**Universidade de Aveiro**

Departamento de Eletrónica, Telecomunicações e Informática
Engenharia Computadores e Telemática

**Base de Dados**

# Relatório Projecto:

## Aplicação para gestão de um Sistema Bancário

**Professores:**
Prof. Carlos Costa
Prof. José Luís Oliveira

**Alunos:**
Gustavo Bica, 50413
Joel Pinheiro, 65151

# ÍNDICE

# Introdução

Atualmente as base de dados constituem um repositório um repositório de armazenamento indispensável no dia a dia dos utilizadores de sistemas informáticos que as usam para persistência dos dados dos serviços que providenciam ao exterior ou internamente. No entanto, é comum o desenvolvimento de base de dados não modeladas e não implementadas da melhor forma criando problemas de desempenho, escabilidade e adaptação ao futuro.

O trabalho proposto para o projeto prático da unidade curricular de Base de Dados foi a criação de um sistema de gestão bancário minimalista não concorrente, no qual, tentou-se aplicar da melhor forma os conhecimentos adquiridos. A ideia deste trabalho surgiu em virtude da curiosidade do grupo em criar uma base de dados para um banco e perceber a complexidade a que um sistema deste género implica. Uma vez que uma base de dados para um sistema bancário é algo que envolve uma complexidade enorme decidimos criar uma réplica minimalista de um sistema bancário onde permitisse a gestão de clientes, funcionários, balcões do banco, contas e quatro operações monetárias: empréstimos, levantamentos, transferências e depósitos. Note-se que este sistema de gestão bancário é apenas apropriado para um banco de pequena dimensão.

Para além da base de dados foi desenvolvida uma aplicação em Windows Forms C# que simula o ambiente de um sistema de gestão bancária e que permite a manipulação dos dados da base de dados. Esta base de dados deve fornecer ferramentas que permitam a criação, remoção, alteração e consulta dos dados. A nível interno pretende-se que a base de dados realize essas funções de modo eficiente e robusto. Este relatório reflete os passos e as decisões tomadas na criação da base de dados que sustenta o projeto bem como uma descrição das capacidades da aplicação cliente desenvolvida em simultâneo.

Para a criação da base de dados foi seguido um processo que foi leccionado nas aulas. Este processo segue as seguintes fases: análise de requisitos, desenho conceptual, desenho do esquema lógico, desenho do esquema físico e administração. Neste relatório vamos apresentar a criação da base de dados segundo a sua concepção para facilitar a leitura e compreensão do mesmo.

# Análise de Requisitos

A análise de requisitos foi uma parte importante do processo de criação da base de dados uma vez que nos ajudou a ter uma visão mais clara do que o nosso sistema teria de suportar. Após uma troca de ideias entre os elementos do grupo achamos que os seguintes tipos e caracterização seriam importantes num sistema de gestão de base de dados.

O **cliente** é identificado por:

- número de conta;
- nome do cliente;
- balcão onde abriu a conta;
- morada;
- data de nascimento;
- gênero;
- número de telemóvel;
- estado civil;
- profissão;
- estado atual (se esta reformado ou não);
- NIF;
- código de repartição de finanças;
- país de residência fiscal;
- tipo de documento.

A **conta** é identificada por:

- número de conta;
- tipo de conta;
- balcão associado;
- saldo.

O **empréstimo** é identificado por:

- taxa de juro;
- número de conta;
- valor de empréstimo;
- tipo de empréstimo.

O **depósito** é identificado por:

- pelo número da conta;
- valor do depósito.

A **transferência** é identificada por:

- número da conta destinatária;
- número de conta origem;
- valor da transferência.

OBJ

O **levantamento** é identificado por:

- número de conta;
- montante do levantamento.

O **balcão** é identificado por:

- número do balcão;
- nome do balcão;
- morada;
- fax;
- telefone;
- horário;
- serviços.

O **funcionário** é identificado por:

- número do funcionário;
- nome do funcionário;
- balcão de trabalho;
- salário;
- horário de trabalho;
- função;
- morada;
- data de nascimento;
- gênero;
- número de telemóvel;
- NIF.

A **identificação** é constituída por:

- número de identificação;
- tipo de documento;
- data de emissão;
- país de emissão.

O **tipo de documento** deve ser:

- BI nacional ou estrangeiro;
- passaporte ou cartão de cidadão.

A **morada** é identificada por:

- código postal;
- endereço;
- localidade;
- país.

Algumas especificações foram registadas para o futuro desenho conceptual:

- Só existem dois tipos de identificação, funcionário ou cliente;

- Num balcão, podem trabalhar vários funcionários, mas cada funcionário trabalha apenas num balcão;

- Um cliente só tem um balcão origem, um balcão pode ter vários clientes;

- Cada funcionário, pode realizar várias operação, uma operação só pode ser realizada por um funcionário ou nenhum funcionário (net banking);

- Um cliente pode solitar várias operações, uma operação pode ser solicitada por vários clientes. Um cliente pode ter várias contas e uma conta pode pertencer a vários clientes;

- Uma operação pode estar associada a uma ou mais contas, uma conta pode estar associada a várias operações;

- Uma operação só pode pertencer, no máximo, a uma subclasse de especialização ,neste caso, empréstimo, levantamento, transferência ou depósito.

# Desenho Conceptual

Depois da análise de requisitos seguiu-se a criação do desenho conceptual da base de dados. Esse desenho foi descrito através de um diagrama ER (imagem 1). No diagrama, foram definidas entidades, atributos e relações entre tabelas, suas cardinalidades e dependências da relação.

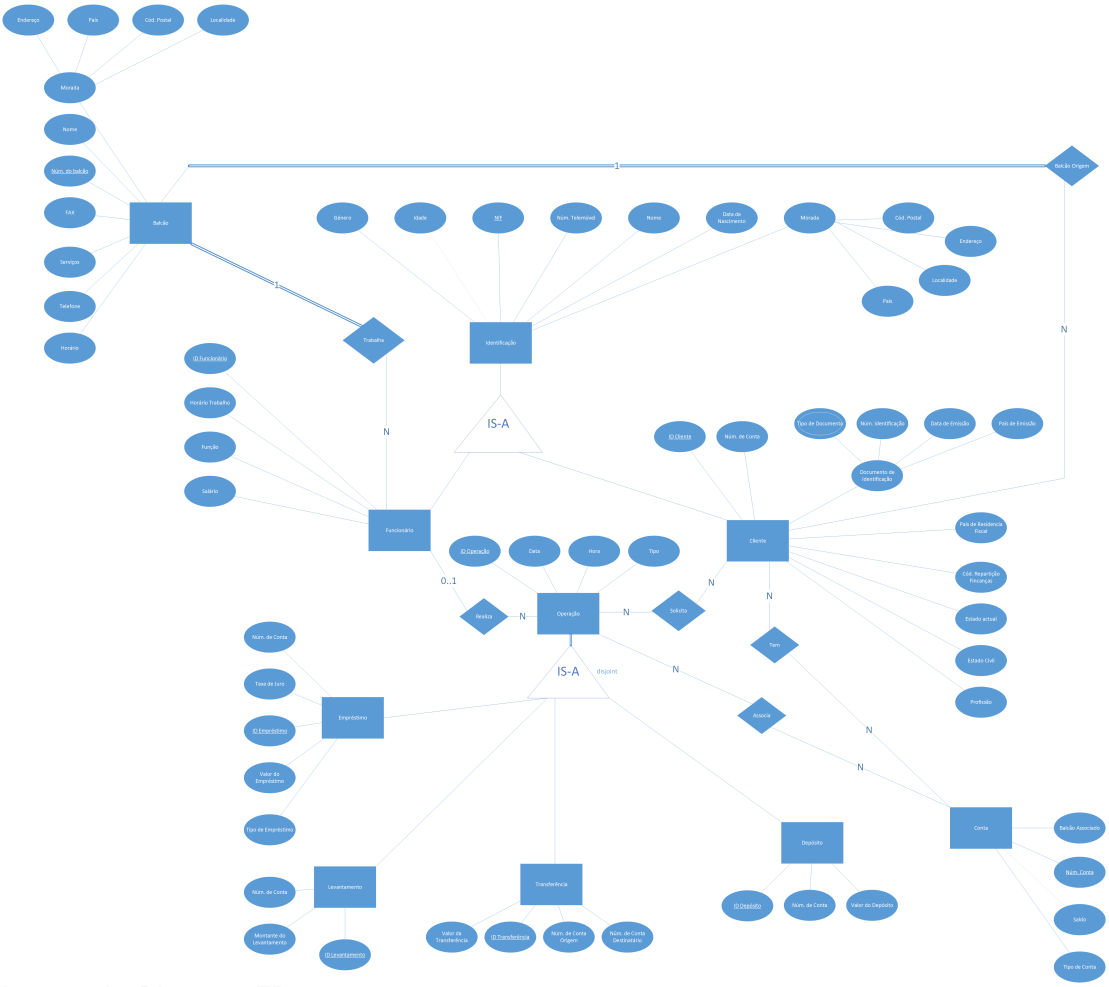



**Imagem 1 – Diagrama ER**

As entidades da base de dados e respectivos atributos são as seguintes:

**Balcão** – Entidade onde são mostrados os balcões presentes no sistema. Está associado com ao funcionário e ao cliente.

**Identificação** - Entidade onde são mostrados os dados gerais da identificação quer para funcionários e clientes.

**Funcionário** – É que realiza as operações solicitadas pelo cliente e que trabalha num balcão.

**Cliente** – Pode ter zero ou várias contas e realiza operações sobre essas contas. Está associado a um balcão (balcão de criação da conta).

**Operação** – Entidade onde são feitas as operações bancárias. Está associado a funcionários, clientes, contas, e às várias operações possíveis de se realizar (relação de herança).

**Empréstimo/Levantamento/Transferências/Depósito** – Sub-membro da entidade operação.

**Conta** – Entidade associada a um ou vários clientes sobre a qual podem ser realizadas várias operações.

No que diz respeito aos graus das relações, podemos indicar que todas as relações são binárias.

# Desenho do Esquema Físico

Após a construção do desenho conceptual, procedeu-se à elaboração do Modelo Relacional. Este modelo foi construído tendo por base o diagrama ER transformando um conjunto de tabelas. O algoritmo utilizado seguiu o processo dado nas aulas teóricas de BD. Após a transformação foi criado o Modelo Relacional (Imagem 2):

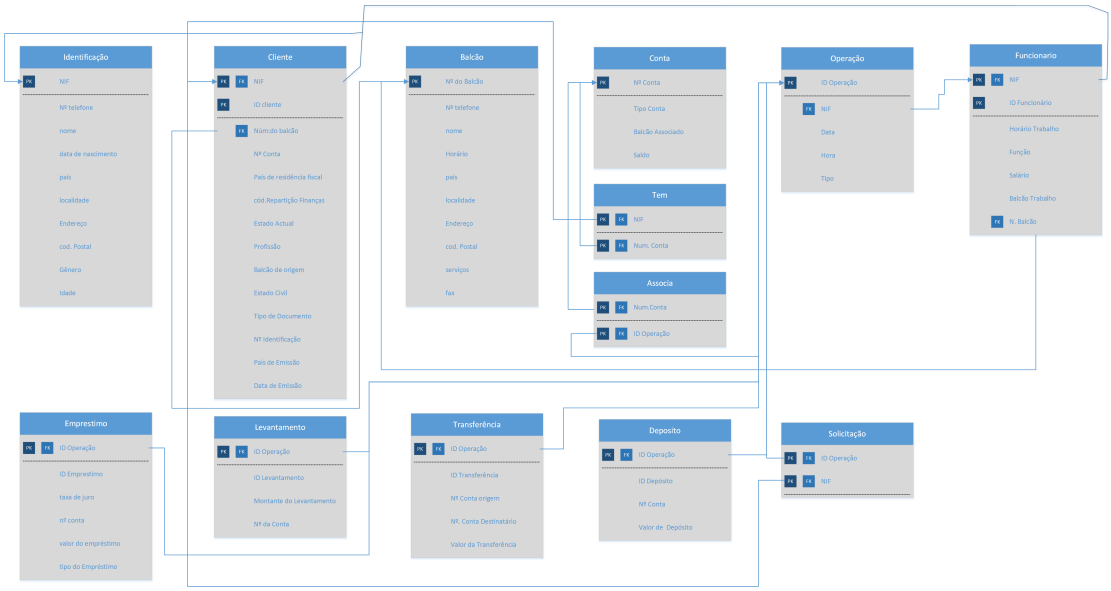



Imagem 2 – Modelo Relacional

De forma a ter um projeto mais eficiente ao eliminar redundâncias e ao permitir a integridade referencial entre relações tivemos de usar normalizações.

Para realizar a normalização do nosso projeto tivemos de ter em atenção vários conceitos:

As verificações são feitas começando pela 1FN, seguindo para a 2FN, passando para a
3FN acabando na BCFN.

Uma relação diz-se na 1FN quando:
- Não contém atributos multivalor/compostos;
- Não contém atributos repetitivos.

Uma relação diz-se na 2FN quando:
- Está na 1FN;

- Todos os atributos não chave dependem funcionalmente da totalidade da chave.

Uma relação diz-se na 3FN quando:
- Está na 2FN;
- Todos os atributos não chave não dependem funcionalmente uns dos outros, ou seja,
são funcionalmente dependentes só e apenas da chave da relação.

Uma relação diz-se na BCFN quando:
- Está na 3FN;
- Numa relação existem múltiplas chaves candidatas compostas, em que estes se
sobrepõem.

Após a analise do nosso projeto podemos concluir que já se encontra na BCFN e assim sendo podemos passar para a próxima fase. De forma a demonstrar como foi feita esta analise temos um exemplo:

| N.º Conta | Tipo Conta | Balcão Associado | Saldo |
|---|---|---|---|

Neste exemplo temos a nossa relação conta, como se pode verificar não existem atributos repetidos nem multivalores ou compostos, estando assim na 1FN.

Esta relação também se encontra na 2FN, pois esta na 1FN e todos os atributos dependem da chave primária da relação, ou seja, do **Nº conta**.

Para 3FN a relação também cumpre os requisitos, não existem dependências funcionais, ou
seja, nenhum atributo não chave pertence a outro atributo não chave.

Finalmente, para BCFN, podemos constatar mais uma vez, que
os atributos são dependentes da chave da relação Conta, de toda a chave
nº conta e nada mais. Estando assim na forma BCFN.

# Índices

À medida que o volume de dados vai aumentando o tempo de resposta aos pedidos de consulta também aumenta devido à desorganização dos dados existentes nas tabelas. Uma das formas que pode ser utilizada para combater esta tendência é manter as tabelas organizadas por forma a que as consultas sejam executadas mais rapidamente. Esta solução consiste em criar um
índice nos campos que serão mais frequentemente utilizados para fazer pesquisas criando assim um ponteiro para a posição real de cada registo.

No nosso trabalho incluímos vários índices de forma a tornar mais rápida e eficiente as nossas pesquisas. Os índices que foram utilizados neste projeto são do tipo NonCustered, já que os Clustered são automaticamente criados
na definição e construção das tabelas, pois vão ser as chaves primárias.

Exemplo do índice “indexemprestimoconta”:

```
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o número
de conta, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela emprestimo,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexemprestimonconta
  ON dbo.emprestimo(nconta)
  WITH (FILLFACTOR=75, pad_index=ON);
```

Restantes (conteúdo dos restantes em anexo):

```
indexemprestimoid

indexlevantamentonconta

indexlevantamentoid

indextransferenciaid

indextransferenciacontao

indextransferenciacontad

indexdepositonconta

indexnomeident

indexcodpostalident

indexnomebalcao

indexcodpostalbalcao
```

indexnumcontacliente

indexbalcaoorigemcliente

indextipoconta

indexsaldoconta

indexfuncaofuncionario

indexbalcaotrabalhofuncionario

# View’s

Foram criadas várias views. O motivo para o uso de view's foi o facto de se poder ter de forma simples e pratica informação processada numa tabela, tornando mais fácil o trabalho de quem as utiliza.
As view's foram concebidas para criar uma camada de abstração entre o utilizador e a base de dados. Apesar de num primeira abordagem ter-se a ideia de usar UDF's no lugar de views principalmente porque se tinha a ideia de usar UDF's nas pesquisas e por estas ultimas serem mais seguras. Mas devido a complicações que surgiram na sua correcta integração com o interface gráfico acabamos por decidir usar view's.

As views criadas foram:

**transferenciaview:**
Devolve informação sobre as transferências existentes na BD. Esta informação é o resultado da agregação entre as entidades transferência e operação.

**emprestimoview:**
Devolve informação sobre os empréstimos existentes na BD. Esta informação é o resultado da agregação entre as entidades empréstimos e operação.

**depositoview :**
Devolve informação sobre os depósito existentes na BD. Esta informação é o resultado da agregação entre as entidades transferência e operação.

**levantamentosview:**
Devolve informação sobre os levantamentos existentes na BD. Esta informação é o resultado da agregação entre as entidades levantamentos e operação.

**operacaoview:**
Devolve informação sobre todas as operações existentes na BD (depósitos, transferências, levantamentos e empréstimos), mas tendo mais atenção ao cliente que solicita a operação e ao funcionário que a realiza. Esta informação é o resultado da agregação entre as entidades operação, identificação, solicitação, cliente e funcionário.

**contaview:**
Devolve informação sobre as contas bancárias existentes na BD, como por exemplo o(s) nome(s) do(s) dono(s) de uma respectiva conta, o saldo da conta ou saldo existente na conta e por ai adiante . Esta informação é o resultado da agregação entre as entidades identificação, cliente, conta e tem.

**identificacaoview:**

Devolve informação sobre todas as identificações existentes na entidade identificação (esta view acabou por não ser usada, foi criada numa fase inicial do projecto).

**clientesviewidentificacao:**
Devolve informação sobre os clientes ,existentes na BD, de forma mais completa . Esta informação é o resultado da agregação entre as entidades cliente e identificação.

**funcionariosviewidentificacao:**
Devolve informação sobre os funcionários ,existentes na BD, de forma mais completa . Esta informação é o resultado da agregação entre as entidades funcionários e identificação.

**balcaoview:**
Devolve toda a informação sobre os balcões existentes na BD. Esta informação surge da entidade balcões.

Exemplo:

```
--Foram criadas Views para se poder visualizar em cada uma das forms
criadas para a nossa aplicação, o conjunto de
--dados pretendidos (estes dados sao adquiridos de várias entidades
diferentes existentes na nossa BD)
--Transferencia
go

CREATE VIEW transferenciaview
WITH schemabinding, encryption
-- Schemabinding serve para que os objectos referenciados pelas nossa
udf(que devolve uma view) não podem ser alvo de um drop
-- Encrypton para evitar que a view criada possa ser vista por qualquer
utilizador usando sp_helptext ou syscomments
AS
  SELECT operacao.idoperacao AS "ID Operação",
         --Seleção dos campos pretendidos para a nossa view e como os
queremos visiveis na nossa aplicação (por exemplo: ID Operação em vez
de idOperacao)
         idtranferencia      AS "ID Transferência",
         ncontaorigem        AS "Número Conta Origem",
         ncontadestinatario  AS "Número Conta Destinatário",
         valortransferencia  AS "Valor Transferência",
         dbo.operacao.nif    AS "NIF Funcionario",
         dbo.operacao.tipo   AS "Tipo",
         dbo.operacao.data   AS "Data",
         dbo.operacao.hora   AS "Hora"
  FROM   dbo.transferencia,
         dbo.operacao
  WHERE  operacao.idoperacao = transferencia.idoperacao; --É utilizado
o where para garantir que na view é visualizado apenas dados associados
entre si
```

# Transactions

Embora o projecto se trate de um sistema bancário não concorrente, as operações envolvidas (transferências, levantamentos, empréstimos, depósitos) vão actualizar as contas do sistema. Assim, tivemos de garantir que as store procedures responsáveis por fazer inserções de operações bancárias na BD sejam feitas de um modo atómico para garantir que em caso de falha a BD continue consistente. Com isto, queremos dizer que as instruções de insert da operação e update de saldo são concluídas em conjunto. Ou funcionam todas as instruções de uma vez ou não é feita nenhuma (rollback). Este mecanismo assegura-nos o saldo das contas está certo desde que a operação seja inserida com sucesso.

Podemos tomar como exemplo a seguinte store procedure com transaction:

```
create PROCEDURE [dbo].[Spinsertlevantamento] @idOperacao
INT,
                                                --Paramentros de entrada
necessários para a criação de um levantamento
                                                @idLevantamento       INT,
                                                @montanteLevantamento
MONEY,
                                                @nConta               INT
AS
    IF( EXISTS(SELECT *
                FROM   conta
                WHERE  nconta = @nConta
                       AND saldo < @montanteLevantamento) )
      -- Condição de forma a garantir que é possivel realizar a
operação pedida
      BEGIN
          RETURN 'Não tem saldo suficiente na conta!' --Mensagem de
erro
      END

  BEGIN TRANSACTION
        INSERT INTO dbo.levantamento -- Começa a transation
                  --Sendo permitida a operação, definimos onde vamos
colocar os dados recebidos como parametros
                  ([idoperacao],
                   [idlevantamento],
                   [montantelevantamento],
                   [nconta])
      VALUES      (@idOperacao,
                   @idLevantamento,
                   @montanteLevantamento,
                   @nConta)

      UPDATE conta
```

```
      SET    saldo = saldo - @montanteLevantamento
      WHERE  nconta = @nConta -- Update do saldo na conta afectada pela
operação

        COMMIT TRAN -- transation é bem sucedida
        BEGIN              -- transation não foi bem sucedida, é feito
o rollback
            PRINT 'A transacção precisa de rollback'
                  ROLLBACK     TRAN
        END
```

# Store Procedures (SP)

Devido à robustez que estas estruturas oferecem, decidiu-se apenas permitir o acesso à base de dados(Inserts, Updates e Deletes) a partir de Storage Procedures(SP),criando com elas uma camada de abstração entre o utilizador e a base de dados. A lógica dos vários SP é bastante semelhante, os únicos que necessitaram de uma lógica mais "complexa" foram os relacionados com operações, pois é necessário realizar várias verificações (se determinada conta existe e se tem saldo suficiente para realizar a operação pretendida) e ao realizar um insert para além das verificações é tambem preciso realizar um update às contas envolvidas de forma a ter os saldos das contas actualizados.

Os Store Procedures criados para inserts e deletes correspondem a cada uma das entidades da nossa base de dados (tem, associa, solicitacao, transferencia, levantamento, deposito, emprestimo, balcao, cliente, funcionario e conta). Decidimos não permitir updates nas operações, pois, no ponto de vista de um banco, a partir do momento que é realizado uma dada operação, essa operação deixa de ser possível de alterar, sendo assim apenas é possível criar ou eliminar uma operação.

Exemplo:

```
--Paramentros de entrada necessários para a criação de um levantamento
                                        @idLevantamento       INT,
                                        @montanteLevantamento
MONEY,
                                        @nConta               INT
AS
    IF( EXISTS(SELECT *
               FROM   conta
               WHERE  nconta = @nConta
                      AND saldo < @montanteLevantamento) )
      -- Condição de forma a garantir que é possivel realizar a
operação pedida
      BEGIN
          RETURN 'Não tem saldo suficiente na conta!' --Mensagem de
erro
      END

  BEGIN TRANSACTION
        INSERT INTO dbo.levantamento
                  --Sendo permitida a operação, definimos onde vamos
colocar os dados recebidos como parametros
                  ([idoperacao],
                   [idlevantamento],
                   [montantelevantamento],
                   [nconta])
      VALUES      (@idOperacao,
                   @idLevantamento,
                   @montanteLevantamento,
```

```
                @nConta)

      UPDATE conta
      SET    saldo = saldo - @montanteLevantamento
      WHERE  nconta = @nConta -- Update do saldo na conta afectada pela
operação

        COMMIT TRAN
        BEGIN
            PRINT 'A transacção precisa de rollback'
                  ROLLBACK     TRAN
        END
```

# User-Defined Funcions (UDF’s)

As UDF’s para nós foram vistas como uma boa alternativa às views, principalmente no ponto de vista de pesquisa, pois não só são mais seguras do que as views como também permitem associar lógica, podendo assim criar uma base de dados mais robusta. Apesar de ter sido essa a nossa intenção, já na fase de implementação da interfase, deparamo-nos com problemas, nomeadamente em conseguir executar correctamente e visualizar as tabelas devolvidas pelas UDF's. Uma vez que, já não tínhamos muito tempo decidimos usar as views criadas anteriormente.

A lógica associada às UDF's consiste em ter sempre 3 parâmetros de entrada, que podem receber algum valor ou então "ficar" com o valor null ,consoante os parâmetros recebidos era realizada a pesquisa que se pretendia realizar, mas sempre tendo em conta que estes parâmetros devem ser constituídos pelas chaves primarias e chaves candidatas, de forma a tornar a pesquisa mais eficiente, isto porque, não só são criados automaticamente índices para as chaves primarias, mas também porque criamos vários indicies para as tabelas mais relevantes do ponto de vista de uma pesquisa . Após se ter os parâmetros de entrada era uma questão de se ver quais tinham sido realmente preenchidos e consoante isso era feita uma pesquisa na Base de dados de modo a retornar, as consulta para o filtro dado.

Exemplo:

```
-- AS UDF foram criadas para a pesquisa por campos da nossa aplicação,
foram criadas UDF's para cada uma das operações visiveis para o
utilizador e para a entidade cliente, apesar de no backend ser
necessário recorrer a mais entidades.
--Cada UDF recebe como parametro 3 campos, estes campos foram
escolhidos por serem os mais unicos para a respectivel pesquisa.O
objectivo destas UDF era tambem permitir a pesquisa usando qualuqer uma
das possiveis combinaições (usar apenas 2 campos para pesquisar,usar os
3, usar apenas um, etc)
--Deposito
go

CREATE FUNCTION dbo.Udfdeposito(@idOperacao INT=0,
                                @idDeposito INT=0,
                                @nconta     INT=0) --Parametros de
entrada
returns @table TABLE (
  "nome"         VARCHAR(200),
  "nif"          INT,
  "id operação"  INT,
  "id deposito"  INT,
  "número conta" INT )
--Maneira como queremos que fique escrito na nossa base de dados os
dados que iremos apresentar
```

```
WITH schemabinding, encryption
  -- Schemabinding serve para que os objectos referenciados pelas nossa
udf(que devolve uma view) não podem ser alvo de um drop
  BEGIN
      -- Encrypton para evitar que a view criada possa ser vista por
qualquer utilizador usando sp_helptext ou syscomments
      --Declaração dos varios if para cada um dos cenarios de pesquisa
      --quando necessario atraves do where é definido as condiçoes que
devem ser cumpridas para aparecerem apenas os dados pretendidos
      IF( @idOperacao = NULL
          AND @nconta = NULL
          AND @idDeposito = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            idoperacao             AS "ID Operação",
                            iddeposito             AS "ID Deposito",
                            nconta                 AS "número conta"
            FROM   dbo.identificacao,
                   dbo.deposito
        END

      IF( @idOperacao >= 0
          AND @nconta >= 0
          AND @idDeposito >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome       AS "Nome",
                            nif        AS "NIF",
                            idoperacao AS "ID Operação",
                            iddeposito AS "ID Deposito",
                            nconta     AS "número conta"
            FROM   dbo.identificacao,
                   dbo.deposito
            WHERE  nconta = @nconta
                   AND iddeposito = @idDeposito
                   AND idoperacao = @idOperacao
        END

      IF( @idOperacao = NULL
          AND @nconta >= 0
          AND @idDeposito >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome       AS "Nome",
                            nif        AS "NIF",
                            idoperacao AS "ID Operação",
                            iddeposito AS "ID Deposito",
                            nconta     AS "número conta"
            FROM   dbo.identificacao,
                   dbo.deposito
            WHERE  nconta = @nconta
                   AND iddeposito = @idDeposito
        END
```

```
IF( @idOperacao = NULL
    AND @nconta >= 0
    AND @idDeposito = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome       AS "Nome",
                      nif        AS "NIF",
                      idoperacao AS "ID Operação",
                      iddeposito AS "ID Deposito",
                      nconta     AS "número conta"
      FROM   dbo.identificacao,
             dbo.deposito
      WHERE  nconta = @nconta
  END

IF( @idOperacao = NULL
    AND @nconta = NULL
    AND @idDeposito >= 0 )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome       AS "Nome",
                      nif        AS "NIF",
                      idoperacao AS "ID Operação",
                      iddeposito AS "ID Deposito",
                      nconta     AS "número conta"
      FROM   dbo.identificacao,
             dbo.deposito
      WHERE  iddeposito = @idDeposito
  END

IF( @idOperacao >= 0
    AND @nconta = NULL
    AND @idDeposito = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome       AS "Nome",
                      nif        AS "NIF",
                      idoperacao AS "ID Operação",
                      iddeposito AS "ID Deposito",
                      nconta     AS "número conta"
      FROM   dbo.identificacao,
             dbo.deposito
      WHERE  idoperacao = @idOperacao
  END

IF( @idOperacao >= 0
    AND @nconta >= 0
    AND @idDeposito = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome       AS "Nome",
                      nif        AS "NIF",
                      idoperacao AS "ID Operação",
                      iddeposito AS "ID Deposito",
                      nconta     AS "número conta"
      FROM   dbo.identificacao,
```

```
                dbo.deposito
            WHERE  nconta = @nconta
                AND idoperacao = @idOperacao
        END

    IF( @idOperacao >= 0
        AND @nconta = NULL
        AND @idDeposito >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome       AS "Nome",
                            nif        AS "NIF",
                            idoperacao AS "ID Operação",
                            iddeposito AS "ID Deposito",
                            nconta     AS "número conta"
            FROM   dbo.identificacao,
                   dbo.deposito
            WHERE  idoperacao = @idOperacao
                   AND iddeposito = @idDeposito
        END

    RETURN
END;
```

# Triggers

Os triggers foram usados com abundância no projecto para garantir uma maior consistência e integridade da base de dados. Um dos triggers tem com objectivo actualizar os clientes sempre que o seu balcão de origem é eliminado passado assim o cliente a não ter balcão origem. Os restantes tem como objectivo garantir a consistência da BD.

Por exemplo, no caso de ser removido um empréstimo a operação associada também é:

```
-- No caso de um emprestimo ser removido, remover também a operação
associada
go

CREATE TRIGGER triggerdeleteemprestimo
ON dbo.emprestimo
after DELETE
AS
  BEGIN
      DELETE dbo.operacao
      FROM   dbo.operacao,
             dbo.emprestimo
      WHERE  dbo.operacao.idoperacao = dbo.emprestimo.idoperacao

      DELETE dbo.associa
      FROM   dbo.operacao,
             dbo.emprestimo,
             dbo.associa
      WHERE  dbo.operacao.idoperacao = dbo.emprestimo.idoperacao
             AND dbo.associa.idoperacao = dbo.operacao.idoperacao

      DELETE dbo.solicitao
      FROM   dbo.operacao,
             dbo.emprestimo,
             dbo.solicitao
      WHERE  dbo.operacao.idoperacao = dbo.emprestimo.idoperacao
             AND dbo.solicitao.idoperacao = dbo.operacao.idoperacao
  END;
```

# Cursores

Numa primeira abordagem decidiu-se descartar os cursores para o nosso projeto. O motivo deveu-se ao facto de desde o início se ter optado por uma abordagem Set-Based SQL queries. Após uma análise mais cuidada decidiu-se que o utilizador da aplicação pode querer fazer uma pesquisa diferente daquelas que tínhamos. Assim, optou-se pelo o uso de cursores na BD para a pesquisa de qualquer informação que exista na tabela sem específicar qual coluna a pesquisa seria feita. Com o cursor a pesquisa é feita utilizando um ciclo while que percorre todas as colunas da tabela à procura do argumento dado. O uso desta ferramenta de pequisa é de evitar uma vez que o tempo de pesquisa em muito maior que o numa pesquisa Set-Based SQL query.

O cursor feito foi o seguinte:

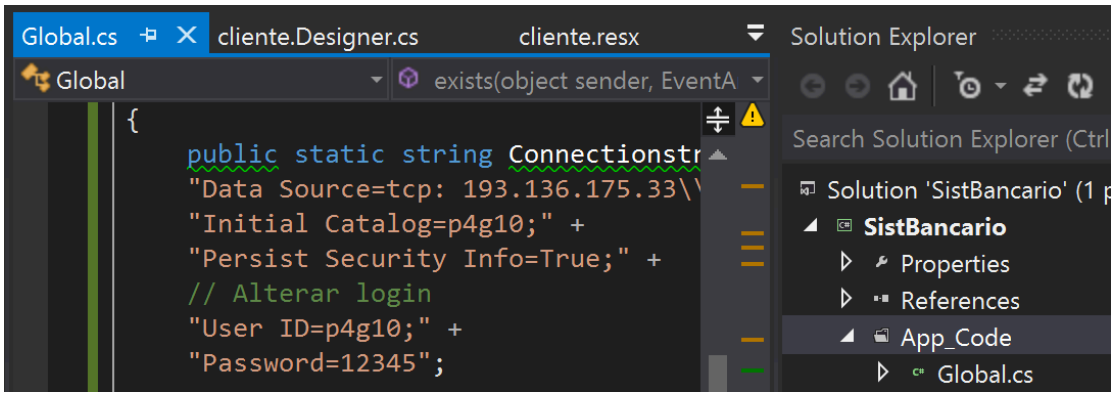
```
EXEC Sp_findstringintable
  'tuga',
  dbo,
  balcao

CREATE PROCEDURE Sp_findstringintable @stringToFind VARCHAR(100),
                                      @schema       SYSNAME,
                                      @table        SYSNAME
AS
    DECLARE @sqlCommand VARCHAR(8000)
    DECLARE @where VARCHAR(8000)
    DECLARE @columnName SYSNAME
    DECLARE @cursor VARCHAR(8000)

  BEGIN try
      SET @sqlCommand = 'SELECT * FROM [' + @schema + '].[' + @table
                        + '] WHERE'
      SET @where = ''
      SET @cursor = 'DECLARE col_cursor CURSOR FOR SELECT COLUMN_NAME
FROM ' + Db_name()
                    + '.INFORMATION_SCHEMA.COLUMNS     WHERE
TABLE_SCHEMA like ''' + @schema
                    + '''     AND TABLE_NAME like ''%' + @table
                    +
'%''     AND DATA_TYPE IN
(''char'',''nchar'',''ntext'',''nvarchar'',''text'',''varchar'')'

    EXEC (@cursor)

    OPEN col_cursor

    FETCH next FROM col_cursor INTO @columnName

    WHILE @@FETCH_STATUS = 0
      BEGIN
          IF @where <> ''
            SET @where = @where + ' OR'
```

```
            SET @where = @where + ' [' + @columnName + '] LIKE ''%'
                         + @stringToFind + '%'''

            FETCH next FROM col_cursor INTO @columnName
        END

    CLOSE col_cursor

    DEALLOCATE col_cursor

    SET @sqlCommand = @sqlCommand + @where

    --PRINT @sqlCommand
    EXEC (@sqlCommand)
END try

  BEGIN catch
      PRINT 'Houve um erro. Verifique se o objecto existe!'

      IF Cursor_status('variable', 'col_cursor') <> -3
        BEGIN
            CLOSE col_cursor

            DEALLOCATE col_cursor
        END
  END catch
```

# Aplicação Cliente

A aplicação cliente para este projecto - Sistema de Gestão Bancário - consiste numa aplicação desenvolvida em Windows Forms (C#) feita exclusivamente para esta disciplica currícular. Esta aplicação, permite a manipulação da BD com um interface gráfico simples e claro para o utilizador. Possibilita a inserção, actualização e remoção das entidades da BD com excepção das operações bancárias que não é permitido a atualização destas (tal como acontece numa aplicação bancária real). A aplicação foi desenvolvida com o Visual Studio 2012 e testada no Windows 7 e 8.

Para usar a aplicação dever alterar a **ConnectionString**. Para isso deve ir à parta do projecto e em App_Code/global.cs alterar a string da ligação à BD:



# Conclusão

Os objectivos foram alcançados e que com este trabalho foi possível abordar todos os conceitos leccionados na disciplina de Base de Dados e assim consolidar todo o conhecimento adquirido ao longo das aulas teóricas e práticas o que sem este trabalho não seria possível. Durante o desenvolvimento da base de dados foram surgindo alguns erros/problemas que tiveram que ser corrigidos voltando a uma fase anterior da construção da base de dados.

A fase inicial exigiu muita atenção uma vez que esta fase teria consequências imediatas na elaboração das fases seguintes. Depois da primeira fase o processo evolutivo foi fluindo bem com mais ou menos obstáculos mas todos eles ultrapassados num relativo curto espaço de tempo. A aplicação cliente deu algum trabalho mas com algum esforço foi feita uma aplicação que permitisse o uso de todos os conceitos que fomos utilizando na elaboração da base de dados. Para concluir, o trabalho foi muito útil para praticar os conceitos aprendidos ao longo de todo o semestre.

# Referências Bibliográficas


- www.**stackoverflow**.com
- www.**w3schools**.com
- www.**msdn**.microsoft.com
- www.**learnentityframework**.com/
- Slides das aulas teóricas do prof. Carlos Costa responsável pela UC de Base de Dados

# Anexos

## Scripts dos Índices

```
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o número
de conta, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela emprestimo,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexemprestimonconta
  ON dbo.emprestimo(nconta)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o ID do
empréstimo, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela emprestimo,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
CREATE NONCLUSTERED INDEX indexemprestimoid
  ON dbo.emprestimo(idemprestimo)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o número
de conta, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela levantamento,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexlevantamentonconta
  ON dbo.levantamento(nconta)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o ID do
levantamento, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela levantamento,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
CREATE NONCLUSTERED INDEX indexlevantamentoid
  ON dbo.levantamento(idlevantamento)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o ID da
transferência, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela transferência,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
```

```
go

CREATE NONCLUSTERED INDEX indextransferenciaid
  ON dbo.transferencia(idtranferencia)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o número
de conta origem, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela transferência,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
CREATE NONCLUSTERED INDEX indextransferenciacontao
  ON dbo.transferencia(ncontaorigem)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o número
de conta destinatário, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela transferência,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
CREATE NONCLUSTERED INDEX indextransferenciacontad
  ON dbo.transferencia(ncontadestinatario)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o número
de conta, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela depósito, este
tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexdepositonconta
  ON dbo.deposito(nconta)
  WITH (FILLFACTOR=75, pad_index=ON);

---------- IDENTIFICACAO ----------------
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por nome, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexnomeident
  ON dbo.identificacao (nome)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por CodPostal, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
```

```
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexcodpostalident
  ON dbo.identificacao (codpostal)
  WITH (FILLFACTOR=75, pad_index=ON);

---------- BALCAO ----------------
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por nome do balcão, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexnomebalcao
  ON dbo.balcao (nome)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por nome do balcão, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexcodpostalbalcao
  ON dbo.balcao (codpostal)
  WITH (FILLFACTOR=75, pad_index=ON);

---------- CLIENTE ----------------
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por número de conta, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexnumcontacliente
  ON dbo.cliente (nconta)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por balcão origem, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexbalcaoorigemcliente
  ON dbo.cliente (balcaoorigem)
  WITH (FILLFACTOR=75, pad_index=ON);
```

```
---------- CONTA ----------------
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por tipo de conta, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indextipoconta
  ON dbo.conta (tipoconta)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
saldo, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexsaldoconta
  ON dbo.conta (saldo)
  WITH (FILLFACTOR=75, pad_index=ON);

---------- FUNCIONARIO ----------------
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por função, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexfuncaofuncionario
  ON dbo.funcionario (funcao)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a balcão
de trabalho, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexbalcaotrabalhofuncionario
  ON dbo.funcionario (balcaotrabalho)
  WITH (FILLFACTOR=75, pad_index=ON);
```

Scripts das Views

```
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o número
de conta, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela emprestimo,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexemprestimonconta
  ON dbo.emprestimo(nconta)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o ID do
empréstimo, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela emprestimo,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
CREATE NONCLUSTERED INDEX indexemprestimoid
  ON dbo.emprestimo(idemprestimo)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o número
de conta, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela levantamento,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexlevantamentonconta
  ON dbo.levantamento(nconta)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o ID do
levantamento, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela levantamento,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
CREATE NONCLUSTERED INDEX indexlevantamentoid
  ON dbo.levantamento(idlevantamento)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o ID da
transferência, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela transferência,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go
```

```
CREATE NONCLUSTERED INDEX indextransferenciaid
  ON dbo.transferencia(idtranferencia)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o número
de conta origem, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela transferência,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
CREATE NONCLUSTERED INDEX indextransferenciacontao
  ON dbo.transferencia(ncontaorigem)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o número
de conta destinatário, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela transferência,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
CREATE NONCLUSTERED INDEX indextransferenciacontad
  ON dbo.transferencia(ncontadestinatario)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE o número
de conta, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela depósito, este
tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexdepositonconta
  ON dbo.deposito(nconta)
  WITH (FILLFACTOR=75, pad_index=ON);

---------- IDENTIFICACAO ----------------
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por nome, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexnomeident
  ON dbo.identificacao (nome)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por CodPostal, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go
```

```
CREATE NONCLUSTERED INDEX indexcodpostalident
  ON dbo.identificacao (codpostal)
  WITH (FILLFACTOR=75, pad_index=ON);

---------- BALCAO ----------------
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por nome do balcão, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexnomebalcao
  ON dbo.balcao (nome)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por nome do balcão, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexcodpostalbalcao
  ON dbo.balcao (codpostal)
  WITH (FILLFACTOR=75, pad_index=ON);

---------- CLIENTE ----------------
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por número de conta, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexnumcontacliente
  ON dbo.cliente (nconta)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por balcão origem, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexbalcaoorigemcliente
  ON dbo.cliente (balcaoorigem)
  WITH (FILLFACTOR=75, pad_index=ON);

---------- CONTA ----------------
```

```
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por tipo de conta, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indextipoconta
  ON dbo.conta (tipoconta)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
saldo, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexsaldoconta
  ON dbo.conta (saldo)
  WITH (FILLFACTOR=75, pad_index=ON);

---------- FUNCIONARIO ----------------
-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a procura
por função, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexfuncaofuncionario
  ON dbo.funcionario (funcao)
  WITH (FILLFACTOR=75, pad_index=ON);

-- Uma vez que é frequentemente utilizado nas cláusulas WHERE a balcão
de trabalho, este atributo deve ser um índice
-- Como já existe um índice do tipo CLUSTERED na tabela identificacao,
este tem obrigatóriamente de ser do tipo NONCLUSTER
-- Porque não tem ordem natural e pode ser inserido em qualquer parte
da b-tree, serão inseridos os filtos FILLFACTOR e PAD_INDEX
go

CREATE NONCLUSTERED INDEX indexbalcaotrabalhofuncionario
  ON dbo.funcionario (balcaotrabalho)
  WITH (FILLFACTOR=75, pad_index=ON);
```

Script das Transactions

```
--De forma a actualizar os saldos das contas após uma dada operação
foram usados store procedures que para além de criarem a operação
propriamente dita, verificam se é possivel
--realizar a operação. Caso nao seja nao criam a operação e emitem uma
mensagem de erro. Caso seja possivel, criam a operação pretendida e
actualizam os saldo(s) da(s) conta(s) envolvida(s).
-- Foram usadas transactions para permitir que a actualização da BD
seja feita com maior segurança
--Levantamento
USE [p4g10]

go

SET ansi_nulls ON

go

SET quoted_identifier ON

go

create PROCEDURE [dbo].[Spinsertlevantamento] @idOperacao
INT,
                                                --Paramentros de entrada
necessários para a criação de um levantamento
                                                @idLevantamento       INT,
                                                @montanteLevantamento
MONEY,
                                                @nConta               INT
AS
    IF( EXISTS(SELECT *
               FROM   conta
               WHERE  nconta = @nConta
                      AND saldo < @montanteLevantamento) )
      -- Condição de forma a garantir que é possivel realizar a
operação pedida
      BEGIN
          RETURN 'Não tem saldo suficiente na conta!' --Mensagem de
erro
      END

  BEGIN TRANSACTION
        INSERT INTO dbo.levantamento -- Começa a transation
                  --Sendo permitida a operação, definimos onde vamos
colocar os dados recebidos como parametros
                  ([idoperacao],
                   [idlevantamento],
                   [montantelevantamento],
                   [nconta])
      VALUES      (@idOperacao,
                   @idLevantamento,
```

```
                @montanteLevantamento,
                @nConta)

      UPDATE conta
      SET    saldo = saldo - @montanteLevantamento
      WHERE  nconta = @nConta -- Update do saldo na conta afectada pela
operação

        COMMIT TRAN -- transation é bem sucedida
        BEGIN              -- transation não foi bem sucedida, é feito
o rollback
            PRINT 'A transacção precisa de rollback'
                  ROLLBACK     TRAN
        END


--Deposito
go

create PROCEDURE dbo.Spinsertdeposito @idOperacao    INT,
                                      @idDeposito    INT,
                                      @nconta        INT,
                                      @valorDeposito MONEY
AS
    BEGIN TRANSACTION
      INSERT INTO dbo.deposito
                ([idoperacao],
                 [iddeposito],
                 [nconta],
                 [valordeposito])
    VALUES      ( @idOperacao,
                  @idDeposito,
                  @nconta,
                  @valorDeposito)

    UPDATE conta
    SET    saldo = saldo + @valorDeposito
    WHERE  nconta = @nConta;

      COMMIT TRAN
        BEGIN
            PRINT 'A transacção precisa de rollback'
                  ROLLBACK     TRAN
        END


--TRANSFERENCIA

GO
create PROCEDURE dbo.Spinserttransferencia @idOperacao         INT,
                                           @idTranferencia     INT,
                                           @nContaOrigem       INT,
                                           @nContaDestinatario INT,
                                           @valorTransferencia MONEY
AS
    IF( EXISTS(SELECT *
```

```
                FROM   conta
                WHERE  nconta = @nContaOrigem
                       AND saldo < @valorTransferencia) )
      BEGIN
          RETURN 'Não tem saldo suficiente na conta!'
      END

      BEGIN TRANSACTION
    INSERT INTO dbo.transferencia
                ([idoperacao],
                 [idtranferencia],
                 [ncontaorigem],
                 [ncontadestinatario],
                 [valortransferencia])
    VALUES      ( @idOperacao,
                  @idTranferencia,
                  @nContaOrigem,
                  @nContaDestinatario,
                  @valorTransferencia)

    UPDATE conta
    SET    saldo = saldo + @valorTransferencia
    WHERE  nconta = @nContaDestinatario

    UPDATE conta
    SET    saldo = saldo - @valorTransferencia
    WHERE  nconta = @nContaOrigem;

      COMMIT TRAN
        BEGIN
            PRINT 'A transacção precisa de rollback'
                  ROLLBACK     TRAN
        END
        A

--Emprestimo
go

create PROCEDURE dbo.Spinsertemprestimo @idOperacao      INT,
                                        @idEmprestimo    INT,
                                        @taxaJuro        INT,
                                        @nconta          INT,
                                        @valorEmprestimo MONEY,
                                        @tipoEmprestimo  VARCHAR(100)
AS
    BEGIN TRANSACTION
      INSERT INTO dbo.emprestimo
                ([idoperacao],
                 [idemprestimo],
                 [taxajuro],
                 [nconta],
                 [valoremprestimo],
                 [tipoemprestimo])
    VALUES      ( @idOperacao,
                  @idEmprestimo,
                  @taxaJuro,
```

```
                @nconta,
                @valorEmprestimo,
                @tipoEmprestimo)

UPDATE conta
SET    saldo = saldo + @valorEmprestimo
WHERE  nconta = @nConta;

  COMMIT TRAN
    BEGIN
        PRINT 'A transacção precisa de rollback'
              ROLLBACK     TRAN
    END
```

Script das Store-Procedures

```
--Neste script foram criadas as stores procedures tanto para a
inserção,update e delete das várias entidades existentens na nossa base
de dados
SET ansi_nulls ON

go

SET quoted_identifier ON

go

--Associa
CREATE PROCEDURE dbo.Spinsertfuncionario @NIF            INT,
                                         --Paramentros de entrada para
a criação de um funcionário, todos estes paramentros
                                         @idfuncionario  INT,
                                         --são obrigatorios (not null)
para ser posivel criar um novo funcionário
                                         @horarioT       VARCHAR(200),
                                         @funcao         VARCHAR (200),
                                         @salario        MONEY,
                                         @balcaoTrabalho VARCHAR(100),
                                         @nbalcao        INT
AS
  BEGIN
      SET nocount ON

      INSERT INTO dbo.funcionario
                  --Indicação de onde se coloca os paramentros de
entrada
                  (nif,
                   idfuncionario,
                   horariot,
                   funcao,
                   salario,
                   balcaotrabalho,
                   nbalcao)
      VALUES      ( @NIF,
                    @idfuncionario,
                    @horarioT,
                    @funcao,
                    @salario,
                    @balcaoTrabalho,
                    @nbalcao)
  END

SET ansi_nulls ON

--Update Funcionario
go

CREATE PROCEDURE [dbo].[Spupdatefuncionario] @NIF            INT,
                                             @idFuncionario  INT,
```

```
                                                @horarioT
VARCHAR(200),
                                                @funcao
VARCHAR(200),
                                                @salario        MONEY,
                                                @balcaoTrabalho
VARCHAR(100),
                                                @nbalcao        INT
AS
    UPDATE [dbo].[funcionario]
    SET    nif = @NIF,
           idfuncionario = @idFuncionario,
           horariot = @horarioT,
           funcao = @funcao,
           salario = @salario,
           balcaotrabalho = @balcaoTrabalho,
           nbalcao = @nbalcao
    WHERE  nif = @NIF

--Associa
go

CREATE PROCEDURE dbo.Spinsertassocia @nConta     INT,
                                     @idOperacao INT
AS
    INSERT INTO dbo.associa
                ([nconta],
                 [idoperacao])
    VALUES      (@nConta,
                 @idOperacao)

--Tem
go

CREATE PROCEDURE dbo.Spinserttem @nConta INT,
                                 @NIF    INT
AS
    INSERT INTO dbo.tem
                ([nconta],
                 [nif])
    VALUES      (@nConta,
                 @NIF)

--SOLICITACAO
go

CREATE PROCEDURE dbo.Spinsertsolicitacao @idOperacao INT,
                                         @NIF        INT
AS
    INSERT INTO dbo.solicitacao
                ([idoperacao],
                 [nif])
    VALUES      (@idOperacao,
                 @NIF )

--OPERACAO
```

```
go

CREATE PROCEDURE dbo.Spinsertoperacao @idOperacao INT,
                                      @tipo       VARCHAR(40),
                                      @data       DATE,
                                      @hora       TIME(7),
                                      @NIF        INT
AS
    INSERT INTO dbo.operacao
                ([idoperacao],
                 [tipo],
                 [data],
                 [hora],
                 [nif])
    VALUES      (@idOperacao,
                 @tipo,
                 @data,
                 @hora,
                 @NIF )

--DEPOSITO
go

CREATE PROCEDURE dbo.Spinsertdeposito @idOperacao    INT,
                                      @idDeposito    INT,
                                      @nconta        INT,
                                      @valorDeposito MONEY
AS
    INSERT INTO dbo.deposito
                ([idoperacao],
                 [iddeposito],
                 [nconta],
                 [valordeposito])
    VALUES      ( @idOperacao,
                  @idDeposito,
                  @nconta,
                  @valorDeposito)

--TRANSFERENCIA
go

CREATE PROCEDURE dbo.Spinserttransferencia @idOperacao         INT,
                                           @idTranferencia     INT,
                                           @nContaOrigem       INT,
                                           @nContaDestinatario INT,
                                           @valorTransferencia MONEY
AS
    INSERT INTO dbo.transferencia
                ([idoperacao],
                 [idtranferencia],
                 [ncontaorigem],
                 [ncontadestinatario],
                 [valortransferencia])
    VALUES      ( @idOperacao,
                  @idTranferencia,
                  @nContaOrigem,
```

```
                @nContaDestinatario,
                @valorTransferencia)

--Conta
go

CREATE PROCEDURE dbo.Spinsertconta @nConta          INT,
                                   @tipoconta       VARCHAR(100),
                                   @balcaoAssociado VARCHAR(100),
                                   @saldo           MONEY
AS
    INSERT INTO dbo.conta
                ([nconta],
                 [tipoconta],
                 [balcaoassociado],
                 [saldo])
    VALUES      (@nConta,
                 @tipoconta,
                 @balcaoAssociado,
                 @saldo)

go

ALTER PROCEDURE dbo.Spupdateconta @nConta          INT,
                                  @balcaoAssociado VARCHAR(100)
AS
    UPDATE dbo.conta
    SET    balcaoassociado = @balcaoAssociado
    WHERE  nconta = @nConta

--LEVANTAMENTO
go

CREATE PROCEDURE dbo.Spinsertlevantamento @idOperacao           INT,
                                          @idLevantamento       INT,
                                          @montanteLevantamento MONEY,
                                          @nConta               INT
AS
    INSERT INTO dbo.levantamento
                ([idoperacao],
                 [idlevantamento],
                 [montantelevantamento],
                 [nconta])
    VALUES      (@idOperacao,
                 @idLevantamento,
                 @montanteLevantamento,
                 @nConta)

--Deletes-----------------------------------------------------------------
-------------------------------------------------------------------------
-------------------------------
--Associa
go

CREATE PROCEDURE dbo.Spdeleteassocia @nConta     INT,
                                     @idOperacao INT
```

```
AS
    DELETE FROM dbo.associa
    WHERE  nconta = @nConta
           AND idoperacao = @idOperacao

--Tem
go

CREATE PROCEDURE dbo.Spdeletetem @nConta INT,
                                 @NIF    INT
AS
    DELETE FROM dbo.tem
    WHERE  nconta = @nConta
           AND nif = @NIF

--Conta
go

CREATE PROCEDURE dbo.Spdeleteconta @nConta INT
AS
    DELETE FROM dbo.conta
    WHERE  nconta = @nConta

--SOLICITACAO
go

CREATE PROCEDURE dbo.Spdeletesolicitacao @idOperacao INT
AS
    DELETE FROM dbo.solicitacao
    WHERE  idoperacao = @idOperacao

--OPERACAO
go

CREATE PROCEDURE dbo.Spdeleteoperacao @idOperacao INT
AS
    DELETE FROM dbo.operacao
    WHERE  idoperacao = @idOperacao

--DEPOSITO
go

CREATE PROCEDURE dbo.Spdeletedeposito @idOperacao INT
AS
    DELETE FROM dbo.deposito
    WHERE  idoperacao = @idOperacao

--TRANSFERENCIA
go

CREATE PROCEDURE dbo.Spdeletetransferencia @idOperacao INT
AS
    DELETE FROM dbo.transferencia
    WHERE  idoperacao = @idOperacao

--LEVANTAMENTO
```

```
go

CREATE PROCEDURE dbo.Spdeletelevantamento @idOperacao INT
AS
    DELETE FROM dbo.levantamento
    WHERE  idoperacao = @idOperacao

--EMPRESTIMO
go

CREATE PROCEDURE dbo.Spdeleteemprestimo @idOperacao INT
AS
    DELETE FROM dbo.emprestimo
    WHERE  idoperacao = @idOperacao

------------------------------------------------------------------------
------------------------------------------------------------------
--EMPRESTIMO
go

CREATE PROCEDURE dbo.Spinsertemprestimo
  -- Add the parameters for the stored procedure here
  @idOperacao      INT,
  @idEmprestimo    INT,
  @taxaJuro        INT,
  @nconta          INT,
  @valorEmprestimo MONEY,
  @tipoEmprestimo  VARCHAR(100)
AS
    INSERT INTO dbo.emprestimo
                ([idoperacao],
                 [idemprestimo],
                 [taxajuro],
                 [nconta],
                 [valoremprestimo],
                 [tipoemprestimo])
    VALUES      ( @idOperacao,
                  @idEmprestimo,
                  @taxaJuro,
                  @nconta,
                  @valorEmprestimo,
                  @tipoEmprestimo)

    SET ansi_nulls ON

go

SET quoted_identifier ON

--Cliente
go

CREATE PROCEDURE dbo.Spinsertcliente @NIF                 INT,
                                     @idCliente           INT,
                                     @nBalcao             INT,
                                     @nConta              INT,
```

```
                                    @paísResidenciaFiscal VARCHAR(60),
                                    @codReparticaoFiscal  INT,
                                    @estadoActual         VARCHAR(60),
                                    @profissao
VARCHAR(100),
                                    @balcaoOrigem
VARCHAR(200),
                                    @estadoCivil          VARCHAR
(50),
                                    @tipodocumento        VARCHAR(50),
                                    @nIdentificacao       INT,
                                    @paísEmissao
VARCHAR(100),
                                    @dataEmissao          DATE
AS
  BEGIN
      SET nocount ON

      INSERT INTO dbo.cliente
                  (nif,
                   idcliente,
                   nbalcao,
                   nconta,
                   paísresidenciafiscal,
                   codreparticaofiscal,
                   estadoactual,
                   profissao,
                   balcaoorigem,
                   estadocivil,
                   tipodocumento,
                   nidentificacao,
                   paísemissao,
                   dataemissao)
      VALUES      ( @NIF,
                    @idCliente,
                    @nBalcao,
                    @nConta,
                    @paísResidenciaFiscal,
                    @codReparticaoFiscal,
                    @estadoActual,
                    @profissao,
                    @balcaoOrigem,
                    @estadoCivil,
                    @tipodocumento,
                    @nIdentificacao,
                    @paísEmissao,
                    @dataEmissao )
  END

SET ansi_nulls ON

go

SET quoted_identifier ON

--Balcao
```

```
go

CREATE PROCEDURE dbo.Spinsertbalcao @nBalcao    INT,
                                    @ntelefone  INT,
                                    @nome       VARCHAR(100),
                                    @horario    VARCHAR(300),
                                    @país       VARCHAR(100),
                                    @localidade VARCHAR(100),
                                    @endereco   VARCHAR(200),
                                    @codPostal  VARCHAR(10),
                                    @servicos   VARCHAR(200),
                                    @fax        INT
AS
  BEGIN
      SET nocount ON

      INSERT INTO dbo.balcao
                  (nbalcao,
                   ntelefone,
                   nome,
                   horario,
                   país,
                   localidade,
                   endereco,
                   codpostal,
                   servicos,
                   fax)
      VALUES      ( @nBalcao,
                    @ntelefone,
                    @nome,
                    @horario,
                    @país,
                    @localidade,
                    @endereco,
                    @codPostal,
                    @servicos,
                    @fax )
  END

SET ansi_nulls ON

go

SET quoted_identifier ON

--Identificação
go

ALTER PROCEDURE dbo.Spinsertidentificacao @NIF            INT,
                                          @nTelefone      INT,
                                          @nome           VARCHAR(100),
                                          @dataNascimento DATE,
                                          @país           VARCHAR(60),
                                          @localidade     VARCHAR(100),
                                          @endereco       VARCHAR(200),
                                          @codPostal      VARCHAR(10),
```

```
                                        @gênero         CHAR(1),
                                        @idade          INT
AS
  BEGIN
      SET nocount ON

      INSERT INTO dbo.identificacao
                  (nif,
                   ntelefone,
                   nome,
                   datanascimento,
                   país,
                   localidade,
                   endereco,
                   codpostal,
                   gênero,
                   idade)
      VALUES      ( @NIF,
                    @nTelefone,
                    @nome,
                    @dataNascimento,
                    @país,
                    @localidade,
                    @endereco,
                    @codPostal,
                    @gênero,
                    @idade )
  END

--Update Cliente
go

CREATE PROCEDURE [dbo].[Spupdatecliente] @NIF                   INT,
                                         @idCliente             INT,
                                         @nBalcao               INT,
                                         @nConta                INT,
                                         @países
VARCHAR(60),
                                         @codReparticaoFiscal   INT,
                                         @estadoActual
VARCHAR(60),
                                         @profissao
VARCHAR(100),
                                         @balcaoOrigem
VARCHAR(200),
                                         @estadoCivil           VARCHAR
(50),
                                         @tipodocumento
VARCHAR(50),
                                         @nIdentificacao        INT,
                                         @países
VARCHAR(100),
                                         @dataEmissao           DATE
AS
    UPDATE [dbo].[cliente]
    SET    nif = @NIF,
```

```
            idcliente = @idCliente,
            nbalcao = @nBalcao,
            nconta = @nConta,
            paísresidenciafiscal = @paísResidenciaFiscal,
            codreparticaofiscal = @codReparticaoFiscal,
            estadoactual = @estadoActual,
            profissao = @profissao,
            balcaoorigem = @balcaoOrigem,
            estadocivil = @estadoCivil,
            tipodocumento = @tipodocumento,
            nidentificacao = @nIdentificacao,
            paísemissao = @paísEmissao,
            dataemissao = @dataEmissao
    WHERE   nif = @NIF

--Update Balcao
go

CREATE PROCEDURE [dbo].[Spupdatebalcao] @nBalcao    INT,
                                        @ntelefone  INT,
                                        @nome       VARCHAR(100),
                                        @horario    VARCHAR(300),
                                        @país       VARCHAR(100),
                                        @localidade VARCHAR(100),
                                        @endereco   VARCHAR(200),
                                        @codPostal  VARCHAR(10),
                                        @servicos   VARCHAR(200),
                                        @fax        INT
AS
    UPDATE [dbo].[balcao]
    SET    nbalcao = @nBalcao,
           ntelefone = @ntelefone,
           nome = @nome,
           horario = @horario,
           país = @país,
           localidade = @localidade,
           endereco = @endereco,
           codpostal = @codPostal,
           servicos = @servicos,
           fax = @fax
    WHERE  nbalcao = @nBalcao

--Update identificação
go

CREATE PROCEDURE [dbo].[Spupdateidentificacao] @NIF            INT,
                                               @nTelefone      INT,
                                               @nome
VARCHAR(100),
                                               @dataNascimento DATE,
                                               @país
VARCHAR(60),
                                               @localidade
VARCHAR(100),
                                               @endereco
VARCHAR(200),
```

```
                                                @codPostal
VARCHAR(10),
                                                @gênero         CHAR(1),
                                                @idade          INT
AS
    UPDATE [dbo].[identificacao]
    SET    ntelefone = @nTelefone,
           nome = @nome,
           datanascimento = @dataNascimento,
           país = @país,
           localidade = @localidade,
           endereco = @endereco,
           codpostal = @codPostal,
           gênero = @gênero,
           idade = @idade
    WHERE  nif = @NIF

--Delete Cliente
go

CREATE PROCEDURE [dbo].[Deletecliente] @NIF INT
AS
    DELETE FROM cliente
    WHERE  nif = @NIF

--Delete funcionario
go

CREATE PROCEDURE [dbo].[Deletefuncionario] @NIF INT
AS
    DELETE FROM funcionario
    WHERE  nif = @NIF

--Delete Balcao
go

CREATE PROCEDURE [dbo].[Deletebalcao] @nBalcao INT
AS
    DELETE FROM balcao
    WHERE  nbalcao = @nBalcao

--Delete identificacao
go

CREATE PROCEDURE [dbo].[Deleteidentificacao] @NIF INT
AS
    DELETE FROM cliente
    WHERE  nif = @NIF


--De forma a actualizar os saldos das contas após uma dada operação
foram usados store procedures que para além de criarem a operação
propriamente dita, verificam se é possivel
--realizar a operação. Caso nao seja nao criam a operação e emitem uma
mensagem de erro. Caso seja possivel, criam a operação pretendida e
actualizam os saldo(s) da(s) conta(s) envolvida(s).
```

```
-- Foram usadas transactions para permitir que a actualização da BD
seja feita com maior segurança
--Levantamento
USE [p4g10]

go

SET ansi_nulls ON

go

SET quoted_identifier ON

go

create PROCEDURE [dbo].[Spinsertlevantamento] @idOperacao
INT,
                                                --Paramentros de entrada
necessários para a criação de um levantamento
                                                @idLevantamento       INT,
                                                @montanteLevantamento
MONEY,
                                                @nConta               INT
AS
    IF( EXISTS(SELECT *
               FROM   conta
               WHERE  nconta = @nConta
                      AND saldo < @montanteLevantamento) )
      -- Condição de forma a garantir que é possivel realizar a
operação pedida
      BEGIN
          RETURN 'Não tem saldo suficiente na conta!' --Mensagem de
erro
      END

  BEGIN TRANSACTION
        INSERT INTO dbo.levantamento -- Começa a transation
                  --Sendo permitida a operação, definimos onde vamos
colocar os dados recebidos como parametros
                  ([idoperacao],
                   [idlevantamento],
                   [montantelevantamento],
                   [nconta])
      VALUES      (@idOperacao,
                   @idLevantamento,
                   @montanteLevantamento,
                   @nConta)

      UPDATE conta
      SET    saldo = saldo - @montanteLevantamento
      WHERE  nconta = @nConta -- Update do saldo na conta afectada pela
operação

        COMMIT TRAN -- transation é bem sucedida
        BEGIN             -- transation não foi bem sucedida, é feito
o rollback
```

```
            PRINT 'A transacção precisa de rollback'
                  ROLLBACK     TRAN
        END


--Deposito
go

create PROCEDURE dbo.Spinsertdeposito @idOperacao    INT,
                                      @idDeposito    INT,
                                      @nconta        INT,
                                      @valorDeposito MONEY
AS
    BEGIN TRANSACTION
      INSERT INTO dbo.deposito
                 ([idoperacao],
                  [iddeposito],
                  [nconta],
                  [valordeposito])
    VALUES       ( @idOperacao,
                   @idDeposito,
                   @nconta,
                   @valorDeposito)

    UPDATE conta
    SET    saldo = saldo + @valorDeposito
    WHERE  nconta = @nConta;

      COMMIT TRAN
        BEGIN
            PRINT 'A transacção precisa de rollback'
                  ROLLBACK     TRAN
        END


--TRANSFERENCIA

GO
create PROCEDURE dbo.Spinserttransferencia @idOperacao         INT,
                                           @idTranferencia     INT,
                                           @nContaOrigem       INT,
                                           @nContaDestinatario INT,
                                           @valorTransferencia MONEY
AS
    IF( EXISTS(SELECT *
               FROM   conta
               WHERE  nconta = @nContaOrigem
                      AND saldo < @valorTransferencia) )
      BEGIN
          RETURN 'Não tem saldo suficiente na conta!'
      END

      BEGIN TRANSACTION
    INSERT INTO dbo.transferencia
                ([idoperacao],
                 [idtranferencia],
```

```
                [ncontaorigem],
                [ncontadestinatario],
                [valortransferencia])
    VALUES      ( @idOperacao,
                  @idTranferencia,
                  @nContaOrigem,
                  @nContaDestinatario,
                  @valorTransferencia)

    UPDATE conta
    SET    saldo = saldo + @valorTransferencia
    WHERE  nconta = @nContaDestinatario

    UPDATE conta
    SET    saldo = saldo - @valorTransferencia
    WHERE  nconta = @nContaOrigem;

      COMMIT TRAN
        BEGIN
            PRINT 'A transacção precisa de rollback'
                  ROLLBACK     TRAN
        END
        A

--Emprestimo
go

create PROCEDURE dbo.Spinsertemprestimo @idOperacao      INT,
                                        @idEmprestimo    INT,
                                        @taxaJuro        INT,
                                        @nconta          INT,
                                        @valorEmprestimo MONEY,
                                        @tipoEmprestimo  VARCHAR(100)
AS
    BEGIN TRANSACTION
      INSERT INTO dbo.emprestimo
                ([idoperacao],
                 [idemprestimo],
                 [taxajuro],
                 [nconta],
                 [valoremprestimo],
                 [tipoemprestimo])
    VALUES      ( @idOperacao,
                  @idEmprestimo,
                  @taxaJuro,
                  @nconta,
                  @valorEmprestimo,
                  @tipoEmprestimo)

    UPDATE conta
    SET    saldo = saldo + @valorEmprestimo
    WHERE  nconta = @nConta;

      COMMIT TRAN
        BEGIN
            PRINT 'A transacção precisa de rollback'
```

```
                ROLLBACK     TRAN
END
```

Script das User-Defined Functions (UDF's)

```
-- AS UDF foram criadas para a pesquisa por campos da nossa aplicação,
foram criadas UDF's para cada uma das operações visiveis para o
utilizador e para a entidade cliente, apesar de no backend ser
necessário recorrer a mais entidades.
--Cada UDF recebe como parametro 3 campos, estes campos foram
escolhidos por serem os mais unicos para a respectivel pesquisa.O
objectivo destas UDF era tambem permitir a pesquisa usando qualuqer uma
das possiveis combinaições (usar apenas 2 campos para pesquisar,usar os
3, usar apenas um, etc)
--Deposito
go

CREATE FUNCTION dbo.Udfdeposito(@idOperacao INT=0,
                                @idDeposito INT=0,
                                @nconta     INT=0) --Parametros de
entrada
returns @table TABLE (
  "nome"         VARCHAR(200),
  "nif"          INT,
  "id operação"  INT,
  "id deposito"  INT,
  "número conta" INT )
--Maneira como queremos que fique escrito na nossa base de dados os
dados que iremos apresentar
WITH schemabinding, encryption
  -- Schemabinding serve para que os objectos referenciados pelas nossa
udf(que devolve uma view) não podem ser alvo de um drop
  BEGIN
      -- Encrypton para evitar que a view criada possa ser vista por
qualquer utilizador usando sp_helptext ou syscomments
      --Declaração dos varios if para cada um dos cenarios de pesquisa
      --quando necessario atraves do where é definido as condiçoes que
devem ser cumpridas para aparecerem apenas os dados pretendidos
      IF( @idOperacao = NULL
          AND @nconta = NULL
          AND @idDeposito = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            idoperacao             AS "ID Operação",
                            iddeposito             AS "ID Deposito",
                            nconta                 AS "número conta"
            FROM   dbo.identificacao,
                   dbo.deposito
        END

      IF( @idOperacao >= 0
          AND @nconta >= 0
          AND @idDeposito >= 0 )
        BEGIN
            INSERT @table
```

```
        SELECT DISTINCT nome       AS "Nome",
                        nif        AS "NIF",
                        idoperacao AS "ID Operação",
                        iddeposito AS "ID Deposito",
                        nconta     AS "número conta"
        FROM   dbo.identificacao,
               dbo.deposito
        WHERE  nconta = @nconta
               AND iddeposito = @idDeposito
               AND idoperacao = @idOperacao
    END

IF( @idOperacao = NULL
    AND @nconta >= 0
    AND @idDeposito >= 0 )
    BEGIN
        INSERT @table
        SELECT DISTINCT nome       AS "Nome",
                        nif        AS "NIF",
                        idoperacao AS "ID Operação",
                        iddeposito AS "ID Deposito",
                        nconta     AS "número conta"
        FROM   dbo.identificacao,
               dbo.deposito
        WHERE  nconta = @nconta
               AND iddeposito = @idDeposito
    END

IF( @idOperacao = NULL
    AND @nconta >= 0
    AND @idDeposito = NULL )
    BEGIN
        INSERT @table
        SELECT DISTINCT nome       AS "Nome",
                        nif        AS "NIF",
                        idoperacao AS "ID Operação",
                        iddeposito AS "ID Deposito",
                        nconta     AS "número conta"
        FROM   dbo.identificacao,
               dbo.deposito
        WHERE  nconta = @nconta
    END

IF( @idOperacao = NULL
    AND @nconta = NULL
    AND @idDeposito >= 0 )
    BEGIN
        INSERT @table
        SELECT DISTINCT nome       AS "Nome",
                        nif        AS "NIF",
                        idoperacao AS "ID Operação",
                        iddeposito AS "ID Deposito",
                        nconta     AS "número conta"
        FROM   dbo.identificacao,
               dbo.deposito
        WHERE  iddeposito = @idDeposito
```

```
        END

      IF( @idOperacao >= 0
          AND @nconta = NULL
          AND @idDeposito = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome       AS "Nome",
                            nif        AS "NIF",
                            idoperacao AS "ID Operação",
                            iddeposito AS "ID Deposito",
                            nconta     AS "número conta"
            FROM   dbo.identificacao,
                   dbo.deposito
            WHERE  idoperacao = @idOperacao
        END

      IF( @idOperacao >= 0
          AND @nconta >= 0
          AND @idDeposito = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome       AS "Nome",
                            nif        AS "NIF",
                            idoperacao AS "ID Operação",
                            iddeposito AS "ID Deposito",
                            nconta     AS "número conta"
            FROM   dbo.identificacao,
                   dbo.deposito
            WHERE  nconta = @nconta
                   AND idoperacao = @idOperacao
        END

      IF( @idOperacao >= 0
          AND @nconta = NULL
          AND @idDeposito >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome       AS "Nome",
                            nif        AS "NIF",
                            idoperacao AS "ID Operação",
                            iddeposito AS "ID Deposito",
                            nconta     AS "número conta"
            FROM   dbo.identificacao,
                   dbo.deposito
            WHERE  idoperacao = @idOperacao
                   AND iddeposito = @idDeposito
        END

      RETURN
  END;

--Emprestimo
go

CREATE FUNCTION dbo.Udfemprestimo(@idOperacao   INT=0,
```

```
                                @idEmprestimo INT=0,
                                @nconta       INT=0)
returns @table TABLE (
  "nome"          VARCHAR(200),
  "nif"           INT,
  "id operação"   INT,
  "id emprestimo" INT,
  "número conta"  INT )
WITH schemabinding, encryption
  BEGIN
      IF( @idOperacao = NULL
          AND @nconta = NULL
          AND @idEmprestimo = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            idoperacao             AS "ID Operação",
                            idemprestimo           AS "ID Emprestimo",
                            nconta                 AS "número conta"
            FROM   dbo.identificacao,
                   dbo.emprestimo
        END

      IF( @idOperacao >= 0
          AND @nconta >= 0
          AND @idEmprestimo >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome         AS "Nome",
                            nif          AS "NIF",
                            idoperacao   AS "ID Operação",
                            idemprestimo AS "ID Emprestimo",
                            nconta       AS "número conta"
            FROM   dbo.identificacao,
                   dbo.emprestimo
            WHERE  nconta = @nconta
                   AND idemprestimo = @idEmprestimo
                   AND idoperacao = @idOperacao
        END

      IF( @idOperacao = NULL
          AND @nconta >= 0
          AND @idEmprestimo >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome         AS "Nome",
                            nif          AS "NIF",
                            idoperacao   AS "ID Operação",
                            idemprestimo AS "ID Emprestimo",
                            nconta       AS "número conta"
            FROM   dbo.identificacao,
                   dbo.emprestimo
            WHERE  nconta = @nconta
                   AND idemprestimo = @idEmprestimo
        END
```

```
IF( @idOperacao = NULL
    AND @nconta >= 0
    AND @idEmprestimo = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome         AS "Nome",
                      nif          AS "NIF",
                      idoperacao   AS "ID Operação",
                      idemprestimo AS "ID Emprestimo",
                      nconta       AS "número conta"
      FROM   dbo.identificacao,
             dbo.emprestimo
      WHERE  nconta = @nconta
  END

IF( @idOperacao = NULL
    AND @nconta = NULL
    AND @idEmprestimo >= 0 )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome         AS "Nome",
                      nif          AS "NIF",
                      idoperacao   AS "ID Operação",
                      idemprestimo AS "ID Emprestimo",
                      nconta       AS "número conta"
      FROM   dbo.identificacao,
             dbo.emprestimo
      WHERE  idemprestimo = @idEmprestimo
  END

IF( @idOperacao >= 0
    AND @nconta = NULL
    AND @idEmprestimo = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome         AS "Nome",
                      nif          AS "NIF",
                      idoperacao   AS "ID Operação",
                      idemprestimo AS "ID Emprestimo",
                      nconta       AS "número conta"
      FROM   dbo.identificacao,
             dbo.emprestimo
      WHERE  idoperacao = @idOperacao
  END

IF( @idOperacao >= 0
    AND @nconta >= 0
    AND @idEmprestimo = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome         AS "Nome",
                      nif          AS "NIF",
                      idoperacao   AS "ID Operação",
                      idemprestimo AS "ID Emprestimo",
                      nconta       AS "número conta"
```

```
            FROM   dbo.identificacao,
                   dbo.emprestimo
            WHERE  nconta = @nconta
                   AND idoperacao = @idOperacao
        END

      IF( @idOperacao >= 0
          AND @nconta = NULL
          AND @idEmprestimo >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome         AS "Nome",
                            nif          AS "NIF",
                            idoperacao   AS "ID Operação",
                            idemprestimo AS "ID Emprestimo",
                            nconta       AS "número conta"
            FROM   dbo.identificacao,
                   dbo.emprestimo
            WHERE  idoperacao = @idOperacao
                   AND idemprestimo = @idEmprestimo
        END

      RETURN
  END;

--Levantamento
go

CREATE FUNCTION dbo.Udflevantamento(@idOperacao     INT=0,
                                    @idLevantamento INT=0,
                                    @nConta         INT=0)
returns @table TABLE (
  "nome"            VARCHAR(200),
  "nif"             INT,
  "id operação"     INT,
  "id levantamento" INT,
  "número conta"    INT )
WITH schemabinding, encryption
  BEGIN
      IF( @idOperacao = NULL
          AND @nConta = NULL
          AND @idLevantamento = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            idoperacao             AS "ID Operação",
                            idlevantamento         AS "ID
Levantamento",
                            nconta                 AS "número conta"
            FROM   dbo.identificacao,
                   dbo.levantamento
        END

      IF( @idOperacao >= 0
          AND @nConta >= 0
```

```
    AND @idLevantamento >= 0 )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome           AS "Nome",
                      nif            AS "NIF",
                      idoperacao     AS "ID Operação",
                      idlevantamento AS "ID Levantamento",
                      nconta         AS "número conta"
      FROM   dbo.identificacao,
             dbo.levantamento
      WHERE  nconta = @nConta
             AND idlevantamento = @idLevantamento
             AND idoperacao = @idOperacao
  END

IF( @idOperacao = NULL
    AND @nConta >= 0
    AND @idLevantamento >= 0 )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome           AS "Nome",
                      nif            AS "NIF",
                      idoperacao     AS "ID Operação",
                      idlevantamento AS "ID Levantamento",
                      nconta         AS "número conta"
      FROM   dbo.identificacao,
             dbo.levantamento
      WHERE  nconta = @nConta
             AND idlevantamento = @idLevantamento
  END

IF( @idOperacao = NULL
    AND @nConta >= 0
    AND @idLevantamento = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome           AS "Nome",
                      nif            AS "NIF",
                      idoperacao     AS "ID Operação",
                      idlevantamento AS "ID Levantamento",
                      nconta         AS "número conta"
      FROM   dbo.identificacao,
             dbo.levantamento
      WHERE  nconta = @nConta
  END

IF( @idOperacao = NULL
    AND @nConta = NULL
    AND @idLevantamento >= 0 )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome           AS "Nome",
                      nif            AS "NIF",
                      idoperacao     AS "ID Operação",
                      idlevantamento AS "ID Levantamento",
                      nconta         AS "número conta"
```

```
            FROM   dbo.identificacao,
                   dbo.levantamento
            WHERE  idlevantamento = @idLevantamento
        END

    IF( @idOperacao >= 0
        AND @nConta = NULL
        AND @idLevantamento = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome           AS "Nome",
                            nif            AS "NIF",
                            idoperacao     AS "ID Operação",
                            idlevantamento AS "ID Levantamento",
                            nconta         AS "número conta"
            FROM   dbo.identificacao,
                   dbo.levantamento
            WHERE  idoperacao = @idOperacao
        END

    IF( @idOperacao >= 0
        AND @nConta >= 0
        AND @idLevantamento = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome           AS "Nome",
                            nif            AS "NIF",
                            idoperacao     AS "ID Operação",
                            idlevantamento AS "ID Levantamento",
                            nconta         AS "número conta"
            FROM   dbo.identificacao,
                   dbo.levantamento
            WHERE  nconta = @nConta
                   AND idoperacao = @idOperacao
        END

    IF( @idOperacao >= 0
        AND @nConta = NULL
        AND @idLevantamento >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome           AS "Nome",
                            nif            AS "NIF",
                            idoperacao     AS "ID Operação",
                            idlevantamento AS "ID Levantamento",
                            nconta         AS "número conta"
            FROM   dbo.identificacao,
                   dbo.levantamento
            WHERE  idoperacao = @idOperacao
                   AND idlevantamento = @idLevantamento
        END

    RETURN
END;

--Transferencia
```

```
go

CREATE FUNCTION dbo.Udftransferencia(@idOperacao     INT=0,
                                     @idTranferencia INT=0,
                                     @nContaOrigem   INT=0)
returns @table TABLE (
  "nome"                VARCHAR(200),
  "nif"                 INT,
  "id operação"         INT,
  "id transferência"    INT,
  "número conta origem" INT )
WITH schemabinding, encryption
  BEGIN
      IF( @idOperacao = NULL
          AND @nContaOrigem = NULL
          AND @idTranferencia = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            idoperacao             AS "ID Operação",
                            idtranferencia         AS "ID
Transferência",
                            ncontaorigem           AS "número conta"
            FROM   dbo.identificacao,
                   dbo.transferencia
        END

      IF( @idOperacao >= 0
          AND @nContaOrigem >= 0
          AND @idTranferencia >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome           AS "Nome",
                            nif            AS "NIF",
                            idoperacao     AS "ID Operação",
                            idtranferencia AS "ID Transferência",
                            ncontaorigem   AS "número conta"
            FROM   dbo.identificacao,
                   dbo.transferencia
            WHERE  ncontaorigem = @nContaOrigem
                   AND idtranferencia = @idTranferencia
                   AND idoperacao = @idOperacao
        END

      IF( @idOperacao = NULL
          AND @nContaOrigem >= 0
          AND @idTranferencia >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome           AS "Nome",
                            nif            AS "NIF",
                            idoperacao     AS "ID Operação",
                            idtranferencia AS "ID Transferência",
                            ncontaorigem   AS "número conta"
            FROM   dbo.identificacao,
```

```
            dbo.transferencia
      WHERE  ncontaorigem = @nContaOrigem
            AND idtranferencia = @idTranferencia
  END

IF( @idOperacao = NULL
    AND @nContaOrigem >= 0
    AND @idTranferencia = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome           AS "Nome",
                      nif            AS "NIF",
                      idoperacao     AS "ID Operação",
                      idtranferencia AS "ID Transferência",
                      ncontaorigem   AS "número conta"
      FROM   dbo.identificacao,
             dbo.transferencia
      WHERE  ncontaorigem = @nContaOrigem
  END

IF( @idOperacao = NULL
    AND @nContaOrigem = NULL
    AND @idTranferencia >= 0 )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome           AS "Nome",
                      nif            AS "NIF",
                      idoperacao     AS "ID Operação",
                      idtranferencia AS "ID Transferência",
                      ncontaorigem   AS "número conta"
      FROM   dbo.identificacao,
             dbo.transferencia
      WHERE  idtranferencia = @idTranferencia
  END

IF( @idOperacao >= 0
    AND @nContaOrigem = NULL
    AND @idTranferencia = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome           AS "Nome",
                      nif            AS "NIF",
                      idoperacao     AS "ID Operação",
                      idtranferencia AS "ID Transferência",
                      ncontaorigem   AS "número conta"
      FROM   dbo.identificacao,
             dbo.transferencia
      WHERE  idoperacao = @idOperacao
  END

IF( @idOperacao >= 0
    AND @nContaOrigem >= 0
    AND @idTranferencia = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT nome           AS "Nome",
```

```
                        nif            AS "NIF",
                        idoperacao     AS "ID Operação",
                        idtranferencia AS "ID Transferência",
                        ncontaorigem   AS "número conta"
            FROM   dbo.identificacao,
                   dbo.transferencia
            WHERE  ncontaorigem = @nContaOrigem
                   AND idoperacao = @idOperacao
        END

      IF( @idOperacao >= 0
          AND @nContaOrigem = NULL
          AND @idTranferencia >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT nome           AS "Nome",
                        nif            AS "NIF",
                        idoperacao     AS "ID Operação",
                        idtranferencia AS "ID Transferência",
                        ncontaorigem   AS "número conta"
            FROM   dbo.identificacao,
                   dbo.transferencia
            WHERE  idoperacao = @idOperacao
                   AND idtranferencia = @idTranferencia
        END

      RETURN
  END;

--Conta
go

CREATE FUNCTION dbo.Udfconta(@Nome   VARCHAR(200),
                             @NIF    INT=0,
                             @nConta INT=0)
returns @table TABLE (
  "nome"             VARCHAR(200),
  "nif"              INT,
  "saldo"            MONEY,
  "tipo conta"       VARCHAR(200),
  "balcão associado" VARCHAR(200))
WITH schemabinding, encryption
  BEGIN
      IF( @Nome = NULL
          AND @nConta = NULL
          AND @NIF = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                        nif                    AS "NIF",
                        saldo                  AS "Saldo",
                        tipoconta              AS "Tipo Conta",
                        balcaoassociado        AS "Balcão
Associado"
            FROM   dbo.identificacao,
                   dbo.conta
```

```
            WHERE  dbo.identificacao.nif = dbo.conta.nif
        END

      IF( Datalength(@Nome) != 0
          AND @nConta >= 0
          AND @NIF >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            saldo                  AS "Saldo",
                            tipoconta              AS "Tipo Conta",
                            balcaoassociado        AS "Balcão
Associado"
            FROM   dbo.identificacao,
                   dbo.conta
            WHERE  nconta = @nConta
                   AND nif = @NIF
                   AND nome = @Nome
                   AND dbo.identificacao.nif = dbo.conta.nif
        END

      IF( Datalength(@Nome) = 0
          AND @nConta >= 0
          AND @NIF >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            saldo                  AS "Saldo",
                            tipoconta              AS "Tipo Conta",
                            balcaoassociado        AS "Balcão
Associado"
            FROM   dbo.identificacao,
                   dbo.conta
            WHERE  nconta = @nConta
                   AND nif = @NIF
                   AND dbo.identificacao.nif = dbo.conta.nif
        END

      IF( Datalength(@Nome) = 0
          AND @nConta >= 0
          AND @NIF = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            saldo                  AS "Saldo",
                            tipoconta              AS "Tipo Conta",
                            balcaoassociado        AS "Balcão
Associado"
            FROM   dbo.identificacao,
                   dbo.conta
            WHERE  nconta = @nConta
                   AND dbo.identificacao.nif = dbo.conta.nif
        END
```

```
      IF( Datalength(@Nome) = 0
          AND @nConta = NULL
          AND @NIF >= 0 )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            saldo                  AS "Saldo",
                            tipoconta              AS "Tipo Conta",
                            balcaoassociado        AS "Balcão
Associado"
            FROM   dbo.identificacao,
                   dbo.conta
            WHERE  nif = @NIF
                   AND dbo.identificacao.nif = dbo.conta.nif
        END

      IF( Datalength(@Nome) != 0
          AND @nConta = NULL
          AND @NIF = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            saldo                  AS "Saldo",
                            tipoconta              AS "Tipo Conta",
                            balcaoassociado        AS "Balcão
Associado"
            FROM   dbo.identificacao,
                   dbo.conta
            WHERE  nome = @Nome
                   AND dbo.identificacao.nif = dbo.conta.nif
        END

      IF( Datalength(@Nome) != 0
          AND @nConta >= 0
          AND @NIF = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            saldo                  AS "Saldo",
                            tipoconta              AS "Tipo Conta",
                            balcaoassociado        AS "Balcão
Associado"
            FROM   dbo.identificacao,
                   dbo.conta
            WHERE  nconta = @nConta
                   AND nome = @Nome
                   AND dbo.identificacao.nif = dbo.conta.nif
        END

      IF( Datalength(@Nome) != 0
          AND @nConta = NULL
          AND @NIF >= 0 )
```

```
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            nif                    AS "NIF",
                            saldo                  AS "Saldo",
                            tipoconta              AS "Tipo Conta",
                            balcaoassociado        AS "Balcão
Associado"
            FROM   dbo.identificacao,
                   dbo.conta
            WHERE  nome = @Nome
                   AND nif = @NIF
                   AND dbo.identificacao.nif = dbo.conta.nif
        END

      RETURN
  END;

--Cliente
go

CREATE FUNCTION dbo.Udfcliente(@Nome      VARCHAR(200)=NULL,
                               @NIF       INT=0,
                               @idCliente INT=0)
returns @table TABLE (
  "nome"            VARCHAR(200),
  "nif"             INT,
  "id cliente"      INT,
  "número de conta" INT,
  "saldo"           MONEY )
WITH schemabinding, encryption
  -- Schemabinding serve para que os objectos referenciados pelas nossa
udf não podem ser alvo de um drop
  BEGIN
      -- Encrypton para evitar que a view criada possa ser vista por
qualquer utilizador usando sp_helptext ou syscomments
      --Declaração dos varios if para cada um dos cenarios de pesquisa
      --quando necessario atraves do where é definido as condiçoes que
devem ser cumpridas para aparecerem apenas os dados pretendidos
      IF( @Nome = NULL
          AND @NIF = NULL
          AND @idCliente = NULL )
        BEGIN
            INSERT @table
            SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                            cliente.nif            AS "NIF",
                            idcliente              AS "ID Cliente",
                            conta.nconta           AS "Número conta",
                            saldo                  AS "Saldo"
            FROM   dbo.identificacao,
                   dbo.cliente,
                   dbo.conta
            WHERE  identificacao.nif = cliente.nif
                   AND conta.nconta = cliente.nconta;
        END
```

```
IF( @Nome >= 0
    AND @NIF >= 0
    AND @idCliente >= 0 )
  BEGIN
      INSERT @table
      SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                      cliente.nif            AS "NIF",
                      idcliente              AS "ID Cliente",
                      conta.nconta           AS "Número conta",
                      saldo                  AS "Saldo"
      FROM   dbo.identificacao,
             dbo.cliente,
             dbo.conta
      WHERE  cliente.nif = @NIF
             AND idcliente = @idCliente
             AND nome = @Nome
             AND identificacao.nif = cliente.nif
             AND conta.nconta = cliente.nconta;
  END

IF( @Nome = NULL
    AND @NIF >= 0
    AND @idCliente >= 0 )
  BEGIN
      INSERT @table
      SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                      cliente.nif            AS "NIF",
                      idcliente              AS "ID Cliente",
                      conta.nconta           AS "Número conta",
                      saldo                  AS "Saldo"
      FROM   dbo.identificacao,
             dbo.cliente,
             dbo.conta
      WHERE  cliente.nif = @NIF
             AND nome = @idCliente
             AND identificacao.nif = cliente.nif
             AND conta.nconta = cliente.nconta;
  END

IF( @Nome = NULL
    AND @NIF >= 0
    AND @idCliente = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                      cliente.nif            AS "NIF",
                      idcliente              AS "ID Cliente",
                      conta.nconta           AS "Número conta",
                      saldo                  AS "Saldo"
      FROM   dbo.identificacao,
             dbo.cliente,
             dbo.conta
      WHERE  cliente.nif = @NIF
             AND identificacao.nif = cliente.nif
             AND conta.nconta = cliente.nconta;
  END
```

```
IF( @Nome = NULL
    AND @NIF = NULL
    AND @idCliente >= 0 )
  BEGIN
      INSERT @table
      SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                      cliente.nif            AS "NIF",
                      idcliente              AS "ID Cliente",
                      conta.nconta           AS "Número conta",
                      saldo                  AS "Saldo"
      FROM   dbo.identificacao,
             dbo.cliente,
             dbo.conta
      WHERE  idcliente = @idCliente
             AND identificacao.nif = cliente.nif
             AND conta.nconta = cliente.nconta;
  END

IF( @Nome >= 0
    AND @NIF = NULL
    AND @idCliente = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                      cliente.nif            AS "NIF",
                      idcliente              AS "ID Cliente",
                      conta.nconta           AS "Número conta",
                      saldo                  AS "Saldo"
      FROM   dbo.identificacao,
             dbo.cliente,
             dbo.conta
      WHERE  nome = @Nome
             AND identificacao.nif = cliente.nif
             AND conta.nconta = cliente.nconta;
  END

IF( @Nome >= 0
    AND @NIF >= 0
    AND @idCliente = NULL )
  BEGIN
      INSERT @table
      SELECT DISTINCT dbo.identificacao.nome AS "Nome",
                      cliente.nif            AS "NIF",
                      idcliente              AS "ID Cliente",
                      conta.nconta           AS "Número conta",
                      saldo                  AS "Saldo"
      FROM   dbo.identificacao,
             dbo.cliente,
             dbo.conta
      WHERE  nome = @Nome
             AND idcliente = @idCliente
             AND identificacao.nif = cliente.nif
             AND conta.nconta = cliente.nconta;
  END
```

```
    RETURN
END;
```

Script dos Triggers

```
DROP TRIGGER triggerdeletebalcao

DROP TRIGGER triggerdeleteemprestimo

DROP TRIGGER triggerdeletelevantamento

DROP TRIGGER triggerdeletetransferencia

DROP TRIGGER triggerdeletedeposito

USE p4g10

-- Caso um balcão tenha sido eliminado, actualiza o campo 'função' da
tabela funcionário
go

CREATE TRIGGER triggerdeletebalcao
ON dbo.balcao
after DELETE
AS
  BEGIN
      UPDATE dbo.funcionario
      SET    funcao = 'Sem Balcão Origem'
      WHERE  EXISTS (SELECT *
                     FROM   dbo.funcionario,
                            dbo.balcao
                     WHERE  dbo.funcionario.nbalcao =
dbo.balcao.nbalcao);
  END;

-- No caso de um emprestimo ser removido, remover também a operação
associada
go

CREATE TRIGGER triggerdeleteemprestimo
ON dbo.emprestimo
after DELETE
AS
  BEGIN
      DELETE dbo.operacao
      FROM   dbo.operacao,
             dbo.emprestimo
      WHERE  dbo.operacao.idoperacao = dbo.emprestimo.idoperacao

      DELETE dbo.associa
      FROM   dbo.operacao,
             dbo.emprestimo,
             dbo.associa
      WHERE  dbo.operacao.idoperacao = dbo.emprestimo.idoperacao
             AND dbo.associa.idoperacao = dbo.operacao.idoperacao

      DELETE dbo.solicitao
```

```
      FROM   dbo.operacao,
             dbo.emprestimo,
             dbo.solicitao
      WHERE  dbo.operacao.idoperacao = dbo.emprestimo.idoperacao
             AND dbo.solicitao.idoperacao = dbo.operacao.idoperacao
  END;

-- No caso de um levantamento ser removido, remover também a operação
associada
go

CREATE TRIGGER triggerdeletelevantamento
ON dbo.levantamento
after DELETE
AS
  BEGIN
      DELETE dbo.operacao
      FROM   dbo.operacao,
             dbo.levantamento
      WHERE  dbo.operacao.idoperacao = dbo.levantamento.idoperacao

      DELETE dbo.associa
      FROM   dbo.operacao,
             dbo.levantamento,
             dbo.associa
      WHERE  dbo.operacao.idoperacao = dbo.levantamento.idoperacao
             AND dbo.associa.idoperacao = dbo.operacao.idoperacao

      DELETE dbo.solicitao
      FROM   dbo.operacao,
             dbo.levantamento,
             dbo.solicita
      WHERE  dbo.operacao.idoperacao = dbo.levantamento.idoperacao
             AND dbo.solicitao.idoperacao = dbo.operacao.idoperacao
  END;

-- No caso de uma transferência ser removida, remover também a operação
associada
go

CREATE TRIGGER triggerdeletetransferencia
ON dbo.transferencia
after DELETE
AS
  BEGIN
      DELETE dbo.operacao
      FROM   dbo.operacao,
             dbo.transferencia
      WHERE  dbo.operacao.idoperacao = dbo.transferencia.idoperacao

      DELETE dbo.associa
      FROM   dbo.operacao,
             dbo.transferencia,
             dbo.associa
      WHERE  dbo.operacao.idoperacao = dbo.transferencia.idoperacao
             AND dbo.associa.idoperacao = dbo.operacao.idoperacao
```

```
      DELETE dbo.solicitao
      FROM   dbo.operacao,
             dbo.transferencia,
             dbo.solicitao
      WHERE  dbo.operacao.idoperacao = dbo.transferencia.idoperacao
             AND dbo.solicitao.idoperacao = dbo.operacao.idoperacao
  END;

-- No caso de um deposito ser removido, remover também a operação
associada
go

CREATE TRIGGER triggerdeletedeposito
ON dbo.deposito
after DELETE
AS
  BEGIN
      DELETE dbo.operacao
      FROM   dbo.operacao,
             dbo.deposito
      WHERE  dbo.operacao.idoperacao = dbo.deposito.idoperacao

      DELETE dbo.associa
      FROM   dbo.operacao,
             dbo.deposito,
             dbo.associa
      WHERE  dbo.operacao.idoperacao = dbo.deposito.idoperacao
             AND dbo.associa.idoperacao = dbo.operacao.idoperacao

      DELETE dbo.solicitao
      FROM   dbo.operacao,
             dbo.deposito,
             dbo.solicitao
      WHERE  dbo.operacao.idoperacao = dbo.deposito.idoperacao
             AND dbo.solicitao.idoperacao = dbo.operacao.idoperacao
  END;
```

Script do Cursor

```
EXEC Sp_findstringintable
  'tuga',
  dbo,
  balcao

CREATE PROCEDURE Sp_findstringintable @stringToFind VARCHAR(100),
                                      @schema       SYSNAME,
                                      @table        SYSNAME
AS
    DECLARE @sqlCommand VARCHAR(8000)
    DECLARE @where VARCHAR(8000)
    DECLARE @columnName SYSNAME
    DECLARE @cursor VARCHAR(8000)

  BEGIN try
      SET @sqlCommand = 'SELECT * FROM [' + @schema + '].[' + @table
                        + '] WHERE'
      SET @where = ''
      SET @cursor = 'DECLARE col_cursor CURSOR FOR SELECT COLUMN_NAME
FROM ' + Db_name()
                    + '.INFORMATION_SCHEMA.COLUMNS     WHERE
TABLE_SCHEMA like ''' + @schema
                    + '''     AND TABLE_NAME like ''%' + @table
                    +
'%''     AND DATA_TYPE IN
(''char'',''nchar'',''ntext'',''nvarchar'',''text'',''varchar'')'

    EXEC (@cursor)

    OPEN col_cursor

    FETCH next FROM col_cursor INTO @columnName

    WHILE @@FETCH_STATUS = 0
      BEGIN
          IF @where <> ''
            SET @where = @where + ' OR'

          SET @where = @where + ' [' + @columnName + '] LIKE ''%'
                       + @stringToFind + '%'''

          FETCH next FROM col_cursor INTO @columnName
      END

    CLOSE col_cursor

    DEALLOCATE col_cursor

    SET @sqlCommand = @sqlCommand + @where

EXEC (@sqlCommand)
END try
```

```
BEGIN catch
    PRINT 'Houve um erro. Verifique se o objecto existe!'

    IF Cursor_status('variable', 'col_cursor') <> -3
      BEGIN
          CLOSE col_cursor

          DEALLOCATE col_cursor
      END
END catch
```

Script dos Creates

```
CREATE TABLE identificacao
  (
     nif            INT NOT NULL,
     ntelefone      INT NOT NULL,
     nome           VARCHAR(100) NOT NULL,
     datanascimento DATE NOT NULL,
     país           VARCHAR(60) NOT NULL,
     localidade     VARCHAR(100) NOT NULL,
     endereco       VARCHAR(200) NOT NULL,
     codpostal      VARCHAR(10) NOT NULL,
     gênero         CHAR(1) NOT NULL,
     idade          INT NOT NULL,
     PRIMARY KEY (nif)
  );

CREATE TABLE balcao
  (
     nbalcao    INT NOT NULL,
     ntelefone  INT,
     nome       VARCHAR(100) NOT NULL,
     horario    VARCHAR(300) NOT NULL,
     país       VARCHAR(100) NOT NULL,
     localidade VARCHAR(100) NOT NULL,
     endereco   VARCHAR(200) NOT NULL,
     codpostal  VARCHAR(10) NOT NULL,
     servicos   VARCHAR(200),
     fax        INT,
     PRIMARY KEY (nbalcao)
  );

CREATE TABLE cliente
  (
     nif                  INT UNIQUE NOT NULL,
     idcliente            INT NOT NULL,
     nbalcao              INT NOT NULL,
     nconta               INT NOT NULL,
     paísresidenciafiscal VARCHAR(60) NOT NULL,
     codreparticaofiscal  INT NOT NULL,
     estadoactual         VARCHAR(60) NOT NULL,
     profissao            VARCHAR(100),
     balcaoorigem         VARCHAR(200) NOT NULL,
     estadocivil          VARCHAR (50),
     tipodocumento        VARCHAR(50) NOT NULL,
     nidentificacao       INT NOT NULL,
     paísemissao          VARCHAR(100) NOT NULL,
     dataemissao          DATE NOT NULL,
     PRIMARY KEY (nif, idcliente)
  );

CREATE TABLE funcionario
  (
     nif            INT UNIQUE NOT NULL,
     idfuncionario  INT NOT NULL,
     horariot       VARCHAR(200) NOT NULL,
```

```
    funcao         VARCHAR(200) NOT NULL,
    salario        MONEY NOT NULL,
    balcaotrabalho VARCHAR(100) NOT NULL,
    nbalcao        INT NOT NULL,
    PRIMARY KEY (nif, idfuncionario)
  );

CREATE TABLE transferencia
  (
    idoperacao         INT NOT NULL,
    idtranferencia     INT NOT NULL,
    ncontaorigem       INT NOT NULL,
    ncontadestinatario INT NOT NULL,
    valortransferencia MONEY NOT NULL,
    PRIMARY KEY (idoperacao)
  );

CREATE TABLE levantamento
  (
    idoperacao           INT NOT NULL,
    idlevantamento       INT NOT NULL,
    montantelevantamento MONEY NOT NULL,
    nconta               INT NOT NULL,
    PRIMARY KEY (idoperacao)
  );

CREATE TABLE emprestimo
  (
    idoperacao      INT NOT NULL,
    idemprestimo    INT NOT NULL,
    taxajuro        INT NOT NULL,
    nconta          INT NOT NULL,
    valoremprestimo MONEY NOT NULL,
    tipoemprestimo  VARCHAR(100) NOT NULL,
    PRIMARY KEY (idoperacao)
  );

CREATE TABLE conta
  (
    nconta          INT NOT NULL,
    tipoconta       VARCHAR(100) NOT NULL,
    balcaoassociado VARCHAR(100) NOT NULL,
    saldo           MONEY NOT NULL,
    PRIMARY KEY (nconta)
  );

CREATE TABLE operacao
  (
    idoperacao INT NOT NULL,
    tipo       VARCHAR(40) NOT NULL,
    data       DATE NOT NULL,
    hora       TIME NOT NULL,
    nif        INT NOT NULL,
    PRIMARY KEY (idoperacao)
  );
```

```
CREATE TABLE solicitacao
  (
     idoperacao INT NOT NULL,
     nif        INT NOT NULL,
     PRIMARY KEY (idoperacao, nif)
  );

CREATE TABLE deposito
  (
     idoperacao    INT NOT NULL,
     iddeposito    INT NOT NULL,
     nconta        INT NOT NULL,
     valordeposito MONEY NOT NULL,
     PRIMARY KEY (idoperacao)
  );

CREATE TABLE associa
  (
     nconta     INT NOT NULL,
     idoperacao INT NOT NULL,
     PRIMARY KEY (nconta, idoperacao)
  );

CREATE TABLE tem
  (
     nconta INT NOT NULL,
     nif    INT NOT NULL,
     PRIMARY KEY (nconta, nif)
  );

ALTER TABLE cliente
  ADD CONSTRAINT fobalcaonbalcao FOREIGN KEY (nbalcao) REFERENCES
balcao(nbalcao
  );

ALTER TABLE cliente
  ADD CONSTRAINT foclienteiden FOREIGN KEY (nif) REFERENCES
identificacao(nif);

ALTER TABLE funcionario
  ADD CONSTRAINT fofuncionaiden FOREIGN KEY (nif) REFERENCES
identificacao(nif);

ALTER TABLE operacao
  ADD CONSTRAINT foopnif FOREIGN KEY (nif) REFERENCES funcionario(nif);

ALTER TABLE funcionario
  ADD CONSTRAINT fofunnbalcao FOREIGN KEY (nbalcao) REFERENCES
balcao(nbalcao);

ALTER TABLE solicitacao
  ADD CONSTRAINT solicitacaooperacaoidop FOREIGN KEY (idoperacao)
REFERENCES
  operacao(idoperacao);

ALTER TABLE solicitacao
```

```
  ADD CONSTRAINT solicitacaooperacaonif FOREIGN KEY (nif) REFERENCES
cliente(nif
  );

ALTER TABLE tem
  ADD CONSTRAINT temclientenif FOREIGN KEY (nif) REFERENCES
cliente(nif);

ALTER TABLE tem
  ADD CONSTRAINT temcontanconta FOREIGN KEY (nconta) REFERENCES
conta(nconta);

ALTER TABLE associa
  ADD CONSTRAINT associacontanconta FOREIGN KEY (nconta) REFERENCES
conta(nconta
  );

ALTER TABLE associa
  ADD CONSTRAINT associaoperacaoidop FOREIGN KEY (idoperacao)
REFERENCES
  operacao(idoperacao);

ALTER TABLE emprestimo
  ADD CONSTRAINT emprestimooperacaoidop FOREIGN KEY (idoperacao)
REFERENCES
  operacao(idoperacao);

ALTER TABLE levantamento
  ADD CONSTRAINT levantamentooperacaoidop FOREIGN KEY (idoperacao)
REFERENCES
  operacao(idoperacao);

ALTER TABLE transferencia
  ADD CONSTRAINT transferenciaoperacaoidop FOREIGN KEY (idoperacao)
REFERENCES
  operacao(idoperacao);

ALTER TABLE deposito
  ADD CONSTRAINT depositooperacaoidop FOREIGN KEY (idoperacao)
REFERENCES
```

Scripts dos Inserts

```
INSERT INTO identificacao
VALUES     (1234567,
            100100100,
            'Antonio Silva',
            '02/02/1991',
            'Portugal',
            'Abrantes',
            'Rua da esquina',
            '2000-012',
            'M',
            '23');

INSERT INTO identificacao
VALUES     (1234566,
            100100101,
            'Imaculada Albertina',
            '02/02/1981',
            'Espanha',
            'Madrid',
            'Rua dali',
            '3000-020',
            'M',
            '33');

INSERT INTO identificacao
VALUES     (1234565,
            100100102,
            'Joana Joaquim',
            '02/02/1971',
            'Holanda',
            'Amsterdam',
            'Rua Ta tudo',
            '3000-032',
            'F',
            '43');

INSERT INTO identificacao
VALUES     (1234564,
            100100103,
            'Paula Santos',
            '02/02/1961',
            'Escocia',
            'Edimburg',
            'Rua do castelo',
            '6000-200',
            'F',
            '53');

INSERT INTO identificacao
VALUES     (1234563,
            100100104,
            'Cesar Figueira',
            '02/02/1951',
            'França',
```

```
            'Paris',
            'Rua Saint qualquer coisa',
            '2000-011',
            'F',
            '63');

INSERT INTO identificacao
VALUES     (1234562,
            100100105,
            'Jon Doe',
            '02/02/1941',
            'Estados Unidos',
            'New York',
            'Rua nada',
            '2000-012',
            'M',
            '73');

INSERT INTO identificacao
VALUES     (1234561,
            100100106,
            'Jonh Jonhson',
            '02/02/1931',
            'Inglaterra',
            'London',
            'Rua de outra coisa',
            '5000-012',
            'M',
            '83');

INSERT INTO identificacao
VALUES     (1234560,
            100100107,
            'Mario Super',
            '02/02/1921',
            'Italia',
            'Rome',
            'Rua da pizza',
            '2000-012',
            'M',
            '93');

INSERT INTO identificacao
VALUES     (1234557,
            100100108,
            'Jacqueline Ant',
            '02/02/1911',
            'Suiça',
            'Neuchatel',
            'Rua do queijo',
            '3240-012',
            'F',
            '103');

INSERT INTO identificacao
VALUES     (1234547,
```

```
            100100109,
            'Wory Wonk',
            '02/02/1994',
            'Dinamarca',
            'Copenhage',
            'Rua do frio',
            '2200-012',
            'F',
            '20');

INSERT INTO identificacao
VALUES     (1234568,
            100100110,
            'Antonio Silva',
            '02/02/1991',
            'Portugal',
            'Abrantes',
            'Rua da esquina',
            '2000-062',
            'M',
            '23');

INSERT INTO identificacao
VALUES     (1234569,
            100100111,
            'Antonia da Silva',
            '02/02/1991',
            'Portugal',
            'Abrantes',
            'Rua da esquino',
            '2000-032',
            'F',
            '23');

INSERT INTO identificacao
VALUES     (1234570,
            100100112,
            'Jaquino Silva',
            '02/02/1991',
            'Portugal',
            'Abrantes',
            'Rua da cena',
            '2000-002',
            'M',
            '23');

INSERT INTO identificacao
VALUES     (1234571,
            100100113,
            'Anácio Silva',
            '02/02/1991',
            'Portugal',
            'Abrantes',
            'Rua diferente das outras',
            '2000-042',
            'M',
```

```
            '23');

INSERT INTO identificacao
VALUES      (1234572,
             100100114,
             'Joca Silva',
             '02/02/1991',
             'Portugal',
             'Abrantes',
             'Rua onde vivo',
             '2000-022',
             'M',
             '23');

INSERT INTO balcao
VALUES      (0,
             200200200,
             'Banco Lisboa',
             '8h-10h e 13h00h',
             'Portugal',
             'Lisboa',
             'Rua do banco',
             '2000-012',
             'todos',
             0);

INSERT INTO balcao
VALUES      (1,
             200200201,
             'Banco Lisboa novo',
             '6h-11h e 18h00h',
             'Portugal',
             'Lisboa',
             'Rua do banco',
             '2000-012',
             'todos',
             1);

INSERT INTO balcao
VALUES      (2,
             200200202,
             'Banco Porto',
             '8h-10h e 13h16h',
             'Portugal',
             'Porto',
             'Rua do banco',
             '2000-012',
             'todos',
             2);

INSERT INTO balcao
VALUES      (3,
             200200203,
             'Banco Porto novo',
             '6h-11h e 18h00h',
             'Portugal',
```

```
            'Porto',
            'Rua do banco',
            '2000-012',
            'todos',
            3);

INSERT INTO balcao
VALUES     (4,
            200200204,
            'Banco Madrid',
            '8h-10h e 20h00h',
            'Espanha',
            'Espanha',
            'Rua do banco',
            '2000-012',
            'todos',
            4);

INSERT INTO balcao
VALUES     (5,
            200200205,
            'Banco Abrantes',
            '5h-10h e 19h00h',
            'Portugal',
            'Abrantes',
            'Rua do banco',
            '2000-012',
            'todos',
            5);

INSERT INTO balcao
VALUES     (6,
            200200206,
            'Banco Paris',
            '8h-10h e 13h00h',
            'França',
            'Paris',
            'Rua do banco',
            '2000-012',
            'todos',
            6);

INSERT INTO balcao
VALUES     (7,
            200200207,
            'Banco Madrid novo',
            '8h-10h e 13h16h',
            'Espanha',
            'Madrid',
            'Rua do banco',
            '2000-012',
            'todos',
            7);

INSERT INTO balcao
VALUES     (8,
```

```
            200200208,
            'Banco London',
            '8h-10h e 13h16h',
            'Inglaterra',
            'London',
            'Rua do banco',
            '2000-012',
            'todos',
            8);

INSERT INTO balcao
VALUES     (9,
            200200209,
            'Banco New York',
            '7h-10h e 13h30h',
            'Estados Unidos',
            'New York',
            'Rua do banco',
            '2000-012',
            'todos',
            9);

INSERT INTO cliente
VALUES     (1234567,
            0,
            0,
            0,
            'Portugal',
            0,
            'nenhum',
            'Professor',
            'Banco Lisboa',
            'Solteiro',
            'B.I',
            0,
            'Portugal',
            '02/02/2000');

INSERT INTO cliente
VALUES     (1234566,
            1,
            1,
            1,
            'Portugal',
            1,
            'nenhum',
            'Atleta',
            'Banco Porto',
            'Solteiro',
            'B.I',
            1,
            'Portugal',
            '02/02/2000');

INSERT INTO cliente
VALUES     (1234565,
```

```
            2,
            2,
            2,
            'Portugal',
            2,
            'nenhum',
            'Pintor',
            'Banco Madrid',
            'Solteiro',
            'B.I',
            2,
            'Portugal',
            '02/02/2000');

INSERT INTO cliente
VALUES     (1234564,
            3,
            3,
            3,
            'Portugal',
            3,
            'nenhum',
            'Musico',
            'Banco Lisboa novo',
            'Solteiro',
            'B.I',
            3,
            'Portugal',
            '02/02/2000');

INSERT INTO cliente
VALUES     (1234563,
            4,
            4,
            4,
            'Portugal',
            4,
            'nenhum',
            'Universitario',
            'Banco Lisboa',
            'Solteiro',
            'B.I',
            4,
            'Portugal',
            '02/02/2000');

INSERT INTO cliente
VALUES     (1234568,
            5,
            5,
            5,
            'Portugal',
            5,
            'nenhum',
            'Professor',
            'Banco Lisboa',
```

```
            'Solteiro',
            'B.I',
            5,
            'Portugal',
            '02/02/2000');

INSERT INTO cliente
VALUES     (1234569,
            6,
            6,
            6,
            'Portugal',
            6,
            'nenhum',
            'Atleta',
            'Banco Porto',
            'Solteira',
            'B.I',
            6,
            'Portugal',
            '02/02/2000');

INSERT INTO cliente
VALUES     (1234570,
            7,
            7,
            7,
            'Portugal',
            7,
            'nenhum',
            'Pintor',
            'Banco Madrid',
            'Casado',
            'B.I',
            7,
            'Portugal',
            '02/02/2000');

INSERT INTO cliente
VALUES     (1234571,
            8,
            8,
            8,
            'Portugal',
            8,
            'nenhum',
            'Musico',
            'Banco Lisboa novo',
            'Solteiro',
            'B.I',
            8,
            'Portugal',
            '02/02/2000');

INSERT INTO cliente
VALUES     (1234572,
```

```
            9,
            9,
            9,
            'Portugal',
            9,
            'nenhum',
            'Universitario',
            'Banco Lisboa',
            'Solteiro',
            'B.I',
            9,
            'Portugal',
            '02/02/2000');

INSERT INTO funcionario
VALUES     (1234562,
            0,
            '8h-16h',
            'Caixa',
            100,
            'Banco Lisboa',
            0);

INSERT INTO funcionario
VALUES     (1234561,
            1,
            '8h-16h',
            'Contabilista',
            200,
            'Banco Lisboa nova',
            1);

INSERT INTO funcionario
VALUES     (1234560,
            2,
            '8h-16h',
            'Supervisor',
            1000,
            'Banco Madrid',
            4);

INSERT INTO funcionario
VALUES     (1234557,
            3,
            '8h-16h',
            'Gerente',
            2000,
            'Banco Porto',
            2);

INSERT INTO funcionario
VALUES     (1234547,
            4,
            '8h-16h',
            'Caixa',
            100,
```

```
            'Banco Abrantes',
            5);

INSERT INTO conta
VALUES      (0,
            'Poupança',
            'Banco Lisboa',
            500000);

INSERT INTO conta
VALUES      (1,
            'Investimento',
            'Banco Lisboa nova',
            600000);

INSERT INTO conta
VALUES      (2,
            'Poupança',
            'Banco Porto',
            700000);

INSERT INTO conta
VALUES      (3,
            'Investimento',
            'Banco Porto nova',
            800000);

INSERT INTO conta
VALUES      (4,
            'Poupança',
            'Banco Madrid',
            300450);

INSERT INTO conta
VALUES      (5,
            'Poupança',
            'Banco Madrid nova',
            400000);

INSERT INTO conta
VALUES      (6,
            'Poupança',
            'Banco Abrantes',
            300000);

INSERT INTO conta
VALUES      (7,
            'Poupança',
            'Banco Lisboa',
            900000);

INSERT INTO conta
VALUES      (8,
            'Poupança',
            'Banco Lisboa',
            1200000);
```

```
INSERT INTO conta
VALUES     (9,
            'Poupança',
            'Banco London',
            100000);

INSERT INTO operacao
VALUES     (0,
            'transferencia',
            '02/03/2014',
            '18:10:10 PM',
            1234562);

INSERT INTO operacao
VALUES     (1,
            'transferencia',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (2,
            'transferencia',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (3,
            'transferencia',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (4,
            'transferencia',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (5,
            'transferencia',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (6,
            'transferencia',
            '02/03/2014',
            '8:10:10 AM',
            1234562);
```

```
INSERT INTO operacao
VALUES     (7,
            'transferencia',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (8,
            'transferencia',
            '02/03/2014',
            '14:20:10 PM',
            1234562);

INSERT INTO operacao
VALUES     (9,
            'transferencia',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (10,
            'levantamento',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (11,
            'levantamento',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (12,
            'levantamento',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (13,
            'levantamento',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (14,
            'levantamento',
            '02/03/2014',
            '10:10:10 AM',
            1234562);
```

```
INSERT INTO operacao
VALUES     (15,
            'levantamento',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (16,
            'levantamento',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (17,
            'levantamento',
            '02/03/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (18,
            'levantamento',
            '02/03/2014',
            '10:10:10 AM',
            1234561);

INSERT INTO operacao
VALUES     (19,
            'levantamento',
            '02/03/2014',
            '10:10:10 PM',
            1234561);

INSERT INTO operacao
VALUES     (20,
            'emprestimo',
            '02/03/2014',
            '11:10:10 AM',
            1234560);

INSERT INTO operacao
VALUES     (21,
            'emprestimo',
            '02/05/2014',
            '10:10:10 PM',
            1234560);

INSERT INTO operacao
VALUES     (22,
            'emprestimo',
            '02/05/2014',
            '10:10:10 AM',
            1234560);
```

```
INSERT INTO operacao
VALUES     (23,
            'emprestimo',
            '02/05/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (24,
            'emprestimo',
            '02/05/2014',
            '10:10:10 PM',
            1234560);

INSERT INTO operacao
VALUES     (25,
            'emprestimo',
            '02/05/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (26,
            'emprestimo',
            '02/04/2014',
            '10:10:10 PM',
            1234562);

INSERT INTO operacao
VALUES     (27,
            'emprestimo',
            '02/04/2014',
            '10:10:10 AM',
            1234562);

INSERT INTO operacao
VALUES     (28,
            'emprestimo',
            '02/04/2014',
            '10:10:10 PM',
            1234561);

INSERT INTO operacao
VALUES     (29,
            'emprestimo',
            '02/04/2014',
            '10:10:10 PM',
            1234561);

INSERT INTO operacao
VALUES     (30,
            'deposito',
            '02/04/2014',
            '10:10:10 PM',
            1234561);
```

```
INSERT INTO transferencia
VALUES     (0,
            0,
            2,
            1,
            10000);

INSERT INTO transferencia
VALUES     (1,
            1,
            0,
            1,
            1000);

INSERT INTO transferencia
VALUES     (2,
            2,
            0,
            1,
            20000);

INSERT INTO transferencia
VALUES     (3,
            3,
            0,
            1,
            3000000);

INSERT INTO transferencia
VALUES     (4,
            4,
            9,
            1,
            1);

INSERT INTO transferencia
VALUES     (5,
            5,
            3,
            1,
            10);

INSERT INTO transferencia
VALUES     (6,
            6,
            4,
            1,
            1000);

INSERT INTO transferencia
VALUES     (7,
            7,
            0,
            1,
            11000);
```

```
INSERT INTO transferencia
VALUES     (8,
            8,
            0,
            1,
            12000);

INSERT INTO transferencia
VALUES     (9,
            9,
            9,
            1,
            13000);

INSERT INTO levantamento
VALUES     (10,
            0,
            100,
            0);

INSERT INTO levantamento
VALUES     (11,
            1,
            1000,
            1);

INSERT INTO levantamento
VALUES     (12,
            2,
            10000,
            2);

INSERT INTO levantamento
VALUES     (13,
            3,
            10010,
            3);

INSERT INTO levantamento
VALUES     (14,
            4,
            2000,
            4);

INSERT INTO levantamento
VALUES     (15,
            5,
            100,
            5);

INSERT INTO levantamento
VALUES     (16,
            6,
            1,
            6);
```

```
INSERT INTO levantamento
VALUES     (17,
            7,
            500,
            7);

INSERT INTO levantamento
VALUES     (18,
            8,
            5000,
            8);

INSERT INTO levantamento
VALUES     (19,
            9,
            1000000000,
            9);

INSERT INTO emprestimo
VALUES     (20,
            0,
            1,
            0,
            10000,
            'Casa');

INSERT INTO emprestimo
VALUES     (21,
            1,
            10,
            1,
            1000,
            'Carro');

INSERT INTO emprestimo
VALUES     (22,
            2,
            1,
            2,
            1000000,
            'Casa');

INSERT INTO emprestimo
VALUES     (23,
            3,
            10,
            3,
            10000,
            'Carro');

INSERT INTO emprestimo
VALUES     (24,
            4,
            2,
            4,
            10000,
```

```
            'Universidade');

INSERT INTO emprestimo
VALUES     (25,
            5,
            1,
            5,
            10000,
            'Casa');

INSERT INTO emprestimo
VALUES     (26,
            6,
            1,
            6,
            100000,
            'Casa');

INSERT INTO emprestimo
VALUES     (27,
            7,
            2,
            7,
            100000,
            'Universidade');

INSERT INTO emprestimo
VALUES     (28,
            8,
            20,
            8,
            500,
            'Viagem');

INSERT INTO emprestimo
VALUES     (29,
            9,
            50,
            9,
            1000000,
            'Casa de férias');

INSERT INTO deposito
VALUES     (30,
            0,
            0,
            100);

INSERT INTO associa
VALUES     (0,
            0);

INSERT INTO associa
VALUES     (1,
            1);
```

```
INSERT INTO associa
VALUES     (0,
            2);

INSERT INTO associa
VALUES     (0,
            3);

INSERT INTO associa
VALUES     (9,
            4);

INSERT INTO associa
VALUES     (3,
            5);

INSERT INTO associa
VALUES     (4,
            6);

INSERT INTO associa
VALUES     (0,
            7);

INSERT INTO associa
VALUES     (0,
            8);

INSERT INTO associa
VALUES     (9,
            9);

INSERT INTO associa
VALUES     (0,
            10);

INSERT INTO associa
VALUES     (1,
            11);

INSERT INTO associa
VALUES     (2,
            12);

INSERT INTO associa
VALUES     (3,
            13);

INSERT INTO associa
VALUES     (4,
            14);

INSERT INTO associa
VALUES     (5,
            15);
```

```
INSERT INTO associa
VALUES     (6,
            16);

INSERT INTO associa
VALUES     (7,
            17);

INSERT INTO associa
VALUES     (8,
            18);

INSERT INTO associa
VALUES     (9,
            19);

INSERT INTO associa
VALUES     (0,
            20);

INSERT INTO associa
VALUES     (1,
            21);

INSERT INTO associa
VALUES     (2,
            22);

INSERT INTO associa
VALUES     (3,
            23);

INSERT INTO associa
VALUES     (4,
            24);

INSERT INTO associa
VALUES     (5,
            25);

INSERT INTO associa
VALUES     (6,
            26);

INSERT INTO associa
VALUES     (7,
            27);

INSERT INTO associa
VALUES     (8,
            28);

INSERT INTO associa
VALUES     (9,
            29);
```

```
INSERT INTO tem
VALUES     (0,
            1234567);

INSERT INTO tem
VALUES     (1,
            1234566);

INSERT INTO tem
VALUES     (2,
            1234565);

INSERT INTO tem
VALUES     (3,
            1234564);

INSERT INTO tem
VALUES     (4,
            1234563);

INSERT INTO tem
VALUES     (5,
            1234568);

INSERT INTO tem
VALUES     (6,
            1234569);

INSERT INTO tem
VALUES     (7,
            1234570);

INSERT INTO tem
VALUES     (8,
            1234571);

INSERT INTO tem
VALUES     (9,
            1234572);

INSERT INTO solicitacao
VALUES     (3,
            1234564);

INSERT INTO solicitacao
VALUES     (4,
            1234563);

INSERT INTO solicitacao
VALUES      (30,
             1234567);

INSERT INTO solicitacao
VALUES      (10,
             1234567);
```

```
INSERT INTO solicitacao
VALUES      (11,
             1234566);

INSERT INTO solicitacao
VALUES      (12,
             1234565);

INSERT INTO solicitacao
VALUES     (20,
            1234567);

INSERT INTO solicitacao
VALUES     (21,
            1234566);

INSERT INTO solicitacao
VALUES     (22,
            1234565);

INSERT INTO solicitacao
VALUES     (23,
            1234564);

INSERT INTO solicitacao
VALUES     (1,
            1234566);

INSERT INTO solicitacao
VALUES     (2,
            1234567);

INSERT INTO solicitacao
VALUES     (5,
            1234564);

INSERT INTO solicitacao
VALUES     (6,
            1234563);

INSERT INTO solicitacao
VALUES     (7,
            1234567);

INSERT INTO solicitacao
VALUES     (8,
            1234567);

INSERT INTO solicitacao
VALUES     (9,
            1234572);

INSERT INTO solicitacao
VALUES     (24,
            1234563);
```

```
INSERT INTO solicitacao
VALUES     (25,
            1234568);

INSERT INTO solicitacao
VALUES     (26,
            1234569);

INSERT INTO solicitacao
VALUES     (27,
            1234570);

INSERT INTO solicitacao
VALUES     (28,
            1234571);

INSERT INTO solicitacao
VALUES     (29,
            1234572);

INSERT INTO solicitacao
VALUES     (13,
            1234564);

INSERT INTO solicitacao
VALUES     (14,
            1234563);

INSERT INTO solicitacao
VALUES     (15,
            1234568);

INSERT INTO solicitacao
VALUES     (16,
            1234569);

INSERT INTO solicitacao
VALUES     (17,
            1234570);

INSERT INTO solicitacao
VALUES     (18,
            1234571);

INSERT INTO solicitacao
VALUES     (19,
            1234572);
```