UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE

UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

KARLA HARYANNA SANTOS MOURA

OpMM - FERRAMENTA PARA EXTRAÇÃO DE CONHECIMENTO UTILIZANDO PRIMITIVAS TEMPORAIS: UM ESTUDO DE CASO EM PACIENTES INTERNADOS EM UTI

MOSSORÓ – RN

2013

KARLA HARYANNA SANTOS MOURA

OpMM - FERRAMENTA PARA EXTRAÇÃO DE CONHECIMENTO UTILIZANDO PRIMITIVAS TEMPORAIS: UM ESTUDO DE CASO EM PACIENTES INTERNADOS EM UTI

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação – associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dra. Cicília Raquel Maia Leite – UERN.

MOSSORÓ – RN

2013

Catalogação da Publicação na Fonte.

Universidade do Estado do Rio Grande do Norte.

Moura, Karla Haryanna Santos

OPMM - Ferramenta Para Extração De Conhecimento Utilizando Primitivas Temporais: Um Estudo De Caso Em Pacientes Internados Em UTI/ Karla Haryanna Santos Moura – Mossoró, RN, 2013.

68 f. Il.

Orientador(a): Prof. Dra. Cicília Raquel Maia Leite

Dissertação de Mestrado (Mestre). Universidade do Estado do Rio Grande do Norte e Universidade Federal Rural do Semi-Árido. Programa de Pós-Graduação em Ciência da Computação.

1. Banco de Dados - Dissertação. 2. Unidade de Terapia Intensiva - UTI. 3.

Bibliotecário: Sebastião Lopes Galvão Neto – CRB - 15/486

KARLA HARYANNA SANTOS MOURA

OpMM - FERRAMENTA PARA EXTRAÇÃO DE CONHECIMENTO UTILIZANDO PRIMITIVAS TEMPORAIS: UM ESTUDO DE CASO EM PACIENTES INTERNADOS EM UTI

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 10 / 06 / 2013.

Banca Examinadora

_____

Prof. Dra. Cicília Raquel Maia Leite – UERN

Presidente

_____

Prof. Dr. Carlos Heitor Pereira Liberalino – UERN

Segundo Membro

_____

Prof. Dra. Ana Maria Guimarães Guerreiro – UFRN

Terceiro Membro

Ao meu grande companheiro Marcilio, que sempre me deu apoio nas atividades acadêmicas e as minhas filhas Mirna e Maria Alice, que me confortam com seus sorrisos sempre, minha família.

Agradecimentos

A Deus, por ser meu condutor e edificar minha vida, me mantendo capaz de ter sabedoria e discernimento em tudo que faço.

A Professora orientadora Cicília Raquel, que não hesitou em momento algum a juntas elaborarmos esse projeto, e persistência que teve em todos os sentidos.

A minha mãe e meu pai, os quais se dedicaram com grande esforço para minha educação, me fizeram uma pessoa digna para a sociedade, pela constante colaboração em educar minhas filhas.

Aos meus irmãos Kallyo e Hammara, que acompanham minha jornada e compartilham os bons momentos.

Aos Professores da banca Heitor Liberalino e Ana Maria pela disponibilidade e prontidão.

Ao grupo de estudo de engenharia de Software GES, Marlos Antonio, Lenardo, Ronnison Regis, Denilson em especial a Kayo Luann e Suellem Stephanne, que participaram desse processo de conhecimento.

Ao Professor Marcelino, pela contribuição em várias fases de minha vida acadêmica.

Ao Professor Pedro Fernandes, pelas lições constantemente apresentadas. Ao qual faz merecer o título de dirigente máximo da UERN.

Ao amigos Maximilliano e Estela, por fazer parte de minha família mesmo sem grau de parentesco.

A tia Socorro e Rivânia grandes companheiras, que com incentivo e demonstração de preocupação sempre, me fortaleceram com a esperança que tudo daria certo.

A UERN e UFERSA, pela formalização do mestrado.

Resumo

A quantidade de informações armazenadas em banco de dados está crescendo em uma velocidade maior que a capacidade computacional e humana de interpretar esses amontoados de dados. Logo, a medida que o uso de sistemas de banco de dados cresce, os usuários exigem funcionalidades adicionais, com a finalidade de facilitar a implementação de aplicações do usuário mais avançadas e mais complexas. A busca por informações contidas em bancos de dados tem acontecido de forma ampla e generalizada e, mais especificamente, banco de dados médico, foco deste trabalho, tem sido alvo de muitas pesquisas, como: mineração de dados, segurança, ética e/ou sigilo dos dados, modelo de predição, sistemas de apoio a decisão, utilização de técnicas inteligentes, entre outros. Diante do exposto, o objetivo deste trabalho é o desenvolvimento de uma ferramenta para extração de conhecimento utilizando primitivas temporais, intitulada OpMM, baseado nas etapas do *Knowledge Discovery Database* (KDD), a fim de subsidiar profissionais da área médica para auxílio ao diagnóstico médico. A ferramenta está baseada em técnicas e algoritmos de mineração de dados, através do algoritmo de árvore de decisão. O OpMM tem como princípio a transformação dos dados em informações, por meio do conhecimento dos especialistas e dos parâmetros de normalidade dos sinais vitais de pacientes. Para validação da ferramenta utilizou-se um estudo de caso em pacientes internados em UTI através do banco dados público *Multiparameter Intelligent Monitoring in Intensive Clinical* (MIMIC II).

Palavras-chave: Mineração de dados, Unidade de Terapia Intensiva, Descoberta de Conhecimento em Banco de Dados, Sinais Vitais.

Abstract

The amount of information stored in database is growing at a speed greater than the computational and human capacity to interpret these heaps of data. Thus, as the use of database systems grows users require additional functionality in order to facilitate the implementation of more advanced and more complex user applications. The search for information contained in databases has been broadly and widespread and, more specifically medical database, focus of this work has been the subject of much research, such as: data mining, security, ethics and/or data confidentiality, prediction model, decision support systems, intelligent techniques among others. Based on the considerations above, the objective of this work is to develop a tool for extracting knowledge using primitive temporal titled OpMM, based on the steps of the Knowledge Discovery Database (KDD), assisting medical professionals to aid medical diagnosis. The tool is based on techniques and data mining algorithms, through the decision tree algorithm. The OpMM has as principle the transformation of data into information through the of the experts' knowledge and normality parameters of the patient vital signs. Case study in patients admitted to the ICU through public data bank Multiparameter Intelligent Monitoring in Intensive Clinical (MIMIC II) was used to validate the tool.

Keywords: Data Mining, Intensive Care Unit, Knowledge Discovery in Database, Vital Signs.

Lista de Tabelas

Lista de Figuras

Lista de Siglas

τ	Transação
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
BRP	*Bioengineering Research Partnership*
BIDMC	*Boston's Israel Deaconess Medical Center*
CID	Classificação Internacional de Doenças
CTI	Centro de Tratamento Intensivo
FC	Frequência Cardíaca
FR	Frequência Respiratória
HGT	Hemoglicoteste
JDBC	*Java Database Connectivity*
KDD	*Knowledge Discovery Database*
LC-BDTR	Linguagem de Consulta em Banco de Tempo-Real
OpMM	*Open Medical Mining*
MIMIC	*Multiparameter Intelligent Monitoring in Intensive Clinical*
NIBIB	*National Institute of Biomedical Imaging and Bioengineering*
NIH	*National Institutes of Health*
PA	Pressão arterial
SGBD	Sistema de Gerenciamento de Banco de Dados
SGBD-TR	Sistema de Gerenciamento de Banco de Dados em Tempo-Real
STR	Sistemas em Tempo-Real
UTI	Unidade de Tratamento Intensivo

Sumário

Capítulo 1

Introdução

1.1 Contextualização

A quantidade de informações armazenadas em banco de dados está crescendo em uma velocidade maior que a capacidade computacional e humana de interpretar os dados. O uso universalizado de leitores magnéticos, aquisição automática de dados, equipamentos médicos e a Internet, estão produzindo grande quantidade de dados numa rapidez sem precedentes. E ainda, o custo do poder de processamento dos computadores tem diminuído drasticamente. Somado ao crescimento de dados, as tecnologias de software, sistemas de gerenciamento de banco de dados têm apresentado grandes avanços. Contudo, a tecnologia da rede mundial tem avançado com grande rapidez, permitindo o acesso aos dados de forma transparente.

Com a necessidade de transformar estes *terabytes* de dados em informações significativas, utiliza-se mineração de dados em diversas áreas, como: análise de dados médicos, análise de riscos, marketing direcionado, controle de qualidade, análise de dados científicos, entre outros. Sua utilização permite avanços tecnológicos, descobertas científicas, e soluções na execução de tarefas, contribuindo dessa forma na agilidade dos processos da organização, como até mesmo redução de custos em geral.

Pensando na área da saúde como um todo, em operações que poderiam extrair melhores observações em relação a certo tipo de doença, e nos processos de automação hospitalar, torna-se imprescindível o apoio de tecnologias no auxílio das tomadas de decisão.

Com o surgimento de tecnologias capazes de automatizar ambientes hospitalares e capturar tais informações, é possível aperfeiçoar processos da área da saúde, sendo, portanto uma forma de minimizar riscos, contribuindo de maneira efetiva para a melhoria qualidade da saúde da população.

1.2 Problemática

O ambiente hospitalar requer, demanda de tecnologia, desde o simples atendimento em ambulatório até os equipamentos de monitoramentos de pacientes internados em Unidade de Terapia Intensiva (UTI). Através dessas tecnologias abordadas na área médica, é que surge a necessidade de integração de dados. Todas as informações geradas a partir de um paciente em atendimento contribuem para o aperfeiçoamento dos resultados médico.

Assim, a necessidade de inovação tecnológica e consequentemente o surgimento de novos produtos nessa área é vital, visto que a maioria dos hospitais ainda encontra-se realizando os procedimentos de forma manual, aspecto que dificulta o controle dos dados e o gerenciamento das informações, podendo levar a erros graves em relação à vida dos pacientes.

1.3 Trabalhos Relacionados

Com a necessidade de desenvolver novas tecnologias no âmbito da saúde, principalmente pela conjuntura de saúde e tecnologia, que vem se tornando uma grande área de pesquisas, e ainda necessitando de muitas definições e integração com a área médica. Seguindo esse contexto, as pesquisas foram direcionadas para Banco de Dados em Tempo-Real, Mineração de Dados e automação hospitalar. Vários trabalhos abordam as integrações de tecnologias e problema médico.

Steiner (2006) trata de todo o processo de "Descoberta de Conhecimento em Bases de Dados" *(Knowledge Discovery in Databades, KDD)* em todas as suas etapas, mostrando a influência da análise exploratória dos dados no desempenho das técnicas de mineração, quanto à classificação de novos padrões, por meio de um problema médico.

Virone (2008) propôs uma extensão de um modelo em um programa de software personalizado, estendido para prever a progressão de uma determinada atividade com base em um período, detectando eventos inesperados, como lentidão ou interrupções que possam indicar mudanças no estado de saúde, cognitivas ou motoras do paciente (VIRONE, 2008).

Minnie (2011) analisa as funções de um contador automático de células sanguíneas a partir de um laboratório de patologia clínica e propõe a aplicação da fase de pré-processamento de dados do *Knowledge Discovery in Databases* (KDD), como a limpeza de dados, seleção de dados e transformação de dados no contador de células sanguíneas dados para melhorar a qualidade do sistema.

Moura (2007) utilizou a ferramenta de mineração de dados *Spotfire*, em dados sobre a assistência a pacientes que fizeram exames de glicemia em um período de seis anos, para ilustrar as idades que estão realmente procurando tratamento adequado. Os dados selecionados foram preparados e avaliados e seguiu as etapas do processo de KDD. A utilização do algoritmo apropriado facilitou a sua procura de padrões nos resultados.

Já Tsumoto (2007) desenvolveu uma aplicação de mineração de dados para gerenciamento de risco médico para evitar acidentes e alcançar a segurança médica. O autor classifica os erros em sistemáticos e erros pessoais, que podem ser evitados por ações adequadas, e a mineração de dados está prevista como uma ferramenta para a análise dos erros.

As tecnologias voltadas à saúde abrangem desde os sistemas de gestão hospitalar quanto às aplicações de auxílio à necessidade de informações a respeito do diagnóstico do paciente, disponibilizando ao médico formas adaptativas e personalizadas de utilização, configuração e controle do prontuário eletrônico do paciente.

Abe (2007) criou um ambiente que integra métodos de séries temporais com padrão de extração, métodos de indução de regras e métodos de avaliação de regras com interface que facilite a interação homem-máquina. Como validação, realizou-se um estudo de caso com as regras de séries temporais de uma base de dados de teste bioquímico de sangue ou urina em pacientes com hepatite crônica.

A necessidade de análise das características dos dados médicos, favoreceu a Zhao (2010) propor um quadro geral para a mineração de dados médicos, que consiste na seleção, pré-processamento, mineração e de avaliação de conhecimentos em dados médicos.

Machado e Augustin (2013, p. 50), ilustra o uso da computação pervasiva para otimizar e automatizar as atividades clínicas através da prototipação de uma arquitetura de software para um sistema de informação em saúde ubíquo.

O contexto mais dirigido à automação hospitalar, as principais pesquisas estão orientadas ao desenvolvimento de sistemas com registros de acompanhamento de pacientes, monitoramento de pacientes, utilização de tecnologias para melhorar o nível da segurança, comunicação, usabilidade, confiabilidade e desempenho das aplicações, novos ambientes de cuidados à saúde através da telemedicina (telediagnóstico e telemonitoramento). Sobre a perspectiva do monitoramento de pacientes, os trabalhos seguem a linha de automatização do processo de monitoramento.

Segundo Baura (2004), o monitoramento de pacientes é um processo que demanda observação contínua. Este aspecto é um requisito que exige dos sistemas de monitoramento garantias de disponibilidade. Deste modo, verifica-se a relevância deste tema no contexto da automação hospitalar. Isso porque, trata-se de um processo que ao ser automatizado deve considerar os requisitos pertinentes ao monitoramento de pacientes de forma a garantir a qualidade do serviço, a fim de preservar efetivamente a vida humana.

Diante dos trabalhos descritos, muitas pesquisas estão sendo realizadas, entretanto o objeto de estudo é agregar conceitos, técnicas e tecnologias de forma diferenciada dos demais sistemas apresentados, combinando um conjunto de técnicas de sistemas inteligentes que visa realizar aquisição, gerenciamento de dados em tempo real e processamento, classificação de dados, contribuindo para o envio de informações úteis e em tempo hábil para quem de direito.

1.4 Objetivo Geral

Considerada toda a problemática exposta, objetiva-se nesta dissertação o desenvolver uma ferramenta para extração de conhecimento utilizando primitivas temporais, intitulada OpMM, baseado nas etapas do *Knowledge Discovery Database* (KDD), com a finalidade de subsidiar profissionais da área médica para auxílio ao diagnóstico médico. A ferramenta está baseada em técnicas de pré-processamento, classificação e mineração de dados através do algoritmo de árvore de decisão.

1.5 Relevância

A relevância desta dissertação é disponibilizar uma ferramenta para extração de conhecimento utilizando primitivas temporais e permitir, através de pesquisas e prática do processo do KDD, o estudo das potencialidades e aplicabilidade de técnicas e algoritmos, para obter informações preciosas que estavam escondidas na base de dados e possam ser transformados em ações para o auxílio ao diagnóstico médico.

1.6 Organização do Trabalho

A dissertação está organizada da seguinte forma: Capítulo 2 descreve o Sistema de Gerenciamento de Banco de Dados em Tempo-Real tratando das características dos dados e transações em Tempo-Real. O Capítulo 3 demonstra todo o processo de Descoberta de Conhecimento em Banco de Dados, com os conceitos fundamentais de mineração de dados, situando o trabalho no escopo mais específico da mineração de dados médico. O Capítulo 4 descreve-se a implementação e a interface, além de mostrar o estudo de caso junto aos resultados do trabalho. Por último, no Capítulo 5 apresenta-se as conclusões, contribuições e perspectivas futuras.

Capítulo 2

Sistema de Gerenciamento de Banco de Dados em Tempo-Real

2.1 Introdução

O Sistema de Gerenciamento de Banco de Dados (SGBD) estabelece uma padronização de adequação dos dados e são projetados para gerenciar um grande volume de dados. Obedecendo a um conjunto de definições, como controle de acesso de usuários, devendo rejeitar toda tentativa da quebra de segurança e integridade, definida pelo administrador do banco de dados. Podendo assim, monitorar toda e qualquer requisições de usuários e o controle de transações como recuperação e concorrência de dados. Elsmari (2011), cita que de forma mais direta que:

> O SGBD é um *sistema de software de uso geral* que facilita o processo de *definição, construção, manipulação e compartilhamento* de banco de dados entre diversos usuários e aplicações. Definir um banco de dados envolve especificar os tipos, estrutura e restrição dos dados a serem armazenados. (ELSMARI, 2011, p. 3)

A definição de armazenamento do banco de dados é totalmente controlada pelo SGBD, onde as manipulações dos dados podem ocorrer de diversas maneiras, como uma simples consulta ao banco, ou a modificação dos dados. Assim, a operabilidade acontece com o controle do sistema de gerenciamento do banco de dados, onde uma vez compartilhado, podem ocorrer acessos simultâneos de usuários e programas. Favorecendo a um ambiente eficiente para recuperação e armazenamento das informações do banco de dados.

Os Sistemas de Gerenciamento de Banco de Dados (SGBD) convencionais são projetados para executar transações concorrentes enquanto mantém a consistência lógica dos dados e transações. Por outro lado, os Sistemas de Gerenciamento de Banco de Dados em Tempo-Real (SGBD-TR) são projetados para executar transações concorrentes com restrições temporais e garantir a consistência lógica e temporal dos dados e transações.

2.2 Características de um SGBD-TR

Os SGBD-TR podem ser vistos como a integração de um SGBD convencional com um sistema em Tempo-Real (STR). Como um SGBD processa transações e garante a integridade lógica dos dados e como um STR garante restrições temporais das transações. Portanto, um SGBD-TR possui como principais características o tratamento de dados com validade temporal e transações com restrições explícitas de tempo de execução (PERKUSICH; TURNNEL; PERKUSICH, 1999). Estas características são úteis para aplicações com tempo crítico que necessitam coletar, modificar e recuperar grandes volumes de dados compartilhados (TESANOVIC et al., 2002).

Assim, um SGBD-TR deve fornecer capacidade para gerenciar transações em tempo-real, onde as propriedades ACID (acrônimo para Atomicidade, Consistência, Isolamento e Durabilidade) são aplicadas seletivamente. Ele ainda deve fornecer suporte para os usuários especificarem as restrições temporais dos dados e das transações, isto é, um intervalo de tempo durante o qual um dado é considerado válido, e o tempo máximo que deve ser utilizado para executar uma transação. Nas Seções seguintes são apresentadas as principais características dos dados e das transações de um SGBD-TR.

2.3 Características dos Dados

Os dados usados em um SGBD-TR devem refletir o estado real do ambiente da aplicação, portanto a estrutura desses dados indica que os valores gravados são válidos apenas por um determinado intervalo de tempo. Assim, os intervalos de tempo especificam a validade temporal dos dados. Desta forma, um SGBD-TR busca garantir a consistência temporal dos mesmos através da validação desses intervalos. A consistência temporal pode ser medida através da consistência absoluta dos dados e através da consistência relativa dos dados.

A consistência absoluta é a medida entre o estado real do ambiente e como ele é armazenado no banco de dados. Então, um item de dado deve ser gravado dentro de intervalos de tempo que garantam que ele reflita o estado real do ambiente. Essa medida surge da

necessidade de manter a visão do sistema de controle consistente com o estado real do ambiente.

A consistência relativa é a medida entre os dados que são usados na computação de outros dados. Então, um conjunto de itens de dados deve ser gravado dentro de um mesmo intervalo de tempo, que possa representar aproximadamente o mesmo instante de tempo. Essa medida surge da necessidade de produzir novos dados a partir de dados gravados em tempos aproximados.

2.4 Características das Transações

A maioria das transações de um SGBD-TR usa dados com validade temporal, conseqüentemente as mesmas precisam estar associadas às restrições temporais para sua execução, tais como: tempo de início, tempo de término, tempo de processamento, tempo computacional, prazo e periodicidade.

As transações podem ser de três tipos: estrito *(hard),* suave *(soft)* e firme *(firm).*

- Estrito - Uma transação é do tipo estrito quando qualquer resultado que é produzido após seu prazo é inútil para o sistema e é considerada uma falha fatal;
- Suave - Uma transação é do tipo suave quando o não cumprimento de seus prazos não acarreta problemas sérios, no entanto, o desempenho do sistema diminui à medida que várias transações com prazos suaves deixam de atender suas restrições de tempo;
- Firme - Uma transação é do tipo firme se a perda do prazo final não gerar nenhum efeito ou valor negativo para o sistema. Geralmente estas transações são recomeçadas quando perdem seu prazo final.

Muitas transações em um SGBD-TR são usadas para ler e gravar dados coletados de dispositivos, tais como sensores, visando monitorar um determinado ambiente e gerenciar os eventos ocorridos em tais ambientes.

Geralmente, em SGBD-TR, essas transações podem ser executadas de três diferentes formas: periódica, aperiódica e esporádica.

- Periódica - Uma transação é do tipo periódica quando ela é executada repetidamente em um intervalo de tempo regular para desempenhar uma função do sistema;
- Aperiódica - Uma transação é do tipo aperiódica quando não existe um intervalo de tempo regular para a execução da mesma;
- Esporádica - Uma transação é do tipo esporádica quando ela possui intervalo regular, podendo ou não ser executada.

Ramamrithman (1993) caracteriza as transações de um SGBD-TR baseado na natureza das transações de três diferentes maneiras:

Podem ser caracterizadas de acordo com as origens das restrições temporais impostas às transações. Algumas restrições vêm dos requisitos de consistência temporal dos dados, e algumas vêm dos requisitos impostos pelo ambiente. Geralmente, elas têm a forma de requisitos de periodicidade. Por exemplo, a cada 30 segundos verifique se o valor da temperatura medida pelo sensor de temperatura do ambiente X passou do limite;

Elas são caracterizadas baseadas nos prazos impostos às transações aperiódicas. Por exemplo, se o valor da temperatura medida pelo sensor de temperatura for maior que 37 graus, dentro de 20 segundos, dispare o alarme;

Finalmente, elas são caracterizadas baseadas no efeito de perder seu prazo como estrita, suave e firme, exatamente como acontece para um STR.

Quanto ao tipo uma transação pode ser classificada como: transação de sensor (ou escrita), transação de atualização e transação de leitura.

- Transação de sensores - é uma transação que obtém o estado do ambiente e escreve os valores no banco de dados;
- Transação de atualização - é uma transação que pode ler e escrever no banco de dados;
- Transação de leitura - é uma transação que pode ler dados do banco de dados.

2.4.1 Propriedades Atomicidade, Consistência, Isolamento e Durabilidade

Um projeto de banco de dados convencional deve garantir a manutenção da consistência lógica dos dados, isto é, evitar que diferentes transações escrevam informações contraditórias no banco de dados, visando manter a consistência interna do sistema. Para garantir a consistência interna, deve-se implementar corretamente todas as transações buscando assegurar as propriedades ACID, definidas a seguir, (ELSMARI, 2011):

- Atomicidade - Uma transação deve ser totalmente executada ou nenhum passo dela deve ser considerado, ou seja, qualquer passo realizado deve ser desfeito;
- Consistência - A execução de uma transação deve sempre transformar o estado consistente de um banco de dados em outro estado consistente;
- Isolamento - Os efeitos gerados por uma transação no banco de dados não devem ser visíveis por nenhuma outra transação até que ela seja comprometida;
- Durabilidade - Os efeitos de uma transação devem ser permanentes.

Tais propriedades são bastante restritivas, portanto as mesmas foram redefinidas para.

Sistema de Gerenciamento de Banco de Dados em Tempo Real:

- Atomicidade - A execução atômica é seletivamente aplicada às subtransações que necessitam tratar com dados totalmente consistentes, ao invés de aplicá-la à transação toda. Por exemplo, considere que uma transação que está atualizando dados a respeito de um ambiente, perde seu prazo e deve parar de executar. Geralmente, é desnecessário desfazer as operações, uma vez que os valores anteriores também estão desatualizados. Além do mais, pode ser mais interessante dispor de um banco de dados parcialmente ou recentemente atualizado que um banco de dados totalmente desatualizado.
- Consistência - Algumas transações podem ter restrições temporais e podem usar dados que precisam ser temporalmente consistentes. Assim, os dados gerados por uma transação não terminada podem ser vistos por outras transações para evitar que se tornem desatualizados ou que não satisfaçam suas restrições temporais.

- Isolamento - As transações podem precisar se comunicar e sincronizar com outras transações de forma a executar funções de controle, portanto, suas ações podem ser vistas por outras transações mesmo antes delas terem terminado.
- Durabilidade - A propriedade de durabilidade, alguns dados têm validade temporal e não precisam ser gravados. Por outro lado, o banco de dados deve refletir o estado do ambiente, portanto, é fácil recriá-lo a partir da leitura dos sensores, ao invés de recriá-lo no tempo em que ocorreu uma falha, pois esses dados provavelmente já serão inválidos.

2.5 Dados e Transação de Tempo-Real

Para atender os requisitos temporais de um SGBD-TR considera-se uma estrutura para os atributos, representado como uma quádrupla (*ordem; avi; tsr; imprec*), onde: *ordem*: número de seqüência do atributo; *avi*: é o intervalo de validade absoluta, ou seja, é o intervalo de tempo no qual o dado é considerado válido temporalmente; *tsr*: é o rótulo de tempo, isto é, o tempo no qual o dado foi gravado no sistema; *imprec*: é a quantidade de imprecisão acumulada para o atributo. A definição dessa estrutura visa armazenar todas as informações pertinentes a um item de dado do ponto de vista temporal. Também, considera-se uma estrutura para cada transação (τi) definida por uma quádrupla (t_{li}; t_{ci}; p_{ri}; p_{ei}), onde: t_{li} é o tempo de liberação de τi , isto é, o momento no qual todos os recursos necessários à execução de τi estão disponíveis, a partir desse momento τ*i* estará pronta para ser executada; t_{ci} representa o tempo computacional de τ*i*, isto é, o tempo de processamento necessário para a executá-la; p_{ri} especifica o prazo máximo para a execução de τi; p_{ei} indica a periodicidade de τi, isto é, a freqüência que a transação será executada.

Basicamente os SGBD-TR apresentam três características principais:

1. A exigência de tratar dados lógicos e temporalmente consistentes;
2. A exigência de satisfazer as restrições temporais das transações;
3. A exigência de permitir a negociação entre restrições lógicas e temporais.

Estas características adicionam novas exigências às linguagens de consultas para SGBD, assim como ao processamento das transações.

Além do suporte convencional de um SGBD, um SGBD-TR deve oferecer suporte para o tratamento dos dados e das transações com restrições temporais. Portanto, um SGBD-TR deve fornecer suporte para os usuários especificarem as restrições temporais dos dados, isto é, um intervalo de tempo durante o qual um dado é considerado válido, além de definir os prazos das transações, ou seja, o tempo no qual uma transação deve ser executada. Esses requisitos são bastante complexos, no entanto várias pesquisas estão sendo realizadas no sentido de atendê-los (LEITE, 2005; NETO, 2004; FERNANDES, 2005).

2.6 Conclusões

O Sistema de Gerenciamento de Banco de Dados em Tempo Real buscam eficientemente o controle de tarefas e dados que possuam restrições temporais. Para que as aplicações como, os sistemas de internet, sistemas bancários, de aviação, redes de sensores entre outras aplicações, possam garantir um melhor resultado na execução das tarefas que dependam de restrição de tempo.

Capítulo 3

Processo de Descoberta de Conhecimento

3.1 Introdução

Para a grande quantidade de dados armazenados, surge a necessidade de extrair informações úteis e relevantes, consideradas decisivas em um processo questionável, onde diagnósticos devem ser concisos. Para a demanda excessiva de dados surgem as oportunidades de aprender fatos surpreendentes a partir de banco de dados existentes (GARCIA, 2001).

A integração da quantidade de dados armazenados, seja por uma instituição governamental ou não, necessita de técnicas e ferramentas adequadas para gerar informações relevantes na tomada de decisão de qualquer área a ser pesquisada. Esses dados, contudo, nem sempre são transformados em informação e nem sempre produzem conhecimento.

Oliveira e Garcia (2006) trazem a diferença entre os conceitos de dados e de informação. O dado pode ser considerado o constituinte básico dos sistemas de informação, sendo uma descrição limitada do real, desvinculada de um referencial explicativo e difícil de ser utilizada como informação por não ser interpretado, ou seja, um dado tem como característica o fato de ser incompleto. Já a informação consiste na representação simbólica de fatos ou idéias potencialmente capazes de alterar o nível de conhecimento de alguém a respeito de alguma coisa, diminuindo seu grau de incerteza.

Para que aconteça esse processo de transformação, é necessária a utilização de algumas técnicas específicas.

A Descoberta de Conhecimento em Bancos de Dados, também conhecida como KDD (*Knowledge Discovery in Databases*) é o processo não trivial de identificar em dados padrões que sejam válidos, novos (previamente desconhecidos), potencialmente úteis e compreensíveis, visando melhorar o entendimento de um problema ou um procedimento de tomada de decisão (FAYYAD, 1996).

Para agarantir a integridade do processo KDD, é importante que as etapas, como mostra a Figura1, se adéqüem a base de dados. Assim, Malucelli (2010), enfatiza a sequência de cada etapa e suas devidas importância.

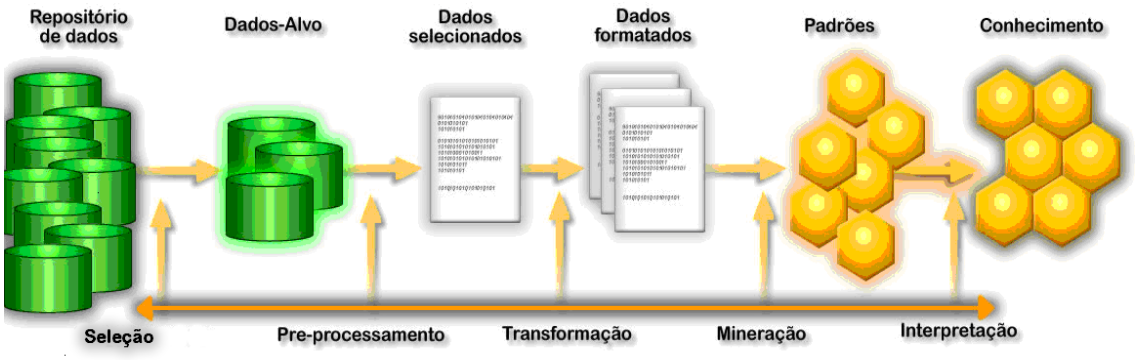

Figura 1: Etapas do Processo KDD. Fonte: Adaptado de Fayyad, 1996.

> O KDD é composto das seguintes etapas: pré-processamento, mineração de dados e pós-processamento. A etapa de pré-processamento é considerada de grande importância e tem como objetivo adequar as bases para extrair padrões. Após a etapa de pré-processamento, inicia-se a fase de Mineração de Dados (Data Mining), a qual é considerada como a etapa central na descoberta do conhecimento e envolve a escolha e aplicação da ferramenta e algoritmo a ser utilizado. Dentre os possíveis algoritmos a serem utilizados nesta etapa estão os indutores de regras, os algoritmos genéticos, entre outros. Por fim, ocorre o pós-processamento, no qual os resultados obtidos são analisados e interpretados. Nesta fase os padrões descobertos são avaliados no sentido de verificar se satisfazem o critério necessário para constituir um elemento importante para o apoio à decisão (MALUCELLI, 2010).

O KDD é um processo composto pela seleção de dados, pré-processamento, transformação dos dados e estabelecimento de padrões úteis na extração de conhecimento, ou seja, tradução de dados brutos em informações relevantes (VIANA et al., 2010).

> O KDD é responsável pela seleção dos métodos a serem utilizados para localizar padrões nos dados, seguida da efetiva busca por padrões de interesse numa forma particular de representação, juntamente com a busca

> pelo melhor ajuste dos parâmetros do algoritmo para a tarefa em questão" (SILVA, 2004).

A afirmação de Silva (2004) refere-se às etapas que compreendem o KDD, onde a importância de cada fase complementa a efetivação do KDD. No entanto, a definição e ajuste de representação cabem à etapa de mineração de dados.

3.2 Mineração de Dados

Para a aplicação da mineração de dados ao banco de dados, é necessária toda a adaptação ao processo do KDD, onde as etapas são necessárias para a obtenção do objetivo que são os resultados obtidos através da mineração de dados.

> A fase de Mineração de Dados consiste em "aplicar algoritmos para extração de padrões nos dados sem a realização dos passos adicionais do processo de descoberta de conhecimento, tais como a incorporação de conhecimento anterior e a interpretação de resultados" (RAMOS e SANTOS, 2003).

Informações podem ser obtidas a partir de dados através de técnicas de interpretação, anotação, classificação, agrupamento ou sumarização destes dados ou de técnicas que permitam a associação e correlação de outras informações (possivelmente de outras fontes). Várias destas técnicas fazem parte do conjunto de técnicas, ferramentas, procedimentos e algoritmos conhecidos comumente como Mineração de Dados.

Atualmente, as técnicas de Mineração de Dados consistem, sobretudo, na análise dos dados após a extração, buscando-se levantar as necessidades reais e hipotéticas de cada usuário. A tecnologia disponível gira em torno de algoritmos avançados, computadores multiprocessados e bancos de dados grandes e poderosos.

O seu processo de desenvolvimento atual é composto de métodos e algoritmos que possibilitam a extração de novos conhecimentos. Entre os seus vários algoritmos, destacam-se algumas que são as mais utilizadas: associação, classificação, regressão, clusterização e sumarização (GALVÃO, 2012).

3.2.1 Tipos de Algoritmos

Os algoritmos de Mineração de Dados, independente de sua forma de extração de conhecimento, segue a aplicação de sequências estruturais de análise de dados. Conforme a aplicação dos diversos tipos de algoritmos, o mais importante é adequação dos dados para melhor inferência das regras.

Galvão (2012) conceitua a seguir os tipos de algoritmos:

- Classificação: consiste na predição de uma variável categórica, ou seja, descobrir uma função que mapeie um conjunto de registros em um conjunto de variáveis predefinidas, denominadas classes;
- Regressão: buscam-se funções lineares ou não, sendo que a variável a ser predita consiste em um atributo numérico (contínuo) presente em banco de dados com valores reais;
- Clusterização: precisa, automaticamente, identificar os grupos de dados, aos quais o pesquisador deverá atribuir as variáveis;
- Associação: tarefa de associação consiste em identificar e descrever associações entre variáveis no mesmo item ou associações entre itens diferentes que ocorram simultaneamente, de forma frequente em banco de dados;
- Sumarização: procura identificar e indicar características comuns entre um conjunto de dados.

Segundo Elsmari (2011), o resultado da mineração de dados é descobrir um tipo de informação nova, que estabelece através da regra de associação os fatos e padrões sequenciais. Os fatos se aproximam através de registros do banco e árvore de classificação. Essa última, a árvore de classificação utilizada na ferramenta OpMM é melhor descrita no item 3.2.2.

3.2.2 Árvores de classificação

O algoritmo de classificação tem o objetivo de encontrar uma hierarquia de classes predeterminadas, ligada ao método de aprendizado supervisionado, com o objetivo de induzir conceitos a partir de atributos que estejam rotulados. Atribuindo essa estratégia de modelo, pode-se classificar novos dados.

> Cada registro nos dados de treinamento contém um atributo, chamado rótulo de classe, que indica a que classe o registro pertence. O modelo produzido normalmente está na forma de uma árvore de decisão ou um conjunto de regras. (ELSMARI, 2011, p. 634)

O algoritmo de árvore de decisão J48 se baseia no algoritmo de árvores de decisão C4.5, que forma a árvore mais adequada sobre o conjunto de dados, podando as regras que melhoram a correlação de resultados. Assim, sua forma gráfica de apresentar os resultados, mostra que esses resultados atuam diretamente no ganho de informação a respeito dos dados.

A principal vantagem das árvores de decisão é o fator de indução a uma escolha no processo de decisão, levando em consideração os atributos que são mais relevantes, além de serem compreensíveis para os usuários (STEINER, 2006).

Witten e Frank descrevem que os atributos são identificados quanto a sua importância, a árvore de decisão facilita para compreensão de qual atributo influencia mais em seus resultados (WITTEN e FRANK, 2005).

A árvore de decisão apresenta um conjunto de dados, onde todas as amostras estão representadas na raiz da árvore. Tais amostras são classificadas com sub-rotinas baseadas nos atributos que forem selecionados. A árvore de decisão representa a descrição de cada classe de atributo através das regras de classificação, com a representação gráfica de uma árvore. A Figura 2, de Elsmari (2011) representa uma árvore de classificação, para aplicação especificamente de cartão de crédito, com o atributo casado, sendo o atributo mais abrangente, e especificamente cada ramificação restringe a uma situação.

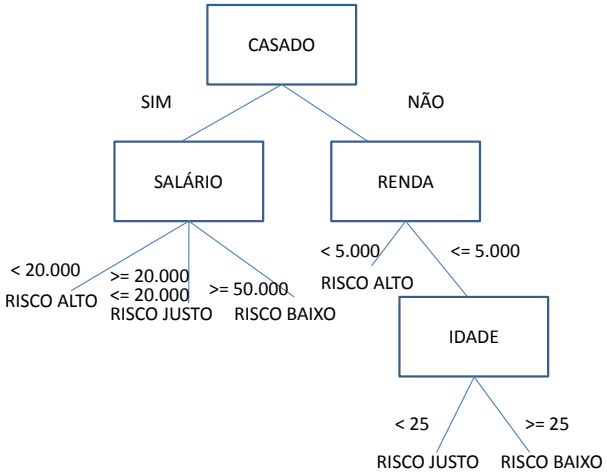

Figura 2: Árvore de decisão (ELSMARI, 2011, p. 710).

3.3 Mineração de Dados Médico

A informatização tem potencial para ser utilizada na área da Saúde, com significativos benefícios para os seus diversos contextos. Uma das inúmeras possibilidades do uso da Informática na área da saúde é a aplicabilidade do uso da Descoberta de Conhecimento em Bancos de Dados na Medicina e na saúde pública, tendo em vista a grande quantidade de dados produzidos (em prontuários, sistemas de informação, dados hospitalares, estatísticas, dentre outros) e a importância das informações que podem ser extraídas a partir daí.

Na área médica, distinguem-se duas frentes para a Mineração: Administração/Gestão e Diagnóstico.

Na Administração/Gestão os sistemas lidam com os serviços oferecidos aos pacientes e com a identificação de pacientes de risco. As limitações financeiras impostas ao setor da saúde tornam fundamental que o processo de tomada de decisão neste segmento seja executado com base em conhecimento estratégico obtido a partir de uma base íntegra de dados. Em termos de Diagnóstico, a utilização da Mineração de Dados possibilita o desenvolvimento de sistemas de adaptação aos procedimentos médicos.

A maior parte das tarefas clínicas requer a captação de numerosos e múltiplos tipos de dados de pacientes, frequentemente através de meios eletrônicos, sendo que para fazer diagnósticos ou tomar decisões terapêuticas é necessário uma interpretação destes dados. A maioria dos dados armazenados inclui uma validade temporal em que estes são considerados válidos.

As tendências temporais e os padrões dos dados clínicos adicionam introspecções significativas à análise estática. Assim, é desejável criar automaticamente abstrações (curtas, informativas e interpretativas) de dados clínicos em tempo-real e poder responder a perguntas sobre tais abstrações. Ao fornecer esta capacidade de abstrair dados com restrição de tempo, consegue-se ajudar o médico, criando uma ferramenta de apoio à decisão automatizada (BOAZ e SHAHAR, 2005).

Pode-se aplicar as técnicas KDD em diversas bases de dados, no entanto o foco dessa dissertação se concentra em base de dados médica, envolvendo todo o processo de informatização hospitalar de pacientes internados que estão sendo monitorados de alguma forma, seja monitoramento manual, automático, ou através de exames médicos.

3.3.1 Base de Dados Médica

Neste trabalho utiliza-se a base de dados médica *Multiparameter Intelligent Monitoring in Intensive Clinical* (MIMIC II), que é um projeto criado em outubro de 2003, a partir do *Bioengineering Research Partnership* (BRP) com a parceria do *National Institute of Biomedical Imaging and Bioengineering* (NIBIB), uma base de dados pública, disponível na internet através do domínio da Physionet, que só libera o acesso após curso de proteção humana, através do *National Institutes of Health* (NIH). A base possui coleta dos dados de pacientes internados em Unidade de Terapia Intensiva (UTI) no *Beth Boston's Israel Deaconess Medical Center* (BIDMC).

A classificação dos pacientes é dada por faixa etária os pacientes de 0 a 28 dias são considerados neonatais, 29 dias a 18 anos pediátricos e acima de 18 anos são considerados adultos. Assim, classificados também como clínica médica cardiológica, coronariana, neurológica, respiratória, traumática, queimados, dentre outras. Destaca-se ainda o Centro de

Tratamento Intensivo (CTI), com o agrupamento, numa mesma área física, de duas ou mais UTI, incluindo-se, quando existentes, as unidades de tratamento semi-intensivo.

O banco de dados MIMIC II contém informações do leito de UTI do paciente, como também informações arquivadas de interesse do hospital. Os dados foram coletados entre o período de 2001 e 2008 de pacientes em leitos de UTI, seja clínico, cirúrgico, coronariano e neonatal. O MIMIC II é dividido em dois tipos de informações, clínicas e em dados gráficos, onde os dados clínicos são toda e qualquer informação fisiológica do paciente e os dados em sinais de alta resolução, de medições fisiológicas minuto a minuto, facilitando comparações de dados clínicos com dados gráficos.

Para identificação dos dados alguns equipamentos de monitoramento, que se pode aplicar ao paciente, onde os dados capturados são armazenados em banco, como: termômetro, oxímetro de pulso, eletrocardiográfico com freqüência cardíaca e medida intermitente de pressão arterial, monitor de pressão arterial, capnógrafo, monitor cardíaco, máscara e cateter de oxigênio, cateter central, tubo ortotraqueal e ventilador mecânico. Ainda como formas de monitoramento existem as observações médicas que são informadas manualmente, os registros de exames laboratoriais, os campos com exames de imagem dentre outros diversos registros.

3.4 Conclusões

Nesta perspectiva, está sendo definida uma arquitetura de um sistema de monitoramento através da mineração de dados médico em tempo-real, que perfaz todo objeto de estudo desta dissertação, apresentada no próximo capítulo.

Capítulo 4

Implementação e Estudo de Caso

4.1 Introdução

A aplicação foi concebida com intuito de auxiliar o desenvolvimento de aplicações com fluxo de informação contínua e que necessitam de um acompanhamento e extração de conhecimento dos dados. Com a finalidade de uma aplicação de iniciativa no auxílio da tomada de decisão médica, em dados médicos genéricos, ou seja, independente do problema médico o comportamento do OpMM será o mesmo.

Para o desenvolvimento da aplicação de integração foi utilizada uma base de dados médica MIMICII, onde a aplicação oferece a funcionalidade de consultas como um SGBD, através da conexão com o banco de dados, onde estão armazenadas as informações de interesse, os registros de cadastro de pacientes, são extraídos do servidor de dados, através de um banco de dados, conforme Figura 3.

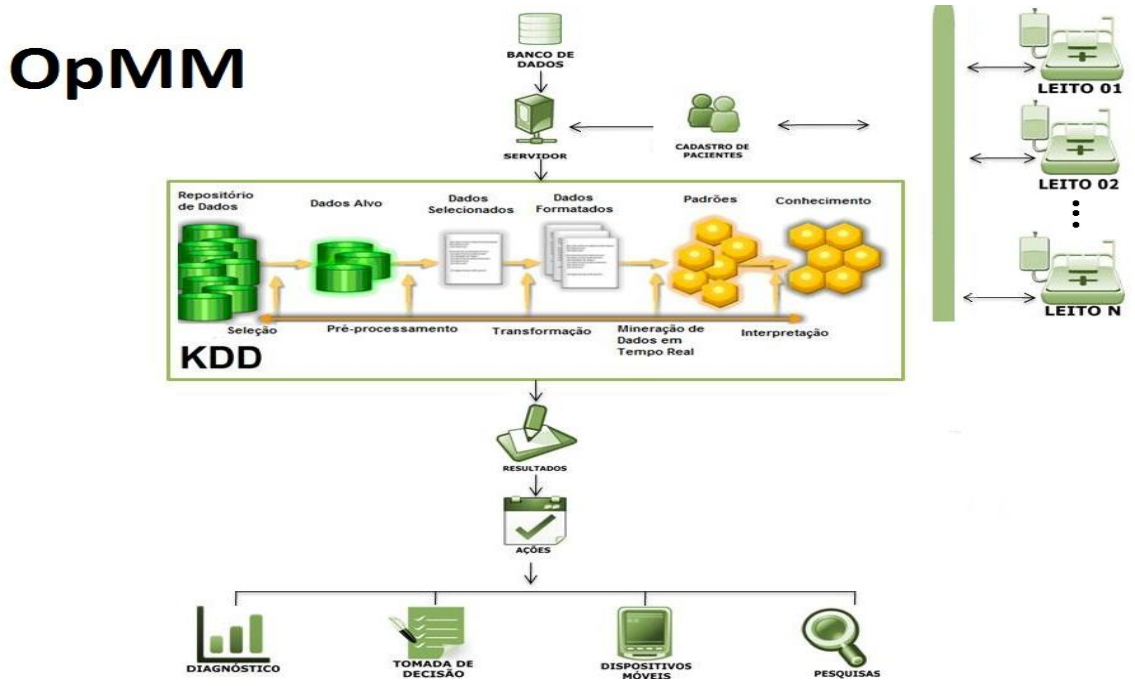

Figura 3: Mineração de Dados Médico em Tempo-Real.

O processo do KDD está totalmente integrado, as coletas contínuas dos dados, como foram apresentadas na Figura 3. O Banco de Dados com o armazenamento dos dados do paciente de cada leito direcionado para o servidor. Assim, os padrões são identificados gerando conhecimento válido no processo do KDD, da base médica em análise, podendo ser disseminados através de diagnóstico, tomada de decisão, alerta de mensagens dos dispositivos móveis e computadores e pesquisas.

A aplicabilidade do OpMM, no ambiente médico, enfatiza a possibilidade de compreensão de adequação a qualquer base de dados da área, tornando a extração de conhecimento em um grande volume de dados, descobrindo novas correlações, padrões e tendências entre as informações de um paciente.

O fluxo de dados na arquitetura é adaptado à base na qual esteja instalado, independente de qual tipo de dados médico seja analisado. Os dados gerados periodicamente são pré-processados, com a aplicação da técnica de mineração e a obtenção de resultados são permitidas munidas das aplicações das ações de interesse, conforme ilustrado na Figura 4.

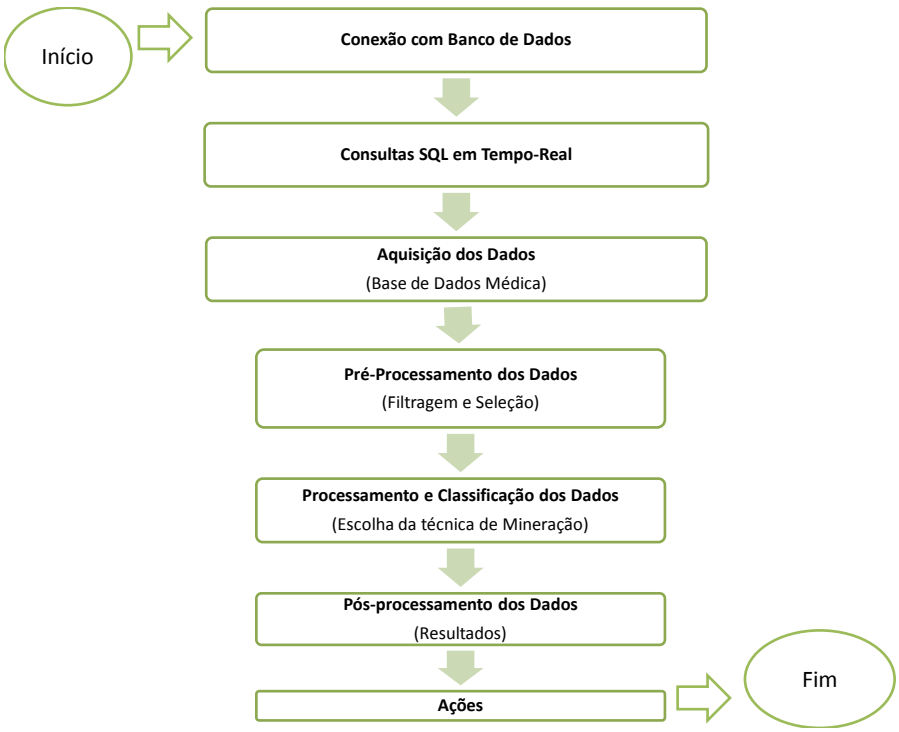

Figura 4: Fluxograma do OpMM.

Todo o processo do fluxograma ocorre desde a conexão com o banco de dados a ser ajustado a arquitetura do OpMM, com a consulta SQL em tempo real já é possível a restrição aos dados e ao intervalo de tempo, para as informações de interesse que é a aquisição dos dados. Na etapa de pré-processamento dos dados identifica-se os registros inconsistentes ou inválidos. No processamento e classificação dos dados é feito o processo de mineração de dados e na etapa do pós-processamento dos dados já acontece os resultados da mineração de dados. Finalizando o fluxograma com as ações que o usuário irá seguir com os resultados da mineração.

Assim, o fluxograma define o processo de mineração de dados juntamente com a linguagem de consulta em tempo-real e o processamento do algoritmo de mineração, buscando a disseminação de resultados para a necessidade de um usuário.

4.2 Descrição das Primitivas

Para diferenciar os comandos SQL-99 dos comandos com restrições temporais específicos da Linguagem de Consulta em Banco de Tempo-Real (LC-BDTR), sem a existência de um compilador, foi criada uma classe que se comporta como um interpretador, que recebe um comando e o interpreta, visando detectar se o mesmo é um comando da SQL-99 ou da LC-BDTR. Caso seja da LC-BDTR, a mesma extrai as informações temporais necessárias para execução do comando.

Para cada tipo de comando é definido um método para a execução do mesmo. A seguir, será detalhado o processo de implementação para cada primitiva temporal. Maiores detalhes da implementação da interface se encontram no Anexo I. As primitivas a seguir foram implementadas através da Classe SQL, conforme Leite (2005).

4.2.1 *Initialize*

A primitiva *Initialize* é baseada em duas classes: *Java Timer* e *TimerTask*. O Timer é a classe que agenda a execução do *TimerTask*. O agendamento é baseado na data extraída a partir do *timestamp* da sintaxe desta primitiva obtido na linha 1 do código a seguir.

```
1 Initialize initialize = new Initialize(commands.get(3).toString(), query,
type, id);

2 Timer timer = new Timer(true);

3 timer.schedule(initialize,date);
```

Na 1º linha, foi declarado um objeto *Initialize*. Na 2º linha, foi declarado um objeto Timer. E na 3º linha, foi agendada a execução do objeto *Initialize*. A relevância deste trecho está no fato de apresentar a característica fundamental da primitiva, onde o tempo de início para começar a execução de uma transação é assegurada.

4.2.2 Terminate

A execução de comandos com esta primitiva requer a validação da data que o comando começou a executar, obtida através do *timestamp* da mesma. O trecho de código essencial para o funcionamento dessa primitiva é:

```
1 Calendar end = GregorianCalendar.getInstance();

2 if(dateTerminate.before(end.getTime())){
```

Na 1ª linha, é marcado o tempo de término da transação. Na 2*a* linha, ocorre a validação temporal do comando. Para isso, o tempo de término do comando é comparado com o *timestamp* passado como parâmetro na primitiva. Se ele for posterior, o comando é avaliado em falso, caso contrário ele é avaliado em verdadeiro.

4.2.3 *Period*

A sintaxe desta primitiva vem acompanhada de outras duas primitivas, o *Initialize* e o *Terminate*. Desta forma, foi necessário criar uma *classe Period* que é um *TimerTask.*

Como o *Period* indica a freqüência na qual o comando deve ser executado, ao agendar esta tarefa no Timer, informa-se a freqüência, que é extraída a partir da sintaxe da primitiva *Period*. O trecho de código essencial para o funcionamento dessa primitiva é:

```
1 initialize = dateFormat.parse(commands.get(4).toString() + " " +
commands.get(5).toString());

2 Timer timer = new Timer(true);

3 Period period = new
Period(commands.get(6).toString(),query,type,id,timer);

4 Integer hour = (Integer)commands.get(1);

5 Integer min = (Integer)commands.get(2);

6 Integer sec = (Integer)commands.get(3);

7 long frequency = ((hour.longValue()*3600) + (min.longValue()*60) +
sec.longValue()) * 1000;

8 timer.schedule(period,initialize,frequency);
```

A 1ª linha desta primitiva é semelhante a 1º linha da primitiva *Initialize*, definido na Seção 4.2.1. Na 2ª linha é declarado um objeto *Timer*. Na 3ª linha, é declarado um objeto *period*. Da 4ª a 6ª linha, as variáveis para hora, minuto e segundo são declaradas. Na 7ª linha, a freqüência com que o comando deve ser executado é calculado. E finalmente, na 8ª linha, uma tarefa é agendada através do objeto *Timer*. Esse agendamento requer como parâmetro a data de início (I*nitialize*), o *TimerTask* que deve ser executado (*Period*) e a freqüência da execução (*Frequency*).

4.2.4 *Computation*

Para implementar esta primitiva é necessário registrar o tempo de início e o tempo de término e comparar a diferença desses dois valores com o valor passado pela mesma. O trecho de código essencial para o funcionamento dessa primitiva é:

```
1 begin = GregorianCalendar.getInstance();
2 statement = DatabaseManager.connection.createStatement();
3 statement.getConnection().setAutoCommit(false);
4 ResultSet rs = statement.executeQuery(commands.get(3).toString());
5 end = GregorianCalendar.getInstance();
6 computation = (Calendar) begin.clone();
7 computation.add(Calendar.MINUTE,((Integer)commands.get(1)).intValue());
8 computation.add(Calendar.SECOND,((Integer)commands.get(2)).intValue());
9 if (computation.before(end))
```

Na 1ª linha, é marcado o tempo de início do processamento do comando. Da 2ª a 4ª linha, é gerenciado a conexão com o banco de dados. Na 5ª linha, é marcado o tempo de término do processamento do comando. As linhas 6, 7 e 8 foram utilizadas para viabilizar a implementação desta primitiva. Para isso, foi criando um clone do tempo de início e adicionado a esse tempo os minutos e segundos do *timestamp* da primitiva. Com isso, foi obtida a data limite na qual essa computação poderia terminar para obedecer ao tempo de computação requerido. Desta forma, na 9ª linha, compara-se o tempo de término com o tempo limite. Se o tempo limite (*computation*) for anterior ao tempo de fim (*end*), então o comando não é validado, já que sua computação ultrapassou o tempo requerido. Se o tempo limite for posterior, então o comando é validado.

4.2.5 *Return_tp*

Para implementar esta primitiva é necessário registrar o tempo de início e tempo de fim da execução do comando e subtrair tais tempos. Visando obter o tempo de processamento do mesmo. O trecho de código essencial para o funcionamento dessa primitiva é:

```
1 begin = Calendar.getInstance().getTime();
2 int code = statement.executeUpdate(sql);
3 end = Calendar.getInstance().getTime();
4 result = "Result code is " + code +"\n tp = " + (end.getTime() -
begin.getTime()) + "milisegundos";
```

Na 1ª linha, é marcado o tempo de início do processamento do comando. Na 2ª linha, o comando é executado. Na 3ª linha, é marcado o tempo de término do comando. Finalmente, na 4ª linha, a subtração dos tempos é realizada e o tempo de processamento do comando é obtido.

4.2.6 *Validate*

Para implementar esta primitiva os atributos *avi* e *tsr* são recuperados. Ou seja, essa primitiva é restrita apenas para realizar consultas que retornam tuplas contendo todas as colunas das tabelas, visando garantir a presença das colunas *avi* e *tsr*. Portanto, o funcionamento desta primitiva consiste na varredura das tuplas recuperadas e validação dos dados com base nos valores de *avi* e *tsr*. O trecho de código essencial para o funcionamento dessa primitiva é:

```
1 if(end.getTime().getTime() - tsr.getTime() <= avi_inMillis)
```

Na linha 1, o valor do *tsr* é subtraído do valor do tempo corrente e comparado ao valor do *avi*, visando verificar a validade temporal dos dados.

4.3 Descrição Geral da Interface

A interface descrita a seguir dá suporte à elaboração de consultas utilizando a sintaxe da Linguagem de Consulta em Tempo-Real, permitindo a manipulação de dados armazenados em um SGBD convencional, no entanto, tal manipulação obedece às restrições de tempo. O OpMM disponibiliza duas interfaces: uma interface para realizar conexão com o banco de dados e outra interface para declarar consultas. As mesmas serão definidas com mais detalhes nas Seções 4.4 e 4.5, respectivamente.

Para implementar a interface, foi aplicado o padrão de projeto *Observer*, visando manter o relacionamento entre a interface e as regras de negócio. Desta forma, foi desacoplado o código da interface, do código da lógica de negócio, fazendo com que a comunicação entre eles ocorra através de eventos.

A interface foi desenvolvida através da linguagem Java, que disponibiliza um conjunto de características fundamentais para sua implementação, tais como: portabilidade, segurança, robustez, bibliotecas para acesso a banco de dados e pacotes que disponibilizam primitivas de tempo-real (Eckel, 2002). O ambiente de desenvolvimento utilizado foi o Eclipse. O SGBD utilizado foi o PostgreSQL, com código aberto e totalmente compatível com ACID, entretanto qualquer gerenciador poderia ter sido usado.

4.3.1 Especificação Java

O OpMM foi implementado utilizando a linguagem de programação Java, com a ferramenta de desenvolvimento Eclipse, versão *Indigo*. A linguagem Java disponibiliza um pacote java.util que contém uma Classe denominada *Timer* e uma Classe denominada *TimerTask.* A Classe *Timer* disponibiliza um conjunto de primitivas de tempo e métodos para o agendamento de tarefas. O método *schedule*( ), por exemplo, pode receber até três

parâmetros: uma tarefa, uma data para a tarefa iniciar e a freqüência que a mesma deve ser realizada.

A Classe *TimerTask* disponibiliza recursos para tratar com tarefas que devam ser tratadas em tempos futuros e/ou executadas repetidas vezes. A Classe *TimerTask* possui um método denominado *run*( ), que é invocado pela Classe Timer, através do método *schedule*( ). O método *run*( ) deverá conter o código a ser executado cada vez que a tarefa for escalonada.

A Classe *Instance Query* é responsável por criar a base de dados. A base de dados propriamente dita é identificada através da classe *Instances.* A classe *ResultSet* faz o retorno do banco de dados, repassando para a *Instance Query*. A classe J48 repassa os dados para execução do algoritmo j48. Algumas outras informações sobre o código no Anexo I.

4.3.2 Conexão com o SGBD

O Sistema de Gerenciamento de Bando de Dados, utilizado para a implementação foi o PostgreSQL, onde através da definição do caminho e do driver de conexão, foi integrada a ferramenta através da linguagem Java, como ilustra o exemplo abaixo:

```
conectarBD("jdbc:postgresql://localhost:8080/"+digit,radix,senha1);

Class.forName("org.postgresql.Driver");
```

A base de dados MIMIC II foi toda inserida no SGBD Postgres, onde a manipulação e visualização real dos dados poderiam ser feita, mas como a finalidade era o desenvolvimento de uma ferramenta integrada, a comunicação do OpMM ao SGBD favorece ao usuário final que realiza todo o processo de mineração através da ferramenta.

4.4 Interface de Conexão

Pela conexão definida no ambiente Java. A comunicação é feita, através da interface desenvolvida, o usuário passa como parâmetros o seu nome (*User*), uma senha (*Password*) e o nome de um banco de dados válido (*Database*). Caso os parâmetros sejam validados, a

conexão com o banco de dados é estabelecida e uma mensagem de conexão estabelecida é exibida no campo Output da interface. Caso contrário uma mensagem de erro será exibida no mesmo campo Output da interface, conforme Figura 5.

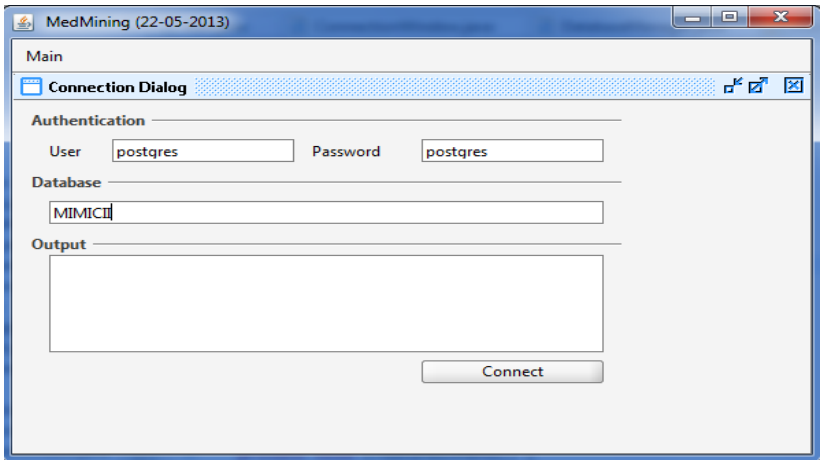

Figura 5: Interface de Conexão.

4.5 Interface de Consulta

A interface de consulta segue a mesma tela de visualização de um SGBD, com a implementação das consultas em tempo-real e a execução, conforme Figura 6.

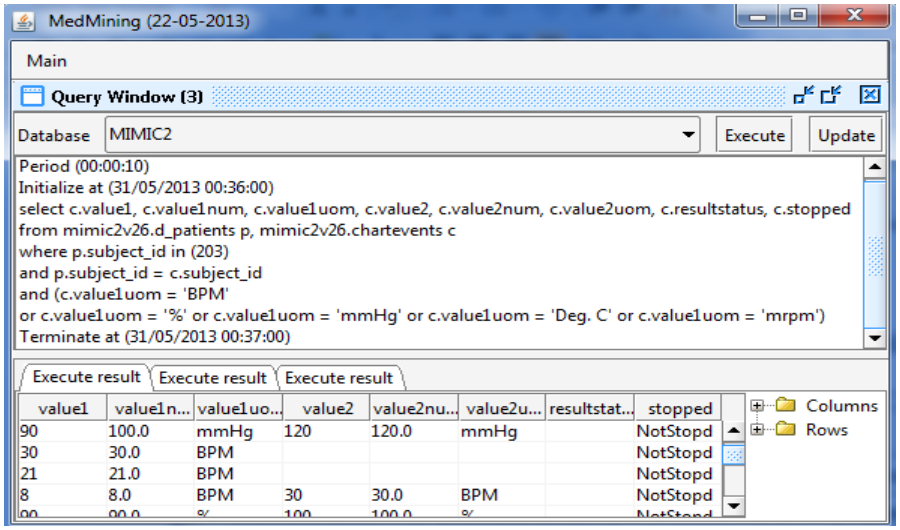

Figura 6: Interface de Consulta.

Os resultados da consulta são apresentados na mesma tela da consulta, ainda sem a aplicação do algoritmo. Porém, logo que o algoritmo finaliza o processo, a tela de aplicação do algoritmo será apresentado conforme Figura 7.

4.6 Descrição da Implementação

A aplicação do algoritmo será apresentado na Figura 7, com uma tela de mineração de dados, dividida em classificação de dados e a árvore de decisão.

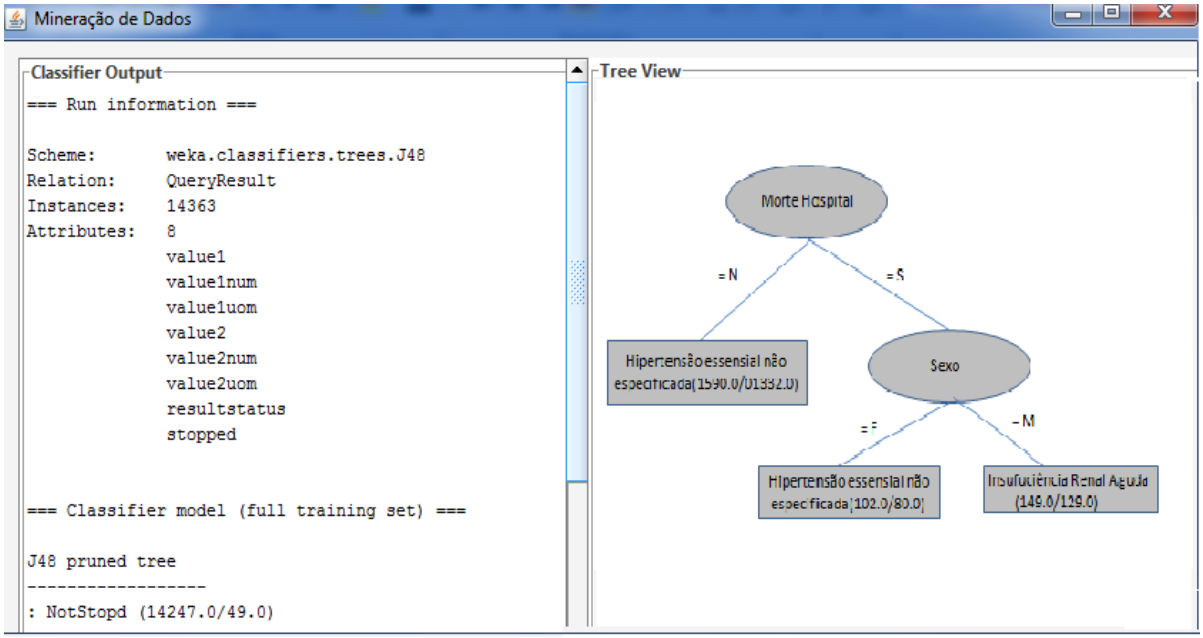

Figura 7: Mineração dos dados.

4.7 Utilizando a aplicação da Ferramenta na UTI

A utilização da ferramenta em um ambiente de UTI foi favorável pela disponibilidade dos dados e por se tratar de um ambiente com informações relevantes, em se tratando de área médica.

Para definição das informações relevantes, realizou-se uma entrevista junto a 02 médicos intensivista, do Centro de Oncologia e Hematologia de Mossoró, discutindo quais seriam as informações relevantes inerentes ao estado do paciente na entrada de um médico plantonista, como na entrega de plantão na UTI.

Os médicos informaram que ao assumir os pacientes da UTI, realiza-se um levantamento de informações, desde o prontuário do paciente até a evolução da enfermagem. No entanto, consideram-se de significativa relevância os sinais vitais para a avaliação do paciente, que reproduz o estado do paciente em um determinado instante. As avaliações dos sinais vitais, junto ao histórico de diagnósticos do paciente, facilita a conduta médica.

4.7.1 Unidade de Terapia Intensiva

A Unidade de Terapia Intensiva (UTI) é o setor que trata de pacientes graves ou de risco que necessitam de assistência médica e de enfermagem permanentes, com pessoas capacitadas, assim utiliza equipamentos específicos próprios e outras tecnologias destinadas a diagnóstico e tratamento.

Devido o grande tempo de permanência de pacientes debilitados em leitos de UTI, conforme ilustrado na Figura 8 ocorre um grande monitoramento tecnológico para os devidos cuidados intensivos. Desde o armazenamento de informações dos sinais vitais, favorecendo uma interação através de dispositivos de comunicação.

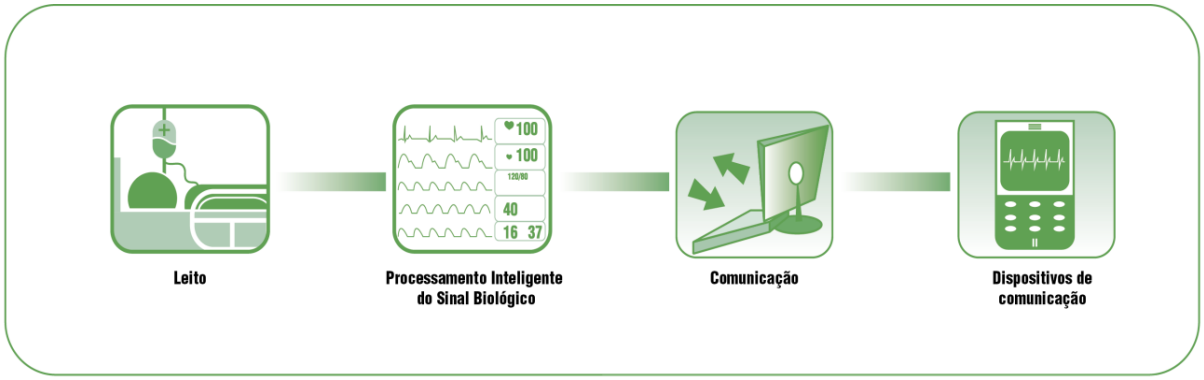

Figura 8: Ambiente de Monitoramento. Fonte: Leite (2011).

A grande quantidade de dados avalia e compara resultados dos pacientes através de monitores cardíacos, monitores de ventilação, exames laboratoriais, exames de imagens, entre outros. Tais dados registram o grau de gravidade de cada indivíduo, fornecendo características da doença de cada paciente internado na UTI.

Segundo a prática toda mudança de plantão o médico recebe informações de cada leito de paciente, informando o diagnóstico, a idade, a terapêutica utilizada como medicação e procedimento realizado no paciente, se ocorreu ou não intercorrências com o paciente em outros momentos.

Os sinais vitais são também informações indispensáveis ao médico, uma vez que é de responsabilidade da enfermagem e equipe técnica a coleta dos dados, caso não exista um armazenamento de tais informações através do monitoramento, a coleta dos dados deve ser realizada a cada hora no leito de cada paciente.

Os principais sinais vitais são identificados por Freqüência Cardíaca (FC), Frequência Respiratória (FR), Pressão Arterial (PA) e temperatura. Porém, outras informações vitais como dor, glicemia capilar e nível de saturação de oxigênio no sangue (SPO2), são relevantes em se tratando de conduta médica. A Tabela 1 abaixo ilustra a classificação de normalidade nos parâmetros de sinais vitais.

Tabela 1: Principais Parâmetros Vitais dos Pacientes

Nome	**Abrev**	**Valor de Referência Normal**
Frequência Cardíaca	Fc	Recém Nascido – 100 a 160 bpm Criança – 80 a 120 bpm Adulto – 60 a 100 bpm
Frequência Respiratória	Fr	Recém Nascido – 30 a 60 mrpm Criança até 6 anos – 20 a 30 mrpm Adulto – 12 a 20 mrpm
Temperatura	T	Axilar:36^0C a 37^0C Oral: $36,2^0$C a $37,2^0$C Retal: $36,4^0$C a $37,4^0$C
Pressão Arterial	Pa	04 anos – 85/60 mmHg; 06 anos – 95/62 mmHg;

		10 anos – 100/65 mmHg; 12 anos – 108/67 mmHg; 16 anos – 118/75 mmHg; Adultos – 120/80 mmHg; Idosos – 140 a 160/90 a 100 mmHg.

Assim, ressalta-se ainda a importância da equipe médica receber informações em tempo hábil sobre o quadro clínico dos pacientes internados sempre que for detectada uma possível situação de risco. Podendo ainda requisitar outros sinais indiretos, de modo a complementar os quatros sinais vitais clássicos, tais como: coloração da mucosa labial, ocular, diâmetro e reflexo pupilar, entre outros.

De acordo com a Portaria nº 466/Secretaria de Vigilância da Saúde (SVS), de 04 de junho de 1998, toda UTI deve dispor de no mínimo um responsável técnico, com título de especialidade em medicina intensiva, específico para a modalidade de UTI. Assim, sob sua responsabilidade, um enfermeiro chefe, exclusivo da unidade, responsável pela área de enfermagem, obedecendo também a regra que para cada 10 leitos um médico diarista para cada 10 leitos ou fração, especialista em medicina intensiva, responsável pelo acompanhamento diário da evolução clínica dos pacientes internados na UTI, ou na semi-intensiva, quando existente entre outros como um fisioterapeuta, burocrata e um auxiliar de enfermagem para cada 02 leitos de UTI adulto (Brasil, 2009).

No entanto, as informações de interesse para o monitoramento dos pacientes, podem ser armazenadas através da automação. O monitoramento é contínuo e direto, ou seja, para cada paciente, um monitor de cabeceira e/ou transporte é disponibilizado.

Para o médico as hipóteses que favorecem a admissão e conhecimento do estado de cada paciente, seria o índice apontado pelos sinais vitais. Assim, para a consulta (1), procurou-se na base de dados os registros relacionados com Batimentos cardíacos Por Minuto (BPM), nível de oxigenação sanguínea, Pressão Arterial (PA), temperatura e Movimentos Respiratórios Por Minuto (MRPM).

Para demonstração da execução das consultas em tempo-real, foi selecionado o paciente com *subject_id* ‘203’ e os sinais vitais do mesmo, em um intervalo de dados que inicialize na data e hora relacionanda `(25/05/2013 14:00:00)` e termine no `(25/05/2013 18:00:00)`, alguns exemplos de consulta serão apresentados, logo a seguir:

Consulta (1):

```
INITIALIZE AT (25/05/2013 14:00:00)

PERIOD (00:00:10)

select c.value1, c.value1num, c.value1uom, c.value2, c.value2num,
c.value2uom, c.resultstatus, c.stopped

from mimic2v26.d_patients p, mimic2v26.chartevents c

where p.subject_id in (203)

and p.subject_id = c.subject_id and (c.value1uom = 'BPM'or c.value1uom =
'%' or c.value1uom = 'mmHg' or c.value1uom = 'Deg. C' or c.value1uom =
'mrpm')

TERMINATE AT (25/05/2013 18:00:00)
```

A Consulta (1) recupera do paciente ‘203’ (p.subject_id in (203)) os dados de monitoramento (c.value1, c.value1num, c.value1uom, c.value2, c.value2num,), onde armazena os valores referentes aos sinais vitais, dentro do período de 04 horas (INITIALIZE AT (25/05/2013 14:00:00)),(TERMINATE AT (25/05/2013 18:00:00) e a cada 10 (PERIOD (00:00:10)) segundos. Com a Consulta (1), pode-se observar o comportamento do paciente e suas intercorrências, dentro do intervalo de tempo citado.

Aplicando o algoritmo supervisionado de classificação por árvore de decisão J48, obteve-se padrões de resultados para o paciente ‘203’, como Pressão Arterial normal, com febre, apresentou Frequência Respiratória baixa, que serão melhor apresentados na Tabela 2, com os resultados da mineração.

Na Consulta (2) faz observações como sexo, morte ou não no hospital, idade, peso, nos registros da base, sem a inserção de restrição temporal, apenas por finalidade de conhecimento dos dados médico. Preparando os dados para execução, identificando apenas alguns atributos, dentre vários existentes, como sexo, Classificação Internacional de Doenças (CID), morte no hospital, estado civil, etnia, convênio, religião, tipo de admissão, fonte de admissão, idade, peso e altura.

Consulta (2):

```
INITIALIZE AT (26/05/2013 14:00:00)
PERIOD (00:00:10)
```

```
select o.subject_id, o.medication, o.status, o.frequency, i.description,
d.admission_type_descr, t.icustay_age_group as idade, p.sex as sexo,
p.hospital_expire_flg as morte_hospital from mimic2v26.poe_order o,
mimic2v26.icd9 i, mimic2v26.demographic_detail d, mimic2v26.d_patients p,
mimic2v26.icustay_detail t

where o.subject_id = p.subject_id and o.subject_id = i.subject_id
and o.subject_id = d.subject_id and d.subject_id = t.subject_id and
o.subject_id = t.subject_id and p.subject_id = t.subject_id and
i.subject_id = t.subject_id and d.subject_id = p.subject_id and
i.subject_id = d.subject_id and i.subject_id = p.subject_id and
p.hospital_expire_flg = 'Y' and i.code in
('V29.0','V05.3','584.9','599.0','285.9','401.9','414.01','250.00''428.0','
427.31','486.0','272.0','518.81')

TERMINATE AT (26/05/2013 18:00:00)
```

A Consulta (2) recupera os pacientes (`o.subject_id`), por medicação (`o.medication`), tipo de admissão (`d.admission_type_descr`), idade (`t.icustay_age_group`), sexo (`p.sex`) e se foi a óbito (`p.hospital_expire_flg`) com restrições algumas doenças identificadas por CID (`'V29.0','V05.3','584.9','599.0','285.9','401.9','414.01','250.00''428.0','427.31','486.0','272.0','518.81'`). Essa consulta foi realizada baseada na hipótese de relacionamento entre as doenças do paciente, sexo e idade.

As aplicações do filtro de classificação com atributos de sexo e CID identificaram visualmente, o cruzamento das informações, que geraram os resultados de doenças com maiores incidência nos 10.817 pacientes que foram admitidos pelo hospital, sendo que, 292 apresentaram registros de hipertensão, 216 de aterosclerose coronária e 146 de fibrilação atrial, onde pode ser observado que 800 do sexo feminino e 1041 do sexo masculino, conforme apresentado na Figura 9.

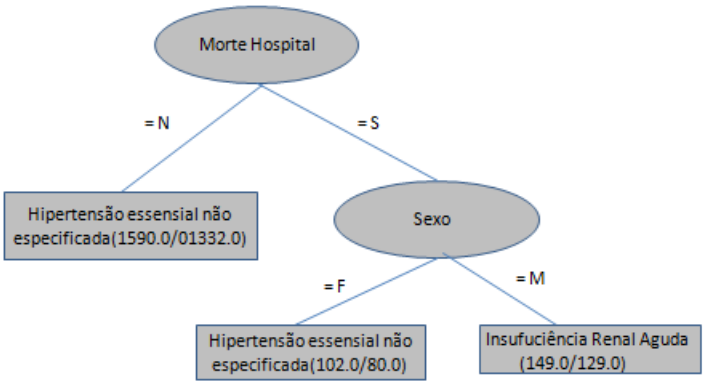

Figura 9: Árvore de Decisão.

A partir de observações realizadas na Consulta (2) após análise de pós-processamento, outra consulta SQL, Consulta (3), foi desenvolvida com base no relato de doenças específicas, analisou-se informações com base em faixa de idade observando as doenças como: infarto, arritmias cardíacas, hipertensão, diabetes, linfoma, aids, dentre outros.

Consulta (3):

```
INITIALIZE AT (27/05/2013 14:00:00)

PERIOD (00:00:10)

select p.subject_id as prontuario, p.sex as sex, p.hospital_expire_flg as
death_hospital, t.icustay_age_group as age_group, t.height as height,
t.weight_first as weight, dischstatus as status, congestive_heart_failure,
cardiac_arrhythmias, valvular disease, pulmonary_circulation,
peripheral_vascular, hipertension, paralysis, other_neurological,
chronic_pulmonary, diabetes_uncomplicated, diabetes_complicated,
hypothyroidism, renal_failure, liver_disease, peptic_ulcer, aids, lymphoma,
metastatic_cancer, solid_tumor, rheumatoid_arthritis

from  mimic2v26.icustay_detail t, mimic2v26.d_patients p,
mimic2v26.demographic_detail d, mimic2v26.censusevents v,
mimic2v26.comorbidity_scores s

where d.subject_id = p.subject_id and t.subject_id = d.subject_id

and p.subject_id = t.subject_id and v.subject_id = t.subject_id and
p.subject_id = v.subject_id and d.subject_id = v.subject_id and
t.subject_id = s.subject_id and p.subject_id = s.subject_id and
v.subject_id = s.subject_id and d.subject_id = s.subject_id and
p.subject_id = d.subject_id

TERMINATE AT (27/05/2013 14:00:00)
```

A Consulta (3) recupera as mesma informações da Consulta (2), aborda-se precisamente a altura (`t.height`) peso (`t.weight_first`) junto as doenças complexas em atendimento hospitalar, como por exemplo arritmia cardíaca (`cardiac_arrhythmias`). Na expectativa de relacionar peso e altura, tipos de admissão e quantificar os tipos de atendimentos relacionadas com determinadas doenças apontadas na Consulta (3).

4.8 Resultados

Abordando os atributos, conforme Consulta (2) de sexo, morte no hospital e CID. Onde foi possível gerar regras do tipo:

- Pacientes que morreram no hospital, do sexo feminino, tiveram como maior índice o diagnóstico de hipertensão.

- Pacientes que morreram no hospital, do sexo masculino, tiveram como maior índice o diagnóstico de doença renal aguda.

Uma avaliação feita em pacientes com linfoma foi possível identificar a predominância de outras doenças que os acometem, como vários tipos de diabetes, casos de hipertensão, infarto agudo do miocárdio e gastroparesia, que é uma diminuição da força de contração da musculatura do estômago.

Com a aplicação da técnica de árvore de decisão, vários resultados foram gerados, de acordo com os atributos de sexo, fonte de admissão e morte no hospital, como apresentado na Tabela 2:

Tabela 2: Resultados da Mineração

ATRIBUTOS	RESULTADOS
Sinais Vitais	• Paciente ‘203’ com PA normal, com febre, apresentou FR baixa. • Paciente ‘203’ com proximidade de temperatura elevada, apresentou FR baixa. • FC baixa, a temperatura e a FR baixa. • PA Normal, FC normal, apresentaram febre e FR baixa.
Sexo	• Prevalecem 6.398 são pacientes do sexo masculino.
Fonte de Admissão	• A maior admissão é na sala de emergência com 5.853, transferência de pacientes com 1.848, somente como urgência e emergência, possuindo maior incidência em pacientes do sexo masculino e casados. • Os 1.106 recém nascidos foram admitidos pelo hospital como encaminhamento normal e prematuro. • Para o tipo de admissão eletiva tiveram somente fonte de admissão normal sem referência específica.
Morte no Hospital	• Dos 10.817 pacientes admitidos 251 foram a óbito com maioria do sexo masculino, casado e admitidos em urgência.

4.9 Conclusões

Os resultados do OpMM apresentados foram executados através da interface implementada neste trabalho, permitindo a inserção de consultas com interesse, com a visualização de novos resultados através do algoritmo em questão.

Neste capítulo, apresentou-se o desenvolvimento de uma interface através o uso de um banco de dados médico, com a ferramenta de mineração de dados em tempo-real, com o intuito de favorecer o apoio na tomada de decisão.

Capítulo 5

Conclusão

Neste trabalho foi desenvolvida uma ferramenta denominada OpMM com o objetivo de transformar e processar dados em informações úteis e estratégicas até então escondidas nas bases de dados.

Este processamento foi realizado através do conhecimento de especialistas médicos que ajudam a gerar hipóteses com o uso de algoritmos de mineração de dados.

Logo, com as informações extraídas é possível auxiliar e apoiar a equipe médica na transformação de ações em diagnósticos, com necessidade de avaliações precisas e automáticas.

As contribuições do OpMM é favorável para o desenvolvimento de uma ferramenta que poderá ser utilizada em qualquer ambiente que necessite de exploração de dados, facilitando para o usuário a conexão com qualquer SGBD, de maneira que não fique limitada a nenhum SGBD.

Assim as tecnologias utilizadas em todo o processo proposto, como o desenvolvimento de uma ferramenta com aplicação específica para a descoberta de conhecimento na área médica buscando atender às exigências de monitoramento da automação hospitalar.

Durante o desenvolvimento deste trabalho, foram identificados alguns aspectos que podem ser explorados ou estendidos a partir do estudo apresentado nesta dissertação. A seguir, destacam-se alguns tópicos de pesquisa que precisam ser realizados:

- A importância da análise exploratória dos dados e validação de outros algoritmos, com a aplicação destas a um problema médico real;
- Desenvolvimento de um sistema *on-line* que permita à equipe médica e aos familiares terem acesso ao quadro clínico do paciente, mediante normas estabelecidas;
- Validação da ferramenta em um cenário e ambiente real, confrontando em tempo-real a geração dos alarmes e a recepção das mensagens;
- Utilização da ferramenta para as aplicações de dispositivos móveis de comunicação;

- Integração de vários módulos inteligentes que possam dar um diagnóstico mais complexo, utilizando histórico, antecedentes familiares e novos parâmetros específicos para cada módulo;
- Aplicação em outro cenário de monitoramento e envio de alertas.

Ressalta-se que a ferramenta OpMM é de grande importância para a base de dados que necessitem da extração de informações com restrição temporal, o que amplia sua contribuição. Assim, com os resultados da mineração de dados com restrição temporal, no momento que se descobre novas informações o apoio a decisão será imediato.

Diante das contribuições expostas, está dissertação possibilitará importantes impactos de natureza científica e tecnológica na automação hospitalar.

Referências

Abe, H.; Yokoi, H.; Ohsaki, M.; Yamaguchi, T., "Developing an Integrated Time-Series Data Mining Environment for Medical Data Mining," *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on* , vol., no., pp.127,132, 28-31 Oct. 2007

Baura, G.D. System theory in industrial patient monitoring: an overview, Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE, vol.2, pp.5356-5359, 1-5 Sept. 2004.

Brasil, Secretaria de Vigilância Sanitária. Portaria nº 466/SVS, de 4 de junho de 1998. Estabelece o regulamento técnico para o funcionamento dos serviços de tratamento intensivo. Brasília: Ministério da Saúde; 1998. Disponível em: http:// sna.saude.gov.br/legisla/legisla/uti/GM_P466_98uti.doc. Acessado em 31 de março de 2012.

Boaz, D., Shahar, Y. A framework for distributed mediation of temporal-abstraction queries to clinical databases. Journal - Artificial Intelligence in Medicine. ISSN 0933-3657, 2005.

Carvalho, J. S. Verificação Automatizada de Sistemas de Tempo Real Críticos. Dissertação de Mestrado. Universidade da Beira Interior, 2009.

Eckel, B. Thinking in Java. 3rd. ed. 2002. Available in web, http://www.bruceeckel.com, Acesso em 02 maio, 2013.

Elmasri, R.; Navathe, S. B. Sistemas de Banco de Dados. 6º. ed. São Paulo: Pearson Addison Wesley, 2011.

Farines, J. M., Fraga, J da S., Oliveira, R. S. Sistemas de Tempo Real. Universidade Federal de Santa Catarina, Jul, 2000.

Fayyad, U. M., Piatesky-Shapiro,G.; Smyth, P. e Uthurusamy, R., From Data Mining to Knowledge Discovery in Database. Cambridge, American Association for Artificial Intelligence/MIT Press, 1996.

Fernandes, Y. Y. M. P. Técnica de Controle de Concorrência Semântico para Sistemas de Gerenciamento de Banco de Dados em Tempo-Real. Dissertação de Mestrado, UFCG. Campina Grande, 2005.

Garcia-Molina, Hector; Ullman, Jeffrey D.; Widom, Jennifer. Implementação de sistemas de bancos de dados. Rio de Janeiro: Campus, 685p. il, 2001.

GALVÃO, Noemi Dreyer; MARIN, Heimar de Fátima. Técnica de Mineração de Dados: uma revisão da literatura. Actapaul. enferm., São Paulo, v. 22, n. 5, Oct. 2009. http://www.scielo.br/scielo.php?script=sci_arttext&pid=S010321002009000500014&lng=en&nrm=iso>. Acesso em 12 Jan. 2012.

J. Friedlin, M. Mahoui, J. Jones, P. Jamieson, “Knowledge Discovery and Data Mining of Free Text Radiology Reports,”, First IEEE International Conference on Healthcare Informatics, Imaging and Systems Biology (HISB), pp.89-96, 2011.

J. Han; M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2nd, ed., 2000.

Leite, C.R.M., “Linguagem de Consulta para aplicações em Tempo-Real.” Dissertação de Mestrado, Campina Grande, PB, 2005.

Leite, C.R.M., “Arquitetura Inteligente Fuzzy para monitoramento de Sinais Vitais de Paciente: Um estudo de caso em UTI”, M.S. thesis Dept. Elect. Eng. and Comput, U.F.R.N – Natal, RN, 2011.

Liu, J. W. S. Real-Time Systems. 1. ed. 2000.

Lopes, M.A.S., “Controle de Concorrência em Sistemas de Tempo Real Utilizando Redes Neurais Artificiais”. Dissertação de Mestrado, Mossoró, RN, 2011.

Machado, A.; Augustin, I. Sistema Pervasivo de Informação em Saúde Projetado para ser Programado pelo Usuário Clínico. v. 20, n.1, 2013.

Malucelli, Andreia et al . Classificação de microáreas de risco com uso de Mineração de Dados. Rev. Saúde Pública, São Paulo, v.44, n.2, Apr. 2010. Disponível em:<http://www.scielosp.org/scielo.php?script=sci_arttext&pid=S0034-89102010000200009&lng=en&nrm=iso>. Acessado em 16/01/2012.

Minnie, D; Srinivasan S., "Application of Knowledge Discovery in Database to Blood Cell Counter Data to Improve Quality Control in Clinical Pathology,", 2011 Sixth International Conference on Bio-Inspired Computing: Theories and Applications, pp.338-342, 2011.

Moura, K.H.S. Mineração de Dados Petrolífero Utilizando a Ferramenta WEKA. Monografia – UERN. Mossoró, 2007.

Neto. C. B. Pressão Arterial: Conceito e Técnica. Publicado: Revista da APCD de São Caetano do Sul - Espelho Clínico - Ano VIII, Nº45, p. 10-11, Ago 2004.

Neto, P. R. et al. Uma aplicação de bancos de dados em tempo-real para redes de sensores. VI Workshop de Tempo Real(WTR) - SBRC, p. 45_52, 2004.

Oliveira, S. P.; Garcia, A. C. P. Variáveis e indicadores para análise de recursos humanos em saúde no Brasil – Rio de Janeiro: ENSP/FIOCRUZ. 2006.

PhysioNet/MIMIC II - Multi-parameter Intelligent Monitoring for Intensive Care (MIMIC II) Database [Available online at:http://www.physionet.org/physiobank/database/mimic2db/] Accessed: december, 12, 2011.

PostgreSQL. [Available online at: http://www.postgresql.org.br/downloads]. Accessed december, 20, 2012.

Perkusich, M.; Turnnel, M. d. F.; Perkusich, A. Modelagem de Banco de Dados em Tempo-real. XIV Simpósio Brasileiro de Banco de Dados - SBBD'99, 1999.

Ramamrithman, K. Real-Time Databases. Distribuited and Parallel Databases. v. 1, 1993. 199-226 p.

Ramos, I., Santos, M.Y. Data Mining no suporte à construção de Conhecimento Organizacional. Atas da 4ª Conferência da Associação Portuguesa de Sistemas de Informação, Porto.Salamonson, Y., Everett, B. 2003.

SECCHIN, Laura de Souza Bechara. UTI: Onde a vida pulsa. Epistemo-somática [online] 2006, vol.3, n.2, pp. 223-230. ISSN 1980-2005.

Silbershatz, Abraham; Korth, Henry F; Sudarshan, S. Sistema de banco de dados. Tradução de Marilia Guimarães Pinheiro, Claudio Cesar Canhette. 3. ed. São Paulo: Makron Books, 1999. 778 p. ISBN 85-346-1073-8.

Steiner, M. T.; Soma, N.Y.;Shimizu, T.; Nievola, J.C.; Neto, P.J.. "Abordagem de um problemamédicopormeio do processo de KDD com ênfase à análise exploratória dos dados," Gestão e Produção, vol.13, n.2, pp. 325–337, mai.-ago. 2006.

Sun Microsystems. The Real Time Specification for Java. [S.l.], Acesso em março 2013. Disponível na Web, http://rtsj.dev.java.net.

Silva, M. P. S. Mineração de dados – Conceitos, Aplicações e Experimentos com Weka [online] portalsbc.sbc.org.br/download.php?paper=35, 2004.

Takemura, T.; Shimai, K.; Matsui, H.; Manabe, S.; Ashida, N.; Yoshihara, H., "An extraction of medical knowledge on text mining for ubiquitous medicine," *Enterprise Networking and Computing in Healthcare Industry, 2004. HEALTHCOM 2004. Proceedings. 6th International Workshop on* , vol., no., pp.114,117, 28-29, 2004 doi: 10.1109/HEALTH.2004.1324488

Tesanovic, A. et al. Embedded databases for embedded real-time systems: A component-based approach. [S.l.], 2002.

Tesanovic, A.; Manev, G.; Pechenizkiy, M., E. Vasilyeva, "eHealth Personalization in the Next Generation RPM Systems," Computer-Based Medical Systems, 2009. CBMS 2009. 22nd IEEE International Symposium, PP.1-8, 2009.

Tsumoto, S.; Yokoyama, S.; Matsuoka, K., "Mining Risk Information in Hospital Information Systems as Risk Mining," *Complex Medical Engineering, 2007. CME 2007. IEEE/ICME International Conference on* , vol., no., pp.1917,1921, 23-27 May 2007 doi: 10.1109/ICCME.2007.4382082.

Valentim, R. A. de M; Morais, A. H. F, Bezerra, H. U; AraújoJúnior, H. B; Xavier, M. A; Guerreiro; A. M. G; Brandão, G. B; Araújo, C. A. P. MP-HA: Multicycles Protocol for Hospital Automation over Multicast. World Congress on Engineering (WCE 2008), ICSBB 2008 - The 2008 International Conference of Systems Biology and Bioengineering. London, U.K., 2-4 July 2008.

Vianna, R. C. X. F. Mineração de Dados e Características da Mortalidade Infantil. Secretaria do Estado da Saúde do Paraná, Curitiba, Brasil. 2010.

Virone, G.; Sixsmith, A. Activity Prediction for In-home Activity Monitoring. Intelligent Environments, 2008 IET 4th International Conference on, p. 1-4, ISSN 0537-9989, 21-22 July 2008.

Weka. [Available online at: http://www.cs.waikato.ac.nz/ml/weka/]. Accessed january, 03, 2012.

Witten, I. H. & Frank, E. Data Mining: Practical Machine Learning Tools and Techniques, second edition, Elsevier: San Francisco, ISBN 0-12-088407-0, 2005.

Wehrmeister, M. A. Framework Orientado a Objetos para projetos de Hardware e Software Embarcado para Sistemas Tempo-Real. Dissertação de Mestrado, UFRGS. Porto Alegre, 2005.

Yanbiao, Z., Cunxi, X. Zhaohua, L. "Intelligent Analysis System in Time Series of Smart Health Home On-line Monitoring Data", IEEE International Conference on Control and Automation, (ICCA) 2007, pp.1785-1790, 2007.

Zhao, J.; Wang, T, "A general framework for medical data mining," *Future Information Technology and Management Engineering (FITME), 2010 International Conference on*, vol.2, no., pp.163, 165, 9-10 Oct. 2010

Anexo A

A implementação do OpMM aborda integração com banco de dados e a restrição de tempo, que recebe pedidos de execução de consultas. A Classe *SQLCompiler* é utilizada para descobrir o tipo de comando e decidir a maneira que as consultas devem ser executadas. Caso os comandos iniciem com Period ou Initialize, faz uso das Classes *Period* e *Initialize* para execução dos mesmos, do contrário executa os comandos de acordo com o exigido pelas demais primitivas. Possui métodos para avisar aos interessados (SQL Listener's) sobre o resultado da execução de um comando. E parte da Classe SQL é implementada conforme Código a seguir:

```
public class SQLCompiler {


	private Logger log;
	private SQL sql;


	private final String INITIALIZE = "INITIALIZE";
	private final String TERMINATE = "TERMINATE";
	private final String PERIOD = "PERIOD";
	private final String COMPUTATION = "COMPUTATION";
	private final String RETURN_TC = "RETURN_TC";
	private final String VALIDATE = "VALIDATE";


	private final String SYNTAX_INIT_TERM_MSG = "Try again! \nCorrect sintax:
\nINITIALIZE AT (dd/mm/yyyy hh:mm:ss) \nTERMINATE AT (dd/mm/yyyy hh:mm:ss)";
	private final String SYNTAX_PERIOD_MSG = "Try again! \nCorrect sintax:
PERIOD_INITIALIZE (hh:mm:ss)";
	private final String SYNTAX_COMPUTATION_MSG = "Try again! \nCorrect sintax:
COMPUTATION (mm:ss)";
	public SQLCompiler(SQL sql) {
		this.sql = sql;
		log = Logger.getLogger(SQLCompiler.class);
		BasicConFigurator.configure();
```

```
            log.setLevel(Level.WARN); }

      public boolean hasPrimitive(String command) {

            BufferedReader reader = new BufferedReader(new
StringReader(command));

            String line = "";

            boolean isPrimitive = false;

            try {

                  line = reader.readLine();

                  while (!isPrimitive && line != null) {

                        StringTokenizer tokenizer = new StringTokenizer(line);

                        if(tokenizer.countTokens() > 0)

                              isPrimitive = isPrimitive(tokenizer.nextToken());

                        line = reader.readLine();

                  }

            } catch (IOException e) {

                  log.warn(e.getMessage());

            }

            return isPrimitive;

      }



      public ArrayList compile(String sql2) {


            ArrayList lines = readLines(sql2);

            if(lines.size() < 2){

                  lines.add(new Integer(sql.ERROR));

                  lines.add("Verifique se o comando SQL-99 está na mesma linha da
primitiva");

                  return lines;

            }

            String firstLine = ((String) lines.get(0)).toUpperCase();

            String lastLine = ((String) lines.get(lines.size() -
1)).toUpperCase();
```

```
            if (firstLine.startsWith(INITIALIZE))              return
initialize(lines, firstLine);

            if (firstLine.startsWith(PERIOD))                  return
period(lines, firstLine);

            if (lastLine.startsWith(VALIDATE))

                  return validate(lines, lastLine);

            if (lastLine.startsWith(TERMINATE))                return
terminate(lines, lastLine);

            if (lastLine.startsWith(COMPUTATION))

                  return computation(lines, lastLine);

            if (lastLine.startsWith(RETURN_TC))

                  return computationTime(lines);

            log.debug("Não deveria chegar aqui");

            return null;

      }



private boolean isPrimitive(String str) {

            return str.equalsIgnoreCase(INITIALIZE)

                        || str.equalsIgnoreCase(TERMINATE)

                        || str.equalsIgnoreCase(PERIOD)

                        || str.equalsIgnoreCase(COMPUTATION)

                        || str.equalsIgnoreCase(VALIDATE)

                        || str.equalsIgnoreCase(RETURN_TC);

      }



      private ArrayList readLines(String sql) {

            ArrayList lines = new ArrayList();

            BufferedReader reader = new BufferedReader(new StringReader(sql));

            String line;

            try {
```

```
                line = reader.readLine();
                while (line != null) {
                        lines.add(line);
                        line = reader.readLine();
                }
        } catch (IOException e) {
                log.warn(e.getMessage());
        }
        return lines;
}

        private String readSql(int min, int max, ArrayList lines){
        String sql = "";
        for (int i = min; i < max; i++)
                sql = sql + "\n" + lines.get(i);
        return sql;
}


private ArrayList initialize(ArrayList lines, String firstLine) {
        ArrayList result = new ArrayList();
        try {
                String[] date_hour = treatInitializeTerminate(firstLine);
                String lastLine = ((String) lines.get(lines.size() -
1)).toUpperCase();
                if(lastLine.startsWith(TERMINATE))
                        return initializeTerminate(lines, lastLine, date_hour);
                result.add(new Integer(sql.INITIALIZE));
                result.add(date_hour[0]);
                result.add(date_hour[1]);
                result.add(readSql(1,lines.size(),lines));
        } catch (IncorrectSintaxException e) {
```

```
                log.warn(e.getMessage());
                result.add(new Integer(sql.ERROR));
                result.add(e.getMessage());
        }
        return result;
    }
```

A Classe responsável por preparar os dados e minerar fazendo uso do algoritmo de mineração J48, recebe a base de dados no formato *ResultSet* pelo método*setRs*(), após isso deve ser chamado o método *minerar*( ) onde a base é convertida para um objeto do tipo *Instances* pelo Objeto *InstaceQuery* e passado por parâmetro para o método *buildClassifier*() do objeto J48, sendo por ultimo avaliado por um objeto do tipo *Evaluation*, possui também o método *getResultados*( ) que retorna uma *string* com a saída formatada contendo o resultado da avaliação do objeto *Evaluation* semelhante a aplicação WEKA e o método *getArvore*() retorna uma *string*.

```
Package rtdbms.db;

importjava.sql.ResultSet;
importjava.util.Random;

importweka.classifiers.Evaluation;
import weka.classifiers.trees.J48;
importweka.core.Instances;
importweka.experiment.InstanceQuery;

publicclassMineracaoDados {

    privatestaticMineracaoDados instance;

    privateInstanceQueryquery;
    private Instances data;
```

```
privateResultSetrs;
private J48 tree;
private Evaluation eval;

privateMineracaoDados() {
}

publicstaticMineracaoDadosgetInstance() {

	if(instance == null) {
		instance = newMineracaoDados();
	}

	returninstance;
}

public String minerar() {

	String retorno = "";
	try {

		query = newInstanceQuery();
		data = InstanceQuery.retrieveInstances(query, rs);

		if (data.classIndex() == -1) {
data.setClassIndex(data.numAttributes() - 1);
	  }

	tree = new J48();
	tree.buildClassifier(data);

	eval = new Evaluation(data);
```

```
        Random rand = newRandom(1);
        int folds = 10;
        eval.crossValidateModel(tree, data, folds, rand);

        } catch (Exception e) {
              e.printStackTrace();
        }

        returnretorno;
}

public String getResultados() {

        String string = "";

        string += "=== Run information ===\n" +
                    "\nScheme:       "+ tree.getClass().getName()+
                    "\nRelation:     " + data.relationName()+
                    "\nInstances:    " + data.numInstances()+
                    "\nAttributes:   " + data.numAttributes()+"\n";

        if (data.numAttributes() < 50) {
        for (int i = 0; i <data.numAttributes(); i++) {
              string += "              " + data.attribute(i).name()+"\n";
        }
}
else {
        string += "              [list of attributes omitted]\n";
}
        string += "\n\n=== Classifier model (full training set) ===\n\n";
        string += tree.toString()+"\n";
        string += "=== Stratified cross-validation ===\n" +
```

```
                    "=== Summary ===\n" +
                    eval.toSummaryString()+"\n";
try {
        string += eval.toClassDetailsString()+"\n";
        string += eval.toMatrixString()+"\n";
        } catch (Exception e) {
                e.printStackTrace();
        }

return string;

}

public String getArvore() {
        Stringstring = "";
        try {
                string = tree.graph();
        } catch (Exception e) {
                e.printStackTrace();
        }
        return string;
}

publicInstanceQuerygetQuery() {
        returnquery;
}

publicvoidsetQuery(InstanceQuery query) {
        this.query = query;
}

public Instances getData() {
```

```
        return data;
    }

    public void setData(Instances data) {
        this.data = data;
    }

    public ResultSet getRs() {
        return rs;
    }

    public void setRs(ResultSet rs) {
        this.rs = rs;
    }

}
```