UNIVERSIDADE FEDERAL DE JUIZ DE FORA

Departamento de Ciência da Computação

Acoplamento de Tarefas de Mineração de Dados em Sistemas de Gerenciamento de Banco de Dados

Victor Marques de Assis

Juiz de Fora, MG
Julho de 2008

Acoplamento de Tarefas de Mineração de Dados em Sistemas de Gerenciamento de Banco de Dados

Victor Marques de Assis

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharel em Ciência da Computação

Orientador: Prof. Custódio Gouvêa Lopes da Motta

Juiz de Fora, MG
Julho de 2008

Acoplamento de Tarefas de Mineração de Dados em Sistemas de Gerenciamento de Banco de Dados

Victor Marques de Assis

Monografia submetida ao corpo docente do Departamento de Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora, como parte integrante dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Aprovada pela banca constituída pelos seguintes professores:

Prof. Custódio Gouvêa Lopes da Motta - orientador
Mestre em Engenharia Civil / Sistemas Computacionais, COPPE/UFRJ

Prof. Carlos Cristiano Hasenclever Borges
Doutor em Engenharia Civil, COPPE/UFRJ

Prof. Raul Fonseca Neto
Pós-Doutor em Modelagem Computacional, LNCC/MCT

Juiz de Fora, MG
Julho de 2008

AGRADECIMENTOS

A minha linda esposa Dani, por tudo...
A minha família e amigos, por tudo que representam em minha vida,

Ao professor Custódio, pelas orientações no decorrer do desenvolvimento desse trabalho,
Aos professores Carlos Cristiano e Raul Fonseca, por aceitarem fazer parte dessa banca,

Ao LNCC, especialmente na figura do Professor Cristiano,
Ao CNPQ, pela bolsa de iniciação científica.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

O presente trabalho tem por objetivo apresentar de maneira breve e direta métodos de acoplamento de tarefas de mineração de dados em Sistemas de Gerenciamento de Bancos de Dados (SGBD).

Será apresentada duas técnicas distintas de implementação de novas funcionalidades ao SQL Server 2005, SGBD mais popular da Microsoft.

Será exposta, também, uma breve explicação sobre mineração de dados bem como a apresentação do algoritmo escolhido para implementação ao final do trabalho, o *k*-Means, um algoritmo de agrupamento simples e prático dentre os mais conhecidos.

Complementando a composição dessa monografia, será mostrada uma abordagem a cerca do estado da arte de dois dos mais usuais Sistemas de Gerenciamento de Bancos de Dados, no que diz respeito aos recursos disponíveis e possíveis para mineração.

Serão mostradas as principais semelhanças entre os softwares mais populares de duas dentre as mais importantes empresas do segmento de banco de dados, Oracle e Microsoft.

CAPÍTULO 01

INTRODUÇÃO

A partir da década de 1990, pesquisas relativas ao processo de Descoberta de Conhecimento em Bancos de Dados (*Knowledge Discovery in Databases* - KDD) ganharam rápido crescimento, motivadas pela evolução da tecnologia e pela ampliação crescente das áreas de aplicações.

A principal etapa do processo de KDD é chamada de Mineração de Dados (*Data Mining* - DM) e constitui-se de um campo de pesquisa multidisciplinar envolvendo desde técnicas de estatística até computação de alto desempenho. A mineração é responsável pela Descoberta de Conhecimento propriamente dita.

Existem hoje, diversos métodos de mineração de dados distintos, cada um deles atendendo a uma funcionalidade específica. A necessidade do usuário é sempre ter disponível um método adequado a sua aplicação.

Diante desta realidade, desenvolvedores de Sistemas de Gerenciamento de Banco de Dados (SGBD) vêm, cada vez mais, incluindo tarefas de DM e/ou permitindo o acoplamento de tarefas ainda não implementadas em seus softwares.

O estudo e a apresentação de métodos de acoplamento de novas funcionalidades de DM aos SGBD constituem o principal objetivo dessa monografia.

1.1 ORGANIZAÇÃO DO TRABALHO

Esse trabalho apresenta, no segundo capítulo, as principais etapas do processo de KDD, exibe as motivações que levaram a origem de uma forma alternativa de abordagem da análise de dados, define *Data Mining* e introduz um algoritmo de agrupamento, que será implementado posteriormente.

Uma apresentação do estado da arte de dois dos mais usuais SGBD, no que diz respeito aos recursos disponíveis e possíveis para mineração de dados, constitui o objetivo principal do terceiro capítulo.

Os SGBD's escolhidos para análise foram o Oracle 10g, da Oracle, pelo fato dessa empresa ser a atual líder do mercado e o Microsoft SQL Server 2005, da Microsoft, devido sua importância no cenário mundial de desenvolvimento de software.

Por fim, o quarto capítulo apresenta duas tentativas de acoplamento de novas funcionalidades ao SGBD escolhido, o SQL Server 2005, uma ferramenta popular e robusta, na forma de implementação do algoritmo K-means.

Serão apresentadas as seguintes tentativas de acoplamento:

i) utilizando os mecanismos de extensão do Microsoft Visual Studio 2005, integrando-o ao *Analysis Services* e,

ii) utilizando *Stored Procedure* executadas diretamente no Servidor SQL Server 2005.

O Microsoft Visual Studio 2005 é um pacote de programas da Microsoft, para desenvolvimento de Software, especialmente dedicado ao framework .NET e às linguagens Visual Basic (VB), C , C++ (C Plus Plus), C# (C Sharp) e J# (Jey Sharp).

O Microsoft Analysis Services é uma ferramenta que oferece um conjunto de serviços, incluindo métodos de *Data Mining*, e é fornecida juntamente com o SQL Server 2005.

1.3 REVISÃO BIBLIOGRÁFICA

O desenvolvimento do trabalho foi iniciado pelo estudo do livro "*Data Mining with SQL Server 2005*" de ZhaoHui Tang e Jamie MacLennan.

Por apresentar uma forma prática e eficiente de trabalho com os algoritmos de Data Mining do Microsoft SQL Server 2005 e disponibilizar ao leitor uma abordagem para o desenvolvimento de funcionalidades adicionais às já existentes, este livro foi utilizado como referência para o desenvolvimento da monografia.

Porém, o livro mostrou-se um pouco deficiente em relação aos conceitos de mineração de dados apresentados.

Optou-se, então, pela realização de um estudo do livro "*Discovering Knowledge in Data: An Introduction to Data Mining*" de Daniel T. Larose, consultas ao clássico "*Data Mining*" de Han e Kamber e consultas às notas de aula da disciplina "Tópicos em Computação Científica I (*Data Mining*)" do professor Custódio Motta, de modo a solidificar um melhor embasamento teórico sobre o assunto.

A seguir, o trabalho continuou com a análise das referências:

"*SQL Server 2005 Books Online - Working with Data Mining*", disponível em http://msdn2.microsoft.com/en-us/library/ms174861.aspx,

"*SQL Server 2005 Books Online - SQL Server Integration Services*", disponível em http://msdn2.microsoft.com/en-us/library/ms141026.aspx,

"*SQL Server 2005 Books Online - Working with Data Sources (Analysis Services)*",disponível em http://msdn2.microsoft.com/en-us/library/ms175608.aspx.

A leitura destes livros revelou-se necessária pela dificuldade de trabalho com a ferramenta *Analysis Services*, componente do SQL Server, devido à complexidade de realização de algumas tarefas disponibilizadas pela ferramenta.

Atividades como a validação de modelos *Data Mining* e o trabalho com DMX (*Data Mining Extensions*), relevaram-se muito complexas de serem entendidas e realizadas apenas mediante o estudo do livro de referência, visto que ele apresentava pouca e dispersas informações referentes a dúvidas que surgiram em relação ao funcionamento de algumas particularidades.

Após o estudo destas referências, houve ainda a necessidade da aquisição de uma compreensão mais profunda do SQL Server, de modo a proporcionar uma melhor segurança na manipulação desta ferramenta.

Esta informação foi obtida mediante consultas ao livro "*Programming SQL Server 2005*" de Bill Hamilton que ofereceu, principalmente, uma forma de simplificar algumas tarefas e também pelo estudo do livro "SQL Server 2000", de Renata Leão e João C Silva, que apresenta de forma sucinta os comandos deste SGBD que possuem similaridade com os comandos do SQL Server 2005.

Pelo fato de alguns tutoriais e a maioria dos exemplos dos livros considerados serem implementados na linguagem C#, a realização de um estudo do livro "*C# 2005 Programmer's Reference*" de Adrian Kingsley-Hughes, Kathie Kingsley- Hughes, do tutorial "*C# Station Tutorial*", disponível em www.csharp- station.com/Tutorial.aspx e o estudo de alguns artigos sobre C# disponíveis em www.csharpbr.com.br, mostrou-se fundamental para reforçar os conhecimentos básicos a cerca da linguagem.

CAPÍTULO 02

DATA MINING – MINERAÇÃO DE DADOS

Necessity is the mother of invention. (*English proverb*) –

Necessidade é a mãe da invenção. (Provérbio Inglês)

2.1 MOTIVAÇÕES

A máxima popular transcrita acima, citada no capítulo introdutório de *Data Mining: Concepts and Techniques* (Han & Kamber, 2001), descreve uma das principais motivações ao estudo e desenvolvimento de uma forma alternativa de abordagem da análise de grande quantidade de dados: *Data mining* ou Mineração de Dados.

A cada dia várias empresas concentram grandes porções de novas informações em suas bases de dados.

A cada momento institutos de pesquisa registram em seus bancos de dados uma considerável quantidade de novas informações relacionadas a seus mais promissores projetos em curso.

Exemplos desses tipos de empresas são abundantes, por exemplo, na web, principalmente na figura dos grandes portais de e-commerce, que registram a todo instante em seus bancos de dados, muitas informações relacionadas a seus clientes.

O mapeamento do genoma humano e de outros organismos por institutos de pesquisa gera diariamente um elevado volume de informações que são sistematicamente armazenadas em bancos de dados, sendo estas informações fontes de estudo para a biologia e medicina através da bioinformática (Wieczorek & Leal, 2002).

Se por um lado muitas informações têm sido constantemente geradas e/ou coletadas, o que dizer a respeito do conhecimento contido nesses dados?

2.2 ERA DO CONHECIMENTO

John Naisbitt - consagrado especialista na previsão de tendências globais - sabiamente observou que "Estamos nos afogando em informação, mas sedentos por conhecimento" (Naisbitt,1986).

Em "*Knowledge discovery in databases: an overview"*, Frawley et al apontam o fato de que "Computadores nos prometeram uma fonte de sabedoria, mas nos entregaram uma torrente de dados" (Frawley et al. 1991)

Toda essa informação gerada e armazenada pode e deve contribuir para a geração de conhecimento, sugerindo tendências e apresentando particularidades visando uma rápida, produtiva e eficiente ação de seus gestores. (Fernandes, 2006)

É fato que nossa memória possui pouca utilidade quanto desvinculada do uso da inteligência, pois esta permite, a partir da análise da memória, o estabelecimento de modelos, relações e formulações de novas idéias, a fim de realizarmos previsões sobre o futuro. (Vasconcelos, 2002)

Como os bancos de dados constituem a memória dos sistemas de informação, capacitá-los com algum mecanismo de inteligência constitui pedra fundamental para início do processo de extrair conhecimento a partir dos dados armazenados por eles.

Valendo-se de algoritmos de Aprendizagem de Máquina (*Machine Learning*), é possível realizar o procedimento de extração de conhecimento em um processo conhecido por Descoberta de Conhecimento em Bancos de Dados (*Knowledge Discovery in Databases*) da sigla inglesa KDD.

O ato de obter informação e conhecimento útil a partir de registros pré-existentes sempre foi de interesse não só de empresas e centros de pesquisa (Azevedo & Côrtes, 2007):

- Gerentes, analistas de negócio e profissionais de tecnologia de informação preocupados em obter vantagens competitivas a partir da utilização e exploração de informações sobre clientes e mercados,
- Profissionais que trabalham com análise de dados,
- Profissionais de empresas e pesquisadores interessados em melhor explorar um acervo de dados para potencializar sua atuação,
- Profissionais de informática familiarizados com sistemas de informação voltados para análise de dados e/ou tomada de decisões,
- Usuários de informática que necessitem entender melhor os processos da construção e exploração de dados para busca de conhecimento e tomada de decisões,
- Profissionais de marketing, etc.

Ou seja, profissionais voltados para o processo de Descoberta de Conhecimento em Bancos de Dados possuem algum grau de interesse nesse conhecimento oculto.

Atualmente, em razão da globalização da economia e da acelerada evolução da tecnologia (rede mundial, softwares e computadores), o valor não está mais em simplesmente dominar a informação, mas sim em como trabalhar com o Conhecimento relacionado a ela. (Motta, 2007)

2.3 DESCOBERTA DE CONHECIMENTO EM BANCOS DE DADOS

O termo KDD foi utilizado pela primeira vez em 1989 para enfatizar que o conhecimento é o produto final de uma pesquisa baseada nos dados. (Vasconcelos, 2002)

Esse termo pode ser definido como o processo de identificação de padrões embutidos nos dados.

Porém, os padrões identificados devem ser válidos, novos, potencialmente úteis e compreensíveis (Fayyad et al., 1996).

As pesquisas relativas a este processo ganharam rápido crescimento a partir da década de 1990, motivadas pela evolução da tecnologia que vem permitindo a coleta, o armazenamento e o gerenciamento de quantidades cada vez maiores de dados (Fayyad et al., 1995).

Um outro fator considerado motivador deste crescimento é a ampliação das áreas de aplicações de KDD.

Como exemplos de áreas de aplicações, podem ser citados (Curotto, 2003):

- Bancária - na figura de aprovação de créditos;
- ciências e medicina - com descoberta de hipóteses, predição, classificação e diagnóstico;
- comercialização - nos setores de segmentação, localização de consumidores, identificação de hábitos de consumo;
- engenharia - na simulação e análise, reconhecimento de padrões, processamento de sinais e planejamento;
- financeira - no apoio para investimentos, controle de carteira de ações;
- gerencial - na tomadas de decisão, gerenciamento de documentos;

- internet - com ferramentas de busca, navegação, extração de dados;
- manufatura - com a modelagem e controle de processos, controle de qualidade, alocação de recursos;
- segurança – na detecção de bombas, icebergs e fraudes.

2.3.1 Etapas do KDD

O processo de descoberta de conhecimento em base de dados envolve diversas etapas, destacando-se a seguinte seqüência (Fayyad et al., 1996):

1. Consolidação de dados: estes são obtidos a partir de diferentes origens e consolidados numa única fonte.
2. Seleção e pré-processamento: são aplicadas diversas transformações sobre os dados para obtenção de um conjunto preparado para utilização dos algoritmos de mineração.
3. Mineração de dados: é a escolha da tarefa de mineração (algoritmo) que será utilizado.
4. Avaliação e interpretação: são examinados o desempenho e qualidade das regras extraídas, e também é verificada a facilidade de interpretação dessas regras.

2.4 MINERAÇÃO DE DADOS

A etapa mais importante do processo de KDD é conhecida como Mineração de Dados pois realiza a extração propriamente dita de padrões (conhecimento).

Segundo a revista online de tecnologia *ZDNET News* de fevereiro de 2001, "(...) a mineração de dados será um dos mais revolucionários desenvolvimentos da próxima década" (Konrad, 2001).

De fato, no ano de 2001 o *MIT Technology Review* incluiu a mineração de dados em sua lista anual contendo as "10 tecnologias emergentes que mudarão o mundo" (Werff, 2001).

A Mineração de Dados, do inglês *Data Mining* (DM), é uma metodologia que consiste em técnicas e processos que apresentam o ferramental necessário para a descoberta de conhecimento interessante, isto é, útil, durante exploração de bases de dados, extraindo, dessas bases, conhecimento na forma de hipóteses e regras.

Como passo inicial para sua execução, deve-se escolher a tarefa de mineração conforme os objetivos desejados para a solução procurada, isto é, conforme o tipo de conhecimento que se espera extrair do banco de dados.

A Figura a seguir ilustra as tarefas de mineração organizadas em atividades preditivas e descritivas:

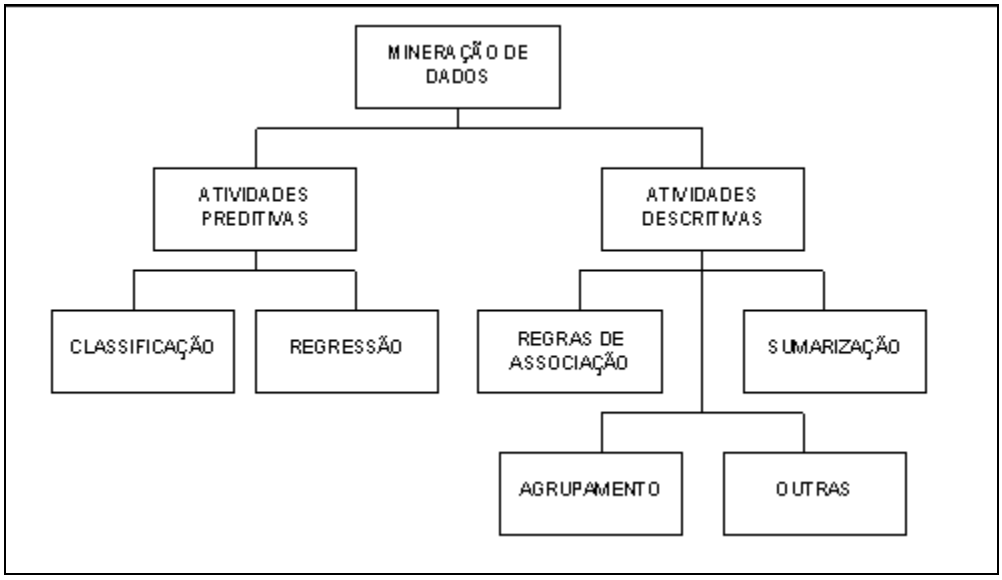

Figura 2.1: Atividades e Tarefas de Mineração de Dados (Rezende et al., 2003)

As atividades preditivas buscam identificar a classe de uma nova amostra de dados, a partir do conhecimento adquirido de um conjunto de amostras com classes conhecidas (Motta, 2004).

As atividades descritivas trabalham com um conjunto de dados que não possuem uma classe determinada, buscando identificar padrões de comportamento comuns nestes dados (Motta, 2004).

No passo seguinte é escolhido o algoritmo que atenda a tarefa de mineração eleita e que possa representar satisfatoriamente os padrões a serem encontrados.

Os algoritmos de mineração mais comuns são: Algoritmos Estatísticos, Algoritmos Genéticos, Árvores de Decisão, Regras de Decisão, Redes Neurais Artificiais, Algoritmos de Agrupamento, Lógica *Fuzzy*.

A mineração de dados é, na verdade, uma atividade interdisciplinar pela diversidade de tecnologias que podem estar envolvidas. A Figura a seguir sintetiza os assuntos envolvidos com DM.

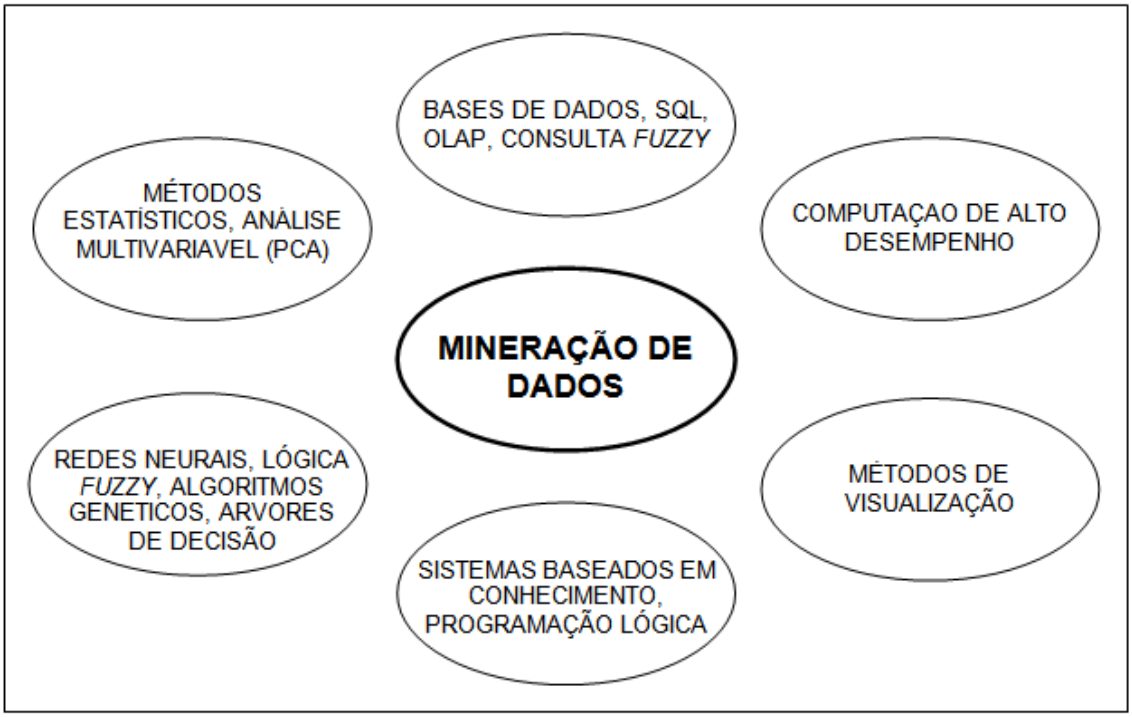

Figura 2.2: Assuntos envolvidos com Mineração de Dados (Han & Kamber, 2001)

Recentemente, *Data Mining* vem se tornando um conceito popular em termos de "uma ferramenta de gerenciamento de informação" (Silva, 2006), que revela estruturas de conhecimento que possam guiar ações estratégicas com objetivo de nortear decisões em condições de certeza limitada.

2.5 UM ALGORITMO DE MINERAÇÃO DE DADOS

Como o objetivo do presente trabalho é apresentar o acoplamento de um método não existente em um SGBD, torna-se necessário a implementação de um algoritmo de DM.

Desta forma, foi escolhido um algoritmo chamado *k*-Means, que é um método de agrupamento dentre os mais conhecidos e mais usuais.

Um método de agrupamento é utilizado para organizar os objetos representados no banco de dados em *k* (inteiro e maior que 1) grupos, onde cada objeto pertence a um único grupo e cada grupo deve conter pelo menos um objeto. No final, os objetos de um grupo são considerados similares entre si e dissimilares em relação aos objetos dos outros grupos.

Essa tarefa de particionamento é inteiramente baseada na análise e comparação automática entre os valores dos dados dos objetos do banco de dados, sem a supervisão humana e sem uma pré-classificação existente.

Em outras palavras, seja *D* um conjunto de dados com *n* objetos e *k* o número de grupos a serem formados, então o algoritmo de agrupamento organiza os objetos em *k* partições ($1 < k \leq n$), onde cada partição representa um grupo.

2.5.1 Funcionamento do algoritmo ***k***-Means

No algoritmo *k*-Means é dado inicialmente o número *k* de grupos. Então o método cria *k* partições iniciais para, em seguida, usar uma técnica de realocação iterativa, de forma a melhorar a posição dos objetos dentro de cada grupo.

Seja, por exemplo, o banco de dados com 21 objetos representados no espaço bi-dimensional abaixo:

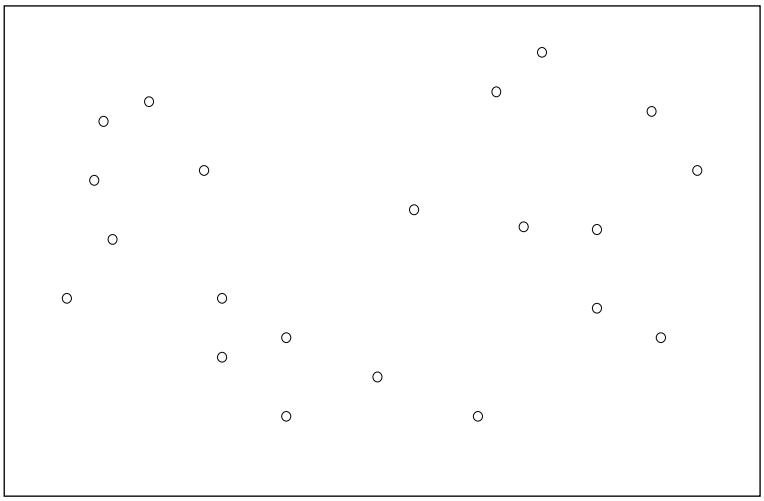

Figura 2.3: Banco de Dados inicial

Dado o número desejado de partições k ($1 < k \leq n$), o *k*-Means escolhe aleatoriamente *k* pontos chamados centróides.

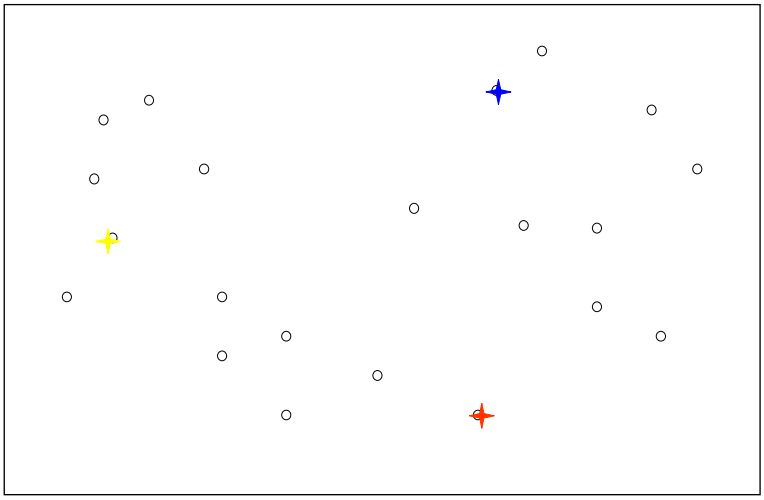

Figura 2.3.1: Primeiros Centróides

Em seguida, para cada objeto é calculada a sua distância em relação ao centróide, considerando que o objeto pertence ao grupo correspondente a menor distância calculada.

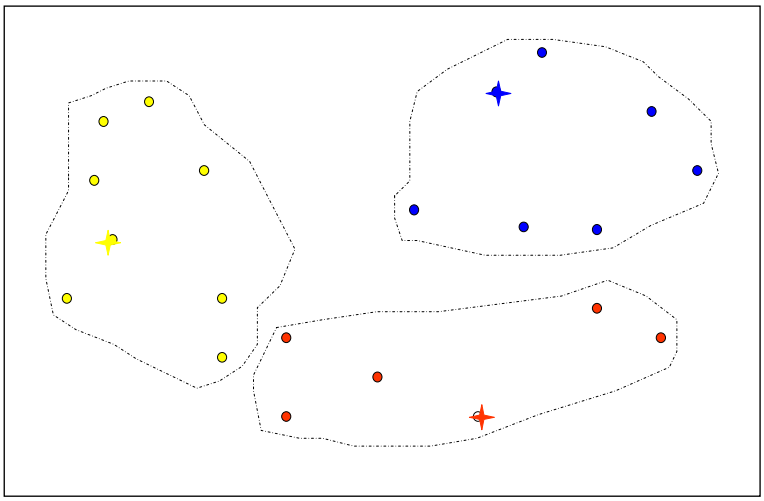

Figura 2.3.2: Agrupamento Inicial

Para cada grupo, calcula a média aritmética das coordenadas (valores dos atributos) dos objetos. Os resultados são as coordenadas dos novos centróides de cada grupo.

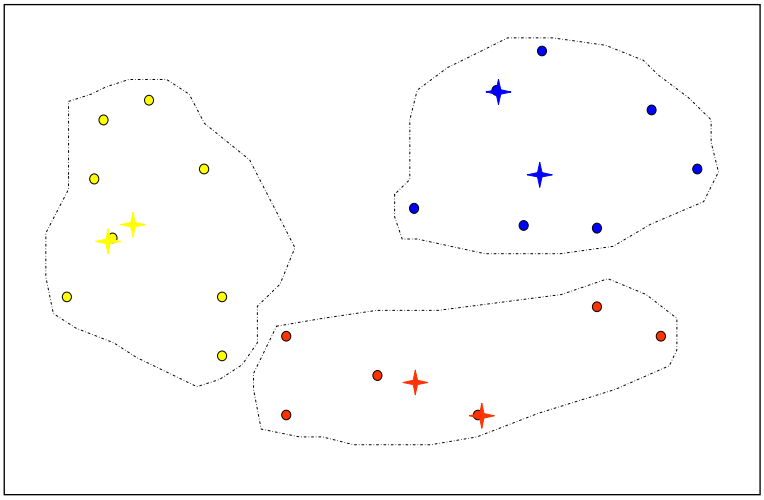

Figura 2.3.3: Novos Centróides Calculados

Recalcula as distâncias e reorganiza os grupos.

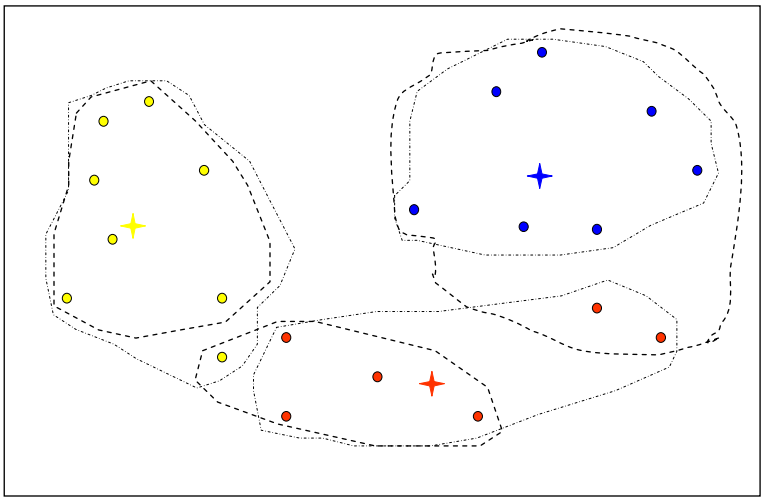

Figura 2.3.4: Início do Reagrupamento

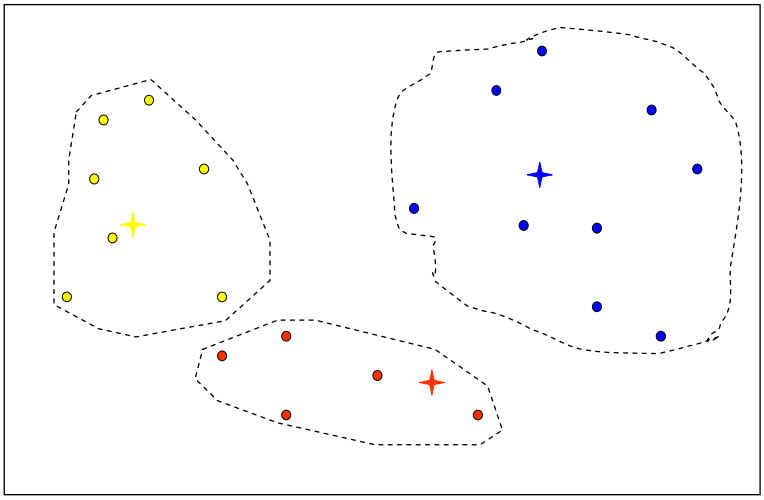

Figura 2.3.5: Reagrupamento Concluído

Repete o processo até alcançar convergência.

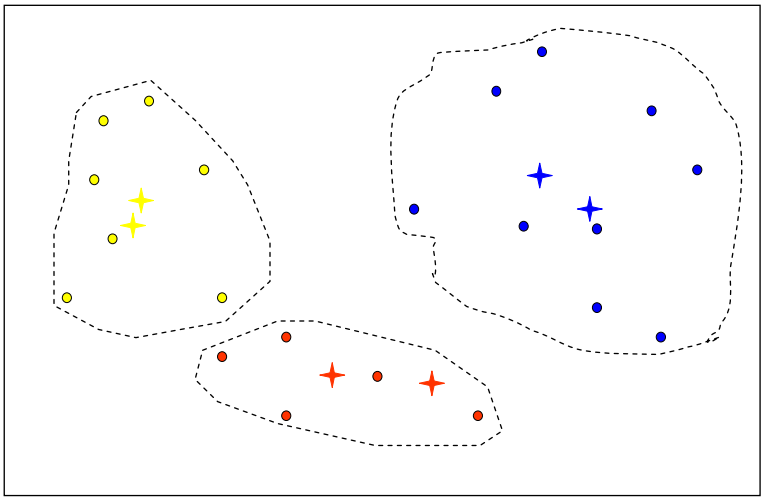

Figura 2.3.6: Cálculo dos Novos Centróides

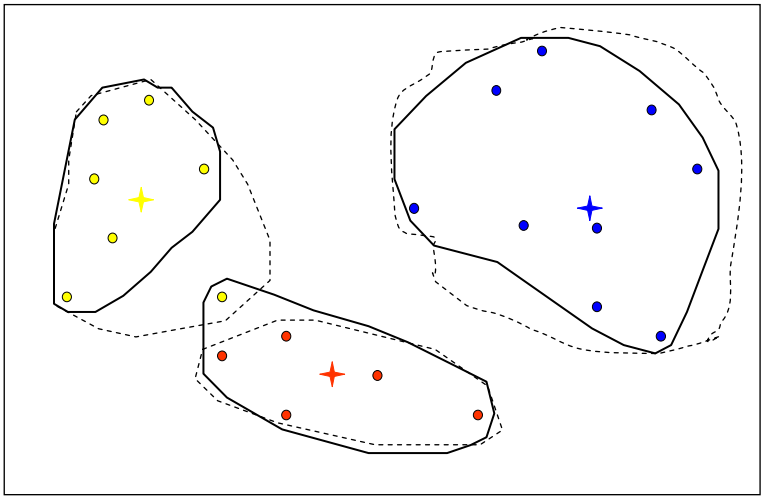

Figura 2.3.7: Início do Novo Reagrupamento

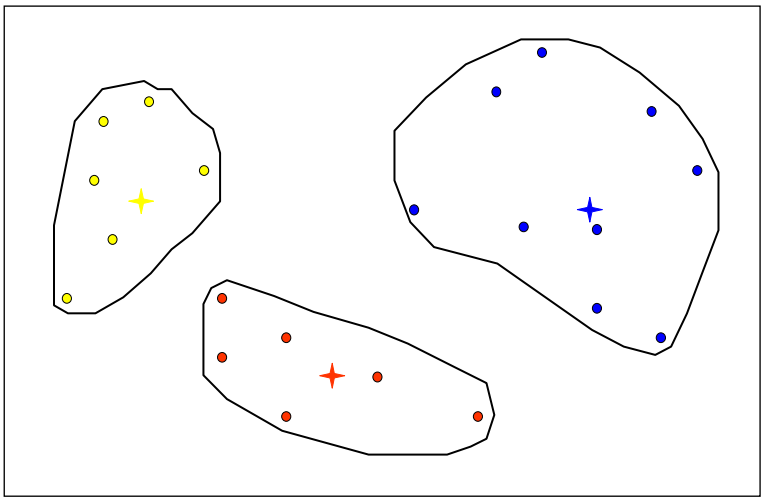

Figura 2.3.8: Agrupamento Final

O algoritmo alcançará convergência quando os centróides não se alterarem mais ou se alterarem em uma quantidade menor que uma tolerância previamente estabelecida.

O algoritmo *k*-Means poderia ser assim enunciado:

Entrada: *D*: um conjunto de dados com *n* objetos.

k: o número de grupos.

Saída: Um conjunto de *k* grupos.

(1) "Escolha arbitrariamente *k* objetos de *D* como os centros iniciais dos grupos";

(2) Repita

(3) "Atribua cada objeto para o grupo que tenha o centro mais próximo do objeto";

(4) "Calcule os novos centros dos grupos (novo centro do grupo = valor médio dos objetos do grupo)"

(5) Até "Não ocorrer mais mudanças nos centros"

O fluxograma ilustrado pela figura a seguir demonstra o funcionamento do algoritmo (Souza, 2007):

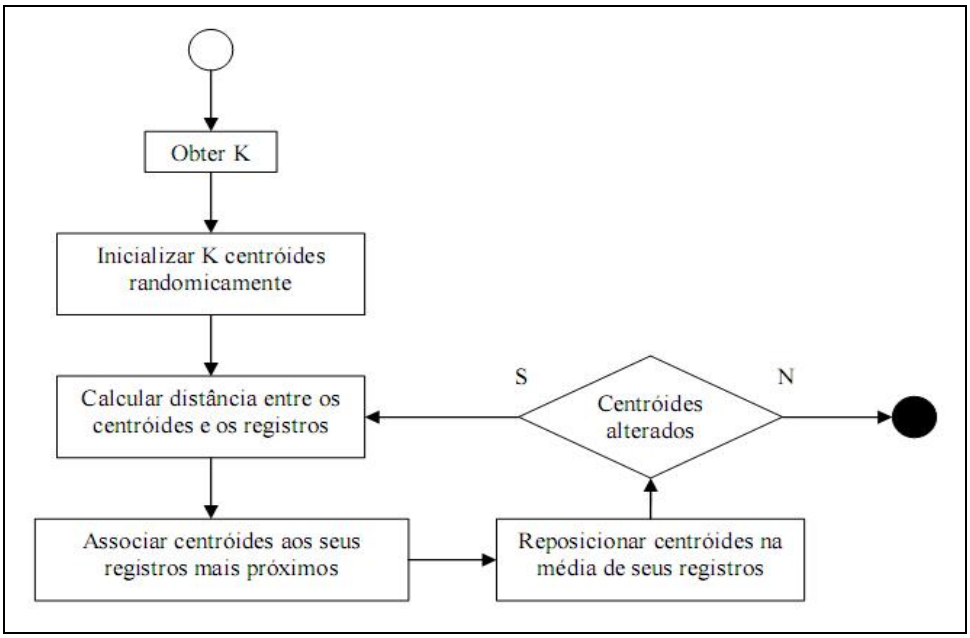

Figura 2.4: Fluxograma do algoritmo *k*-Means (Souza, 2007)

2.5.2 Métricas

Métricas são medidas de distância necessárias para o cálculo vetorial entre os centróides e os demais objetos, dentro da implementação do algoritmo *k*-Means.

Para a composição do presente trabalho foram utilizadas duas métricas, formuladas e apresentadas abaixo:

- Distância euclidiana:

A distância euclidiana entre objetos é definida como a raiz quadrada da soma dos quadrados das diferenças das coordenadas (valores dos atributos) de cada objeto.

A figura abaixo sintetiza essa idéia:

$$D_{euclid}(i,j) = \sqrt{\sum_{l=1}^{p} (x_{il} - x_{jl})^2}$$

Figura 2.5: Distância Euclidiana

- Distância de Manhattan:

A distância de Manhattan entre objetos é definida como a soma das diferenças absolutas entre as coordenadas (valores dos atributos) dos objetos:

$$D_{mh}(i,j) = \sum_{l=1}^{p} | x_{il} - x_{jl} |^2$$

Figura 2.6: Distância de Manhattan

A título de curiosidade, a distância de Manhattan recebeu esse nome pelo fato de definir a menor distância possível que um carro é capaz de percorrer numa malha urbana reticulada ortogonal, tal como se encontram em zonas como Manhattan.

CAPÍTULO 03

RECURSOS DE MINERAÇÃO EM SGBD

3.1 BANCOS DE DADOS E SGBD

Um banco de dados consiste de "uma coleção de dados inter-relacionados, representando informações sobre um domínio específico" (Korth & Silberschatz, 1994).

Dessa forma, uma lista telefônica ou um catálogo de produtos de beleza são considerados como bancos de dados.

Por outro lado, um Sistema de Gerenciamento de Bancos de Dados (SGBD) é constituído por um conjunto de *softwares* responsáveis pelo gerenciamento de dados em um banco de dados, disponibilizando uma interface prática para persistência, organização e recuperação.

Um SGBD provê um tipo de acesso otimizado aos dados propiciando uma recuperação rápida, segura e eficiente das informações.

Porém, um software é considerado um SGBD somente se apresentar alguns pré-requisitos, como por exemplo (Leão & Silva, 2002):

- Controle de redundância de dados: deve fornecer mecanismos para que seja possível impedir ou controlar dados redundantes;
- Compartilhamento dos dados: deve fornecer mecanismos de controle de acesso simultâneo aos dados, impedindo que uma informação que esteja sendo alterada seja acessada;

- Controle de atomicidade: deve fornecer meios para que sejam realizadas, dentro de uma transação, as atualizações pertencentes a uma operação, mantendo a integridade e consistência dos dados;

- Controle de acesso: deve fornecer mecanismos para decisão sobre quais usuários poderão acessar os dados do sistema e quais serão as tarefas que esses usuários poderão realizar com esses dados;

- Visão dos dados: deve permitir que os dados sejam visualizados de forma abstrata, ou seja, de uma forma diferente daquela fisicamente armazenada.

Ao conjunto Banco de Dados e Sistema de Gerenciamento de Bancos de Dados dá-se o nome de Sistema de Banco de Dados (SBD).

Uma definição mais específica atribui essa nomeação ao conjunto composto por dados, hardware, software e usuários (Figura 3.1).

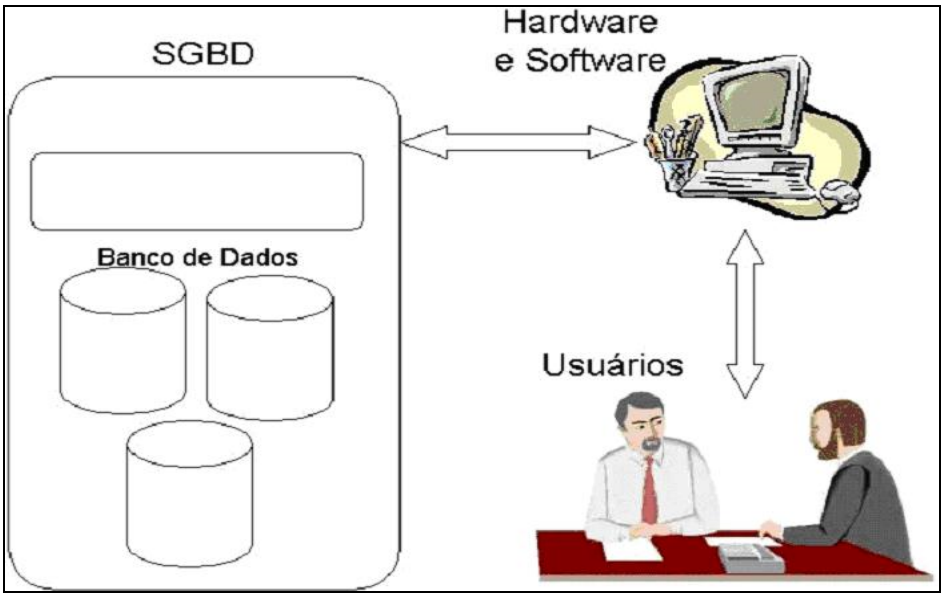

Figura 3.1: Componentes de um Sistema de Banco de Dados (Rezende, 2006)

Baseado nessa definição, os objetivos de um Sistema de Banco de Dados são o de isolar o usuário dos detalhes internos do banco (provendo a abstração de dados) e promover a independência dos dados em relação às aplicações, ou seja, tornar a estratégia de acesso e a forma de armazenamento independente da aplicação (Rezende, 2006).

Em junho de 1970 um pesquisador da IBM chamado E. F. Codd desenvolveu um Modelo de Dados Relacional, que serviu como base da moderna tecnologia dos bancos de dados atual.

Nesse modelo, as bases de dados eram visualizadas como um conjunto de tabelas, cada uma representando uma relação, onde os dados poderiam ser relacionados através de atributos com mesmo domínio (conjunto de valores previamente definidos) e semântica (dados de uma coluna sempre serão do tipo de dado definido na criação da coluna).

Hoje esse modelo é amplamente difundido, com uma base firmemente estabelecida no mundo das aplicações de bancos de dados e contando com numerosos produtos de pequeno, médio e grande porte (Vasconcelos, 2002).

No entanto, devido à demanda gerada por novas aplicações, Sistemas de Gerenciamento de Bases de Dados baseados em outros modelos surgiram. Como exemplo destes novos sistemas tem-se os chamados Sistemas Orientados a Objeto e os Sistemas Objeto-Relacional.

3.2 SOFTWARES DE MINERAÇÃO

Em meados da década de 1990, ainda em ambiente acadêmico, os primeiros softwares utilizados para mineração de dados começaram a ser desenvolvidos.

Hoje, algumas dezenas de ferramentas comerciais para DM são disponibilizadas por várias empresas como SAS (*Enterprise Miner*), IBM (*Intelligent Miner*) e SPSS (*Clementine*) além de outra dezena de ferramentas *free* e/ou *open source*.

A maioria dos sistemas comerciais encontrados no mercado atual já demonstrou sua capacidade de atuar como importante ferramenta de apoio no processo de tomada de decisões.

Porém, a popularidade desses softwares ainda é baixa se comparada com a popularidade das ferramentas comerciais de data *warehousing* e OLAP (Gonçalves, 2007).

Artigos recentes apontam três motivos principais para explicar esta situação (Gonçalves, 2007):

- Os *softwares* comerciais de DM possuem um custo elevado;
- Muitos *softwares* não realizam a mineração diretamente sobre as tabelas de um banco de dados. Em muitos casos, é necessário exportar os dados para um repositório auxiliar;
- O terceiro motivo está no fato de as ferramentas de mineração serem complexas em sua utilização: muitos *softwares* exigem que o usuário tenha algum conhecimento a respeito do funcionamento dos algoritmos de mineração implementados na ferramenta.

3.3 OS SGBD ATUAIS

Atualmente o mercado demonstra estar realizando amplos esforços para corrigir os problemas citados anteriormente.

Um exemplo neste sentido é dado por duas das maiores empresas da indústria de *software*: Oracle**,** atual líder do mercado de SGBD com 44% e Microsoft**,** terceira na lista da *Gartner RAS Core Research* com 19%.

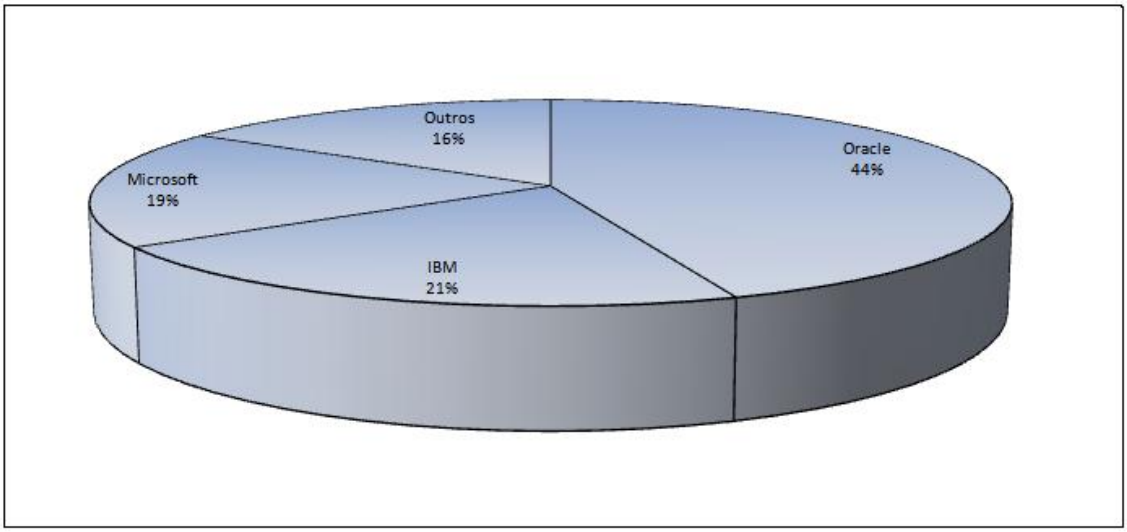

Figura 3.2: Distribuição do mercado de SGBD em 2006

As versões mais populares dos SGBD's comercializados por estas empresas, Oracle 10g e SQL Server 2005, são dotadas de recursos inovadores e revolucionários para a utilização dos algoritmos de DM.

A principal intenção dessas empresas ao elaborar as funcionalidades dessas versões, foi resolver dois dos mais importantes problemas que assolam o universo dessas aplicações de uma só vez:

- O custo elevado das ferramentas de mineração de dados;
- A impossibilidade da mineração direta.

Logo, usuários do Oracle 10g e do SQL Server 2005 que estejam interessados em analisar dados através de técnicas de *data mining* não precisam mais adquirir uma ferramenta de mineração externa e a um custo elevado, uma vez que é possível utilizar os algoritmos e ferramentas disponibilizados *built-in* nestes sistemas.

Além disso, as técnicas implementadas são capazes de minerar as tabelas de forma direta, sem necessidade de exportar dados para tabelas auxiliares.

A maior vantagem proporcionada pelos sistemas dessas empresas consiste no fato de possuírem metodologias próprias embutidas para a mineração de dados.

Essas metodologias são simples, intuitivas e mais amigáveis ao usuário se comparadas com os modelos utilizados pelas ferramentas tradicionais de DM.

Em ambos os sistemas a mineração de dados funciona em um esquema dividido em três etapas:

- Criação de um modelo de mineração de dados;
- Treinamento do modelo criado;
- Consulta dos resultados.

Durante a criação do modelo, o usuário especifica o tipo de problema que deseja resolver utilizando uma técnica da mineração.

Nessa etapa, poderiam ser formulados problemas como "quais produtos freqüentemente são comprados em conjunto?" ou "qual o perfil do cliente que realiza grandes gastos com freqüência?".

Após especificar os problemas que serão atacados, o usuário seleciona as tabelas do banco que deseja investigar, assim como os seus atributos.

Na etapa seguinte, de treinamento do modelo, o usuário comanda a execução do algoritmo de mineração sobre as tabelas selecionadas.

Como resultado obtém-se um conjunto de regras e padrões extraídos da base.

Durante esse treinamento o usuário tem a liberdade para determinar qual a faixa de dados que será examinada. Por exemplo, ele poderia informar ao sistema que apenas os dados referentes às vendas de produtos do mês de dezembro de 2007 deveriam ser analisados.

Na etapa final de consulta dos resultados, o usuário possui condições de explorar o conjunto de regras geradas e os padrões descobertos pelo algoritmo escolhido.

A exploração das regras e padrões é disponibilizada de maneira simples e intuitiva através de uma *view* (visão) do banco de dados, de modo que a consulta e avaliação dos resultados descobertos seja facilitada.

No Microsoft SQL Server 2005, é possível realizar consultas específicas sobre itens de interesse como, por exemplo, "quais produtos costumam ser comprados em conjunto com fraldas às sextas-feiras em alguns supermercados?".

Esse sistema também disponibiliza uma linguagem denominada DMX (*Data Mining eXtensions*) para a manipulação de seus recursos de mineração.

A DMX é *SQL-Like*, ou seja, possui sua sintaxe e semântica bem parecida com a linguagem SQL, sendo, portanto, de fácil utilização.

Abaixo segue um exemplo da sintaxe para criação de uma nova estrutura de mineração no banco de dados, utilizando a DMX:

```
CREATE [SESSION] MINING STRUCTURE <structure>
(
    [(<column definition list>)]
)
[(<parameter list>)]
```

Figura 3.3: Sintaxe de Criação de Estrutura de Mineração.

Pelo exposto, constata-se que, com o advento dos mais atuais SGBD's, que inclusive oferecem suporte para mineração direta sobre as tabelas, não mais existe a

necessidade das consultas em bancos de dados serem exclusivamente realizadas por profissionais da área de computação.

Desta forma, empresas de pequeno e médio porte também podem contar com métodos ágeis e extremamente eficientes de mineração, que possuem a característica de extrair conhecimento útil contido em seus bancos de dados, de forma rápida e simples e por um custo mais acessível a sua realidade.

CAPÍTULO 04

IMPLEMENTAÇÃO DE ACOPLAMENTOS

O SGBD escolhido para a realização dos acoplamentos foi o SQL Server 2005, uma ferramenta popular e robusta da empresa Microsoft, disponibilizado em conjunto com o Visual Studio 2005 versão 8.0.

A escolha desse sistema se deveu ao fato de permitir vários mecanismos de acoplamento de novas funcionalidades, atendendo perfeitamente à proposta do trabalho.

Apesar de ser um produto para venda, uma versão de testes desse software pode ser obtida em http://msdn.microsoft.com/en-us/vs2005/default.aspx e usada por um período de 180 dias.

Essa versão possui todos os requisitos necessários para a execução dos testes realizados a seguir.

4.1 TIPOS DE ACOPLAMENTO NO MICROSOFT SQL SERVER 2005

O SGBD Microsoft SqlServer 2005 provê quatro principais mecanismos de extensão que permitem o acoplamento de funcionalidades ao servidor e às ferramentas inicialmente disponibilizadas ao usuário. São elas:

- uso de *stored procedures*,
- uso dos mecanismos de extensão do Microsoft Visual Studio 2005,
- desenvolvimento de *plug-ins* que estendem o conjunto de algoritmos disponíveis,

- criação (escrita) de novos *viewers* para visualização dos modelos de mineração.

Neste capítulo são apresentados o mecanismo de desenvolvimento de *plug-ins* e o uso de *stored procedures* para o Microsoft SqlServer 2005.

4.2 DESENVOLVIMENTO DE *PLUG-IN*

Os algoritmos acoplados, valendo-se desse mecanismo de extensão, operam usando essencialmente a mesma interface que os algoritmos fornecidos pelo SQLServer e sem perda de performance (MacLennan & Tang, 2005).

De fato, esses acoplamentos consistem em objetos COM (*Component Object Model*) que são objetos usados para permitir comunicação interprocessos e criação dinâmica de objetos, no sistema Windows, em qualquer linguagem de programação que disponibiliza suporte a essa tecnologia.

Para um acoplamento efetivo e funcional, esses objetos precisam executar ao menos três tarefas:

- Descrever suas características,
- Trabalhar com padrões em dados (detectar, persistir e usar esses padrões) e
- Expor os padrões detectados

Os objetos COM devem, assim, implementar e compreender um conjunto de interfaces especificadas pelos desenvolvedores do SGBD de modo a permitir a perfeita execução das tarefas necessárias a efetiva integração com o SQLServer:

- A descrição das características é manipulada pela interface **IDMAlgorithmMetadata**;

- Padrões são detectados e usados através da inteface **IDMAlgorithm** e persistidos pela interface **IDMPersist**.
- O padrão de navegação é realizado através da interface **IDMAlgorithmNavigation**.

Portanto, durante o processo de desenvolvimento do acoplamento deve-se implementar um conjunto de pelo menos três classes, uma para cada tarefa descrita acima:

- A classe **Metadata**, que exibe as características e cria os objetos,
- A classe **Algorithm**, que detecta, persiste e usa os padrões encontrados nos dados,
- A classe **Navigator**, que exibe os padrões encontrados pela classe **Algorithm**.

Após as implementações e o registro do acoplamento no Analysis Services, a nova funcionalidade encontra-se disponível para uso podendo ainda ser distribuída para outros usuários através da criação de um pacote de redistribuição.

4.2.1 O DMPluginWrapper.dll

A Biblioteca de ligação dinâmica (*Dynamic-link library - dll***) DMPluginWrapper.dll** é parte essencial no processo de desenvolvimento do acoplamento, pois traduz as chamadas COM do servidor Analysis Services para o algoritmo acoplado.

Ela foi escrita usando o Microsoft Visual C++ 2005 e seu código fonte é disponibilizado não compilado, por isso necessita do Microsoft Visual Studio 2005 para ser criada.

Essa biblioteca foi desenvolvida pela equipe responsável pelo SqlServer 2005 e pode ser baixada em **http://download.microsoft.com/download/f/7/4/f74cbdb1-87e2-**

4794-9186-e3ad6bd54b41/DMMgdPlugInAPI.msi, devendo ser compilada para uso no sistema onde o acoplamento será implementado.

Como a biblioteca usa a pasta de inclusão "C:\Program Files\Microsoft SQL Server\90\SDK\Include" para procurar pelo arquivo "oledbdm.h" que é instalado juntamente com o SQL Server 2005 pode ser necessário editar o projeto e mudar o diretório de inclusão para que ele aponte para a localização exata do arquivo:

Botão direito sobre o nome do projeto → *Properties* → *Configuration Properties* → C/C++ → *General*

Criar a biblioteca **DMPluginWrapper.dll** é parte necessária ao início do desenvolvimento do acoplamento.

Portanto, deve-se iniciar o projeto **DMPluginWrapper** e compilá-lo de modo que seja construída tal dll.

4.2.2 Passos para o acoplamento

Os passos abaixo descrevem o processo de implementação necessário para o acoplamento de algoritmos na ferramenta de mineração *Business Intelligence Development Studio* integrado ao *Analysis Services*.

O código apresentado resulta em um *stub* com o qual é possível integrar rapidamente um algoritmo ao *Analysis Services*.

Isso significa que é necessário apenas concentrar esforços na implementação do algoritmo escolhido, não se importando com a integração que ocorrerá por conseqüência.

Deve-se seguir os passos:

1 - Criar um novo projeto (Visual C# Projects) no Microsoft Visual Studio® 2005 (versão 8.0 ou superior);

2 - "*Sign the assembly*" de modo que ele possa ser carregado no *Global Assembly Cache* (GAC):

Menu *Project* → *Properties* → *Signing*. Marcar *'Sign the assembly'* e no campo "*Choose a strong name key file*" selecione <New...>);

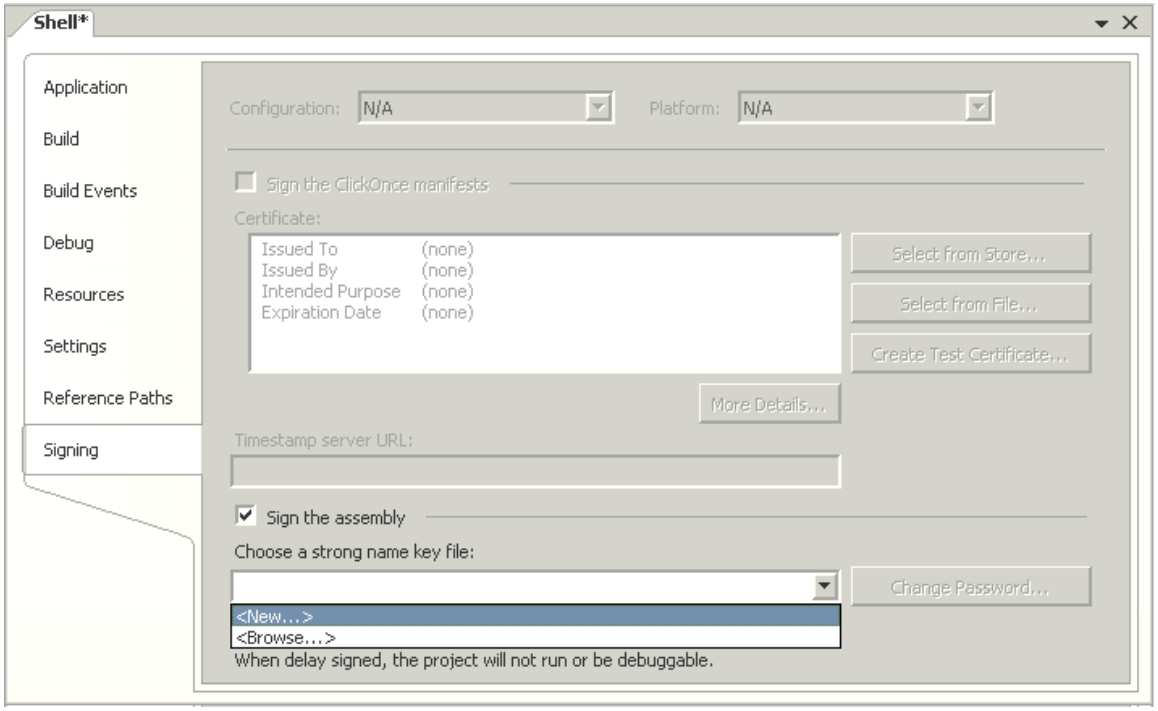

Figura 4.1: *Sign the assembly*

3 - Adicionar o projeto recém criado a *Solution* para gerar a dll "DMPluginWrapper":

Esse projeto já deve estar criado e compilado sem erros.

4 - Adicionar uma referencia ao "DMPluginWrapper":

No menu *Project*, clicar em *Add reference*. Na aba *Projects* escolha o projeto DMPluginWrapper;

5 - Adicionar os seguintes "*post-build steps*" ao projeto:

Menu *Project* → *Properties* → *Build Events* → *Post-build event command line*. Adicionar os *post-build* abaixo:

"C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\RegAsm.exe" "$(TargetPath)"

"C:\Arquivos de Programas\Microsoft Visual Studio 8\SDK\v2.0\Bin\gacutil.exe" /u $(TargetName)

"C:\Arquivos de Programas\Microsoft Visual Studio 8\SDK\v2.0\Bin\gacutil.exe" /if "$(TargetPath)"

6 - Criar as classes descritas anteriormente com o código contido nos anexos:

6.1 - Classe Metadata.cs

Anexo A.

6.2 - Classe Algorithm.cs

Anexo B.

6.3 – Classe AlgorithmNavigator.cs

Anexo C.

Esses códigos são fornecidos em uma *Sollution* juntamente com o download do *plugin* **DMPluginWrapper.dll**.

Com as implementações acima definidas, parte-se, de fato, para o acoplamento. Este acontece ao registrar o plug-in junto ao *Analysis Services*:

7 - Na pasta onde o DMPluginWrapper foi criado, executar os comandos:

"%ProgramFiles%\Microsoft Visual Studio 8\SDK\v2.0\Bin\gacutil.exe" /u DMPluginWrapper

"%ProgramFiles%\Microsoft Visual Studio 8\SDK\v2.0\Bin\gacutil.exe" /if DMPluginWrapper.dll

8 - Na pasta onde o projeto do acoplamento foi criado, executar os comandos:

"%WINDIR%\Microsoft.NET\Framework\v2.0.50727\RegAsm.exe" Shell.dll

"%ProgramFiles%\Microsoft Visual Studio 8\SDK\v2.0\Bin\gacutil.exe" /u Shell

"%ProgramFiles%\Microsoft Visual Studio 8\SDK\v2.0\Bin\gacutil.exe" /if Shell.dll

Com esses comandos os *assemblies* são registrados e carregados no *Global Assembly Cache* (GAC).

Maiores informações sobre o GAC podem ser encontradas em http://msdn2.microsoft.com/en-us/library.

9 - Abrir o SQL Server Management Studio;

10 - Conectar ao servidor *Analysis Services* alvo;

11 - Selecionar File\New\Analysis Services XMLA Query

12 - Executar o XMLA apresentado no Anexo D, onde "00000000-0000-0000-0000-000000000000" deve ser substituído pelo **Guid** (*Globally unique identifier*) gerado na aplicação e "NOME_DO_ACOPLAMENTO" deve ser substituído pelo ServiceName do algoritmo;

Informações sobre XMLA podem ser encontradas em: http://www.xmlforanalysis.com/

Guid é um número inteiro de 128 bits que pode ser usado para identificar algo de modo único. Nesse caso, ele é utilizado para identificar o projeto criado.

13 - Reiniciar o *Analysis Services*:

Iniciar → Painel de Controle → Ferramentas Administrativas → Serviços → Clicar com botão direito sobre "SQL Server Analysis Services" → Clicar em Reiniciar.

Com esses passos, o acoplamento estará efetivado, restando apenas testá-lo:

1 - Abrir o "*Business Intelligence Development Studio*";

2 - Selecionar File → New → Project;

3 - Em "*Project Types*" escolher "*Business Intelligence Projects*";

4 - Em "*Templates*" escolher "*Analysis Services Project*";

No "*Solution Explorer*":

5 - Conectar a uma base de dados;

6 - Definir uma *View*;

7 - Definir uma estrutura de mineração:

Clicar com o botão direito em "*Mining Structures*" e escolha "*New Mining Structure*";

Clicar em "*Next*" e em "*Next*" outra vez;

O acoplamento deve aparecer na tela "*Select the Data Mining Technique*" no *drop-down* "*Which data mining technique do you want to use?*"

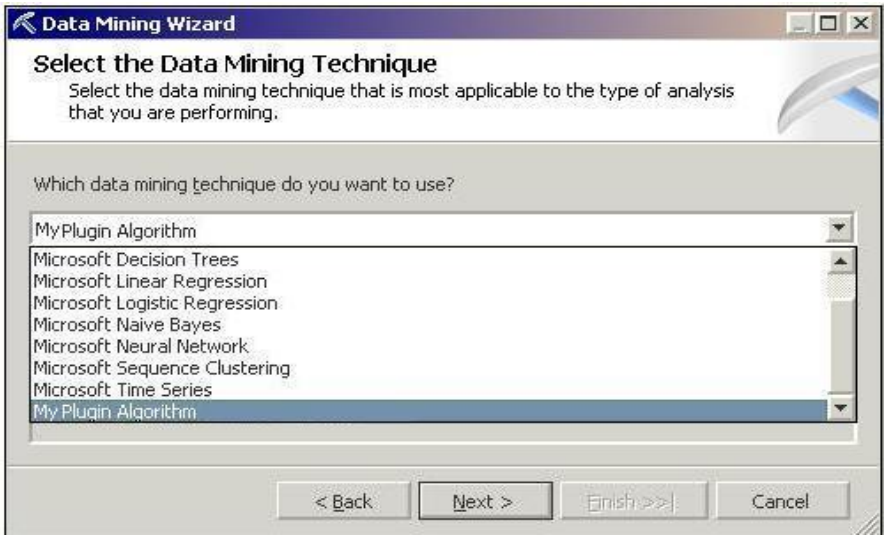

Figura 4.2: Acoplamento

Se ele não estiver sendo apresentado, significa que o *Analysis Services* estava sendo executado na última vez em que o projeto foi compilado.

Neste caso, executar os seguintes passos para corrigir o problema:

1 - Salvar a solução e fechar o "*Business Intelligence Development Studio*";

2 - Parar o *Analysis Services* e compilar o projeto outra vez;

3 - Iniciar o *Analysis Services*, abrir o "*Business Intelligence Development Studio*" e abrir a solução.

Com esses passos o acoplamento está efetivado. Resta, então, proceder apenas a implementação propriamente dita de algum algoritmo de mineração.

Isso pode ser realizado revendo os códigos das classes apresentadas e alterando-os de acordo com a necessidade, por exemplo:

1 - Alterar o método "*GetServiceType*" para identificar o algoritmo escolhido;

2 - Alterar os métodos "*GetServiceName*/*GetDisplayName*/*GetServiceDescription*" para apresentar a descrição do algoritmo;

3 - Alterar o método "*GetViewerType*" para retornar um viewer específico, etc.

4.3 S*TORED PROCEDURES*

Esse acoplamento acontece ao implementar uma *Stored Procedure* e executá-la no *Analysis Services*.

Uma *Stored Procedure* é um tipo de programa, criado no servidor (no caso o SQL Server 2005), dentro de um banco de dados específico, que possui como principal objetivo realizar operações com os dados deste banco de dados.

A *Stored Procedure* apresentada faz uso de outras *Procedures* bem como algumas funções definidas pelo usuário (*User Defined Functions*) que também são um tipo de programa criado no servidor, dentro de um banco de dados específico, para retornar um valor (ou um conjunto de valores) para um usuário, para outra função, para uma *Stored Procedure* (como é o caso) ou para uma aplicação.

A *Procedure CHECKTABLE* verifica a existência de uma tabela no banco de dados. Já *CHECKCOLUMN* verifica a existência de uma coluna em uma tabela. Ambas são necessárias para realizar a validação dos parâmetros de entrada da *Procedure* principal.

CHECKCLUSTERS valida a quantidade de clusters solicitada pelo usuário, ou seja, se $1 < k \leq n$.

SHOWCLUSTERS exibe, se solicitado, as "coordenadas" dos centróides de cada cluster ao final da execução.

SHOWRESULT exibe o resultado final da clusterização.

Já a *Function EUCLIDEANDISTANCE* calcula e retorna a distância euclidiana entre pontos. *MANHATTANDISTANCE* por sua vez, calcula e retorna a distância de Manhattan entre pontos.

O procedimento que implementa a execução do algoritmo *k*-Means, *Procedure KMEANS*, as *Procedures* e as duas *Functions* descritas acima encontram-se no Anexo E.

CAPÍTULO 05

CONSIDERAÇÕES FINAIS

A tentativa de realização do acoplamento utilizando os mecanismos de extensão do Microsoft *Visual Studio* 2005 e os de desenvolvimento de *plug-ins* para o Microsoft *SqlServer* 2005 foi interrompida na etapa de registro do *plug-in* junto ao *Analysis Services*, de modo que essa etapa do trabalho não foi concluída.

Com sucesso contatou-se a equipe da Microsoft de desenvolvedores do plug-in de acoplamento para o SQL Server 2005 que prontamente respondeu em um fórum criado com a finalidade de atender os usuários desse plug-in de modo a tentar solucionar os problemas técnicos que surgissem durante a utilização.

Porém, estes problemas se revelaram extremamente complexos, pelo fato de requererem conhecimentos extras sobre o Sistema Operacional, o que impediu a realização da conclusão dessa etapa do trabalho.

Contudo, a realização do acoplamento utilizando *Stored Procedure* procedeu-se como esperado além de mostrar-se válida pelo fato de apresentar resultados satisfatórios, como os exibidos na Figura 5.1.

Os testes comparativos da execução do algoritmo foram realizados utilizando a base de dados da flor Íris com um total de 150 mostras, cada amostra com quatro atributos contínuos (largura e comprimento da pétala e da sépala), num total de três classes, cada uma com 50 amostras.

Essa base de dados encontra-se disponível em ftp://ftp.ics.uci.edu/pub/machine-learning-databases/iris.

O algoritmo foi executado e seus resultados comparados com a classificação oficial da base de dados da flor Íris, utilizando-se as duas métricas apresentadas anteriormente (Figura 5.1).

Resultado	Euclidiana	Manhattan
Acertos:	134	133
Erros:	16	17
Total:	150	150
% Acertos:	89,33%	88,67%

Figura 5.1 - Resultados Comparativos

Os testes mostraram, também, que depois de construída e acoplada, a utilização de nova funcionalidade se torna bastante amigável e com desempenho computacional satisfatório.

Como visto anteriormente, várias são as maneiras de realizar um acoplamento de novas funcionalidades as ferramentas apresentadas neste trabalho.

Porém, a utilização de S*tored Procedures* mostrou-se adequada possuindo as seguintes vantagens sobre os demais métodos de acoplamento:

- melhor desempenho pelo fato da *procedure* estar armazenada no servidor, o que possibilita sua execução mais rápida;

- como o procedimento fica armazenado no servidor, um completo isolamento entre o modelo de dados e a aplicação consumidora é propiciado, o que não afeta essa aplicação no caso de haver algum ajuste no modelo para, por exemplo, realizar melhorias de desempenho ou inclusão de novas colunas como entrada para a execução do algoritmo;

- reduz consideravelmente o tráfego de rede, no caso de acesso remoto a essa nova funcionalidade, pois para a execução da *procedure* não é necessário o envio de dados do cliente ao servidor, mas apenas o nome do procedimento e seus parâmetros;

- o processamento das regras de negócio, no caso deste trabalho o algoritmo *k*-Means, é executado inteiramente no servidor, que geralmente é uma máquina mais robusta, deixando assim a máquina cliente mais leve;

- aumento na segurança das informações, pois as *Stored Procedures* executam operações em tabelas ainda que os usuários não disponham de privilégios de acesso sobre as mesmas.

Considerando que atualmente as duas empresas que tiveram seus softwares analisados no decorrer do desenvolvimento deste trabalho já apresentaram a seus consumidores versões mais atuais de seus produtos, Oracle 11g e Microsoft SQL Server 2008, um novo trabalho poderia ser realizado, porém focando-se nos novos e promissores softwares dessas empresas.

Neste novo trabalho poderiam ser apresentadas as novas ferramentas fornecidas para a realização dos acoplamentos, bem como uma análise comparativa com a versão imediatamente anterior dos softwares, apresentando as possíveis melhorias e facilidades de uso.

REFERÊNCIAS BIBLIOGRÁFICAS

CÔRTES, S., AZEVEDO, H. L. C., "*Data Mining - Conceitos, Técnicas, Ferramentas e Aplicações*". Disponível na Internet. http://www.cce.puc-rio.br/informatica/dataminingcentro.htm. Acessado em 05 de novembro de 2007.

CUROTTO, C. L., "*Integração de Recursos de Data Mining com Gerenciadores de Bancos de Dados Relacionais*", Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2003.

FAYYAD, U.M., PIATETSKY-SHAPIRO, G. & SMITH, P. "*From Data Mining to Knowledge Discovery: An Overview*", In: Advances in Knowledge Discovery and Data Mining, FAYYAD, U.M. et alii (eds.), AAAI/MIT Press, pp. 1-36, 1995.

FAYYAD, U.M., PIATETSKY-SHAPIRO, G. & SMITH, P. "*From Data Mining to Knowledge Discovery in Databases*", Al Magazine, Vol. 17, No. 3, pp. 37-54, 1996.

FERNANDES, S. M. S. P., "*Apoio à Tomada de Decisão em Empresas de Serviço de Turismo: Um Estudo de Caso em Agência de Viagens*", Dissertação de Mestrado, CITMA, Funchal, Portugal, 2006.

FRAWLEY, W.J, PIATETSKY-SHAPIRO, G., MATHEUS , C.J., "*Knowledge discovery in databases: an overview*". In: G. Piatetsky-Shapiro and W.J Frawley (Eds.) Knowledge discovery in databases. 1-27 AAAI/MIT Press, 1991.

GONÇALVES, E. C., "*Data Mining – Novos Recursos nos Sistemas de Banco de Dados*", Disponível na Internet. http://www.devmedia.com.br/articles/viewcomp.asp?comp=5892. Acessado em 07 de Dezembro de 2007.

HAN, J. & KAMBER, M., "*Data Mining: Concepts and Techniques*", 1st ed., San Francisco California, USA, Morgan Kaufmann Publishers, 2001.

KONRAD, Rachel. "*Data mining: Digging user info for gold*". Disponível na Internet. http://zdnet.com.com/2100-11-528032.html?legacy=zdnn, Acessado em 23 de novembro de 2007.

KORTH, H. F., SILBERSCHATZ, A., "*Sistemas de Banco de Dados*", Makron Mooks, 2a. Edição Revisada, 1994.

LEÃO, R. O., SILVA, J. C., "*SQL Server 2000*", Série Banco de Dados, Ed Érica, 1ª edição, São Paulo, SP, 2002.

MACLENNAN, J., TANG, Z., "*Data Mining with SQL Server 2005*", Wiley Published, 1ª edição, Indianápolis, Indiana, EUA, 2005.

MOTTA, C. G. L., Mini-Curso C05 - Programa de Verão – Laboratório Nacional de Computação Científica – LNCC/MCT, Petrópolis, RJ, 15, 17 e 19 de janeiro de 2007.

MOTTA, C. G. L., "*Sistema Inteligente para Avaliação de Riscos em Vias de Transporte Terrestre*", Tese de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2004.

NAISBITT, John, Megatrends, 6th ed., WarnerBooks, NewYork, 1986.

REZENDE, R., "*Conceitos Fundamentais de Banco de Dados*". Disponível na Internet. http://devmedia.com.br/articles/viewcomp.asp?comp=1649&hl=daremos. Acessado em 11 de Novembro de 2007.

REZENDE, S. O., PUGLIESI, J. B., MELANDA E. A. & DE PAULA, M. F., "*Mineração de Dados*", In: Sistemas Inteligentes: Fundamentos e Aplicações, Barueri, SP, Brasil, Rezende, S. O. (coord.), Editora Manole Ltda., Cap. 12, pp. 307-336, 2003.

SILVA, T. M. A., "*Conceitos, técnicas, ferramentas e aplicações de Mineração de Dados para gerar conhecimento a partir de bases de dados*", TCC, UFPE, Recife, PE, Brasil, 2006.

SOUZA, R. C. F, "*Agrupamento Ótimo de Informações Não-Estruturadas Implementado em um Gerenciador de Banco de Dados Relacional*", Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2007.

VASCONCELOS, B. S., "*Mineração de Regras de Classificação com Sistemas de Banco de Dados Objeto-Relacional*", Dissertação de Mestrado, UNIVERSIDADE FEDERAL DE CAMPINA GRANDE, Campina Grande, Paraíba, Brasil, 2002.

WERFF, T. J., "*10 Emerging Technologies That Will Change the World*". Disponível na Internet. http://www.globalfuture.com/mit-trends2001.htm, Acessado em 20 de novembro de 2007.

WIECZOREK, Emilio Mario; LEAL, Eduardo. "*Caminhos e Tendências do Uso de Banco de Dados em Bioinformática*". IV Encontro de Estudantes de Informática do Estado do Tocantins. Palmas-TO, Brasil, 2002.

ANEXO A

IMPLEMENTAÇÃO DA CLASSE Metadata.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;
using Microsoft.SqlServer.DataMining.PluginAlgorithms;
namespace ClusteringSample
{
    [ComVisible(true)]
    [Guid("839EA6DE-160B-44D9-8CF6-300C983D7F3E")]
    [MiningAlgorithmClass(typeof(Algorithm))]
    public class Metadata : AlgorithmMetadataBase
    {

        protected MiningParameterCollection parameters;

        internal static MiningModelingFlag MainAttributeFlag =
        MiningModelingFlag.CustomBase + 1;

        public Metadata()
        {
                  parameters = DeclareParameters();
        }

            static public MiningParameterCollection DeclareParameters()
            {
                  MiningParameterCollection parameters
                    = new MiningParameterCollection();
                  MiningParameter param;
                  param = new MiningParameter(
                                    "PARAM1",
                                    "Integer Parameter 1 for training",
                                    "0",
                                    "[0.0, 100.0)",
                                    true,
                                    true,
                                    typeof(System.Int32));
                  parameters.Add(param);
                  param = new MiningParameter("PARAM2", typeof(String));
                  param.DefaultValue = "StringParamValue";
                  param.Description = "String Parameter 2 for training";
                  parameters.Add(param);
                  return parameters;
            }

        public override string GetServiceName()
        {
            return "MyPlugin_Algorithm";
```

```
}

public override string GetDisplayName()
{
    return "MyPlugin Algorithm";
}

public override string GetServiceDescription()
{
    return "MyPlugin Algorithm";
}

public override PlugInServiceType GetServiceType()
{
    return PlugInServiceType.ServiceTypeOther;
}

public override string GetViewerType()
{
    return string.Empty;
}

public override MiningScaling GetScaling()
{
    return MiningScaling.Medium;
}

public override MiningTrainingComplexity GetTrainingComplexity()
{
    return MiningTrainingComplexity.Low;
}

public override MiningPredictionComplexity
GetPredictionComplexity()
{
    return MiningPredictionComplexity.Low;
}

public override MiningExpectedQuality GetExpectedQuality()
{
    return MiningExpectedQuality.Low;
}

public override bool GetSupportsDMDimensions()
{
    return false;
}

public override bool GetSupportsDrillThrough()
{
    return false;
}

public override bool GetDrillThroughMustIncludeChildren()
{
    return false;
```

```
}

public override bool GetCaseIdModeled()
{
    return false;
}
public override MarginalRequirements GetMarginalRequirements()
{
    return MarginalRequirements.AllStats;
}
public override MiningParameterCollection
GetParametersCollection()
{
    if (parameters == null)
    {
        DeclareParameters();
    }
    return parameters;
}
public override object ParseParameterValue(int parameterIndex,
                      string parameterValue)
{
    object retVal = null;
    if (parameterIndex == 0)
    {
        int dVal = System.Convert.ToInt32(parameterValue);
        retVal = dVal;
    }
    else if (parameterIndex == 1)
    {
        string strVal = parameterValue;
        retVal = strVal;
    }
    else
    {
        throw new
                System.ArgumentOutOfRangeException("paramIndex");
    }
    return retVal;
}
public override MiningModelingFlag[] GetSupModelingFlags()
{
    MiningModelingFlag[] arModelingFlags =
        new MiningModelingFlag[] {
                MainAttributeFlag
                  };
    return arModelingFlags;
}
public override string GetModelingFlagName(MiningModelingFlag
                      flag)
{
    if (flag == MainAttributeFlag){
        return "MAIN";
    }
    else {
        throw new System.Exception("Unknown modeling flag : " +
```

```
            flag.ToString());
        }
    }
    public override MiningColumnContent[] GetSupInputContentTypes()
    {
        MiningColumnContent[] arInputContentTypes = new
        MiningColumnContent[] {
                    MiningColumnContent.Discrete,
                     MiningColumnContent.Continuous,
                    MiningColumnContent.Discretized,
                     MiningColumnContent.NestedTable,
                    MiningColumnContent.Key
              };
        return arInputContentTypes;
    }
    public override MiningColumnContent[] GetSupPredictContentTypes()
    {
        MiningColumnContent[] arPredictContentTypes = new
        MiningColumnContent[] {
                    MiningColumnContent.Discrete,
            MiningColumnContent.Continuous,
                    MiningColumnContent.Discretized,
            MiningColumnContent.NestedTable,
                    MiningColumnContent.Key
              };
        return arPredictContentTypes;
    }
    public override SupportedFunction[]
                     GetSupportedStandardFunctions()
    {
        SupportedFunction[] arFuncs = new SupportedFunction[] {

            SupportedFunction.PredictSupport,
            SupportedFunction.PredictHistogram,
            SupportedFunction.PredictProbability,
            SupportedFunction.PredictAdjustedProbability,
            SupportedFunction.PredictAssociation,
            SupportedFunction.PredictStdDev,
            SupportedFunction.PredictVariance,
            SupportedFunction.RangeMax,
            SupportedFunction.RangeMid,
            SupportedFunction.RangeMin,
            SupportedFunction.DAdjustedProbability,
            SupportedFunction.DProbability,
            SupportedFunction.DStdDev,
            SupportedFunction.DSupport,
            SupportedFunction.DVariance,
            SupportedFunction.IsDescendent,
            SupportedFunction.PredictNodeId,
            SupportedFunction.IsInNode,
            SupportedFunction.DNodeId,
              };
        return arFuncs;
    }
    public override void ValidateAttributeSet(AttributeSet
    attributeSet)
```

```
        {
            uint nCount = attributeSet.GetAttributeCount();
            int mainAttrs = 0;
            int inputAttrs = 0;

            for (uint nIndex = 0; nIndex < nCount; nIndex++)
            {
                bool thisAttIsInput = false;
                if ((attributeSet.GetAttributeFlags(nIndex) &
                    AttributeFlags.Input) != 0)
                {
                    inputAttrs++;
                    thisAttIsInput = true;
                }
                MiningModelingFlag[] modelingFlags =
                attributeSet.GetModelingFlags(nIndex);
                for (int flagIndex = 0; flagIndex < modelingFlags.Length;
                flagIndex++)
                {
                    if (modelingFlags[flagIndex] == MainAttributeFlag)
                    {
                        if (!thisAttIsInput)
                        {
                            string strMessage = string.Format(
                                                "{0} can only be applied
                                                  to an input attribute",
                            GetModelingFlagName(MainAttributeFlag));
                            throw new System.Exception(strMessage);
                        }
                        mainAttrs++;
                    }
                }
            }
            if (inputAttrs == 0)
                throw new System.Exception("At least one input attribute
                is required");
            if (mainAttrs > 1)
            {
                string strMessage = string.Format("Only one attribute may
                have the {0} flag",
                GetModelingFlagName(MainAttributeFlag));
                throw new System.Exception(strMessage);
            }
            return;
        }
        public override AlgorithmBase CreateAlgorithm(ModelServices
        model)
        {
            return new Algorithm();
        }
    }
}
```

ANEXO B

IMPLEMENTAÇÃO DA CLASSE Algorithm.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.SqlServer.DataMining.PluginAlgorithms;
namespace ClusteringSample
{
    public class Algorithm : AlgorithmBase
    {
        protected MiningParameterCollection algorithmParams;
        public TaskProgressNotification trainingProgress;
        protected System.UInt32 MainAttribute;
        protected double MainMean;
        protected bool MainContinuous;
        public InternalCluster[] Clusters;

        public Algorithm()
        {
            algorithmParams = Metadata.DeclareParameters();
            MainAttribute = 0;
            MainContinuous = false;
            MainMean = 0.0;
        }
        protected override void Initialize()
        {
            this.algorithmParams["PARAM1"].Value = 0;
            this.algorithmParams["PARAM2"].Value = "";
        }
        protected override void InsertCases(PushCaseSet caseSet,
        MiningParameterCollection trainingParams)
        {

        }
        protected override object GetTrainingParameterActualValue(int
        paramOrdinal)
        {
            return algorithmParams[paramOrdinal].Value;
        }
        protected override void LoadContent(PersistenceReader reader)
        {
        }
        protected override void SaveContent(PersistenceWriter writer)
        {
        }
        protected override void Predict(MiningCase inputCase,
        PredictionResult predictionResult)
        {
            AttributeGroup targetAttributes =
```

```
            predictionResult.OutputAttributes;
            targetAttributes.Reset();
            uint nAtt = AttributeSet.Unspecified;
            while (targetAttributes.Next(out nAtt))
            {
                AttributeStatistics result = new
                AttributeStatistics();
                result.Attribute = nAtt;
                AttributeStatistics trainingStats =
                this.MarginalStats.GetAttributeStats(nAtt);
                result.AdjustedProbability =
                trainingStats.AdjustedProbability;
                result.Max = trainingStats.Max;
                result.Min = trainingStats.Min;
                result.Probability = trainingStats.Probability;
                result.Support = trainingStats.Support;
                if (predictionResult.IncludeStatistics)
                {
                    for (int nIndex = 0; nIndex <
                    trainingStats.StateStatistics.Count; nIndex++)
                    {
                        bool bAddThisState = true;
                        if (trainingStats.StateStatistics[0]
                            .Value.IsMissing)
                        {
                            bAddThisState =
                            predictionResult.IncludeMissingState;
                        }
                        if (bAddThisState)
                        {
                            result.StateStatistics.Add(trainingStats.
                            StateStatistics[(uint)nIndex]);
                        }
                    }
                }

                if (predictionResult.IncludeNodeId)
                {
                    result.NodeId = "000";
                }
                predictionResult.AddPrediction(result);
            }
        }
        protected override AlgorithmNavigationBase GetNavigator(bool
        forDMDimensionContent)
        {
            return new AlgorithmNavigator(this,
            forDMDimensionContent);
        }
    }
}
```

ANEXO C

IMPLEMENTAÇÃO DA CLASSE AlgorithmNavigator.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.SqlServer.DataMining.PluginAlgorithms;
namespace ClusteringSample{
    class AlgorithmNavigator : AlgorithmNavigationBase{
        Algorithm   algorithm;
        bool        forDMDimension;
        int         currentNode;
        public AlgorithmNavigator(Algorithm currentAlgorithm, bool
        dmDimension){
            algorithm = currentAlgorithm;
            forDMDimension = dmDimension;
            currentNode = 0;
        }

        protected override bool MoveToNextTree(){
            return false;
        }
        protected override int GetCurrentNodeId(){
            return currentNode;
        }
        protected override bool ValidateNodeId(int nodeId){
            return (nodeId == 0);
        }
        protected override bool LocateNode(int nodeId){
            if (!ValidateNodeId(nodeId) )
                return false;
            currentNode = nodeId;
            return true;
        }
        protected override int GetNodeIdFromUniqueName(string
                                nodeUniqueName){
            int nNode = System.Convert.ToInt32(nodeUniqueName);
            return nNode;
        }
        protected override string GetUniqueNameFromNodeId(int nodeId)
        {
            return nodeId.ToString("D3");
        }
        protected override uint GetParentCount(){
            return 0;
        }
```

```
protected override void MoveToParent(uint parentIndex)
{}
protected override int GetParentNodeId(uint parentIndex){
    return 0;
}
protected override uint GetChildrenCount()
{
    return 0;
}
protected override void MoveToChild(uint childIndex)
{ }
protected override int GetChildNodeId(uint childIndex)
{
    return -1;
}
protected override NodeType GetNodeType()
{
    return NodeType.Model;
}
protected override string GetNodeUniqueName()
{
    return GetUniqueNameFromNodeId(currentNode);
}
protected override uint[] GetNodeAttributes()
{
    return null;
}
protected override double GetDoubleNodeProperty(NodeProperty
                           property)
{
    double dRet = 0;
    double dTotalSupport =
    algorithm.MarginalStats.GetTotalCasesCount();

    double dNodeSupport = 0.0;
    dNodeSupport = dTotalSupport;

    switch (property)
    {
        case NodeProperty.Support:
            dRet = dNodeSupport;
            break;
        case NodeProperty.Score:
            dRet = 0;
            break;
        case NodeProperty.Probability:
            dRet = dNodeSupport / dTotalSupport;
            break;
        case NodeProperty.MarginalProbability:
            dRet = dNodeSupport / dTotalSupport;
            break;
    }
    return dRet;
}

protected override string GetStringNodeProperty(NodeProperty
```

```
                                property)
        {
            string strRet = "";
            switch (property)
            {
                case NodeProperty.Caption:
                    {
                        strRet = algorithm.Model.
                        FindNodeCaption(GetNodeUniqueName());
                        if (strRet.Length == 0)
                        {
                            strRet = "All";
                        }
                    }
                    break;
                case NodeProperty.ConditionXml:
                    strRet = "";
                    break;
                case NodeProperty.Description:
                    strRet = "All Cases";
                    break;
                case NodeProperty.ModelColumnName:
                    strRet = "";
                    break;
                case NodeProperty.RuleXml:
                    strRet = "";
                    break;
                case NodeProperty.ShortCaption:
                    strRet = "All";
                    break;
            }
            return strRet;
        }
        protected override AttributeStatistics[] GetNodeDistribution()
        {
            int attStats = (int)algorithm.AttributeSet.
                           GetAttributeCount();
            AttributeStatistics[] marginalStats = new
                              AttributeStatistics[attStats];
            for (uint nIndex = 0; nIndex < attStats; nIndex++)
            {
                marginalStats[nIndex] = algorithm.MarginalStats.
                                        GetAttributeStats(nIndex);
            }
            return marginalStats;
        }
    }
}
```

ANEXO D

XMLA DE ACOPLAMENTO

```
<Alter
      AllowCreate="true"
      ObjectExpansion="ObjectProperties"
      xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
  <Object />
  <ObjectDefinition>
    <Server
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <Name>.</Name>
      <ServerProperties>
        <ServerProperty>
          <Name>DataMining\Algorithms\NOME_DO_ACOPLAMENTO\Enabled
          </Name>
          <Value>true</Value>
        </ServerProperty>
        <ServerProperty>
          <Name>DataMining\Algorithms\NOME_DO_ACOPLAMENTO\CLSID</Name>
          <Value>00000000-0000-0000-0000-000000000000</Value>
        </ServerProperty>
      </ServerProperties>
    </Server>
  </ObjectDefinition>
</Alter>
```

ANEXO E

IMPLEMENTACAO DA STORED PROCEDURE

```
/*************************************************************************/
/*************************************************************************/
--
--    IMPLEMENTACAO EM STORED PROCEDURE DO ALGORITMO DE CLUSTERIZAÇÃO
--    'K-MEANS'
--    PARA QUATRO ATRIBUTOS E PARA DUAS MÉTRICAS DIFERENTES
--
--    VICTOR MARQUES DE ASSIS
--
--    TRABALHO FINAL DE CONCLUSÃO DE CURSO / BOLSA DE INICIACAO
--    CIENTÍFICA LNCC/CNPQ
--
--    TÍTULO: ACOPLAMENTO DE TAREFAS DE MINERAÇÃO DE DADOS EM SGBD
--
--    JULHO DE 2008
--
--
/*************************************************************************/
/*************************************************************************/

/*************************************************************************/

               -- PROCEDIMENTOS E FUNÇÕES AUXILIARES --

/*************************************************************************/

-------------------------------------------------------------------------
--
--  'CHECKTABLE'
--  PROCEDURE QUE VERIFICA A EXISTÊNCIA DE UMA TABELA NO BANCO DE DADOS
--  PARÂMETROS DE ENTRADA:
--    '@TABLE_NAME' -> NOME DA TABELA A SER VERIFICADA
--  RETORNO:
--    INTEIRO INDICANDO QUE A TABELA NÃO EXISTE NO BANCO DE DADOS
--
-------------------------------------------------------------------------

CREATE PROCEDURE CHECKTABLE @TABLE_NAME VARCHAR(100)
AS
BEGIN
IF NOT EXISTS(SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME =
@TABLE_NAME)
      BEGIN
            SELECT 'NÃO EXISTE A TABELA "'+@TABLE_NAME+'" NO BANCO DE
                    DADOS' AS ERRO
            RETURN 1
      END
END
```

```
-------------------------------------------------------------------------
--
--  'CHECKCOLUMN'
--  PROCEDURE QUE VERIFICA A EXISTÊNCIA DE UMA COLUNA EM UMA TABELA
--  PARÂMETROS DE ENTRADA:
--    '@TABLE_NAME'  -> NOME DA TABELA
--    '@COLUNM_NAME' -> NOME DA COLUNA A SER VERIFICADA
--  RETORNO:
--    INTEIRO INDICANDO QUE A COLUNA NÃO EXISTE NA TABELA
--
-------------------------------------------------------------------------

CREATE PROCEDURE CHECKCOLUMN @TABLE_NAME VARCHAR(100),
                             @COLUNM_NAME VARCHAR(100)
AS
BEGIN
IF NOT EXISTS(SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME =
@TABLE_NAME AND COLUMN_NAME = @COLUNM_NAME)
      BEGIN
            SELECT 'A COLUNA "' + @COLUNM_NAME + '" NÃO EXISTE NA TABELA
                   ' + @TABLE_NAME AS ERRO
            RETURN 1
      END
END

-------------------------------------------------------------------------
--
--  'CHECKCLUSTERS'
--  PROCEDURE QUE VERIFICA A QUANTIDADE DE CLUSTERS SOLICITADA PELO
--  USUÁRIO
--  PARÂMETROS DE ENTRADA:
--    '@K' -> QUANTIDADE DE CLUSTERS
--  RETORNO:
--    INTEIRO INDICANDO UMA QUANTIDADE INVALIDA PARA '@K'
--
-------------------------------------------------------------------------

CREATE PROCEDURE CHECKCLUSTERS @K TINYINT
AS
BEGIN
IF (@K <= 1)
      BEGIN
            SELECT 'QUANTIDADE DE CLUSTERS DEVE SER MAIOR QUE UM' AS ERRO
            RETURN 1
      END
ELSE
      IF (@K > (SELECT COUNT(*) FROM #DATACOPY))
      BEGIN
            SELECT 'QUANTIDADE DE CLUSTERS DEVE SER MENOR QUE A
                    QUANTIDADE DE DADOS' AS ERRO
            RETURN 1
      END
END
```

```
-------------------------------------------------------------------------
--
--  'SHOWCLUSTERS'
--  PROCEDURE QUE EXIBE AS "COORDENADAS" DOS CENTRÓIDES DE CADA CLUSTER
--  PARÂMETROS DE ENTRADA:
--    NENHUM
--  RETORNO:
--    COORDENADAS DOS CENTRÓIDES
--
-------------------------------------------------------------------------

ALTER PROCEDURE SHOWCENTROIDES
AS
BEGIN
      SELECT * FROM #TB_CENTROIDES
      RETURN
END

-------------------------------------------------------------------------
--
--  'SHOWRESULT'
--  PROCEDURE QUE EXIBE O RESULTADO FINAL DA CLUSTERIZACAO
--  PARÂMETROS DE ENTRADA:
--      NENHUM
--  RETORNO:
--    ATRIBUTOS CLUSTERIZADOS
--
-------------------------------------------------------------------------

ALTER PROCEDURE SHOWRESULT
AS
BEGIN
      SELECT X,Y,W,Z,CLUSTER
      FROM #DATACOPY
      ORDER BY X
      RETURN
END

-------------------------------------------------------------------------
--
--  'EUCLIDEANDISTANCE'
--  FUNCTION QUE CALCULA A DISTÂNCIA EUCLIDIANA ENTRE DOIS PONTOS
--  PARÂMETROS DE ENTRADA:
--    '@X1,@Y1,@W1,@Z1' -> COORDENADAS DO PRIMEIRO PONTO
--    '@X2,@Y2,@W2,@Z2' -> COORDENADAS DO SEGUNDO PONTO
--  RETORNO:
--    VALOR DA DISTÂNCIA EUCLIDIANA ENTRE OS DOIS PONTOS
--
-------------------------------------------------------------------------

ALTER FUNCTION DBO.EUCLIDEANDISTANCE(
                              @X1 NUMERIC(20,4),@Y1 NUMERIC(20,4),
                              @W1 NUMERIC(20,4),@Z1 NUMERIC(20,4),
                              @X2 NUMERIC(20,4),@Y2 NUMERIC(20,4),
                              @W2 NUMERIC(20,4),@Z2 NUMERIC(20,4))
RETURNS NUMERIC(20,4)
```

```
AS
BEGIN
      DECLARE @DIST NUMERIC(20,4)
      SET @DIST = SQRT(POWER(@Z2-@Z1,2)+POWER(@W2-@W1,2)+POWER(@Y2-
@Y1,2)+POWER(@X2-@X1,2))
      RETURN @DIST
END

-----------------------------------------------------------------------
--
--  'MANHATTANDISTANCE'
--  FUNCTION QUE CALCULA A DISTÂNCIA DE MANHATTAN ENTRE DOIS PONTOS
--  PARÂMETROS DE ENTRADA:
--    '@X1,@Y1,@W1,@Z1' -> COORDENADAS DO PRIMEIRO PONTO
--    '@X2,@Y2,@W2,@Z2' -> COORDENADAS DO SEGUNDO PONTO
--  RETORNO:
--    VALOR DA DISTÂNCIA DE MANHATTAN ENTRE OS DOIS PONTOS
--
-----------------------------------------------------------------------

ALTER FUNCTION DBO.MANHATTANDISTANCE(
                              @X1 NUMERIC(20,4),@Y1 NUMERIC(20,4),
                              @W1 NUMERIC(20,4),@Z1 NUMERIC(20,4),
                              @X2 NUMERIC(20,4),@Y2 NUMERIC(20,4),
                              @W2 NUMERIC(20,4),@Z2 NUMERIC(20,4))
RETURNS NUMERIC(20,4)
AS
BEGIN
      DECLARE @DIST NUMERIC(20,4)
      SET @DIST = ABS(@X1-@X2)+ABS(@Y1-@Y2)+ABS(@W1-@W2)+ABS(@Z1-@Z2)
      RETURN @DIST
END

/*********************************************************************/

                    -- IMPLEMENTACAO DO ALGORITMO --

/*********************************************************************/

/*********************************************************************/
--  'KMEANS'
--  PROCEDURE QUE IMPLEMENTA O ALGORITMO DE MINERACAO 'KMEANS' PARA
--  QUATRO ATRIBUTOS
--  PARÂMETROS DE ENTRADA:
--    '@TAB'  -> NOME DA TABELA
--    '@COLX' -> PRIMEIRO ATRIBUTO
--    '@COLY' -> SEGUNDO ATRIBUTO
--    '@COLW' -> TERCEIRO ATRIBUTO
--    '@COLZ' -> QUARTO ATRIBUTO
--    '@K'    -> QUANTIDADE DE CLUSTERS
--    '@SHOW' -> INFORMA SE DEVEM SER EXIBIDAS APENAS AS COORDENADAS DOS
--               CLUSTERS
--  RETORNO:
--    DADOS CLASSIFICADOS E AGRUPADOS EM K CLUSTERS
--
/*********************************************************************/
```

```
ALTER PROCEDURE KMEANS @TAB  VARCHAR(100),
                       @COLX VARCHAR(100),
                       @COLY VARCHAR(100),
                       @COLW VARCHAR(100),
                       @COLZ VARCHAR(100),
                       @K    TINYINT,
                       @SHOW BIT = 0
AS
BEGIN
      SET NOCOUNT ON

      -- VARIÁVEL UTILIZADA PARA COMPOR A QUERY USADA PARA COPIAR OS
      -- DADOS DA TABELA ORIGINAL
      DECLARE @SQL VARCHAR(8000)

      -- VARIÁVEL UTILIZADA NAS VERIFICAÇÕES INICIAIS
      DECLARE @TESTE INT

      -- VERIFICAÇÕES INICIAIS
      EXEC @TESTE = CHECKTABLE @TAB
      IF @TESTE = 1 RETURN

      EXEC @TESTE = CHECKCOLUMN @TAB, @COLX
      IF @TESTE = 1 RETURN

      EXEC @TESTE = CHECKCOLUMN @TAB, @COLY
      IF @TESTE = 1 RETURN

      EXEC @TESTE = CHECKCOLUMN @TAB, @COLW
      IF @TESTE = 1 RETURN

      EXEC @TESTE = CHECKCOLUMN @TAB, @COLZ
      IF @TESTE = 1 RETURN

      -- CRIAÇÃO DAS TABELAS TEMPORÁRIAS UTILIZADAS PELA PROCEDURE

      -- TABELA PARA ARMAZENAR A CÓPIA DOS DADOS
      CREATE TABLE #DATACOPY
      (
            ID INT IDENTITY(1,1),
            X NUMERIC(20,4),
            Y NUMERIC(20,4),
            W NUMERIC(20,4),
            Z NUMERIC(20,4),
            CLUSTER TINYINT
      )

      -- TABELA PARA A MATRIZ DE DISTÂNCIAS
      CREATE TABLE #DISTANCEMATRIX
      (
            ID_PONTO INT,
            ID_CLUSTER INT,
            DISTANCIA NUMERIC(20,4)
      )
```

```
-- TABELA AUXILIAR
CREATE TABLE #A
(
      ID_PONTO INT,
      DISTANCIA NUMERIC(20,4)
)

-- TABELA AUXILIAR
CREATE TABLE #B
(
      ID_PONTO INT,
      DISTANCIA NUMERIC(20,4),
      ID_CLUSTER INT
)

-- TABELA AUXILIAR
CREATE TABLE #F
(
      ID INT,
      CX NUMERIC(20,4),
      CY NUMERIC(20,4),
      CW NUMERIC(20,4),
      CZ NUMERIC(20,4)
)

-- TABELA PARA ARMAZENAR AS COORDENADAS DOS CENTRÓIDES
CREATE TABLE #TB_CENTROIDES
(
      ID INT,
      CX NUMERIC(20,4),
      CY NUMERIC(20,4),
      CW NUMERIC(20,4),
      CZ NUMERIC(20,4)
)

-- COPIANDO OS DADOS DA TABELA ORIGINAL
SET @SQL = 'SELECT DISTINCT CONVERT(NUMERIC(20,4),' + @COLX + ')'
SET @SQL = @SQL + ', CONVERT(NUMERIC(20,4),' + @COLY + ')'
SET @SQL = @SQL + ', CONVERT(NUMERIC(20,4),' + @COLW + ')'
SET @SQL = @SQL + ', CONVERT(NUMERIC(20,4),' + @COLZ + ')'
SET @SQL = @SQL + ', 1 AS CLUSTER'
SET @SQL = @SQL + ' FROM ' + @TAB

INSERT #DATACOPY
EXEC (@SQL)

-- VERIFICAÇÃO DA QUANTIDADE DE CLUSTERS
EXEC @TESTE = CHECKCLUSTERS @K
IF @TESTE = 1 RETURN


-- PASSO 1: FORNECER UM VALOR PARA OS CENTRÓIDES

-- OS K CENTRÓIDES RECEBEM VALORES INICIAIS;
-- PARA ISSO, SÃO ESCOLHIDOS OS K PRIMEIROS PONTOS;
```

```
        -- PARA QUE O ALGORITMO INICIE O PROCESSAMENTO, TODOS OS PONTOS SÃO
        -- COLOCADAS EM UM CENTRÓIDE QUALQUER.

        SET ROWCOUNT @K

        INSERT #TB_CENTROIDES
        SELECT ID,X,Y,W,Z
        FROM #DATACOPY

        SET ROWCOUNT 0

        -- INÍCIO DO LOOP
        DECLARE @LOOP BIT
        SET @LOOP = 0
        WHILE @LOOP = 0
        BEGIN

              -- PASSO 2: GERAR A MATRIZ COM AS DISTÂNCIAS ENTRE TODOS OS
              -- PONTOS E OS CENTRÓIDES

              -- DISTÂNCIA ENTRE CADA PONTO E OS CENTRÓIDES Eh DETERMINADA;
              -- Eh POSSÍVEL ALTERAR A MÉTRICA DO CÁLCULO DA DISTÂNCIA.

              TRUNCATE TABLE #DISTANCEMATRIX
              TRUNCATE TABLE #A
              TRUNCATE TABLE #B

              INSERT      #DISTANCEMATRIX
              SELECT A.ID, B.ID,
DBO.EUCLIDEANDISTANCE(A.X,A.Y,A.W,A.Z,B.CX,B.CY,B.CW,B.CZ)

              -- DESCOMENTE A LINHA ABAIXO E COMENTE A LINHA ACIMA PARA
              -- MUDAR A MÉTRICA
              --  SELECT A.ID,B.ID,DBO.MANHATTANDISTANCE(A.X,A.Y,A.W,A.Z,B.CX,B.CY,B.CW,B.CZ)

              FROM #DATACOPY AS A, #TB_CENTROIDES AS B

              INSERT #A
              SELECT ID_PONTO, MIN(DISTANCIA)
              FROM #DISTANCEMATRIX
              GROUP BY ID_PONTO

              INSERT #B
              SELECT ID_PONTO, DISTANCIA,
                  (SELECT TOP 1 ID_CLUSTER FROM #DISTANCEMATRIX AS B WHERE
                   A.ID_PONTO = B.ID_PONTO AND A.DISTANCIA = B.DISTANCIA)
                     AS CLUSTER
              FROM #A AS A

              -- PASSO 3: COLOCAR CADA PONTO NOS CLUSTERS DE ACORDO COM SUA
              -- DISTÂNCIA

              -- OS PONTOS SÃO AGRUPADOS BASEADOS NA MENOR DISTÂNCIA ENTRE
              -- ELES E O CENTRÓIDE DE CADA CLUSTER;
              -- SE NENHUM PONTO FOR INCORPORADO A UMA CLASSE DIFERENTE DA
              -- QUE SE ENCONTRAVA ANTES DESSE PASSO,
```

```
            -- O ALGORITMO TERMINA.

            UPDATE #DATACOPY
            SET CLUSTER = Z.ID_CLUSTER
            FROM #DATACOPY T INNER JOIN #B Z
            ON T.ID = Z.ID_PONTO
            AND CLUSTER <> Z.ID_CLUSTER

            IF @@ROWCOUNT = 0
            BEGIN
                  SET @LOOP = 1
                  CONTINUE
            END

            -- PASSO 4: REFINAR OS CENTRÓIDES DE CADA CLASSE

            -- CALCULAR OS NOVOS CENTRÓIDES PARA CADA CLUSTER USANDO A
            -- MÉDIA DAS COORDENADAS PARA CADA PONTO;
            -- SOMENTE PARA OS GRUPOS QUE POSSUEM MAIS DE 1 PONTO;

            TRUNCATE TABLE #F

            INSERT #F
            SELECT CLUSTER,AVG(X),AVG(Y),AVG(W),AVG(Z)
            FROM #DATACOPY
            GROUP BY CLUSTER
            HAVING COUNT(*) > 1

            UPDATE #TB_CENTROIDES
            SET CX = F.CX,CY = F.CY,CW = F.CW,CZ = F.CZ
            FROM #TB_CENTROIDES A INNER JOIN #F F
            ON A.ID = F.ID

            -- PASSO 5: REPETIR ITERATIVAMENTE O REFINAMENTO DO CÁLCULO
            -- DAS COORDENADAS DOS CENTRÓIDES

            -- VOLTAR AO PASSO 2

      END -- WHILE

      -- EXIBIR OS RESULTADOS

      -- SE DESEJA EXIBIR SOMENTE AS COORDENADAS DOS K CLUSTERS
      IF @SHOW = 1
      BEGIN
            EXEC SHOWCENTROIDES
            RETURN
      END

      EXEC SHOWRESULT

      SET NOCOUNT OFF
      RETURN
END – KMEANS
```