**Project 2**

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

**1. BCNF Decomposition**

텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

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 dependency는 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**

**텍스트, 도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명**

**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.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

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

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

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.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

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.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

(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.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

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.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

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.

-텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

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

텍스트, 스크린샷, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명

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.

텍스트, 스크린샷, 패턴, 패브릭이(가) 표시된 사진

자동 생성된 설명

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.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

출력결과로 잘 출력되는 모습을 볼 수 있다.