

은행 데이터베이스

데이터베이스_01분반
컴퓨터공학과
20134888 임 형 열
담당교수 정 현 숙

1. 개발 동기

- 짝수 학번(4888) : 은행 데이터베이스

2. 개발 설계 목적

- 자바 GUI로 구현, 고객 현황과 고객별 계좌개설 현황을 확인
- 고객 명단 추가/삭제를 통해 실제 DB까지 추가/삭제가 반영되도록 함

3. 개발 환경

Operating System (OS, Version)	MicroSoft Windows 10.0 (Education) 18362_64Bit
Oracle DataBase (Version)	Oracle DataBase 11g_Express Edition
ERwin Data Modeler (Version)	ERwin Data Modeler_r7 (Release 7)
Eclipse IDE (Version)	Eclipse 2019-06 (4.12.0)
JAVA (JDK / JRE, Version)	1.8.0_221 / Java(TM) SE Runtime Environment (build 1.8.0_221-b11)

4. 요구사항 분석

- 은행에서 은행별 지점, 지점별 계좌개설 현황과 고객 명단을 통합 관리하고자 함
- 초보자들도 쉽게 고객 명단을 추가/삭제할 수 있도록 'SQL문'이 아닌 좀 더 쉬운 'GUI'를 활용하고자 함

5. 논리적 설계 - 데이터모델링 7단계(ERD 도출)

- 1단계. 업무분석

- 1) 실체는 **녹색 볼드체**, 속성은 **청색 볼드체**, 관계(관계 테이블)는(은) **적색 볼드체**로 표시하였음
* 실체는 사각형 박스로, 속성은 밑줄로, 관계는 마름모를 그려 표시해야 함

은행에도 기업은행, 우리은행, 광주은행 등 여러 은행이 있다. 각 은행은 해당하는 **고유번호(코드)**가 부여되며 **소재지(주소)**도 가지고 있다. 또한 은행별로 **지점**이 여러 개 존재하는데 해당 지점도 지점별 **고유번호(코드)**, **지점 이름**과 **지점 소재지(주소)**를 가지고 있다.

은행을 이용하는 **고객**들 또한 존재한다. 각 고객은 **주민등록번호**, **이름**, **전화번호**, **주소**로 구성된다. 그리고 은행별 **개설된 계좌현황**도 존재하며 이는 각각 계좌번호, 계좌 주인의 주민등록번호, 은행과 지점코드, 예/적금 여부와 저축금액으로 구성된다. 더해서 **은행별 가능한 대출현황**도 **대출상품(타입)**, **은행**과 **지점코드**, **대출가능금액**으로 구성된다.

한 계좌는 한 명의 사람만 주인으로 가질 수 있으며, 한 사람은 여러 계좌를 개설할 수 있다. 그러나 대출은 하나의 상품이어도 여러 명이 대출을 받을 수 있으므로 다대다 관계이다. 따라서 **고객별 대출 현황**을 알 수 있어야 한다.

- 2/3/4단계. 실체 및 속성 도출, 속성 중 식별자 도출(파란색 볼드체가 식별자(PK)임)

1) 실체와 그것에 대한 속성은 다음과 같음

- 은행 : **은행코드**, 은행이름, 은행소재지(주소)
- 은행지점 : **은행코드(FK)**, **은행지점코드**, 은행지점명, 은행지점 소재지(주소)
- 고객 : ***주민등록번호**, 고객이름, 고객전화번호, 고객주소
 - * 주민등록번호는 편의상 앞자리만 입력했으며 동일한 생년월일이 없다는 것을 전제로 함(UNIQUE)
- 계좌 : **계좌번호**, **고객 주민등록번호(FK)**, **은행코드(FK)**, **은행지점코드(FK)**, 타입(예/적금), 저축금액
- 대출 : ***대출상품타입**, **은행코드(FK)**, **은행지점코드(FK)**, 대출가능금액
 - * 대출상품타입은 여러 가지가 있으나 본 실습에선 모기지/학자금 두 부류로 나누었음
- **고객대출현황** : **고객 주민등록번호(FK)**, **대출상품타입(FK)**, **은행코드(FK)**, **지점코드(FK)**, 대출식별코드

- 5/6단계 관계 및 관계차수 설정

1) 은행과 은행지점 간의 관계

- 은행은 여러 지점을 가질 수 있으나 지점당 하나의 은행만이 본점으로 존재할 수 있음(weak-entity)

2) 대출현황과 고객 간 관계 : 고객별 대출현황 (N:M)

- 하나의 대출상품은 여러 고객이 받을 수 있음, 한 고객이 여러 대출상품을 가질 수 있음

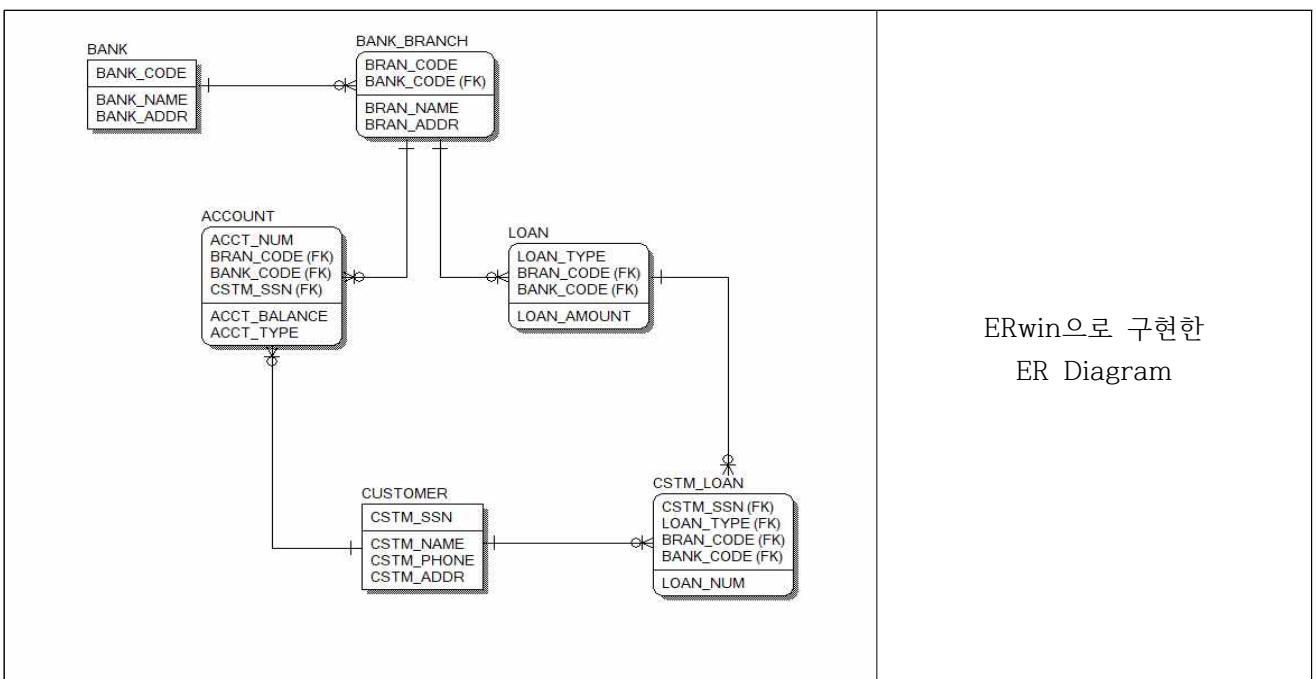
3) 다대다 관계 해소를 위한 관계테이블 생성 : 고객별 대출현황

- 고객별 대출현황 : 고객 주민번호 / 대출타입 / 은행코드와 지점코드를
 각각 고객 / 대출 / 은행지점 테이블에서 참조해 옴(외래키)
 + 대출식별코드(대출한 건수마다 각각 식별번호가 존재해야 하므로)

- 7단계 관계 식별여부 파악

- 1) 은행과 은행지점 간 관계 : 은행코드가 은행지점의 PK(primary key)로 들어감 : 식별 관계
- 2) 은행지점과 대출/계좌 간 관계 : 은행지점의 PK가 대출/계좌의 PK로 들어감 : 식별 관계
- 3) 고객과 계좌 간 관계 : 고객의 PK가 계좌의 PK로 들어감 : 식별 관계
- 4) 고객/고객별 대출현황/대출현황 간 관계 : 고객/대출현황의 PK가 고객별 대출현황의 PK로 : 식별 관계
 - * 부모 테이블의 주 식별자가 자식 테이블의 주 식별자로 들어갈 경우 식별, 일반 속성이면 비식별

- ERwin으로 구현한 모델은 다음과 같음



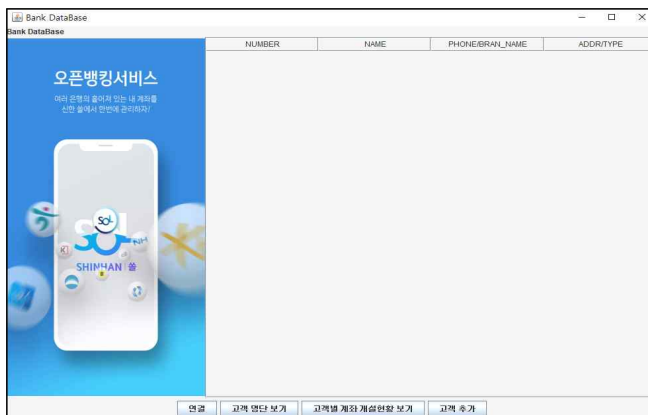
ERwin으로 구현한
ER Diagram

6. 물리적 설계 - ERD(물리) 도출 : 동봉된 '입력자료.hwp' 참조

7. 구현(JAVA GUI)

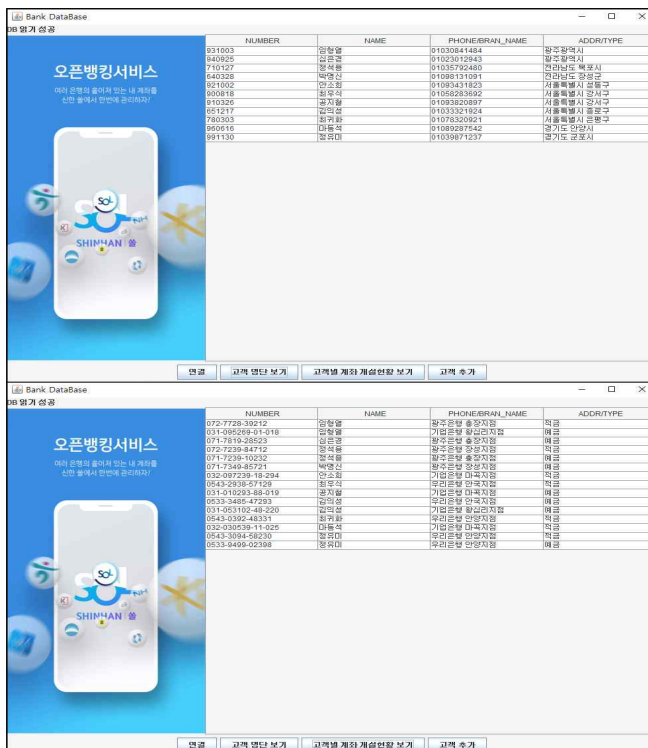
- 자바 소스는 동봉된 자료 BankDB.java 참조
- DB연결 / 고객명단 보기 / 고객별 계좌 개설현황 보기 / 고객 추가 버튼으로 구성
 1. DB 연결 : Oracle DB 연동 기능
 2. 고객 명단 보기 : select문을 이용하여 customer 테이블 속성 보기
 3. 고객별 계좌 개설현황 보기 : select문을 이용하여 acctstat2 뷰 속성 보기
 - * 해당 기능 구현을 위해 계좌번호/이름/지점이름/계좌 타입 속성을 선택한 뷰 acctstat2 생성하였음
- JAVA GUI의 Dialog를 사용, 입력된 값을 문자열 변수에 저장(String SSN에 입력받은 값(주민번호) 저장함)
- 저장된 문자열 변수와 SQL문을 사용하여 완전한 SQL문을 문자열 변수에 저장


```
ex) String Finn = "insert into customer values(" + SSN + "," + name + "," + PHONE + "," + ADDR + ")";
```
- 해당 문자열 변수를 SQL문처럼 사용하여 테이블/뷰의 정보를 가져오거나 추가/삭제하는 기능을 구현하였음
- 시작 화면(BankDB.java)



BankDB.java 파일 실행시 초기화면
연결버튼을 눌러 Oracle DB에 연결


- 고객 명단 보기 / 고객별 계좌 개설현황 보기 버튼 누른 후 화면



고객명단 보기 버튼을 누른 후 화면
고객 테이블 정보가 출력됨
(SELECT문)

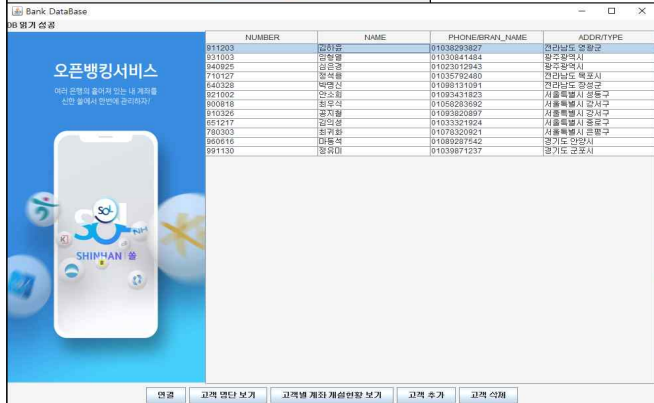
고객별 계좌 개설현황 보기 버튼을 누른 후 화면
계좌 개설현황 뷰(ACCTSTAT2)의 정보가 출력됨
(SELECT문)

- 고객 추가




고객 추가 버튼을 누를 경우
순차적으로 4개의 Dialog 창이 생성됨

빈 칸에 해당하는 값을 입력하고 확인을 누르면
해당하는 문자열 변수에 입력한 값이 저장됨




입력한 값
(911203, 김하윤, 01038293827, 전라남도 영광군)
에 해당하는 행이 추가되었음을 확인

- 고객 삭제



고객 삭제 버튼을 누를 경우
1개의 Dialog 창이 생성됨(PK - 주민번호)
입력받은 값을 문자열 변수에 저장 후
이를 활용하여 delete 연산 실행
(where절, 입력한 값과 동일한 PK 가진 행 삭제하도록)



입력한 값
(911203)
에 해당하는 행이 삭제되었음을 확인

- 상위 기술된 모든 그림 파일은 압축파일의 사진 폴더에, 실행 영상 캡처는 동영상 폴더에 모아놓았습니다.
잘 보이지 않아 확인하기 힘들 경우 해당 경로의 파일을 참고해주시면 감사하겠습니다.

8. 소감

한 학기동안 계속 실습을 진행해서 그런 것인지 ER Diagram, DB 설계/SQL문 작성에는 그렇게 많은 시간이 소요되지 않았습니다. 그러나 JAVA GUI를 구현하는 것에 있어 많은 어려움을 겪었고, 2학년에 수강했던 JAVA 교재를 보고 공부해가며 소스를 추가하느라 이 부분에서 많은 시간이 소요되었습니다. 또한 과제 보고서 양식에서처럼 새로운 창을 띄워서 일괄적으로 입력한 후 그것을 기반으로 SQL문을 수행하게 하고 싶었으나 이 또한 마음대로 되지 않아 Dialog를 활용하여 해결하였습니다.

이번 과제에서 핵심은 그동안 배웠던 모든 개념을 활용하는 것(A to Z, 데이터 모델 구성의 7단계 방법부터 컬럼의 데이터 타입, 제약조건과 정규화/뷰를 통해 적절하게 제어하는 것까지)이라고 생각합니다. 또한 JAVA GUI를 활용하기 위해 이 부분에 대한 지식도 어느정도는 갖추고 있어야 한다고 봅니다. (저는 다행히 이에 대한 지식이 남아있어서 과제를 어느정도 수행하였습니다만, 한 학기 동안 수강한 데이터베이스 과목에 대한 전반적 이해가 없는 학생들은 첫 단계부터 어려움을 겪을 것 같습니다.)

마치며, 한 학기 동안 데이터베이스 과목을 수강하며 가장 크게 느꼈던 점이라면 데이터베이스에서 SQL문도 중요하지만 가장 필수적으로 알아두어야 할 것이 '관계' 라고 생각합니다. 테이블 간, 속성 간 등등 여러 구성 요소에서 어떤 관계인가에 따라 구현해야 할 것이 크게 달라지기 때문입니다. 추가적으로 이 과목을 통해 자기 주도적 학습(실습)을 많이 하였고, 그로 인해 완벽하게까지는 아니더라도 데이터베이스의 기본 개념을 잡고 가게 된 것 같아서 마음이 놓입니다. 더하여 1학기에 배웠던 빅 데이터(분산 데이터베이스)에서 이해되지 않았던 개념도 어느 정도 잡게 된 것 같아 뿌듯하기도 합니다.