UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

MINERAÇÃO DE DADOS EM ARQUIVOS DE LOG GERADOS POR SERVIDORES DE PÁGINAS WEB

LEONARDO JOSÉ CORREIA

BLUMENAU
2004

2004/1-24

LEONARDO JOSÉ CORREIA

MINERAÇÃO DE DADOS EM ARQUIVOS DE LOG GERADOS POR SERVIDORES DE PÁGINAS WEB

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Ricardo A. Azambuja - Orientador

BLUMENAU
2004

2004/2-24

MINERAÇÃO DE DADOS EM ARQUIVOS DE LOG GERADOS POR SERVIDORES DE PÁGINAS WEB

Por

LEONARDO JOSÉ CORREIA

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Ricardo A. Azambuja, Dr. – Orientador, FURB
Membro:	Prof. Ademir Goulart, FURB
Membro:	Prof. Oscar Dalfovo, FURB

Blumenau, 26 de junho de 2004

Dedico este trabalho a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste. À minha adorável esposa pelo apoio e amor. Ao meu pai (em memória).

Os bons livros fazem “sacar” para fora o que a pessoa tem de melhor dentro dela.

Lina Sotis Francesco Moratti

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, que mesmo longe, sempre esteve presente.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador prof. Ricardo Alencar Azambuja, por ter acreditado na conclusão deste trabalho.

Ao professor Paulo Rodacki, que soube me tranqüilizar e direcionar, me ajudando a compreender o algoritmo Apriori e a técnica de Regras de Associação.

RESUMO

A rede mundial de computadores provocou uma reviravolta no mundo das comunicações. O que há bem pouco tempo era trabalhoso, ficou resumido aos cliques de um mouse. Estes cliques geralmente são ordenados em seqüência, de forma a organizar os comandos de seleção da comunicação. Normalmente, as seqüências de cliques são capturadas e levadas para um ambiente familiar de banco de dados em que as empresas possam efetivamente utilizá-la. Os servidores web gravam em seus *logs* essas seqüências de cliques, destinando-se à análise de tráfego e o desempenho do servidor *web*. Este trabalho concentra-se na construção de um sistema para mineração de dados dos *logs* de *Web Services: Apache e IIS*.

Palavras-chave: *Log Mining, Web Mining, Data Mining, Apache, IIS, Log*.

ABSTRACT

The internet has caused a revolution in the world of communications. Hard tasks are now as easy as the mouse's clicking. These clicks are generally done in a sequence to organize the selection commands of the communication. Usually, the sequences of clicks are captured and recorded in a database so that the Companies can use it. The web servers records in your logs those sequences of clicks. This paper will focus on the construction of a system to mine the data obtained from the logs of the Apache and IIS Web Services destining in the analysis of traffic and the acting of the *web server*.

Key-Words: Log Mining, Web Mining, Data Mining, Text Mining, Apache, IIS, Web Service, Log.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AIFF - Attribute-Relation File Format

Authuser - Authenticated User

CDR - Call Detail Record

CGI - Common Gateway Interface

CLF - Common Log Format

CSV - Comma Separated Volume

DNS - Data Source Name

ECLF - Extended Common Log Format

FTP - File Transfer Protocol

GUI - Graphical User Interface

GUID - Global User ID

HTML – Hyper Text Markup Language

Identd - Identification Daemon.

IIS - Internet Information Services

IP - Internet Protocol

KDD - Knowledge Discovery in Database (Descoberta do Conhecimento em BD)

NCSA - National Center for Supercomputing Applications

NLP - Natural Language Processing

OLTP - On-line Transaction Processing

PHP – Personal Home Page

RAM - Random Access Memory

SL - Secure Sockes Layer

URL - Uniform Resource Locator

URI - Uniform Resource Indicator

UTC - Universal Time Coordinate

W3C - World Wide Web Consortium

WWW - World Wide Web

WEKA - Waikato Environment Knowledge Analysis

LISTA DE SÍMBOLOS

% - por cento

- sustenido

/ - barra

' - aspa simples

- hífen

=>

SUMÁRIO

1 INTRODUÇÃO

O uso dos *Website*[1] para busca de informações ou realizar transações comerciais têm se tornado comum nos dias de hoje. A *web* fornece uma nova fonte de dados, denominada como seqüência de cliques, a qual é literalmente, um *log*[2] de cada gesto efetuado por visitante a cada *site* da *web*. Nesse contexto, surgem os mecanismos que resgatam esses dados transformando-os em informações propriamente dita.

Estes mecanismos são compostos por técnicas de *Data Mining*[3] (mineração de dados) para extrair tais informações, aplicado as mais diversas áreas de concentração dessas informações. Segundo Xue (2000), os métodos de *Data Mining*, integrados com o serviço das máquinas de busca *web*, aumentam a qualidade das buscas. A máquina de busca pode assim, procurar um conjunto de documentos relevantes, que é menor que o conjunto procurado pelos métodos atuais.

Freqüentemente, pesquisadores precisam limpar, condensar e transformar os dados coletados para recuperar e analisar informações úteis. Eles têm usado arquivos de *log* para analisar o desempenho do sistema, melhorar o sistema de *caching* e determinar a natureza do tráfego na *web*. Com esta técnica, analisando e explorando regularmente os registros de *log* da *web*, pode-se melhorar a qualidade de entrega de serviços de informação da internet para o usuário final, permitindo o ganho de desempenho dos servidores, identificando consumidores em potencial para o comércio eletrônico. A análise de *log* também pode ajudar a construir serviços personalizados para usuários, e ainda, registrar sinais diretos de satisfação e descontentamento. O procedimento permitirá responder efetivamente ao cliente individual.

Segundo Tulloch (2001) & Kabir (2002), os gerenciadores de *Website* IIS e Apache já têm a capacidade de guardar estatísticas desses acessos em arquivos, os chamados *logs* de acesso. No entanto esses *logs* têm pouco valor se as informações extraídas não se apresentarem de forma objetiva e gerencial.

---

[1] WebSite é um concentrador de informações que foram publicadas ou postadas em formato HTML.

[2] Seqüência de cliques que identificam uma atividade de busca de um assunto na web

[3] Área da Computação especializada na mineração de dados.

Pelas inúmeras vantagens existentes em uma arquitetura *web*, as empresas têm investido na criação de *softwares* que executam e acessam páginas na Internet, centralizadas em servidores espalhados na rede com o auxílio de um navegador.

Com o sucesso da Internet, tem havido uma proliferação de jogadores, com investimentos econômicos e intelectuais na rede. Todo esse sucesso será alavancado em grande parte, pela qualidade da resposta dos *sites* comerciais da *web* a seus visitantes individuais.

Dentre as vantagens existentes em uma arquitetura *Web* e graças as suas potencialidades, hoje podemos ver nas páginas da Internet, documentos formatados (cores, efeitos e etc...), escutar música, assistir a vídeos, e muito mais. Enfim, através da Internet podemos transferir arquivos e realizar outras inúmeras operações.

Segundo Rodrigues (2000), ao longo do tempo se percebeu que a velocidade de coleta de informações era muito maior que a velocidade de processamento das operações ou análise delas. Num ambiente comercial mutável, torna-se necessário à aplicação de técnicas e ferramentas que agilizem o processo de extração de informações relevantes de grandes volumes de dados.

Entre as técnicas de extração destacamos a metodologia de Descoberta do Conhecimento em Banco de Dados que veio preencher essa lacuna na necessidade de análise desse volume de informação. Para que o conhecimento seja descoberto, técnicas de *Data Mining* devem ser utilizadas. Nesse contexto que desenvolveremos o trabalho.

A tecnologia provocou mudanças, sendo que, uma delas, refere-se ao aumento da importância da informação no mundo científico e comercial. A informação se tornou um grande diferencial, permitindo, por exemplo, no caso de uma empresa, o auxílio desde a tomada de decisão até a descoberta de fraudes ou perfil de consumidores (FAYYAD et al., 1996).

No entanto, para que as informações sejam extraídas corretamente, é necessária a utilização de técnicas e ferramentas que propiciem a descoberta ou mineração de padrões (MITTCHEL, 1997). Na verdade, a extração de padrões é apenas uma etapa de um processo maior denominado de Extração de Conhecimento de Dados (*Knowledge Discovery in Database* - KDD), que será apresentado e aplicado no decorrer deste trabalho.

Técnicas *Data Mining* e descoberta de conhecimento, normalmente são aplicadas em conjunto com Banco de Dados pelo grande volume de dados manipulados. Neste trabalho foram utilizadas técnicas de Regras de Associação, porém, sem o uso de Banco de Dados, neste contexto pois os dados estão armazenados em arquivos de *log*, não estruturados, já divididos em tamanho limitado, permitindo seu manuseio em memória. Isto torna desnecessário o uso de Banco de Dados para esse tipo de análise.

Este trabalho apresenta o desenvolvimento de um protótipo para extração de conhecimento a partir de dados encontrados em arquivos de *log* de servidores *web*. Para isso, foram utilizados os sistemas de aprendizado do paradigma simbólico Apriori que trabalha com a tarefa de associação, disponíveis na ferramenta *Waikato Environment Knowledge Analysis* - WEKA

1.1 OBJETIVOS DO TRABALHO

1.1.1 Objetivos Gerais

Construir um protótipo para mineração de dados em arquivos de *log* gerados pelos *Web Servers: Apache e IIS,* capaz de ler um arquivo de *log* gravado nos formatos permitidos pelo gerenciador de páginas *web* e aplicar técnicas de *Data Mining*, montar estatísticas e apresentá-las de forma objetiva, simples e de fácil entendimento através de uma estrutura de manipulação de dados em memória.

1.1.1.1 Objetivos Específicos

a) disponibilizar uma interface que permita descobrir como os visitantes navegam por um *Website*.
b) disponibilizar uma interface que permita descobrir quais são as páginas mais acessadas, em que horários os visitantes mais acessam e quanto tempo permanecem no *site*.
c) disponibilizar informações de quanto de banda do link esses acessos consomem;
d) disponibilizar uma interface amigável que permita ao usuário extrair informações de arquivos *log* gerados por gerenciadores de páginas web.
e) disponibilizar uma interface que permita ao usuário imprimir relatórios estatísticos com as informações obtidas;

f) disponibilizar um relatório que permita descobrir quantos bytes foram enviados e quantos *bytes* foram recebidos durante a navegação no *site*.
g) testar e validar o ambiente desenvolvido através de sua aplicação prática.

1.2 ESTRUTURA DO TRABALHO

No Capítulo 1, com uma introdução de todo o contexto desta pesquisa, a fim de propiciar uma base sólida para o entendimento do que foi pesquisado, gera novos consumidores buscando conquistá-los e fidelizá-los através da satisfação de um espectro cada vez maior de suas necessidades (SHARMA, 2001) e principalmente manter a fidelidade dos clientes que a empresa já possui.

As pesquisas e análises realizadas para o planejamento e desenvolvimento do protótipo para mineração de dados em arquivos de *log* são apresentadas no Capítulo 1, com uma introdução de todo o contexto desta pesquisa, a fim de propiciar uma base sólida para o entendimento do que foi pesquisado.

No Capítulo 2, apresenta-se as técnicas de *Data Mining*, suas aplicações e como foi utilizada durante o desenvolvimento deste trabalho.

No Capítulo 3, *Web Servers*, em especial *Apache* e *IIS Websites.*

No capítulo 4, apresenta-se os arquivos de *log*, mostrando como os dados de acesso aos *sites* são armazenados e que tipo de informações podem e não podem ser obtidas através desses *log*s.

No capítulo 5, apresenta-se o processo de implementação do Protótipo proposto no objetivo deste trabalho.

Finalmente, apresenta-se uma conclusão de todas as análises e pesquisas realizadas, além de algumas possibilidades de continuação do trabalho.

2 DATA MINING

Mena (1999) define *log* como sendo um arquivo que contém diversas transações. Num contexto geral, *log* é definido como o registro das operações de processamento em um computador. Em um arquivo de *log* são gravados todos os eventos ou transações que ocorreram em ordem cronológica.

Segundo Kimball (2000), transação pode ser definida como sendo um conjunto de operações que ocorreram em um dado instante. Em um banco de dados, por exemplo, um *log* de transação é um registro serial de todas as modificações que ocorreram em um determinado intervalo de tempo. No caso de servidores *web*, as transações representam a atividade do *site*. Cada vez que uma pessoa visita um *web site* o arquivo de *log* é atualizado com as páginas que foram acessadas ou ignoradas e diversas outras informações que serão mostradas a adiante.

Segundo Mena (1999), um acesso a uma página *web*, ou um arquivo gera um *hit* no servidor *web*. Por exemplo, se a página contém 10 figuras, uma visita a essa página gera 11 *Hits*, sendo 1 (um) *hit* para a própria página e 10 *hits* para as figuras. Se um visitante acessa 3 páginas e cada uma delas contém 10 figuras, no *log* desse servidor serão gravados 33 *hits*, 3 páginas e 1 visitante.

A área de *Data Mining* é por si só de extrema relevância atualmente. Esta importância é destacada pelo crescimento expressivo da *Internet*, que recebe a cada dia um grande volume de informações.

Através de pesquisas realizadas para o desenvolvimento deste trabalho, identificou-se a existência de algumas ferramentas que auxiliam o trabalho de extração de informações para servidores de páginas *web*. Estes softwares apresentam custo elevado, talvez por integrarem um pacote completo, com outros aplicativos, não justificando sua compra pelas empresas que necessitem apenas desse tipo de ferramenta. Além disso, não há conhecimento ou registros de um trabalho semelhante desenvolvido no meio acadêmico, tampouco como Trabalho de Conclusão de Curso desta instituição de ensino. Na pesquisa, foram encontrados alguns softwares disponíveis no mercado e que são correlatos a este trabalho, sendo:

a) *123LogAnalyzer* (ZY COMPUTING, 2003);

b) *Funnel Web Professional* (QUEST SOFTWARE, 2003);

c) *HitBox Professional* (WEBSIDESTORY CORPORATION, 2003);

d) *NetTracker* (SANE SOLUTIONS, 2003);

e) *SuperStats* (VERISIGN COMPANY, 2003);

f) *Urchin* (URCHIN SOFTWARE CORPORATION, 2003);

g) *WebTrends Reporting Center* (NETIQ CORPORATION, 2003).

A Tabela 1, adaptada de PETTT (2003), faz uma comparação de algumas das ferramentas de análise de *log* da *web* disponíveis no mercado.

Segundo Batista (2001), a massificação do uso do *Internet* fez a extração automática do conhecimento dos arquivos de *log* da *web* uma necessidade. Os provedores de informação estão interessados nas técnicas que poderiam aprender informações de necessidades e preferências dos usuários da *web*. Isto pode ser usado para melhorar a eficácia desses *web sites* adaptando a estrutura da informação dos *sites* ao comportamento dos usuários. Entretanto, é difícil encontrar ferramentas apropriadas para analisar dados de registro crus dos arquivos de *log* da *web* para recuperar a informação significativa e útil. Atualmente, há diversas ferramentas genéricas de análise do registro de *log* da *web*, mas a maioria delas não é muito apreciada por seus usuários e são consideradas demasiadamente lentas, inflexíveis, caras, difíceis de manter ou muito limitadas nos resultados que podem fornecer. É possível que futuras versões dos sistemas operacionais e gerenciadores de páginas *web*, já tragam consigo uma série de estatísticas de seus próprios *log*s.

Esta evolução nas características dos sistemas operacionais e gerenciadores de páginas *web* pode ser considerada uma tendência natural, assim como já evoluíram até o momento. Segundo Kabir (2002) à medida que os servidores *web* começaram a aparecer no mercado, também começaram a proliferar programas de análise de servidores *web*. Esses programas se tornaram parte do dia-a-dia de muitos administradores *Web*. Como conseqüência desta proliferação surgiu a era das incompatibilidades de arquivos de *log*, o que tornou difícil e incomoda a análise de *logs*; um único software de análise não funcionava com todos arquivos de *log*.

Com o advento da especificação *Common Log Format* – formato de *log* comum (CLF) permitiu que todos os servidores *web* escrevessem *logs* de maneira razoavelmente semelhante, tornando mais fácil à análise de *logs* de um servidor para outro (KABIR, 2002, p. 239).

Capturar transações é simples, e é o que tem sido feito pelos *Warehouse*s de dados, por mais de dez anos. Atualmente este trabalho torna-se mais ardiloso: capturar, analisar e entender o comportamento dos usuários que clicam nos *sites* da *web* (KIMBALL, 2000, p. 21).

Cliques de usuários nos *sites* da *web* são denominados como uma seqüência de cliques que é potencialmente um registro muito melhor de comportamento do que outras fontes detalhadas de dados mais tradicionais. Os famosos dados de registro de detalhe de chamada *Call Detail Record* (CDR) das companhias de telecomunicações empalidecem quando comparados à seqüência de cliques. Mesmo as fontes de dados de processamento de transações on-line *On-line Transaction Processing* (OLTP) muito importantes omitem muitas informações interessantes. A seqüência de cliques é na realidade, uma coleção desenvolvida de fontes de dados. Há mais do que uma dúzia de formatos de arquivos de *log* para capturar dados de seqüência de cliques. Estes formatos de arquivos de *log* têm componentes opcionais de dados que, se utilizados, podem ser muito úteis na identificação de usuários, de sessões e do verdadeiro sentido do comportamento (KIMBALL, 2000, p. 21).

Features List

URL	www.bazaarsuite.com	www.netgen.com/produccts/netanalysis/index.shtml	www.accrue.com/products/insight.html	www.macromedia.com/ebusiness	www.webtrends.com/products/log/default.htm	www.ics.uci.edu/pub/websoft/wwwstat/
Name	Bazaar Analyzer	NetAnalysis	Insight	Aria	Web Trends	wwwstat
Individual Counts						
Hits					✔	
Page Views					✔	
Time/page			✔			
Date/time of transaction						
Session Information						
Visitor sessions					✔	
Session time			✔		✔	
Pages/session						
Unique visitors					✔	
One-time visitors					✔	
Multi-session visitors					✔	
Location of user	✔		✔		✔	
Site Analysis						
Path Analysis					✔ list path--steps	
Visualization	✔ by individual only	something	BuyPath		no	
Entry pages	✔ & exit		exit		✔ & exit	
Hi-freq page	✔				✔	
Low-freq page	✔				✔	
Errors	✔					
Search terms within site						

Tabela 1 – Comparação de algumas ferramentas de análise de *log* da web. Adaptada de PETTT (2003).

External Information						
Referring url	✔		✔ & search engines		✔	
Bookmarked					Solaris, RHLinux, NT/2000	
Search term to find site					✔	
Backoffice/Interface Info.						
Interface		browser?	browser?		browser, nice	
Run location			us--w/sup			
Data format						
Platforms		NT, Sun Solaris Oracle & Sybase, IBM	HTTP web server, Sun Solaris, NT			
Export format	✔					
Cost	$1000 unlimited users	$20,000 +	$17,000 +			
General Impressions						
	few report options apparent	biz--enterprise solution. Lots of reports. Looks promising, no demo reports online	aimed at big biz for developing "unified view of the customer." More appropriate than HitList. See also Vista --suite of reporting options	strong package--biz oriented. No online report demos.	Enterprise Reporting Server, Enterprise Suite; designed for hi-traffic sites	

Tabela 1 – Comparação de algumas ferramentas de análise de *log* da web (Continuação). Adaptada de PETTT (2003).

Das 20 ferramentas encontradas na planilha original, disponível em PETTT (2004), seis mereceram a análise, aquelas que as características são convergentes com o nosso propósito, as informações seguem abaixo:

wwwstat (Unix/Linux)

Preço: Livre

Originalmente desenvolvida por Roy Fielding como parte do projeto Acádia e *WebSoft* da Universidade da Califórnia. Eles criaram a ferramenta de análise de *logs* que trabalha no Unix e ambientes *Linux*. O ***wwwstat*** proporciona estatísticas básicas com um mínimo de exibição e informações acessórias. Na parte superior, você pode chegar ao código fonte e pode modificar o software para ajustar às suas necessidades específicas. Um programa similar, o ***gwstat***, apresenta a produção do ***wwwstat*** e os gráficos ilustram seu tráfico.

netAnalysis 4 (Sol Solaris, IBM AIX, o Windows NT)

Preço: U$20.000 e acima

Dependendo do sistema operacional e opções de suporte e treinamento o ***netAnalysis*** prevê 150 tipos de relatórios completos gerados pela *Web* local, o ***netAnalysis*** pode automatizar o processo de análise de dados de *Web* para informar e controlar produção e distribuição. A característica de programação o deixa recobrar automaticamente, importar, e estocar dados da *Web* diários, semanais, ou mensais. Cria histórico em tempo-real, informa as capacidades. O ***netAnalysis*** permite localizar o comportamento dos usuários e suas interações locais.

Bazaar Analyzer Pro 2.0 (desenvolvido em Java, avaliação por qualquer plataforma)

Preço: U$320 para a licença de um único-usuário; U$999 para usuários ilimitados.

Analisador de bazar Pró 2.0 é veloz, flexível, e fácil usar. Foi feito em ferramenta Java, é multiplataforma e pode analisar grandes arquivos de *log* de até 100MB em menos de três minutos. Pode gerar relatórios na Internet local como também relatórios de servidores de acessos. Podem ser gerados relatórios do comportamento interno do software ou podem ser exportados a outras ferramentas como Word ou Excel. A característica de *path View* lhe dá

uma exibição em display gráfico do usuário em seu local de *Web* determinando a otimização do layout da sua *Web-local*. O filtro flexível permite a análise e informações do acesso ao servidor de acesso de até sete camadas externas dos sub-domínios do produto.

WebTrends Log Analyzer 4.52 (Windows 95/98/NT)

Preço: U$399; U$478 para plano de subscrição mais plano de apoio e atualizações grátis durante um ano.

O ***WebTrends*** produz uma gama extensa de produtos para minerar dados na *Web* e tráfico informação, para administração, análise e desempenho de tráfico de *Web*, e segurança da *Web local*. Sua base é o produto de analise de *log*, de *bak-bones* com pacotes de baixo uso e baixo custo com muitas características úteis e valiosas. Relatórios gerados pelo **WebTrends** de analise de *logs* dão ao gerente da *Web* uma clara e fácil leitura das mesas em gráficos coloridos que mostram tendências, uso de *band wicth*, e do mercado. Você pode produzir a visão dos relatórios em Microsoft Word e Excel e em HTML, texto ASCII, e delimitador de formatos. O analisador de *logs* reconhece dados de *logs* do apache, *Netscape*, *Microsoft*, *Oracle*, *Dominó*, *IBM*, *CERN*, *NCSA*, e outros servidores, e trafegar dados até 30MB por minuto. O analisador de *logs* localiza sessões de usuário com procura de página, *download* de arquivo, execução de *scritps*, cidades, e outros dados, lhe dando acesso a dados críticos para análise. Se você estiver usando sua *Web* em Windows, o ***WebTrends*** analisador de *logs* deveria estar em sua caixa de ferramentas.

Aria 3.0 (Windows NT, Sun Solaris, Netscape Enterprise, Apache, Microsoft IIS)

Preço: negociado no pedido (não fornecido)

O Ária de Andromedia foi a escolha de *CNET Builder* como a melhor ferramenta de análise de tráfico em 1998. A mais recente versão melhora em uma expansão impressionante de ferramentas. Disponível em três módulos diferentes, Andromedia prevê ferramentas para *host* simples (Ária), ou multi-servidores locais geograficamente espalhados (Empreendimentos Ária), e locais de e-comércio plataformas de e-comércio em três dimensões correntes (*e-commerce* de Ária). Três componentes: o Monitor, um servidor API; o gravador que recebe processos, e armazena dados em tempo real; e o reportador que produz de hora em hora os relatórios *on-demand* -- monitorando o trabalho para permitir acesso e desempenho vital de dados. Ária pode analisar as URLs e os sistemas de publicação,

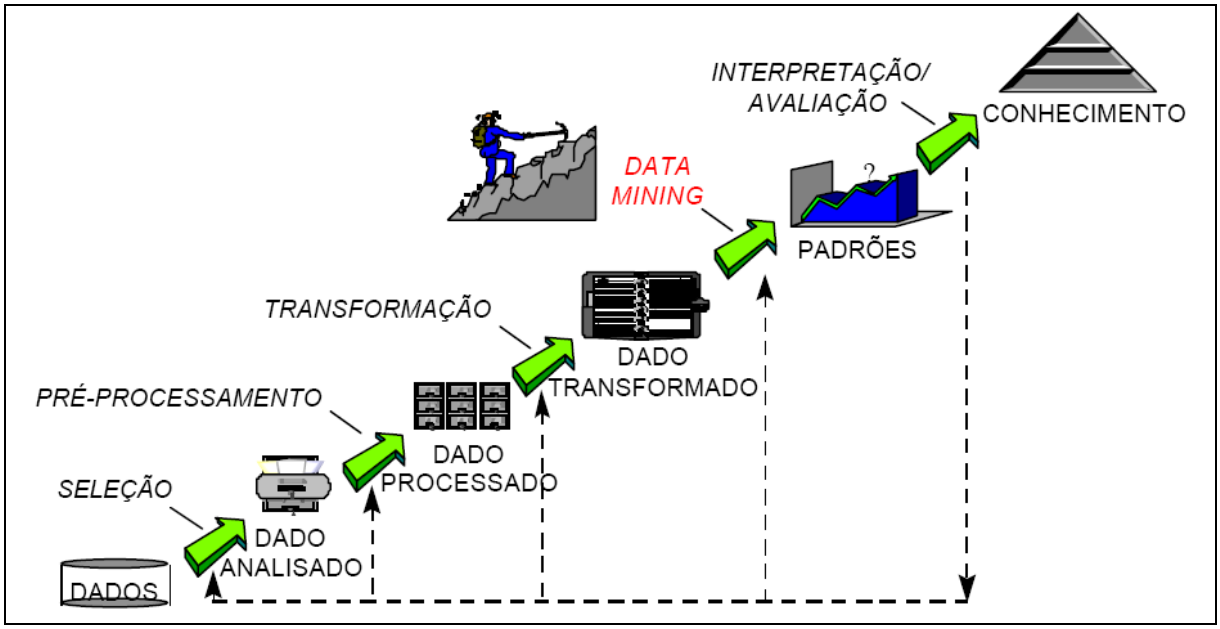
aplicação de servidores *Web*, e *In-House CGI* aplicações de *Java*, localizando URLs dinâmicas que recorrem ao mesmo conteúdo como se recorresse a uma única URL. Seus relatórios são mais significantes, enquanto focaliza nos agrupamentos atuais.

Accrue Insight 3.0 (Solaris 2.5+ or IRIX 6.4+; Java-enabled browser required)

Preço: $17,000 e acima

Usando o pacote farejador, ***Accrue Insight*** monitora transações de rede e arquivos de *logs*, enquanto relata os dados específicos em seu local de *Web* e suas visitas. ***Accrue Insight*** pode lhe informar de fato o número de páginas abertas, não só as visitadas. Também faz o registra de quantos usuários carregam uma página, a velocidade de conexão de uma visita, e outras métricas. ***Accrue Insight*** é potente, pois foi projetado para operar efetivamente por locais de *Web* que recebem 50 milhões de acessos por dia e centenas de servidores de *Web* que suportam mais de 2,000 locais de *Web*.

Segundo Fayyad et al. (1996), o processo de *Data Mining* é apenas um dos estágios no processo de descoberta do conhecimento, simplificando e dividindo em etapas conforme demonstra a Figura 1.



Fonte: Fayyad (1996, p.37-54).

Figura 1 - Etapas do Processo KDD.

O problema de automatizar análise de dados cresceu debaixo do rótulo de *Data Mining* ou, mais geralmente, *Knowledge Discovery in Database*. KDD não é uma técnica nova, ao

invés disso, é uma área de pesquisa bastante interdisciplinar, agrupada de resultados de pesquisas de várias comunidades científicas como: tecnologia de banco de dados, estatísticas, aprendizado de máquina, e visualização de dados (FAYYAD, 1996).

De acordo com Adriaans & Zantinge (apud Rodrigues, 2000, p. 12), existe uma confusão entre os termos *Data Mining* e KDD (*Knowledge Discovery in Databases* ou Descoberta de Conhecimento em Banco de Dados). O termo KDD é empregado para descrever o processo de extração de conhecimento de um conjunto de dados. Neste contexto, conhecimento significa relações e padrões entre os elementos dos conjuntos de dados. O termo *Data Mining*, segundo os autores, deve ser usado exclusivamente para o estágio de descoberta do processo de KDD. Nesse contexto, o processo de KDD se divide em 7 estágios: (1) Definição do Problema, (2) Seleção dos dados, (3) Eliminação de incongruências/erros dos dados ("limpeza" dos dados), (4) Enriquecimento dos dados, (5) Codificação dos dados, (6) *Data Mining* e (7) Relatórios.

Segundo Fayyad et al. (1996), *Data Mining* é um passo do processo de KDD que está definido como um processo não trivial de identificar padrões válidos, originais, potencialmente úteis, e por fim compreensíveis em dados. O termo padrão aqui se refere alguma representação abstrata de um subconjunto de dados nos dados, quer dizer, uma expressão em algum idioma que descreve um subconjunto de dados ou um modelo aplicável para este subconjunto.

Na Figura 2 é definida a arquitetura geral de mineração de dados em arquivos de *log* de servidor *web* que habilita a implementação genérica de um processo KDD para extração de conhecimento nos acessos gravados nesse tipo de arquivo.

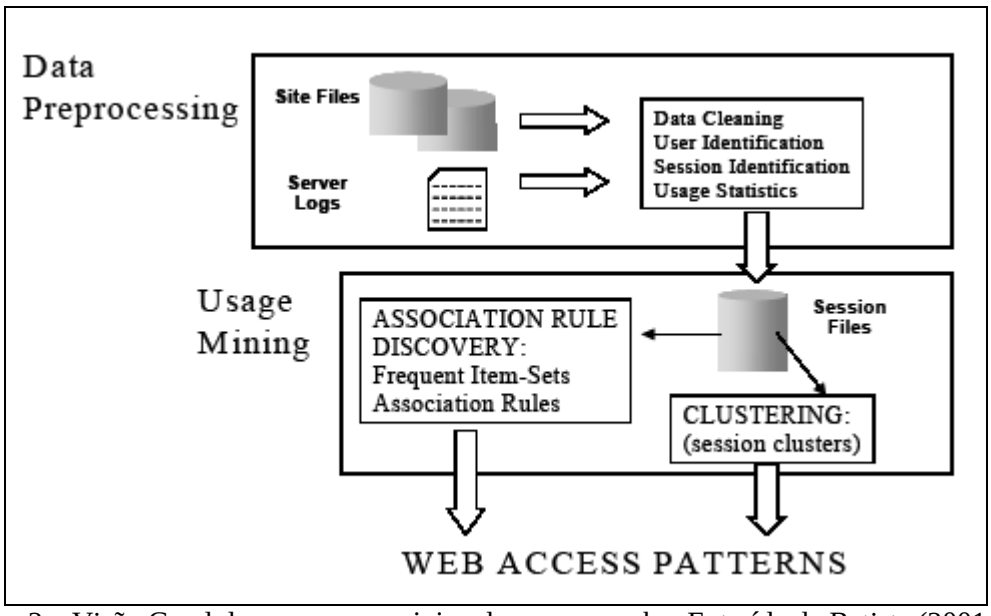


Figura 2 – Visão Geral dos processos *mining* de acessos web – Extraída de Batista (2001, p.3)

Segundo Davis King (apud Rodrigues, 2000, pg. 13) *Data Mining* é um modo de procurar relações escondidas em um grande conjunto de dados. Por interessante, pode ser definida qualquer estrutura necessária para investigação, como padrões de *clustering*, aproximações de funções. As técnicas de *Data Mining* podem ser a princípio semelhantes às análises de regressão.

Segundo Davis King (apud Rodrigues, 2000).O método científico de *Data Mining*, consiste basicamente de 5 etapas: (1) Definir o problema, (2) Gerar hipóteses / modelos, (3) Coletar dados / conduzir experimentos que gerem dados, (4) Testar modelos em confrontação com os dados e (5) Utilizar os resultados para gerar novas hipóteses. Raramente é um processo automatizado, com uma grande intervenção do analista que conduz o estudo.

A aplicação típica de *Data Mining* começa com um grande conjunto de dados e poucas definições. A maioria dos algoritmos trata os dados iniciais como uma "caixa-preta", com nenhuma informação disponível sobre o que os dados descrevem, quais relações existem entre os dados e se contêm erros. Ao examinar os dados, um algoritmo pode explorar milhares de prováveis regras, utilizando diversas técnicas para escolher entre elas.

Conforme Robert Grossman (apud, Rodrigues 1997, p. 13) *Data Mining* é definida como uma descoberta de padrões, associações, mudanças, anomalias e estruturas estatísticas e eventos em dados. A análise de dados tradicional é baseada na suposição, em que uma hipótese é formada e validada através dos dados. Por outro lado, as técnicas de *Data Mining*

são baseadas na descoberta na medida que os padrões são automaticamente extraídos do conjunto de dados.

De acordo com Bruce Moxon (apud Rodrigues, 2000, p.13), *Data Mining* é um conjunto de técnicas utilizadas para explorar exaustivamente e trazer à superfície relações complexas em um conjunto grande de dados. Uma diferença significante entre as técnicas de *Data Mining* e outras ferramentas analíticas é a abordagem utilizada para explorar as inter-relações entre os dados. Semelhantemente à abordagem dada por Grossman (apud Rodrigues, 2000 p. 13-14), também diferencia as técnicas de *Data Mining* com relação às técnicas analíticas entre a abordagem de suposição e a abordagem de descoberta. Segundo ele, as técnicas de *Data Mining* não pressupõem que as relações entre os dados devam ser conhecidas a priori.

Segundo Nardelli (2000), as expressões *Data Mining*, mineração de dados ou garimpagem de dados referem-se ao processo de extrair dados potencialmente úteis a partir de dados brutos que estão armazenados em banco de dados dos diversos sistemas implantados nas organizações. A tecnologia utilizada no *Data Mining* utiliza-se da procura em grandes quantidades de dados armazenados procurando extrair padrões e relacionamentos que podem ser fundamentais para os negócios da organização. O *Data Mining* utiliza-se de um conjunto de técnicas avançadas para identificar os padrões e associações que os dados refletem, com isso oferecendo conclusões que podem trazer valiosas vantagens em nível de mercado para a organização.

2.1 CLASSIFICAÇÕES DAS TÉCNICAS E MÉTODOS USADOS EM *DATA MINING*.

A fim de organizar as técnicas e métodos mais utilizados no processo de *Data Mining*, para um melhor entendimento do assunto, são apresentadas a seguir, abordagens dos principais conceitos, sob uma classificação do ponto de vista de autores diferentes dando ênfase ao método de Regras de Associação que foi objeto de estudo deste trabalho. O objetivo não é esgotar o assunto, mas apresentar uma idéia sucinta do tema. Para um entendimento mais aprofundado e detalhado sugere-se a leitura do material citado como referência.

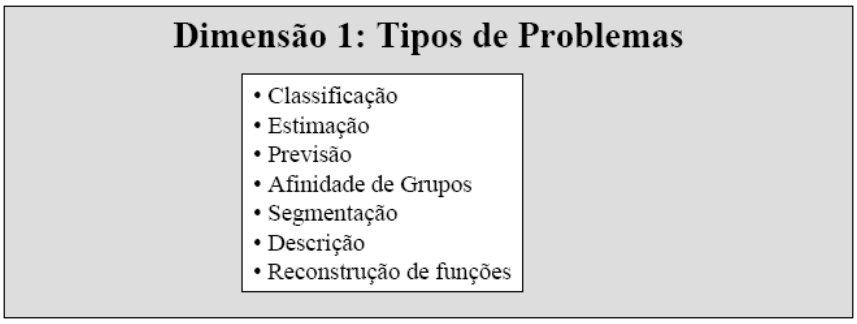
Segundo Batista (2001, p. 2), “Os passos de *Data Mining* referem-se à aplicação de métodos e ao tipo de descoberta podendo ser não-supervisionados e supervisionados (automatizado ou humano-assistido) a fim de determinar padrões de, ou ajustando modelos

para, dados observados. Este passo implica em seguir identificando as funções de *Data Mining* e escolhendo os algoritmos adequados para executar tais funções".

Existem diversas abordagens e classificações para as técnicas e métodos usados no processo de *Data Mining* conforme se pode observar. Em meio a tantos conceitos dados ao tema numa área ainda pouco explorada, Rodrigues (2000) apresentou uma proposta de classificação para ferramentas de extração de informações, conhecidas como *Data Mining*. Nesse contexto, essas técnicas e métodos foram apresentados sob o ponto de vista do usuário, com uma classificação em Dimensões conforme se pode observar nos Figuras 3 e 4.

2.1.1 DIMENSÃO 1: TIPOS DE PROBLEMAS.

A primeira dimensão para classificação é usada para determinação dos tipos de problemas envolvidos no universo de *Data Mining*. Foram determinamos sete tipos gerais de problemas, conforme apresentado no Figura 1.



Fonte: Rodrigues (2000, p.23)

Figura 3 - Dimensão 1 de classificação: Tipos de Problemas

2.1.1.1 Classificação.

Rodrigues (2000) afirma que a classificação é o tipo de problema mais comum em *Data Mining*. A classificação consiste em examinar as características de um objeto e associar essas características a classes pré-determinadas. A classificação pode ser do tipo simples ou múltipla. A classificação simples consiste na identificação de uma característica binária. Ou seja, a determinação da existência ou não de uma determinada característica. A classificação múltipla consiste em identificar a classe de um determinado objeto. O objetivo é a construção de um modelo que seja capaz de gerar classificações de novos objetos ou novos dados. Dessa forma, esse tipo de problema é caracterizado por uma definição detalhada das classes, possuindo um conjunto de dados para treinamento com exemplos pré-classificados.

Um exemplo bastante representativo desse tipo de problema está em sua utilização em campanhas de mala direta. De posse de uma base de dados com características dos clientes e sua resposta para determinado tipo de campanha promocional, duas classes são definidas: aqueles que responderão positivamente à campanha, adquirindo o produto ou serviço, e aqueles que responderão negativamente. Um modelo é desenvolvido a partir dos dados históricos classificando os clientes nos dois grupos: os que responderão positivamente e os que responderão negativamente. Em seguida, esse modelo pode ser utilizado para classificar novos clientes quanto à sua potencialidade de compra de novos produtos ou serviços, ou para melhor direcionar o envio de material promocional (seria enviado apenas para aqueles do grupo com potencialidade positiva de resposta) (RODRIGUES, 2000, p.23-24).

Rodrigues (2000) cita ainda outros exemplos desse tipo de problema:

- Definição de palavras-chave em artigos para publicação acadêmica;
- Análise de risco de crédito para empréstimos;
- Detecção de fraudes;
- Previsão de falências.

Segundo Batista (2001, p. 2), Classificação é a tarefa de aprendizado de uma função que mapeie um item de dados dentro de uma das várias classes pré-definidas. Associa ou Classifica um registro em classes predefinidas, determinando com que grupo de entidades já classificadas determinado dado apresenta mais semelhança.

2.1.1.2 Estimação.

Segundo Rodrigues (2000), enquanto o problema de classificação lida com valores discretos (sim ou não, classes, etc.), o problema de estimação consiste na determinação de valores contínuos. Através de dados de entrada, utilizamos a estimação para gerar o valor de alguma variável contínua, como a renda, altura, ou saldo de cartão de crédito de um indivíduo.

Na prática, a estimação é utilizada para conduzir uma tarefa de classificação. Uma empresa de cartão de crédito, por exemplo, interessada em anunciar uma promoção de equipamentos de *ski* deseja construir um modelo que classifique seus clientes em dois conjuntos: praticante de *ski* e não praticante de *ski*. Uma alternativa a essa abordagem é a construção de um modelo que associe a cada cliente um valor (entre 0 e 1) para a probabilidade de uso de equipamentos de *ski*. Dessa forma, estaríamos estimando a

probabilidade do cliente praticar o esporte. A grande vantagem de obter modelos de estimação está na possibilidade de ordenar os dados. A empresa poderia ter 1.5 milhões de clientes, mas por questões de custo só poderia enviar 500.000 correspondências. Ordenando os registros de acordo com a variável estimada de probabilidade da prática de *ski*, poderíamos aumentar a taxa de resposta ao material promocional (RODRIGUES, 2000, p.24-25).

Como alguns exemplos desse tipo de problema citados por Rodrigues (2000):

- Estimação do número de crianças em uma família;
- Estimação da renda total de uma família;
- Estimação do valor do cliente para a empresa;
- Estimação da probabilidade de resposta para mala direta.

2.1.1.3 Previsão.

Segundo Rodrigues (2000), Previsão consiste em utilizar um grande número de observações passadas de uma ou mais séries temporais para descobrir valores futuros dessas séries. Técnicas utilizadas nos problemas de classificação e estimação podem ser adaptadas para o uso de previsões.

Como exemplo:

- previsão de vendas;
- previsão de níveis de gastos;
- previsão de fenômenos físicos e naturais.

2.1.1.4 Afinidade de Grupos.

Segundo Rodrigues (2000), o objetivo desse tipo de problema é a identificação de relações diretas entre objetos. O exemplo mais representativo desse tipo de problema é a identificação de padrões de compra em supermercado (*market basket analysis*). Através de identificação de padrões de compras, estratégias de planificação de produtos em prateleiras e mostruários podem ser otimizadas. Esse tipo de problema pode identificar produtos cuja venda está relacionada e indicar potenciais para aumento de venda. A afinidade de grupos pode ser abordada através da utilização de regras de associação.

Como exemplo, se pode citar:

- Vendas associadas entre diferentes produtos;

- Serviços associados a diferentes clientes.

2.1.1.5 Segmentação.

Segundo Rodrigues (2000), Segmentação é a criação de uma divisão em um conjunto de dados no qual todos os membros de cada subconjunto são semelhantes de acordo com alguma medida. O que diferencia os segmentos (*clusters*) de uma classificação é o fato de não necessitar de classes pré-definidas. Na classificação, a população é subdividida e colocada em alguma das classes definidas enquanto que em um problema de segmentação, os registros são agrupados com base em similaridades, e cabe ao analista validar os significados e representatividade dos agrupamentos.

A segmentação de acordo com alguma similaridade aparece em exemplos não só nas ciências sociais e de marketing, mas também em ciências físicas. Geralmente acontece nas etapas preliminares de um projeto de Descoberta do Conhecimento, com o objetivo de segmentar o estudo em grupos de dados semelhantes (RODRIGUES, 2000, p. 26).

Como exemplos, Rodrigues (2000) cita:

- Determinação do número de região de vendas;
- Determinação de grupos de consumidores potenciais.

2.1.1.6 Descrição.

Rodrigues (2000) cita que algumas vezes, o objetivo de um projeto de *Data Mining* é simplesmente descrever o que está acontecendo em um banco de dados para melhor entender as relações entre pessoas, produtos ou processos. Uma boa descrição do comportamento dos dados geralmente sugere uma explicação, ou no mínimo, onde iniciar a busca por explicação.

Um exemplo desse tipo de problema poderia vir de uma análise do perfil eleitoral de um partido político: "mulheres apóiam o partido X em um número maior que homens". Essa descrição pode provocar grande interesse e necessidade de análises detalhadas por parte de jornalistas, economistas, cientistas políticos, além dos próprios partidos (RODRIGUES, 2000, p.27).

2.1.1.7 Reconstrução de funções.

Consiste em reconstruir uma função, através de um número de valores conhecidos dessa mesma função, pelo menos em uma dada região de seu domínio.

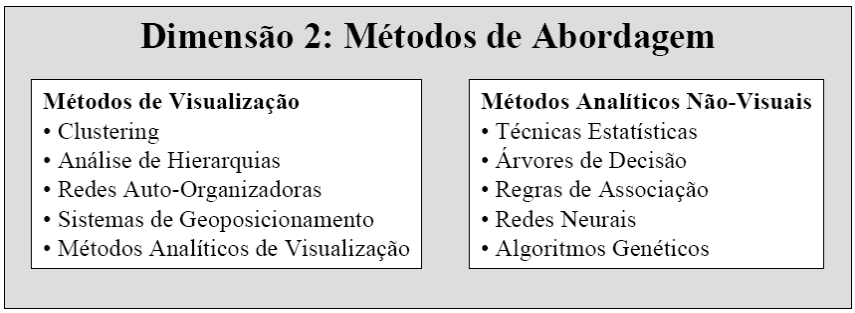
Problemas desse tipo aparecem:

- Computação de superfícies de demanda;
- Cálculo da elasticidade de preço;
- Processos de ajuste de máquinas;
- Interpretação de experimentos físicos.

2.1.2 DIMENSÃO 2 : MÉTODOS DE ABORDAGEM.

Na Figura 4 são apresentados os principais métodos de abordagem de *Data Mining*, constituindo a segunda dimensão de classificação segundo Rodrigues (2000). Entende-se por método de abordagem, as técnicas gerais que são utilizadas no processo de Descobrimento do Conhecimento.

Segundo Rodrigues (2000), esses métodos originaram-se de diferentes áreas de conhecimento e se subdividem em dois grandes conjuntos: Métodos de Visualização e Métodos Analíticos Não-Visuais. Embora nem todos autores usem essa subdivisão, todos concordam com a relação aos métodos descritos em cada grupo.



Fonte: Rodrigues (2000, p.28)

Figura 4 - Dimensão 2 de classificação: Métodos de Abordagem

2.1.2.1 MÉTODOS DE VISUALIZAÇÃO.

As técnicas de visualização são métodos úteis para a descoberta de padrões em um conjunto de dados. Geralmente, a análise de dados é feita utilizando métodos analíticos não-visuais, como veremos em seguida. Entretanto, os métodos analíticos requerem que uma hipótese deva ser criada e que exista um conhecimento anterior do conjunto de dados. Os

métodos de visualização, por outro lado, permitem a descoberta de padrões gerais no banco de dados, ao mesmo tempo em que possibilita a definição de pequenos padrões que podem ser de extrema importância. As técnicas de visualização têm se desenvolvido rapidamente: técnicas gráficas avançadas em realidade virtual possibilitam o tratamento de dados em espaços artificiais, enquanto que o desenvolvimento histórico de um conjunto de dados pode ser acompanhado através de uma animação ou filme (RODRIGUES 2000, p.28-29).

Segundo Rodrigues (2000), os métodos de visualização e analíticos normalmente são usados em conjunto combinando diferentes métodos durante um processo de Descoberta do Conhecimento, sendo os de visualização os primeiros, como forma de exploração inicial do grande conjunto de dados.

Rodrigues (2000) cita que os benefícios utilização dos métodos de visualização são:

- Observações podem ser feitas inexistindo hipóteses ou concepções anteriores sobre as informações contidas no banco de dados;
- Padrões de ultrapassagem de limites, freqüências de ocorrências, e interdependências entre dados são rapidamente apontados.

No entanto uma das grandes desvantagens está no fato de que os métodos de visualização possuem contribuição limitada quando o número de dimensões dos dados é elevado.

Segundo Rodrigues (2000), a Visualização é uma abordagem interativa, onde diferentes cenários devem ser testados e observados, sendo analisados em seguida e para que os dados possam ser analisados, primeiramente devem ser modelados de um modo posicional, geralmente em 2 ou 3 coordenadas. Dessa forma, traduzindo os dados através de atributos em um modelo posicional, podem ser processados utilizando métodos como *clustering*, hierarquias, superfícies e outros formatos.

2.1.2.1.1 Agrupamento (*Clustering*).

Segundo Batista (2001) e Rodrigues (2000), Agrupamento (*Clustering*) é o processo de se agrupar um conjunto de itens (sem um atributo de classe pré-definido), baseado no princípio de semelhança entre valores. Os dados são colocados no gráfico baseados em valores de algum de seus atributos em uma das coordenadas em observação (x, y ou z). Tipicamente, quando esses valores representam um conjunto arbitrário de descrições (ex.:

nomes, doenças, números de identificação, tipos de conta) aparecerão concentrações de dados com valores em comum.

2.1.2.1.2 Análise de Hierarquias.

Quando a posição de um objeto é baseada na relação com outros objetos, podem ser geradas hierarquias na visualização. Essa abordagem é utilizada principalmente quando o conjunto de dados possui tipos e classes de objetos que estão ligados uns aos outros. A parte superior é chamada de nó raiz e sua seleção determina como uma hierarquia vai ser construída para os propósitos da visualização. Como cada nível está conectado ao nível anterior, não existe ligações entre objetos do mesmo nível ou relações assumindo ligações bidirecionais. Dependendo do tipo de aplicação, podemos ter mais de um nó raiz (RODRIGUES, 2000, p. 31).

2.1.2.1.3 Métodos Analíticos de Visualização.

Segundo Rodrigues (2000), muitas inferências podem ser feitas através da simples análise da configuração de dados. Padrões não usuais podem ser rapidamente detectados. A análise desse grupo geralmente é representada na forma de gráficos do tipo *scatter plots* e demonstram variáveis que estão fora dos limites aceitáveis, identificando padrões anômalos ou erros de coleta de dados. Algumas vezes o objetivo da análise é a descoberta de registros que deveriam existir no conjunto de dados. Isso ocorre principalmente em dados que estão organizados em par (como chamadas telefônicas, onde deve existir uma origem e um destino). Qualquer dado perdido consiste em uma falha no sistema em estudo.

2.1.2.2 MÉTODOS ANALÍTICOS NÃO-VISUAIS.

2.1.2.2.1 Técnicas estatísticas

O uso da estatística descritiva e inferencial é um dos mais difundidos em projetos de *Data Mining*. A maioria dos produtos comerciais, como veremos em seguida, utiliza técnicas estatísticas de uma forma direta ou indireta.

A utilização de técnicas estatísticas requer o uso de dados quantitativos. Dados qualitativos devem ser traduzidos em valores numéricos. Os testes estatísticos podem ser usados para a comparação de valores de amostras de grupos diferentes. Nos casos mais

simples, a estatística descritiva é utilizada para obter uma visão geral das características dos dados analisados.

A estatística descritiva inclui medidas como a média (valor médio), mediana (valor que divide a distribuição pela metade), moda (valor mais comum), desvio-padrão (medida de variância), intervalos de confiança (maior e menor valores) e distribuição dos dados.

Dois grupos de testes estatísticos merecem destaque e estão sendo amplamente utilizados em etapas iniciais de projetos de *Data Mining*: Análise de Diferenças de Grupos e Análise de Previsão utilizando Regressão. (RODRIGUES, 2000, p.34).

2.1.2.2.2 Modelagem de Dependência (*Dependency Modeling*).

Segundo Batista (2001, p. 2), “objetivo é desenvolver um modelo para representar dependências significantes ou relações entre as variáveis nos dados; normalmente, são especificadas intensidades de dependência em uma escala numérica e devem satisfazer um definido Suporte e Confiança.”

2.1.2.2.3 Análise Seqüencial (*Sequence Analysis*).

Segundo Batista (2001, p. 2), “esta técnica de descoberta de padrão tenta identificar padrões semelhantes que estão repetidos em intervalos de tempo ordenados.”

2.1.2.2.4 Árvores de decisão (*Decision Trees*).

Segundo Batista (2001, p. 2), Árvores de Decisão “representa uma classificação da população em segmentos diferentes, baseado nos valores dos atributos pertinentes. A estrutura da árvore de decisão explicitamente representa o raciocínio da tomada de decisão.”

Segundo Rodrigues (2000), as árvores de decisão podem ser utilizadas para a classificação de determinados tipos de dados. Seu nome é originado pela característica de sua estrutura, formada por ramos e nós. Em cada nó, uma decisão é tomada ou a ocorrência de uma probabilidade é avaliada. Dependendo da direção tomada, um outro ramo será tomado levando ao próximo nó de decisão.

2.1.2.2.5 Vizinho mais Próximo (*nearest neighbor*)

Batista (2001, p. 2), afirma que o elemento fundamental desses algoritmos é a disponibilidade de uma medida de semelhança que é capaz de identificar os vizinhos de um ponto particular em um espaço métrico.

2.1.2.2.6 Redes Neurais (*Neural Network*).

Segundo Batista (2001, p. 2), estes algoritmos criam e treinam uma rede neural, que é uma rede com unidades de processamento conectadas por pesos adaptáveis. Definindo os campos de entrada e saída dos dados em treinamento, a rede de neural cria modelos de previsão que "aprendem" a classificar os valores dos campos de saída pelos valores dos campos de entrada.

A origem da teoria de Redes Neurais remonta aos modelos matemáticos e aos modelos de engenharia, de neurônios biológicos. Baseiam-se na representação dos impulsos nervosos ou potenciais de ação dos neurônios biológicos. O processo de transmissão de um impulso elétrico através de um neurônio se dá basicamente por um potencial de ação que se propaga através do axônio quando uma despolarização da membrana do neurônio cruza um valor conhecido como liminar de disparo. É possível estabelecer um modelo matemático da freqüência média de impulsos nervosos em um intervalo de tempo T. Tal relação é conhecida como função de ativação do neurônio. Tal função geralmente é representada por uma reta, uma função degrau, uma função sigmóide (*Logit*) ou a tangente hiperbólica (RODRIGUES, 2000, p. 41).

2.1.2.2.7 Algoritmos Genéticos

A utilização de algoritmos genéticos está associada à otimização. Através de analogias desenvolvidas com a teoria da evolução, os algoritmos genéticos iniciam com uma população de itens e procuram alterar e eventualmente otimizar sua composição para solucionar determinado problema. Uma definição formal é que um algoritmo genético é um tipo de algoritmo de computação que modela o processo biológico genético, incluindo as trocas cruzadas (*crossover*) e operações de mutação (RODRIGUES, 2000, p. 44-46).

De acordo com Rodrigues (2000), os algoritmos genéticos são assim chamados pelo processo de seleção implementado que é análogo ao processo de seleção natural que ocorre na

evolução. A seleção é baseada no princípio de sobrevivência daquele que melhor se adapta ao ambiente.

Algoritmos genéticos devem ser usados quando o principal objetivo é à busca de soluções ótimas, dado determinado problema. Além disso, deve existir uma grande uniformidade no banco de dados, já que os itens a serem analisados devem ser codificados em vetores de mesma dimensão. Por essa razão, é improvável sua utilização em situações onde os dados foram combinados de diversas fontes e tipos de informação. Sua maior utilização está em áreas como maximização do lucro através de combinações de *mix* de produtos. Além disso, tem sido utilizado em aplicações que envolvem organização de cronograma e análise de séries temporais. Sua combinação com redes neurais é amplamente utilizada, gerando diminuições em tempos de processamento (RODRIGUES, 2000, p. 44-46).

2.1.2.2.8 Regras de Associação (*Association Rules*).

Segundo Batista (2001, p. 2), Regras de Associação descobre relacionamentos entre conjuntos de itens. Geralmente, estes algoritmos calculam o nível de Suporte (a porcentagem de todos os registros onde eventos múltiplos aconteceram), e então calcula o nível de Confiança (a porcentagem de transações onde um evento aconteceu e o outro evento também aconteceu).

Conforme Miranda et al. (2003), as Regras de Associação representam padrões onde a ocorrência de eventos em conjunto é alta. Pode-se dizer que é a probabilidade de que um conjunto de itens apareça em uma transação, dado que outro conjunto esteja presente. Um exemplo de tal tipo de regra seria "75% dos consumidores que compram o produto A e B também compram o produto C".

Quanto ao tipo de descoberta, as regras de Associação são consideradas um método não supervisionado, e obtém o segundo lugar em percentual de utilização em aplicações, somente atrás de Classificação. O objetivo de minerar regras de Associação, é encontrar todos os conjuntos de itens que freqüentemente ocorrem de forma conjunta na base de dados e formar regras a partir destes conjuntos (MIRANDA et al., 2003).

Segundo Miranda et al. (2003), As Regras de Associação são representadas da seguinte forma: X => Y (lê-se X implica em Y) , onde X é o antecessor e Y o conseqüente e X e Y são dois *itemsets* distintos na Base de Dados. Cada regra da forma X => Y possui dois atributos

que determinam sua validade no conjunto de dados e também limitam a quantidade de regras extraídas. São eles o suporte e a confiança. Estes possibilitam o descarte das regras julgadas de pouco interesse, já que são menos freqüentes e confiáveis. A função do Suporte é determinar a freqüência que ocorre um *itemset* dentre todas as transações da Base de Dados, é a porcentagem de transações onde este *itemset* aparece. Um *itemset* será considerado freqüente se o seu suporte for maior ou igual a um suporte mínimo estabelecido previamente.

$$\text{Suporte } (X \cup Y) = \frac{\text{Nº de registros com } (X \cup Y)}{\text{Nº Total de transações da BD}}$$

A toda regra X => Y associamos um grau de confiança. Ela é a medida da força da regra e determina a sua validade. A probabilidade condicional de se encontrar Y, já tendo encontrado X é dado pela confiança. Assim como o suporte, também é estabelecido um nível mínimo de confiança para as regras (MIRANDA et al., 2003).

$$\text{Conf } (X \Rightarrow Y) = \frac{\text{Nº de transações que suportam } (X \cup Y)}{\text{Nº de transações que suportam } (X)}$$

Miranda et al. (2003), afirma que a confiança é para a regra, e o grau de confiança não é suficiente para que se considere uma regra como válida. Para garantir que uma regra X => Y seja válida, é necessário que o suporte também seja alto. Portanto, uma regra é interessante se Sup(R) = i e Conf(R) = j, sendo i e j um grau mínimo de suporte e confiança. Este é um grande problema da mineração de regras de associação, descobrir regras que realmente sejam úteis e possuam padrões interessantes. Regras com padrões óbvios, não são interessantes e devem ser descartadas, daí a importância de se estipular um nível mínimo para o suporte e para a confiança. Estes atributos são melhor exemplificados logo abaixo na execução do Algoritmo Apriori.

2.1.2.2.9 Formalização do Problema:

Itemset = É um conjunto de ítens que ocorrem juntamente em uma transação.

k-*itemsets* = *Itemset* com k elementos.

Dado um conjunto de itens I={ i1, i2, i3, ... in} e um conjunto de Transações T em uma Base de Dados D, onde cada transação é um conjunto de itens em I tal que T está contido ou é igual a I. Sendo X um conjunto de itens em I, uma transação T contém X, se X estiver contido ou for igual a T. Uma regra de associação é uma implicação do tipo X => Y, onde X está contido em I, Y está contido em I e X interseção com Y = vazio.A regra X => Y é válida no conjunto de transações D com o grau de confiança c, se c% das transações em D que contêm X também contêm Y. A regra X => Y tem suporte s , se s% das transações em D contêm X U Y. (MIRANDA et al., 2003).

Segundo Miranda et al. (2003), o processo de descobrir todas as Regras de Associação, pode ser decomposto em duas etapas:

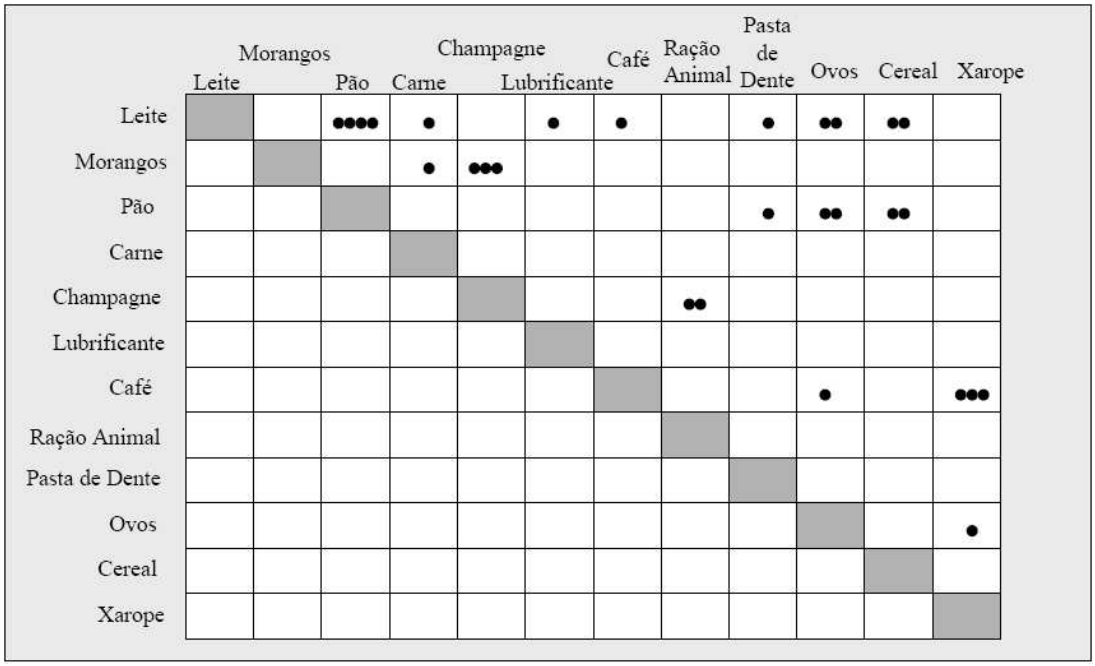
1) Encontrar todos os conjuntos de itens freqüentes, pois exige sucessivas buscas no BD e é responsável pela maior parte do processamento;

2) Utilizar os *itemsets* freqüentes obtidos para gerar as regras de associação.

Existem vários métodos e algoritmos que possibilitam encontrar regras de Associação em grandes bases de dados, entre eles estão os algoritmos PARTITION, DIC , DHP, DLG, e o mais comum é a família do Algoritmo APRIORI e suas variações, como o APIORIO TID e o APRIORI HYBRID. Neste trabalho está sendo abordado somente o algoritmo APRIORI, visando detalhar seu funcionamento e suas funções.

Rodrigues (2000) afirma que as Regras de Associação são derivadas de um tipo de análise que extrai informação de padrões que se repetem ou coincidências no banco de dados. Muitas vezes, esse tipo de análise é chamado de *market basket analysis*. O exemplo clássico desse tipo de análise é a determinação de padrões de consumo em cadeias de supermercados, por exemplo.

Segundo Rodrigues (2000), as Regras de Associação são obtidas através de matriz de inter-relação, onde a probabilidade do acontecimento conjunto de cada evento é calculada. Podemos exemplificar através de um exemplo de padrões de consumo em uma loja de conveniências. No exemplo em questão, como podemos observar na Figura 5, considera-se a análise de 12 produtos. No entanto, a análise poderia ser estendida para n produtos ou dimensões.



Fonte: Rodrigues (2000, p.39)

Figura 5 - Exemplo de Inter-relação para Regras de Associação.

A matriz nos mostra produtos que são muitas vezes comprados em conjunto (células possuindo mais de um círculo), produtos que algumas vezes são comprados em conjunto (células que possuem apenas um círculo) e aqueles pares que não possuem relações em padrões de compra (células em branco). Através da matriz podemos notar que existe uma forte relação entre a compra de pão e de leite. O mesmo acontece com relação a cereais e leite. São relações que inicialmente são intuitivas e que, a princípio, não trazem muito esclarecimento na análise. No entanto, ao utilizar essa abordagem, notamos relações não tão diretas, quanto à correlação entre morangos e *champagne* e entre ração animal e *champagne*. O grande benefício dessa análise está na descoberta desses padrões não intuitivos e sua posterior interpretação (RODRIGUES, 2000, p.39).

Por outro lado, o uso de uma matriz de inter-relação possui suas limitações. Embora a relação entre *champagne* e morango possa ser compreendida, a relação entre *champagne* e ração animal parece ser de pouco uso. Suponha que as pessoas que possuam um animal de estimação comprem ração semanalmente. No entanto, a pessoa não consome ração em conjunto com *champagne*. No entanto, a regra de associação relaciona os dois produtos, já que são comprados em conjunto regularmente. A extrapolação dessas regras para padrões gerais de consumo pode não ser uma boa estratégia de marketing. Por exemplo, não é intuitivo que todas as pessoas que consomem *champagne* estejam interessadas em consumir

ração animal. Logo, torna-se papel do analista determinar quais são as "boas" regras de associação e as regras "sem uso" (RODRIGUES, 2000, p.40).

Segundo Rodrigues (2000), as Regras de Associação possuem grande aplicação em processos de análise exploratória de dados, em busca de relações interessantes que possam existir no conjunto de dados. Para as regras identificadas como úteis, podemos utilizá-las para prever padrões de consumo e atuar com estratégias de marketing. No entanto, o fato de detectarmos eventos que ocorrem em conjunto não necessariamente indica que essa relação é significativa ou possa ser generalizada. Logo, as regras não intuitivas devem ser cuidadosamente estudadas utilizando algum outro método.

Witten & Frank (2000, p. 63), afirmam que as Regras de Associação não são realmente diferentes de Regras de Classificação exceto que elas podem predizer algum atributo, não só a classe, e isso lhes dão a liberdade para predizer combinações de atributos também. Regras de Associação também não tem a intenção se serem usadas junto com um conjunto, como as regras de Classificação são. Diferentes Regras de Associação expressam diferentes regularidades que estão por baixo de um *dataset*, e elas geralmente predizem coisas diferentes. Isso porque tantas Regras de Associação diferentes podem ser derivadas de até mesmo um minúsculo *dataset*, cujo interesse está restrito a esses que aplicam a um número razoavelmente grande de instâncias e tem uma precisão (*accuracy*) razoavelmente alta nas instâncias que eles aplicam. A cobertura (*coverage*) de uma Regra de Associação é o número de instâncias para o qual ela prediz corretamente – isto é freqüentemente chamado de suporte. E sua precisão (*accuracy*) – freqüentemente chamada de confiança (*confidence*) – é o número de instâncias que ela prediz corretamente, expresso como uma proporção de todas as instâncias que ela foi aplicada. Por exemplo, com a regra:

Se temperatura = frio então umidade = normal

A cobertura é o número de dias que são ambos frio e umidade = normal (4 como mostra a Tabela 2), e a precisão (*accuracy*) é a proporção de dias frios que tem umidade normal (100% nesse caso). Isso é habitual para especificar os valores mínimos de cobertura e precisão e para procurar somente aquelas regras cuja cobertura e precisão são pelo menos os mínimos especificados. No exemplo do Tempo, há 58 regras em que a cobertura e precisão são pelo menos 2 e 95% respectivamente. (Ao invés disso pode ser conveniente também

especificar a cobertura como uma percentagem do número total de instâncias) (WITTEN & FRANK, 2000, p.63).

Witten & Frank (2000, p. 63), afirmam que Regras de Associação que predizem múltiplas conseqüências devem ser interpretadas cuidadosamente. No exemplo do Tempo, mostrado na Tabela 2 pode-se ver a seguinte regra:

Se vento = falso e jogar = não então previsão = ensolarado e umidade = alta

Esta não é somente uma expressão de taquigrafia para as duas regras separadas:

Se vento = falso e jogar = não então previsão = ensolarado Se vento = falso e jogar = não então umidade = alta

Isto realmente insinua que estas regras excedem a cobertura e precisões mínimas estimadas – mas insinua mais. A regra original significa que o número de exemplos que são não-venta, não-joga e com previsão ensolarada e umidade alta é pelo menos tão grande quanto o especificado na cobertura mínima estimada. E isto também significa que o número de tais dias, expressos por uma proporção de dias que não-venta, não-joga é pelo menos o especificado na precisão mínima estimada. Isto implica que a regra

Se umidade = alta e vento = falso e jogar = não então previsão = ensolarado

também se sustenta, pois ela tem a mesma cobertura que a regra original, e sua precisão seja pelo menos tão alta quanto as regras originais, porque o número de dias que a umidade é alta, não-venta, não-joga é necessariamente menor que aqueles dias que não-venta, não-joga – que torna e precisão maior.

Como se pode observar, há relacionamento entre Regras de Associação particulares: algumas regras implicam em outras. Para reduzir o número de regras que são produzidas, nos casos onde várias regras estão relacionadas faz sentido apresentar somente as regras mais fortes ao usuário. No exemplo acima, somente a primeira regra deveria ser mostrada.

Previsão do Tempo X Jogar Tênis

Previsão	Temperatura	Umidade	Vento	Jogar

Ensolarado	Quente	Alta	Falso	Não
Ensolarado	Quente	Alta	Verdadeiro	Não
Nublado	Quente	Alta	Falso	Sim
Chuvoso	Moderado	Alta	Falso	Sim
Chuvoso	Frio	Normal	Falso	Sim
Chuvoso	Frio	Normal	Verdadeiro	Não
Nublado	Frio	Normal	Verdadeiro	Sim
Ensolarado	Moderado	Alta	Falso	Não
Ensolarado	Frio	Normal	Falso	Sim
Chuvoso	Moderado	Normal	Verdadeiro	Sim
Ensolarado	Moderado	Normal	Verdadeiro	Sim
Nublado	Moderado	Alta	Verdadeiro	Sim
Nublado	Quente	Normal	Falso	Sim
Chuvoso	Moderado	Alta	Verdadeiro	Não

Fonte: Witten & Frank (2000)

Tabela 2 - *Dataset* com a previsão do tempo versus jogar tênis

Segundo Witten & Frank (2000, p. 63), Regras de Associação são como Regras de Classificação. E é possível encontrá-las seguindo o mesmo caminho, executando o procedimento chamado de regra de indução *separate-and-quonquer* para cada expressão possível do lado direito da regra (o lado que precede o símbolo de => ou a palavra então). Entretanto, um atributo pode ocorrer no lado direito com um possível valor, e uma regra de associação simples pode predizer o valor em mais de um atributo. Para encontrar tais regras, deve-se executar o procedimento de regra de indução uma vez para cada combinação possível dos atributos com todas as possibilidades de combinações possíveis de valores, do lado direito da regra. Isto deve resultar em um número enorme de Regras de Associação, que deve então ser podado com base na cobertura (número de instâncias que eles predizem corretamente) – e precisão (o mesmo número expresso como uma proporção do número de instâncias que a regra se aplica). Vale lembrar que a cobertura (*coverage*) é freqüentemente chamada de suporte (*support*) e a precisão (*accuracy*) é frequentemente chamada de confiança (*confidence*).

A primeira coluna da Tabela 3 mostra os itens individualmente para os dados da Previsão do tempo versus Jogar Tênis mostrados na tabela 2, com o número de vezes que cada item aparece no *dataset* pelo lado direito da regra. Esses são os conjuntos de 1 (um) item. O

próximo passo é gerar os conjuntos de 2 (dois) itens levando pares de conjuntos de 1 (um) item. Evidentemente sem gerar conjuntos com os dois itens do mesmo atributo (tal como previsão = ensolarado e previsão = nublado) (WITTEN & FRANK, 2000, p.105).

No exemplo mostrado na tabela 2 assume-se que foram procuradas regras com cobertura de 2 (dois): isto descarta qualquer conjunto de item que a cobertura for menor do que 2 (duas) instâncias. Isto nos leva a um total de 47 *itensets* de dois itens, alguns que são mostrados na segunda coluna juntamente com o número de vezes que eles aparecem. O próximo passo é gerar os *itensets* com 3 itens, dos quais 39 tiveram cobertura (*coverage*) de dois ou mais. Há 6 *itensets* com 4 itens e não há com 5 itens – para esses dados, os conjuntos com 5 (cinco) itens com cobertura de 2 ou mais correspondem apenas a instâncias repetidas. A primeira linha da tabela 3, por exemplo, mostra que há 5 dias onde a previsão = ensolarado, dois dos quais temos temperatura = moderado e, de fato, em ambos esses dias a umidade = alta e jogar = não (WITTEN & FRANK, 2000, p.105).

Itensets - Previsão do Tempo X Jogar Tênis

	Itensets de um item	*Itensets* de dois itens	*Itensets* de três itens	*Itensets* de quatro itens
1	Previsão = Ensolarado (5)	Previsão = Ensolarado Temperatura = Moderado (2)	Previsão = Ensolarado Temperatura = Quente Umidade = Alta (2)	Previsão = Ensolarado Temperatura = Quente Umidade = Alta Jogar = Não (2)
2	Previsão = Nublado (4)	Previsão = Ensolarado Temperatura = Quente (2)	Previsão = Ensolarado Temperatura = Quente Jogar = Não (2)	Previsão = Ensolarado Umidade = Alta Vento = Falso Jogar = Não (2)
3	Previsão = Chuvoso (5)	Previsão = Ensolarado Umidade = Normal (2)	Previsão = Ensolarado Umidade = Normal Jogar = Sim (2)	Previsão = Nublado Temperatura = Quente Vento = Falso Jogar = Sim (2)
4	Temperatura = Frio (4)	Previsão = Ensolarado Umidade = Alta (3)	Previsão = Ensolarado Umidade = Alta Vento = Falso (2)	Previsão = Chuvoso Umidade = Moderado Vento = Falso Jogar = Sim (2)
5	Temperatura = Moderado (6)	Previsão = Ensolarado Vento = Verdadeiro (2)	Previsão = Ensolarado Umidade = Alta Jogar = Não (3)	Previsão = Chuvoso Umidade = Normal Vento = Falso Jogar = Sim (2)

Fonte: Witten & Frank (2000, p.106-107)

Tabela 3 - *Itensets* para os dados do Tempo com cobertura de dois ou mais

6	Temperatura = Quente (4)	Previsão = Ensolarado Vento = Falso (3)	Previsão = Ensolarado Vento = Falso Jogar = Não (2)	Temperatura = Frio Umidade = Normal Vento = Falso Jogar = Sim (2)
7	Umidade = Normal (7)	Previsão = Ensolarado Jogar = Sim (2)	Previsão = Nublado Temperatura = Quente Vento = Falso (2)	
8	Umidade = Alta (7)	Previsão = Ensolarado Jogar = Não (3)	Previsão = Nublado Umidade = Normal Jogar = Sim (2)	
9	Vento = Verdadeiro (6)	Previsão = Nublado Temperatura = Quente (2)	Previsão = Nublado Umidade = Alta Jogar = Sim (2)	
10	Vento = Falso (8)	Previsão = Nublado Umidade = Normal (2)	Previsão = Nublado Vento = Verdadeiro Jogar = Sim (2)	
11	Jogar = Sim (9)	Previsão = Nublado Umidade = Alta (2)	Previsão = Nublado Vento = Verdadeiro Jogar = Sim (2)	
12	Jogar = Não (5)	Previsão = Nublado Umidade = Verdadeiro (2)	Previsão = Nublado Vento = Falso Jogar = Sim (2)	
13		Previsão = Nublado Vento = Falso (2)	Previsão = Chuvoso Temperatura = Frio Umidade = Normal (2)	
...	...	...	...	
38		Umidade = Normal Vento = Falso (4)	Umidade = Normal Vento = Falso Jogar = Sim (4)	
39		Umidade = Normal Jogar = Sim (6)	Umidade = Alta Vento = Falso Jogar = Não (2)	
40		Umidade = Alta Vento = Verdadeiro (3)		
...	...	...	...	
47		Vento = Falso Jogar = Não (2)		

Tabela 3 - *Itensets* para os dados do Tempo com cobertura de dois ou mais. (Continuação).

2.1.3 O ALGORITMO APRIORI.

Miranda et al. (2003) afirma que o algoritmo APRIORI é considerado um clássico na extração de Regras de Associação. Ele foi proposto pela equipe de pesquisa QUEST da IBM que deu origem ao *Software Intelligent Miner*.

Esse algoritmo faz recursivas buscas no Banco de Dados à procura dos conjuntos freqüentes (conjuntos que satisfazem um suporte mínimo estabelecido).Possui diversas propriedades que otimiza o seu desempenho, como por exemplo, a propriedade de antimonotonia da relação, que diz que para um *itemset* ser freqüente, todo o seu subconjunto também devem ser, além de utilizar recursos da memória principal e estrutura *hash* (MIRANDA et al., 2003).

Três fases compõem o APRIORI. São elas:

1- Geração dos conjuntos Candidatos;

2- Poda dos conjuntos Candidatos;

3- Contagem do Suporte: Nesta fase é necessário visitar o BD.

Segundo Miranda et al. (2003) a propriedade de Antimonotonia da Relação ou Propriedade Apriori, pode ser explicada da seguinte forma:

Se X está contido em Y e X não é freqüente, logo Y também não é freqüente. Isto implica uma diminuição do tempo de execução, pois se X não é freqüente, então não será necessário calcular o suporte de Y, e o BD não precisará ser varrido.

```
L1 = {large 1-itemsets};
for (k=2; Lk-1 ≠ ∅; k++) do begin
   Ck = apriori_gen(Lk-1);  // Novos candidatos
   forall transactions t ∈ D do begin
      Ct = subset(Ck, t);  // Candidatos contidos em t
      forall candidates c ∈ Ct do
         c.count++;
   end
   Lk = {c ∈ Ck | c.count ≥ minsup};
end
Answer = ∪k Lk;
```

Fonte: Miranda et al. (2003).

Figura 6 - Algoritmo APRIORI.

2.1.3.1 Funcionamento do Algoritmo Apriori

Esse algoritmo gera um conjunto de itens freqüentes a cada uma de suas passagens. Com base nestes será gerado um outro conjunto C_k, conjunto de itens candidatos, esse consta os itens do conjunto freqüente (L_k) com minsup maior que o estabelecido. O conjunto

candidato é resultado do conjunto de freqüentes da passagem anterior em união com ele mesmo. Posteriormente o conjunto candidato é podado, seu suporte é contado e os itens que tem suporte acima do estabelecido serão os itens freqüentes da próxima passagem (L_{k+1}).

O Algoritmo Apriori utiliza os itens freqüentes obtidos pelo comando executado em SQL, sendo a primeira passagem k=1. Para k=2, enquanto o conjunto obtido na passagem anterior não for vazio então k será incrementado e o conjunto de candidatos receberá os itens retornados pela função apriori_gen.

A Função Apriori-gen é responsável pela união dos conjuntos freqüentes a fim de formar o conjunto candidato com k itens. Para isso tem os itens freqüentes da passagem anterior como parâmetro. Ela faz também a poda dos candidatos.

Então para todas as transações t contidas no conjunto de transações é adicionado um contador de suporte, verificando assim quais itens do conjunto candidato estão contidos em cada uma das transações.

Para o processo de contagem do suporte dos candidatos, os conjuntos são dispostos em uma árvore *Hash*. Esse é um método de espalhar os elementos de um conjunto seguindo uma dada função (função *hash*) com ela é possível realizar uma busca direta pelo elemento desejado, evitando a principio buscas seqüenciais em todo conjunto, acarretando em um ganho significativo em tempo de execução.

Um nó em uma árvore *hash* ou contém uma lista de conjuntos de itens (nó folha), ou contém uma tabela *hash* (nó interno) essa é usada quando o número máximo de elementos em uma folha excede o limite estabelecido.

Quando um conjunto candidato é adicionado, inicia-se da raiz da árvore até alcançar uma folha, a definição do caminho a ser seguido é dada pela função *hash* calculada para este anteriormente. Inicialmente cada nó é criado como sendo uma folha.

A poda é realizada se algum subconjunto do conjunto candidato não estiver presente no conjunto de itens freqüentes da passagem anterior. Como meio de otimização, a poda dos conjuntos também pode ser feita através de uma árvore *hash*, mas no algoritmo original ela é feita através da função Apriori_Gen.

A Função *Subset* é encarregada de contar o suporte dos itens candidatos. Ela toma como parâmetros o conjunto candidato (C_k) e o conjunto de Transações (T). Primeiramente ela faz as combinações entre os itens da transação da seguinte forma: um item é combinado com todos os outros que estão imediatamente a sua frente e assim até o ultimo elemento. Posteriormente verificam-se quais destes estão presentes na árvore *hash*, da seguinte forma:

É calculada a função *hash* dos subconjuntos obtidos com a transação e então esses são comparados com a árvore, sendo que se presentes na árvore haverá um contador de suporte que incrementará o suporte deste *itemset*.

Feito isso teremos o suporte dos itens candidatos, os itens que possuírem suporte maior que o estabelecido formarão o conjunto de itens freqüentes desta passagem.

O algoritmo Apriori termina quando o conjunto de itens freqüentes da passagem anterior for igual a zero, e retorna como resultado de sua execução a união de todos os itens freqüentes de todas as passagens.

2.1.3.2 Algoritmo Apriori Passo a Passo

Devido à complexidade do algoritmo, nos próximos parágrafos o algoritmo está dividido em 7 passos que facilitam seu entendimento.

2.1.3.2.1 1ª Parte:

L_1 = {large 1-itemset};

INPUT:

ß - Suporte Mínimo

D - Banco de Dados de transações

L_1- Conjunto de itens freqüentes com 1 elemento.

Este conjunto é formado por todos os elementos do Banco de Dados isoladamente que correspondem ao nível mínimo de suporte. Ex.: {{A}; {B}; {C}; {D}; {E}; {F}}

Entende-se que é um grande conjunto composto de vários subconjuntos.

L_1 pode ser obtido através de um comando SQL:

INSERT INTO L1 SELECT pn, *count*(*) FROM D GROUP BY pn *HAVING count*(*) >2

Tendo-se como resultado da *query* na base de dados D:

Nt	pn
T1	A
T1	B
T1	C
T2	B
T2	C
T3	A
T3	B

pn - é a descrição do item na transação, como mostrado na tabela abaixo.

2 - nível mínimo de suporte

count – é um contador de suporte.

O Comando SQL descrito percorre a Base de Dados D e agrupa todos os elementos pn que tem o suporte maior que o suporte mínimo estabelecido (itens freqüentes).

No caso da tabela acima: A=2; B=3 e C=2, então L_1={{A}; {B}; {C}}

2.1.3.2.2 2ª Parte:

for (k=2; L_{k-1} # Ø; k++) do begin

Essa parte do algoritmo inicia um laço de repetição onde logo em seguida, começam a ser gerado os conjuntos candidatos.

K – é um contador e inicia-se com 2. Identifica o número da passagem pelo algoritmo. Também serve como base para identificar o número de elementos do conjunto. Por exemplo, a passagem 2 forma conjuntos candidatos com 2 elementos. K=2 forma o conjunto C2; K=3 forma o conjunto C3. Inicia-se com 2 porque o algoritmo inicia-se com o L1.

L_{k-1} # Ø - Condição de encerramento do laço de repetição. Este encerra quando o conjunto da passagem anterior for um conjunto vazio. É decrescido 1 (um) de k porque quando K=2 o conjunto de entrada será o conjunto inicial onde k=1, e C2 será gerado através de L_1. Quando K=3 o conjunto de entrada será o conjunto da passagem anterior onde k=2, ou seja, C3 será formado pelo conjunto L2, e assim por diante, até que a condição de parada seja verdadeira.

k++ - Incrementa o contador K.

2.1.3.2.3 3ª Parte:

C_k = APRIORI_GEN (L_{k-1});

C_k - Conjunto de itens candidatos com k elementos. K também é o número da passagem no algoritmo.

APRIORI_GEN (L_{k-1}) - Esta função recebe o conjunto L_{k-1} como parâmetro e, a partir dele gera os conjuntos pré-candidatos. Em uma outra parte isolada da função APRIORI_GEN, é realizada a poda dos pré-candidatos, gerando assim o conjunto candidato C de índice K, contendo K elementos, que é a resposta do APRIORI_GEN ao algoritmo principal apriori.

Devido a sua complexidade, a função APRIORI_GEN e o seu funcionamento serão tratados separadamente neste documento. O que foi descrito nesta seção é apenas um resumo do que ela faz e qual a sua importância no algoritmo.

2.1.3.2.4 4ª Parte:

forall transactions t em D do begin

Ct = *subset*(C_k, t);

A função *Subset* recebe o conjunto C_k com os itens candidatos já podados, gerados pela função APRIORI_GEN, e recebe também as transações t da Base de Dados D, uma a uma seqüencialmente.

Esta função cria uma árvore *hash* e armazena todos conjuntos candidatos em um nó folha da árvore, que são apontados pelos nós internos chamados tabelas *hash*. Com a árvore montada, a função joga na árvore a transação correspondente e verifica quais conjuntos candidatos presentes na árvore estão contidos na transação.

A função *subset* insere na árvore cada item da transação, um por um, como sendo um item inicial de um conjunto candidato. Aplica-lhe a função *hash*, para saber qual ramo da árvore deve ser seguido e, logo após calcula a função *hash* do próximo elemento da transação, inserindo-o na árvore (Isso é feito para todos os elementos da transação) formando combinações com o item inicial. Com o uso da função *hash*, é possível ir direto à folha onde o elemento está, fazendo uma busca randômica (direta) na árvore, não sendo necessário percorrer todas as folhas a cada vez que um item da transação é acrescentado à árvore.

Se o conjunto candidato formado estiver presente na folha, a função guarda-o no conjunto resposta C_t. Dessa forma, concluímos que C_t é formado por todos os conjuntos candidatos (*itemsets*) de C_k presentes em uma transação t.

Modelo de árvore *hash*: C2= {{A,B}; {A,E}; {B,E}}

Função *Hash*: A =1; B=2; C=1; D=2; E=1.

2.1.3.2.5 5ª Parte:

forall candidatos c em C_t do

c.count++;

Para todos os subconjuntos c pertencentes ao conjunto C_t (Conjuntos candidatos encontrados na Base de Dados de transações) incrementa-se o seu contador de suporte.

Ex.: C_t= {{A,B}; {A,C}; {D,E}} c_1={A,B} c_2={A,C} c_3={D,E}

Terminada a primeira transação, inicia-se a segunda e assim por diante. Isso é feito para cada transação isoladamente. O Algoritmo percorre toda a Base de Dados atrás dos conjuntos freqüentes.

Obs: A 4ª e 5ª parte do algoritmo são executadas seqüencialmente.

2.1.3.2.6 6ª Parte:

L_k = {c em Ck | c.*count* >= suporte_mínimo};

L_k = Conjuntos freqüentes são armazenados em L_k.

C_k = Conjuntos candidatos com elementos de tamanho k.

L_k recebe todos os conjuntos c pertencentes a C_k cujo suporte seja maior ou igual ao suporte mínimo estabelecido na primeira parte do algoritmo. O conjunto L_k definido neste comando servirá de entrada para a próxima iteração do algoritmo, e formarão novos conjuntos candidatos.

2.1.3.2.7 7ª Parte:

Answer = U_k L_k;

Answer é uma variável que recebe a união de todos os valores de L_k.

L_1 U L_2 U L_3 U L_4 ... U L_n

2.1.4 Geração de Regras

No processo de Geração de Regras não é preciso visitar novamente a Base de Dados, pois a contagem do suporte já foi feita em fases anteriores do Apriori.

forall Lk , k>=2

genrules (l_k, l_k);

O algoritmo Apriori pega todos os conjuntos freqüentes l_k de L_k, onde K>=2 e chama a função genrules passando todos os conjuntos l_k como parâmetro, um de cada vez.

A função genrules recebe o conjunto freqüente L_k por duas vezes, a primeira como l_k e a segunda como a_m, conjunto freqüente com m-itens, onde inicialmente m=k, e l_k = a_m.

Em seguida, encontramos todos os subconjuntos não nulos de a_m com a_m-1 elementos. Estes subconjuntos são armazenados em conjunto chamado de A. Para cada subconjunto am-1 de am pertencente a A, escrevemos a regra da forma a_{m-1} => (l_k - a_{m-1}), se a razão conf=sup(l_k) / sup(a_{m-1}) for maior ou igual a confiança mínima. Considera-se todos os subconjuntos de a_m para gerar as regras com múltiplos tamanhos.

O algoritmo divide o conjunto freqüente am em vários subconjuntos de m-1 elementos em busca das regras válidas.

No próximo passo testa se o valor de m-1 é maior que 1, e se a condição for verdadeira chama novamente a função genrules, num processo recursivo, passando o conjunto l_k e o conjunto a_{m-1}.

Novamente, a função genrules recebe o conjunto l_k e o conjunto a_{m-1} como a_m, e repete todo o processo até que o conjunto a_{m-1} contenha somente um elemento.

O Algoritmo Simples:

```
procedure genrules(lk, am)

A= {(m-1)-itens am-1 | am-1 ⊂ am }      // gera o conjunto A contendo os subconjuntos de am com am-1
elementos
 forall am-1 ∈ A do
   conf=support(lk) / support(am-1)      // calcula a confiança para a regra de am-1
   if (conf ≥ minconf) then
      output am-1 → (lk - am-1), with confidence=conf and support= support (lk)
     if (m-1 >1) then                   //se am-1 tiver mais de um elemento, chama novamente a função
        genrules(lk, am-1);              genrules, passando o conjunto lk e o conjunto am-1 para gerar
     end                                 para gerar novas regras com os subconjuntos de am-1.
  end
end
```

Fonte: Miranda et al. (2003)

Figura 7 – Algoritmo APRIORI.

2.1.5 Função APRIORI_GEN

```
insert into Ck
   select p.item1, p.item2, ..., p.itemk-1, q.itemk-1
   from Lk-1 p, Lk-1 q
   where p.item1 = q.item1 and p.item2 = q.item2 and ...
      and p.itemk-2 = q.itemk-2 and p.itemk-1 < q.itemk-1;

forall itemsets c ∈ Ck do
   forall (k-1)-subsets s of c do
      if (s ∉ Lk-1) then
         delete c from Ck;
```

Fonte: Miranda et al. (2003)

Figura 8 – Função APRIORI_GEN

insert into C_k

C_k é dado à função como uma tabela resposta, ou seja, o resultado da função SQL será lançado em C_k que por sua vez é repassado ao algoritmo Apriori.

select p.item$_1$, p.item$_2$, ..., p.item$_{k-1}$, q.item$_{k-1}$

Garante que serão selecionados somente a coluna 1 da tabela p e a coluna 1 da tabela q, visto que estas são formadas por duas colunas onde a primeira contém os elementos e a segunda o contador de suporte.

from L_{k-1} p, L_{k-1} q

Utiliza L_{k-1} para gerar os conjuntos pré-candidatos. Indica onde estão as colunas selecionadas anteriormente. Faz um produto cartesiano do conjunto freqüente L_{k-1} com ele mesmo: L_{k-1}* L_{k-1}

L_{k-1} = Conjunto freqüente da passagem anterior

where $p.item_1 = q.item_1$ and $p.item_2 = q.item_2$ and ...

and $p.item_{k-2} = q.item_{k-2}$ and $p.item_{k-1} < q.item_{k-1}$;

Garante que os itens do conjunto candidato formado estarão ordenados lexicograficamente, eliminando aqueles que possuírem itens equivalentes. Por exemplo:

AA - será eliminado

AB - não será eliminado

BA - será eliminado

forall itemsets c em C_k do

forall (k-1)-subsets s of c do

if (s não pertencer a L_{k-1}) then

delete c from C_k;

Esta parte da função faz a poda dos conjuntos pré-candidatos. Se tivermos um conjunto C_k com 3 elementos, verifica-se se todos os seus subconjuntos de K-1 itens (subconjuntos com 2 elementos) estão presentes no conjunto de freqüentes da passagem anterior L_{k-1}.

Se por acaso o subconjunto s de c não estiver presente no conjunto de freqüentes da passagem anterior, então c será excluído de C_k. Este passo ocorre devido a propriedade do algoritmo APRIORI que diz que para um conjunto ser freqüente todos os seus subconjuntos também devem ser freqüentes.

3 WEB SERVERS

Um *web server* é um servidor de páginas *web* que pode conter um ou mais *Websites* armazenados em sua estrutura, permitindo e controlando o acesso a essas páginas de forma confiável e rápida. Já um *Website* é apenas um conjunto de páginas ligadas entre si e contendo imagens, sons e textos. A combinação de um *web server* com um *Website*, conectados a internet permite que essas páginas possam ser consultadas em qualquer parte do mundo, a qualquer hora, por qualquer pessoa, através de um serviço tipo *World Wide Web*. O serviço *World Wide Web* pode ser acessado por um *web browser* (navegador *web*).

Segundo Microsoft (2003), pode-se configurar *sites* da *web* ou *File Transfer Protocol* (FTP) para registrar entradas de *log* geradas a partir das atividades do servidor e dos usuários. Os dados do *log* do *Internet Information Services* (IIS) podem ajudar a controlar o acesso ao conteúdo, determinar a popularidade do conteúdo, planejar os requisitos de segurança e solucionar possíveis problemas dos *sites* da *Web* ou *File Transfer Protocol* (FTP). O *log* de atividade do *site* do IIS não deve ser confundido com o *log* de eventos executado pelo Windows XP, o qual é exibido através do visualizar de eventos. O *log* no IIS é mais abrangente.

O *log* para um *site* da *web* é efetuado por módulos que operam de forma independente de outras atividades no servidor.

O servidor ou servidores da *web* que compõe um *site* da *web* sofre demandas de desempenho extraordinárias. Eles devem ser capazes de responder a centenas ou milhares de solicitações de informações a cada segundo e retornar os dados solicitados para a *web*. A largura de banda do fluxo de dados da rede no *site* da *web* pode ser pequena, mas a largura de banda do fluxo de dados da rede de informações do *site* da *web* para a rede é enorme. Um *site* da *web* deve ser projetado para suportar cargas em momentos de pico, fornecendo, em tempo aceitável, respostas para usuários que são notoriamente impacientes. Os servidores da *web* para transações comerciais farão uso intensivo da largura de banda, mas terão modestas necessidades de armazenamento de dados. Acima de tudo, servidores da *web* devem ser confiáveis, se necessário empregando até processadores redundantes, bancos de dados de reserva imediatos (*hot standly databases*), armazenamento de disco RAID e diversas rede de roteamento (KIMBALL, 2000).

É possível escolher o formato dos *logs* para cada *site* da *web* ou FTP individual. Se o *log* estiver ativo para um *site*, pode-se desativá-lo ou ativá-lo para diretórios individuais daquele *site*.

Formatos de *log* diferentes usam fusos horários diferentes como base para os horários listados nos *logs*. O formato estendido do W3C[4] (*World Wide Web Consortium*) usa a *Universal Time Coordinaté* (UTC), coordenada universal de horários, chamada anteriormente Hora de *Greenwich*. Os outros formatos usam a hora local. Os horários listados nos arquivos de *log* refletem o horário que o servidor usa para processar solicitações e respostas. Esses horários não refletem o tempo de percurso da rede até o cliente ou o horário de processamento do cliente.

Os formatos de *log* pré-definidos, disponíveis para o IIS são:

- *log* estendido do *World Wide Web Consortium* (W3C);
- *log* do Microsoft IIS;
- *log* comum do NCSA.

Os formatos de *log* pré-definidos, disponíveis para o Apache são:

- *NCSA extended/combined log format;*
- *Common Log Format with Virtual Host;*
- *Referer log format;*
- *Agent (Browser) log format.*

O formato NCSA *Common* é também conhecido como CLF. Este formato de *log* é utilizado tanto para o IIS como para o Apache. Sendo que no Apache a configuração é feita através de diretivas de *log*. Já no IIS, a configuração é feita através da seleção do "Formato do arquivo de *log* comum do NCSA".

Os dados de *log* do servidor da *web* são a fonte primária da seqüência de cliques. Toda vez que o servidor da *web* responde a uma solicitação HTTP, uma entrada é feita no arquivo de *log* do servidor da *web*. O arquivo de *log* apresenta um desafio analítico particularmente difícil. Embora uma entrada seja feita para cada resposta de serviço, o servidor poderá estar mantendo centenas ou até milhares de sessões de usuários simultaneamente. Por causa disso, as entradas para uma sessão particular não são contíguas. Os registros individuais que

[4] O W3C é um grupo de organizações internacionais, neutro quanto a fabricantes, cujo objetivo é promover o desenvolvimento de protocolos-padrão a *World Wide Web*.

abrangem os rastros da sessão estão dispersos por todo o *log* e devem ser reunidos antes que uma análise completa de sessão possa ser concluída.

Segundo Kimball (2002), através de técnicas de *Webhouse* é possível seguir o cliente durante toda sua "viagem" de compras – para identificar o cliente, antes mesmo que ele veja a primeira página do *site*, e para segui-lo por sua experiência total de compras. Podemos medir o que ele olha, quanto tempo olha, o que seleciona e o que rejeita.

Segundo Kimball (2000), um *Data Webhouse:*

a) Armazena e publica dados de seqüência de cliques e outros dados comportamentais da *web* que guiam uma compreensão do comportamento do cliente;
b) É uma fonte adaptável e flexível de informações;
c) É extensível aos novos meios da *web*, incluindo imagens paradas (*still images*), imagens gráficas, áudio e vídeo;
d) É um bastão seguro que publica dados para clientes, parceiros de negócio e funcionários de forma adequada, mas que, ao mesmo tempo, protege os ativos de dados da empresa contra utilização não intencionada;
e) É a base para as decisões de conversão para a *we*b.

O servidor de aplicativo de data *Webhouse* é o painel de controle de todas as atividades dos usuários qualificados. Ele é capaz de acessar o *cache* de resposta automática, todos os mecanismos relacionais de banco de dados e os outros servidores de documentos e multimídia.

Este servidor entrega tudo em formato compatível com o navegador. Não importa se o usuário está dentro dos domínios físicos da empresa, ou se está realmente remotamente localizado na *web* (KIMBALL, 2000).

3.1 SERVIDOR APACHE

De acordo com uma importante empresa de pesquisas em servidores *web* chamada *Netcraft* (www.netcraft.co.uk/Survey), mais de 60% dos servidores *web* no mundo utilizam o Apache.

Segundo Kabir (2002), o Apache é um servidor *web* altamente configurável com um projeto modular, tecnologia gratuita e aberta.

Funciona muito bem com Perl, PHP e outras linguagens de scripts. Sendo que, um dos maiores recursos que o Apache oferece, é o fato de funcionar em quase todas as plataformas de computação mais utilizadas.

Na sua fase inicial o Apache era principalmente um servidor *web* baseado no Unix, mas essas atualmente, o Apache funciona também no Windows 2000/NT/9x e em muitos outros sistemas operacionais das classes de servidores e de *desktop*, tais como o "Amiga, OS 3.x e o OS/2" (KABIR, 2002).

Dentre os recursos que o Apache oferece, destaca-se o recurso de status do servidor e *log*s personalizáveis, pois conhecer o status e as informações de configuração do servidor é útil para gerenciar o servidor; porém, saber quem ou o que esta acessando o(s) *web site(s)* também é muito importante, além de ser interessante. É possível descobrir essas informações usando os recursos de registro de *log* do servidor Apache (KABIR, 2002).

O Apache informa os erros, gravando arquivos de *log*s de erros. Sem o registro de *log* de erros, não será possível descobrir o que está errado e onde ocorreu o erro. A fim de fazer uso dos dados registrados, a análise de *log* precisa ter a possibilidade de variar.

3.1.1 Criação de arquivos de *log*

Por default, a distribuição normal do Apache inclui um módulo chamado mod_log_config, responsável pelo registro de *log* básico, e ele grava arquivos de *log* CLF por default. A CLF atende aos requisitos de registro de *log* na maioria dos ambientes.

Os arquivos de *log* CLF contêm uma linha separada para cada solicitação. Uma linha é composta por vários *tokens* separados por espaço:

Host ident authuser date-time request status bytes

Se um *token* não tiver um valor, então ele será representado por um hífen (-). Os *tokens* têm os seguintes significados:

- ***Authuser***: se a URL solicitada exigir uma autenticação básica de http bem-sucedida, o nome do usuário será o valor desse *token*.
- ***Bytes***: o número de bytes no objeto retornado ao cliente, excluindo todos os cabeçalhos de http.

- ***Date***: a data e a hora da solicitação.
- ***Host***: o nome de domínio completamente qualificado do cliente, ou seu endereço IP.
- ***Ident***: se a diretiva *IdentyCheck* estiver ativa e a maquina cliente executar *identd*, então essas serão as informações de identidade relatadas pelo cliente.
- ***Request***: a linha de solicitação do cliente, colocada entre aspas (“).
- ***Status***: o código de status de http de três dígitos retornado ao cliente.

3.1.2 Diretivas de registro de *Log*

Uma diretiva é simplesmente um comando que faz o Apache agir de determinada maneira. Usando diretivas, um administrador do Apache pode controlar o comportamento do servidor *web*. (KABIR, 2002).

3.1.2.1 Diretiva TransferLog

TransferLog define o nome do arquivo de *log* ou programa para onde as informações de *log* deverão ser enviadas. Por default, as informações de *log* estão no formato CLF. Esse formato pode ser personalizado com o uso da diretiva de *LogFormat*. Contudo, se uma diretiva *LogFormat* não for encontrada dentro do contexto, será usado o formato de *log* do servidor.

Sintaxe: TransferLog *nomearq* | ″| caminho_para_programa/externo″ Configuração default: Nenhuma Contexto: Configuração do servidor, *host* virtual

A diretiva *TransferLog* recebe um caminho de arquivo de *log* ou um *pipe* (canal) para um programa externo como argumento. O nome do arquivo de *log* é considerado relativo à configuração do *ServerRoot*, se não for encontrado caractere / inicial. Quando o argumento é um *pipe* para um programa externo, as informações de *log* são enviadas para a entrada padrão do programa externo (STDIN).

3.1.2.2 Diretiva LogFormat

LogFormat define o formato do arquivo de *log* default denominado pela diretiva *TransferLog*. Se incluir um apelido para o formato na linha da diretiva, podemos usá-lo em

outras diretivas *LogFormat* e *CustomLog*, em vez de repetir a string de formato inteiro. Uma diretiva *LogFormat* que define um apelido não faz nada mais; isto é; ela só define o apelido e, na realidade, não aplica o formato.

Sintaxe: LogFormat formato [apelido] Configuração *default*: LogFormat %h %1 %u %t \ ²%r\² %>s %b Contexto: Configuração do servidor, *host* virtual.

3.1.2.3 Diretiva CustomLog

Como a diretiva *TransferLog*, essa diretiva permite enviar informações de registro de *log* a um arquivo de *log* ou a um programa externo. Porém, diferente da diretiva *TransferLog*, ela oferece a possibilidade de utilizar um formato de *log* personalizado que pode ser especificado como um argumento.

Sintaxe: CustomLog arquivo | *pipe* [formato | apelido] [env=[!]variável_de_ambiente] Configuração *default*: Nenhuma Contexto: Configuração do servidor, *host* virtual

As opções disponíveis para o formato são exatamente as mesmas para o argumento da diretiva *LogFormat*. Se o formato incluir espaços (o que acontecerá em quase todos os casos), ele deverá ser colocado entre aspas. Em vez de uma string de formato real, podemos usar um apelido de formato definido com a diretiva *LogFormat*.

3.1.2.4 Diretiva *CookieLog*

CookieLog permite registrar informações de *cookies* em um arquivo relativo ao caminho apontado pela diretiva *ServerRoot*. Essa diretiva não é recomendável, porque não é provável que ela tenha suporte no Apache por muito tempo. Para registrar dados nos *cookies* no *log*, é necessário utilizar o módulo de acompanhamento de usuários (*mod_usertrack*).

Sintaxe: CookieLog nomearq Configuração default: Nenhuma Contexto: Configuração do servidor, *host* virtual

3.1.2.5 Diretiva *LogLevel*

A diretiva *LogLevel* define o nível de detalhe da mensagem de *log* armazenada no arquivo de *log* de erros. Quando especificamos um nível de *log*, todas as mensagens de nível mais alto são escritas em um *log*. Assim, se especificarmos o nível como *crit*, somente os erros *emerg*, *alert* e *crit* serão registrados.

Sintaxe: LogLevel nível Configuração default: LogLevel erro Contexto: Configuração do servidor, host virtual

A Tabela 3 mostra os níveis disponíveis (em ordem decrescente) com seus respectivos significados.

Nível	Significado
Emerg	Situação de extrema emergência
Alert	Ação imediata exigida
Crit	Erros críticos
Error	Condições de erro
Warn	Mensagens de advertência
Notice	Notas de vários tipos
Info	Mensagens informativas
Debug	Mensagens de depuração

Tabela 4 - Níveis de LogDirective

A diretiva de *ErrorLog* especifica o nome do arquivo de *log* usado para registrar no *log* as mensagens de erros que o servidor produz. Se o nome do arquivo não começar com uma barra (/), ele será considerado relativo a *ServerRoot*.

Sintaxe: ErrorLog nomearq Configuração default: ErrorLog logs/error_log Contexto: configuração do servidor, host virtual

Para desativar o registro de *log* de erros, é necessário utilizar a seguinte Sintaxe: ErrorLog /dev/null

É muito importante que as configurações de permissões do diretório de *log* do servidor indiquem que apenas o usuário do Apache (especificado pela *diretive User*) terá permissão para acesso de leitura/gravação. Permitir que qualquer outra pessoa grave nesse diretório pode criar furos de segurança potenciais.

3.1.2.6 Diretiva PidFile

Usando a diretiva *PidFile*, podemos dar instruções ao Apache para gravar a ID de processo (ou PID) do servidor primário (isto é, o processo *daemon*) em um arquivo. Se o nome de arquivo não começar com uma barra (/), ele será considerado relativo a *ServerRoot*. A diretiva *PidFile* só é usada no modo autônomo.

Sintaxe: PidFile nomearq Configuração default: PidFile logs/httpd.pid Contexto: Configuração do servidor

O principal uso da diretiva PiFile consiste em tornar conveniente para o administrador do Apache encontrar a PID primária do Apache, necessária para enviar sinais ao servidor. Por exemplo, se o arquivo PID for mantido no diretório /usr/local/httpd/logs e seu nome for httpd.pid, um administrador poderá forçar o servidor Apache a reler sua configuração, enviando um sinal SIGHUP a partir do *prompt do Shell* (como raiz) desta forma:

```
Kill  HUP  'cat  / usr / local / httpd / logs / httpd.pid'
```

O mesmo comando faz o Apache reabrir *ErrorLog* e *TransferLog*.

3.1.2.7 Diretiva *ScoreBoardFile*

A diretiva *ScoreBoardFile* define o caminho até o arquivo usado para armazenar dados de processos internos. Se o nome de arquivo não começar com uma barra (/), ele será considerado relativo a *ServerRoot*. Esse arquivo é usado pelo processo do servidor primário para se comunicar com os processos filhos.

Sintaxe: ScoreBoardFile nomearq Configuração default: ScoreBoardFile logs/apache_status Contexto: Configuração do servidor

Para descobrir se o sistema exige o arquivo, é necessário simplesmente executar o servidor Apache e verificar se foi criado um arquivo no local especificado. Se a arquitetura do sistema exigir o arquivo, é necessário assegurar que esse arquivo não será utilizado ao mesmo tempo por mais de uma invocação do Apache. Além disso, é necessário certificar-se de que

nenhum outro usuário terá acesso de leitura ou gravação para esse arquivo, ou mesmo para o diretório em que ele é mantido.

3.1.3 Personalização de arquivos de *Log*

Embora o formato CLF *default* atenda à maioria dos requisitos de *log*, às vezes é útil poder personalizar dados de registro de *log*. Os formatos personalizados são definidos com as diretivas *LogFormat* e *CustomLog* do módulo. Uma *string* é o argumento de formato para *LogFormat* e *CustomLog*. Essa *string* de formato pode ter tanto caracteres literais quanto especificadores de formato especiais %. Quando são usados valores literais nessa *string*, eles são copiados no arquivo de *log* a cada solicitação. Contudo, os especificadores % são substituídos por valores correspondentes. Os especificadores % especiais são mostrados na Tabela 5 a seguir (KABIR, 2002).

Especificador %	Descrição
%a	Endereço IP de cliente
%A	Endereço IP de servidor
%B	Bytes enviados, excluindo cabeçalhos HTTP; 0 para nenhum byte enviado
%b	Bytes enviados, excluindo cabeçalhos HTTP; - para nenhum byte enviado
%c	Status da conexão quando a resposta termina. O *caracter* "X" é escrito se a conexão foi abortada pelo cliente antes da resposta poder ser concluída. Se o cliente usar um protocolo *keep-alive*, um símbolo "+" será escrito para mostrar que a conexão foi mantida ativa após a resposta e até o tempo limite. O sinal "-" é escrito para indicar que a conexão foi fechada após a resposta.
%{ mycookie } C	O conteúdo do *cookie* chamado *mycookie*
%D	O período de tempo (em microssegundos) gasto até a conclusão da resposta.
%{ myenv }e	O conteúdo de uma variável de ambiente chamado *myenv*
%f	O nome de arquivo da solicitação
%h	O *host* remoto que fez a solicitação
%H	O protocolo da solicitação (por exemplo, HTTP 1/1)
%{ IncomingHeader }i	O conteúdo de *IncomingHeade*r; isto é, a(s) linha(s) de cabeçalho na solicitação enviada ao servidor. O caractere i no final denota que esse é um cabeçalho de cliente (de entrada)
%I	Se a diretiva *IdentityCheck* estiver ativa e a máquina cliente executar *identd*, estas serão as informações de identidade relatadas pelo cliente.
%m	O método de solicitação (GET, POST, PUT e assim por diante
%{ ModuleNote}n	O conteúdo da nota *ModuleNote* de outro módulo
%{ OutgoingHeader }o	O conteúdo de *OutgoingHeade*r; isto é, a(s) linha(s) de cabeçalho na resposta. O caractere o no final denota que esse é um cabeçalho do servidor (de saída)
%p	A porta na qual a solicitação foi servida
%P	A ID de processo do filho que atendeu à solicitação
%q	A string de consulta
%r	A primeira linha da solicitação
%s	Status retornado pelo servidor em resposta à solicitação. Observe que, quando a solicitação é redirecionada, o valor desse especificador de formato ainda é o status da solicitação original. Se quiser armazenar o status de solicitação redirecionada, use %>s
%t	Hora da solicitação. O formato de hora é igual ao do formato CLF
%{format}t	A hora, na forma dada pelo formato. (Também é possível examinar a página man de strftime em sistemas Unix.)
%T	O tempo gasto para atender à solicitação, em segundos
%u	Se a URL solicitada exigiu uma autenticação básica de HTTP bem-sucedida, o nome de usuário será o valor desse especificador de formato. O valor pode ser falso, se o servidor retornar o status 401 (*Authentication Required*) após a tentativa de autenticação
%U	O caminho da URL solicitada
%v	O nome do servidor ou o *host* virtual para o qual a solicitação chegou
%V	O nome do servidor, de acordo com a diretiva *UseCanonicalName*

Tabela 5 - Especificadores % especiais para entradas de *log*

O formato CLF ou *NCSA Common*, são restritos a apenas alguns destes especificadores, sendo: %h %l %u %t %r %s %b, conforme mostrado anteriormente. A configuração *default* do Apache já vem com esse formato.

É possível incluir informações condicionais em cada um dos especificadores precedentes. As condições podem ser a presença (ou ausência) de certo(s) código(s) de status HTTP.

3.1.4 Criação de vários arquivos de *log*

É possível criar vários arquivos de *log*, usando a diretiva *TransferLog* e/ou a diretiva *CustomLog* do módulo *mod_log_config*. Basta repetir essas diretivas para criar mais de um arquivo de *log*.

Se por exemplo, quisermos criar um *log* de acesso CLF padrão e um *log* personalizado de todas as URLs de referência, pode-se usar:

```
TransferLog  logs/access_log
CustomLog  logs/referrer_log  ″%{Referer}i″
```

Quando tiver *TransferLog* ou *CustomLog* definidas na configuração do servidor primário e tiver um *host* virtual definido, o registro de *host* relacionado ao *host* virtual também está executado nesses *logs*. Por exemplo:

```
TransferLog  logs/access_log
CustomLog   log/agents_log  ″%{Referer}i″
<Virtual  Host  206.171.50.51>
        ServerName  reboot.nitec.com
```

```
        DocumentRoot  ″/www/reboot/public/cgi-bin/″
        ScriptAlias /cgi-bin/  ″/www/reboot/public/cgi-bin/″
<VirtualHost>
```

O *host* virtual reboot.nitec.com não tem uma diretiva *TransferLog* ou *CustomLog* definida dentro das *tags* de contêiner do *host* virtual. Todas as informações de registro de *log* são armazenadas em *logs/access_log* e *logs/agents_log*. Se a próxima linha for acrescentada

no interior do contêiner de *host* virtual*: TransferLog vhost_logs/reboot_access_log*, todo o registro de *log* para o *host* virtual reboot.nitec.com será feito no arquivo vhost_logs/reboot_access_log. Nenhum dos arquivos *logs/access_log* e *logs/agents_log* serão usados para o *host* virtual denominado *reboot.nitec.com*.

3.1.5 Análise de arquivos de *log*

É necessário um modo para analisar os *logs* criados, a fim de fazer uso dos dados registrados. A análise de *log* precisa ter a possibilidade de variar.

Segundo Kabir (2002), às vezes é necessário produzir relatórios extensivos, ou fazer uma simples verificação nos *logs*. A maioria dos sistemas Unix tem utilitários e ferramentas de scripts suficientes para fazer o trabalho.

Empregando utilitários do Unix para obter uma lista de todos os *hosts*, utilizando o recurso de *log default* ou um *log* personalizado com suporte CLF, é possível encontrar uma lista de todos os *hosts* com bastante facilidade. Por exemplo: Cat /path/to/httpd/access_log | awk ' {print $1} ' , imprime todos os endereços IP de *hosts*. O utilitário *cat* lista o arquivo *access_log* e a saída resultante é canalizada (*piped*) para o interpretador *awk*, que imprime apenas o primeiro campo de cada linha, usando a instrução *print*. Isso imprime todos os *hosts*. E para excluir os *hosts* da rede: cat / path / to / httpd / access_log | awk ' {print $1} ' | egrep –v ' (^206.171.50) ' , onde 2006.171.50 deve ser substituído pelo endereço de rede. Supondo que, a rede é da classe C, pois uma rede de classe B, só precisa dos dois primeiros *octetos* do endereço IP. Essa versão permite excluir os *hosts* com o utilitário *egrep*, configurado para exibir (via –v) apenas os *hosts* que não começam com o endereço de rede 206.171.50. Contudo, é possível que isso ainda não seja satisfatório, porque é provável que haja repetições. Desse modo a versão final é: cat /path/to/httpd/access_log | awk ' {print $1} ' | uniq | egrep –v ' (^206.171.50) '

O utilitário *uniq* filtra repetições e mostra apenas uma listagem por *host*. Para ver o número total de *hosts* exclusivos que acessaram o *web site*, pode-se canalizar (*pipe*) o resultado final para o utilitário wc com a opção –1, é da seguinte forma: cat /path/to/httpd/access_log | awk ' {print $1} ' | \ uniq | egrep –v ' (^206.171.50) ' | wc -1, fornecendo assim a contagem total de linhas (isto é, o número de acessos de *hosts* exclusivos) (KABIR, 2002).

Estão disponíveis muitas ferramentas de análise de *log* de servidores *web* de outros fornecedores. A maior parte dessas ferramentas espera que os arquivos de *log* estejam no formato CLF; assim, é necessário certificar-se de ter a formatação CLF em seus *logs*.

Conforme Kabir (2002), o melhor caminho para saber que ferramenta será mais útil é experimentar todas elas, ou pelo menos visitar os *web sites* para poder comparar suas características.

3.1.6 Manutenção do *log*

Os *logs* necessitam de manutenção. Conforme Kabir (2002), em *sites* do Apache com altas taxas de acesso ou muitos domínios virtuais, os arquivos de *log* podem se tornar enormes em pouco tempo, o que talvez cause uma crise de disco. Quando os arquivos de *log* se tornarem muito grande, é necessário fazer o rodízio dos arquivos.

Duas opções para fazer o rodízio dos *logs*: empregar um utilitário que acompanha o Apache, chamado *rotatelog*, ou usar *logrotate*, um recurso disponível na maioria dos sistemas Linux.

3.1.6.1 Como usar *rotatelog*

O Apache contêm o utilitário de suporte *rotatelog* que pode ser usado da seguinte forma:

```
TransferLog "| / caminho / para / rotatelogs  arquivodelog
Tempo_de_rotação_em_segundos>"
```

Por exemplo, para fazer o rodízio de *log* a cada 86.400 segundos (isto é, 24 horas):

```
TransferLog "| / caminho / para / rotatelogs  / var / logs / httpd  86400"
```

As informações de *log* de acesso de cada dia serão armazenadas em um arquivo chamado / var / logs / httpd . nnnn, onde nnnn representa um numero longo. Outra forma de fazer o rodízio é utilizando *piped logs*.

3.1.6.2 Como usar *logrotate*

O utilitário *logrotate* faz o rodízio, compacta e envia arquivos de *log* por correio eletrônico. Seu design foi desenvolvido para facilitar a administração de arquivos de *log* do sistema. Ele permite a rotação automática, a compactação, a remoção e o envio de arquivos de *log* diária, semanal ou mensalmente, ou de acordo com o tamanho. Em geral, *logrotate* é executado como um trabalho *cron* diário.

Para utilizar *logrotate* é necessário criar um script chamado / etc / logrotate.d / apache.

3.2 SERVIDOR IIS

3.2.1 Como configurar registros de *log*

A Figura 8, mostra a página de *propriedades de Site da Web padrão*, na qual será ativado o formato do *log*. Assim, é necessário selecionar *Ativar Logs* para ligar os recursos de *log* do servidor WWW. O registro de *log* é habilitado por default para que administradores possam rastrear os *sites* que estão sendo acessados e por quais usuários, a freqüência com que estes *sites* estão sendo acessados, se as respostas do servidor foram bem sucedidas e assim por diante (TULOCH e SANTRY, 2001).

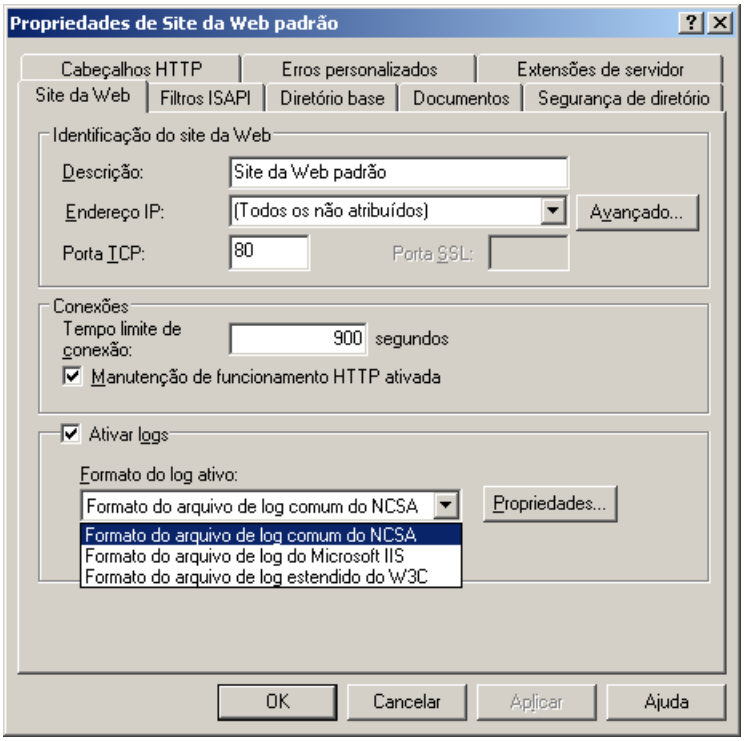


Figura 8 – Página de propriedades de S*ite* da *Web* padrão

É necessário selecionar um registro de *Formato de log ativo* da lista *drop-down*. No IIS, os registros de *log* possuem os seguintes formatos pré-definidos:

- *Microsoft IIS Log File Format;*
- *NCSA Common Log File Format;*
- *W3C Extended Log File Format;*
- *ODBC Logging.*

Os três primeiros formatos produzem simples registros de *log* de texto ASCII. Estes podem ser vistos em um editor simples, como o *Notepad*, importado para um banco de dados, ou importado para programas de análise de registro de *log*.

Além disso, o *W3C Extended Log File Format* é personalizável, permitindo aos administradores selecionarem os parâmetros a incluir no registro de *log*. O formato *default* para registro de *log* IIS é o *W3C Extended Log File Format.*

Segundo Tuloch e Santry (2001), para ajustar as opções de registro de *log* para estes três formatos, é necessário selecionar um dos formatos e clicar no botão *Propriedades* para acessar a página *Propriedades de log do NCSA* conforme mostra a Figura 9. Nesta página é necessário selecionar:

- O período de tempo de registro de *log*, por exemplo:
  - Criar novos registros de *log* em uma base diária, semanal ou mensal;
  - Permitir que o arquivo de registro de *log* cresça a um tamanho ilimitado;
  - Criar um novo arquivo de registro de *log* quando o antigo atingir um limite fixo especificado;
- A localização dos arquivos de registro de *log*, que, por *default*, é no diretório C:\winnt\system32\logfile\;
- Quais parâmetros incluir no registro de *log* (apenas para W3C *Extended Log File Format*).

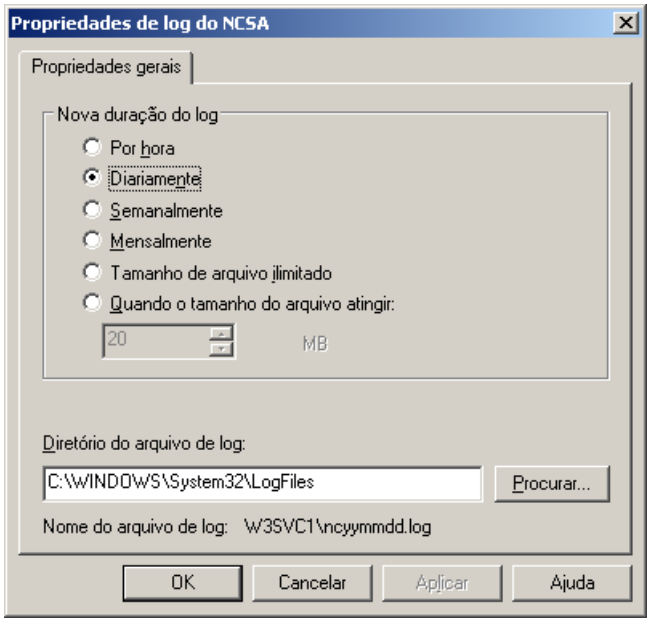


Figura 9 – Página de propriedades de *log* do NCSA

O exemplo mostrado na Tabela 6 apresenta o *formato do arquivo de log comum do NCSA* que será usado para o protótipo, pois esse formato pré-definido está disponível tanto para o IIS quanto para o Apache. A seguir serão apresentados outros formatos, conforme Tuloch e Santry (2001).

3.2.2 Como entender o formato de arquivo de registro de *log IIS Microsoft*.

Este formato é um arquivo de texto ASCII de campo fixo delimitado por vírgula. Um nome típico para tal arquivo de registro de *log* pode ser:

In000109.log	Criado em 9 de janeiro, 2000;
Inetsv20.log	O vigésimo arquivo de registro de *log* de tamanho limitado criado;

Segundo Tuloch e Santry (2001), a saída típica desse registro será como descrito a seguir:

10.107.3.201,-,01/9/00,11:02:22,W3SVC1,SERVER1,10.107.3.200,53998,275, 5585, 200, 0, GET, /Default.asp, - ,
10.107.3.201,-,01/9/00,11:02:28,W3SVC1,SERVER1,10.107.3.200,2354,395,200,0,GET,/iissamples/default/nav2.gif,-,
10.107.3.201,-,01/9/00,11:02:28,W3SVC1,SERVER1,10.107.3.200,2634,935,644,200,0,GET, /iissamples/default/msft.gif,-,

Para ajudar na interpretação do *Microsoft IIS Log File Format*, a Tabela 5 relaciona os campos contidos na última entrada do registro de *log* anterior e seus significados.

Dados de elemento	Valor típico

Endereço IP do cliente	10.107.3.201
Nome de usuário do cliente (– se anônimo)	-
Data de solicitação	9/1/00
Horário de solicitação	01:02:56
Serviço solicitado (W3SVC1 é o servidor WWW)	W3SVC1
Nome do servidor	SERVER1
Endereço IP do servidor	10.107.3.200
Tempo de espera de processador do servidor (ms)	1282
Bytes enviados pela solicitação do cliente	395
Bytes retornados pela resposta do servidor	2167
Código de status http retornado	200
Código de status Win32 retornado	0
Método solicitado	GET (obter)
Documento solicitado	//iissamples/default/nav2.gif

Tabela 5 - Campos do arquivo de *log* no formato *Microsoft IIS Log File Format*

3.2.3 Como entender o formato de arquivo de registro de *log* comum NCSA

O *NCSA Log File Format* é um arquivo de texto ASCII de campo fixo delimitado por espaço. Um nome típico para tal arquivo de registro de *log* pode ser:

nc000109.log	Criado em 9 de janeiro, 2000;
ncsa20.log	O vigésimo arquivo de registro de log de tamanho limitado criado;

Segundo Tuloch e Santry (2001), a saída típica de tal registro pode ser como descrito a seguir:

```
10.107.3.201 - - [09/Jan/2000:11:06:39 -0600] “GET /iissamples/default/iis3.GIF HTTP/1.0” 200 3558
10.107.3.201 - - [09/Jan/2000:11:06:42 -0600] “GET / default.asp HTTP/1.0”  200 5518
10.107.3.201 - - [09/Jan/2000:11:06:46 -0600] “GET /iissamples/default/IE.GIF HTTP/1.0”  200 8866
```

Para ajudar na interpretação do formato de arquivo de registro de *log* comum NCSA, a Tabela 6 relaciona os campos contidos na última entrada do registro de *log* anterior e seus significados.

Dados de elemento	Valor típico
Endereço IP do cliente	201.4.239.102
Domínio/nome de usuário do cliente	-
Authuser	-

Data e horário da solicitação	25/Apr/2004:04:15:38
Diferença GMT	-0300
HTTP solicitado	"GET /~maw/img/pacer.gif HTTP/1.1"
Código de situação HTTP retornado	404
Bytes retornados pela resposta do servidor	672

Tabela 6 - Campos do arquivo de *log* no formato *NCSA Common Log File Format*

3.2.4 Como entender o W3C *Extended Log File Format*

O formato de arquivo de registro de *log* estendido W3C é um arquivo de texto ASCII de campo variável, delimitado por espaço, com cabeçalhos. Um nome típico para tal arquivo de registro de *log* pode ser:

ex000109.log	Criado em 9 de janeiro, 2000;
extend20.log	O vigésimo arquivo de registro de log de tamanho limitado criado;

Segundo Tuloch e Santry (2001), a saída típica deste registro pode ser como descrito a seguir, quando todas as opções de registro de *log* estendidas são selecionadas:

#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2000-01-10 01:02:54
#Fields: date time c-ip cs-username s-sitename s-computername s-ip cs-method cs-uri-stem cs-uri-query sc-status sc-win32-status sc-bytes time-taken s-port cs(User-Agent) cs(Cookie) cs(Referer)
2000-01-10 01:02:54 10.107.3.201 - W3SVC1 SERVER1 10.107.3.200 GET /Default.asp - 200 0 5585 275 4697 80 Mozilla/2.0+(compatible;+MSIE3.0;+Windows195) - - 2000-01-10 01:02:56 10.107.3.201 - W3SVC1 SERVER1 10.107.3.200 GET /iissamples/default/IISTitle.gif - 200 0 21576 399 761 80 Mozilla/2.0+(compatible;+MSIE3.0;+Windows195) ASPSESSIONIDGQGGGGYP5HILJEGPCGGLNFFKEKHEBCGHM http://server1/ 2000-01-10 01:02:56 10.107.3.201 - W3SVC1 SERVER1 10.107.3.200 GET /iissamples/default/nav2.gif - 200 0 2167 395 1282 80 Mozilla/2.0+(Compatible;+MSIE13.0;Windows195) ASPSESSIONIDGOGGGGYP5HILJEGPCGGLNFFKEKHEBCGHM http://server1

Para ajudar na interpretação do W3C *Extend Log File Format*, a Tabela 7 relaciona os campos contidos na última entrada do registro de *log* anterior e seus significados.

Dados de elemento	Valor típico
Data de solicitação	10-01-2000
Horário da solicitação	01:02:56
Endereço IP do Cliente	10.107.3.201

Nome de usuário do cliente (- se anônimo)	-
Serviço solicitado (W3SVC1 é o serviço WWW)	W3SVC1
Nome do servidor	SERVER1
Endereço IP do servidor	10.107.3.200
Método solicitado	GET
Documento solicitado	/iissamples/default/nav2.gif
Query de busca (se houver)	-
Código de status http retornado	200
Código de status de Win32 retornado	0
Bytes retornados pela resposta do servidor	2167
Bytes enviados pela solicitação do cliente	395
Tempo de espera do processador do servidor (ms)	1282
Porta TCP do servidor	80
Tipo de cliente (conhecido como agente-usuário)	Mozilla/4.0+(compatible;+MSIE+4.0; +Windows+98
Cookie (se houver)	ASPSESSIONIDGQGGGGYP = HILJEGPCGGLNFFKEKHEBCGHM
Referido (o *site* que contém o link que o usuário clicou para obter esta página)	http://server1

Tabela 7 - Campos do arquivo de *log* no formato *W3C Extendedn Log File Format*

3.2.5 Como entender o registro de *log* ODBC

O quarto formato de registro de *log* é o ODBC *Logging*, específico do IIS. O ODBC *Logging* permite aos administradores registrar transações WWW diretamente em um banco de dados compatível com ODBC, tal como *Microsoft SQL Server ou Microsoft Access*. As etapas envolvidas no registro de *log* em um banco de dados são:

1. Criação de um banco de dados e definição de uma tabela dentro dele. É necessário atribuir um nome à tabela (o nome *default* sugerido é *Internet Log*).

2. Criação dos seguintes campos dentro da tabela, para conter os dados registrados:

Campo	Tipos de dados
ClientHost (Cliente host)	varchar (255)
Username (Nome do usuário)	varchar (255)

LogTime (Horário de registro)	Datetime
Service (Serviço)	varchar (255)
Machine (Equipamento)	varchar (255)
ServerIP (Servidor IP)	varchar (50)
ProcessingTime (Tempo de processamento)	int
BytesRecvd (Bytes recebidos)	int
BytesSent (Bytes enviados)	int
ServiceStatus (Status do servico)	int
Win32Status (Status de Win32)	Int
Operation (Operação)	varchar (255)
Target (Alvo)	varchar (255)
Parameters (Parâmetros)	varchar (255)

Tabela 8 – Campos do arquivo de *log* no formato ODBC

3. Na página de propriedades *System* DNS da opção ODBC no painel de controle, é necessário fornecer ao banco de dados um sistema DNS. Isto é necessário para que ODBC possa referenciar a tabela. O default DNS sugerido é HTTPLOG.

4. É necessário selecionar o ODBC *Logging* como formato de registro de *log Active* na guia *Web site* da página WWW *Service Master Properties*. Em seguida, é necessário clicar no botão *Properties* para acessar a página de propriedades ODBC *Logging* (Figura 10).

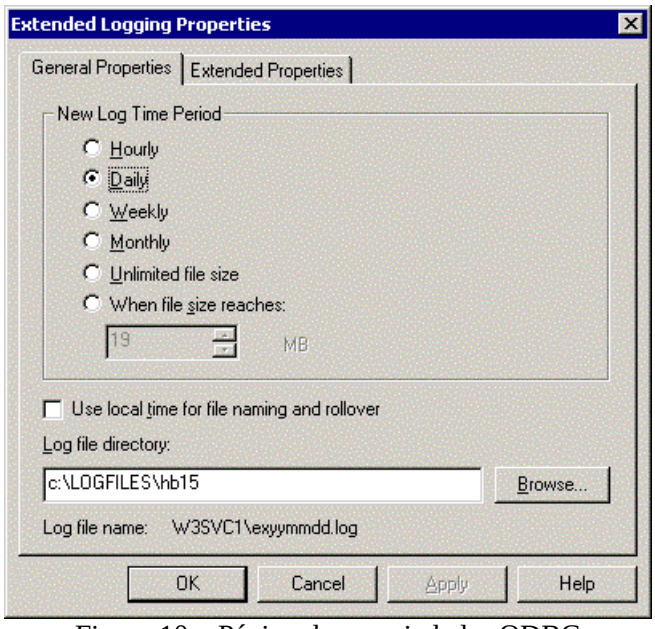


Figura 10 – Página de propriedades ODBC

5. É necessário entrar com o DNS, o nome da tabela, o nome do usuário e a senha, se estes forem necessários para conectar o banco de dados. Em seguida, é necessário clicar em OK para começar a registrar.

3.2.6 Como definir *log* personalizado.

Além dos formatos de *log* pré-definido, é possível definir manualmente os campos que o servidor IIS irá gravar no arquivo de *log*. A Figura 11 mostra a tela de propriedades estendidas usada para esse propósito.

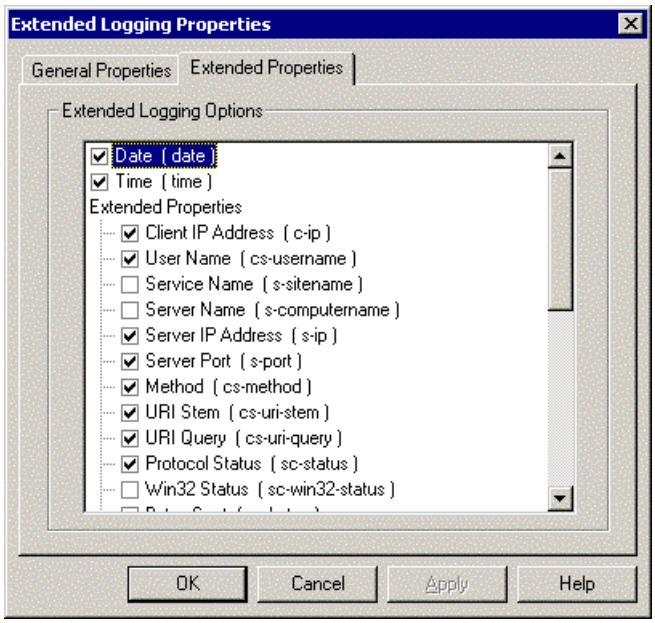


Figura 11 – Página de propriedades estendidas do log.

3.3 MENSAGENS DE ERRO HTTP.

Conforme Tuloch e Santry (2001), quando um cliente (*browser*) faz uma solicitação http a um servidor, o servidor responde, enviando uma série de cabeçalhos de resposta, seguida pelo arquivo ou arquivos solicitado(s). Se a transação é bem sucedida, o primeiro cabeçalho solicitado enviado pelo servidor, tipicamente, é: **HTTP/1.1 200 OK**

O número 200 no cabeçalho de resposta é um código de status HTTP que indica que a transação foi bem sucedida. Códigos de status HTTP geralmente pertencem a uma das três categorias:

a) 200 a 299 – significa que a transação foi bem sucedida;
b) 300 a 399 – significa que ocorreu um redirecionamento;
c) 400 a 599 – significa que ocorreu algum tipo de erro.

4 LOGS DE SERVIDOR DA WEB

Segundo Kimball (2000), os servidores *web* têm a capacidade de registrar interações de clientes em um ou mais arquivos de *log* ou banco de dados, ou para canalizar as informações de *log* para outros aplicativos em tempo real. Esses elementos de dados também estão disponíveis para serem passados para aplicativos de tempo real utilizando a *Common Gateway Interface* (CGI) do servidor da *web*. A Tabela 9 lista alguns elementos de dados típicos disponíveis na maioria dos servidores da *web*.

O padrão original para *log*s de servidor da *web* era o formato comum de *log* (*Commom Log Format* – CLF), às vezes chamado de CLOG. Esse padrão incluiu os sete elementos de dados marcados na coluna CLF na Tabela 9. Dois elementos adicionais foram somados no padrão estendido de formato comum de *log* (*Extended Common Log Format* – ECLF), e estes estão marcados na coluna ECLF da Tabela 9. Vários servidores da *web* adicionam outros parâmetros registráveis, mas este inevitavelmente limitado pelas informações contidas no protocolo básico de HTTP. Os elementos de dados de *log* são discutidos com mais detalhes nos parágrafos a seguir.

Os dados de *log* do servidor da *web* são a fonte primária da seqüência de cliques. Toda vez que o servidor *web* responde a uma solicitação HTTP, uma entrada é feita no arquivo de *log* do servidor da *web*. O arquivo de *log* apresenta um desafio analítico particularmente difícil. Embora uma entrada seja feita para cada resposta de serviço, o servidor poderá estar mantendo centenas ou até milhares de sessões de usuários simultaneamente. Por causa disso, as entradas para uma sessão particular não são contíguas. Os registros individuais que abrangem os rastros da sessão estão dispersos por todo o *log* e devem ser reunidos antes que uma análise completa de sessão possa ser concluída. (KIMBALL, 2000).

Servidores de *Web* registram uma entrada para todo único acesso que eles recebem, registrando pedaços importantes de informação: o URL requisitado, o endereço IP origem de qual o pedido originou e um tempo de duração. Uma entrada de *log* é automaticamente somada cada vez que um pedido para um recurso alcança o servidor web. A Figura 12 mostra um fragmento de um arquivo de *log*.

Alguns recursos podem ser gravados em *cache* pelo *browser* do usuário ou por um *proxy*, e o conteúdo das páginas gravado em *cache* não é registrado nos *logs* do servidor *web*.

```
204.184.107.249 - - [26/Jul/1999:10:30:40 -0800] "GET -lpease HTTP/1.0" 302
202.231.204.76 - - [26/Jul/1999:10:31:44 -0800] "GET /~webcom/html/publiciz
198.253.36.22 - - [26/Jul/1999:10:31:44 -0800] "GET /~teapot/vr9.html HTTP/
209.185.188.232 - - [26/Jul/1999:10:31:44 -0800] "GET -progsys HTTP/1.0" 30
208.255.225.254 - - [26/Jul/1999:10:32:48 -0800] "GET / HTTP/1.0" 200 9345
199.206.254.61 - - [26/Jul/1999:10:34:56 -0800] "GET / HTTP/1.0" 400 848 "*
216.81.26.224 - - [26/Jul/1999:10:34:56 -0800] "GET /index.html HTTP/1.0* 2
162.119.232.100 - - [26/Jul/1999:10:34:56 -0800] "GET /info/raves/ HTTP/1.0
207.34.100.198 - - [26/Jul/1999:10:37:04 -0800] "GET /duplex/sk8_1184.html
206.154.159.40 - - [26/Jul/1999:10:38:08 -0800] "GET /~ctt/insects.html HTT
195.122.15.31 - - [26/Jul/1999:10:38:08 -0800] "GET /jyda HTTP/1.0" 302 390
166.70.45.193 - - [26/Jul/1999:10:28:32 -0800] "POST /cgi-bin/dns_check HTT
205.142.31.16 - - [26/Jul/1999:10:34:56 -0800] "POST /cgi-bin/dns_check HTT
205.142.31.16 - - [26/Jul/1999:10:36:00 -0800] "POST /cgi-bin/dns_check HTT
199.8.80.189 - - [26/Jul/1999:10:33:52 -0800] "POST /cgi-bin/dns_check HTTP
204.184.107.249 - - [26/Jul/1999:10:30:40 -0800] "GET -lpease HTTP/1.0" 302
202.231.204.76 - - [26/Jul/1999:10:31:44 -0800] "GET /~webcom/html/publiciz
198.253.36.22 - - [26/Jul/1999:10:31:44 -0800] "GET /~teapot/vr9.html HTTP/
209.185.188.232 - - [26/Jul/1999:10:31:44 -0800] "GET -progsys HTTP/1.0" 30
208.255.225.254 - - [26/Jul/1999:10:32:48 -0800] "GET / HTTP/1.0" 200 9345
199.206.254.61 - - [26/Jul/1999:10:34:56 -0800] "GET / HTTP/1.0" 400 848 "*
216.81.26.224 - - [26/Jul/1999:10:34:56 -0800] "GET /index.html HTTP/1.0* 2
162.119.232.100 - - [26/Jul/1999:10:35:58 -0800] "GET /help/webcommerce/bui
207.34.100.198 - - [26/Jul/1999:10:37:04 -0800] "GET /duplex/sk8_1184.html
206.154.159.40 - - [26/Jul/1999:10:38:08 -0800] "GET /~ctt/insects.html HTT
195.122.15.31 - - [26/Jul/1999:10:38:08 -0800] "GET /joda HTTP/1.0" 302 390
205.188.208.70 - - [26/Jul/1999:10:36:00 -0800] "GET /~pinknoiz/covert/wilc
198.253.36.22 - - [26/Jul/1999:10:33:52 -0800] "GET /~amraam/es3a.html HTTP
207.34.100.198 - - [26/Jul/1999:10:39:12 -0800] "GET /duplex/sk8_1188a.htm
142.165.107.52 - - [26/Jul/1999:10:39:12 -0800] "GET -dynamic HTTP/1.0" 302
207.34.100.198 - - [26/Jul/1999:10:34:56 -0800] "GET /duplex/sk8_1058.html
162.119.232.100 - - [26/Jul/1999:10:38:08 -0800] "GET /help/webcommerce/sho
207.138.42.10 - - [26/Jul/1999:10:39:12 -0800] "GET -wrsl HTTP/1.0" 302 178
129.74.235.127 - - [26/Jul/1999:10:40:16 -0800] "GET /oldgolf/feathery.html
161.142.78.82 - - [26/Jul/1999:10:37:04 -0800] "GET / HTTP/1.0* 200 9345 [2
209.197.236.4 - - [26/Jul/1999:10:42:24 -0800] "GET / HTTP/1.0* 200 9345 "*
208.250.29.19 - - [26/Jul/1999:10:44:32 -0800] "GET /tedsite/drillcompfluid
205.142.31.16 - - [26/Jul/1999:10:36:00 -0800] "POST /cgi-bin/dns_check HTT
208.255.225.254 - - [26/Jul/1999:10:39:12 -0800] "POST /cgi-bin/dns_check H
24.4.254.39 - - [26/Jul/1999:10:34:56 -0800] "POST /cgi-bin/contract HTTP/1
24.4.254.39 - - [26/Jul/1999:10:34:56 -0800] "POST /cgi-bin/contract HTTP/1
205.142.31.16 - - [26/Jul/1999:10:37:04 -0800] "POST /cgi-bin/dns_check HTT
208.255.225.254 - - [26/Jul/1999:10:40:16 -0800] "GET /cgi-bin/application
129.188.33.221 - - [26/Jul/1999:10:32:48 -0800] "GET /help/start.shtml HTTP
216.77.241.11 - - [26/Jul/1999:10:31:44 -0800] "GET /help/webcommerce/build
192.31.7.244 - - [26/Jul/1999:10:38:08 -0800] "GET /help/form_proc/ HTTP/1.
192.31.7.244 - - [26/Jul/1999:10:37:04 -0800] "GET /html/tutor/forms/intro.
209.203.119.62 - - [26/Jul/1999:10:37:04 -0800] "GET /html/tutor/forms/intr
209.30.244.168 - - [26/Jul/1999:10:39:12 -0800] "GET /info/raves/ HTTP/1.0*
208.255.225.254 - - [26/Jul/1999:10:38:08 -0800] "GET /info/options.shtml H
12.10.41.244 - - [26/Jul/1999:10:32:48 -0800] "POST /cgi-bin/dns_check HTTP
203.134.2.91 - - [26/Jul/1999:10:34:56 -0800] "POST /cgi-bin/dns_check HTTP
```

Figura 12 – Fragmento de arquivo de *log* de um servidor *web*

Elementos de Dados	CLF*	ECLF*	Descrição
Host	✓	✓	Nome de domínio plenamente qualificado do cliente, ou seu endereço de IP se o nome não estiver disponível.
Ident	✓	✓	Informações de identidade fornecidas pelo cliente, se o identd estiver ativado.
Authuser	✓	✓	Se a solicitação foi para um documento protegido por senha, então esse é o ID de usuário utilizado na solicitação.
Time	✓	✓	A data/hora em que a solicitação alcançou o servidor em formato de tempo CLF {zona dd/Mmm/aaaa:hh:mm:ss}.
Request	✓	✓	A primeira linha da solicitação do cliente (normalmente entre aspas).
Status	✓	✓	Código de status de três dígitos retornado para o cliente.
Bytes	✓	✓	Numero de bytes retornado para o cliente excluindo cabeçalhos de HTTP.
Referrer		✓	Representa o *site* que contém o *link* que o usuário clicou para obter esta página, ou seja, a URL do servidor de referência.
User-agent		✓	Nome e versão do cliente (navegador).
Filename			Nome de arquivo.
Time-to-server			Tempo para atender à solicitação (em segundos).
IP-address			Endereço de IP do *host* remoto (veja "*host*" a cima).
Server-port			Porta canônica do servidor que satisfaz a solicitação.
Process ID			ID de processo filho que fez a manutenção da solicitação.
Formatted-time			A data/hora, em formato strftime (3) especificado.
URL-requested			O caminho de URL solicitado
Server-name			O nome canônico do servidor que atende à solicitação.
Cookie			O valor do *cookie* recuperado do arquivo de *cookie* do cliente.
* Formato comum de *log* (*Common Log Format* – CLF) e Padrão estendido de formato comum de *log* (*Extended Common Log Format* – ECLF).			

Tabela 9 - Os elementos de *log* de servidor da *web*

4.1 ESPECIFICAÇÃO DOS CAMPOS DE UM LOG.

Abaixo está a descrição dos principais campos de um arquivo de *log*, conforme Kimball (2000):

4.1.1 Host

O *host* é o endereço de Internet do navegador ou de outro agente fazendo a solicitação HTTP. Esse é o endereço para o qual a resposta do servidor será enviada. O *host* é inicialmente adquirido pelo servidor da *web* como endereço IP numérico, como “209.4.5.122.19”. A maioria dos servidores da *web* tem a capacidade de transformar esse endereço em um nome de domínio de texto utilizando um protocolo de consulta de Internet denominado pesquisa inversa de endereço (*reverse address lookup*). O endereço IP é enviado de volta pela Internet para um roteador que, com autoridade, pode converter o nome de domínio que, então, é retornado para o solicitante. Isso torna os *log*s muito mais legíveis, mas fazer pesquisa inversa de endereço em tempo real pode aumentar a carga no servidor em até 40 por cento que representa um *overhead* inaceitável em sistemas de alto volume.

A maioria dos PCs em rede não tem endereços de IP fixos. Em vez disso, o endereço de IP é atribuído dinamicamente para o PC quando o usuário faz uma conexão com seu provedor através de uma conexão de modem discada ou a cabo. Mesmo que o endereço IP seja dinâmico, ele permanece fixo durante uma sessão de navegador e pode ser utilizado para amarrar os eventos de sessão.

Por causa do predomínio de atribuições dinâmicas de IP, muitos nomes de *hosts,* por si próprios, fornecem poucas informações de valor. No entanto, eles realmente fornecem uma chave para amarrar diversos eventos de sessão de usuário na ausência de um mecanismo mais confiável, como *cookies* ou IDs de sessão gerados por servidor. Mesmo que muitos nomes de *hosts* tenham pouco significado, alguns podem fornecer informações importantes.

4.1.2 Ident

O elemento de dados *Ident* é um identificador arbitrário fornecido pelos aplicativos clientes que suportam um protocolo chamado *identd* (*identification daemon*).

Sempre aparece como um hífen (-). Os *logs* CLF originais deveriam armazenar alguma informação retornada pelo *identd* para a conexão. *Identd* é somente um serviço UNIX e é raramente utilizado por navegadores da *web*. O *ident* também as causa utilização excessiva da largura de banda da rede (*network bandwidth*) e recursos extras do servidor (*overhead*), por estas razões ele não é gravado.

4.1.3 Authuser

O elemento de dados *Authuser (authenticated user)* é um ID de usuário passado em uma solicitação feita via o *Secure Sockes Layer* (SSL) do HTTP. Esse campo será preenchido se um usuário passou corretamente um *logon* seguro de servidor, e poderá ser utilizado para associar registros de *log* de usuário quando operando sob protocolo SSL.

4.1.4 Time

Este é normalmente a data/hora em que o servidor da *web* completou a resposta à solicitação de HTTP.

4.1.5 Request

Request é a linha real de solicitação do navegador. Em geral, ela se parece com o seguinte: “GET / images/under-c.gif http/1.0”.Esta contém informações sobre o que o arquivo do cliente quer e como exatamente ele deveria ser enviado.

Por exemplo: "method url protocol/version" = "GET /index.html HTTP/1.0", sendo:

method	=	métodо HTTP , bem como GET ou POST
url	=	A URL solicitada pelo cliente
protocol	=	Sempre HTTP
version	=	O número da versão do protocolo.

4.1.6 Status

Status é o código de três dígitos que o servidor retorna para o navegador, tal como 200 (OK) ou 404 (Não localizado). Uma classificação para os códigos existentes está no capítulo 4.3.

4.1.7 Bytes

Bytes é a contagem de bytes retornada ao cliente pelo servidor.

4.1.8 Referrer

A origem do link (*referrer*) é uma string de texto que pode ser enviada pelo cliente para indicar a fonte original de uma solicitação ou de um link. Esse elemento de dados foi

adicionado ao protocolo HTTP 1.0 para permitir a um *site* da *web* rastrear retroativamente referências a suas origens. Isso permite dar créditos a *clickthoughs* de anúncios bem como outros tipos de créditos de referência. Uma entrada de *log* de origem do link (*referrer*) pode tornar a forma do URL de origem, seguida pelo recurso para o qual a referência aponta:

http://www.webcom.com/megasite/ -> index.html

Este caminho pode não ser enviado do cliente para o servidor a cada *request*, então será representado por um hífen "-" no arquivo de *log*.

4.1.9 User-agent

O agente de usuário é o nome e a versão do software do cliente fazendo a solicitação e o sistema operacional sob o qual o cliente está operando. Essas informações são utilizadas pelo servidor *web* para determinar o conjunto de recursos suportado pelo navegador do cliente e assegurar que a resposta contenha somente itens que possam ser adequadamente interpretados e exibidos pelo navegador. Por exemplo, se um navegador não suporta objetos *ActiveX*, então não é necessário incluir tais objetos para obter respostas. Agentes de usuários não precisam ser constrangidos por navegadores, conforme a segunda dessas duas entradas:

“Mozilla / 4.0 (compatible; MSIE 4.01; Windows98)”

“Scooter / 2.0 G.R.A.B v.1.0”

A primeira das duas entradas de *log* de usuário-agente vem de um navegador convencional do Microsoft Internet Explorer sendo executado sob Windows 98. A segunda vem de um *spider* de sistema de pesquisa do Alta Vista sendo executado sobre um sistema operacional não especificado.

4.1.10 Filename

O nome do arquivo é parte de um URL que especifica o caminho e o nome de um arquivo sendo acessado. Às vezes, é expresso como um caminho qualificado completamente (completo) e, às vezes, como um caminho relativo a *home page* em uma estrutura de diretórios de *site* da *web*.

4.1.11 Time-to-Server

Esse é o tempo que um servidor da *web* leva para enviar a resposta a uma solicitação de HTTP. Essa métrica é útil para computar o tempo de visualização da página pelo usuário e o desempenho de servidor, mas não pode estar disponível em alguns servidores da *web*.

4.1.12 IP Address

É o endereço IP numérico de um *site web*. Ele pode ser expresso por quatro *octetos* binários (125.32.221.26) ou por um número decimal, *octal* ou hexadecimal.

4.1.13 Server Port

Este é o número da porta de TCP/IP em um *host*, que serviu à atividade de registro. É prática padrão da *web* reservar a porta 80 para um servidor de HTTP e a porta 443 para um servidor seguro de HTTP (que esteja utilizando SSL).

4.1.14 Process ID

É o número do processo filho de servidor da *web* que serviu à solicitação.

4.1.15 URL

Um URL (*Uniform Resource Locator*) completo contém vários segmentos como demonstrado a seguir.

http://ralphkimball.com:433/seminars/schedule.html?tokyo+fall+2001

Esquema	=	http://
Nome do host	=	ralphkimball.com
Porta	=	: 433
Caminho de documento	=	/seminars/schedule.html
String de consulta	=	?tokyo+fall+2001 (tudo depois do ponto de interrogação)

O esquema especifica um protocolo de comunicação tal como HTTP ou FTP. O nome de *host* é o endereço de Internet totalmente qualificado do *host* que contém as informações, e pode ser um nome ou endereço de IP. O número (opcional) de porta é a porta no *host* para a qual endereçar a mensagem. O caminho do documento é um nome de caminho absoluto ou

relativo para o documento solicitado. A string de consulta é uma string de texto que se segue ao caminho que será utilizada para uma ampla gama de propósitos específicos do aplicativo, além de simplesmente transportar consultas.

Ocasionalmente poderá aparecer o termo URI (*Uniform Resource Indicator*), que é uma categoria genérica que inclui tanto URLs como *Uniform Resource Names*. Um *Uniform Resource Name* é um termo não rigorosamente especificado para um nome que identifica um recurso de maneira exclusiva, como um documento, na Internet.

4.1.16 Cookies

O mecanismo de *cookie* fornece a um servidor da *web* a capacidade de armazenar uma string de texto no computador de um cliente que, mais tarde, poderá ser lida pelo servidor. Os *cookie*s podem ser persistentes ou de nível de sessão. Um *cookie* de sessão é armazenado na memória do computador do cliente, mas não é retirado quando o navegador é fechado. Um *cookie* persistente é armazenado em disco e poderá ser lido mais tarde. Os *cookies* são pouco seguros, porque somente podem ser lidos por um cliente de HTTP no (ou destinado a estar no) nome de domínio armazenado no arquivo de *cookie*.

5 IMPLEMENTAÇÃO DO PROTÓTIPO

O protótipo foi implementado em Java utilizando o paradigma da Orientação a Objetos. Foi utilizada ferramenta Eclipse para desenvolvimento do protótipo.

O algoritmo Apriori utilizado para fazer as regras de Associação tem sua origem no projeto Weka, disponível em Weka (2004), sendo que a biblioteca para esse algoritmo foi incluída no pacote com pequenos ajustes para funcionar com o protótipo.

Não foi utilizado banco de dados para importar os arquivos de *log* pela questão de *performance*, pois a intenção era que o protótipo pudesse carregar um *log* muito grande para memória, como por exemplo, o *log* de um provedor de acesso, e dessa forma, o tempo de I/O (Input / Output) gasto para importar as transações para o banco de dados seria eliminado.

5.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O protótipo desenvolvido obedeceu a alguns Requisitos Funcionais (RF) e alguns Requisitos Não Funcionais (RNF), os quais são:

a) Lê arquivos de *log* do Internet Information Services (IIS) ou APACHE (RF);
b) Analisa o arquivo lido contando e agrupando informações, registradas nos *logs* de acesso do IIS e APACHE (RF);
c) Disponibiliza gráficos estatísticos dos acessos ao *site*, pelo endereço Internet Protocol (IP) do cliente ou estação (RF);
d) Mostra a quantidade de usuários que acessam um *Website* (RF);
e) O arquivo de *log* deve estar em um dos formatos gerado por um servidor com o IIS ou APACHE (RF);
f) O computador que rodar o protótipo deve possuir no mínimo 128 MB de memória RAM (Random Acess Memory) (RNF);
g) Cada arquivo de *log* a ser analisado pelo protótipo não deve ultrapassar a movimentação máxima de um dia inteiro ou 150 MB (RNF);
h) O protótipo é amigável (RNF);
i) O protótipo roda em um web browser (RNF).

5.2 ESPECIFICAÇÃO

A metodologia utilizada para especificar esse sistema de mineração de dados em arquivos de *log* gerados por servidores de páginas *web* foi à análise orientada a objetos, baseada em diagrama de caso de uso, diagrama de classes e diagrama de atividades.

Nesta especificação foi utilizada a ferramenta UML Poseidon como recurso na modelagem orientada ao objeto.

O desenvolvimento de um sistema para mineração de dados requer em primeira instância um estudo criterioso sobre as técnicas de *Data Mining* existentes, com o intuito de descobrir qual a mais sugerida para o resultado que se deseja alcançar.

5.2.1 DIAGRAMA DE CASO DE USO.

Através da ferramenta Poseidon, é apresentada a especificação formal do problema utilizando o diagrama de caso de uso (Figura 13). Para o desenvolvimento deste diagrama é necessário saber quais as principais tarefas relacionadas com o problema e quem irá executar estas tarefas.

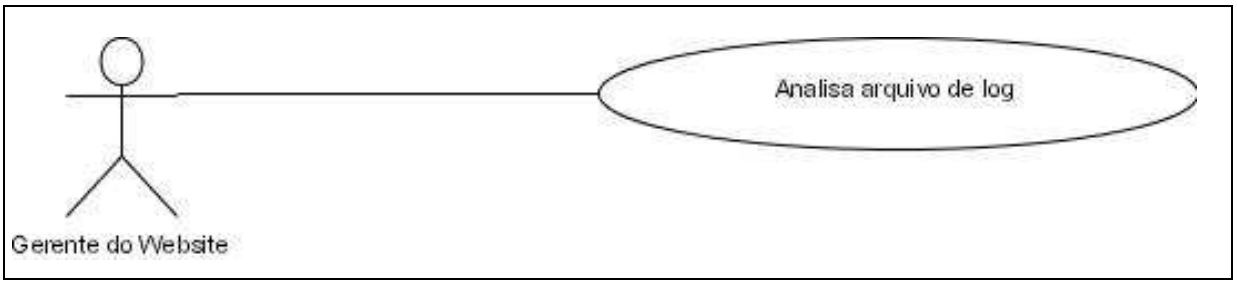


Figura 13 - Diagrama de caso de uso

5.2.2 DIAGRAMA DE CLASSES.

Na Figura 14 é apresentado o diagrama de classes, que demonstra os relacionamentos entre as classes. Através deste diagrama são demonstradas quais informações precisam ser guardadas e de quais objetos.

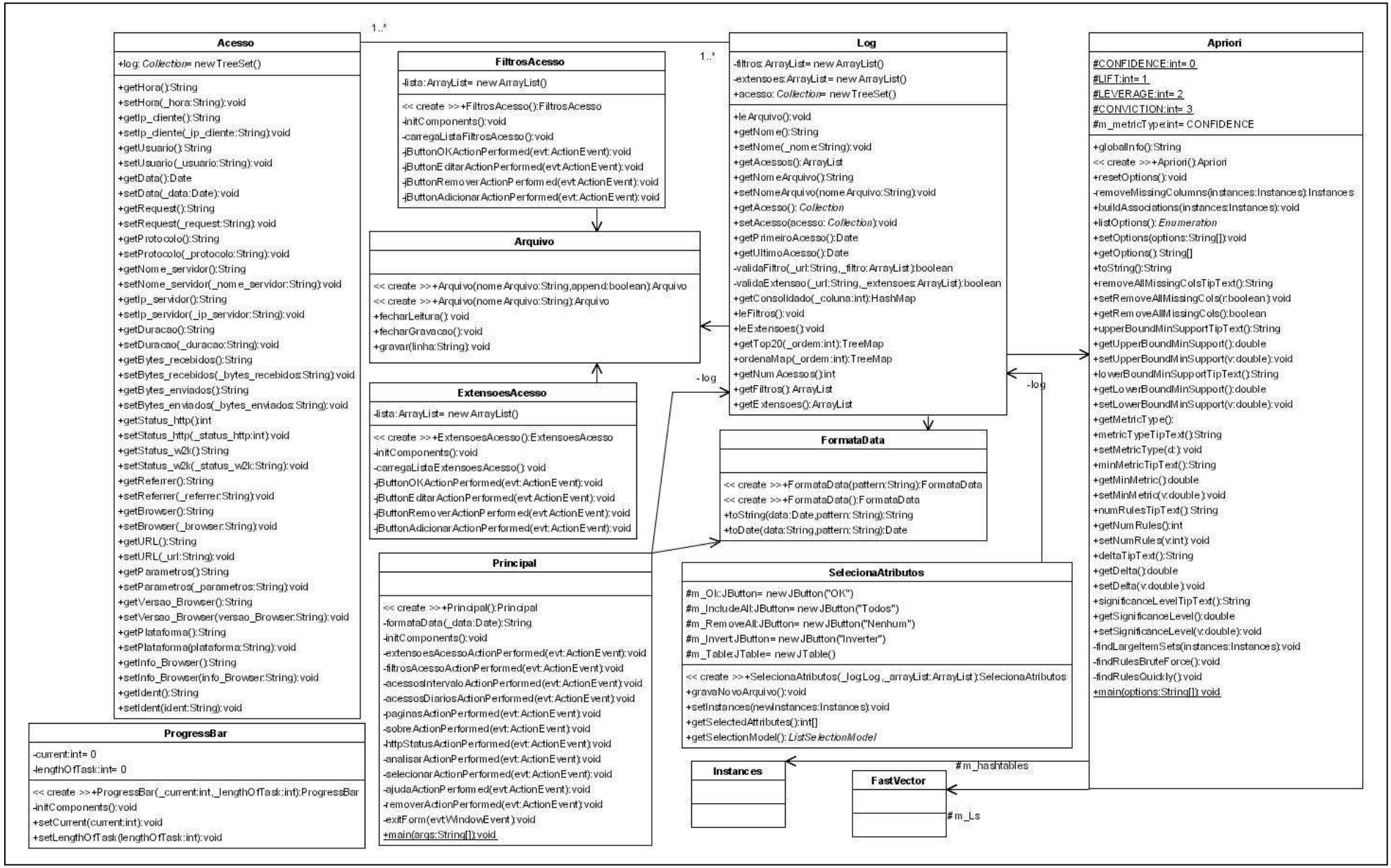


Figura 14 - Diagrama de Classes

5.2.3 DIAGRAMA DE ATIVIDADES.

Na figura 11 é apresentado o diagrama de atividades do sistema. Os diagramas de atividades mostram o fluxo entre atividades (ações não-atômicas). Ex: fluxos de processos, fluxos de eventos, detalhamento de operações. Normalmente é feito um diagrama de atividades para cada caso de uso.

Foi escolhido o diagrama de atividades por ser o que melhor representa o processo e por conter apenas um caso de uso que precisava ser explicado seu fluxo. A ferramenta utilizada para fazer o diagrama foi o Poseidon.

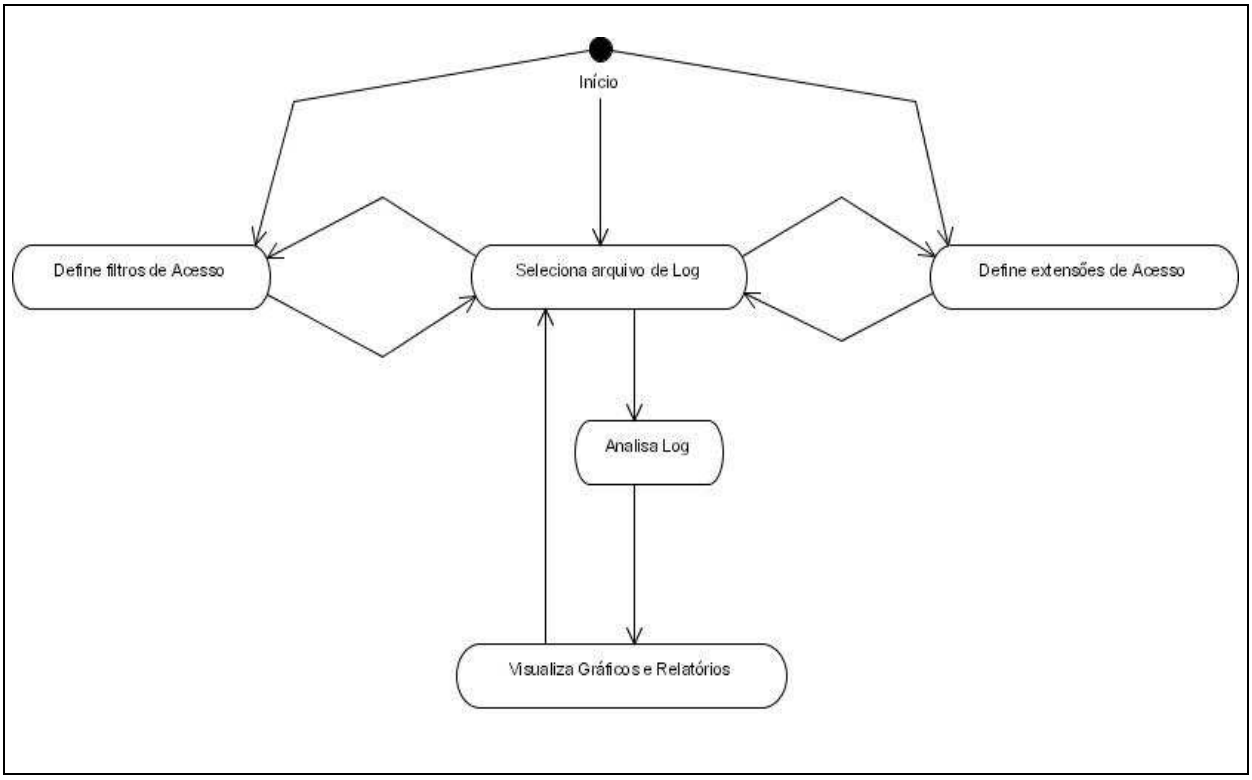


Figura 15 - Diagrama de atividades.

5.3 IMPLEMENTAÇÃO

A seção 2 descreve os conceitos básicos usados durante o desenvolvimento desse trabalho trazendo consigo exemplos de trabalhos correlatos, bem como definições de transação, *log* e seqüência de cliques.

Na seção 3 *Data Mining* é definida sob a óptica de vários autores, trazendo uma classificação para as principais técnicas, até chegar nas Regras de Associação que foi utilizada

para o desenvolvimento do sistema. É nessa seção também que o algoritmo Apriori é explicado passo a passo.

Como o objetivo principal do trabalho é fazer *Data Mining* em *log* de servidores *web*, mais especificamente os mais populares do mercado (IIS e Apache), a seção 4 trata das características principais desses *web servers*, principalmente no que diz respeito às configurações necessárias para se obter o arquivo de *log* no padrão NCSA que é o padrão em que o sistema se baseia.

A seção 5 detalha minuciosamente o arquivo de *log*, comentando seus campos e o comparando entre os padrões mais usados.

5.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

A técnica de *Data Mining* usada neste trabalho para extração de conhecimento foi à chamada Regra de Associação. Para esta técnica, os algoritmos mais conhecidos são o Tertius e Apriori (e suas variantes). Durante essa parte do estudo cogitou-se a possibilidade do uso de *Text Mining* já que os dados estariam armazenados em arquivos texto no formato ASCII. Depois de demandado um tempo para estudado o assunto percebeu-se que não seria o ideal, pois a área de *Text Mining* trata de reconhecimento de padrão em linguagem natural – *Natural Language Processing* (NLP).

O projeto WEKA, conforme Weka (2004), é um sistema criado para apoio ao aprendizado de algoritmos de *Data Mining*. Este sistema está todo implementado em Java e traz consigo diversos algoritmos usados para extração do conhecimento em Banco de dados ou mesmo arquivos texto com formato. Ele foi desenvolvido na Universidade de Waikato na Nova Zelândia. Para fazer uso do sistema fez-se necessário um estudo aprofundado do seu funcionamento, pois ele foi a principal ferramenta de apoio neste trabalho.

5.3.1.1 WEKA

WEKA apresenta um ambiente gráfico para facilitar o aprendizado e testes dos algoritmos já implementados. Esse ambiente é chamado de *WEKA GUI Chooser*, conforme pode ser visto na Figura 16 e subdivide-se em:

a) *Simple CLI* – Provê uma interface simples de linha de comando que permite a execução direta de comandos WEKA;

b) *WEKA Knowledge Explorer* – Um ambiente para explorar dados com WEKA. Este trabalho concentrou-se na utilização do algoritmo apriori presente neste módulo;
c) *Experimenter* – Um ambiente para fazer experiências e conduzir os testes estatísticos entre os esquemas de aprendizado.

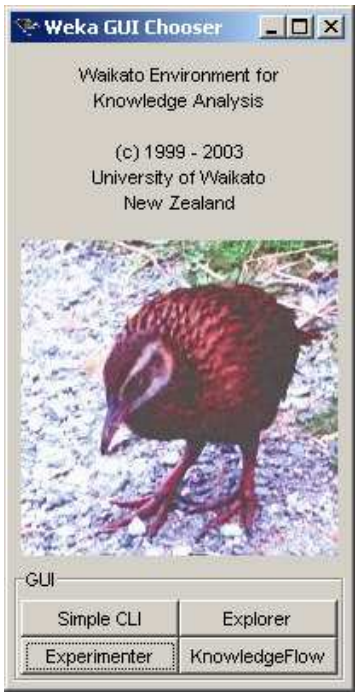


Figura 16 – Weka GUI Chooser

5.3.1.2 WEKA Knowledge Explorer

Logo no topo da janela, logo abaixo do título é mostrada uma linha com abas (orelhas) para seleção das opções disponíveis no ambiente Weka conforme mostra a figura 17. Inicialmente somente a primeira aba aparece ativa, pois é necessário abrir um arquivo com *Dataset* para começar a explorar os dados. As abas disponíveis são:

a) Preprocess – Seleciona e modifica os dados que o sistema irá atuar a partir dos filtros disponíveis;
b) Classify – Treina e testa esquemas de aprendizado que usam o método de Classificação e regressão, descritos na seção 2.1.1.1 deste trabalho;
c) Cluster – Aprendizado com o método de Clustering descrito na seção 2.1.2.1.1 deste trabalho;
d) Associate – Aprendizado utilizando a técnica de Regras de Associação;
e) Select Attributes – Seleciona os atributos mais relevantes nos dados;
f) Visualize – mostra um gráfico interativo de duas dimensões dos dados.

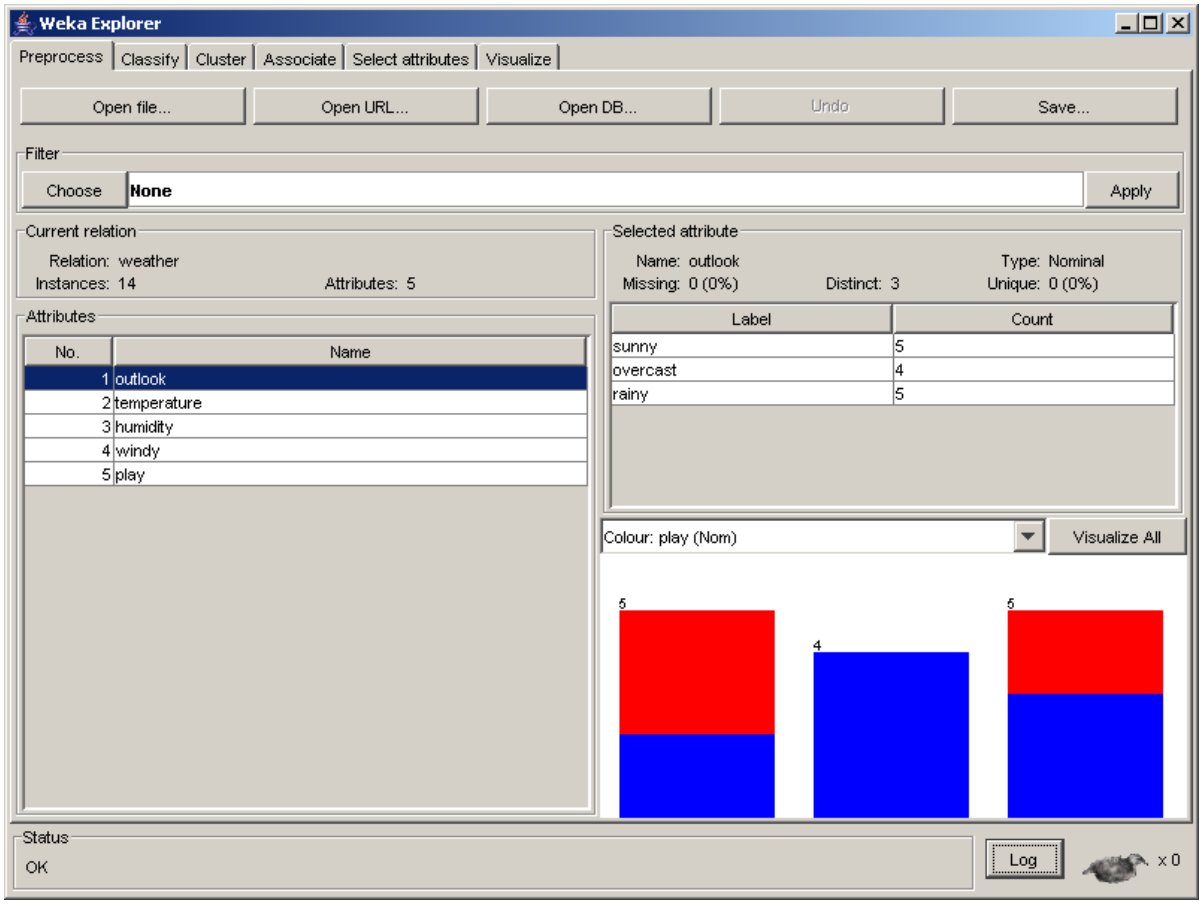


Figura 17 - Weka Knowledge Explorer

Na parte inferior da tela é exibido um quadro de *status* com um botão Log que permite visualizar o histórico de ações executadas. Além disso, ainda são mostrados vários outros objetos, dependendo da tela escolhida, tais como:

a) Quadro onde são exibidos os resultados do algoritmo aplicado;
b) Seletor de algoritmo e filtro onde é possível alternar entre os algoritmos disponíveis, dependendo da técnica selecionada;
c) Gráfico 2D com resultados de Pré-Processamento;
d) Quadro para visualização, adição, remoção de atributos.

5.3.1.3 O formato ARFF

WEKA espera que os dados estejam no formato ARFF. Este formato de um arquivo nada mais é do que um texto (ASCII) delimitado por vírgulas ou ponto-e-vírgulas ou ainda espaços. Na verdade o padrão sugerido é CSV que usa vírgulas como delimitadores, porém WEKA aprimorou a técnica para reconhecer arquivos com estes outros delimitadores. Uma situação comum é que os dados estejam armazenados em planilhas ou em um banco de dados,

entretanto WEKA espera esses dados em um arquivo ARFF. Isto é necessário para saber o tipo de informação de cada atributo que não pode ser deduzido automaticamente dos valores desses atributos.

Antes de aplicar qualquer algoritmo, os dados devem ser convertidos para o formato ARFF. Este protótipo prevê uma interface para facilitar a conversão dos dados, gerando automaticamente os atributos. Isto é possível somente por se conhecer a estrutura dos dados de entrada no sistema, pois eles vêm no formato NCSA. Dessa forma a conversão é de um arquivo no formato NCSA para um arquivo ARFF.

A estrutura do arquivo ARFF pode ser vista no exemplo da Figura 18 que simula a decisão de um atleta em jogar ou não jogar Tênis baseado na previsão do Tempo. O mesmo exemplo foi usado para explicar as Regras de Associação na Seção 2.1.2.2.8 deste trabalho. O conjunto de todas as informações é chamado de *dataset*. Por essa razão dizemos *Dataset* com as informações da Previsão do Tempo versus Jogar Tênis.

A maioria dos bancos de dados e editores de planilhas existentes no mercado permite salvar arquivos no formato CSV – como uma lista de registros onde os itens são separados por vírgulas. A conversão para o formato CSV não é necessária, pois o padrão NCSA já vem num padrão reconhecido pelo algoritmo WEKA.

As primeiras linhas do arquivo descrevem os atributos e o nome do *Dataset* que estão sendo manuseados. Para isso o formato ARFF pede que algumas *tags* sejam adicionadas logo no início do arquivo. Essas informações que já são gravadas automaticamente pelo protótipo no início do arquivo constituem o cabeçalho do mesmo. As *tags* são precedidas de um símbolo de @. As possíveis *tags* conhecidas e esperadas para um arquivo no formato ARFF são:

a) *@relation* <nome do dataset> – Define um nome para o Dataset;

b) *@attribute* <nome do atributo> <tipo> - Define o nome e o tipo de dados do atributo;

c) *@data* – Define a posição onde começam as instâncias de dados.

Cada coluna do arquivo definida por um delimitador é um conjunto de atributos. É necessário que todas as colunas do arquivo estejam previamente nomeadas com uma *tag @attribute*. Na ausência de um valor o símbolo de ? deve ser inserido no local. Dessa forma o algoritmo irá considerar com um valor perdido e irá contar o número de valores perdidos, dependendo do algoritmo aplicado. O símbolo % é usado para comentar uma linha.

Quando se sabe com antecedência os valores que irão ocorrer em um atributo no *Dataset*, ou seja, os possíveis valores que um determinado atributo pode conter, podem-se descrevê-los entre chaves ao lado do nome do atributo no lugar do tipo. Weka chama esse tipo de atributo de nominal. Isso é muito comum no caso de valores *booleanos true* e *false*.

Os tipos de dados disponíveis para os valores de atributos no ambiente de aprendizado WEKA são:

a) STRING;

b) DATE;

c) INTEGER;

d) NUMERIC;

e) *REAL*.

A Figura 17 mostra um exemplo de arquivo ARFF com cabeçalho nominal. Nesse exemplo pode-se ver o resultado do *dataset* do tempo após aplicar o filtro que remove os atributos 1 e 2.

```
@relation weather-weka.filters.AttributeFilter-R1_2

@attribute humidity real
@attribute windy {TRUE,FALSE}
@attribute play {yes,no}

@data

85,FALSE,no
90,TRUE,no
...
```

Figura 18 - Exemplo de cabeçalho de arquivo ARFF

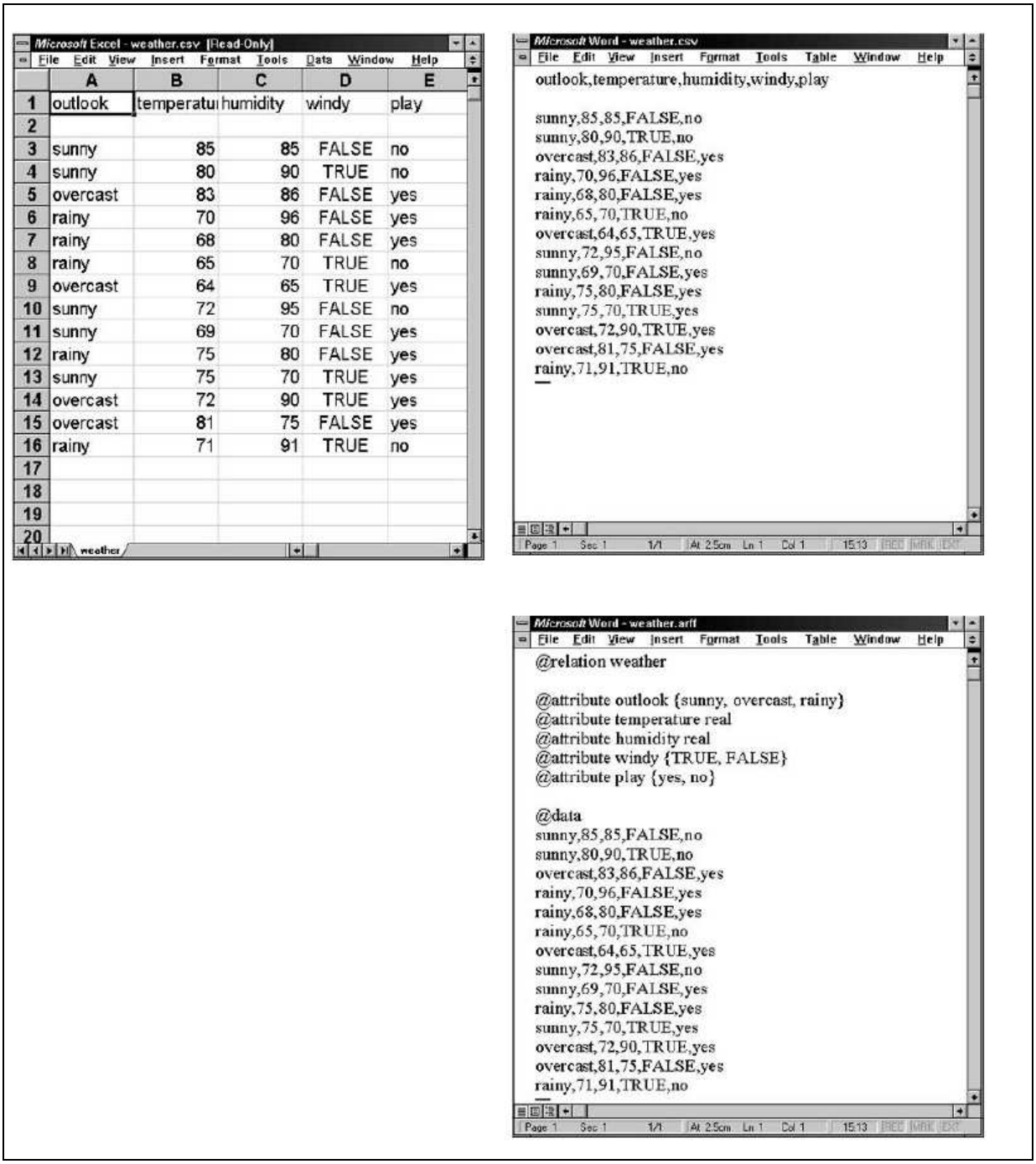


Figura 19 - Exemplo de *DataSet* da Previsão do Tempo X Jogar tênis

5.3.1.4 Trabalhando com Filtros

WEKA traz também um módulo especialmente para trabalhar com filtros. O módulo de pré-processamento permite definir o filtro que será aplicado nos dados. Esta tarefa é importante, pois dependendo do tipo de informação que está sendo minerada, alguns atributos de dados podem ser descartados. WEKA faz isso por meio de uma janela chamada de *Generic Object Editor*. O mesmo tipo de janela também é usada em vários pontos do sistema, tal

como, nas técnicas de Classificação e *Clustering*. Esta janela foi acrescentada ao protótipo e pode ser vista na seção 5.3.2 que trata da operacionalidade da implementação.

No ambiente WEKA, quando um filtro é selecionado, ele não é imediatamente aplicado. Um filtro pode ser selecionado clicando-se no botão <*Choose*> e aplicado clicando-se no botão <*Apply*>. WEKA possui diversos filtros, mas o que se tem interesse em destacar nesse momento é o filtro Remove.

O filtro *Remove* permite eliminar um atributo do *Dataset* e depois salvá-lo com outro nome. Esse procedimento restringe a área de atuação do algoritmo, pois elimina valores irrelevantes. O mesmo resultado pode ser obtido executando-se a classe *AttributeFilter* diretamente por linha de comando chamado o *Java Virtual Machine*.

5.3.1.5 JFreeChart

JFreeChart, conforme David (2004), é uma biblioteca gratuita para construção de gráficos na plataforma Java. Ele foi desenvolvido para ser usado em aplicações, *applets, servlets e JSP*. *JFreeChart* e é distribuído com o seu código fonte sob os termos da GNU.

JFreeChart pode gerar gráficos de pizza, gráficos de barra (regular ou 3D), gráficos de linha, scatter plots, gráficos de séries temporais (com movimento), gráficos de Gantt e muitos outros. Permite efeitos de zoom e exportar para formatos como JPG, PNG, PDF, SVG e mapeamento para HTML.

JFreeChart foi desenvolvido totalmente em Java e roda em qualquer implementação feita na plataforma Java 2 (JDK 1.2.2 ou superior). Ele pode ser obtido pelo *site* (http://www.jfree.org/jfreechart/).

5.3.1.6 Minerando arquivos de *log*

A implementação para o algoritmo Apriori usada nesse trabalho foi obtida do projeto WEKA, citado na seção 5.3.1.1. Seu funcionamento foi o principal objeto de estudo deste trabalho e os resultados serão apresentados a seguir.

A primeira tarefa de pré-processamento foi classificar os atributos e seus respectivos tipos de dados identificando também as sessões dos usuários. De acordo com a W3C, um único indivíduo que está tendo acesso aos dados de um ou mais servidores *web* através de um

browser e uma sessão do usuário é a seqüência de cliques (*click-stream*) da página visitada por um único usuário de um *web site* em particular por um certo período de tempo.

Consegue-se identificar um usuário nos arquivos de *log* pelo registro do mesmo endereço IP durante um intervalo de tempo. A identificação de um usuário no arquivo de *log* é o ponto de partida para uma série de associações que podem ser feitas até chegar na identificação de padrões pela técnica de Regras de Associação utilizando o algoritmo Apriori.

Para entender melhor esse mecanismo, deve-se conhecer os conceitos envolvidos na mineração de dados, no algoritmo Apriori e nas Regras de Associação descritos no capítulo 3. Faremos uma outra abordagem ao assunto para poder exemplificar o tema através de um estudo de caso.

Para obter as regras de associação o caso analisado foi do *log* dos *sites* do domínio .inf da FURB hospedados num servidor Apache que tem uma periodicidade de aproximadamente uma semana cada *log*. Esse intervalo não é exato, pois é possível dividir o arquivo em tamanhos fixos ou por data conforme mostrado na sessão 3.1.6.

Os dados adquiridos para montar os gráficos de Erro HTTP, Acesso e Páginas mais acessadas foi através de operações básicas de estatísticas que ajudam a obter uma visão preliminar sobre os dados. Nessa etapa é possível também utilizar as chamada análise de dados exploratória disponíveis em diversas ferramentas de mineração de dados.

Conforme descrito na seção 5.3.1.3, atributo é o nome dado ao campo que contém os valores das instâncias de um *dataset*. Um *dataset*, por sua vez, é o conjunto de todos os dados que serão analisados conforme mencionado na seção 5.3.1.3. Em um banco de dados, por exemplo, os valores a serem analisados estariam armazenados em uma tabela, onde, os campos representariam os atributos e os dados o *dataset*. No caso de arquivos CSV, como o *log* dos servidores *web*, os atributos correspondem ao nome das colunas e o *dataset* o conjunto de dados existentes neste arquivo.

Cada linha do arquivo ou registro de uma tabela, no caso de banco de dados, corresponde a um acesso ao servidor para buscar um item. Este registro corresponde a uma instância e o *itenset* é a combinação de atributo com um valor conforme mencionado na seção 2.1.2.2.9. Podemos então ter itensets de um, dois, três ou n atributos representando o valor daqueles atributos.

Antes de encontrar as regras é necessário entender o que é suporte (*support*) e confiança (*confidence*), também chamados de cobertura (*coverage*) e precisão (*accuracy*) respectivamente mencionados na seção 2.1.2.2.8.

Podemos dizer que o suporte é o número de vezes que um valor acontece ou se repete no *dataset*. Referindo-se a Regras de Associação, o suporte também se aplica ao conjunto de valores combinados, ou seja, o número de vezes que determinados atributos apareceram associados, ou ainda, o suporte de um determinado *itenset*.

Segundo Witten & Frank (2000), a confiança representa o suporte dividido pelo número de instâncias para o qual as condições anteriores ao símbolo => aconteceram. Interpretado como uma fração, a confiança representa a proporção de instâncias que a regra está correta. Por essa razão recebe o nome de confiança, pois indica pelo percentual de ocorrências de um atributo com um determinado valor o quanto se pode confiar que o mesmo valor se repetirá no futuro.

Contar o número de vezes que os atributos aconteceram com um determinado valor não é o objetivo principal, mas é indispensável para chegar na formação das regras. O objetivo principal é encontrar o suporte e confiança de uma determinada regra. É isso que nos dá a informação de associação de valores.

E o que é uma regra? Uma regra é formada de uma inferência a partir de atributos verdadeiros. Entende-se por verdadeiro, nesse contexto o fato do atributo ter ocorrido com um determinado valor no *dataset*. Podemos dessa forma fazer uma analogia da regra com um silogismo, onde cada premissa é constituída de um ou mais *itensets* e leva a uma conclusão. Normalmente é usado o símbolo => para preceder uma conclusão. Na Figura 20 podemos ver o resultado das 10 melhores regras encontradas após aplicar o algoritmo Apriori no exemplo do *dataset* da previsão do tempo versus jogar Tênis.

```
Apriori
=======

Minimum support: 0.2
Minimum confidence: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 47
Size of set of large itemsets L(3): 39
Size of set of large itemsets L(4): 6

Best rules found:

1 . humidity=normal windy=FALSE 4 ==> play=yes 4 (1)
2 . temperature=cool 4 ==> humidity=normal 4 (1)
3 . outlook=overcast 4 ==> play=yes 4 (1)
4 . temperature=cool play=yes 3 ==> humidity=normal 3 (1)
5 . outlook=rainy windy=FALSE 3 ==> play=yes 3 (1)
6 . outlook=rainy play=yes 3 ==> windy=FALSE 3 (1)
7 . outlook=sunny humidity=high 3 ==> play=no 3 (1)
8 . outlook=sunny play=no 3 ==> humidity=high 3 (1)
9 . temperature=cool windy=FALSE 2 ==> humidity=normal play=yes 2 (1)
10. temperature=cool humidity=normal windy=FALSE 2 ==> play=yes 2 (1)
```

Fonte: Witten & Frank (2000, p.295).

Figura 20 – Exemplo de regras aplicadas ao *dataset* do Tempo X Jogar Tênis

Todas as regras de associação formadas pelo algoritmo Apriori são verdadeiras, pois representam ocorrência de valores no *dataset*. Se essas regras são válidas ou relevantes é uma questão a ser analisada e é nesse ponto que entra o papel do analista, verificando a utilidade da regra.

As regras são montadas combinando todas as possibilidades de associação entre os atributos. Encontrar as melhores regras significa selecionar as regras com a maior confiança possível. No caso desse trabalho, as regras podem ser úteis para encontrar padrões de comportamento de usuários no acesso a um *site* da *web*.

5.3.1.7 Parâmetros do algoritmo Apriori

O algoritmo Apriori implementado no ambiente WEKA está localizado no pacote *weka.associations.Apriori*. Ele pode ser executado diretamente por linha de comando ou de dentro do protótipo onde recebeu alguns tratamentos especiais os quais comentaremos a seguir.

Antes de executar o algoritmo Apriori no *dataset* a fim de buscar as melhores regras é necessário saber informar seus parâmetros. No algoritmo Apriori implementado pelos desenvolvedores do ambiente WEKA, quando o algoritmo é executado sem nenhum parâmetro informado esses parâmetros são exibidos na tela. A Tabela 10 mostra os parâmetros do algoritmo Apriori.

Parâmetros do algoritmo Apriori

Opção	***Função***
-t <arquivo>	Especifica o arquivo que será treinado
-N <nº de regras>	Especifica o número de regras desejado. (*default* = 10)
-C <confiança>	Especifica a confiança ou precisão mínima exigida. (*default* = 0.9 = 90%)
-D <suporte mínimo>	Delta para decréscimo do suporte até chegar ao mínimo
-M <decréscimo do suporte>	A taxa de suporte mínima desejada. (*default* = 0.1 = 10%)
-T <nº corresponde ao tipo>	Tipo de medida a ser alcançada. (*default* = confidence) <0=confidence | 1=lift | 2=leverage | 3=Conviction>
-U	A taxa de suporte máxima desejada. (*default* = 1.0 = 100%)
-S <nível de significância>	Se usado, regras são testadas pela significância do nível que é dada. Isto deixa o algoritmo lento. (*default* = nenhum)
-I	Indica se o conjunto de *itensets* encontrados são também saída. (*default* = não)
-R	Remove colunas sem valores informados. (*default* = não)
-V	Mostra o progresso interativamente. (*default* = não)

Tabela 10 - Parâmetros do algoritmo Apriori

5.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Para fins didáticos o gerente da *web* ou analista será doravante chamado simplesmente de usuário. Este tópico tem o objetivo de apresentar o sistema com suas principais telas explicando suas funcionalidades. O estudo de caso escolhido para demonstrar a operacionalidade da implementação é o *site* da FURB.

A Figura 19 é a primeira e a principal tela do sistema. Nela o analista ou gerente da *web* pode selecionar o arquivo de *log* no Formato Comum do NCSA e analisá-lo para obter as informações que precisa.

O sistema, como se pode observar, apresenta uma interface de fácil entendimento e usabilidade trazendo as opções mais usadas na forma de botões diretamente na tela principal. A qualquer momento o usuário poderá consultar a tela de ajuda que será mais detalhada adiante.

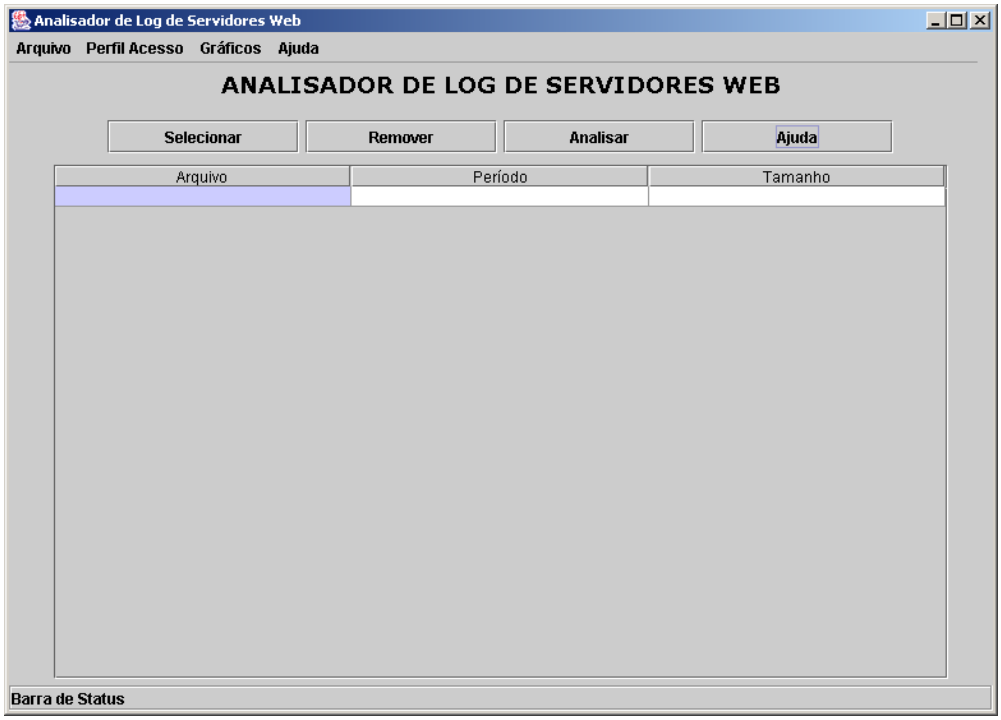


Figura 20- Tela principal do sistema

Para isso, o usuário deve clicar no botão selecionar ou no menu principal escolher esta opção. Em seguida uma tela para escolha do arquivo de *log* é exibida permitindo que o arquivo seja selecionado de uma pasta local ou de um caminho da rede. Por exemplo, numa rede local o diretório do servidor que armazena o arquivo poderia estar mapeado na máquina onde o sistema está rodando permitindo dessa forma buscar o arquivo nesse servidor. A Figura 20 mostra essa tela.

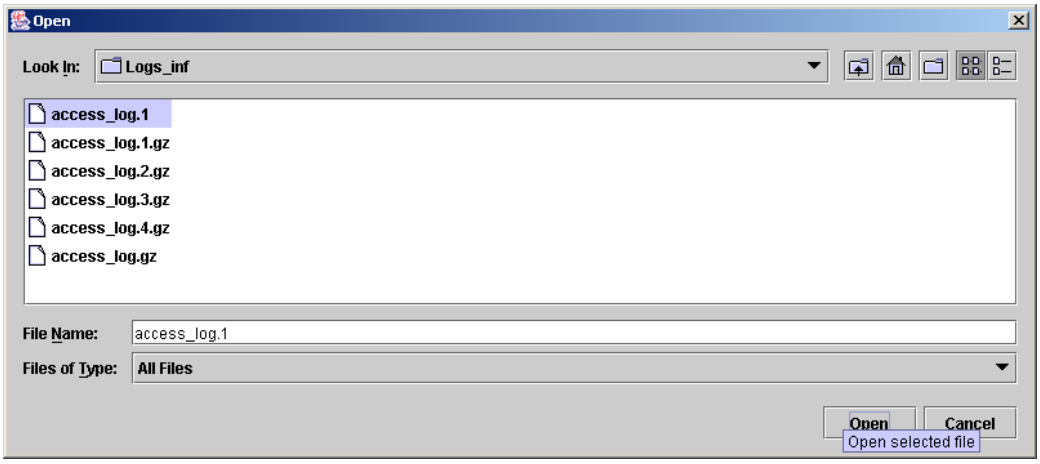


Figura 20 - Tela de seleção do arquivo de *log*.

Nesse momento o sistema apresentará o tamanho do arquivo na tela principal, bem como sua localização. O usuário deverá então mandar o sistema analisar o arquivo. Isto pode ser feito clicando no botão analisar ou mesmo pelo menu de opções.

Caso o arquivo seja muito grande o sistema poderá levar alguns minutos para fazer a análise, pois é nesse momento que os dados são carregados para memória. O tamanho do arquivo pode ter sido previamente especificado pelo analista sendo definido por periodicidade ou tamanho máximo nas configurações do servidor *web*.

Como pode ser observado na Figura 21, nesse momento o sistema já pode informar o intervalo que foi definido sendo apontado na coluna período desta tela.

Figura 17 - Tela principal do sistema após a análise do arquivo.

Na seqüência o usuário tem ao seu dispor alguns gráficos que o ajudam a ter um melhor entendimento das transações contidas no arquivo de *log*. A maioria dos gráficos é obtida por técnicas estatísticas explicadas no capitulo *Data Mining*.

O primeiro gráfico, mostrado na Figura 18, é um gráfico de status HTTP que indica o número de páginas que foram exibidas sem erro para o usuário, bem como o número de páginas que o mesmo foi redirecionado ou ainda que obteve um erro do tipo “A página não pode ser exibida”.

Antes de visualizar qualquer gráfico é necessário selecionar a analisar o arquivo, conforme mostrado nos itens acima, caso contrário, uma mensagem será exibida dizendo: “Você deve primeiro selecionar um arquivo de *log* e analisar”.

O status de cada transação é analisado conforme a seguinte numeração:

- 200 a 299 – significa que a transação foi bem sucedida;
- 300 a 399 – significa que ocorreu um redirecionamento;
- 400 a 599 – significa que ocorreu algum tipo de erro;

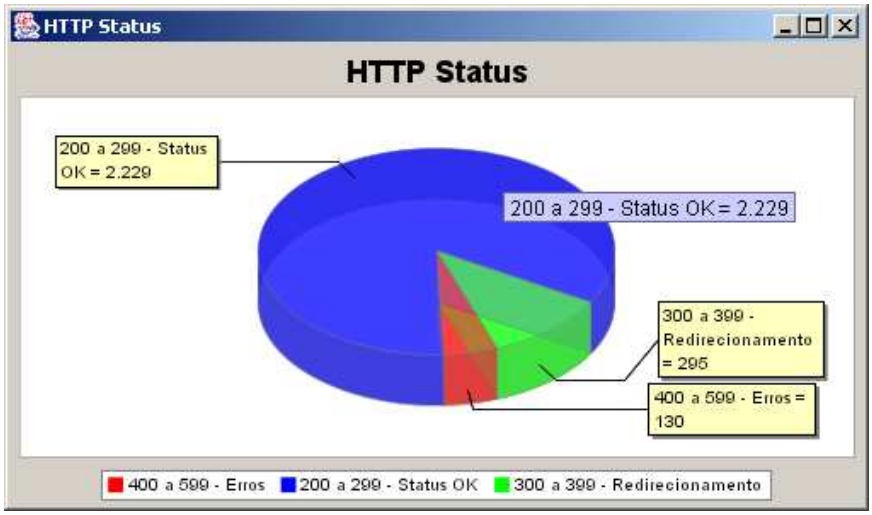


Figura 18 - Página de status HTTP

Como pode ser observado no gráfico de exemplo, temos um número de 170.218 transações bem sucedidas, 90.393 redirecionamentos e 64.019 transações com algum tipo de erro. Isto considerando o intervalo do arquivo de *log* e também todos os registros lidos do arquivo. Um único clique em uma página pode gerar vários registros no arquivo de *log*, dependendo do número de objetos envolvidos.

O sistema traz ainda uma forma de filtrar o conteúdo, buscando por páginas específicas e / ou tipos de arquivo. A Figura 19 mostra a página de Filtros de acesso localizada no menu Perfil Acesso do sistema.

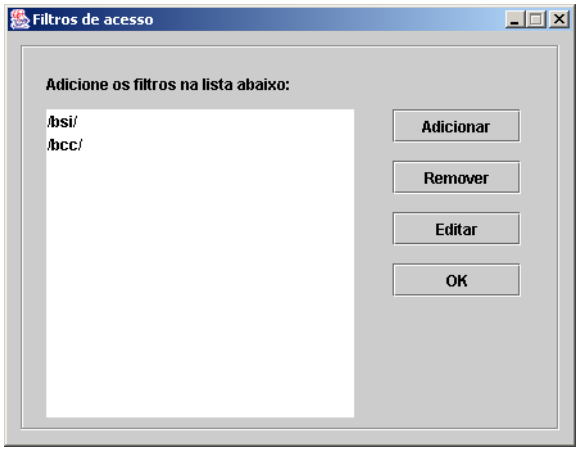


Figura 19 – Tela de Filtros de acesso

Além das palavras chaves que podem ser cadastradas nos Filtros de acesso o usuário pode escolher e combinar com os filtros as extensões dos arquivos a serem analisados. A Figura 20, mostra a tela de Extensões de acesso que ilustra essa funcionalidade.

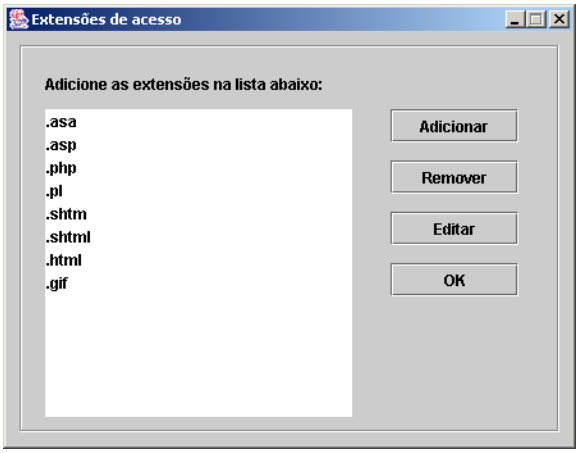


Figura 20 - Tela de Extensões de Acesso

O trabalho de pré-processamento exigiu que outras opções fossem implementadas, tais como a seleção de atributos em um dataset. Selecionar um atributo significa escolher os valores em que o algoritmo será aplicado. Nessa etapa algumas janelas existentes no ambiente WEKA foram aproveitadas e ajustadas para serem utilizadas nesse protótipo. A Figura 21, mostra a tela que seleciona os atributos do *dataset* que serão usados para aplicar o algoritmo Apriori. Apesar de ser possível editar manualmente um arquivo ARFF, conforme mostrado anteriormente, as telas seguintes facilitam a tarefa de pré-processamento dos dados.

Ao confirmar a seleção de atributos um novo arquivo é criado na mesma pasta onde estava o arquivo de *log* selecionado com o mesmo nome acrescentado da extensão ARFF.

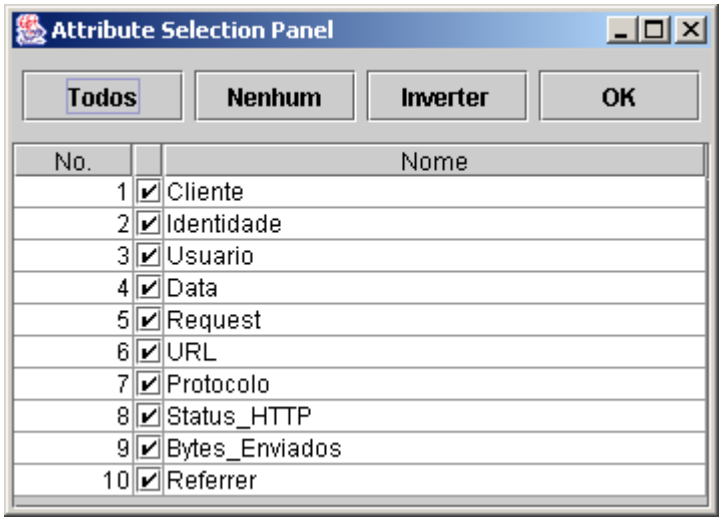


Figura 21 – Tela seleção de atributos

No menu Gráficos, é possível visualizar as páginas mais acessadas no intervalo de um arquivo de *log* conforme mostra a Figura 22. É possível trabalhar somente com um arquivo de cada vez. Isso de certa forma facilita o entendimento das informações por se trabalhar com uma seleção restrita. No caso desse protótipo é possível ver somente as 20 páginas mais acessadas no intervalo do arquivo.

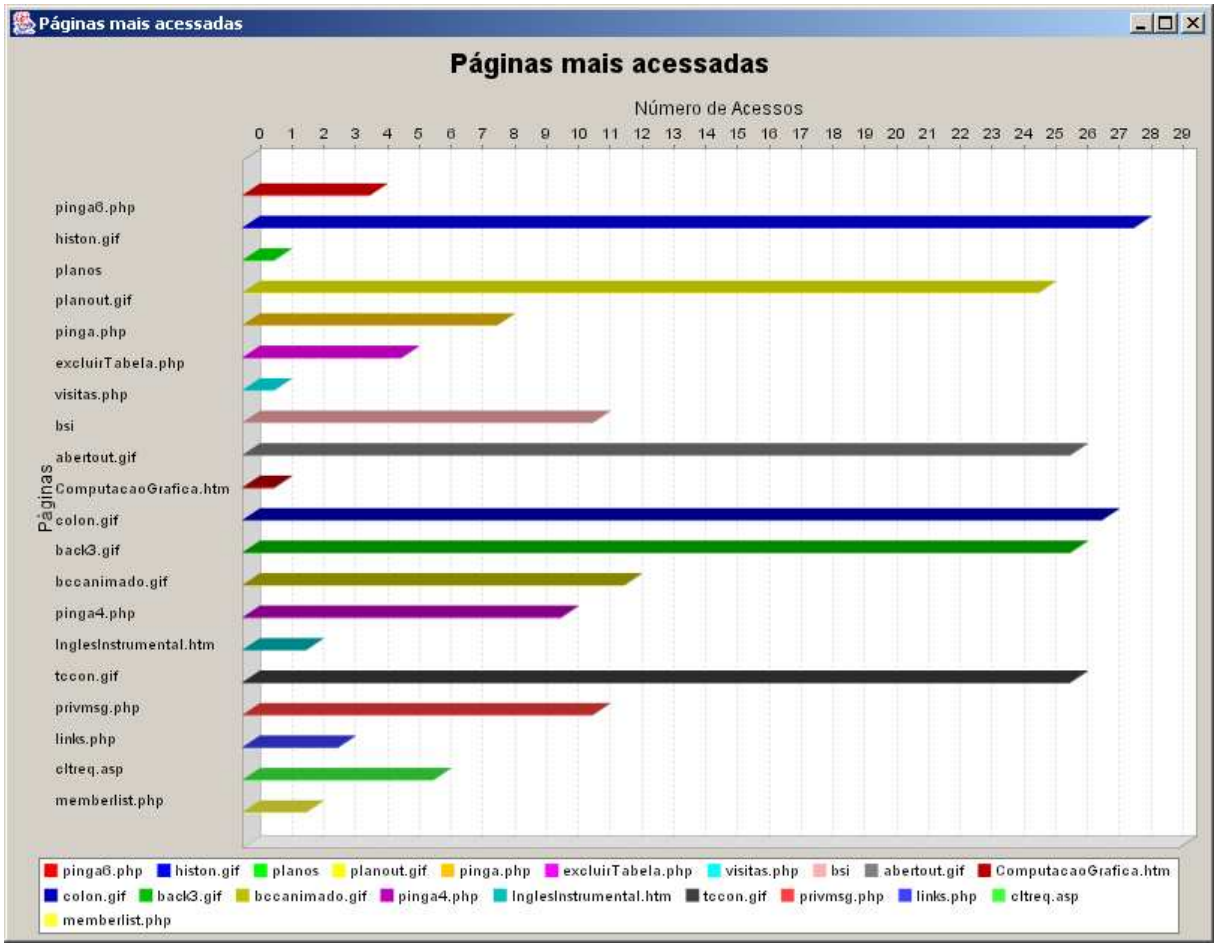


Figura 22 – As 20 páginas mais acessadas no intervalo do *log*

O gráfico de acesso por período mostra o período em que o tráfego no servidor foi mais intenso. Devemos considerar nesse caso o intervalo do *log* a ser analisado. Esse gráfico pode ser visualizado pelo menu Gráficos. Um exemplo pode ser visto na Figura 23.

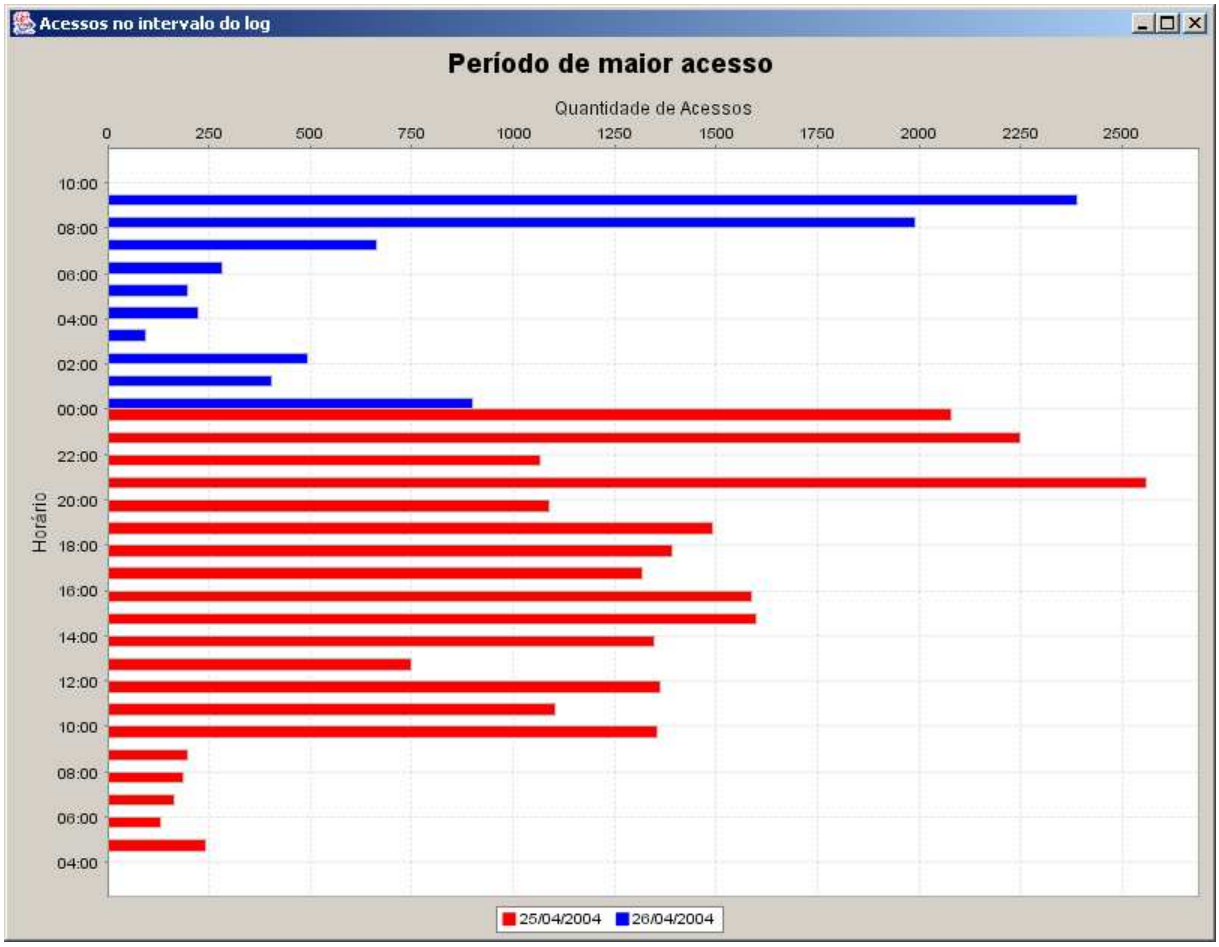


Figura 23 – Gráfico do Período de maior acesso

5.4 RESULTADOS E DISCUSSÃO

Os resultados alcançados até a conclusão desse trabalho serão apresentados nesse tópico sob a forma de relatórios obtidos pela execução do algoritmo Apriori em cima de um fragmento do *log* do domínio .inf da FURB gerado pelo Apache.

Foi usado apenas um fragmento do arquivo de *log*, pois os arquivos originais estavam com muitas transações totalizando mais de 34 MB de dados. A performance do algoritmo foi a primeira constatação. O algoritmo foi muito lento em sua execução a partir do momento em que o *log* ultrapassasse o tamanho de 1 MB de dados, o que representa um volume muito pequeno de dados para um *log* de servidores *web*.

Além disso, em alguns casos, percebemos que o algoritmo não trouxe as melhores regras como se propunha fazer e até fez em testes preliminares. Isso prejudicou muito o desenvolvimento do protótipo, pois esperávamos o resultado desse algoritmo para dar

continuidade nos gráficos que dependiam de regras de associação, como por exemplo, os gráficos do item **a** dos objetivos específicos apresentados na página 8 desse trabalho.

Outro complicador foi o fato de não conseguirmos obter os números do algoritmo, individualmente, para montar os gráficos. Ou seja, a saída do algoritmo padrão é um retorno direto das regras já processadas não disponibilizando métodos para captar cada resultado de cada regra separadamente. Pelo menos, pela pesquisa que fizemos à classe Apriori esse recurso não foi possível.

Mesmo assim, conseguimos inferir algumas informações a partir das regras que obtivemos. As possibilidades são muitas, pela combinação dos atributos que podem ser selecionados e filtrados, como mostrado no 5.3.2, operacionalidade da implementação.

Para este estudo de caso, foi separado um fragmento de *log* contendo 30916 transações. Podemos ver as primeiras linhas desse *log* no quadro abaixo:

```
…
64.68.82.159 - - [25/Apr/2004:04:20:44 -0300] "GET /~jomi/xml/exemplos/bib/soma.xsl HTTP/1.0" 200 287
64.68.82.55 - - [25/Apr/2004:04:21:36 -0300] "GET /~jomi/xml/exemplos/reservas/?C=M;O=A HTTP/1.0" 200 1378
64.68.82.159 - - [25/Apr/2004:04:21:37 -0300] "GET /~jomi/xml/exemplos/oi/?C=M;O=A HTTP/1.0" 200 983
201.4.239.102 - - [25/Apr/2004:04:21:38 -0300] "GET /~maw/img/pacer.gif HTTP/1.1" 404 672
201.4.239.102 - - [25/Apr/2004:04:21:43 -0300] "GET /~maw/eletbasica/maw_style.css HTTP/1.1" 200 1366
201.4.239.102 - - [25/Apr/2004:04:21:44 -0300] "GET /~maw/eletbasica/img/pacer.gif HTTP/1.1" 404 672
201.4.239.102 - - [25/Apr/2004:04:21:41 -0300] "GET /~maw/eletbasica/index.htm HTTP/1.1" 200 16975
201.4.239.102 - - [25/Apr/2004:04:21:45 -0300] "GET /~maw/eletbasica/img/spacer.gif HTTP/1.1" 200 43
201.4.239.102 - - [25/Apr/2004:04:21:45 -0300] "GET /~maw/eletbasica/img/index_r1_c1.gif HTTP/1.1" 200 439
201.4.239.102 - - [25/Apr/2004:04:21:45 -0300] "GET /~maw/eletbasica/img/index_r2_c1.gif HTTP/1.1" 200 954
201.4.239.102 - - [25/Apr/2004:04:21:45 -0300] "GET /~maw/eletbasica/img/index_r2_c3.gif HTTP/1.1" 200 826
201.4.239.102 - - [25/Apr/2004:04:21:46 -0300] "GET /~maw/eletbasica/img/index_r2_c4.gif HTTP/1.1" 200 116
201.4.239.102 - - [25/Apr/2004:04:21:46 -0300] "GET /~maw/eletbasica/img/index_r2_c5.gif HTTP/1.1" 200 818
201.4.239.102 - - [25/Apr/2004:04:21:47 -0300] "GET /~maw/eletbasica/img/index_r2_c6.gif HTTP/1.1" 200 115
201.4.239.102 - - [25/Apr/2004:04:21:47 -0300] "GET /~maw/eletbasica/img/index_r2_c7.gif HTTP/1.1" 200 1613
…
```

No exemplo, percebemos acessos na data de 25/04/2004 às páginas de Programação Orientada a Objetos (POO), laboratório LCI, página do prof. Jomi entre outras.

O algoritmo Apriori foi aplicado nos dados com os seguintes parâmetros:

```
-N 10 -C 0.8 -D 0.2 -U 1.0 -M 0.1
```

Uma explicação de cada parâmetro disponível no algoritmo implementado pelo projeto WEKA pode ser vista no item 6.3.1.7 deste trabalho.

Selecionando os atributos:

@attribute IP_Cliente string
@attribute Data string
@attribute Hora string
@attribute Request string
@attribute URL string
@attribute Status_HTTP string
@attribute Bytes_Enviados string

Conseguimos os seguinte resultados:

Apriori
=======

Minimum support: 0.25
Minimum metric <confidence>: 0.9
Number of cycles performed: 15

Generated sets of large itemsets:

Size of set of large itemsets L(1): 5

Size of set of large itemsets L(2): 6

Size of set of large itemsets L(3): 2

Best rules found:

1. Bytes_Enviados=672 7805 ==> Status_HTTP=404 7805 conf:(1)
2. Request=GET Bytes_Enviados=672 7756 ==> Status_HTTP=404 7756 conf:(1)
3. Request=GET Status_HTTP=404 7781 ==> Bytes_Enviados=672 7756 conf:(1)
4. Status_HTTP=404 7831 ==> Bytes_Enviados=672 7805 conf:(1)
5. Bytes_Enviados=672 7805 ==> Request=GET Status_HTTP=404 7756 conf:(0.99)
6. Status_HTTP=404 Bytes_Enviados=672 7805 ==> Request=GET 7756 conf:(0.99)
7. Bytes_Enviados=672 7805 ==> Request=GET 7756 conf:(0.99)
8. Status_HTTP=404 7831 ==> Request=GET 7781 conf:(0.99)
9. Status_HTTP=404 7831 ==> Request=GET Bytes_Enviados=672 7756 conf:(0.99)
10. Data=25/Abr/2004 23268 ==> Request=GET 23022 conf:(0.99)

Fazendo a análise dos resultados obtidos, percebemos:

O suporte mínimo foi decrescendo até chegar a 0.25 (25%). Com uma confiança de 0.9 (90%) o algoritmo fez 15 ciclos até chegar ao resultado das 10 regras. No final de cada regra pode-se ver uma confiança de 0.99 (99%) e 1 (100%).

Pela regra 1 sabemos que para o atributo Bytes_Enviados=672 todas as repostas foram de página não exibida, ou seja, Status_HTTP=404, que representa o código de erro em página.

Pode-se constatar que temos 7805 registros, os quais 100% o resultado obtido foi essa regra de associação.

Já para a regra 2, temos mais um atributo fazendo parte da análise. São os *itensets* com três atributos. Nesta regra podemos ver que todas as vezes que aconteceu erro de página a quantidade de bytes enviados era a mesma e o tipo de requisição era GET.

Na verdade, a quantidade de bytes enviados sempre será igual nessa ocasião, pois é quando o servidor envia a página padrão para alertar um erro. Essa página está configurada no servidor *web* para ser exibida quando acontecer um erro, tipo arquivo não encontrado. Dessa forma, não encontramos informações úteis nessas regras, pois os atributos associados não nos dão a possibilidade de inferir informações úteis.

Por isso que o analista é a pessoa que deve escolher os atributos de forma a obter informações relevantes a partir dos dados. No caso desse trabalho, não pudemos mostrar um estudo de caso com informações mais úteis, pois o algoritmo não encontrou regras em situações que deveria ter encontrado. Até o momento da conclusão desse trabalho não se chegou a conclusão do motivo desse comportamento.

5.5 DIFICULDADES ENCONTRADAS

Dificuldades não são necessariamente barreiras intransponíveis, ao contrário disso, são elas que nos estimulam a persistir e nos encorajam a enfrentar os problemas. Encontramos dificuldades em várias situações de nossas vidas e na maioria situações em que nos deparamos pela primeira vez. Neste tópico destacamos as principais dificuldades encontradas durante a realização do mesmo:

Não foi possível aplicar com os filtros e algoritmos do ambiente WEKA diretamente no arquivo de *log* em seu estado bruto. Porque o formato NCSA, descrito detalhadamente no capítulo 3.2.3, tem caracteres delimitadores de data e string incompatíveis com o padrão ARFF. Além disso um cabeçalho é exigido pelo formato ARFF. Inicialmente foram feitas tentativas de mudar o algoritmo para entrar com os dados já selecionados e devidamente pré-processados, mas foi uma tentativa sem sucesso. Pela forma como o algoritmo foi construído o mesmo exigia que a entrada fosse dada nos formatos do arquivo ARFF que é descrito detalhadamente na seção 5.3.1.3. Isto teve basicamente duas conseqüências:

1 - Tornou-se necessário ler o arquivo mais de uma vez: Uma vez para montar a análise estatística e outra no momento de aplicar o algoritmo Apriori. Esta foi à única solução encontrada para o problema, mas não é a solução ideal, uma vez que os dados já estavam em memória.

2 - Foi preciso montar um novo arquivo com os atributos desejados para aplicar as Regras de Associação. O pacote de pré-processamento do WEKA já fazia esta tarefa através dos filtros, conforme descrito na seção 5.5, mas não se pode aproveitá-lo por motivos de formato conforme citado acima.

5.6 LIMITAÇÕES

Os itens c e f citados como objetivos específicos desse trabalho,pg 17 e 18 não foram atendidos pelos seguintes motivos:

1. O arquivo de *log* no padrão NCSA não traz o tráfego de entrada e saída. O *log* nesse formato de arquivo traz apenas o tráfego de saída, ou seja, o tráfego retornado pela resposta do servidor ao browser. O padrão NCSA foi escolhido, por ser um padrão para os dois servidores: IIS e Apache.

2. O tráfego do servidor representado pela quantidade de bytes que é gravado no arquivo de log, corresponde apenas ao fluxo de informações na camada de aplicação. Informações de protocolo, bits de controle e diversos outros controles devem ser considerados se o objetivo for estimar o tráfego em um link.

3. Percebeu-se durante o desenvolvimento do trabalho que um estudo mais aprofundado das camadas e protocolos de rede seria necessário para que a informação obtida representasse o consumo de banda medido pelo tráfego de informações. Ainda assim, o valor obtido seria apenas uma estimativa, pois o valor correto não seria possível descobrir com as informações contidas no arquivo de *log*. Isto desviaria muito o foco do trabalho que era puramente voltado à mineração de dados.

É possível ver somente as 20 páginas mais acessadas no intervalo do *log*.

5.7 EXTENSÕES

Destacamos neste tópico algumas idéias para aperfeiçoamentos do protótipo desenvolvido e detalhado nesse trabalho:

f) permitir usar arquivos de *log* em outros formatos diferentes do padrão NCSA;
g) detectar automaticamente o padrão seria uma boa sugestão nesse caso;
h) estimar a quantidade de banda do link de internet para esses usuários pela quantidade de bytes enviados e recebidos;
i) Gerar gráficos de acesso por regiões pela faixa de endereço IP do *browser* que está acessando o *site*;
j) Permitir ler os arquivos de *log* compactados;
k) Permitir configurar o número de páginas mais acessadas;
l) Permitir definir um intervalo para os gráficos. Hoje o protótipo pega o intervalo do *log*;
m) Permitir filtrar e gerar gráficos de páginas por conteúdo;
n) Permitir filtrar e gerar páginas por região de acesso;
o) Explorar a análise por técnicas estatísticas.

6 CONCLUSÕES

Através do processo de pesquisa e desenvolvimento deste trabalho, pode-se concluir que a mineração de arquivos de *log*, através de técnicas de *Data Mining*, fornece informações importantes sobre o uso de um *Website*. Estes subsídios permitem melhor compreensão dessas informações que estão concentradas em arquivos, com formatos específicos e, de difícil entendimento. Isso é possível à medida que as informações são agrupadas, categorizadas, contadas e inter-relacionadas.

Neste processo de contagem, o agrupamento e a categorização dos dados é indispensável para uma ferramenta que permita realizar tal tarefa de forma ágil. Este protótipo mostrou-se adequado para desempenhar tal papel.

A compreensão das informações contidas nesses *logs* é convertida espontaneamente em benefícios significativos para as empresas: mais vendas, menos custos, maior participação no mercado, novos mercados, relações mais estreitas com os clientes, fornecedores e parceiros.

Foi em busca da inteira compreensão das informações que foi dedicado esforço para entendimento do funcionamento da biblioteca JFreeChart, conforme David (2004), que permitiu que os resultados obtidos fossem apresentados na forma de gráficos. Esta biblioteca tem um potencial muito maior a ser explorado. Nem todos os resultados puderam ser apresentados na forma de gráficos.

Além dos gráficos, este trabalho trouxe subsídios que permitem uma maior compreensão de todo o processo que envolve a geração, transformação e utilização de arquivos de *log*, bem como um maior conhecimento dos servidores *web* mais usados na atualidade. A ferramenta WEKA demonstrou ser uma excelente opção para mineração de dados e aprendizado de algoritmos *Data Mining*. É certo que não teríamos ido tão longe sem o ambiente WEKA, pois minerar qualquer tipo de informação não é uma tarefa trivial até mesmo para problemas simples. Mas, além das ferramentas, a presença de um analista é indispensável, por mais eficiente e complexa que seja a técnica aplicada.

No caso de mineração em arquivos de *log* a complexidade aumenta, pois estes arquivos gerados pelo IIS e Apache representam de certa forma um histórico dos acessos a um

servidor, sejam quais forem os *sites* nele hospedados. Isto significa um volume muito grande de dados, muitas vezes de *sites* distintos, com inúmeras variáveis envolvidas.

Como sugestão, após a conclusão deste trabalho, os desenvolvedores de *websites* devem continuar as pesquisas e análises dos processos e técnicas de extração de dados do *log* de um *Website*. Para que o mercado possa usufruir as pesquisas aqui relatadas é necessária a continuidade do módulo de análise e demais aperfeiçoamentos apontados nas extensões desse trabalho.

Por fim, pode-se concluir que a principal vantagem que esse trabalho trouxe a comunidade acadêmica foi à iniciação no estudo de *Data Mining* em um ambiente *web*, apresentando e exemplificando as principais ferramentas, e técnicas. Ainda permite abrir frente para novas e interessantes pesquisas. Uma ferramenta deste tipo contribui significantemente em qualquer que seja o segmento utilizado.

REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023**: Informação e Documentação – Referências – Elaboração. Rio de Janeiro: ABNT, 2002.

BATISTA, Paulo e SILVA, Mário J. **Mining On-line Newspaper Web Access Logs** - Departamento de Informática - Faculdade de Ciências – Universidade de Lisboa - Campo Grande, 2001.

DAVID, Gilbert. **The JFreeChart Class Library**. Site oficial da biblioteca JfreeChart. Disponível em: <http://www.jfree.org/jfreechart/>. Acesso em: 06 Junho 2004.

FAYYAD, U., SHAPIRO, G.P. and SMYTH, P. **From Data Mining to Knowledge Discovery in Databases**. AAAIMIT Press, p.37-54, 1996.

KABIR, Mohammed J. **Apache 2 server, a Bíblia.** Tradução Vandenberg D. Souza. – Rio de Janeiro: Campus, 2002.

KIMBALL, Ralph; Merz, Richard. **Data Webhouse: construindo o Data Webhouse para a WEB.** Tradução Edson Furmankiewicz, Joana Figueiredo. Rio de Janeiro: Campus, 2000.

MENA, Jesus. **Data Mining your Website**. Boston, MA: Digital Press, 1999.

MICROSOFT CORPORATION. **Microsoft Internet Information Service, 2003**. Help do produto. Disponível em: <http://www.microsoft.com/windows2000/en/server/iis/htm/core/iiabtlg.htm>. Acesso em: 13 maio 2004.

MIRANDA, Dhalila; SABORÊDO, Alexandre De Paiva, et al. **Iniciação Científica - Data Mining.** AEDB Associação Educacional Dom Bosco. Resende - Rio de Janeiro. 2003. Disponível em <http://www.inf.aedb.br/datamining/>. Acesso em: 08/05/2004.

MITTCHEL, T. M. **Machine Learning**. McGraw Hill, 1997.

NARDELLI, Bianca. **Protótipo de um sistema de informação gerencial aplicado a central de informação aos alunos da FURB utilizando Data Mining**. Monografia (Graduação em Ciências da Computação) – Centro de Ciências Exatas e Naturais. Universidade Regional de Blumenau. 68 f, 2000.

PETTT - Program for Educational Transformation Through Technology. Disponível em: <http://depts.washington.edu/pettt/papers/reviews/Webstatstoolcomparison.xls>. Artigos publicados. Acesso em: 08/05/2004.

RODRIGUES, Alexandre Medeiros. **Técnicas de Data Mining classificadas do ponto de vista do usuário**. 2000. 104p. Tese – Universidade Federal do Rio de Janeiro, COPPE/Engenharia de Produção, Rio de Janeiro.

SHARMA, V.; SHARMA, R. **Desenvolvendo sites de e-commerce - como criar eficaz e lucrativo site de e-commerce, passo a passo**. Makron Books, 2001.

TULLOCH, Mitch; SANTRY, Patrick. **Dominando IIS 5.0**. Rio de Janeiro: Editora Ciência Moderna Ltda., 2001.

WEKA. The University of Waikato. **Software – Weka 3 – Data Mining Software in Java**. Disponível em: < http://www.cs.waikato.ac.nz/ml/weka/>. Acesso em: 30/05/2004.

WITTEN, I. H; FRANK, Eibe. **Data mining : practical machine learning tools and techniques with Java implementations**. San Francisco : Morgan Kaufmann, 2000. xxvi, 371p.

World Wide Web Consortium (W3C). Disponível em: <http://www.w3.org/>. Acesso em: 08/05/2004.

XUE, Gui-Rong; Zeng, Hua-Jun. **Log Mining to Improve the Performance of Site Search**. Artigo: Shanghai Jiao-Tong University – Computer Science and Engineering, P.R. China. 8p. 2000.