

**Universidade Eduardo Mondlane**

Faculdade de Ciências

Departamento de Matemática e Informática

**Tema:** Sistema de Gestão de Teatro

**Curso:** Licenciatura em Informática

**Nível:** III

**Regime:** Pós-laboral

**Disciplina:** Base de Dados II

**Discentes**:

Edson Da Cruz

**Docente**:

Jordão Uache

Maputo, Outubro de 2024

**Índice**

**[1.](#_Toc91011418)****[Contextualização](#_Toc91011418)** [3](#_Toc91011418)

**[2.](#_Toc91011419)****[Descrição do Sistema](#_Toc91011419)** [4](#_Toc91011419)

[3. Modelo de Base de Dados 5](#_Toc91011420)

[4.](#_Toc91011421) **[Scripts](#_Toc91011421)** [6](#_Toc91011421)

**[FUNÇÕES](#_Toc91011422)** [6](#_Toc91011422)

**[Is\_valid\_login](#_Toc91011423)** [6](#_Toc91011423)

**[Create\_bank\_account](#_Toc91011424)** [6](#_Toc91011424)

**[Get\_commission\_by trans\_type](#_Toc91011425)** [7](#_Toc91011425)

**[PROCEDIMENTOS](#_Toc91011426)** [8](#_Toc91011426)

**[Deposit](#_Toc91011427)** [8](#_Toc91011427)

**[Withdraw](#_Toc91011428)** [10](#_Toc91011428)

**[Get\_account\_statement](#_Toc91011429)** [13](#_Toc91011429)

**[TRIGGERS](#_Toc91011430)** [13](#_Toc91011430)

**[set\_transaction\_commission](#_Toc91011431)** [13](#_Toc91011431)

**[bank\_client\_after\_insert](#_Toc91011432)** [13](#_Toc91011432)

**[Insert\_transaction\_log](#_Toc91011433)** [14](#_Toc91011433)

**[Insert\_commission\_log](#_Toc91011434)** [15](#_Toc91011434)

# **Contextualização**

O uso de Sistemas de Gestão Bancária tornou-se algo indispensável nos últimos anos, tanto com a evolução da tecnologia e com o crescimento da população e a aderência aos serviços da banca.

E no presente trabalho, foi desenvolvido um Sistema de Gestão Bancária em PL/SQL e usamos o modelo de Entidade e Relacionamento, para atender aos pontos propostos para o seu desenvolvimento.

# **Descrição do Sistema**

* É necessário garantir o cadastro dos utilizadores e a respectivas senhas. E cada cliente tem sempre uma conta Poupança e à Ordem. Para aceder a conta deve sempre fornecer o user e password. Sempre que depositar um valor, 5% desse valor vai para a sua conta poupança.
* Garantir que não se levante mais do que o valor existente na conta à ordem. Só se pode levantar dinheiro na conta Poupança se a conta a Ordem estiver sem dinheiro.
* Listar todas as transacções feitas numa determinada conta entre duas datas.

.

O sistema deverá ainda permitir as seguintes operações:

* Levantamento
* Deposito
* Extrato
* Consulta de Saldo

# Modelo de Base de Dados

O modelo usado foi o Modelo de Entidade e Relacionamento (E-R), que representa a realidade através da entidade, relacionamento entre as várias entidades e atributos que compõem uma entidade.

# **Scripts**

## **FUNÇÕES**

### **Is\_valid\_login**

Função que faz a validação dos dados no login. Verifica se o username, user\_password e a conta bancária fornecida existem. A funcção retorna **True** caso verdadeiro e **False** caso contrário.

create or replace function is\_valid\_login (v\_bank\_account IN bank\_account.bank\_account\_id%type,

    v\_username IN bank\_client.username%type,

    v\_password IN bank\_client.user\_password%type)

    return boolean

    is

 v\_client bank\_client.bank\_client\_id%type;

begin

    select count(bank\_account.bank\_account\_id) into v\_client from bank\_client

         join client\_bank\_account on client\_bank\_account.bank\_client\_id = bank\_client.bank\_client\_id

         join bank\_account on bank\_account.bank\_account\_id = client\_bank\_account.bank\_account\_id

         join account\_type on account\_type.account\_type\_id = bank\_account.account\_type\_id

         where username like v\_username and user\_password like v\_password and bank\_account.bank\_account\_id = v\_bank\_account;

    if v\_client > 0 then

        return true;

    else

        return false;

    end if;

end;

### **Create\_bank\_account**

Função que cria a conta bancária, inserindo na tabela Bank\_Account e retorna o bank\_account\_id para ser usado em outras operações.

CREATE OR REPLACE FUNCTION CREATE\_BANK\_ACCOUNT (V\_ACCOUNT\_TYPE BANK\_ACCOUNT.ACCOUNT\_TYPE\_ID%TYPE)

 RETURN bank\_account.bank\_account\_id%TYPE IS

 V\_BANK\_ACCOUNT\_ID BANK\_ACCOUNT.bank\_account\_id%TYPE;

 BEGIN

    INSERT INTO BANK\_ACCOUNT (ACCOUNT\_TYPE\_ID) VALUES (V\_ACCOUNT\_TYPE) RETURNING BANK\_ACCOUNT\_ID INTO V\_BANK\_ACCOUNT\_ID;

    RETURN V\_BANK\_ACCOUNT\_ID;

 END;

### **Get\_commission\_by trans\_type**

Esta função irá retornar o valor da comissão associada ao tipo de transação fornecida, onde pode ser deposito ou levantamento. Os tipos de transação podem ter zero ou mais comissões associadas.

create or replace function get\_commission\_by\_trans\_type (

    v\_transaction\_type in transaction\_type.transaction\_type\_id%type)

return commission.commission%type is

v\_total\_commission commission.commission%type := 0;

v\_commissions\_amount commission.commission%type;

v\_counter integer;

e\_notfound exception;

e\_invalid exception;

cursor c\_commissions is

    select commission.commission

    from transaction\_commission

    join commission on commission.commission\_id = transaction\_commission.commission\_id

    where transaction\_type\_id = v\_transaction\_type;

begin

    if v\_transaction\_type <= 0 then

        raise e\_invalid;

    end if;

    select count(\*) into v\_counter from transaction\_type where transaction\_type\_id = v\_transaction\_type;

    open c\_commissions;

    loop

        fetch c\_commissions into v\_commissions\_amount;

        exit when c\_commissions%notfound;

            v\_total\_commission := v\_total\_commission + v\_commissions\_amount;

    end loop;

    close c\_commissions;

    return v\_total\_commission;

    exception

        when NO\_DATA\_FOUND then

            raise\_application\_error(-20001,'Inseriu c�digo que n�o existe');

        when e\_invalid then

            raise\_application\_error(-20002,'O valor inserido � inv�lido, devem ser fornecidas valores maiores que zero');

        when others then

            raise\_application\_error(-20003,'Ocorreu um erro');

end get\_commission\_by\_trans\_type;

## **PROCEDIMENTOS**

### **Deposit**

Procedimento responsável por fazer o depósito na conta bancária quando fornecida a conta bancária existente, o valor maior que zero, e credencias de acesso válidas.

create or replace procedure deposit (v\_bank\_account IN bank\_account.bank\_account\_id%type,

    v\_amount IN account\_transaction.amount%type,

    v\_username IN bank\_client.username%type,

    v\_password IN bank\_client.user\_password%type) is

    --Helper variables

    v\_bank\_account\_type bank\_account.account\_type\_id%type;

    v\_poupanca\_bank\_account bank\_account.account\_type\_id%type;

    v\_bank\_client bank\_client.bank\_client\_id%type;

    v\_poupanca\_account\_type bank\_account.account\_type\_id%type := 1;

    v\_deposit\_transaction\_type transaction\_type.transaction\_type\_id%type := 1;

    --Exceptions variables declaration

    e\_notfound exception;

    e\_invalid\_account\_type exception;

    e\_invalid\_amount exception;

    e\_invalid\_credential exception;

begin

    select account\_type\_id into v\_bank\_account\_type

        from bank\_account

        where bank\_account\_id = v\_bank\_account;

    if SQL%notfound then

        raise e\_notfound;

    elsif v\_bank\_account\_type = v\_poupanca\_account\_type then

        raise e\_invalid\_account\_type;

    elsif v\_amount <= 0 then

        raise e\_invalid\_amount;

    elsif not is\_valid\_login(v\_bank\_account, v\_username, v\_password) then

        raise e\_invalid\_credential;

    else

        begin

            select BANK\_CLIENT\_ID into v\_bank\_client

                from CLIENT\_BANK\_ACCOUNT

                where bank\_account\_id = v\_bank\_account;

            select bank\_account\_id into v\_poupanca\_bank\_account

                from client\_bank\_account

                where bank\_client\_id = v\_bank\_client and bank\_account\_id <> v\_bank\_account;

                update bank\_account

                set balance = balance + (v\_amount - get\_commission\_by\_trans\_type(v\_deposit\_transaction\_type)) \* 0.05

                where bank\_account\_id = v\_poupanca\_bank\_account;

                update bank\_account

                set balance = balance + (v\_amount - get\_commission\_by\_trans\_type(v\_deposit\_transaction\_type)) \* 0.95

                where bank\_account\_id = v\_bank\_account;

                INSERT INTO account\_transaction (

                    amount,

                    bank\_account\_id,

                    transaction\_type\_id

                ) VALUES (

                    v\_amount,

                    v\_bank\_account,

                    v\_deposit\_transaction\_type

                );

        end;

    end if;

    exception

        when e\_notfound then

            raise\_application\_error(-20001,'Inseriu uma conta banc�ria que n�o existe');

        when e\_invalid\_account\_type then

            raise\_application\_error(-20002,'A conta bancaria que inseriu � de tipo inv�lido, devem ser fornecidas contas a Ordem');

        when e\_invalid\_amount then

            raise\_application\_error(-20003,'O montante deve ser sempre superior que zero');

           when e\_invalid\_credential then

            raise\_application\_error(-20004,'Username ou Senha inv�lidos');

        when others then

            raise\_application\_error(-20005,'Ocorreu um erro durante o dep�sito');

end;

### **Withdraw**

Procedimento responsável por fazer o levantamento na conta bancária quando fornecida a conta bancária existente, o valor maior que zero e menor ou igual ao somatório do saldo da conta a ordem, conta poupança, menos a comissão e credencias de acesso válidas.

create or replace procedure withdraw (v\_bank\_account IN bank\_account.bank\_account\_id%type,

    v\_amount IN account\_transaction.amount%type,

    v\_username IN bank\_client.username%type,

    v\_password IN bank\_client.user\_password%type) is

    --Helper variables

    v\_bank\_account\_type bank\_account.account\_type\_id%type;

    v\_poupanca\_balance account\_transaction.amount%type;

    v\_ordem\_balance account\_transaction.amount%type;

    v\_tmp\_balance account\_transaction.amount%type;

    v\_poupanca\_bank\_account bank\_account.account\_type\_id%type;

    v\_bank\_client bank\_client.bank\_client\_id%type;

    v\_poupanca\_account\_type bank\_account.account\_type\_id%type := 1;

    v\_withdraw\_transaction\_type transaction\_type.transaction\_type\_id%type := 2;

    --Exceptions variables declaration

    e\_notfound exception;

    e\_invalid\_account\_type exception;

    e\_invalid\_amount exception;

    e\_not\_enough\_balance exception;

    e\_invalid\_credential exception;

begin

    select account\_type\_id into v\_bank\_account\_type

        from bank\_account

        where bank\_account\_id = v\_bank\_account;

    if SQL%notfound then

        raise e\_notfound;

    elsif v\_bank\_account\_type = v\_poupanca\_account\_type then

        raise e\_invalid\_account\_type;

    elsif v\_amount <= 0 then

        raise e\_invalid\_amount;

    elsif not is\_valid\_login(v\_bank\_account,v\_username, v\_password) then

        raise e\_invalid\_credential;

    else

        begin

            select BANK\_CLIENT\_ID into v\_bank\_client

                from CLIENT\_BANK\_ACCOUNT

                where bank\_account\_id = v\_bank\_account;

            select bank\_account\_id into v\_poupanca\_bank\_account

                from client\_bank\_account

                where bank\_client\_id = v\_bank\_client and bank\_account\_id <> v\_bank\_account;

            select balance into v\_poupanca\_balance

                from BANK\_ACCOUNT

                where bank\_account\_id = v\_poupanca\_bank\_account;

            select balance into v\_ordem\_balance

                from BANK\_ACCOUNT

                where bank\_account\_id = v\_bank\_account;

                if ((v\_ordem\_balance + v\_poupanca\_balance) - (get\_commission\_by\_trans\_type(v\_withdraw\_transaction\_type) + v\_amount)) < 0 then

                    raise e\_not\_enough\_balance;

                elsif (v\_ordem\_balance - get\_commission\_by\_trans\_type(v\_withdraw\_transaction\_type) + v\_amount) >= 0 then

                    update bank\_account

                    set balance = (balance - (v\_amount + get\_commission\_by\_trans\_type(v\_withdraw\_transaction\_type)))

                    where bank\_account\_id = v\_bank\_account;

                    INSERT INTO account\_transaction (

                        amount,

                        bank\_account\_id,

                        transaction\_type\_id

                    ) VALUES (

                        v\_amount,

                        v\_bank\_account,

                        v\_withdraw\_transaction\_type

                    );

                elsif (v\_poupanca\_balance - (get\_commission\_by\_trans\_type(v\_withdraw\_transaction\_type) + v\_amount)) >= 0 then

                    update bank\_account

                    set balance = (balance - (v\_amount + get\_commission\_by\_trans\_type(v\_withdraw\_transaction\_type)))

                    where bank\_account\_id = v\_poupanca\_bank\_account;

                    INSERT INTO account\_transaction (

                        amount,

                        bank\_account\_id,

                        transaction\_type\_id

                    ) VALUES (

                        v\_amount,

                        v\_poupanca\_bank\_account,

                        v\_withdraw\_transaction\_type

                    );

                else

                      raise e\_not\_enough\_balance;

                end if;

        end;

    end if;

    exception

        when e\_notfound then

            raise\_application\_error(-20006,'Inseriu uma conta banc�ria que n�o existe');

        when e\_invalid\_account\_type then

            raise\_application\_error(-20007,'A conta bancaria que inseriu � de tipo inv�lido, devem ser fornecidas contas a Ordem');

        when e\_invalid\_amount then

            raise\_application\_error(-20008,'O montante deve ser sempre superior que zero');

           when e\_invalid\_credential then

            raise\_application\_error(-20009,'Username ou Senha inv�lidos');

           when e\_not\_enough\_balance then

            raise\_application\_error(-20010,'Insuficiencia de Saldo');

        when others then

            raise\_application\_error(-20011,'Ocorreu um erro durante o levantamento');

end;

### **Get\_account\_statement**

Procedimento responsável por mostrar o extrato da conta bancária num intervalo de datas fornecido. A extrato será apresentado somente se fornecida uma conta bancária existente e credencias de acesso válidas.

## **TRIGGERS**

### **set\_transaction\_commission**

Trigger responsável por definer a comissação da transação, antes da inserção na tabela de Transação em contas.

create or replace trigger set\_transaction\_commission

before insert on account\_transaction

for each row

begin

    :new.commission := get\_commission\_by\_trans\_type(:new.transaction\_type\_id);

end;

### **bank\_client\_after\_insert**

Trigger responsável por criar as contas a ordem e poupança durante a inserção do cliente na tabela bank\_client.

CREATE OR REPLACE TRIGGER BANK\_CLIENT\_AFTER\_INSERT

AFTER INSERT ON BANK\_CLIENT

FOR EACH ROW

DECLARE

    V\_BANK\_ACCOUNT\_ORDEM bank\_account.bank\_account\_id%TYPE;

    V\_BANK\_ACCOUNT\_POUPANCA bank\_account.bank\_account\_id%TYPE;

    V\_BANK\_CLIENT bank\_client.bank\_client\_id%TYPE;

BEGIN

    V\_BANK\_CLIENT := :NEW.BANK\_CLIENT\_ID;

    V\_BANK\_ACCOUNT\_ORDEM := CREATE\_BANK\_ACCOUNT (2);

    V\_BANK\_ACCOUNT\_POUPANCA := CREATE\_BANK\_ACCOUNT (1);

    INSERT INTO client\_bank\_account (bank\_account\_id, bank\_client\_id)

    VALUES (V\_BANK\_ACCOUNT\_ORDEM,V\_BANK\_CLIENT);

    INSERT INTO client\_bank\_account (bank\_account\_id, bank\_client\_id)

    VALUES (V\_BANK\_ACCOUNT\_POUPANCA,V\_BANK\_CLIENT);

END;

### **Insert\_transaction\_log**

Trigger responsável por registar detalhes associados a cada transação, por exemplo durante o deposito será registado o valor total do deposito na tabela account\_transaction mas distribuição do valor nas duas contas do cliente encontramos a informação na tabela transaction\_log.

create or replace trigger insert\_transaction\_log

 after insert on  ACCOUNT\_TRANSACTION

 for each row

declare

    v\_deposit\_transaction\_type transaction\_type.transaction\_type\_id%type := 1;

    v\_poupanca\_bank\_account bank\_account.account\_type\_id%type;

    v\_bank\_client bank\_client.bank\_client\_id%type;

    v\_account\_type account\_type.account\_type\_id%type;

    v\_ordem\_account\_type account\_type.account\_type\_id%type := 2;

begin

    select account\_type\_id into v\_account\_type from bank\_account where bank\_account\_id = :new.bank\_account\_id;

    if :new.transaction\_type\_id = v\_deposit\_transaction\_type then

        if v\_account\_type = v\_ordem\_account\_type then

            select BANK\_CLIENT\_ID into v\_bank\_client

                from CLIENT\_BANK\_ACCOUNT

                where bank\_account\_id = :new.bank\_account\_id;

            select bank\_account\_id into v\_poupanca\_bank\_account

                from client\_bank\_account

                where bank\_client\_id = v\_bank\_client and bank\_account\_id <> :new.bank\_account\_id ;

                insert into transaction\_log (account\_transaction\_id, bank\_account\_id, amount)

                values (:new.account\_transaction\_id, :new.bank\_account\_id , :new.amount\*0.95);

                insert into transaction\_log (account\_transaction\_id, bank\_account\_id, amount)

                values (:new.account\_transaction\_id, v\_poupanca\_bank\_account, :new.amount \*0.05);

        else

            insert into transaction\_log (account\_transaction\_id, bank\_account\_id, amount)

                values (:new.account\_transaction\_id, :new.bank\_account\_id, :new.amount);

        end if;

    else

            insert into transaction\_log (account\_transaction\_id, bank\_account\_id, amount)

            values (:new.account\_transaction\_id, :new.bank\_account\_id , :new.amount);

    end if;

end;

/

### **Insert\_commission\_log**

Trigger responsável por registar detalhes associados as comissões associadas a cada transação, na tabela account\_transaction é registado o valor total da comissão, mas a informação sobre todos tipos de comissão que foram aplicados na transação serão encontrados na tabela commission\_log. Essa tabela ajuda também a guardar histórico sobre o valor de uma comissão numa época, se o valor da comissão mudar num determinado tempo, continuaremos a informação sobre as transações consistente.

create or replace trigger insert\_account\_transation

 after insert on  ACCOUNT\_TRANSACTION

 for each row

declare

    v\_commissions\_amount commission.commission%type;

    v\_commissions\_id commission.commission\_id%type;

    cursor c\_commissions  is

        select commission.commission\_id, commission.commission

        from transaction\_commission

        join commission on commission.commission\_id = transaction\_commission.commission\_id

        where transaction\_type\_id = :new.transaction\_type\_id;

begin

        open c\_commissions;

        loop

            fetch c\_commissions into v\_commissions\_id, v\_commissions\_amount;

            exit when c\_commissions%notfound;

            insert into commission\_log (account\_transaction\_id, commission\_id, COMMISSION\_AMOUNT)

                values (:new.account\_transaction\_id, v\_commissions\_id, v\_commissions\_amount);

        end loop;

        close c\_commissions;

 end;

/