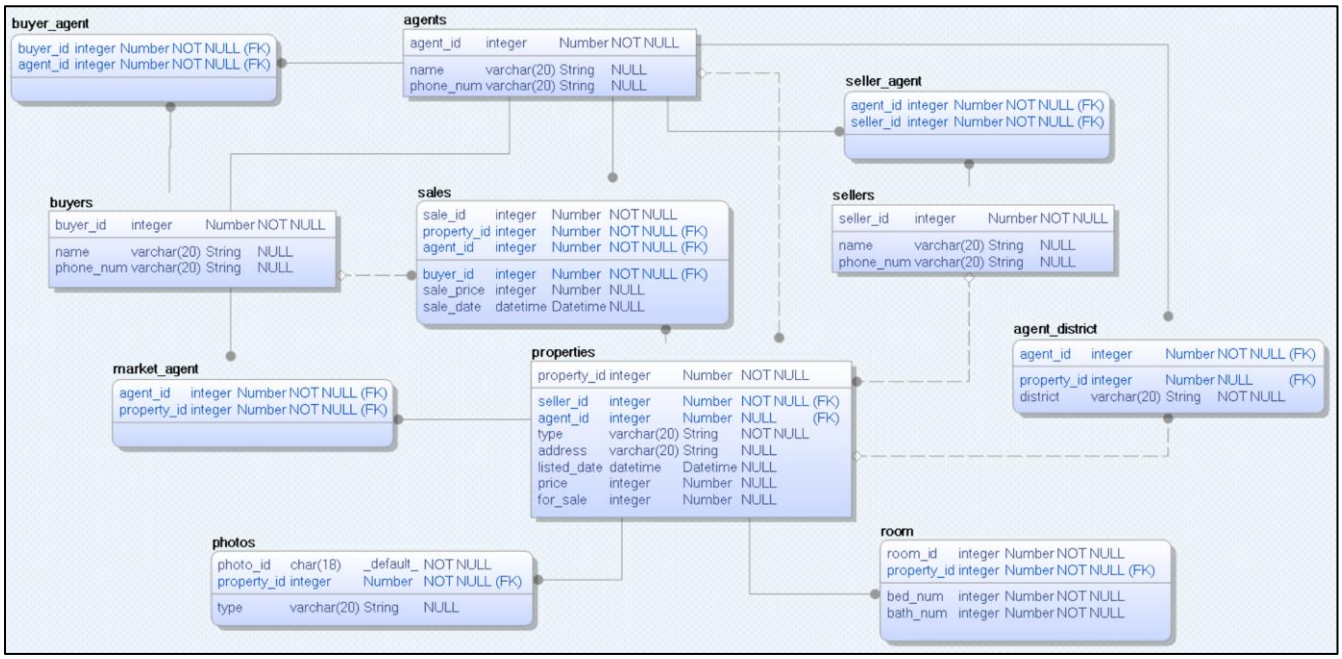
속성: SaleID (PK), PropertyID (FK), SalePrice, SaleDate, BuyerID (FK), AgentID (FK)

8) AgentDistrict 테이블

속성: AgentID (FK), District (PK)

- 이로써 모든 테이블이 BCNF 를 만족하게 되었다. BCNF 분해는 데이터베이스의 정규화를 통해 데이터 무결성과 일관성을 보장하기 위한 과정이다. Properties 테이블의 AgentID 와 District 속성 사이에 부분 함수 종속성이 존재했기 때문에, 이를 해결하기 위해 BCNF 분해를 수행하여 AgentDistrict 테이블을 생성했다. 이로써 각 에이전트가 담당하는 지역 정보를 분리하여 저장하게 되었다.

2. Physical Schema diagram



1) agents 테이블

agent_id: 에이전트의 고유 ID (Primary Key, INTEGER, NOT NULL)

name: 에이전트의 이름 (VARCHAR(20), NULL 허용)

phone_num: 에이전트의 전화번호 (VARCHAR(20), NULL 허용)

- 에이전트의 고유 ID 를 Primary Key 로 설정하여 각 에이전트를 고유하게 식별한다. 에이전트의 이름과 전화번호는 VARCHAR(20) 형식으로 저장하며, NULL 값을 허용한다.

2) buyers 테이블

buyer_id: 구매자의 고유 ID (Primary Key, INTEGER, NOT NULL)

name: 구매자의 이름 (VARCHAR(20), NULL 허용)

phone_num: 구매자의 전화번호 (VARCHAR(20), NULL 허용)

- 구매자의 고유 ID 를 Primary Key 로 설정하여 각 구매자를 고유하게 식별한다. 구매자의 이름과 전화번호는 VARCHAR(20) 형식으로 저장하며, NULL 값을 허용한다.

3) sellers 테이블

seller_id: 판매자의 고유 ID (Primary Key, INTEGER, NOT NULL)

name: 판매자의 이름 (VARCHAR(20), NULL 허용)

phone_num: 판매자의 전화번호 (VARCHAR(20), NULL 허용)

- 판매자의 고유 ID 를 Primary Key 로 설정하여 각 판매자를 고유하게 식별한다. 판매자의 이름과 전화번호는 VARCHAR(20) 형식으로 저장하며, NULL 값을 허용한다.

4) properties 테이블

property_id: 부동산의 고유 ID (Primary Key, INTEGER, NOT NULL)

seller_id: 판매자의 ID (Foreign Key, INTEGER, NOT NULL)

agent_id: 에이전트의 ID (Foreign Key, INTEGER, NOT NULL)

type: 부동산 유형 (VARCHAR(20), NOT NULL)

address: 부동산 주소 (VARCHAR(50), NULL 허용)

listed_date: 부동산 목록에 등록된 날짜 (DATETIME, NULL 허용)

price: 부동산 가격 (INTEGER, NULL 허용)

district: 구역 (VARCHAR(20), NULL 허용)

for_sale: 판매 여부 (BOOLEAN, NULL 허용)

- 부동산의 고유 ID 를 Primary Key 로 설정하여 각 부동산을 고유하게 식별한다. 외래 키로 seller_id 와 agent_id 를 참조하여 각 부동산이 특정 판매자와 에이전트에 연결되도록 한다. 부동산의 유형, 주소, 가격, 구역 등 다양한 속성들은 각각 적절한 데이터 타입으로 설정된다.

5) photos 테이블

photo_id: 사진의 고유 ID (Primary Key, CHAR(18), NOT NULL)

property_id: 부동산의 ID (Foreign Key, INTEGER, NOT NULL)

type: 사진 유형 (VARCHAR(20), NULL 허용)

- 사진의 고유 ID 를 Primary Key 로 설정하여 각 사진을 고유하게 식별한다. 외래 키로 property_id 를 참조하여 각 사진이 특정 부동산에 연결되도록 한다. 사진의 유형은 VARCHAR(20) 형식으로 저장하며, NULL 값을 허용한다.

6) rooms 테이블

room_id: 방의 고유 ID (Primary Key, INTEGER, NOT NULL)

property_id: 부동산의 ID (Foreign Key, INTEGER, NOT NULL)

bed_num: 침실 수 (INTEGER, NOT NULL)

bath_num: 욕실 수 (INTEGER, NOT NULL)

- 방의 고유 ID 를 Primary Key 로 설정하여 각 방을 고유하게 식별한다. 외래 키로 property_id 를 참조하여 각 방이 특정 부동산에 연결되도록 한다. 침실 수와 욕실 수는 INTEGER 형식으로 저장하며, NULL 값을 허용하지 않는다.

7) sales 테이블

sale_id: 판매의 고유 ID (Primary Key, INTEGER, NOT NULL)

property_id: 부동산의 ID (Foreign Key, INTEGER, NOT NULL)

sale_price: 판매 가격 (INTEGER, NULL 허용)

sale_date: 판매 날짜 (DATETIME, NULL 허용)

buyer_id: 구매자의 ID (Foreign Key, INTEGER, NOT NULL)

agent_id: 에이전트의 ID (Foreign Key, INTEGER, NOT NULL)

- 판매의 고유 ID 를 Primary Key 로 설정하여 각 판매를 고유하게 식별한다. 외래 키로 property_id, buyer_id, agent_id 를 참조하여 각 판매가 특정 부동산, 구매자, 에이전트에 연결되도록 한다. 판매 가격과 판매 날짜는 각각 INTEGER 와 DATETIME 형식으로 저장되며, NULL 값을 허용한다.

8) agent_district 테이블

agent_id: 에이전트의 ID (Foreign Key, INTEGER, NOT NULL)

district: 구역 (VARCHAR(20), NOT NULL)

- 이 테이블은 BCNF 분해를 통해 새로 생성된 테이블로, 에이전트와 구역 간의 관계를 관리한다. 외래 키로 agent_id 를 참조하며, 구역은 VARCHAR(20) 형식으로 저장된다. 에이전트 ID 와 구역 모두 NULL 값을 허용하지 않는다.

9) seller_agent 테이블

agent_id: 에이전트의 ID (Foreign Key, INTEGER, NOT NULL)

seller_id: 판매자의 ID (Foreign Key, INTEGER, NOT NULL)

- 이 테이블은 에이전트와 판매자 간의 관계를 관리한다. 외래 키로 agent_id 와 seller_id 를 참조하여 각 관계가 특정 에이전트와 판매자에 연결되도록 한다.

10) market_agent 테이블

agent_id: 에이전트의 ID (Foreign Key, INTEGER, NOT NULL)

property_id: 부동산의 ID (Foreign Key, INTEGER, NOT NULL)

- 이 테이블은 에이전트와 부동산 간의 관계를 관리한다. 외래 키로 agent_id 와 property_id 를 참조하여 각 관계가 특정 에이전트와 부동산에 연결되도록 한다.

11) buyer_agent 테이블

buyer_id: 구매자의 ID (Foreign Key, INTEGER, NOT NULL)

agent_id: 에이전트의 ID (Foreign Key, INTEGER, NOT NULL)

- 이 테이블은 구매자와 에이전트 간의 관계를 관리한다. 외래 키로 buyer_id 와 agent_id 를 참조하여 각 관계가 특정 구매자와 에이전트에 연결되도록 한다.

- 해당 물리적 스키마 다이어그램은 데이터베이스 설계의 핵심 요소를 담고 있다. 모든 테이블은 BCNF 정규화 과정을 거쳐 중복과 이상현상을 최소화하였으며, 각 테이블의 속성들은

명확한 데이터 타입과 제약 조건을 가지고 있다. 특히, agent_district 테이블은 BCNF 분해 과정에서 새롭게 생성된 테이블로, agent_id 를 외래키로 가지며 district 와의 관계를 명확히 하고 있다. 이러한 설계를 통해 데이터베이스는 보다 효율적이고 안정적인 데이터 관리가 가능하게 되었다.

3. ODBC implementation within MySQL

```
CREATE DATABASE project;
USE project;

CREATE TABLE Agents (AgentID VARCHAR(10) NOT NULL, AgentName VARCHAR(100), PhoneNumber VARCHAR(15), PRIMARY KEY
(AgentID));
CREATE TABLE Buyers (BuyerID VARCHAR(10) NOT NULL, BuyerName VARCHAR(100), ContactInfo VARCHAR(150), PRIMARY KEY (BuyerID));
CREATE TABLE Sellers (SellerID VARCHAR(10) NOT NULL, SellerName VARCHAR(100), ContactInfo VARCHAR(150), PRIMARY KEY (SellerID));
CREATE TABLE Properties (PropertyID VARCHAR(10) NOT NULL, Address VARCHAR(150), District VARCHAR(50), Price DECIMAL(18,2),
PropertyType ENUM('studio', 'one-bedroom', 'multi-bedroom', 'detached') NOT NULL, AgentID VARCHAR(10) NOT NULL, SellerID
VARCHAR(10) NOT NULL, ListedDate DATE, ForSale BOOLEAN, PRIMARY KEY (PropertyID), FOREIGN KEY (AgentID) REFERENCES
Agents(AgentID), FOREIGN KEY (SellerID) REFERENCES Sellers(SellerID));

INSERT INTO Agents (AgentID, AgentName, PhoneNumber) VALUES ('A001', 'Alice Johnson', '010-1234-5678');
INSERT INTO Agents (AgentID, AgentName, PhoneNumber) VALUES ('A002', 'Bob Smith', '010-2345-6789');
INSERT INTO Agents (AgentID, AgentName, PhoneNumber) VALUES ('A003', 'Charlie Brown', '010-3456-7890');
INSERT INTO Agents (AgentID, AgentName, PhoneNumber) VALUES ('A004', 'Diana Ross', '010-4567-8901');
INSERT INTO Agents (AgentID, AgentName, PhoneNumber) VALUES ('A005', 'Edward King', '010-5678-9012');

INSERT INTO Buyers (BuyerID, BuyerName, ContactInfo) VALUES ('B001', 'David Johnson', 'david.johnson@example.com');
INSERT INTO Buyers (BuyerID, BuyerName, ContactInfo) VALUES ('B002', 'Eva Green', 'eva.green@example.com');
INSERT INTO Buyers (BuyerID, BuyerName, ContactInfo) VALUES ('B003', 'Frank White', 'frank.white@example.com');
INSERT INTO Buyers (BuyerID, BuyerName, ContactInfo) VALUES ('B004', 'George Harrison', 'george.harrison@example.com');
INSERT INTO Buyers (BuyerID, BuyerName, ContactInfo) VALUES ('B005', 'Helen Hunt', 'helen.hunt@example.com');

INSERT INTO Sellers (SellerID, SellerName, ContactInfo) VALUES ('S001', 'Grace Hopper', 'grace.hopper@example.com');
INSERT INTO Sellers (SellerID, SellerName, ContactInfo) VALUES ('S002', 'Henry Ford', 'henry.ford@example.com');

$$$
DROP TABLE IF EXISTS AgentDistrict;
DROP TABLE IF EXISTS Sales;
DROP TABLE IF EXISTS Rooms;
DROP TABLE IF EXISTS Photos;
DROP TABLE IF EXISTS Properties;
DROP TABLE IF EXISTS Sellers;
DROP TABLE IF EXISTS Buyers;
DROP TABLE IF EXISTS Agents;
```

- CRUD.txt 파일은 데이터베이스와 테이블을 생성하고, 데이터를 삽입한 후, 드롭 명령어를 포함하는 SQL 스크립트를 포함하고 있다. 해당 파일에 대한 예시는 위 사진과 같다.

- 먼저, project 라는 데이터베이스를 생성하고, CREATE TABLE query 를 통해 각각의 테이블을 생성한다.

- 테이블 생성 후, INSERT INTO query 를 사용해 데이터를 삽입하는데, Agents, Buyers, Sellers, Properties, Photos, Rooms, Sales 테이블에 대한 데이터가 삽입된다.

예시: INSERT INTO Agents (AgentID, AgentName, PhoneNumber) VALUES ('A001', 'Alice Johnson', '010-1234-5678');

- 마지막으로 DROP query 를 통해 데이터 삭제를 한다. CRUD.txt 파일에서는 사용자의 입력을 받기 전에는 TABLE 이 DROP 되면 안 되기 때문에 '$$$'로 DROP 과 나머지 부분을 구분하였다. DROP 은 데이터베이스와 테이블을 삭제하는 명령어이다.

예시: DROP TABLE IF EXISTS AgentDistrict; DROP DATABASE project;

- CRUD.txt 파일 생성 후, cpp 파일을 작성하여 프로그램을 실행하게 되는데, 작성한 20212020.cpp 파일은 MySQL 데이터베이스에 연결하고, CRUD.txt 파일을 처리하며, 다양한 유형의 쿼리를 수행하는 C++ 프로그램이다.

- 주요 기능과 각 유형(TYPE)의 처리 방식을 설명하겠다.

1) 주요 기능을 담당하는 함수

(1) connectDatabase() 함수

```
// MySQL 연결 및 설정
MYSQL* connectDatabase() {
    MYSQL* connection = mysql_init(NULL);
    if (connection == NULL) {
        printf("mysql_init() error!");
        return NULL;
    }
    connection = mysql_real_connect(connection, host, user, pw, NULL, 3306, NULL, 0);
    if (connection == NULL) {
        printf("%d ERROR : %s\n", mysql_errno(connection), mysql_error(connection));
    }
    return connection;
}
```

- MySQL 데이터베이스에 연결하는 함수이다. 성공 시 데이터베이스 연결 객체를 반환한다.

(2) processCrudFile() 함수

```
// CRUD 파일 처리 함수
void processCrudFile(MYSQL* connection, const char* filename) {
    ifstream file(filename);
    if (!file.is_open()) {
        printf("Cannot open CRUD file.\n");
        return;
    }
    string line;
    bool isDropSection = false;

    while (getline(file, line)) {
        if (line == "$$$") {
            isDropSection = true;
            continue;
        }

        if (isDropSection) {
            if (!line.empty()) {
                dropCommands.push_back(line);
            }
        }
        else {
            if (!line.empty()) {
                if (mysql_query(connection, line.c_str())) {
                    printf("Query Error: %s\n", mysql_error(connection));
                    printf("Query: %s\n", line.c_str()); // 오류가 발생한 쿼리 출력
                }
            }
        }
    }
    file.close();
}
```

- CRUD.txt 파일을 읽어 테이블 생성, 데이터 삽입 등의 작업을 수행한다. $$$ 이후의 DROP 명령어는 따로 저장되어 이후 사용자로부터 0 을 입력받으면 한꺼번에 수행되도록 코드를 작성하였다.

(3) executeQuery() 함수

```
// 쿼리 수행 함수
void executeQuery(MYSQL* connection, const string& query, bool printHeader) {
    if (mysql_query(connection, query.c_str())) {
        cerr << "Query Error: " << mysql_error(connection) << '\n';
        return;
    }
    MYSQL_RES* res = mysql_store_result(connection);
    if (res) {
        MYSQL_ROW row;
        unsigned int num_fields = mysql_num_fields(res);
        MYSQL_FIELD* fields = mysql_fetch_fields(res);

        if (printHeader) {
            // 컬럼 헤더 출력
            for (unsigned int i = 0; i < num_fields; i++) {
                cout << left << setw(32) << ("[" + string(fields[i].name) + "]");
            }
            cout << '\n';
        }

        // 데이터 출력
        while ((row = mysql_fetch_row(res))) {
            for (unsigned int i = 0; i < num_fields; i++) {
                cout << left << setw(32) << (row[i] ? row[i] : "NULL");
            }
            cout << '\n';
        }
        mysql_free_result(res);
    }
}
```

- 주어진 쿼리를 실행하고 결과를 출력하는 함수이다. printHeader 파라미터를 통해 컬럼 헤더를 출력할지 여부를 설정할 수 있다.

2) 각 유형(TYPE)의 처리 방식

(1) TYPE 1: Mapo 지역의 판매 중인 부동산 조회

- 목적: Mapo 지역에서 판매되고 있는 부동산을 조회한다.

```
---------- SELECT QUERY TYPES ----------

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

Enter choice: 1

[Address]                      [District]
Mapo-gu 114-52                 Mapo
Mapo-gu 195-77                 Mapo
Mapo-gu 678-22                 Mapo
Mapo-gu 234-19                 Mapo
Mapo-gu 156-48                 Mapo
Mapo-gu 119-87                 Mapo
Mapo-gu 117-88                 Mapo
Mapo-gu 145-37                 Mapo
Mapo-gu 149-22                 Mapo
Mapo-gu 110-74                 Mapo
Mapo-gu 148-55                 Mapo
Mapo-gu 143-60                 Mapo
Mapo-gu 112-33                 Mapo
```

- 처음 프로그램을 실행하면 다음과 같이 QUERY TYPE 를 SELECT 하는 menu 가 출력되고, 사용자가 1 을 입력하면 Properties 의 주소가 Mapo 인 지역에서 ForSale=1 인 부동산을 조회한다.

```
------- Subtypes in TYPE 1 -------
       1. TYPE 1-1

Enter choice: 1

[PropertyID]                   [District]                     [Price]
P017                           Mapo                           1100000000.00
P019                           Mapo                           1200000000.00
P025                           Mapo                           1160000000.00
P032                           Mapo                           1170000000.00
P034                           Mapo                           1210000000.00
P040                           Mapo                           1080000000.00

---------- SELECT QUERY TYPES ----------

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

Enter choice:
```

- 이후 TYPE 1 의 Subtypes 를 고르는 menu 가 출력되고, 마찬가지로 사용자가 1 을 입력하면 Properties 의 address 가 Mapo 인 지역에서 ForSale=1 이고 가격이 10 억에서 15 억 사이인 부동산을 조회하게 된다.

- Subtypes 결과가 출력됨과 동시에 다시 초기 QUERY TYPES 를 SELECT 하는 menu 로 돌아오게 된다. 이는 사용자가 0 을 누르기 전까지 무한하게 반복된다.

(2) TYPE 2: 특정 구역의 판매 중인 부동산 조회

- 목적: 여러 구역에서 판매 중인 부동산을 조회한다.

```
---------- SELECT QUERY TYPES ----------

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

Enter choice: 2

[Address]                       [District]
Mapo-gu 114-52                  Mapo
Songpa-gu 343-76                Songpa
Gangdong-gu 527-11              Gangdong
Mapo-gu 195-77                  Mapo
Mapo-gu 678-22                  Mapo
Dongdaemun-gu 146-33            Dongdaemun
Seodaemun-gu 321-45             Seodaemun
Mapo-gu 234-19                  Mapo
Mapo-gu 156-48                  Mapo
Mapo-gu 119-87                  Mapo
Mapo-gu 117-88                  Mapo
Mapo-gu 145-37                  Mapo
Mapo-gu 149-22                  Mapo
Mapo-gu 110-74                  Mapo
Yongsan-gu 213-56               Yongsan
Mapo-gu 148-55                  Mapo
```

- 8 학군에 해당하는 지역에서 ForSale=1 인 부동산을 조회한다.

```
------- Subtypes in TYPE 2 -------
       1. TYPE 2-1

Enter choice: 1

[Address]                       [BedroomCount]                  [BathroomCount]
Seodaemun-gu 321-45             4                               3
Yongsan-gu 213-56               4                               3
Mapo-gu 143-60                  4                               3
Yeongdeungpo-gu 124-75          4                               3
Jung-gu 182-77                  4                               3

---------- SELECT QUERY TYPES ----------

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

Enter choice:
```

TYPE 2-1 은 지정된 구역에서 ForSale=1 이고 침실이 4 개 이상, 욕실이 2 개 이상인 부동산을 조회한다. 이 함수는 사용자가 선택한 옵션에 따라 적절한 쿼리를 실행하고 결과를 출력한다.

(3) TYPE 3: 에이전트의 판매 실적 조회

- 목적: 특정 연도의 에이전트 판매 실적을 조회한다.

```
---------- SELECT QUERY TYPES ----------

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

Enter choice:
3

[AgentName]                     [TotalSales]
Diana Ross                      2430000000.00

------- Subtypes in TYPE 3 -------
       1. TYPE 3-1
       2. TYPE 3-2

Enter choice:
```

- 기본 쿼리: 2022 년 총 판매 금액 기준 가장 많은 판매를 한 에이전트를 조회한다.

```
------- Subtypes in TYPE 3 -------
       1. TYPE 3-1
       2. TYPE 3-2

Enter choice: 1

------- TYPE 3-1 --------

** Then find the top k agents in the year 2023 by total won value. **
 Which K? : 1

[AgentName]                    [TotalSales]
Ivan Martin                    1190000000.00
```

- TYPE 3-1 은 2023 년 총 판매 금액 기준 상위 K 명의 에이전트를 조회한다. 사용자 입력을 통해 K 값을 받아 쿼리를 동적으로 생성한다.

```
------- Subtypes in TYPE 3 -------
       1. TYPE 3-1
       2. TYPE 3-2

Enter choice: 2

[AgentName]                     [TotalSales]
Diana Ross                      800000000.00
```

- TYPE 3-2 는 2021 년 하위 10% 에이전트를 조회한다. 전체 에이전트 수의 10%를 계산한 후, 해당 수 만큼의 하위 에이전트를 조회하는 쿼리를 생성한다. 이는 countQuery 결과를 기반으로 하위 10% 에이전트를 조회하는 쿼리를 이용한다.

(4) TYPE 4: 에이전트의 평균 판매 가격 및 시장에서의 평균 시간 조회

- 목적: 에이전트의 연도별 평균 판매 가격과 시장에 나와 있는 평균 시간을 조회한다.

```
---------- SELECT QUERY TYPES ----------

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

Enter choice: 4

[AgentName]                    [AvgSalePrice]                 [AvgMarketTime]
Charlie Brown                  1130000000.000000              78.0000
Diana Ross                     1215000000.000000              79.5000
Edward King                    930000000.000000               73.0000
Fiona Lee                      1000000000.000000              455.0000
George Clark                   1100000000.000000              440.0000
Hannah Scott                   1250000000.000000              445.0000
Ivan Martin                    920000000.000000               436.0000
Jack Wilson                    870000000.000000               435.0000

------- Subtypes in TYPE 4 -------
       1. TYPE 4-1
       2. TYPE 4-2

Enter choice:
```

- 기본 쿼리는 2022 년 에이전트의 평균 판매 가격과 시장에 나와 있는 평균 시간을 조회한다.

```
------- Subtypes in TYPE 4 -------
       1. TYPE 4-1
       2. TYPE 4-2

Enter choice: 1

[AgentName]                    [MaxSalePrice]
Edward King                    1150000000.00
Fiona Lee                      1010000000.00
George Clark                   1090000000.00
Hannah Scott                   1050000000.00
Ivan Martin                    1190000000.00
Jack Wilson                    970000000.00
```

- TYPE 4-1 은 2023 년 최대 판매 가격을 조회한다.

```
------- Subtypes in TYPE 4 -------
       1. TYPE 4-1
       2. TYPE 4-2

Enter choice: 2

[AgentName]                    [LongestMarketTime]
Alice Johnson                  71
Bob Smith                      72
Charlie Brown                  83
Diana Ross                     84
Edward King                    85
Fiona Lee                      455
George Clark                   442
Hannah Scott                   445
Ivan Martin                    444
Jack Wilson                    445
```

- TYPE 4-2 는 시장에 나와 있는 가장 긴 시간을 조회한다.

(5) TYPE 5: 부동산 유형별 가장 비싼 부동산의 사진 조회

- 목적: 각 부동산 유형에서 가장 비싼 부동산의 사진을 조회한다.

```
---------- SELECT QUERY TYPES ----------

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

Enter choice: 5

[PhotoID]                      [PropertyID]                   [PropertyType]                 [PhotoType]

PH025                          P020                           studio                         Exterior

PH030                          P025                           one-bedroom                    Interior

PH035                          P030                           multi-bedroom                  Exterior

PH019                          P014                           detached                       Exterior
```

- TYPE 5 는 가장 비싼 studio 부동산, one-bedroom 부동산, multi-bedroom 부동산, detached 부동산의 사진을 조회한다.

- 즉, 해당 함수는 각 유형별로 가장 비싼 부동산의 사진을 조회하고, 결과를 출력한다. 첫 번째 쿼리에서는 컬럼 헤더를 출력하고, 이후 쿼리에서는 컬럼 헤더를 생략하여 연속적으로 결과를 표시한다.

(6) TYPE 6: 새로운 판매 기록 추가

- 목적: 새로운 판매 기록을 추가한다.

```
---------- SELECT QUERY TYPES ----------

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

Enter choice: 6


TYPE 6 - Record a sale:

Enter SaleID: SA026
Enter PropertyID: P026
Enter SalePrice: 1150000000
Enter SaleDate (YYYY-MM-DD): 2023-01-25
Enter BuyerID: B004
Enter AgentID: A003
```

- TYPE 6 쿼리 함수는 사용자로부터 판매 기록에 필요한 정보를 입력받는다. 입력된 데이터를 기반으로 Sales 테이블에 삽입하는 쿼리를 생성하고 실행한다.

```
---------- SELECT QUERY TYPES ----------

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

Enter choice: 6


TYPE 6 - Record a sale:

Enter SaleID: SA026
Enter PropertyID: P023
Enter SalePrice: 1560000000
Enter SaleDate (YYYY-MM-DD): 2022-06-19
Enter BuyerID: B002
Enter AgentID: A001
Query Error: Duplicate entry 'SA026' for key 'sales.PRIMARY'
```

- 만약 동일한 SaleID 를 추가한다면 다음과 같이 PRIMARY KEY Error 를 출력한다.

(7) TYPE 7: 새로운 에이전트 추가

- 목적: 새로운 에이전트를 추가한다.

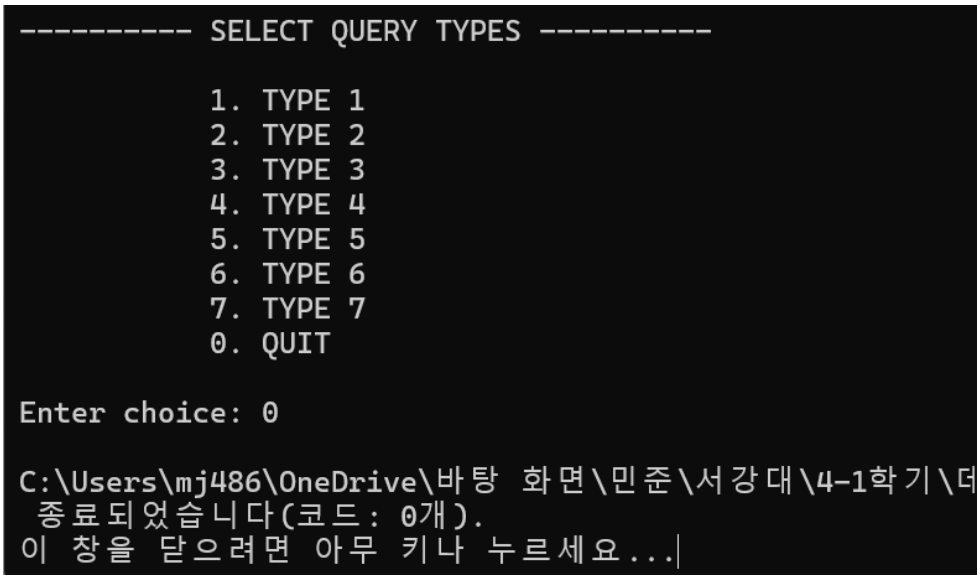
```
---------- SELECT QUERY TYPES ----------

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

Enter choice: 7


TYPE 7 - Add a new agent:

Enter AgentID: A011
Enter AgentName: Frank White
Enter PhoneNumber: 010-1234-5678
```

- 해당 TYPE 은 사용자로부터 에이전트 정보를 입력받는다. 입력된 데이터를 기반으로 Agents 테이블에 삽입하는 쿼리를 생성하고 실행한다.

```
---------- SELECT QUERY TYPES ----------

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

Enter choice: 7


TYPE 7 - Add a new agent:

Enter AgentID: A011
Enter AgentName: MINJUN PARK
Enter PhoneNumber: 010-5471-1238
Query Error: Duplicate entry 'A011' for key 'agents.PRIMARY'
```

- 만약 동일한 AgentID 를 추가한다면 다음과 같이 PRIMARY KEY Error 를 출력한다.

(8) TYPE 0: 프로그램 종료



- 마지막으로 사용자가 0 을 입력하면 CRUD.txt 파일의 DROP TABLE, DROP DATABASE query 가 전부 실행되고 난 뒤에 프로그램이 종료된다. MySQL Workbench 를 확인해보면 해당 DATABASE 와 TABLE 들이 전부 DROP 된 것을 확인할 수 있다.