6
Referências

[1] SANDRI. A.; Detecção e Prevenção de Fraudes utilizando Inteligência Artificial. 2006. Disponível em <http://br.geocities.com/andresandri/artigos/IA/Deteccao_Prevencao_Fraude_IA.doc> Acesso em Junho de 2008.

[2] PASSINI, S.R.R; TOLEDO, C.M.T; Mineração de Dados para Detecção de Fraudes em Ligações de Água. 2002. Disponível em <http://www.inf.furb.br/seminco/2002/artigos/Passini-seminco2002-6.pdf> Acesso em Junho de 2008.

[3] RABELO, E;. Avaliação de Técnicas de Visualização para Mineração de Dados, 2007. Dissertação de Pós-Graduação, Universidade Federal de Maringá. Disponível em <http://www.din.uem.br/pos-graduacao/mestrado-em-ciencia-da-computacao/dissertacoes> Acesso em Junho 2008.

[4] Fayyad, U.; Shapiro, P. G.; Smyth, P.; Knowledge Discovery and Data Mining: Towards a Unifying Framework. Menlo Park, Calif.: AAAI Press; Cambridge, Mass.: MIT Press, 1996. 611p.

[5] FERRO, M., LEE, H. D.; O Processo de KDD – Knowledge Discovery in Database para Aplicações na Medicina, 2001.

[6] ROMÃO, W.; Descoberta de Conhecimento Relevante em Banco de Dados sobre Ciência e Tecnologia. Dissertação de Pós-Graduação, Universidade Federal de Santa Catarina, 2002.

[7] Information Drivers Company, Inglaterra. Disponível em <http://www.information-drivers.com/market_basket_analysis.php>. Acesso em julho 2008.

[8] Direct Magazine, “Match 'Em Up” Outubro 2006. Disponível em <http://directmag.com/disciplines/crm/marketing_match_em>, Acesso em julho 2008.

[9] VELOSO, M. S. A. Regras De Associação Aplicadas A Um Método De Apoio Ao Planejamento De Recursos Humanos. Dissertação de Mestrado, Universidade do Porto.

[10] TAN, P., STEINBACH, M., KUMAR, V., Introduction to Data Mining. Person Addison Wesley Education Press Inc. Boston, 2005.

[11] AZEVEDO, L. V. Maximizando o valor do relacionamento com o cliente: Data Mining e CRM. 2003.

[12] SANTOS, J. G.; Mineração De Dados Aplicada à Gestão De Relacionamento com Clientes. Espaço Científico, Santarém V.6, n.1/2, p.29-36, 2005.

[13] NOGUEIRA, C. F.; Metodologia de Valorização de Clientes, utilizando Mineração de Dados. Dissertação de Pós-graduação em Engenharia Civil da Universidade Federal do Rio de Janeiro, 2004.

[14] ADOMAVICIUS, G., TUZHILIN, A.; Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions On Knowledge And Data Engineering, VOL. 17, No. 6, Junho 2005.

[15] BURKE, R; Knowledge-based Recommender Systems. University of California, Irvine.

[16] MIDDLETON, S. E., ROURE, D., SHADBOLT, N.; Captuiring knowledge of User Preferences: Ontologies in Recommender Systems. University of Southampton. October 22-23, 2001, Victoria, British Columbia, Canada.

[17] TAN, P.; STEINBACH, M.; KUMAR, V.; Introduction to Data Mining. Pearson Addison Wesley 2006.

[18] LIU, F.; LU, Z; LU, S., Mining Association Rules Using Clustering, Intelligent Data Analysis,5, 2001, pp.309-326.

[19] CAVIQUE, L. Graph-based structures for the Market Baskets Analysis. 24 2004, pp.233-246.

[20] LIMEIRA, Tânia m. V. Administração das comunicações em Marketing. In: DIAS, S. R. Gestão de marketing. São Paulo: Saraiva, 2003. p. 301.

[21] Bretze Marketing de Relacionamento 2004, Aumentando a Rentabilidade do Cliente com Database Marketing. Disponível em www.bretzke-marketing.com.br/textos/artigos01.htm. Acesso em janeiro de 2009.

[22] HOUTSMA, M., SWAMI, A. Set-oriented mining for association rules in relational databases. Twente Univ., Enschede; Data Engineering, 1995. Proceedings of the Eleventh International Conference on p. 25-33.

[23] DIPANKAR, D., Artificial Immune System and Their Aplications. Springer-Verlag 1999.

[24] DASGUPTA, D., Ji, Z., González, F., Artificial Immune System (AIS) Research in the Last Five Years. IEEE 2003.

[25] AGRAWAL, R., IMIELINSKI, T., SRIKANT, R., Mining Association Rules between Sets of Items in Large Databases, Proc. of the ACM SIGMOD Intl. Conf. on Management of Data, Washington, Estados Unidos, 1993, 207–216.

[26] C. Borgelt, An Implementation of the FP-growth Algorithm, Workshop Open Source Data Mining Software, OSDM'05, Chicago, IL, 1-5.ACM Press, USA, 2005.

[27] S. Brin, R. Motwani, J.D. Ullman and S.Tsur, Dynamic Itemset Counting and Implication Rules for Market Basket Data, in Proceedings of the 1997 ACM SIGMOD Conference, Tucson, Arizona, 1997, pp.255-264.

[28] J. Han, J. Pei, Y. Yin, “Mining frequent patterns without candidate generation”, Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, United States, 1-12, 2000

7
APÊNDICE A – Código da Implementação

Unit Main – Main.pas

```
(*****************************************************************************
*       PONTIFICIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO - PUC-RIO
*
* Programa de Pós-Graduação em Informática - Mestrado
*
* Programa apresentado como requisito parcial para o cumprimento dos créditos da
* cadeira de Projeto Final de Programação (INF 2102).
*
* Aluna: Livia Fonseca Fracalanza
* Matricula : 0621292
* Orientador: Prof. Marco Antonio Casanova
*****************************************************************************)
unit Main;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, RzGrids, RzShellDialogs, StdCtrls, RzLabel, Mask, RzEdit,
  RzBtnEdt, ExtCtrls, LogView, OleCtrls, SHDocVw,
  MBAObject, DB, ADODB, CheckLst, RzLstBox, RzChkLst;

type
  TFMain = class(TForm)
    Panel3: TPanel;
    Splitter1: TSplitter;
    WebBrowser1: TWebBrowser;
    Panel1: TPanel;
    Panel2: TPanel;
    Panel4: TPanel;
    BtnMaximalCliqueByWeight: TButton;
    RzOpenDialog1: TRzOpenDialog;
    BtnGraphGenerator: TButton;
    BtnMaximalCliqueByAverage: TButton;
    RzLabel1: TRzLabel;
    BtnConnect: TButton;
    BtnDisconnect: TButton;
    cbDriversList: TComboBox;
    RzLabel2: TRzLabel;
    cbBases: TComboBox;
    RzLabel3: TRzLabel;
    ODBCConnect: TADOConnection;
    Button1: TButton;
    edtFilter: TEdit;
    Filtro: TLabel;
    Button2: TButton;
    Memo1: TMemo;
    Label1: TLabel;
```

```
    VarListBox: TRzCheckList;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure btnPrintClick(Sender: TObject);
    procedure FileEdtButtonClick(Sender: TObject);
    procedure BtnMaximalCliqueByWeightClick(Sender: TObject);
    procedure BtnGraphGeneratorClick(Sender: TObject);
    procedure BtnMaximalCliqueByAverageClick(Sender: TObject);
    procedure BtnConnectClick(Sender: TObject);
    procedure BtnDisconnectClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    MBAObj : TMBAObj;
    FFilePath : string;
    recset  : _Recordset;

    procedure ReadFile;
    procedure GetDriversList;
    procedure PrintRelations;
    procedure PrintMaximalClique;
    procedure DisconnectODBC;
  public
//    FLogViewer  : TLogViewer;    // Main log info
  end;

var
  FMain: TFMain;

implementation
{$R *.dfm}

uses
//  StrUtils, // for AnsiStr functions
  Registry,
  LoginForm,
  WINGRAPHVIZLib_TLB; // to use GraphViz to draw Graph gif

(*****************************************************************************)
procedure TFMain.FormCreate(Sender: TObject);
begin
  //Create Log Object
  CreateLogObject;
  FLogViewer.LogUpdate (' *** Início da execução do programa *** ');

  GetDriversList;

  VarListBox.MultiSelect := TRUE;
end;
(*****************************************************************************)
procedure TFMain.FormDestroy(Sender: TObject);
begin

  // Destroy Log Object
  FLogViewer.LogUpdate (' *** Encerrando a execução do programa *** ');
  If Assigned (FLogViewer) then
    FreeAndNil (FLogViewer);
end;
(*****************************************************************************)
procedure TFMain.GetDriversList;
```

```
var
  reg   : TRegistry;
  List  : TStrings;
begin
  reg := TRegistry.Create;
  List := TStringList.Create;
  try
    reg.RootKey := HKEY_LOCAL_MACHINE;
    reg.OpenKey ('SOFTWARE\ODBC\ODBC.INI', False);
    reg.GetKeyNames (List);

    cbDriversList.Items.Assign (List);
  finally
    reg.Free;
    List.Free;
  end;
end;
(*****************************************************************************)
(*****************************************************************************)
procedure TFMain.btnPrintClick(Sender: TObject);
begin
  PrintRelations;
end;
(*****************************************************************************)
(*****************************************************************************
* Procedure: PrintRelations
* Descrição: Procedure passa pelos nós da lista de adjacência imprimindo a rela-
            ção existente entre os mesmos e a quantidade de vezes em que essa
            relação acontece.
*
*****************************************************************************)
procedure TFMain.PrintRelations;
var
  i, j : Integer;
  Produto : TProduct;
  PAdj : TProduct;
  ws : string;
begin
//  Memo1.Lines.Clear;

  for i := 0 to MBAObj.FAdjList.Count-1 do
    begin
      ws := '';

      Produto := TProduct(MBAObj.FAdjList.Objects[i]);
      Memo1.Lines.Add(' ********************************** ');
      Memo1.Lines.Add('Produto : ' + MBAObj.FAdjList.Strings[i]);
      Memo1.Lines.Add('Adjacências: ');

      for j := 0 to Produto.List.Count-1 do
        begin
          PAdj := TProduct(Produto.List.Objects[j]);
          ws := ws + PAdj.ID + ': ' + IntToStr(PAdj.Qtd) + ' -> ';
        end;
      Memo1.Lines.Add(Copy(ws,0, length(ws)-3));
    end;
end;
(*****************************************************************************)
procedure TFMain.FileEdtButtonClick(Sender: TObject);
begin
```

```
  if RzOpenDialog1.Execute then
    FFilePath:= RzOpenDialog1.FileName;

//  FileEdt.Text:= FFilePath;

  //Create MBA Object
  if Assigned (MBAObj) then
    MBAObj.Destroy;

  MBAObj := TMBAObj.Create;

  // Read the Transaction File and Set the Adjacent List, constructing the Graph
  ReadFile;

  //Print the relations at the list of adjacencies
  PrintRelations;
end;
(*****************************************************************************)
(*****************************************************************************
* Procedure: ReadFile
* Descrição: Lê cada linha do arquivo texto, representando uma transação e chama
*           a função FillAdjList para computar a relação de adjacência dos
*           ítens da transação.
*
* AE : -
* AS : -
*****************************************************************************)
procedure TFMain.ReadFile;
var
   MatrixFile : TextFile;
   TransactionLine: string;
   PList : TStringList;
   i : integer;
begin
  if not FileExists(FFilePath) then
    begin
      ShowMessage ('O Arquivo informado não existe.');
      FLogViewer.LogUpdate ('Erro: O Arquivo informado não existe.');
      exit;
    end;

  Memo1.Lines.Clear;
  i := 0;
  try
    AssignFile(MatrixFile, FFilePath);
    Reset(MatrixFile);

    Plist := TStringList.Create;
    PList.Delimiter := ';';
    while not Eof(MatrixFile) do
      begin
        Readln(MatrixFile, TransactionLine);

        Memo1.Lines.Add('Lendo Transação ' + IntTOStr(i+1));
        Memo1.Lines.Add(TransactionLine);
        Memo1.Lines.Add(' ---------- ');

        PList.DelimitedText := TransactionLine;

        MBAObj.FillAdjList (PList, 0);
```

```
      inc(i);
    end;
   FLogViewer.LogUpdate (IntToStr(i) + ' transações lidas.');
  finally
   CloseFile(MatrixFile);
//   FreeAndNil (MatrixFile);
  end;
end;
(*****************************************************************************)
(*****************************************************************************
* Procedure: BtnGraphGeneratorClick
* Descrição: Procedure que implementa o clique do botão de criação da imagem do
*           grafo de acordo com as informações da lista de adjacências.
*
*****************************************************************************)
procedure TFMain.BtnGraphGeneratorClick(Sender: TObject);
var
  Dot: IDot;
  Image: IBinaryImage;
  ImageFileName: string;
  Path : string;
  Data : widestring;
  i, j : Integer;
  Vertex, Adjc : TProduct;
begin
  Memo1.Lines.Clear;
  Memo1.Lines.Add('graph G');
  Memo1.Lines.Add('{');
  Memo1.Lines.Add('label=" Grafo de Transações "');
  Memo1.Lines.Add('node [color=pink, style=filled]');

  for i := 0 to MBAObj.FAdjList.Count-1 do
   begin
    Vertex := TProduct(MBAObj.FAdjList.Objects[i]);
    if MBAObj.FMaximumCliqueList.IndexOf(Vertex.ID) >= 0 then // Node is a part of the
Maximal Clique
      begin
       Memo1.Lines.Add(Vertex.ID + ' [color=yellow]');
      end;

    for j := 0 to Vertex.List.Count-1 do
      begin
       Adjc := TProduct(Vertex.List.Objects[j]);
       if Vertex.ID < Adjc.ID then
         begin
          Memo1.Lines.Add(Vertex.ID + ' -- ' + Adjc.ID + ' [label = ' + IntToStr(Adjc.Qtd) +
', color=green]');
         end;
      end;
   end;
  Memo1.Lines.Add('}');

// Print the Graph
  Data := Memo1.text;
  Path := ExtractFilePath(paramStr(0));
  ImageFileName := Path + 'Graph_image.gif';

  Dot := CoDOT.Create;
  Image := Dot.ToGIF(Data);
```

```
  Image.Save(ImageFileName);
  WebBrowser1.Navigate('file:///' + Path + 'Graph_image.gif');
end;
(*****************************************************************************)
procedure TFMain.BtnMaximalCliqueByWeightClick(Sender: TObject);
begin
  MBAObj.FindTheMaximalClique (M_WEIGHT);

  PrintMaximalClique;
end;
(*****************************************************************************)
procedure TFMain.BtnMaximalCliqueByAverageClick(Sender: TObject);
begin
  MBAObj.FindTheMaximalClique (AVERAGE);

  PrintMaximalClique;
end;
(*****************************************************************************)
procedure TFMain.PrintMaximalClique;
var
  i : Integer;
begin
  // Se foram identificados nós que formam uma clique máxima, imprime a mesma
  if MBAObj.FMaximumCliqueList.Count > 0 then
  begin
    Memo1.Lines.Add (' Clique máxima pelo peso total: ');
    for i := 0 to MBAObj.FMaximumCliqueList.Count -1 do
      begin
        Memo1.Lines.Add(' - '+ MBAObj.FMaximumCliqueList[i]);
      end;
    Memo1.Lines.Add(FloatToStr(MBAObj.FMaxCliqueWeight));
  end
  else
    Memo1.Lines.Add(' -> Não existe uma Clique.');
end;
(*****************************************************************************)
procedure TFMain.BtnConnectClick(Sender: TObject);
var
  ConnectStr,
  DriverName : string;
  List       : TStrings;
begin
  if cbDriversList.ItemIndex < 0 then
    begin
      ShowMessage ('Por favor, selecione um Driver ODBC!');
      exit;
    end;

  if FrmLogin.ShowModal <> mrOK then
    exit;

  DriverName := cbDriversList.Text;
  ConnectStr := 'Provider=MSDASQL.1;Persist Security Info=False;Data Source='
              + DriverName;
  (* connect *)
  try
    ODBCConnect.ConnectionString := ConnectStr;
    ODBCConnect.Open (FrmLogin.UserName, FrmLogin.Password); //  ('ADMCAMP',
'THORAN'); //
```

```
    List := TStringList.Create;
    try
      if ODBCConnect.Connected then
        ShowMessage ('Conectado!')
      else
        begin
          ShowMessage ('Conexão falhou!');
          Exit;
        end;

      ODBCConnect.GetTableNames (List, False);
      cbBases.Items.Assign (List);
    finally
      List.Free;
    end;

      BtnConnect.Enabled := false;
//      Button1.Enabled := true;
      BtnDisconnect.Enabled := true;
  except
    on E : EADOError do
      begin
        ShowMessage (E.Message);
        DisconnectODBC;
      end;
    on E : Exception do
      begin
        ShowMessage ('Conexão Simba falhou : ' + E.Message);
        DisconnectODBC;
      end;
  end;
end;
(*****************************************************************************)
procedure TFMain.BtnDisconnectClick(Sender: TObject);
begin
  DisconnectODBC;
end;
(*****************************************************************************)
procedure TFMain.DisconnectODBC;
begin
  ODBCConnect.Close;
  BtnConnect.Enabled := true;
//  Button1.Enabled := false;
  BtnDisconnect.Enabled := false;
  VarListBox.Clear;
end;
(*****************************************************************************)
procedure TFMain.Button1Click(Sender: TObject);
var
  List       : TStrings;
begin
  if not ODBCConnect.Connected
  then
    begin
      ShowMessage ('Dataset desconectado!');
      Exit;
    end;

  if cbBases.ItemIndex < 0
  then
```

```
    begin
      ShowMessage ('Por favor, selecione um dataset!');
      Exit;
    end;

  List := TStringList.Create;
  VarListBox.Clear;

  try
    ODBCConnect.GetFieldNames (cbBases.Text, List);
    VarListBox.Items.Assign (List);
  finally
    List.Free;
  end;
end;

procedure TFMain.Button2Click(Sender: TObject);
var
  queryStr,
  DataSetName,
  VarListStr    : String;
  Ix, Jx        : Integer;
  SelList       : Array of String;
  SelCount      : Integer;
  timestep      : DWORD;
  FilterStr     : String;
  FullRecovery  : boolean;
  columnsList   : array of string;
  RecLine       : TStringList;
  auxStr        : String;
begin
  if cbBases.ItemIndex < 0 then
    begin
      ShowMessage ('Por favor, selecione um dataset!');
      Exit;
    end;

  SelCount := VarListBox.ItemsChecked;
  if SelCount = 0 then
    begin
      ShowMessage ('Por favor, selecione pelo menos uma variável!');
      Exit;
    end;

  SetLength (SelList, SelCount);
  SetLength (columnsList, SelCount);
  Jx := 0;

  DataSetName := cbBases.Text;
  with VarListBox do
    begin
      for Ix := 0 to Count - 1 do
        begin
          if ItemChecked[Ix] then
            begin
              SelList[Jx] := Items.Strings[Ix];

              if VarListStr <> '' then
                VarListStr := VarListStr + ', ';
```

```
          VarListStr := VarListStr + #34 + SelList[Jx] + #34;
          Inc (Jx);
        end;
      end;
  end;
 FilterStr := '';

 if edtFilter.Text <> '' then
   FilterStr := ' WHERE ' + edtFilter.Text;

 queryStr := 'SELECT ' + VarListStr + ' FROM "' + DataSetName + '"' + FilterStr;
 Memo1.Lines.Add('Query: ' + queryStr);
 try
   timestep := GetTickCount();
   recset := ODBCConnect.Execute (queryStr);
   timestep := GetTickCount() - timestep;
   Memo1.Lines.Add('Query execution time = ms ' + IntToStr (timestep)
             + #13 + #10
             + 'Total de registros: ' + IntToStr(RECSET.RecordCount));
 except
   on E : EADOError do
     begin
       ShowMessage (E.Message);
       Exit;
     end;
   on E2 : Exception do
     begin
       ShowMessage ('Query failed : ' + E2.Message);
       Exit;
     end;
 end;

 try
   for Jx := 0 to SelCount - 1 do
     columnsList [Jx] := SelList[Jx];

   Jx := 0;

   RecLine := TStringList.Create;
   RecLine.Delimiter := ';';

   // Create MBA Object
   if Assigned (MBAObj) then
     MBAObj.Destroy;

   MBAObj := TMBAObj.Create;

   Memo1.Lines.add ('Lendo os registros... Por favor, aguarde...');
   timestep := GetTickCount();
   repeat
     Inc (Jx);
     auxStr := '';

     for Ix := 0 to recset.Fields.Count - 1 do
       begin
         auxStr := auxStr + columnsList [Ix] + '_' + trim(recset.Fields[Ix].Value) + ';';
       end;

     RecLine.DelimitedText := Copy(auxStr, 1, length(auxStr)-1);
```

```
    MBAObj.FillAdjList (RecLine, 0);

    recset.Move (1, EmptyParam);
   until recset.EOF;
  except
   on E : Exception do
    begin
     ShowMessage ('Load result exception : ' + E.Message);
    end;
  end;
  timestep := GetTickCount() - timestep;
  Memo1.Lines.add ('Cálculo das relações durou ' + IntToStr (timestep) + 'ms ...');
  // ShowMessage ('Result Count = ' + inttostr(Jx));
  PrintRelations;
end;

end.
```

Unit MBAObject – MBAObject.pas

```
(*******************************************************************************
*       PONTIFICIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO - PUC-RIO
*
* Programa de Pós-Graduação em Informática - Mestrado
*
* Programa apresentado como requisito parcial para o cumprimento dos créditos da
* cadeira de Projeto Final de Programação (INF 2102).
*
* Aluna: Livia Fonseca Fracalanza
* Matricula : 0621292
* Orientador: Prof. Marco Antonio Casanova
*******************************************************************************)
unit MBAObject;

interface

uses
  SysUtils,  Variants, Classes;

const
  M_WEIGHT = 1;
  AVERAGE = 2;

const
  INITIALWEIGHT = -999999999;

type
  TProduct = class(TObject)
    ID  : string;
    Qtd : Integer;
    List : TStringList;
  private
    //
  public
    constructor Create;
    destructor Destroy;
  end;

type
```

```
  TMBAObj = class
    private
      AdjList          : TStringList; // Has the relations between the elements of all the
transactions
      MaximumCliqueList : TStringList; // Represent the Maximum Clique found
      MaxCWt           : Double;

      function DestroyAdjList : Integer;
      function getResultAdjList: TStringList;
      function getMaximumCliqueList: TStringList;
    public
      constructor Create;
      destructor Destroy;

      procedure FillAdjList (Line : TStringList; Start : Integer);
      function Find_Clique(var Clique: TStringList; var Weight: Double; Criteria: Integer):
boolean;
      function FindTheMaximalClique(Criteria: Integer): boolean;

      property FAdjList : TStringList read getResultAdjList;
      property FMaximumCliqueList : TStringList read getMaximumCliqueList;
      property FMaxCliqueWeight : double read MaxCWt;
    end;

implementation
(*****************************************************************************)
constructor TMBAObj.Create;
begin
  // Initialize List of Adjacencies
  AdjList := TStringList.Create;

  //Initialize the List that will have the Maximal Clique
  MaximumCliqueList := TStringList.Create;
end;
(*****************************************************************************)
destructor TMBAObj.Destroy;
begin
  if Assigned(AdjList) then
    DestroyAdjList;

  if Assigned(MaximumCliqueList) then
    FreeAndNil (MaximumCliqueList);
end;
(*****************************************************************************)
function TMBAObj.DestroyAdjList : Integer;
var
  i : Integer;
  prod : TProduct;
begin
  if Assigned (AdjList) then
    begin
      for i := AdjList.Count-1 downto 0 do
        begin
          if Assigned (AdjList.Objects[i]) then
            begin
              prod := TProduct(AdjList.Objects[i]);
              FreeAndNil(prod);
            end;
        end;
      FreeAndNil (AdjList);
```

```
   end;
end;
(*****************************************************************************)
(*****************************************************************************
* Procedure: FillAdjList
* Descrição: É responsável pelo preenchimento da lista global de adjacências,
*           fazendo as combinações de relacionamento entre os elementos de uma
*           mesma transação.
* Parâmetros:
*   Line  - StringList com cada produto lido na linha da transação em uma posição
*   Start - Inteiro que indica a posição de início da leitura da lista.
*           Importante para a recursão.
*
* AE : Line.Count <> 0
* AS : AdjProd.Count > 0
*****************************************************************************)
procedure TMBAObj.FillAdjList (Line : TStringList; Start : Integer);
var
  i : integer;
  Pix, AIx : integer;
  Produto1, Produto2 : TProduct;
  AdjProd : TProduct;
begin
  if (Line.Count -1) = Start then
    exit;

  Produto1 := TProduct.Create;
  Produto1.ID := Line[Start];
  Produto1.Qtd := 1;

  // Do the items combinations
  for i := Start+1 to Line.Count-1 do
    begin
      Produto2 := TProduct.Create;
      Produto2.ID  := Line[i];
      Produto2.Qtd := 1;

      // Add the product 1 to the Transactions Adj. List
      PIx := AdjList.IndexOf(Produto1.ID);
      if PIx < 0 then // is not at the adj. list
        begin
          //Memo1.Lines.ADD(' -> Adicionando à adj. List ' + Produto1.ID);
          PIx := AdjList.AddObject(Produto1.ID, Produto1);
        end;

      AdjProd := TProduct(AdjList.Objects[PIx]);

      AIx := AdjProd.List.IndexOf(Produto2.ID);
      if AIx < 0 then
        begin
          //Memo1.Lines.ADD(' --> Adicionando à Lista de ' + Produto1.ID + ' o produto ' +
Produto2.ID);
          AdjProd.List.AddObject(Produto2.ID, Produto2);
        end
        else
          TProduct(AdjProd.List.Objects[AIx]).Qtd := TProduct(AdjProd.List.Objects[AIx]).Qtd
+ 1;

      // Add the product 2 to the Transactions Adj. List
      PIx := AdjList.IndexOf(Produto2.ID);
```

```
    if PIx < 0 then // is not at the adj. list
      begin
        //Memo1.Lines.ADD(' -> Adicionando à adj. List ' + Produto2.ID);
        PIx := AdjList.AddObject(Produto2.ID, Produto2);
      end;

    AdjProd := TProduct(AdjList.Objects[PIx]);

    AIx := AdjProd.List.IndexOf(Produto1.ID);
    if AIx < 0 then
      begin
        //Memo1.Lines.ADD(' --> Adicionando à Lista de ' + Produto2.ID + ' o produto ' +
Produto1.ID);
        AdjProd.List.AddObject(Produto1.ID, Produto1);
      end
      else
        TProduct(AdjProd.List.Objects[AIx]).Qtd := TProduct(AdjProd.List.Objects[AIx]).Qtd
+ 1;
   end;

  FillAdjList (Line, Start+1);
end;
(*****************************************************************************)
(*****************************************************************************
* Função: FindTheMaximalClique
* Descrição: Função faz chamadas a função Find_clique para cada nó da lista de
*           adjacências. A clique representada pela variável LocalClique, começa
*           com um elemento da lista de adjacências (elemento da vez na iteração)
*           e com peso negativo (representado pela variável CliqueWeight). A
*           variável MaxCliqueWeight guardará o valor da clique mais pesada de
*           acordo com o critário escolhido, sendo inicializado, também, com va-
*           lor negativo.
*           Ao final do processamento, a clique encontrada é impressa, caso con-
*           trário uma mensagem de que não foi possível encontrar uma clique é
*           exibida.
*
* Parâmetros:
*    Criteria - Inteiro que identifica qual critério será utilizado para o cálculo
              da clique máxima. Assume os seguintes valores:
                - M_WEIGHT = 1; Clique máxima pelo maior peso de relações
                - AVERAGE = 2; Clique máxima pelo peso das relações dividido
                          pelo número de nós pertencentes a clique
*
* AE : -
* AS : -
*****************************************************************************)
function TMBAObj.FindTheMaximalClique (Criteria : Integer) : boolean;
var
 i : Integer;
 MaxCliqueWeight : Double;
 LocalClique : TStringList;
 CliqueWeight : Double;
begin
  MaximumCliqueList.Clear;

  try
    LocalClique := TStringList.Create;
    MaxCliqueWeight := INITIALWEIGHT;

    for i := 0 to AdjList.Count-1 do
```

```
    begin
      LocalClique.Clear;
      CliqueWeight := INITIALWEIGHT;

      LocalClique.Add(TProduct(AdjList.Objects[i]).ID);

      if Find_Clique (LocalClique, CliqueWeight, Criteria) then
        begin
          if CliqueWeight >= MaxCliqueWeight then
            begin
              MaximumCliqueList.Assign(LocalClique);
              MaxCliqueWeight := CliqueWeight;
            end;
        end;
    end;

    MaxCWt := MaxCliqueWeight;
  finally
    FreeAndNil(LocalClique);
  end;
end;
(*****************************************************************************)
(*****************************************************************************
* Função: Find_Clique
* Descrição: Função responsável por encontrar a clique máxima do grafo represen-
*           tado pela lista de adjacências. O cálculo depende do critério esco-
*           lhido. Se for máximo peso (M_WEIGHT) então o peso da clique é o so-
*           somatório dos pesos das relações entre os elementos. No caso do crité-
*           rio de média (AVERAGE), o peso da clique é determinado pelo somatório
*           dos pesos das relações entre os elementos dividido pelo número de
*           nós adicionados na clique.
*
* Parâmetros:
*   Clique - TStringList
*   Weight - Valor Double
*   Criteria - Inteiro que identifica qual critério será utilizado para o cálculo
*              da clique máxima. Assume os seguintes valores:
*                 - M_WEIGHT = 1; Clique máxima pelo maior peso de relações
*                 - AVERAGE = 2; Clique máxima pelo peso das relações dividido
*                             pelo número de nós pertencentes a clique
*
* AE : -
* AS : -
*****************************************************************************)
function TMBAObj.Find_Clique (var Clique : TStringList; var Weight : Double; Criteria :
Integer) : boolean;
var
  i, j : Integer;
  Vertex : TProduct;
  VIx : Integer;
  FirstName : string;
begin
  FirstName := Clique.Strings[0];

  for i := AdjList.Count-1 downto 0 do
    begin
      if TProduct (AdjList.Objects[i]).ID = FirstName then
        continue;

      Vertex := TProduct (AdjList.Objects[i]);
```

```
    for j := Clique.Count-1 downto 0 do
      begin
        VIx := Vertex.List.IndexOf(Clique[j]);

        {Se não é vizinho de algum dos nós já existentes na clique, então esta
        não pode ser uma clique. A iteração é interrompida. }
        if VIx < 0 then
          begin
            Result := FALSE; //Não encontrou uma clique
            break;
          end
          else
            begin
              if Clique.IndexOf(Vertex.ID) < 0 then
                begin
                  Result := TRUE;

                  Clique.Add(Vertex.ID);
                  if Weight = INITIALWEIGHT then //primeiro nó a entrar na lista
                    Weight := Vertex.Qtd
                    else
                      begin
                        if Criteria = M_WEIGHT then
                          Weight := Weight + Vertex.Qtd
                          else
                            Weight := (Weight + Vertex.Qtd)/Clique.Count;
                      end;
                end;
            end;
      end;
  end;
end;
(*****************************************************************************)
function TMBAObj.getResultAdjList: TStringList;
begin
  Result:= AdjList;
end;
(*****************************************************************************)
function TMBAObj.getMaximumCliqueList: TStringList;
begin
  Result:= MaximumCliqueList;
end;
(*****************************************************************************)
(************************* Methods of TProduct Object *************************)
constructor TProduct.Create;
begin
  List := TStringList.Create;
end;
(*****************************************************************************)
destructor TProduct.Destroy;
begin
  If Assigned(List) then
    FreeAndNil (List);
end;
(*****************************************************************************)
end.
```