프로젝트 4: Bank Simulator 프로그램 보고서

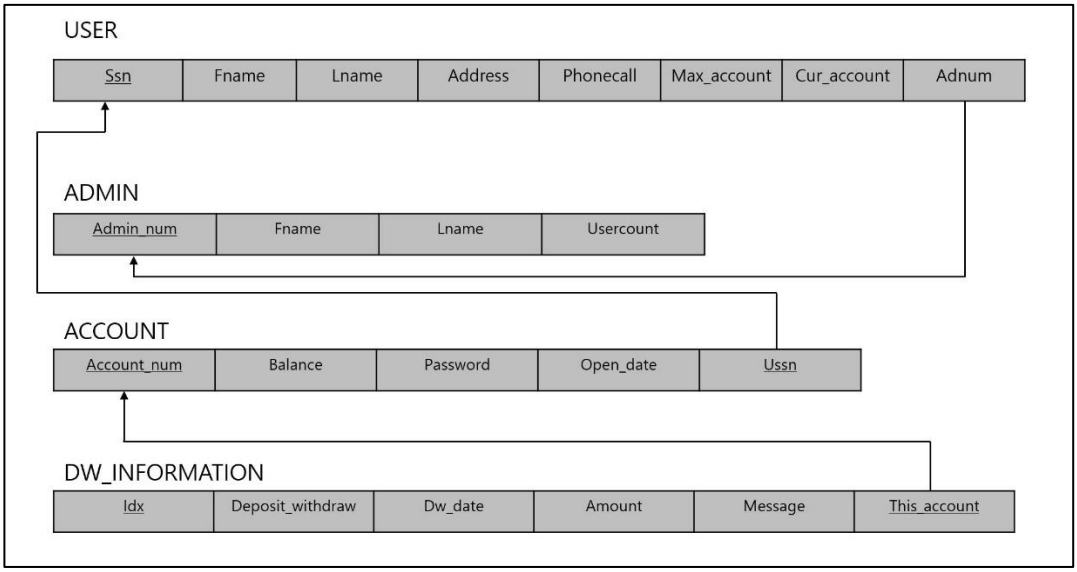
공과대학 컴퓨터소프트웨어학부 2020079689 신다혜

1. 수정 사항

1-1. 수정 사항

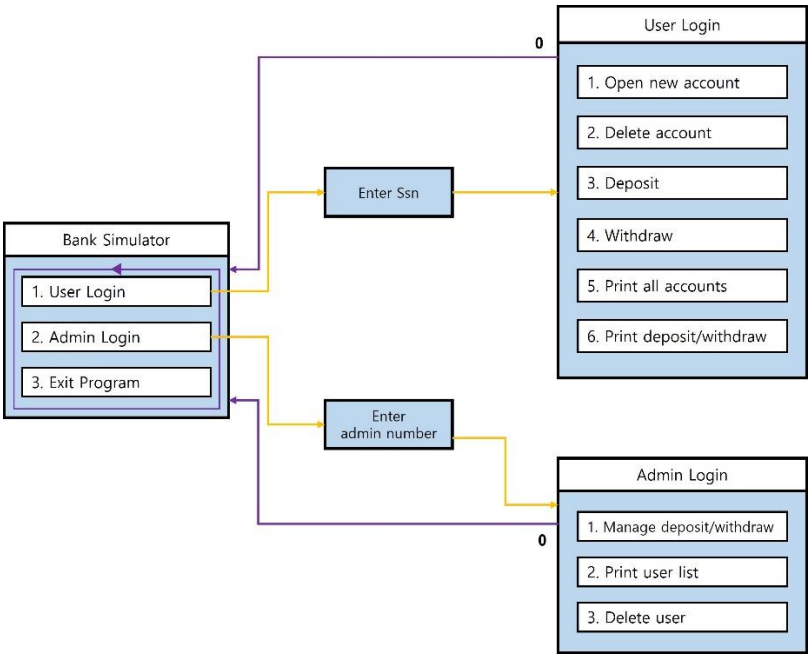
- 모든 table 및 attribute의 이름을 영어로 변경
- USER, ADMIN table의 '이름' attribute를 Fname, Lname으로 분리
- 사용자의 입출금 메시지와 날짜를 DW_INFORMATION table에 저장하도록 분리
- USER가 FK로 ADMIN의 Admin_num을 가진다.
  - P3에서 ADMIN이 FK로 USER의 Ssn을 가지는 오류 수정
- ACCOUNT를 USER 없이 존재할 수 없는 weak entity로 변경
- DW_INFORMATION은 ACCOUNT 없이 존재할 수 없는 weak entity로 설정

1-2. 수정된 관계형 스키마

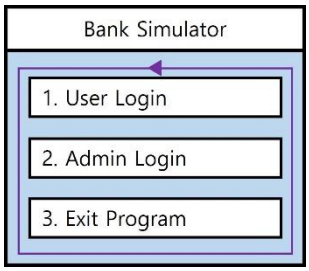


2. 프로그램 동작 방식

2-1. 기본 동작



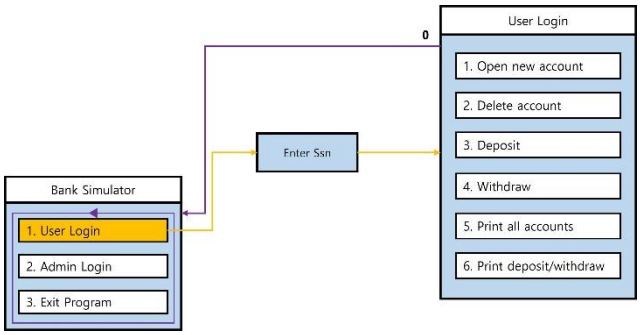
2-2. Bank Simulator



무한으로 반복하며 정수를 입력 받고, 입력 받은 값들에 따라 다음 동작들을 수행

1) User login
2) Admin login
3) Exit program

2-3. User Login



Bank simulator에서 1을 입력하면 User login을 수행

- user의 ssn을 입력 받아 로그인
- 계좌 개설 및 확인, 입출금 기능 수행
- 0을 입력 받으면 시작 메뉴로 돌아감

2-3-1. Open new account

1) 로그인 할 때 입력한 ssn으로 기존 user인지 확인 후 신규 user이면 기본정보 입력
   - 관리자는 ADMIN table에 저장된 관리자 중 랜덤으로 배정
   - '현재 계좌 수 = 최대 계좌 수'인 경우 시작메뉴로 돌아감

2) 입금 금액, 비밀번호 입력 받음
   - 계좌번호는 랜덤으로 생성

3) USER, ACCOUNT, DW_INFORMATION, ADMIN table 수정

2-3-2. Delete account

1) 삭제할 계좌번호 입력
   - 존재하지 않는 계좌인 경우 시작메뉴로 돌아감

2) 해당 계좌 삭제

3) user의 Cur_account 감소

2-3-3. Deposit

1) 입금할 계좌번호 입력
   - 존재하지 않는 계좌인 경우 시작메뉴로 돌아감

2) 입금 금액, 입금 메시지 입력

3) 해당 계좌정보, DW_INFORMATION 수정

2-3-4. Withdraw

1) 출금할 계좌번호 입력
   - 존재하지 않는 계좌인 경우 시작메뉴로 돌아

2) 비밀번호 입력
   - 비밀번호 오류 시 시작메뉴로 돌아감

3) 출금 금액 입력
   - 현재 잔액보다 큰 수 입력 시 시작메뉴로 돌아감

4) 출금 메시지 입력

5) 해당 계좌정보, DW_INFORMATION 수정

2-3-5. Print all accounts

1) 해당 user가 가진 계좌의 계좌번호와 잔액 출력

2-3-6. Print deposit and withdraw history

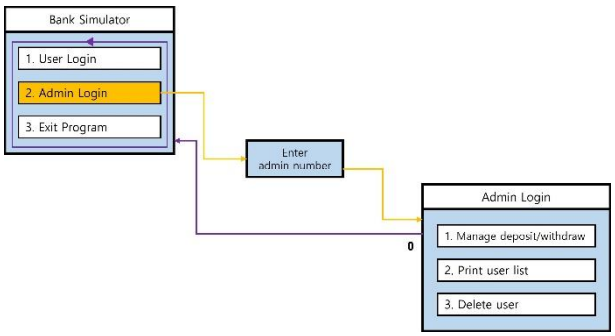
1) 해당 user의 보유 계좌의 계좌번호 출력

2) 입출금 내역을 확인할 계좌번호 입력

- 존재하지 않는 계좌인 경우 시작메뉴로 돌아감

3) 입출금 내역 출력

2-4. Admin Login



Bank simulator에서 2를 입력하면 Admin login을 수행

- admin의 Admin_num을 입력 받아 로그인
- 관리하는 user 및 계좌 확인, user 관리 기능 수행
- 0을 입력 받으면 시작 메뉴로 돌아감

2-4-0. 존재하지 않는 Admin_num인 경우

1) 해당 번호로 새로운 admin 생성여부 확인

- 생성하지 않는 경우 시작메뉴로 돌아감

2) Fname, Lname을 입력 받아 ACCOUNT table의 새로운 tuple 생성

2-4-1. Manage deposit and withdraw

1) 관리하는 사용자의 정보 출력

2) 확인할 사용자의 ssn 입력

- 존재하지 않는 사용자인 경우 시작메뉴로 돌아감

3) '2-3-6' 단계와 같은 동작 수행

4) 삭제할 index 입력 후 삭제

2-4-2. Print user list

1) 관리하는 사용자의 정보 출력

2-4-3. Delete user

1) 관리하는 사용자의 정보 출력

2) 삭제할 사용자의 ssn 입력

- 존재하지 않는 사용자인 경우 시작메뉴로 돌아감

3) 선택한 user 삭제

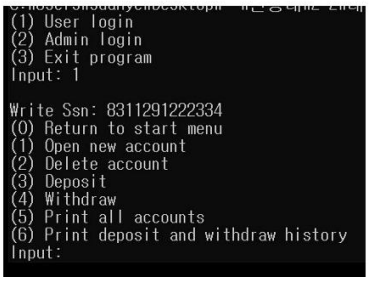
4) admin의 Usercount 감소

3. 프로그램 수행 예시

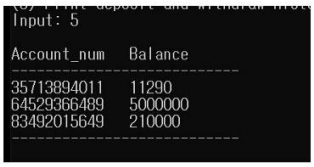
3-1. User Login

아래 예시들은 모두 'Ssn = 8311291222334' user로 실행하였으며,
실행한 순서대로 첨부함

시작메뉴에서 User login까지 한 상태



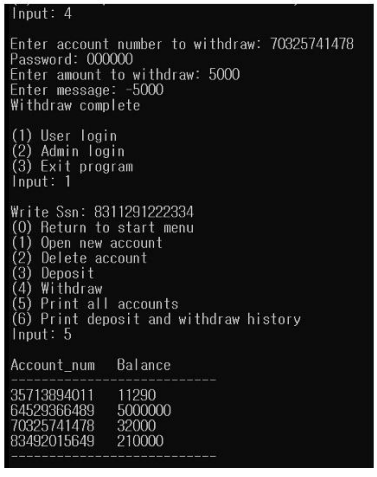
현재 계좌 목록



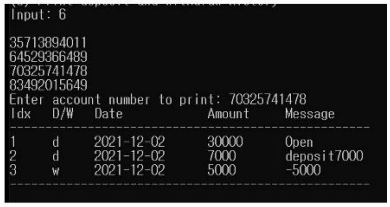
새로운 계좌 생성 후 계좌 목록

만들어진 계좌에 입금 후 잔액 확인

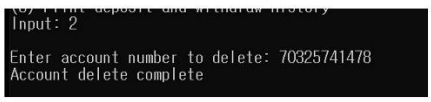
출금 후 잔액 확인



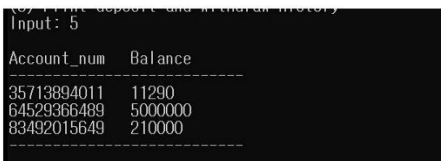
입출금 내역 확인



계좌 삭제



계좌 목록 확인



3-2. Admin Login

아래 예시들은 모두 'Admin_num = 111111111' admin으로 실행하였으며, 실행한 순서대로 첨부함

시작메뉴에서 Admin login까지 한 상태

```
(1) User login
(2) Admin login
(3) Exit program
Input: 2

Write Admin number: 111111111
(0) Return to start menu
(1) Print deposit and withdraw
(2) Print user list
(3) Delete user
Input:
```

현재 관리하는 사용자 목록

```
Input: 2

Ssn             Fname Lname      Address    Phonecall     Max_account   Cur_account
-------------------------------------------------------------------------------------
0109084444444   Shin  Dahye      Seoul      01023233232   4             1
8311291222334   Kim   Haejin     Seoul      01011112222   7             3
8902071444556   Park  Haewoong   Incheon    01033334444   4             2
-------------------------------------------------------------------------------------
```

사용자의 입출금내역 관리

```
Input: 1

Ssn             Fname Lname      Address    Phonecall     Max_account   Cur_account
-------------------------------------------------------------------------------------
0109084444444   Shin  Dahye      Seoul      01023233232   4             1
8311291222334   Kim   Haejin     Seoul      01011112222   7             3
8902071444556   Park  Haewoong   Incheon    01033334444   4             2
-------------------------------------------------------------------------------------
Enter user's ssn to manage: 0109084444444

65242932655
Enter account number to print: 65242932655
Idx   D/W   Date           Amount     Message
-----------------------------------------------
1     d     2021-11-02     5000000    Open
2     d     2021-11-23     700        +700
3     d     2021-12-01     300        300deposit
4     w     2021-12-02     200000     -200000
-----------------------------------------------
Enter index to delete: 3
Delete complete
```

삭제 후 0109084444444 user 입출금내역

```
Input: 6

65242932655
Enter account number to print: 65242932655
Idx   D/W   Date           Amount     Message
-----------------------------------------------
1     d     2021-11-02     5000000    Open
2     d     2021-11-23     700        +700
4     w     2021-12-02     200000     -200000
-----------------------------------------------
```

사용자 삭제 후 사용자 목록 출력

```
Input: 3

Ssn             Fname Lname      Address    Phonecall     Max_account   Cur_account
-------------------------------------------------------------------------------------
0109084444444   Shin  Dahye      Seoul      01023233232   4             1
8311291222334   Kim   Haejin     Seoul      01011112222   7             3
8902071444556   Park  Haewoong   Incheon    01033334444   4             2
-------------------------------------------------------------------------------------

Enter user's ssn to delete: 0109084444444
User delete complete

(1) User login
(2) Admin login
(3) Exit program
Input: 2

Write Admin number: 111111111
(0) Return to start menu
(1) Manage deposit and withdraw
(2) Print user list
(3) Delete user
Input: 2

Ssn             Fname Lname      Address    Phonecall     Max_account   Cur_account
-------------------------------------------------------------------------------------
8311291222334   Kim   Haejin     Seoul      01011112222   7             3
8902071444556   Park  Haewoong   Incheon    01033334444   4             2
-------------------------------------------------------------------------------------
```

3-3. 그 외 동작

- ✓ 새로운 user 정보 입력

```
Write Ssn: 8808141357135
(0) Return to start menu
(1) Open new account
(2) Delete account
(3) Deposit
(4) Withdraw
(5) Print all accounts
(6) Print deposit and withdraw history
Input: 1

Enter first name: Oh
Enter last name: Jinhyuk
Enter address: Seoul
Enter phone number: 01055554321
Enter amount to deposit: 1230000
Enter password: 111111
Account open complete
```

- ✓ '현재 가진 계좌 수 = 최대 계좌 수'여서 새로운 계좌 생성 불가

```
Write Ssn: 0104244321321
(0) Return to start menu
(1) Open new account
(2) Delete account
(3) Deposit
(4) Withdraw
(5) Print all accounts
(6) Print deposit and withdraw history
Input: 1

You already have maximum number of accounts
```

- ✓ 선택지에 없는 값을 입력한 경우

```
(1) User login
(2) Admin login
(3) Exit program
Input: 5

Unexpected input
```

- ✓ 존재하지 않는 계좌, 주민등록번호 입력 시

```
Enter account number to withdraw: 12332123321
Non-existent account number!
```

```
Enter user's ssn to delete: 8003281234567
Non-existent user!
```

- ✓ 출금 비밀번호 오류

```
Input: 4

Enter account number to withdraw: 35713894011
Password: 941573
Wrong password!
```

✓ 출금 가능 금액 초과

```
Enter account number to withdraw: 35713894011
Password: 654321
Enter amount to withdraw: 10000000
Exceeded amount!
```

✓ 존재하지 않는 관리자 번호로 로그인 한 경우

1) 새로운 관리자 생성

```
Write Admin number: 987654321
Non-existent admin!
Do you want to create new admin? (y/n): y

Enter Fname: Yoon
Enter Lname: Sewon
Create new admin complete
```

2) 시작 화면으로

```
Write Admin number: 987654321
Non-existent admin!
Do you want to create new admin? (y/n): n

(1) User login
(2) Admin login
(3) Exit program
Input:
```

3. SQL문 명세 - 작은 따옴표 안의 값은 입력 받은 값

- Ssn이 'ssn'인 user의 계좌번호 SELECT

```
20    sql = """SELECT Account_num
21             FROM account, user
22             WHERE Ssn=%s AND Ussn=Ssn"""
23    cursor.execute(sql, ssn)
```

- 계좌번호가 'selectAccount'인 계좌의 입출금 내역 SELECT

```
36    sql = """SELECT Idx, Deposit_withdraw, Dw_date, Amount, Message
37             FROM dw_information
38             WHERE This_account=%s"""
39    cursor.execute(sql, selectAccount)
```

- Admin_num이 'adminNum'인 관리자가 관리하는 사용자 정보 SELECT

```
49    sql = "SELECT * FROM user WHERE Adnum=%s"
50    cursor.execute(sql, adminNum)
```

- Account_num이 'account'인 계좌의 계좌번호 SELECT (존재여부 확인용)

```
62    sql = "SELECT Account_num FROM account WHERE Account_num = %s AND Ussn = %s"
63    cursor.execute(sql, (account, ssn))
```

- Admin_num이 'adminNum'인 관리자가 관리하는 사람 중 Ssn이 'ssn'인 user의 Ssn SELECT (존재여부 확인용)

```
71    sql = "SELECT Ssn FROM user WHERE Ssn = %s AND Adnum = %s"
72    cursor.execute(sql, (ssn, adminNum))
```

- Ssn이 'ssn'인 user 정보 SELECT

```
99            sql = """SELECT * FROM user WHERE Ssn=%s"""
100           cursor.execute(sql, ssn)
```

- ADMIN table에 있는 모든 tuple의 관리자 SELECT

```
113               sql = "SELECT Admin_num FROM admin"
114               cursor.execute(sql)
```

- 입력 받은 정보와 랜덤 배정된 정보로 신규 사용자 INSERT

```
142               sql = "INSERT INTO user VALUE (%s, %s, %s, %s, %s, %s, %s, %s)"
143               cursor.execute(sql, (ssn, fname, lname, address, phonecall, maxAcc, curAcc, adNum))
```

- Admin_num이 'adNum'인 관리자의 usercount 1 증가하도록 UPDATE

```
147               sql = """UPDATE admin
148                        SET Usercount = Usercount + 1
149                        WHERE Admin_num=%s"""
150               cursor.execute(sql, adNum)
```

- Ssn이 'ssn'인 사용자의 현재 계좌 수 1 증가하도록 UPDATE

```
155               sql = """UPDATE user
156                        SET Cur_account = Cur_account + 1
157                        WHERE Ssn=%s"""
158               cursor.execute(sql, ssn)
```

- 새로운 계좌 INSERT

```
162           sql = "INSERT INTO account VALUE (%s, %s, %s, %s, %s)"
163           cursor.execute(sql, (accountNum, initAmount, password, openDate, ssn))
```

- 새로운 계좌를 생성할 때 입금내역을 DW_INFORMATION table에 INSERT

```
167           sql = "INSERT INTO dw_information VALUE (%s, %s, %s, %s, %s, %s)"
168           cursor.execute(sql, (1, "d", today, initAmount, "Open", accountNum))
```

- Account_num이 'accountNum'인 계좌 DELETE
  - ACCOUNT와 연관된 DW_INFORMATION도 함께 삭제됨 (CASCADE)

```
184           sql = "DELETE FROM account WHERE Account_num=%s"
185           cursor.execute(sql, accountNum)
```

- Ssn이 'ssn'인 사용자의 현재 계좌 수 1 감소하도록 UPDATE

```
189           sql = "UPDATE user SET Cur_account = Cur_account - 1 WHERE Ssn = %s"
190           cursor.execute(sql, ssn)
```

- Ssn이 'ssn'인 사용자의 'dAccount' 계좌에 'dAmount'만큼 입금하도록 UPDATE

```
209             sql = """UPDATE account
210                      SET Balance = Balance + %s
211                      WHERE Account_num = %s AND Ussn = %s"""
212             cursor.execute(sql, (dAmount, dAccount, ssn))
```

- 입금하려는 'dAccount' 계좌 입출금내역의 Idx SELECT

```
217             sql = "SELECT Idx FROM dw_information WHERE This_account=%s"
218             cursor.execute(sql, dAccount)
```

- DW_INFORMATION table에 새로운 입금내역 INSERT

```
226             sql = "INSERT INTO dw_information VALUE (%s, %s, %s, %s, %s, %s)"
227             cursor.execute(sql, (maxindex+1, "d", today, dAmount, dMessage, dAccount))
```

- 계좌번호가 'wAccount'인 계좌의 password SELECT

```
245             sql = "SELECT password FROM account WHERE Account_num = %s"
246             cursor.execute(sql, wAccount)
```

- 계좌번호가 'wAccount'인 계좌의 Balance SELECT

```
254             sql = "SELECT Balance FROM account WHERE Account_num = %s"
255             cursor.execute(sql, wAccount)
```

- Ssn이 'ssn'인 사용자의 'wAccount' 계좌에서 'wAmount'만큼 출금하도록 UPDATE

```
264             sql = """UPDATE account
265                      SET Balance = Balance - %s
266                      WHERE Account_num = %s AND Ussn = %s"""
267             cursor.execute(sql, (wAmount, wAccount, ssn))
```

- 출금하려는 'wAccount' 계좌 입출금내역의 Idx SELECT

```
272             sql = "SELECT Idx FROM dw_information WHERE This_account=%s"
273             cursor.execute(sql, wAccount)
```

- DW_INFORMATION table에 새로운 출금내역 INSERT

```
281             sql = "INSERT INTO dw_information VALUE (%s, %s, %s, %s, %s, %s)"
282             cursor.execute(sql, (maxindex+1, "w", today, wAmount, wMessage, wAccount))
```

- Ssn이 'ssn'인 사용자의 계좌번호와 잔액 SELECT

```
292             sql = """SELECT Account_num, Balance
293                      FROM account, user
294                      WHERE Ssn=%s AND Ussn=Ssn"""
295             cursor.execute(sql, ssn)
```

- Admin_num이 'adminNum'인 관리자의 관리자 번호 SELECT (존재여부 확인용)

```
321         sql = "SELECT Admin_num FROM admin WHERE Admin_num=%s"
322         cursor.execute(sql, adminNum)
```

- 입력 받은 정보로 새로운 관리자 INSERT

```
336                 sql = "INSERT INTO admin VALUE (%s, %s, %s, %s)"
337                 cursor.execute(sql, (adminNum, fname, lname, 0))
```

- This_account가 'selectAccount', Idx가 'index'인 입출금 내역 DELETE

```
398             sql = "DELETE FROM dw_information WHERE This_account=%s AND Idx=%s"
399             cursor.execute(sql, (selectAccount, index))
```

- Ssn이 'deleteUser'인 사용자 DELETE
  - USER와 연관된 ACCOUNT, DW_INFORMATION도 함께 삭제됨 (CASCADE)

```
425             sql = "DELETE FROM user WHERE Ssn=%s"
426             cursor.execute(sql, deleteUser)
```

- Admin_num이 'adminNum'인 관리자의 Usercount 1 감소하도록 UPDATE

```
430             sql = """UPDATE admin
431                     SET Usercount = Usercount - 1
432                     WHERE Admin_num=%s"""
433             cursor.execute(sql, adminNum)
```

4. 실행

소스코드(bankSimulator.py)가 있는 위치에서 bankSimulator.py 실행

```
C:₩Users₩sdahye₩Desktop₩  ₩한양대₩2-2₩데이터베이스시스템및응용₩실습과제₩P4>bankSimulator.py
(1) User login
(2) Admin login
(3) Exit program
Input:
```