Bank DBMS 프로그램 보고서

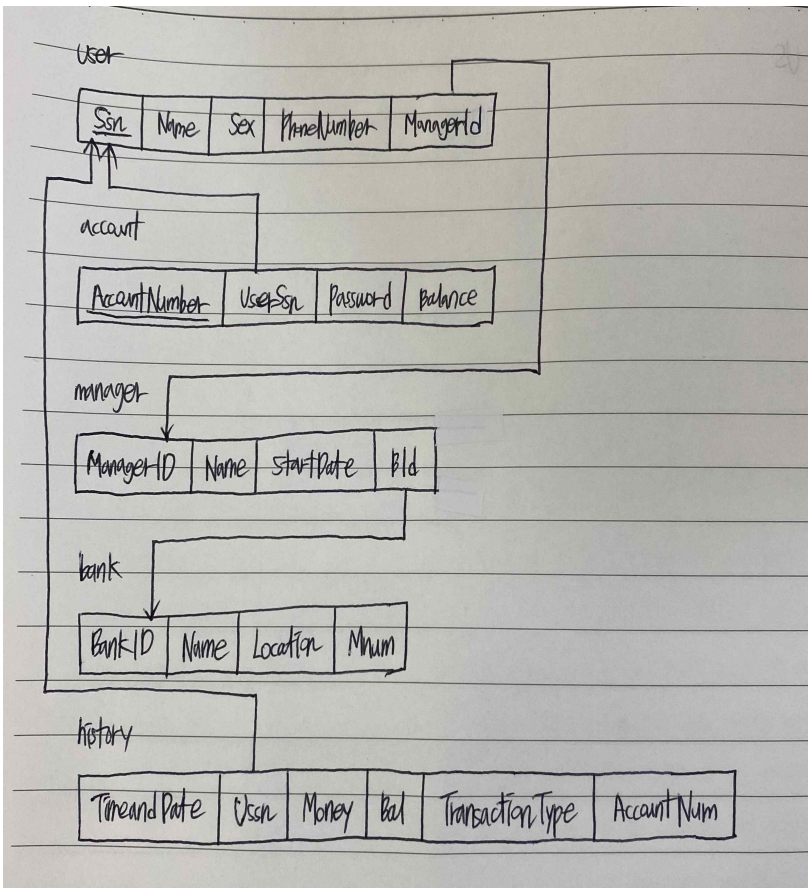
컴퓨터소프트웨어학부
2020078622 강영민

1. 컴파일 및 실행

```
C:₩BankDBMS>javac -classpath mariadb-java-client-2.7.4.jar; BankApp.java -encoding utf8

C:₩BankDBMS>java -cp mariadb-java-client-2.7.4.jar; BankApp
DB 접속 성공
```

2. 변경된 Table Schema와 Relationship



* 직접 프로그래밍해보니 예상했던 schema와 relationship들의 관계를 많이 수정했습니다.

1) user -> account

- deposits : 고객은 계좌에 입금을 합니다.
- withdraws : 고객은 계좌에 출금을 합니다.
- opens : 고객은 새로운 계좌를 개설할 수 있습니다.
- deletes : 고객은 계좌를 삭제합니다.

-> 추가된 foreign key : account.UserSsn

2) manager -> user

- manages : 각 직원은 각 한명의 고객을 담당하여 관리합니다.

-> 추가된 foreign key : user.ManagerId

3) manager -> bank

- belongs to : 각 직원은 한 지점에 속합니다.

-> 추가된 foreign key : manager.bankId

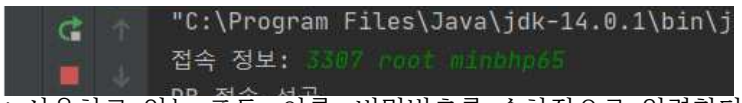
4) manager -> history

- manages : 각 직원은 입출금내역을 관리하는데, 조회 및 검색을 할 수 있습니다.

-> 추가된 foreign key : history.Ussn

5. 프로그램 구동방식

1) 접속 정보 입력하기



: 사용하고 있는 포트, 이름, 비밀번호를 순차적으로 입력한다.

[관련 코드]

```
try {
    System.out.print("접속 정보: ");
    String line = keyboard.nextLine();
    String tmp[] = line.split( regex: " ");
    con = DriverManager.getConnection( url: "jdbc:mariadb://127.0.0.1:" + tmp[0] + "/bank", tmp[1], tmp[2]);

    if (con == null) {
        System.out.println("DB 접속 실패");
        return;
    } else
        System.out.println("DB 접속 성공");

} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("DB 접속 실패");
    return;
}
```

2) 메인 화면 및 모드 선택

```
DB 접속 성공


------------Select Menu-------------
0. 종료
1. 관리자 모드
2. 사용자 모드
3. 사용자 신규 가입
-------------------------------------
위의 메뉴에서 원하시는 모드의 번호를 입력해주세요. :
```

: 관리자 혹은 사용자는 자신이 원하는 모드에 따라 선택할 수 있다.

[관련 코드]

```
public static Integer selectMenu() {
    System.out.println("\n------------Select Menu-------------");
    System.out.println("0. 종료");
    System.out.println("1. 관리자 모드");
    System.out.println("2. 사용자 모드");
    System.out.println("3. 사용자 신규 가입");
    System.out.println("-------------------------------------");
    System.out.print("위의 메뉴에서 원하시는 모드의 번호를 입력해주세요. : ");
    int menu = keyboard.nextInt();
    return menu;
}
```

3) 관리자 및 사용자 모드 접근

3-1. 관리자 접근 권한 확인

```
위의 메뉴에서 원하시는 모드의 번호를 입력해주세요. : 1
관리자 아이디를 입력해주세요. :
```

: 관리자는 자신의 아이디를 입력하여 관리자 모드로 접근할 수 있다.

```
관리자 아이디를 입력해주세요. : 33
관리자 접근 권한이 없습니다.
```

: 입력이 올바르지 않을 시 프로그램은 종료된다.

3-2. 사용자 접근 권한 확인

```
위의 메뉴에서 원하시는 모드의 번호를 입력해주세요. : 2
주민등록번호를 입력해주세요. :
```

[관련 코드]

```
switch (selectMenu()) {
    case 0:
        return;
    case 1:
        System.out.print("관리자 아이디를 입력해주세요. : ");
        int mid = keyboard.nextInt();
        manager(mid);
        break;
    case 2:
        System.out.print("주민등록번호를 입력해주세요. : ");
        int ssn = keyboard.nextInt();
        user(ssn);
        break;
    case 3:
        makeNewUser();
        break;
}
```

4) 사용자 신규 가입

```
------------Select Menu-------------
0. 종료
1. 관리자 모드
2. 사용자 모드
3. 사용자 신규 가입
-----------------------------------
위의 메뉴에서 원하시는 모드의 번호를 입력해주세요. : 3
반갑습니다!
고객님의 성함을 입력해주세요.
강영민
고객님의 성별의 번호 입력해주세요.
1.남 / 2.여
2
고객님의 주민번호(뒤7자리)를 입력해주세요.
3333333
고객님의 전화번호(뒤8자리)를 입력해주세요.
88888888
모든 정보를 입력을 다하셨습니다. 고객님을 관리할 직원의 이름을 선택하시고 입력해주세요.
김이현 / 김영희 /
김이현
축하드립니다! 고객님의 가입이 완료되었습니다. 고객님의 관리 사원은 김이현입니다.

Process finished with exit code 0
```

: 사용자는 이름, 번호, 주민번호, 전화번호, 관리직원을 택할 수 있으며, 가입이 완료되면 user table에 해당하는 값의 tuple이 추가된다. (Insert into user)

[관련 코드]

```
pst = con.prepareStatement( sql: "select ManagerID from manager where Name = ? ");
pst.setString( parameterIndex: 1, Mname);
rs = pst.executeQuery();
rs.next();
pst = con.prepareStatement( sql: "insert into user values ( ?, ?, ?, ?, ? )");
pst.setInt( parameterIndex: 1,ssn);
pst.setString( parameterIndex: 2, name);
String sx = "여";
if(sex == 1)
    sx = "남";
pst.setString( parameterIndex: 3, sx);
pst.setInt( parameterIndex: 4, phonenum);
pst.setInt( parameterIndex: 5, rs.getInt( columnIndex: 1));
pst.executeQuery();
System.out.println("축하드립니다! 고객님의 가입이 완료되었습니다. 고객님의 관리 사원은 " + Mname + "입니다.");
```

5) 관리자 모드

5-1. 업무 선택

```
------------Select Menu-------------
0. 종료
1. 관리자 모드
2. 사용자 모드
3. 사용자 신규 가입
-----------------------------------
위의 메뉴에서 원하시는 모드의 번호를 입력해주세요. : 1
관리자 아이디를 입력해주세요. : 11025
--------------Manager Mode-----------------
안녕하세요! 김이현사원님 | 컴소은행사가정지점 소속
원하시는 업무의 번호를 입력해주세요.
1.입출금 내역 관리 / 2.지점 이동 신청
```

: 올바른 직원 아이디를 입력하면 관리자 모드로 전환되며 두가지 업무 중에 하나를 선택할 수 있다.

5-2. 입출금 내역관리

```
--------------Manager Mode-----------------
안녕하세요! 김이현사원님 | 컴소은행사가정지점 소속
원하시는 업무의 번호를 입력해주세요.
1.입출금 내역 관리 / 2.지점 이동 신청
1
원하시는 업무 번호를 입력해주세요.
1.전체 조회 / 2.검색
1
입출금 내역이 존재하지 않습니다.
또 다른 업무를 진행하시겠습니까?
1.네 / 2.아니오
```

: 입출금 내역을 조회 및 검색할 수 있는데, 내역이 없을 때의 화면이다.

```
--------------Manager Mode-----------------
안녕하세요! 김이현사원님 | 컴소은행사가정지점 소속
원하시는 업무의 번호를 입력해주세요.
1.입출금 내역 관리 / 2.지점 이동 신청
1
원하시는 업무 번호를 입력해주세요.
1.전체 조회 / 2.검색
1
--------------------------------------------------------------------------------
|          날짜 및 시간          | 사용자번호 |  계좌번호  | 거래액수 |   잔액   | 종류 |
--------------------------------------------------------------------------------
| 2021년 12월 03일 22시 15분 17초 | 4444444 | 6620919 | 500000원 | 500000원 | 입금 |
--------------------------------------------------------------------------------
| 2021년 12월 03일 22시 15분 38초 | 4444444 | 6620919 | 40000원 | 460000원 | 출금 |
--------------------------------------------------------------------------------
또 다른 업무를 진행하시겠습니까?
1.네 / 2.아니오
```

: 전체조회에서는 history table의 모든 값을 출력한다.

```
--------------Manager Mode-----------------
안녕하세요! 김이현사원님 | 컴소은행사가정지점 소속
원하시는 업무의 번호를 입력해주세요.
1.입출금 내역 관리 / 2.지점 이동 신청
1
원하시는 업무 번호를 입력해주세요.
1.전체 조회 / 2.검색
2
검색하고자 하는 고객의 주민번호를 입력해주세요.
4444444
--------------------------------------------------------------------------------
|          날짜 및 시간          | 사용자번호 |  계좌번호  | 거래액수 |   잔액   | 종류 |
--------------------------------------------------------------------------------
| 2021년 12월 03일 22시 15분 17초 | 4444444 | 6620919 | 500000원 | 500000원 | 입금 |
--------------------------------------------------------------------------------
| 2021년 12월 03일 22시 15분 38초 | 4444444 | 6620919 | 40000원 | 460000원 | 출금 |
--------------------------------------------------------------------------------
또 다른 업무를 진행하시겠습니까?
1.네 / 2.아니오
```

: 검색하고자 할 때는 알고 싶은 내역의 고객의 주민번호(Ssn)을 입력하여 출력한다. (Select)

[관련 코드]

```
case 2:
    int num;
    int j = 1;
    System.out.println("검색하고자 하는 고객의 주민번호를 입력해주세요.");
    while(j == 1) {
        num = keyboard.nextInt();
        pst = con.prepareStatement( sql: "select * from history where Ussn = ? ");
        pst.setInt( parameterIndex: 1, num);
        rs = pst.executeQuery();
        rs.next();
        if (num != rs.getInt( columnIndex: 2)) {
            System.out.println("입력하신 " + num + "번호는 입출금거래내역에서 존재하지 않습니다. 다시 입력해주세요.");
        } else {
            printHistory(rs);
            j = 0;
        }
    }
    break;
```

5-3. 지점 이동 신청

```
원하시는 업무의 번호를 입력해주세요.
1.입출금 내역 관리 / 2.지점 이동 신청
2
이동을 원하시는 지점의 번호를 입력해주세요. * 사원수가 10명 초과인 곳은 이동 신청이 제한됩니다.
1. 컴소은행사가정지점 | 위치: 중랑구면목동 | 사원수: 1
2. 컴소은행답십리지점 | 위치: 동대문구답십리동 | 사원수: 1
3. 컴소은행한양대지점 | 위치: 성동구행당동 | 사원수: 0
3
이동 신청이 수락되었습니다. 내일부터 컴소은행한양대지점으로 출근하세요.
또 다른 업무를 진행하시겠습니까?
1.네 / 2.아니오
```

: 관리자는 자신의 상황에 맞게 지점 이동 신청을 할 수 있다. 이때 manager의 bId, Bank의 Mnum은 모두 변경된다. (update)

[관련 코드]

```
if(rs.getInt( columnIndex: 4) < 10){
    System.out.println("이동 신청이 수락되었습니다. 내일부터 " + rs.getString( columnIndex: 2) + "으로 출근하세요.");
    int newNum = rs.getInt( columnIndex: 4) + 1;
    int newBank = rs.getInt( columnIndex: 1);
    pst = con.prepareStatement( sql: "update bank set Mnum = ? where BankID = ? ");
    pst.setInt( parameterIndex: 1, newNum);
    pst.setInt( parameterIndex: 2, newBank);
    pst.executeQuery();

    pst = con.prepareStatement( sql: "update bank set Mnum = ? where BankID = ? ");
    pst.setInt( parameterIndex: 1,  x: nowNum - 1);
    pst.setInt( parameterIndex: 2, nowBank);
    pst.executeQuery();//오래된 은행 사원수 선택

    pst = con.prepareStatement( sql: "update manager set BId = ? where ManagerID = ? ");
    pst.setInt( parameterIndex: 1, newBank);
    pst.setInt( parameterIndex: 2, managerID);
    pst.executeQuery();
} else {
    System.out.println(rs.getString( columnIndex: 2) + "은 현재 사원수 초과로 이동이 불가합니다.");
}
```

6) 사용자 모드

6-1. 계좌 개설

```
--------------User Mode-----------------
안녕하세요! 강영민고객님,
원하시는 업무의 번호를 입력해주세요.
1.입출금 거래 / 2.계좌 개설 및 삭제
1
거래하실 계좌번호를 입력해주세요.
2
고객님의 명의로 된 계좌번호가 존재하지 않습니다. 거래하실 계좌번호를 다시 입력해주세요.
계좌가 없으시다면 0을 입력하시고 계좌 개설을 진행하세요.
0
원하시는 업무의 번호를 입력해주세요.
1.입출금 거래 / 2.계좌 개설 및 삭제
2
원하시는 업무의 번호를 입력해주세요.
1.계좌 신규 개설 / 2.계좌 삭제
1
만드실 계좌의 비밀번호를 설정해주세요.
6050
계좌 개설이 완료되었습니다. 고객님의 신규계좌번호는 6620919입니다.
```

: 사용자는 계좌를 새로, 여러 개 개설할 수 있다. 계좌번호는 난수로 생성한다. account에 추가된다. (Insert into account)

[관련 코드]

```
public static Integer makeNewAccount(){
    int newAccount;
    while(true) {
        try {
            newAccount = (int) (Math.random() * 100000000);
            pst = con.prepareStatement( sql: "select * from account where AccountNumber = ?");
            pst.setInt( parameterIndex: 1, newAccount);
            rs = pst.executeQuery();
            if (rs.next())
                continue;
            else
                break;
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }
    return newAccount;
}
```

```
case 1:
    System.out.println("만드실 계좌의 비밀번호를 설정해주세요.");
    int PW = keyboard.nextInt();
    int newAccount = makeNewAccount();
    pst = con.prepareStatement( sql: "insert into account values (?, ?, ?, default)");
    pst.setInt( parameterIndex: 1, newAccount);
    pst.setInt( parameterIndex: 2, ssn);
    pst.setInt( parameterIndex: 3, PW);
    pst.executeQuery();
    System.out.println("계좌 개설이 완료되었습니다. 고객님의 신규계좌번호는 " + newAccount + "입니다.");
    break;
```

6-2 계좌 삭제

```
--------------User Mode-----------------
안녕하세요! 강영민고객님,
원하시는 업무의 번호를 입력해주세요.
1.입출금 거래 / 2.계좌 개설 및 삭제
2
원하시는 업무의 번호를 입력해주세요.
1.계좌 신규 개설 / 2.계좌 삭제
2
삭제하실 계좌의 번호를 입력해주세요.
6620919
삭제하실 계좌의 비밀번호를 입력해주세요. | 계좌번호 : 6620919
6050
계좌 삭제가 완료되었습니다.
또 다른 업무를 진행하시겠습니까?
1.네 / 2.아니오
```

: 사용자가 삭제하고자하는 계좌의 비밀번호를 올바르게 입력하면 계좌가 account table에서 삭제된다. (Delete)

[관련 코드]

```
    pst = con.prepareStatement( sql: "select * from account where UserSsn = ? and AccountNumber = ?");
    pst.setInt( parameterIndex: 1, ssn);
    pst.setInt( parameterIndex: 2, accountNumber);
    rs = pst.executeQuery();
    if (rs.next()) {
        break;
    } else {
        System.out.println("고객님의 명의로 된 계좌번호가 존재하지 않습니다. 삭제하실 계좌번호를 다시 입력해주세요.");
    }
}
System.out.println("삭제하실 계좌의 비밀번호를 입력해주세요. | 계좌번호 : " + accountNumber);
while (true) {
    password = keyboard.nextInt();
    pst = con.prepareStatement( sql: "select * from account where UserSsn = ? and AccountNumber = ? and Password = ?");
    pst.setInt( parameterIndex: 1, ssn);
    pst.setInt( parameterIndex: 2, accountNumber);
    pst.setInt( parameterIndex: 3, password);
    rs = pst.executeQuery();
    if (rs.next()) {
        break;
    } else {
        System.out.println("비밀번호가 틀렸습니다. 삭제하실 계좌의 비밀번호를 다시 입력해주세요.");
    }
}
pst = con.prepareStatement( sql: "delete from account where AccountNumber = ?");
pst.setInt( parameterIndex: 1, accountNumber);
pst.executeQuery();
System.out.println("계좌 삭제가 완료되었습니다.");
break;
```

6-3 입출금

```
원하시는 업무의 번호를 입력해주세요.
1.입출금 거래 / 2.계좌 개설 및 삭제
1
거래하실 계좌번호를 입력해주세요.
6620919
원하시는 거래의 번호를 입력해주세요.
1.입금 / 2.출금
1
입금하실 돈을 입력해주세요.
500000
입금이 완료되었습니다. 잔액은 500000원 입니다.
또 다른 업무를 진행하시겠습니까?
1.네 / 2.아니오
1
원하시는 업무의 번호를 입력해주세요.
1.입출금 거래 / 2.계좌 개설 및 삭제
1
거래하실 계좌번호를 입력해주세요.
6620919
원하시는 거래의 번호를 입력해주세요.
1.입금 / 2.출금
2
출금하실 계좌의 비밀번호를 입력해주세요. | 계좌번호 : 6620919
6050
출금하실 돈을 입력해주세요. | 잔액 : 500000
40000
출금이 완료되었습니다. 잔액은 460000원 입니다.
```

: 사용자는 계좌번호, 비밀번호를 올바르게 입력할 시에만 입출금을 실행할 수 있다. 사용자가 입출금을 실행할 때마다 makeHistory()함수에서 History table에 입출금 내역이 저장된다. (Insert into history)

[관련 코드]

```
case 1:
    transType = "입금";
    System.out.println("입금하실 돈을 입력해주세요.");
    int money = keyboard.nextInt();
    transmoney = money;
    money += rs.getInt( columnIndex: 4);
    pst = con.prepareStatement( sql: "update account set balance = ? where UserSsn = ? and AccountNumber = ?
    pst.setInt( parameterIndex: 1, money);
    pst.setInt( parameterIndex: 2, ssn);
    pst.setInt( parameterIndex: 3, accountNumber);
    pst.executeQuery();
    System.out.println("입금이 완료되었습니다. 잔액은 " + money + "원 입니다.");
    makeHistory(transType, transmoney, money, accountNumber, ssn);
    break;
case 2:
    transType = "출금";
    System.out.println("출금하실 계좌의 비밀번호를 입력해주세요. | 계좌번호 : " + accountNumber);
    while (true) {
        password = keyboard.nextInt();
        pst = con.prepareStatement( sql: "select * from account where UserSsn = ? and AccountNumber = ? and
        pst.setInt( parameterIndex: 1, ssn);
        pst.setInt( parameterIndex: 2, accountNumber);
        pst.setInt( parameterIndex: 3, password);
        rs = pst.executeQuery();
        if (rs.next())
            break;
        else
            System.out.println("비밀번호가 틀렸습니다. 출금하실 계좌의 비밀번호를 다시 입력해주세요.");
    }
```