**2018 Database System Project #2**

Constructing SQL

컴퓨터공학과

박석 교수님

학번 : 20141556

이름 : 이성문

2018.5.11

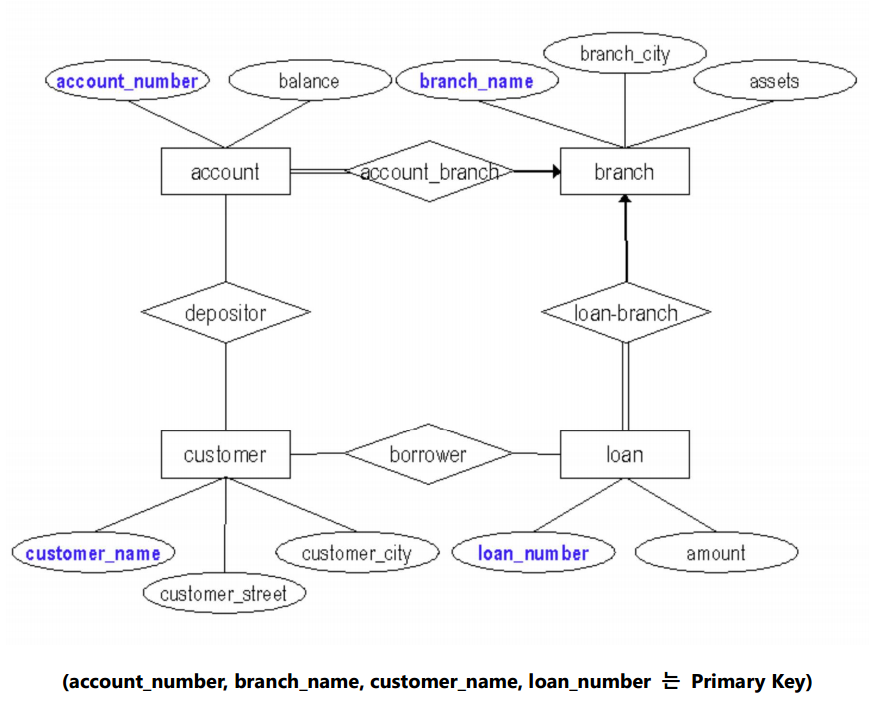
**<목차>**

1. 프로젝트 목표
2. ER 다이아그램
3. Physical Mode
4. 질의 응답 SQL문 및 결과

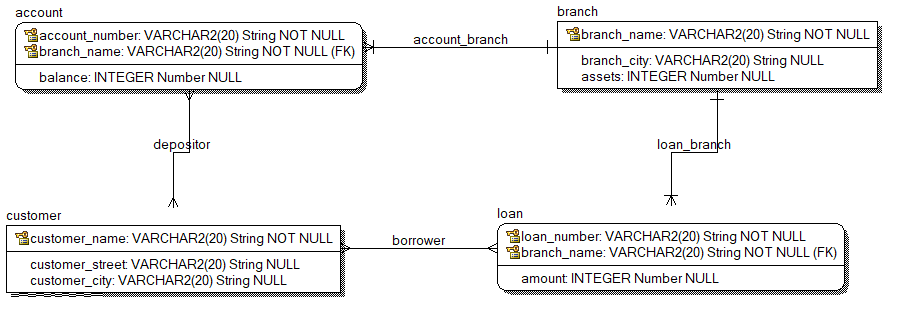
# 1. 프로젝트 목표

\* 프로젝트 목표 : 주어진 ER 다이어그램에 맞는 데이터베이스를 설계하고 입력 데이터를 바탕으로 속성별 타입과 제약을 설정한다. 즉 ER-win을 이용하여 모드 별로 설계한다. 이를 바탕으로 실제 데이터 베이스 서버와 연동한 후 스키마를 생성하고 주어진 문제에 대한 질의를 SQL문으로 작성하여 그에 따른 결과를 확인한다.

# 2. ER 다이아그램

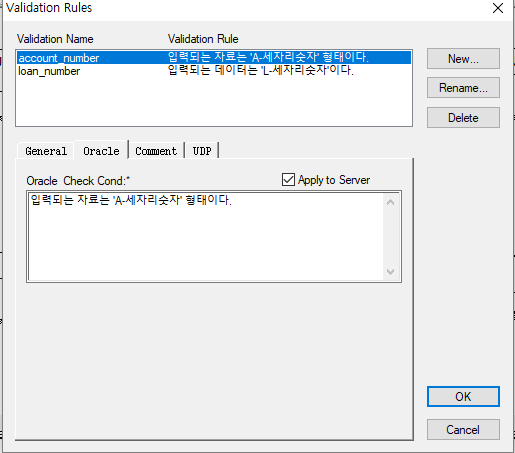


# 3. Physical mode

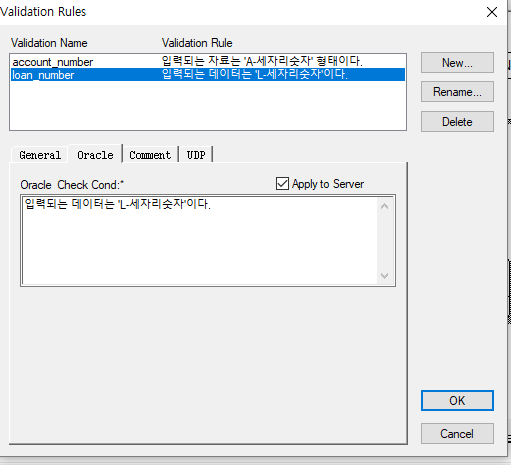


# 3-1. Validation rules

1. Account\_number : 이는 account entity의 account\_number의 validation rule이다.



1. Loan\_number : 이는 loan entity의 loan\_number에 대한 validation rule이다.



3-2. Domain

이번 프로젝트에서는 다양한 데이터들이 많지 않으므로 Erwin 프로그램 기존에 존재하는 Domain을 사용하였다.

1. String : VARCHAR(20)

최대 20까지의 문자 개수를 허용하는 Domain이다. 밑의 attribute들의 데이터들은 문자열 길이 20을 넘는 경우가 없다.

사용 attribute : account\_number, branch\_name,branch\_city, customer\_name, customer\_city, customer\_street, loan\_number, branch\_name

1. Number : Integer

일반 상수로 이 데이터 모델에서는 보통 0이상의 정수를 뜻한다.

사용 attribute : balance, assets, amount

3-3. account

계좌를 나타내는 Entity Set이다.

\*account Entity Set의 Attribute :

(1) account\_number : 계좌의 계좌번호를 나타내는 Attribute이다. 사람마다, 데이터마다, 계좌번호가 다를수 밖에 없기 때문에 account\_number를 Primary Key로 사용하였다..

(2) balance : 계좌의 잔액을 나타내는 Attribute이다.

\* account 의 Relationship :

account 는 customer와 depositor Relationship을 이룬다. 여기서 Cardinality는 many-to-many이다. 그리고 account는 branch와 account\_branch Relationship을 이루고 이때 Cardinality는 many-to-one이다. 그리고 account는 이 Relationship에 total participation임을 알수 있다.. 따라서 account는 branch의 Primary Key인 branch\_name을 Foreign Key로 가진다.

3-4. customer

고객을 나타내는 Entity Set이다.

\*customer Entity Set의 Attribute :

(1) customer\_name : 고객의 이름을 나타내는 Attribute이다. 고객들은 전부 서로 다른 이름을 가지고 있기 때문에 account\_number를 Primary Key로 사용하였다.

(2) customer\_street: 고객의 거주 거리를 나타내는 Attribute이다.

(3) customer\_ciy: 고객의 거주 도시를 나타내는 Attribute이다.

\*customer의 Relationship :

customer는 account와 depositor Relationship을 이루고 이때 Cardinality는 many-to-many이다. 그리고 customer는 loan와 borrower Relationship을 이루고 이때 Cardinality는 many-to-many이다.

3-5. branch

지점을 나타내는 Entity Set이다.

\*branch Entity Set의 Attribute :

(1) branch\_name : 지점의 이름을 나타내는 Attribute이다. 지점마다 이름이 전부 서로 다르기 때문에 account\_number를 Primary Key로 사용하였다.

(2) assets : 지점의 자산을 저장하는 Attribute이다.

(3) branch\_city : 지점의 입주 도시를 나타내는 Attribute이다.

\* branch의 Relationship :

branch는 account와 account\_branch Relationship을 이루고 이때 Cardinality는 one-to-many이다. 그리고 branch는 loan와 loan\_branch Relationship을 이루고 이때 Cardinality는 one-to-many이다. 이때 loan은 이 Relationship에 total participation이다. loan은 branch의 Primary Key인 branch\_name을 Foreign Key로 가진다.

3-6. loan

대출을 나타내는 Entity Set이다.

\*loan Entity Set의 Attribute :

(1) loan\_number : 대출의 번호를 나타내는 Attribute이다. 대출번호가 전부 다 서로 다르기 때문에 account\_number를 Primary Key로 사용하였다.

(2) amount : 대출액수를 저장하는 Attribute이다.

\* loan의 Relationship :

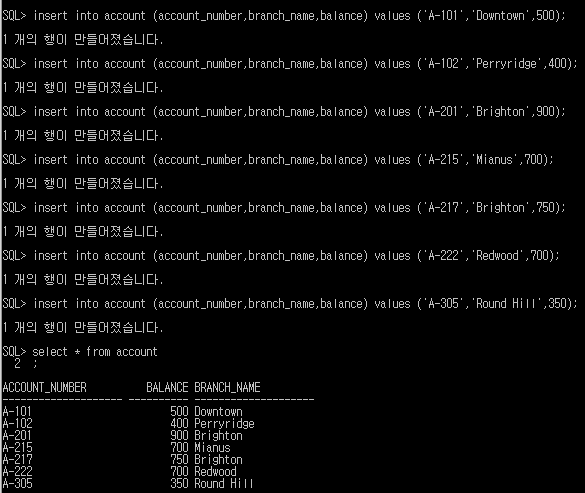
loan은 customer와 borrower Relationship을 이루고 이때 Cardinality는 many-to-many이다. 그리고 loan는 branch와 loan\_branch Relationship을 이루고 이때 Cardinality는 many-to-one이다. 이때 loan은 이 Relationship에 total participation이다. loan은 branch의 Primary key인 branch\_name을 외래키로 가진다.

# 4. 질의 응답 SQL문 및 결과

**(1) 다음 데이터를 입력하시오 (insert)**

여기서 account의 입력을 먼저할 경우 branch\_name 외래 키가 존재해 account 입력에 오류가 뜬다. 왜냐하면 branch entity가 아직 입력이 안돼있으므로 branch 입력을 먼저 하였다.

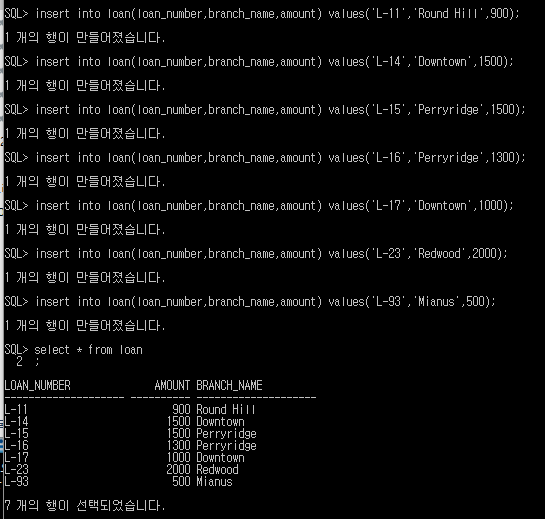
Account의 입력과 결과



Branch의 입력과 결과



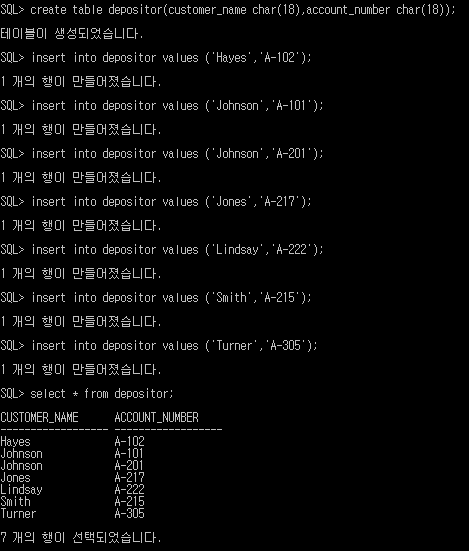
Loan의 입력과 결과



Borrower 테이블의 생성과 입력과 결과



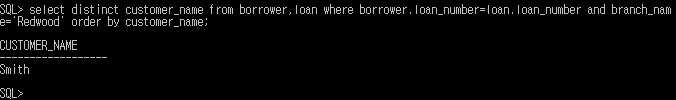
Depositor table의 생성과 입력과 결과



**(2) 중복되지 않은 모든 지점들의 이름을 구하라.**



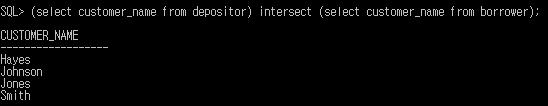
**(3) Redwood 지점의 대출을 가진 모든 고객들을 알파벳 순서로 나열하라.**



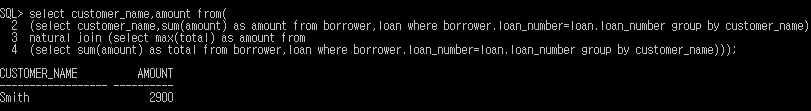
**(4) 은행에서 대출, 계좌 혹은 둘 다를 가진 모든 고객을 나열하라**



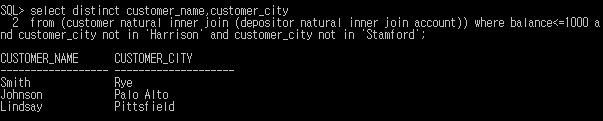
**(5) 은행에 대출과 계좌 모두를 가진 모든 고객을 나열하라.**



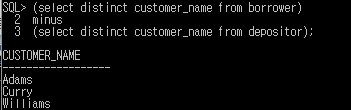
**(6) 대출 총액이 가장 큰 고객의 이름과 대출 총액을 구하여라**



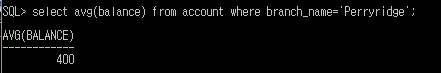
**(7) Harrison과 Stamford에 살지 않으면서 계좌에 잔고가 1000이하 있는 고객의 이름과 고객이 사는 도시를 구하라.**



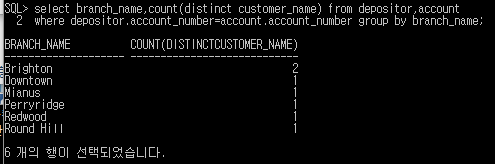
**(8) 은행에 계좌는 가지고 있지 않지만 대출은 가지고 있지 않는 모든 고객들을 나열하라.**



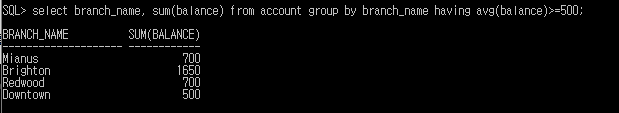
**(9) Perryridge 지점에서 계좌의 평균 잔고를 구하여라**



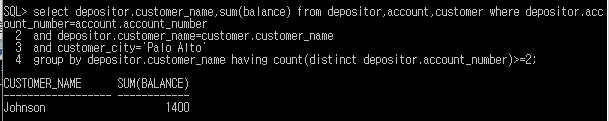
**(10) 각 지점의 예금자들의 수를 구하라.**



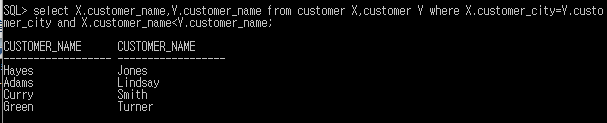
**(11) 평균 잔고가 $500 이상인 지점 이름과 총 잔고를 나열하라.**



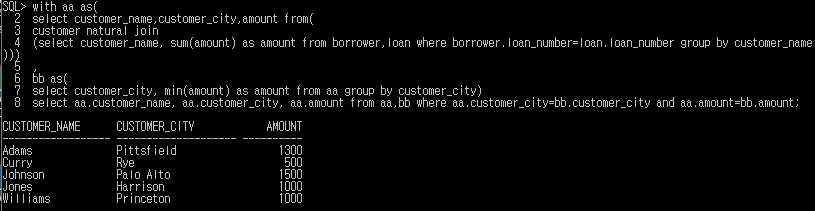
**(12) Palo Alto에 살고 최소한 두 개의 계좌를 가진 각각의 고객들의 이름과 잔고의 합을 구하라**



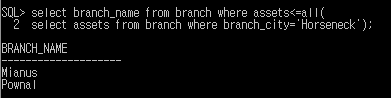
**(13) 같은 도시에 사는 고객의 이름의 쌍을 구하여라**



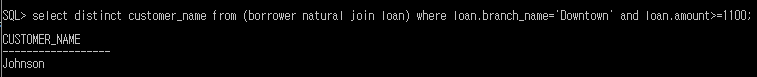
**(14) 각 도시 별로 가장 낮은 대출 총액을 가지고 있는 고객의 이름과 대출 총액을 구하여라. 단, 대출을 가진 고객이 살지 않는 도시는 표시하지 않는다.**



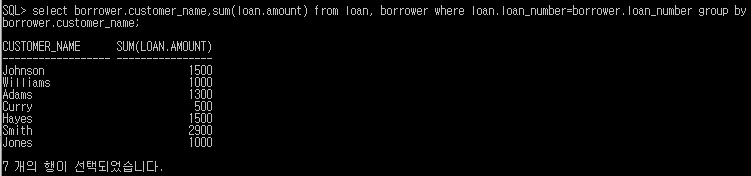
**(15) Horseneck에 있는 각 지점보다 작거나 같은 자산 값을 갖는 모든 지점들의 이름을 나열하라.**



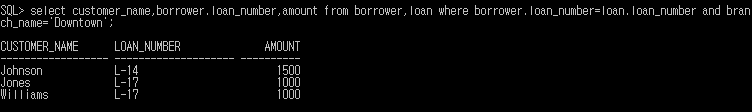
**(16) Downtown 지점에서 $1100 이상의 대출 총액을 지닌 고객들을 전부 구하라**



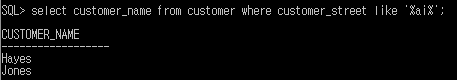
**(17) 은행에 대출을 가지고 있는 모든 고객들에 대해 그들의 이름과 대출 총액을 구하라**



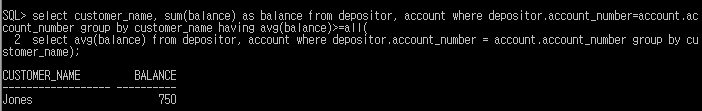
**(18) Downtown 지점의 모든 대출에 대해여 고객의 이름과 대출 번호, 대출액을 구하라.**



**(19) 이름에 'ai'이라는 부분 문자열이 포함된 거리에 살고 있는 모든 고객들의 이름을 구하여라**



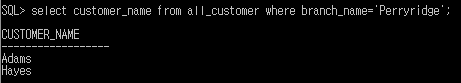
**(20) 가장 높은 평균 잔고를 가진 고객의 이름과 총잔고를 구하라.**



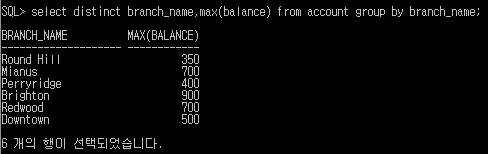
**(21) 지점 이름과 그 지점에 계좌나 대출 둘 중 하나를 가진 고객 이름으로 구성된 View를 작성하라. (단 View의 이름은 all\_customer이다.)**



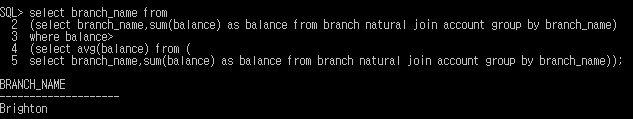
**(22) (21)에서 생성된 View를 이용하여 Perryridge 지점의 모든 고객 이름을 나열하라.**



**(23) 각 지점에서 총 잔고의 최대값을 나열하라.**



**(24) 모든 지점의 총 계좌 예금의 평균보다 많은 총 계좌 예금을 갖는 모든 지점을 나열하라**



**(25) 평균 대출 총액보다 적은 대출 총액을 가지고 있는 고객의 이름과 대출 총액을 구하라.**

