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

Acadêmico: Leonardo José Correia

Orientador: Prof. Ricardo Alencar Azambuja

Blumenau, Julho/2004

Roteiro

Introdução

- Objetivo
- Origem do trabalho
- Problema
- Relevância do trabalho

Fundamentação teórica

- Conceitos básicos

Roteiro

Desenvolvimento

- Requisitos
- Especificação
  - *Técnicas e ferramentas utilizadas*
  - *Apresentação da especificação*
- Implementação
  - *Técnicas e ferramentas utilizadas*
- Resultados e discussões

Considerações finais

- Extensões

INTRODUÇÃO

Objetivo Geral

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,* extraindo desse arquivo informações que permitam montar estatísticas e apresentá-las de forma objetiva, simples e de fácil entendimento através de uma estrutura de manipulação de dados em memória.

Introdução - ORIGEM

- Surgiu na empresa onde trabalho (HBsis informática).
- Monitorar o tráfego do servidor e uso da aplicação.
- Necessário para entender como o *site* ou a aplicação está sendo usada.
- Custo elevado de softwares existentes no mercado.

Introdução – PROBLEMA

- Desafio: Não usar banco de dados.
- Tornar a aplicação mais performática.
- Única forma de avaliar o uso do *site* é através desses *logs*.
- Esse *logs* são difíceis de avaliar sem uma ferramenta que auxilie nessa tarefa.

Introdução - RELEVÂNCIA

- Quantidade de informações disponíveis em página web.
- Informações úteis nos logs dos servidores web.
- Transformar as informações do log em gráficos.
- Monitoração dos acessos a um *site* ou aplicativo *web.*
- Antecipar-se a um acontecimento, aumentar as vendas ou a satisfação dos usuários.

FUNDAMENTAÇÃO TEÓRICA

O formato NCSA:

- Escolhido por ser um formato pré-definido e comum entre os *web servers* Apache e IIS.
- Fragmento de arquivo de log do domínio .inf da FURB:

```
201.4.239.102 - - [25/Apr/2004:04:15:38 -0300] "GET /~maw/img/pacer.gif HTTP/1.1" 404 672
201.4.239.102 - - [25/Apr/2004:04:15:37 -0300] "GET /~maw/softwares.htm HTTP/1.1" 200 14281
201.4.239.102 - - [25/Apr/2004:04:15:40 -0300] "GET /~maw/img/topo.gif HTTP/1.1" 200 802
201.4.239.102 - - [25/Apr/2004:04:16:29 -0300] "GET /~maw/img/pacer.gif HTTP/1.1" 404 672
200.135.24.57 - - [25/Apr/2004:04:18:11 -0300] "GET / HTTP/1.0" 200 5810
81.152.18.197 - - [25/Apr/2004:04:18:54 -0300] "GET /~poo/doc/api/java/security/acl/Acl.html HTTP/1.1" 200 23793
64.68.82.27 - - [25/Apr/2004:04:20:10 -0300] "GET /~jomi/xml/exemplos/vendedor/?C=N;O=D HTTP/1.0" 200 2003
64.68.82.137 - - [25/Apr/2004:04:20:20 -0300] "GET /~jomi/xml/exemplos/vendedor/SaxNomeVendedores.java HTTP/1.0" 304 -
64.68.82.137 - - [25/Apr/2004:04:20:20 -0300] "GET /~jomi/xml/exemplos/vendedor/SaxNomeVendedores.java HTTP/1.0" 200 2700
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
```

- Estrutura do arquivo no formato NCSA:

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

- A rotação do arquivo pode ser feita por tamanho pré-definido ou por periodicidade.

- Cada acesso pode gerar inúmeras transações.

- Entendendo os *hits* de acesso no arquivo de *log*:
  - 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.

- O visitante é identificado pelo seu endereço IP, mas no caso de um compartilhamento por *proxy*, vários visitantes utilização o mesmo IP.

- Código dos erros HTTP:
  - 200 a 299 – transação foi bem sucedida.
  - 300 a 399 – ocorreu um redirecionamento.
  - 400 a 599 – ocorreu algum tipo de erro.

- Data Mining – Regras de Associação:
  - Por que Regras de Associação?
  - Segundo Batista (2001, p. 2), Regras de Associação descobre relacionamentos entre conjuntos de itens.
  - Conforme Miranda et al. (2003), as Regras de Associação representam padrões onde a ocorrência de eventos em conjunto é alta. 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 *itensets* distintos no conjunto de dados.

Fundamentação Teórica – CONCEITOS BÁSICOS

- Um *itenset* corresponde a uma combinação de atributo/valor.
- O *dataset* é o conjunto de todos os registros do *log.*
- Suporte ou Cobertura é o número de vezes que um atributo/valor acontecem ou se repetem no *dataset.*
- Confiança ou Precisão representa o suporte dividido pelo número de instâncias para o qual as condições do lado direito da regra aconteceram (conseqüente, precedido do símbolo =>).
- E o que é uma regra?

- Data Mining – APRIORI:
  - *Header*:

```
@relation Nome_do_Dataset

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

@data
```

  - Indispensável para montar as regras pelo algoritmo Apriori

DESENVOLVIMENTO

Desenvolvimento – METODOLOGIA

- O trabalho foi desenvolvido utilizando a linguagem Java sob o paradigma da Orientação a Objetos;
- Foi utilizado o modelo de ciclo de vida espiral;
- *Data Mining*
  - Utilizado a técnica de Regras de Associação;
  - Algoritmo APRIORI;
  - Implementado no Ambiente WEKA;

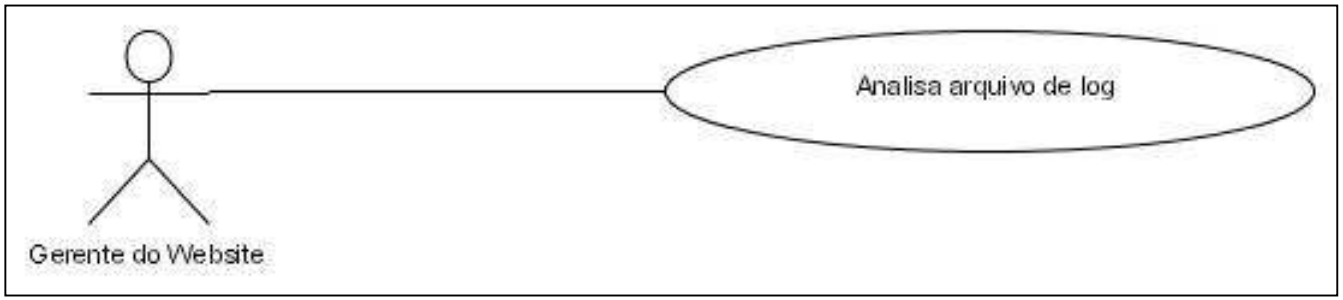
Desenvolvimento – REQUISITOS

- Analisar o arquivo de *log* de acesso do IIS ou APACHE, contando e agrupando as informações registradas;
- Disponibilizar gráficos estatísticos dos acessos ao *site*, pelo endereço Internet Protocol (IP) do cliente ou estação;
- Mostrar a quantidade de usuários que acessam um determinado *website*;
- Mostrar a quantidade de bytes que entram e saem de um servidor, estimando a quantidade de banda do *link* de internet para esses usuários.

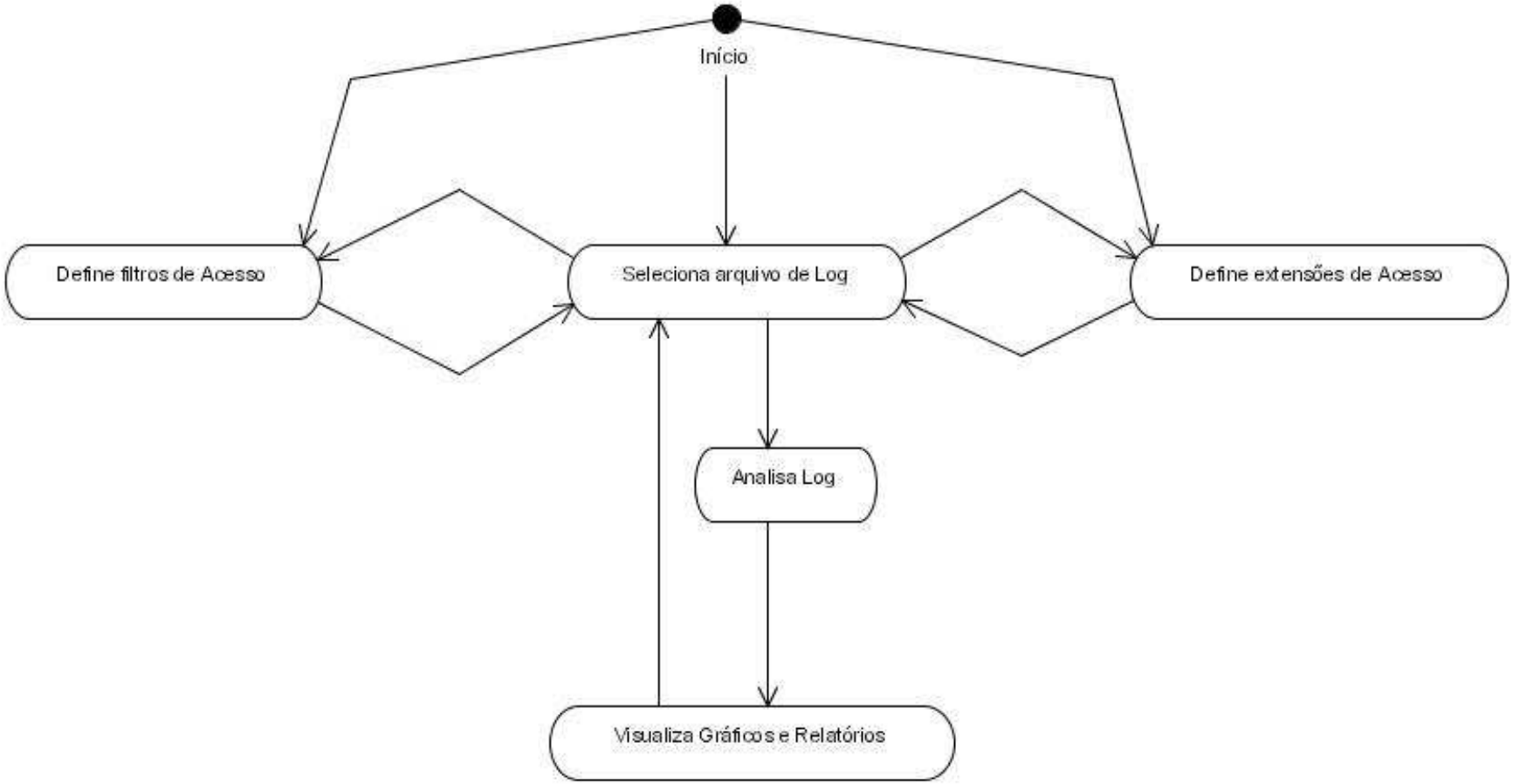
- Identificar padrões de comportamento dos usuários de um *website* ou uma aplicação *web* pela análise dos arquivos de *log* de um servidor IIS ou Apache.

- Utilizar a técnica de Regras de Associação de *Data Mining* para fazer essa identificação.

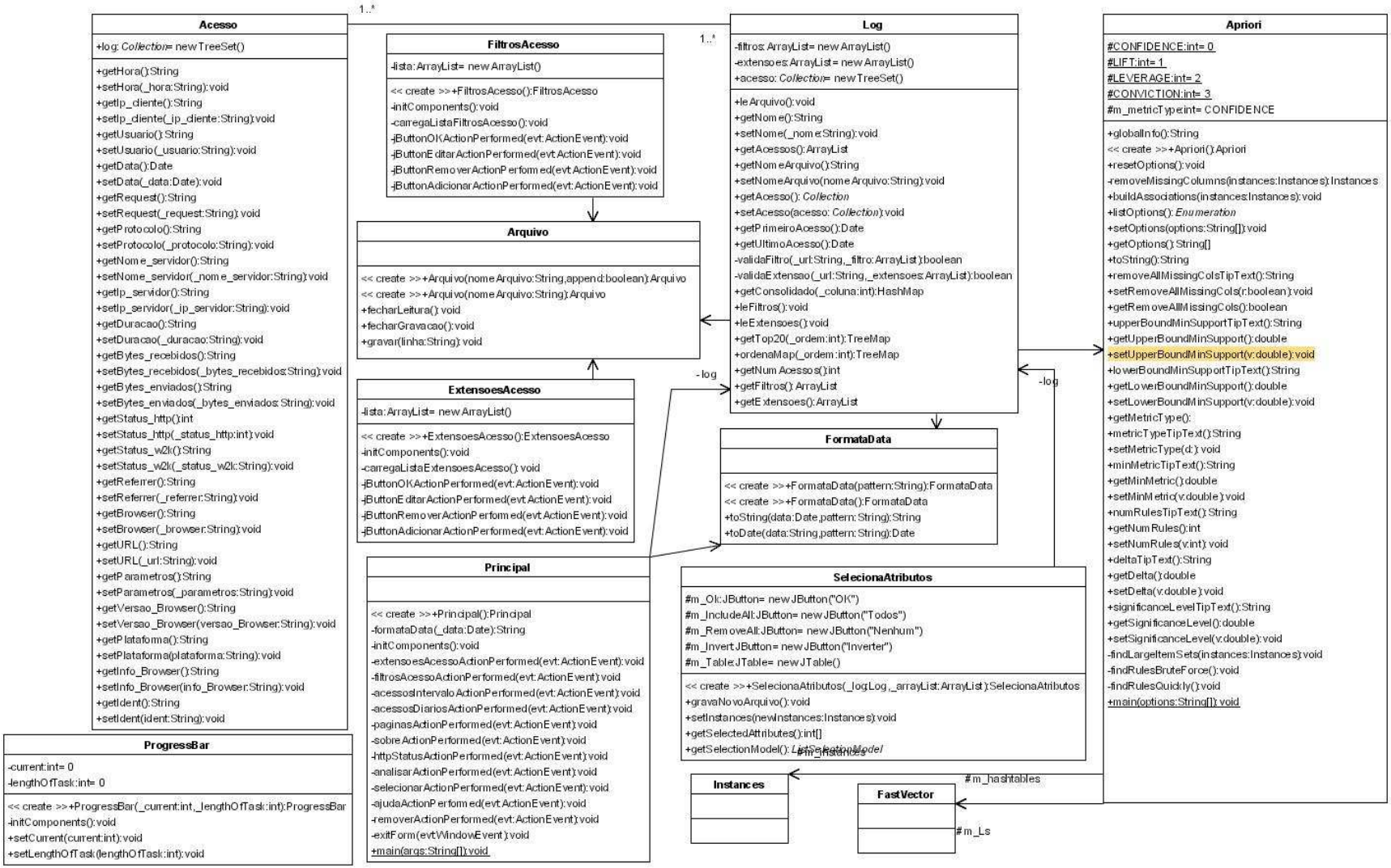
- Diagrama de Casos de Uso:



- Diagrama de atividades:



Desenvolvimento – ESPECIFICAÇÃO



Desenvolvimento – TÉCNICAS E FERRAMENTAS UTILIZADAS

- Poseidon for UML Standard Edition;
- Eclipse 3.0;
- WEKA - *Waikato Environment Knowledge Analysis*;

Desenvolvimento - IMPLEMENTAÇÃO

```
public HashMap getConsolidado(int _param) {
    ArrayList filtros = new ArrayList();
    ArrayList extensoes = new ArrayList();
    filtros = getFiltros();
    extensoes = getExtensoes();
    int status_http;
    String pagina, url, filename;
    Date data = new Date();
    Integer contador = new Integer(0);
    HashMap hashMap = new HashMap();
    File file;
    for (int i = 0; i < arraylist.size(); i++) {
        //pega o primeiro e o último acesso.
        Acesso registro = (Acesso) arraylist.get(i);
        if (i == 0){
            primeiroAcesso = registro.getData();
        }else if (i == arraylist.size()-1){
            ultimoAcesso = registro.getData();
        }
        url = registro.getURL();
        pagina = getPageFromURL(url);

        //param == 1 - Páginas mais acessadas
        if (_param == 1) {
            if (filtros.isEmpty()) {
                if (extensoes.isEmpty()) {
                    if (hashMap.containsKey(pagina)){
                        contador = (Integer) hashMap.get(pagina);
                        hashMap.put(pagina,new Integer(contador.intValue()+1));
                    }else{
                        hashMap.put(pagina,new Integer(1));
                    }
                }else{
                    if(validaExtensao(pagina,extensoes)){
                        if (hashMap.containsKey(pagina)){
                            contador = (Integer) hashMap.get(pagina);
                            hashMap.put(pagina,new Integer(contador.intValue()+1));
                        }else{
                            hashMap.put(pagina,new Integer(1));
                        }
                    }
                }
            }else{
                if(validaFiltro(url,filtros) && validaExtensao(pagina,extensoes)){
                    if (hashMap.containsKey(pagina)){
                        validaExtensao(pagina,extensoes);
                        contador = (Integer) hashMap.get(pagina);
                        hashMap.put(pagina,new Integer(contador.intValue()+1));
                    }else{
                        hashMap.put(pagina,new Integer(1));
                    }
                }
            }
```

```
        }else if (_param == 2){
            status_http = registro.getStatus_http();
            url = registro.getURL();
            if(validaFiltro(url,filtros) && validaExtensao(url,extensoes)){
                if (status_http > 199 && status_http < 300){
                    if (hashMap.containsKey("200 a 299 - Status OK")){
                        contador = (Integer) hashMap.get("200 a 299 - Status OK");
                        hashMap.put("200 a 299 - Status OK",new Integer(contador.intValue()+1));
                    }
                    else{
                        hashMap.put("200 a 299 - Status OK",new Integer(1));
                    }
                }
                else if (status_http > 299 && status_http < 400){
                    //valores[2] = "300 a 399";
                    if (hashMap.containsKey("300 a 399 - Redirecionamento")){
                        contador = (Integer) hashMap.get("300 a 399 - Redirecionamento");
                        hashMap.put("300 a 399 - Redirecionamento",new Integer(contador.intValue()+1));
                    }
                    else{
                        hashMap.put("300 a 399 - Redirecionamento",new Integer(1));
                    }
                }
                else if (status_http > 399 && status_http < 600){
                    //valores[3] = "400 a 599";
                    if (hashMap.containsKey("400 a 599 - Erros")){
                        contador = (Integer) hashMap.get("400 a 599 - Erros");
                        hashMap.put("400 a 599 - Erros",new Integer(contador.intValue()+1));
                    }
                    else{
                        hashMap.put("400 a 599 - Erros",new Integer(1));
                    }
                }
                else if (status_http == 0){
                    //valores[4] = "Inválidos";
                    if (hashMap.containsKey("Inválidos")){
                        contador = (Integer) hashMap.get("Inválidos");
                        hashMap.put("Inválidos",new Integer(contador.intValue()+1));
                    }
                    else{
                        hashMap.put("Inválidos",new Integer(1));
                    }
                }
            }
        }else if (_param == 3){
            data = registro.getData();
            if (hashMap.containsKey(data)){
                contador = (Integer) hashMap.get(data);
                hashMap.put(data,new Integer(contador.intValue()+1));
            }else{
                hashMap.put(data,new Integer(1));
            }
        }
    }
    return hashMap;
```

Desenvolvimento – 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)

CONSIDERAÇÕES FINAIS

CONSIDERAÇÕES FINAIS

- Este trabalho se propôs a explorar a tecnologia de controle de acesso por *log* dos *web servers* no monitoramento de *web sites*, bem como as técnicas mais comuns para extração de informação no contexto da *Word Wide Web* usando *Data Mining* para extrair padrões e estabelecer relacionamentos entre os resultados obtidos da análise dos logs.

- Esses métodos e técnicas podem fornecer informações valiosas sobre o uso de um *website* e ajudar as pessoas a compreender de forma mais objetiva as informações que estão concentradas em arquivos com formatos específicos e de difícil entendimento.

EXTENSÕES

Algumas implementações futuras:

- Permitir usar arquivos de *log* em outros formatos diferentes do padrão NCSA. Detectar automaticamente o padrão seria uma boa sugestão nesse caso.
- Estimar a quantidade de banda do link de internet para esses usuários pela quantidade de bytes enviados e recebidos.
- Gerar gráficos de acesso por regiões pela faixa de endereço IP do *browser* que está acessando o *site.*
- Permitir ler os arquivos de *log* compactados.

EXTENSÕES

- Permitir configurar o número de páginas mais acessadas.
- Permitir definir um intervalo para os gráficos. Hoje o protótipo pega o intervalo do *log*.
- Permitir filtrar e gerar gráficos de páginas por conteúdo.
- Permitir filtrar e gerar gráficos de páginas por região de acesso.
- Explorar a análise por técnicas estatísticas.

APRESENTAÇÃO DO PROTÓTIPO