

# Advanced DataBases C4 Group

## Bank Management System

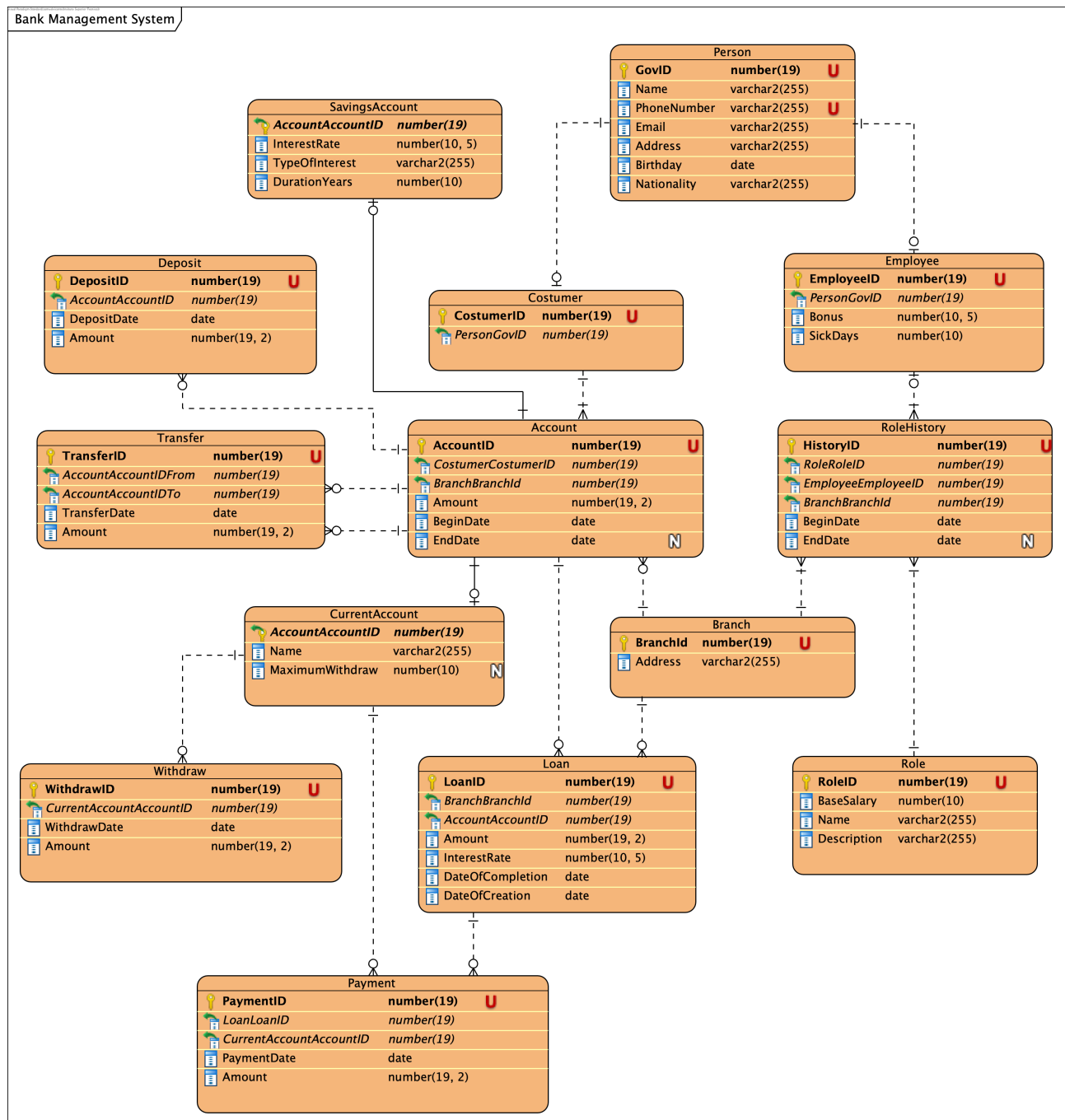
Samuel Lúcio Vicente, 251720      Daniel Silva, 251702  
Jorge Marrero Camiruaga, 251438

Politechnika Wrocławska  
today

### Short Description

This database models a bank with Accounts that can either be a SavingsAccount or a CurrentAccount. There are some operations that are permitted, Withdraw from the CurrentAccount, Transfer between accounts, Loan, Pay the loan, Deposit on Account. There are employees that work in a branch that have roles, this database tracks the history of the all the employees on every branch, and the previous roles that some particular employee has had and which branch he has working at.

# ERD



## Schema

```

1 CREATE TABLE Person (
2   GovID number(19) NOT NULL,
3   Name varchar2(255) NOT NULL,
4   PhoneNumber varchar2(255) NOT NULL UNIQUE,
5   Email varchar2(255) NOT NULL,
6   Address varchar2(255) NOT NULL,
7   Birthday date NOT NULL,
8   Nationality varchar2(255) NOT NULL,
9   PRIMARY KEY (GovID));
    
```

```

10
11 CREATE TABLE Costumer (
12     CostumerID number(19) GENERATED AS IDENTITY,
13     PersonGovID number(19) NOT NULL,
14     PRIMARY KEY (CostumerID));
15
16 CREATE TABLE Employee (
17     EmployeeID number(19) GENERATED AS IDENTITY,
18     PersonGovID number(19) NOT NULL,
19     Bonus number(10, 5) NOT NULL CHECK(Bonus>=0),
20     SickDays number(10) NOT NULL CHECK(SickDays<10),
21     PRIMARY KEY (EmployeeID));
22
23 CREATE TABLE Account (
24     AccountID number(19) GENERATED AS IDENTITY,
25     CostumerCostumerID number(19) NOT NULL,
26     BranchBranchId number(19) NOT NULL,
27     Amount number(19, 2) NOT NULL CHECK(Amount>=0),
28     BeginDate date NOT NULL,
29     EndDate date,
30     PRIMARY KEY (AccountID));
31
32 CREATE TABLE Branch (
33     BranchId number(19) GENERATED AS IDENTITY,
34     Address varchar2(255) NOT NULL,
35     PRIMARY KEY (BranchId));
36
37 CREATE TABLE RoleHistory (
38     HistoryID number(19) GENERATED AS IDENTITY,
39     RoleRoleID number(19) NOT NULL,
40     EmployeeEmployeeID number(19) NOT NULL,
41     BranchBranchId number(19) NOT NULL,
42     BeginDate date NOT NULL,
43     EndDate date,
44     PRIMARY KEY (HistoryID));
45
46 CREATE TABLE Role (
47     RoleID number(19) GENERATED AS IDENTITY,
48     BaseSalary number(10) NOT NULL CHECK(BaseSalary>0),
49     Name varchar2(255) NOT NULL,
50     Description varchar2(255) NOT NULL,
51     PRIMARY KEY (RoleID));
52
53 CREATE TABLE Loan (
54     LoanID number(19) GENERATED AS IDENTITY,
55     BranchBranchId number(19) NOT NULL,
56     AccountAccountID number(19) NOT NULL,
57     Amount number(19, 2) NOT NULL CHECK(Amount>0),
58     InterestRate number(10, 5) NOT NULL,
59     DateOfCompletion date NOT NULL,
60     DateOfCreation date NOT NULL,
61     PRIMARY KEY (LoanID));
62
63 CREATE TABLE Payment (
64     PaymentID number(19) GENERATED AS IDENTITY,
65     LoanLoanID number(19) NOT NULL,
66     CurrentAccountAccountID number(19) NOT NULL,
67     PaymentDate date NOT NULL,
68     Amount number(19, 2) NOT NULL CHECK(Amount>0),
69     PRIMARY KEY (PaymentID));
70
71 CREATE TABLE SavingsAccount (

```

```

72     AccountAccountID number(19) NOT NULL,
73     InterestRate number(10, 5) NOT NULL CHECK(InterestRate>0),
74     TypeOfInterest varchar2(255) NOT NULL,
75     DurationYears number(10) NOT NULL CHECK(DurationYears>0),
76     PRIMARY KEY (AccountAccountID));
77
78 CREATE TABLE Deposit (
79     DepositID number(19) GENERATED AS IDENTITY,
80     AccountAccountID number(19) NOT NULL,
81     DepositDate date NOT NULL,
82     Amount number(19, 2) NOT NULL CHECK(Amount>0),
83     PRIMARY KEY (DepositID));
84
85 CREATE TABLE Transfer (
86     TransferID number(19) GENERATED AS IDENTITY,
87     AccountAccountIDFrom number(19) NOT NULL,
88     AccountAccountIDTo number(19) NOT NULL,
89     TransferDate date NOT NULL,
90     Amount number(19, 2) NOT NULL CHECK(Amount>0),
91     PRIMARY KEY (TransferID));
92
93 CREATE TABLE Withdraw (
94     WithdrawID number(19) GENERATED AS IDENTITY,
95     CurrentAccountAccountID number(19) NOT NULL,
96     WithdrawDate date NOT NULL,
97     Amount number(19, 2) NOT NULL CHECK(Amount>0),
98     PRIMARY KEY (WithdrawID));
99
100 CREATE TABLE CurrentAccount (
101     AccountAccountID number(19) NOT NULL,
102     Name varchar2(255) NOT NULL,
103     MaximumWithdraw number(10),
104     PRIMARY KEY (AccountAccountID));
105
106 ALTER TABLE Costumer ADD CONSTRAINT FKCostumer923053 FOREIGN KEY (PersonGovID) REFERENCES Person (
    ↳ GovID);
107
108 ALTER TABLE Employee ADD CONSTRAINT FKEmployee249023 FOREIGN KEY (PersonGovID) REFERENCES Person (
    ↳ GovID);
109
110 ALTER TABLE Account ADD CONSTRAINT FKAccount895601 FOREIGN KEY (CostumerCostumerID) REFERENCES
    ↳ Costumer (CostumerID);
111
112 ALTER TABLE RoleHistory ADD CONSTRAINT FKRoleHistor647811 FOREIGN KEY (RoleRoleID) REFERENCES Role (
    ↳ RoleID);
113
114 ALTER TABLE RoleHistory ADD CONSTRAINT FKRoleHistor516821 FOREIGN KEY (EmployeeEmployeeID) REFERENCES
    ↳ Employee (EmployeeID);
115
116 ALTER TABLE RoleHistory ADD CONSTRAINT FKRoleHistor171832 FOREIGN KEY (BranchBranchId) REFERENCES
    ↳ Branch (BranchId);
117
118 ALTER TABLE Loan ADD CONSTRAINT FKLoan357293 FOREIGN KEY (BranchBranchId) REFERENCES Branch (BranchId)
    ↳ ;
119
120 ALTER TABLE SavingsAccount ADD CONSTRAINT FKSavingsAcc25288 FOREIGN KEY (AccountAccountID) REFERENCES
    ↳ Account (AccountID);
121
122 ALTER TABLE Payment ADD CONSTRAINT FKPayment955503 FOREIGN KEY (LoanLoanID) REFERENCES Loan (LoanID);
123
124 ALTER TABLE Deposit ADD CONSTRAINT FKDeposit626030 FOREIGN KEY (AccountAccountID) REFERENCES Account (
    ↳ AccountID);

```

```

125
126 ALTER TABLE Transfer ADD CONSTRAINT FKTransfer731892 FOREIGN KEY (AccountAccountIDFrom) REFERENCES
    ↳ Account (AccountID);
127
128 ALTER TABLE Transfer ADD CONSTRAINT FKTransfer432158 FOREIGN KEY (AccountAccountIDTo) REFERENCES
    ↳ Account (AccountID);
129
130 ALTER TABLE Payment ADD CONSTRAINT FKPayment25568 FOREIGN KEY (CurrentAccountAccountID) REFERENCES
    ↳ CurrentAccount (AccountAccountID);
131
132 ALTER TABLE Withdraw ADD CONSTRAINT FKWithdraw546165 FOREIGN KEY (CurrentAccountAccountID) REFERENCES
    ↳ CurrentAccount (AccountAccountID);
133
134 ALTER TABLE CurrentAccount ADD CONSTRAINT FKCurrentAcc16041 FOREIGN KEY (AccountAccountID) REFERENCES
    ↳ Account (AccountID);
135
136 ALTER TABLE Account ADD CONSTRAINT FKAccount396299 FOREIGN KEY (BranchBranchId) REFERENCES Branch (
    ↳ BranchId);
137
138 ALTER TABLE Loan ADD CONSTRAINT FKLoan608793 FOREIGN KEY (AccountAccountID) REFERENCES Account (
    ↳ AccountID);

```

## Transactions

### 1:Changing Query

#### Description

There is 5 CurrentAccount in the system. This transaction doubles the amount of the account that has the biggest value in the database.

#### SQL

```

1 BEGIN TRANSACTION
2 UPDATE Account
3 SET amount = amount*2
4 WHERE AccountID = (
5     SELECT AccountID
6     FROM Costumer c
7     INNER JOIN Person p
8         ON p.GovID = c.PersonGovID
9     INNER JOIN Account
10        ON CostumerID = CostumerCostumerID
11     INNER JOIN CurrentAccount
12        ON AccountAccountID = AccountID
13     WHERE amount >= ALL(
14         SELECT MAX(amount)
15         FROM Account
16     )
17 );
18 COMMIT

```

### 2:Changing Query

#### Description

This transaction preforms a withdraw of 100 units on the CurrentAccount with the AccountAccountID 1. To do this we must first check if the amount we want to withdraw is smaller than the CurrentAccount MaximumWithdraw, after that we update the amount and add an entry to the Withdraw ledger.

## SQL

```
1 BEGIN TRANSACTION
2 UPDATE Account
3 SET amount =
4     CASE
5         WHEN 100<(
6             SELECT MaximumWithdraw
7             FROM CurrentAccount
8             WHERE AccountAccountID=1)
9             THEN amount - 100
10        ELSE amount
11    END
12 WHERE AccountID=(
13     SELECT AccountID
14     FROM CurrentAccount INNER JOIN Account
15         ON AccountID=AccountAccountID
16     WHERE AccountAccountID=1);
17
18 UPDATE Account
19 SET EndDate =
20     CASE
21         WHEN amount=0 THEN CURRENT_DATE
22         ELSE null
23     END
24 WHERE AccountID = (
25     SELECT AccountID
26     FROM CurrentAccount INNER JOIN Account
27         ON AccountID=AccountAccountID
28     WHERE AccountAccountID=1);
29
30 INSERT INTO Withdraw(WithdrawDate, Amount, CurrentAccountAccountID) VALUES(CURRENT_DATE, 100, 1);
31 COMMIT
```

## 3:Changing Query

### Description

This transaction performs a transaction between the SavingsAccount 6 and the CurrentAccount 1, it checks if the period of the SavingsAccount has passed and if so transfers the amount plus interest, if not only the amount. To do this we first add the amount to the CurrentAccount then we add a entry to the Transaction ledger and then we update the amount on the SavingsAccount

## SQL

```
1 BEGIN TRANSACTION
2 UPDATE Account
3 SET amount =
4     CASE
5         WHEN (
6             SELECT EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM (
7                 SELECT BeginDate
8                 FROM Account INNER JOIN SavingsAccount
9                     ON AccountID=AccountAccountID
10                WHERE AccountAccountID=7))
11             AS year FROM dual) > (
12             SELECT DurationYears
13             FROM SavingsAccount
14             WHERE AccountAccountID=7)
15         THEN amount + (
```

```

16         SELECT (amount+1)*12*DurationYears*InterestRate
17         FROM SavingsAccount INNER JOIN Account
18         ON AccountID=AccountAccountID
19         WHERE AccountAccountID=7)
20     ELSE amount + (
21     SELECT amount
22     FROM SavingsAccount INNER JOIN Account
23     ON AccountID=AccountAccountID
24     WHERE AccountAccountID=7)
25 END
26 WHERE AccountID = (
27     SELECT AccountID
28     FROM CurrentAccount INNER JOIN Account
29     ON AccountID=AccountAccountID
30     WHERE AccountAccountID=1);
31
32 INSERT INTO Transfer(TransferDate, Amount, AccountAccountID, AccountAccountID2) VALUES (CURRENT_DATE,
↵ (
33 Select
34 CASE
35     WHEN (
36         SELECT EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM (
37             SELECT BeginDate
38             FROM Account INNER JOIN SavingsAccount
39             ON AccountID=AccountAccountID WHERE AccountAccountID=7))
40         AS year FROM dual) > (
41             SELECT DurationYears
42             FROM SavingsAccount
43             WHERE AccountAccountID=7)
44         THEN amount*12*DurationYears*InterestRate
45         ELSE amount
46     END
47 FROM SavingsAccount INNER JOIN Account
48     ON AccountID=AccountAccountID
49     WHERE AccountAccountID=7), '7', '1');
50
51 UPDATE Account
52 SET amount = 0
53 WHERE AccountID = (
54     SELECT AccountID
55     FROM SavingsAccount INNER JOIN Account
56     ON AccountID=AccountAccountID
57     WHERE AccountAccountID=7);
58 COMMIT

```

## 4:Changing Query

### Description

This transaction upgrades the role of the employee that has been working as a 4 Role for the longest time and upgrades him to a 5

### SQL

```

1 BEGIN TRANSACTION
2 UPDATE RoleHistory
3     SET EndDate = CURRENT_DATE
4     WHERE EmployeeEmployeeID = (
5         SELECT EmployeeEmployeeID
6         FROM RoleHistory
7         WHERE BeginDate = (

```

```
8      SELECT MIN(BeginDate)
9              FROM roleHistory
10             WHERE RoleRoleID = '4' AND EndDate = NULL)
11      AND RoleRoleID = '4' AND EndDate = NULL
12
13 INSERT INTO RoleHistory VALUES('10004',CURRENT_DATE,NULL,'5','84972','16516');
14 COMMIT
```

## 5:Changing Query

Description

SQL

---

## 6:Selecting Query

Description

SQL

---

## 7:Selecting Query

Description

SQL

---