데이터베이스설계 02분반

- 설계과제 -

20195310 김소현

(1) 응용분야 제목

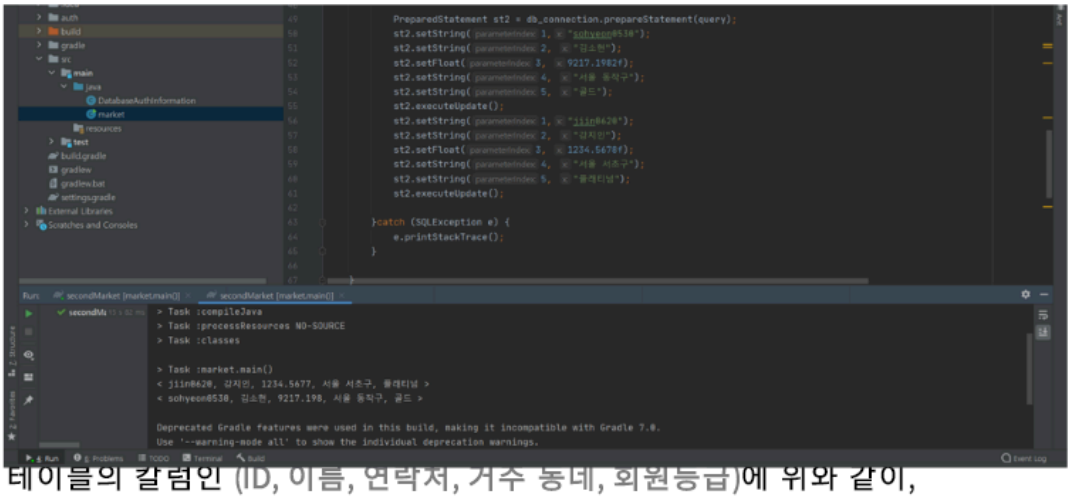
중고거래 장터 시스템

(2) JDBC/MySQL 프로그램

프로그램에서 member(회원) 테이블을 사용하였고, 추가한 tuples은 아래와 같다

```
//쿼리 작성(테이블 생성문)
String query = "CREATE TABLE member(ID VARCHAR(50) NOT NULL, name VARCHAR(50) NOT NULL, phone FLOAT(8,4) NOT NULL, "+
        "town VARCHAR(50) NOT NULL, level VARCHAR(50) NOT NULL, primary key(ID))";
```

(*테이블 작성 쿼리)



테이블의 칼럼인 (ID, 이름, 연락처, 거주 동네, 회원등급)에 위와 같이,

(sohyeon0530, 김소현, 9217.1982, 서울 동작구, 골드)과 (jiin0620, 강지인, 1234.5678, 서울 서초구, 플래티넘)을 tuple 내용으로 삽입하였다. 위 프로그램 실행결과는 Select all 했을 때의 실행결과이다.

이 프로그램에서는 where절을 입력 받아 tuple들을 검색해 출력하도록 하였다.

*소스코드는 아래와 같다.

```
import java.sql.*;
import java.util.*;

public class market {
    public static void main(String[] argv) { //메인함수
        DatabaseAuthInformation db_info = new DatabaseAuthInformation(); //디비정보 호출
        db_info.parse_auth_info("auth/mysql.auth"); //sql 정보 파싱

        String connection_url = String.format("jdbc:mysql://%s:%s/%s?useUnicode=true&" +
                "useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC", db_info.getHost(), db_info.getPort(),
db_info.getDatabase_name());

        Create_Table(db_info.getUsername(), db_info.getPassword(), connection_url);    //테이블 생성 메소드 호출
        Create_Tuple(connection_url, db_info.getUsername(), db_info.getPassword());    //튜플 생성 메소드 호출

        Search(connection_url, db_info.getUsername(), db_info.getPassword()); //조건문에 따라 튜플 검색 메소드 호출
    }



  public static void Search(String url, String userid, String passwd){ //튜플 검색 메소드
        Scanner sc = new Scanner(System.in);
        String rank;
        System.out.print("조건문: "); where = sc.next();
        System.out.println();

        String query = String.format("select ID, level from member where "+where); //입력 받은 조건문으로 select


        try (Connection db_connection = DriverManager.getConnection(url,userid, passwd);
             Statement db_statement = db_connection.createStatement()){

            ResultSet stm = db_statement.executeQuery(query); //등급 select
            while (stm.next()) {
                System.out.println("< " + stm.getString(1) + ", " + stm.getString(2)+" >"); // <정보, 정보, 정보, 정보, 정보> 형식으로 출력
            }

        }catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

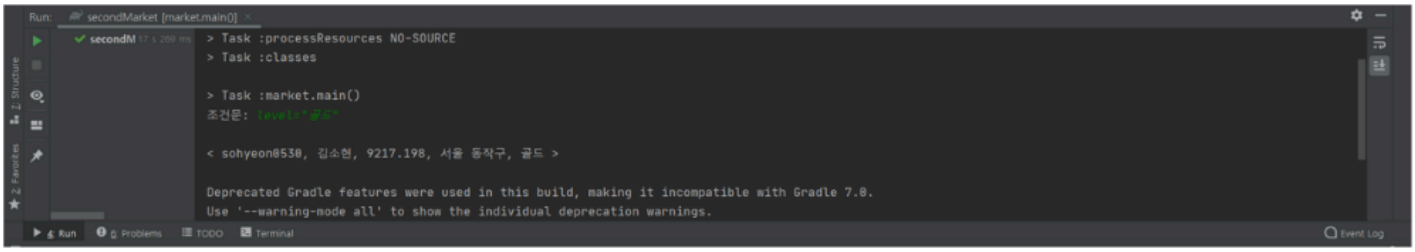
```
public static void Create_Tuple(String url, String userid, String passwd){ //튜플 생성 메소드
    //동적
    try{
        Connection db_connection = DriverManager.getConnection(url, userid, passwd);

        String query = "insert into member (ID, name, phone, town, level) VALUES (?, ?, ?, ?, ?)"; //튜플 5개 삽입 쿼리(회원 테이블을 칼럼이 5개이므로)

        PreparedStatement st2 = db_connection.prepareStatement(query); //쿼리 실행
        st2.setString(1,"sohyeon0530");
        st2.setString(2, "김소현");
        st2.setFloat(3, 9217.1982f);
        st2.setString(4, "서울 동작구");
        st2.setString(5, "골드");
        st2.executeUpdate(); //튜플값 업데이트
        st2.setString(1,"jiin0620");
        st2.setString(2, "강지인");
        st2.setFloat(3, 1234.5678f);
        st2.setString(4, "서울 서초구");
        st2.setString(5, "플래티넘");
        st2.executeUpdate(); //튜플값 업데이트

    }catch (SQLException e) {
        e.printStackTrace();
    }
}

public static void Create_Table(String userid, String passwd, String url){ //테이블 생성 메소드
    //디비 연결

    //쿼리 작성(테이블 생성문)
    String query = "CREATE TABLE member(ID VARCHAR(50) NOT NULL, name VARCHAR(50) NOT NULL, "
+"phone FLOAT(8,4) NOT NULL, town VARCHAR(50) NOT NULL, level VARCHAR(50) NOT NULL, primary key(ID))";

    //정적 sql
    try (Connection db_connection = DriverManager.getConnection(url,userid, passwd);
         Statement db_statement = db_connection.createStatement()){ //상태 객체 만들고
        db_statement.executeUpdate(query); //쿼리문 실행

    }catch (SQLException e) {
        e.printStackTrace();
    }
}

}
```

tuple 검색 조건에 사용한 입력 값은 다음과 같다.

“select ID, level from member where “+where

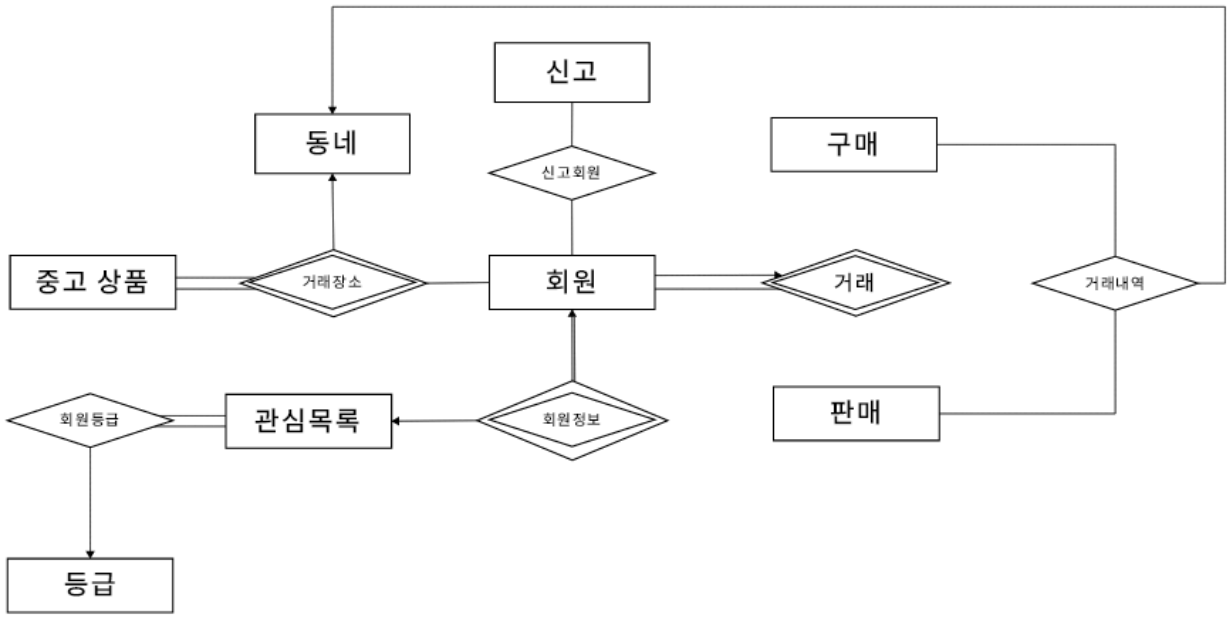
member(회원) 테이블을 from절에서 뽑아와서 입력 받은 조건(where)의 tuple들을 where절에서 뽑아와 모든 칼럼을 select 한다.

프로그램 실행결과는 아래 캡쳐 화면과 같다.



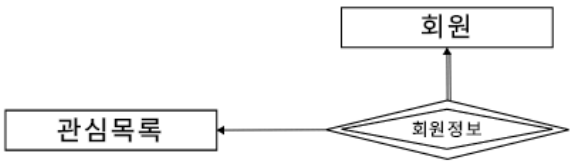
입력 값을 ‘level=“골드”’로 받아서, 회원 등급이 ‘골드’인 회원의 정보 tuple들을 출력하하므로, 위와 같은 결과가 출력된 것을 확인할 수 있다.

(3) ERD



(4) RDB 테이블 스키마

Binary 1:1 관계 집합의 변환



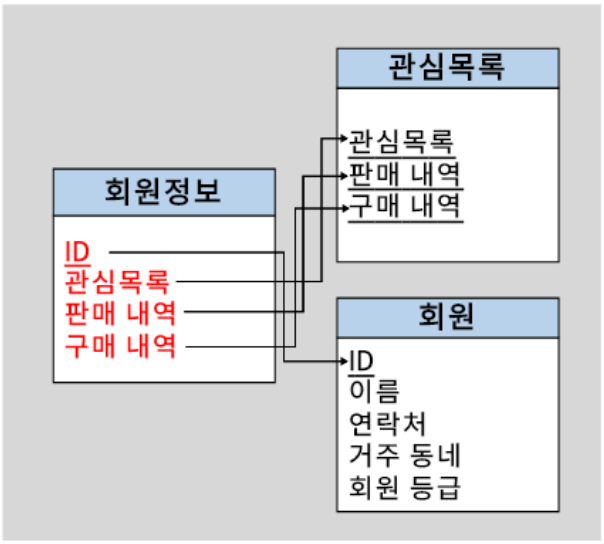
개체 집합 "관심목록", "회원"의 변환:

- 관심목록(관심목록, 판매 내역, 구매 내역)
- 회원(ID, 이름, 연락처, 거주 동네)

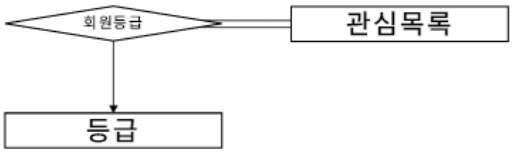
관계 집합 "회원정보"을 독립된 테이블로 변환

- 회원정보(ID, 관심목록, 판매 내역, 구매 내역)

위 변환 과정을 거쳐서 부분적인 스키마의 결과는 오른쪽 그림과 같다.



Binary 1:다 관계 집합의 변환



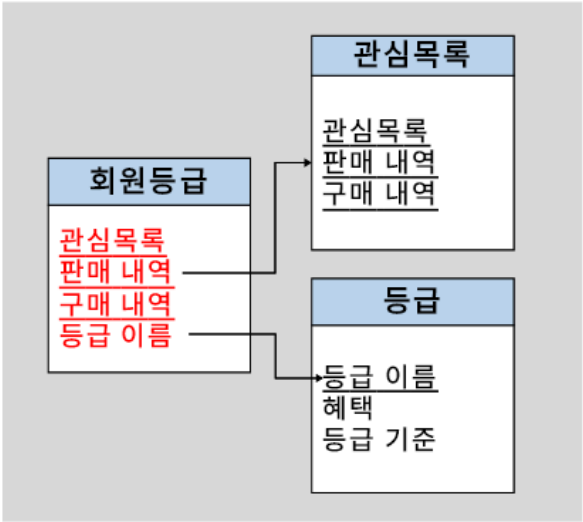
개체 집합 "관심목록", "등급"의 변환:

- 관심목록(관심목록, 판매 내역, 구매 내역)
- 등급(등급 이름, 혜택, 등급 기준)

관계 집합 "회원등급"을 독립된 테이블로 변환

- 회원등급(등급 이름, 관심목록, 판매 내역, 구매 내역)

위 변환 과정을 거쳐서 부분적인 스키마의 결과는 오른쪽 그림과 같다.



Binary 다:다 관계 집합의 변환

개체 집합 "신고", "회원"의 변환:

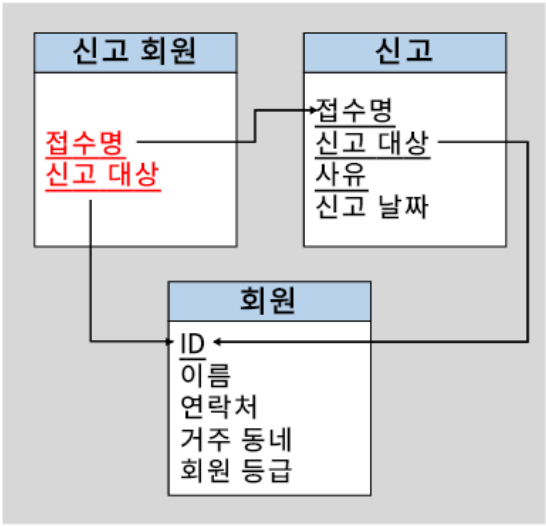
- 신고(접수명, 신고 대상, 사유, 신고 날짜)
- 회원(ID, 이름, 연락처, 거주 동네)

관계 집합 "신고회원"을 독립된 테이블로 변환

- 신고회원(접수명, ID)

신고회원 테이블의 칼럼 이름 변경, 내용은 동일함.)

위 변환 과정을 거쳐서 부분적인 스키마의 결과는 오른쪽 그림과 같다.



Unary 1:다 관계 집합의 변환

회원 간 중고 거래가 이루어지는데, 상품을 판매하는 사람을 판매자, 구매하는 사람을 구매자라고 한다.

관계 집합 "거래"를 독립된 테이블로 변환

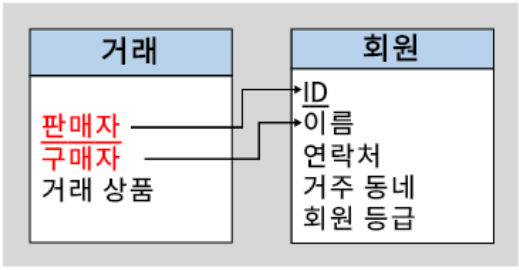
- 회원(ID, 이름, 연락처, 거주 동네)
- 거래(구매자, 판매자, 거래 상품)

Extra 속성을 이용한 변환

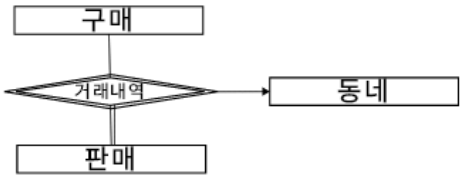
- 스키마1: 회원(ID, 이름, 연락처, 거주 동네, 구매자 ID)
- 스키마2: 회원(ID, 이름, 연락처, 거주 동네, 판매자 ID)

데이터 중복 초래를 막기 위해서 1:다 중 다 쪽에 외래키를 설치하는데, 스키마1을 선택하여, 다 쪽에 해당하는 판매자 쪽에 외래키를 설치한다.

위 변환 과정을 거쳐서 부분적인 스키마의 결과는 오른쪽 그림과 같다.



Ternary 관계 집합의 변환(1)



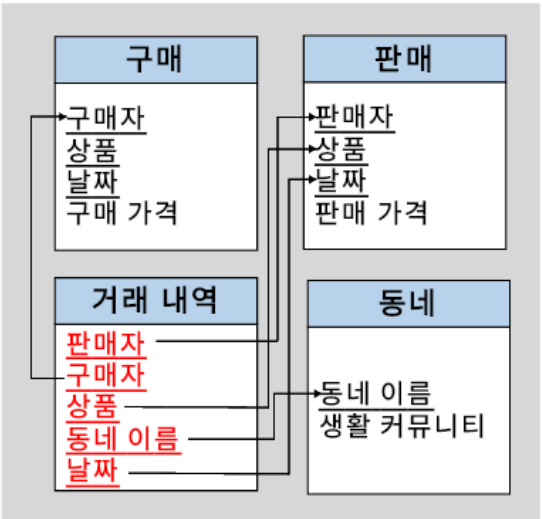
개체 집합의 변환

- 판매(판매자, 상품, 판매 가격, 날짜)
- 구매(구매자, 상품, 구매 가격, 날짜)
- 동네(동네 이름, 생활 커뮤니티)

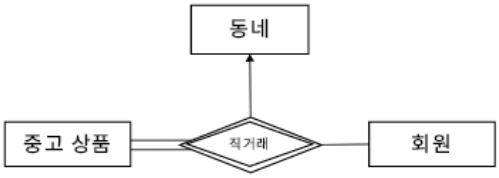
관계 집합 "거래내역"의 변환

- 거래내역(판매자, 구매자, 상품, 동네 이름, 날짜)

위 변환 과정을 거쳐서 부분적인 스키마의 결과는 오른쪽 그림과 같다.



Ternary 관계 집합의 변환(2)



개체 집합의 변환

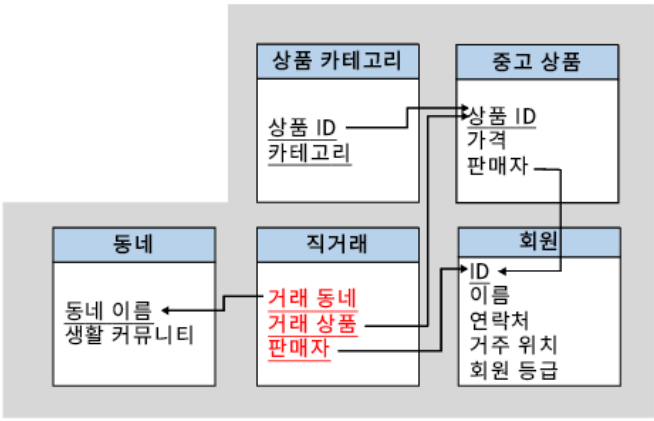
- 회원(ID, 이름, 연락처, 거주 동네)

중고 상품(상품ID, 카테고리, 가격, 판매자)

- 중고 상품(상품ID, 가격, 판매자)
- 상품 카테고리(상품ID, 카테고리)
- 동네(동네 이름, 생활 커뮤니티)

관계 집합 "직거래"의 변환

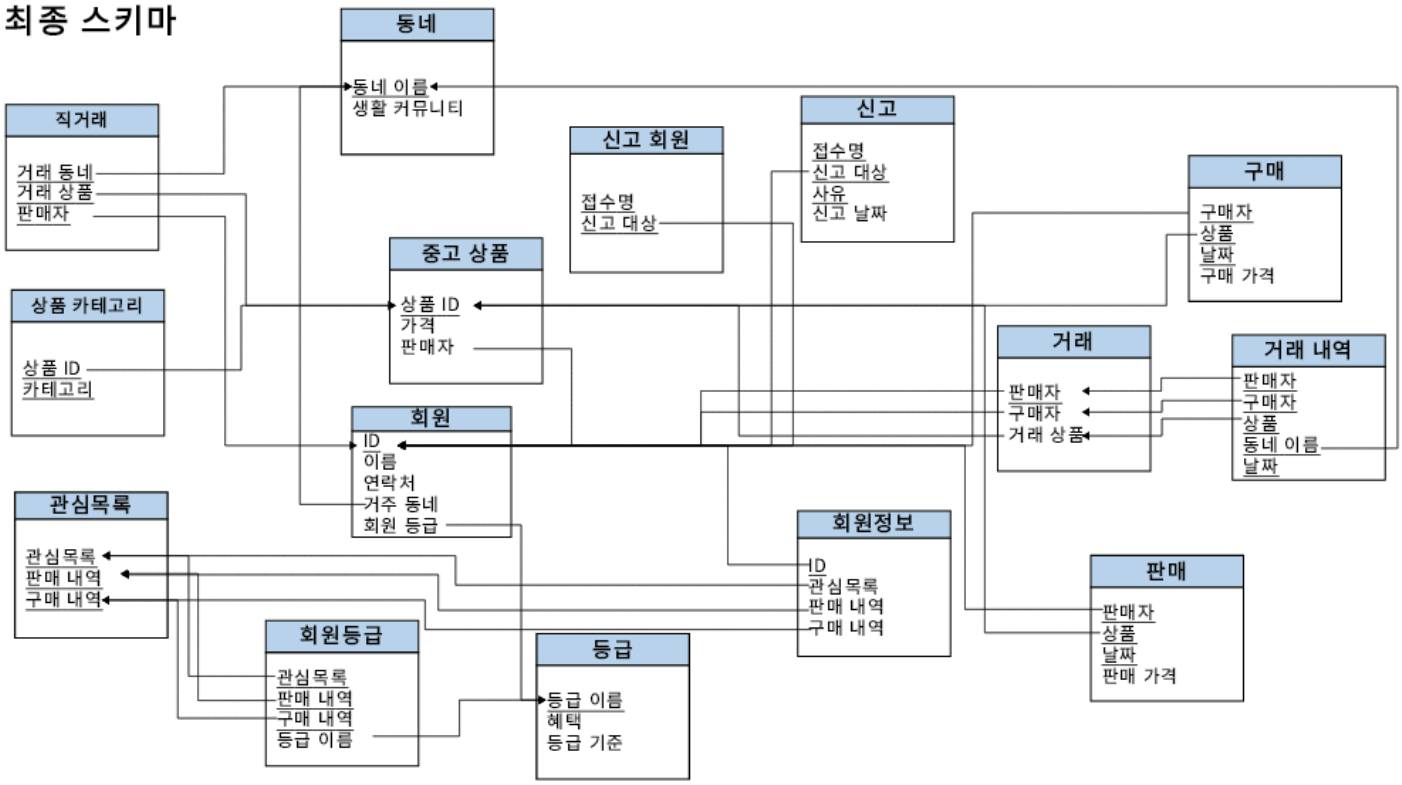
- 직거래(동네 이름, ID, 상품ID)



개체 집합 "중고 상품"에서 '카테고리' 칼럼이 다중 값을 가지므로 개체집합을 두 집합으로 변환하였다.
(거래장소 테이블의 칼럼 이름 변경, 내용은 동일함.)

위 변환 과정을 거쳐서 부분적인 스키마의 결과는 오른쪽 그림과 같다.

최종 스키마



(5) SQL

(1) 2020년 9월에 5개 이상 중고상품을 구매한 구매자의 ID와 이름?

```
    select ID, 이름
     from 구매, 회원
   where ID=구매자 and 2020.09<날짜<2020.10
group by ID, 이름
  having count(상품)>=5
```

from절에서 “구매”와 “회원”을 뽑아 where절에서 동일성 처리를 하고, 날짜가 2020.09와 2020.10 사이인 즉 9월인 조건의 tuple들을 뽑아 ‘ID’와 ‘이름’으로 group by로 묶어 having 절에서 count(상품)>=5 하여 구매 거래한 상품이 5 이상인 tuple들을 골라 내어 ‘ID’와 ‘이름’을 select하여 2020년 9월에 5개 이상 중고상품을 구매한 구매자의 ID와 이름을 구했다.

이는 5개 이상 상품 구매 거래를 한 회원을 대상으로 이달의 감사 회원을 뽑아 시스템의 혜택을 주는 기능을 위함이다.

(2) 거래가 가장 많은 상품의 카테고리?

```
(select 카테고리 from 거래, 상품 카테고리 where 거래 상품=상품 ID)
except
(   select 카테고리
     from 상품 카테고리, 거래 as A, 거래 as B
   where 상품 ID = A.거래 상품
group by 카테고리
  having count(A.거래 상품) < count(B.거래 상품) )
```

from절에서 “거래”와 “상품 카테고리”를 뽑아 동일성 처리 후 뽑아낸 ‘카테고리’ tuple에서,

from절에서 “상품 카테고리”와 A와 B로 rename한 “거래“ 두 테이블을 뽑아 (최대가 아닌) A와 “상품 카테고리“의 칼럼을 동일성 처리하고 ‘카테고리’로 group by한 tuple들 중에서, count(A.거래 상품) < count(B.거래 상품)하여 A의 거래된 상품 수가 최대가 되지 않도록 한 조건의 tuple들을 뽑아내 그 중 select한 ‘카테고리‘ tuple을 except하여 거래 상품 수가 최대가 아닌 tuple들을 뺌으로써 거래 상품 수가 최대인 tuple 값을 구했다.

이는 거래가 가장 많은 상품의 카테고리를 구해서 그 상품의 카테고리를 좀 더 활성화 시키는 기능을 위함이다.

(3) 신고 받은 횟수가 3번 이상인 회원의 ID와 이름?

```
   select ID, 이름
    from 회원, 신고 회원
  where ID=신고 대상
group by ID, 이름
  having count(신고 대상)>=3
```

from절에서 “회원”과 “신고 회원”을 뽑아 where절에서 동일성 처리한 후 ‘ID’와 ‘이름’을 group by하고 having절에서 count(신고 대상)>=3 즉 신고 내역이 3번 이상인 tuple들을 뽑아 그 중 ‘ID’과 ‘이름’을 select하여 신고 받은 횟수가 3번 이상인 회원의 ID와 이름을 구했다.

이는 신고 받은 전적이 여러 번 되는 회원들을 서비스 사용 제한이나 중지 등 활동에 제한을 두는 기능을 위함이다.

(4) 2020년 10월 동안 거래가 50번 이상인 동네의 동네 이름과 생활 커뮤니티?

```
select 동네.동네 이름, 생활 커뮤니티
 from 동네, 거래 내역
where 동네.동네 이름 = 거래 내역.동네 이름 and 2020.10<날짜<2020.11
group by 동네.동네 이름, 생활 커뮤니티
having count(동네 이름)>=50
```

from절에서 “동네”와 “거래 내역”을 뽑아서 where절에서 동일성 처리를 해주고 날짜가 2020년 10월인 (2020.10<날짜<2020.11) tuple들을 뽑아 ‘동네.동네 이름’과 ‘생활 커뮤니티’를 group by한 상태에서 having절에서 count(동네 이름)>=50(거래가 50번 이상 일어남)인 tuple들을 뽑아 그 중 ‘동네.동네 이름’과 ‘생활 커뮤니티‘의 tuple들을 select하여 2020년 10월 동안 거래가 50번 이상인 동네의 동네 이름과 생활 커뮤니티를 구했다.

이는 거래가 많이 일어나는 동네의 커뮤니티를 참조하여 그 동네의 생활 커뮤니티를 활성화 되도록 하고 거래가 많이 일어나는 원인을 조사하는 기능을 위함이다.

(5) 회원 등급이 “다이아”인 회원들의 ID와 이름, 연락처?

```
select ID, 이름, 연락처
 from 회원
where 회원 등급 = “다이아“
```

from절에서 “회원”을 뽑아 where절에서 회원 등급=“다이아”인 tuple들을 뽑아내 그 중의 ‘ID’, ‘이름’, ‘연락처’의 tuple들을 골라내 회원 등급이 “다이아”인 회원들의 ID와 이름, 연락처를 구했다.

이는 회원 등급이 “다이아”인 회원들의 기본 정보들을 뽑아 랜덤으로 상품을 주는 이벤트 기능을 위함이다.

(6) 정규화(BCNF)

판매(판매자, 상품, 날짜, 판매 가격)

FD: ‘판매자, 상품, 날짜‘, ‘판매 가격‘ -> ‘판매자, 상품, 날짜’

// ‘판매자, 상품, 날짜’와 ‘판매 가격’이 ‘판매자, 상품, 날짜’를 함수적으로 결정한다.

(*”판매”는 이루어진 판매 거래 정보로, ‘판매자, 상품, 날짜’와 ‘판매 가격’을 보면 특정 ‘판매자, 상품, 날짜’을 알 수 있으므로 함수적으로 결정한다고 할 수 있다.)

함수종속에서 함수종속성이 trivial하므로 BCNF를 만족한다.

구매(구매자, 상품, 날짜, 구매 가격)

FD: ‘구매자, 상품, 날짜‘, ‘구매 가격‘ -> ‘구매자, 상품, 날짜’

// ‘구매자, 상품, 날짜’와 ‘구매 가격’이 ‘구매자, 상품, 날짜’를 함수적으로 결정한다.

(*”구매”는 이루어진 구매 거래 정보로, ‘구매자, 상품, 날짜’와 ‘구매 가격’을 보면 특정 ‘구매자, 상품, 날짜’을 알 수 있으므로 함수적으로 결정한다고 할 수 있다.)

함수종속에서 함수종속성이 trivial하므로 BCNF를 만족한다.

회원정보(ID, 관심목록, 판매 내역, 구매 내역)

FD: ID -> 관심목록, 판매 내역, 구매 내역

// ID는 관심목록, 판매 내역, 구매 내역을 함수적으로 결정한다.

(*회원의 ID로 회원의 관심목록과 판매 내역, 구매 내역을 알 수 있으므로 함수적으로 결정한다고 할 수 있다.)

함수종속에서 'ID'가 super key이므로 BCNF를 만족한다.

신고(접수명, 신고 대상, 사유, 신고 날짜)

FD: '접수명, 신고 대상, 사유', '신고 날짜' -> '접수명, 신고 대상, 사유'

// '접수명, 신고 대상, 사유'와 '신고 날짜'는 '접수명, 신고 대상, 사유'를 함수적으로 결정한다.

(*"신고"는 접수된 신고 정보로, '접수명, 신고 대상, 사유'와 '신고 날짜'를 보면 특정 '접수명, 신고 대상, 사유'를 알 수 있으므로 함수적으로 종속한다고 할 수 있다.)

함수종속에서 함수종속성이 trivial하므로 BCNF를 만족한다.

회원(ID, 이름, 연락처, 거주 동네, 회원 등급)

(FD1: ID, 이름 -> ID) (FD2: ID, 연락처 -> ID) (FD3: ID, 거주 동네 -> ID) (FD4: ID, 회원 등급 -> ID)

// 함수종속에 사용된 모든 칼럼 회원 스키마에 포함되고, 각 ID를 함수적으로 종속한다.

함수종속에서 4개 모두 함수종속성이 trivial하므로 BCNF를 만족한다.

등급(등급 이름, 혜택, 등급 기준)

FD: 등급 이름 -> 혜택, 등급 기준 // '등급 이름'이 '혜택'과 '등급 기준'을 함수적으로 결정한다.

(*등급에 따라 혜택과 등급 기준이 나뉜다.)

함수종속에서 '등급 이름'이 super key이므로 BCNF를 만족한다.

회원등급(관심목록, 판매 내역, 구매 내역, 등급 이름)

FD: '관심목록, 판매 내역, 구매 내역', '등급 이름' -> '관심목록, 판매 내역, 구매 내역'

// '관심목록, 판매 내역, 구매 내역'과 '등급 이름'이 '관심목록, 판매 내역, 구매 내역'을 함수적으로 결정한다.

(*두 개체집합 사이에서 형성된 관계집합이 변환된 테이블로, '관심목록, 판매 내역, 구매 내역'이 '관심목록, 판매 내역, 구매 내역'과 '등급 이름'에 함수적으로 종속된다.)

함수종속에서 함수종속성이 trivial하므로 BCNF를 만족한다.

동네(동네 이름, 생활 커뮤니티)

FD: 동네 이름 -> 생활 커뮤니티 // '생활 커뮤니티'는 '동네 이름'에 함수적으로 종속된다.

함수종속에서 '동네 이름'이 super key이므로 BCNF를 만족한다.

중고 상품(상품 ID, 가격, 판매자)

FD: 상품 ID -> 가격, 판매자 // '상품 ID'는 '가격'과 '판매자'를 함수적으로 결정한다.

함수종속에서 '상품 ID'가 super key이므로 BCNF를 만족한다.

거래(판매자, 구매자, 거래 상품)

FD: 판매자 -> 구매자, 거래 상품 // '판매자'는 '구매자'와 '거래 상품'을 함수적으로 결정한다.

(*튜플 당 하나의 거래를 나타내는 "거래" 테이블이므로 '구매자'와 '거래 상품'이 '판매자'에 함수적으로 종속됨)

함수 종속에서 '판매자'가 super key이므로 BCNF를 만족한다.

직거래(거래 동네, 거래 상품, 판매자)

: 직거래는 '거래 동네, 거래 상품, 판매자' 자체가 super key로 되므로 그 자체로 고유식별성이 있고, 따라서 BCNF를 만족한다.

상품 카테고리(상품 ID, 카테고리)

:"상품 카테고리"는 "중고상품"의 '카테고리'가 다중 값을 가져서(1NF 위배) 테이블 변환 중 분리되어 만들어진 테이블이다. 그 과정에서 '상품 ID, 카테고리' 자체가 super key가 되고 그 자체로 고유식별성을 가지므로, BCNF를 만족한다.

거래 내역(판매자, 구매자, 상품, 동네 이름, 날짜)

:"거래 내역"은 "구매", "판매", "동네" 3개의 개체집합에서 만들어진 관계집합에서 변환된 테이블로, '판매자, 구매자, 상품, 동네 이름, 날짜' 자체가 super key가 되면서 고유식별성을 가지므로, BCNF를 만족한다.

관심목록(관심목록, 판매 내역, 구매 내역)

:"관심목록"은 판매 내역과 구매 내역에 따른 관심목록을 파악하기 위한 것으로,

'관심목록, 판매 내역, 구매 내역' 자체가 super key가 되고 고유식별성을 가지므로 BCNF를 만족한다.

신고 회원(접수명, 신고 대상)

:"신고 회원"은 두 개체집합 사이에서 다:다로 만들어진 관계집합에서 변환된 테이블로,

'접수명, 신고대상' 자체를 super key가 되고 고유식별성을 가지므로 BCNF를 만족한다.