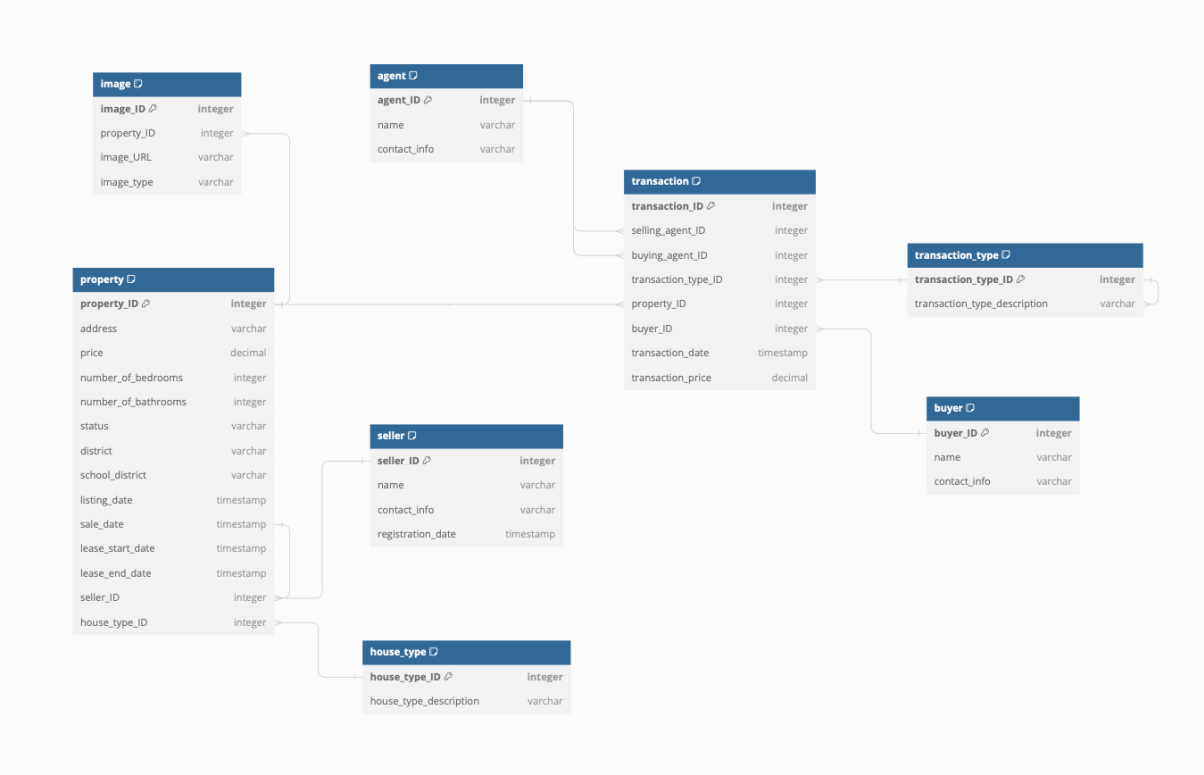
Project 2

과목: 데이터베이스시스템 전공: 컴퓨터공학과 학번: 20211547 이름: 신지원

1. BCNF Decomposition



BCNF를 만족하기 위해서는, 해당 schema의 모든 functional dependency α → β에 대하여, 1. α가 superkey 여야 하고 , 2. trivial하지 않은 α → β가 존재하지 않아야 한다. 이에 부합할 때, 해당 relation schema는 BCNF를 만족한다고 말할 수 있으며, 부합하지 않을 시엔 해당 relation 분해해야 한다. 기존 schema가 R이라 하고, R이 BCNF를 위반하도록 하는 functional dependency가 α → β일 때, R을 (α ∪ β)와 (R – (β – α)) 로 분해해야 한다.

1-1. agent

R(Schema)	agent(agent_ID, name, contact_info)
F+(FD Closure)	agent_ID -> { name, contact_info }

agent_ID는 table 의 primary key 이며, name과 contact_info를 결정하기 때문에 해다 functional dependency는 BCNF를 위배하지 않는다.

∴ agent schema는 BCNF를 만족한다.

1-2. house_type

R(Schema)	house_type(house_type_ID, house_type_description)
F+(FD Closure)	house_type_ID -> { house_type_description }

house_type_ID는 table 의 primary key 이며, house_type_description를 결정하기 때문에 해다 functional dependency는 BCNF를 위배하지 않는다.

∴ house_type schema 는 BCNF를 만족한다.

1-3. seller

R(Schema)	seller(seller_ID, name, contact_info, registration_date)
F+(FD Closure)	seller_ID -> { name, contact_info, registration_date }

seller_ID는 table 의 primary key 이며, name, contact_info, registration_date를 결정하기 때문에 해다 functional dependency는 BCNF를 위배하지 않는다.

∴ seller schema 는 BCNF를 만족한다.

1-4. buyer

R(Schema)	buyer(buyer_ID, name, contact_info)
F+(FD Closure)	buyer_ID -> { name, contact_info }

buyer_ID는 table 의 primary key 이며, name, name과 contact_info를 결정하기 때문에 해다 functional dependency는 BCNF를 위배하지 않는다.

∴ buyer schema 는 BCNF를 만족한다.

1-5. property

R(Schema)	property(property_ID, address, price, number_of_bedrooms, number_of_bathrooms, status, district, school_district, listing_date, sale_date, lease_start_date, lease_end_date, seller_ID, house_type_ID)

F+(FD Closure)	property_ID -> { address, price, number_of_bedrooms, number_of_bathrooms, status, district, school_district, listing_date, sale_date, lease_start_date, lease_end_date, seller_ID, house_type_ID }

property_ID는 table 의 primary key 이며, 모든 속성을 결정하기 때문에 해당 functional dependency는 BCNF를 위배하지 않는다.

∴ property schema 는 BCNF를 만족한다.

1-6. transaction_type

R(Schema)	transaction_type(transaction_type_ID, transaction_type_description)
F+(FD Closure)	transaction_type_ID -> { transaction_type_description }

transaction_type_ID는 table 의 primary key 이며, transaction_type_description을 결정하기 때문에 해당 functional dependency는 BCNF를 위배하지 않는다.

∴ transaction_type schema 는 BCNF를 만족한다.

1-7. transaction

R(Schema)	transaction(transaction_ID, selling_agent_ID, buying_agent_ID, transaction_type_ID, property_ID, buyer_ID, transaction_date, transaction_price)
F+(FD Closure)	transaction_ID -> { selling_agent_ID, buying_agent_ID, transaction_type_ID, property_ID, buyer_ID, transaction_date, transaction_price }

transaction_ID는 table 의 primary key 이며, 모든 속성을 결정하기 때문에 해당 functional dependency는 BCNF를 위배하지 않는다.

∴ transaction schema 는 BCNF를 만족한다.

1-8. image

R(Schema)	image(image_ID, property_ID, image_URL, image_type)
F+(FD Closure)	image_ID -> { property_ID, image_URL, image_type }

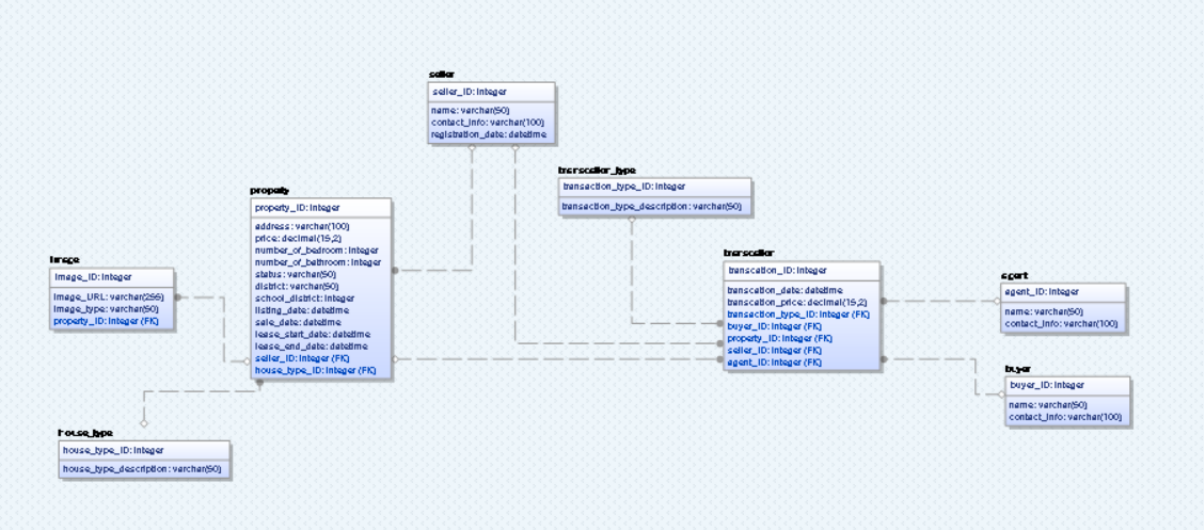
image_ID는 table 의 primary key 이며, 모든 속성을 결정하기 때문에 해당 functional dependency 는 BCNF를 위배하지 않는다.

∴ image schema 는 BCNF를 만족한다.

1.9. 결론

① 모든 table의 primary key는 superkey 이며 ② 각 functional dependency 에서 결정자가 primary key(superkey) 이다. 모든 함수적 종속성은 BCNF의 정의에 맞게 각 table의 primarykey가 결정자로 되어있기 때문에 모든 table 에서 BCNF 를 만족하고 있다.

2. Physical Schema



2-1. agent

- agent_ID (INT, AUTO_INCREMENT, NOT NULL, PRIMARY KEY)
- name (VARCHAR(50))
- contact_info (VARCHAR(100))

2-2. house_type

- house_type_ID (INT, AUTO_INCREMENT, NOT NULL, PRIMARY KEY)
- house_type_description (VARCHAR(50))

2-3. seller

- seller_ID (INT, AUTO_INCREMENT, NOT NULL, PRIMARY KEY)
- name (VARCHAR(50))
- contact_info (VARCHAR(100))
- registration_date (DATE)

2-4. buyer

- buyer_ID (INT, AUTO_INCREMENT, NOT NULL, PRIMARY KEY)
- name (VARCHAR(50))
- contact_info (VARCHAR(100))

2-5. property

- property_ID (INT, AUTO_INCREMENT, NOT NULL, PRIMARY KEY)
- address (VARCHAR(100))
- price (DECIMAL(15, 2))
- number_of_bedrooms (INT)
- number_of_bathrooms (INT)
- status (VARCHAR(50))
- district (VARCHAR(50))
- school_district (INT)
- listing_date (DATE)
- sale_date (DATE)
- lease_start_date (DATE)
- lease_end_date (DATE)
- seller_ID (INT, FOREIGN KEY REFERENCES seller(seller_ID))
- house_type_ID (INT, FOREIGN KEY REFERENCES house_type(house_type_ID))

조건에서 방 개수에 따라 내부나 외관 사진이 보여야 하기 때문에 number_of_bedrooms 를 not null 로 처리할 수 있었지만 house_type_ID 에 의하여 결정할 수 있다고 보았기 때문에 number_of_bedrooms 은 null 일 수 있고, house_type_ID 은 Not null 이어야만 한다고 설계하였다.

2-6. transaction_type

- transaction_type_ID (INT, AUTO_INCREMENT, NOT NULL, PRIMARY KEY)
- transaction_type_description (VARCHAR(50))

2-7. transaction

- transaction_ID (INT, AUTO_INCREMENT, NOT NULL, PRIMARY KEY)
- selling_agent_ID (INT, FOREIGN KEY REFERENCES agent(agent_ID))
- buying_agent_ID (INT, FOREIGN KEY REFERENCES agent(agent_ID))
- transaction_type_ID (INT, FOREIGN KEY REFERENCES transaction_type(transaction_type_ID))
- property_ID (INT, FOREIGN KEY REFERENCES property(property_ID))
- buyer_ID (INT, FOREIGN KEY REFERENCES buyer(buyer_ID))
- transaction_date (DATE)
- transaction_price (DECIMAL(15, 2))

2-8. image

- image_ID (INT, AUTO_INCREMENT, NOT NULL, PRIMARY KEY)
- property_ID (INT, FOREIGN KEY REFERENCES property(property_ID))
- image_URL (VARCHAR(255))
- image_type (VARCHAR(50))

3. CRUD & C++ FILE

CRUD file 에 대해 테이블을 생성하고 데이터를 추가하는 CRUD_1 과 데이터와 테이블을 삭제하는 CRUD_2 로 나누어 프로젝트를 진행하였다.

- CRUD_1

```
-- 테이블 생성
CREATE TABLE agent (
    agent_ID INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    contact_info VARCHAR(100)
);
... 생략

-- 초기 데이터 삽입
INSERT INTO agent (name, contact_info) VALUES
('Agent A', '010-1234-5678'),
('Agent B', '010-8765-4321'),
('Agent C', '010-1357-2468'),
('Agent D', '010-2468-1357'),
('Agent E', '010-1122-3344');
... 생략
```

CRUD1 에서는 각각의 테이블을 모두 생성해주었고 문제에서 요구하는 쿼리문을 만족하기 위하여 각각의 data를 삽입하여 주었다.

- CRUD_2

```
-- 데이터 삭제 및 테이블 삭제

DELETE FROM image WHERE property_ID IN (1, 2);
DELETE FROM transaction WHERE property_ID IN (1, 2);
DELETE FROM property WHERE property_ID IN (1, 2);
DELETE FROM agent WHERE agent_ID IN (1, 2);
DELETE FROM district WHERE district_ID IN (1, 2);

DROP TABLE IF EXISTS image;
DROP TABLE IF EXISTS transaction;
DROP TABLE IF EXISTS property;
```

DROP TABLE IF EXISTS agent;
DROP TABLE IF EXISTS district;

CRUD2 에서는 삽입한 데이터를 삭제하고 테이블 자체를 삭제하기 위하여 DELETE 와 DROP 을 활용하여 구현하였다.

- C++ File

C++ 함수에서는 3개의 함수로 나누어 코드를 구현하였다. 그 전에 sql 과 연결하기 위한 초기화를 진행하였다.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include "mysql.h"
#include <cstring>

#pragma comment(lib, "libmysql.lib")

const char* host = "localhost";
const char* user = "root";
const char* pw = "mysql";
const char* db = "project";

```

위 코드는 mysql 과 정상적으로 연결되어 '`#include "mysql.h"`', '`#pragma comment(lib, "libmysql.lib")`' 의 구문이 코드에 삽입되었음을 나타낸다.

```
void execute_query(MYSQL* connection, const char* query) {
    int state = mysql_query(connection, query);
    if (state == 0) {
        MYSQL_RES* sql_result = mysql_store_result(connection);
        if (sql_result) {
            int num_fields = mysql_num_fields(sql_result);
            MYSQL_ROW sql_row;
            MYSQL_FIELD* fields = mysql_fetch_fields(sql_result);

            for (int i = 0; i < num_fields; i++) {
                printf("%-30s ", fields[i].name);
            }
            printf("\n");

            while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                for (int i = 0; i < num_fields; i++) {
                    printf("%-30s ", sql_row[i] ? sql_row[i] : "NULL");
                }
                printf("\n");
            }
            mysql_free_result(sql_result);
        }
    } else {
        printf("Query execution failed: %s\n", mysql_error(connection));
    }
}
```

excute_query() 는 쿼리를 실행하고 결과를 저장하는 기능을 하는 함수다.
mysql_query(connection, query) 함수를 사용하여 데이터베이스 연결하며 성공할 시엔 mysql_store_result(connection) 함수가 호출되어 쿼리 실행 결과를 MYSQL_RES 구조체로 반환한다.

mysql_num_fields(sql_result) 함수로는 결과 테이블의 열 개수, mysql_fetch_fields(sql_result) 함수로는 결과 테이블의 열 정보를 MYSQL_FIELD 배열로 반환한다.

모든 결과 테이블을 돌고 난 후엔 mysql_free_result(sql_result) 함수를 통하여 결과를 사용한 후 메모리를 해제한다.

각각의 함수에 대한 결과를 0, null, 기타 오류처리를 포함하여 case 를 구분해주어 코드를 구현하였다.

```
39
40  void execute_file_queries(MYSQL* connection, const char* filename) {
41      FILE* fp = fopen(filename, "r");
42      if (fp == NULL) {
43          printf("File open error: %s\n", filename);
44          return;
45      }
46      char query[1024];
47      while (fgets(query, sizeof(query), fp)) {
48          execute_query(connection, query);
49      }
50      fclose(fp);
51  }
52
```

Execute_file_queries 함수를 통해서 file 을 관리한다. 특히나 이 프로젝트에서는 CRUD.txt 를 사용하여 MYSQL 을 연결하도록 요구하기 때문에 위와 같은 함수를 사용하여 CRUD.txt 파일을 열도록 구현하였다.

```
53  int main(void) {
54      MYSQL* connection = NULL;
55      MYSQL conn;
56
57      if (mysql_init(&conn) == NULL) {
58          printf("mysql_init() error!");
59          return 1;
60      }
61
62      connection = mysql_real_connect(&conn, host, user, pw, NULL, 3306, (const char*)NULL, 0);
63      if (connection == NULL) {
64          printf("%d ERROR : %s\n", mysql_errno(&conn), mysql_error(&conn));
65          return 1;
66      } else {
67          printf("Connection Succeeded\n\n");
68
69          execute_file_queries(connection, "20211547_CRUD_1.txt");
70
71          if (mysql_select_db(&conn, db)) {
72              printf("%d ERROR : %s\n", mysql_errno(&conn), mysql_error(&conn));
73              return 1;
74          }
75
```

(main 함수는 길이가 길어 3개로 쪼개어 설명하도록 하겠다.) 우선 가장 먼저 mysql과 연결시켜 주었다. 연결이 성공적으로 되었을 때만 앞서 언급한 함수들을 통해 테이블의 생성과 데이터 삽입을 수행할 수 있는 CRUD_1 파일을 열도록 하였다.

이는 실제 쿼리문이 들어오기 전에 테이블과 데이터를 조성하도록 함을 나타낸다.

```
        while (1) {
            printf("\n\n------- SELECT QUERY TYPES -------\n\n");
            printf("1. TYPE 1\n");
            printf("2. TYPE 2\n");
            printf("3. TYPE 3\n");
            printf("4. TYPE 4\n");
            printf("5. TYPE 5\n");
            printf("6. TYPE 6\n");
            printf("7. TYPE 7\n");
            printf("0. QUIT\n");
            printf("\n-----------------------------------\n");

            int command = 0;
            printf("> Please select a query type : ");
            scanf("%d", &command);

            if (command == 0) break;

            char query[1000] = {0};

            switch (command) {
                case 1: {
                    int subtype = 0;
                    printf("\n\n------- Subtypes in TYPE 1 -------\n\n");
                    printf("1. TYPE 1-1\n");
                    printf("0. BACK\n");
                    printf("\n-----------------------------------\n");

                    printf("> Please select a subtype : ");
                    scanf("%d", &subtype);

                    if (subtype == 0) break;
```

이후엔 사용자가 type 을 선택할 수 있도록 선택창이 나오며 그 이후엔 선택한 type 에 따라 다른 쿼리문을 수행할 수 있도록 나타난다. 쿼리문을 올바르게 선택하였는지 한 번더 물어보기 위하여 출력해주었다. 또한 사용자가 0을 입력하지 않으면(break) 계속해서 type 을 선택할 수 있도록 (프로그램이 종료되지 않도록) 구현하였다.

```

        execute_file_queries(connection, "20211547_CRUD_2.txt");

        printf("Bye\n");
        mysql_close(connection);
    }

    return 0;
}

```

마지막으로 사용자의 선택이 끝나면 (0 또는 quit 되었을 때) CRUD_2 를 불러와 테이블과 데이터를 삭제할 수 있도록 구현하였다.

4. Query

Type 1번을 테스트해보고자 한다.

```
Connection Succeeded

Executing initial setup queries from 20211547_CRUD_1.txt...
Initial setup completed.

------- 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

-----------------------------------
> Please select a query type : 1
```

MYSQL 과 성공적으로 연결되었음이 나타났고 CRUD1.txt 를 열어 테이블과 데이터의 생성, 삽입이 완료되었음을 알 수 있다.

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

1. TYPE 1-1
0. BACK

-----------------------------------
> Please select a subtype : 1
```

그 뒤에 TYPE 1번을 선택하였음이 나타났고, 또 다시 TYPE 1-1 을 선택하고자 하였다.

TYPE 1-1 은

District 이 'Mapo' 인 것 중 거래 가격이 ₩ 1,000,000,000 and ₩1,500,000,000 사이인 데이터를 출력하면 된다.

```
INSERT INTO property (address, price, number_of_bedrooms, number_of_bathrooms, status, district, school_district, listing_date, sale_date,
lease_start_date, lease_end_date, seller_ID, house_type_ID) VALUES
('123 Mapo Street', 1400000000, 4, 2, 'For Sale', 'Mapo', 8, '2023-01-01', NULL, NULL, NULL, 1, 3),
('456 Mapo Street', 1200000000, 3, 1, 'For Sale', 'Mapo', 8, '2023-02-01', NULL, NULL, NULL, 1, 2),
('789 Gangnam Street', 1800000000, 5, 3, 'For Sale', 'Gangnam', 9, '2023-03-01', NULL, NULL, NULL, 2, 4),
('101 Jongno Street', 1000000000, 2, 1, 'For Sale', 'Jongno', 7, '2023-04-01', NULL, NULL, NULL, 3, 2),
('202 Jongno Street', 900000000, 3, 2, 'For Sale', 'Jongno', 7, '2023-05-01', NULL, NULL, NULL, 3, 2),
('303 Seocho Street', 2500000000, 6, 4, 'For Sale', 'Seocho', 6, '2023-06-01', NULL, NULL, NULL, 2, 4),
('404 Yongsan Street', 2000000000, 4, 3, 'For Sale', 'Yongsan', 5, '2023-07-01', NULL, NULL, NULL, 1, 3),
('605 Gangnam Street', 2200000000, 4, 3, 'For Sale', 'Gangnam', 9, '2023-08-01', NULL, NULL, NULL, 2, 3),
('707 Mapo Street', 1300000000, 3, 2, 'For Sale', 'Mapo', 8, '2023-09-01', NULL, NULL, NULL, 1, 2),
('808 Seocho Street', 1700000000, 5, 3, 'For Sale', 'Seocho', 6, '2023-10-01', NULL, NULL, NULL, 4, 4);
('909 Mapo Street', 1800000000, 3, 2, 'For Sale', 'Mapo', 8, '2023-09-01', NULL, NULL, NULL, 1, 2),
```

CRUD_1에서 생성한 property 테이블에서 지역이 mapo 인 곳은

('123 Mapo Street', 1400000000, 4, 2, 'For Sale', 'Mapo', 8, '2023-01-01', NULL, NULL, NULL, 1, 3),

('456 Mapo Street', 1200000000, 3, 1, 'For Sale', 'Mapo', 8, '2023-02-01', NULL, NULL, NULL, 1, 2),

('707 Mapo Street', 1300000000, 3, 2, 'For Sale', 'Mapo', 8, '2023-09-01', NULL, NULL, NULL, 1, 2),

('909 Mapo Street', 1800000000, 3, 2, 'For Sale', 'Mapo', 8, '2023-09-01', NULL, NULL, NULL, 1, 2);

이며 그 중 가장 상단 3개가 TYPE 1-1 로 출력되어야 한다.

```
Address
123 Mapo Street
456 Mapo Street
707 Mapo Street
```

출력결과로 잘 출력되는 모습을 볼 수 있다.