Algoritmo de Sugestão de Origens de Informações em Bancos de Dados para Business Intelligence

Patrícia Teixeira Mello[1], Wamberg Glaucon Chaves de Oliveira[2], Pablo Ximenes[3]

[1]Instituto Atlântico – Fortaleza – CE - Brasil

[2]Banco do Nordeste do Brasil – BNB – Fortaleza – CE - Brasil.

[3]Empresa de Tecnologia da Informação do Ceará – ETICE – Governo do Estado do Ceará – Fortaleza – CE – Brasil

patriciatmello@gmail.com, wamberg@gmail.com, pablo@ximen.es

Abstract. *This article intends to implement an algorithm for suggesting sources of information into databases based on relationship between natural language and database tables. With this in mind, it presents basic concepts on Knowledge Bases and it presents a prototype's development flow of proposed solution. In addition, we made computational tests with developed algorithm and we made analysis and enhancements based on outcomes. Finally, the article presents a resultant conclusion from outcomes' analysis of case study.*

Resumo. *Este artigo tem como objetivo a implementação de um algoritmo de sugestão de origens de informações em bancos de dados baseado na relação entre palavras da linguagem natural e tabelas em bancos de dados. Com esse intuito, apresenta conceitos básicos sobre Bases de Conhecimento e o fluxo de desenvolvimento de um protótipo da solução proposta. Em seguida, realiza testes computacionais sobre o algoritmo desenvolvido e baseado nos resultados obtidos faz análises e melhorias. Finalmente, apresenta uma conclusão resultante das análises feitas no estudo do caso.*

1 INTRODUÇÃO

No ambiente corporativo é cada vez maior o volume de informações geradas por diferentes sistemas computacionais ligados aos processos da empresa. As corporações acabam por manter grandes volumes de dados que, quando dispersos e não inegrados, não servem para subsidiar as decisões da organização. A inteligência de negócios (ou BI de Business Intelligence, do inglês) aparece como alternativa para a transformação dessa massa volumosa de dados sem sentido em informações valiosas para o negócio. Dessa forma, é crescente o uso de informações geradas por técnicas de BI para tomada de decisão em nível gerencial.

Essa pesquisa foi realizada no setor de Inteligência de Negócios de uma instituição financeira. Nesse setor, a demanda de solicitações de informações aleatórias nas diversas bases de dados é frequente. São informações que não estão disponíveis nos relatórios

dos sistemas de uso operacional e gerencial e, geralmente, têm caráter urgente, visando atender gerentes, executivos e órgãos externos. Por essa razão, a equipe de especialistas que compõe esse setor deve ter bastante conhecimento do negócio da empresa e das estruturas das bases de dados, a fim de fazer a tradução do que é solicitado pelo usuário para o que deve ser pesquisado e em qual base de dados deve ser pesquisado. No entanto, parte desse conhecimento é muito intuitivo e dependente da experiência do especialista, o que prejudica o desempenho dos membros inexperientes da equipe, assim como o desenvolvimento dessas demandas por parte deles.

Essa pesquisa foi motivada pela necessidade de uma solução de apoio a identificação das possíveis origens das informações nas diversas bases de dados da instituição financeira em questão, que unificasse o conhecimento, minimizasse a dificuldade dos colaboradores menos experientes e que desse mais celeridade ao processo de geração das informações.

Conforme observações feitas por colaboradores mais experientes do setor de Inteligência de Negócios, foi verificado que os termos usados em uma nova solicitação de informação, na maioria das vezes, refletiam pesquisas nas mesmas bases de dados de solicitações anteriores que faziam uso dos mesmos termos. Isto é, determinadas palavras da linguagem natural usadas pelo usuário final tinham relação direta com sistemas, tabelas e atributos em bases de dados.

Com base nessa hipótese, essa pesquisa irá introduzir conceitos básicos sobre Bases de Conhecimento; implementar um protótipo da base de conhecimento sobre esse domínio e alimentar essa base com um conjunto de demandas; implementar um algoritmo de identificação das possíveis origens das informações nas diversas bases de dados baseado em pesquisa textual; realizar testes com o algoritmo construído, submetendo dados de demandas reais cadastradas ou não na base de conhecimento e avaliar os resultados obtidos.

O restante desse trabalho está estruturado da seguinte forma: apresentação do embasamento teórico da pesquisa; exposição do fluxo de desenvolvimento da solução proposta e de testes computacionais com os resultados obtidos; e a conclusão com sugestões de trabalhos futuros

2 REFERENCIAL TEÓRICO

Antes de qualquer explicação sobre conhecimento, faz-se necessário deixar claro que ele difere de "dado" e "informação", apesar de se relacionar com ambos. Segundo Davenport e Prusak (1998), dados são um conjunto de fatos distintos e objetivos, relativos a eventos. Em um contexto organizacional, dados são registros estruturados de transações. Já a informação é como uma mensagem. Ela tem a finalidade de mudar o modo como o destinatário vê algo, e exercer impacto sobre seu julgamento e comportamento. Davenport e Prusak (1998) dizem: "pensem na informação como dados que fazem a diferença".

O conhecimento é um termo de difícil definição devido a sua característica intuitiva. O conhecimento é típico da natureza humana e por isso sujeito à percepção humana de um determinado assunto. Ele é muito mais profundo do que "dado" e "informação". Davenport e Prusak (1998) referem-se ao conhecimento como: "uma mistura fluida de experiência condensada, valores, informação contextual e *insight* experimentado, a qual proporciona uma estrutura para a avaliação e incorporação de novas experiências e informações. Ele tem origem e é aplicado na mente dos conhecedores".

De maneira geral, o conhecimento pode ser considerado a capacidade de abstração de um determinado assunto de forma a apresentar características, relações com outros assuntos, ações e decisões que podem ser tomadas. Ele pode ser dividido em dois grupos: conhecimento tácito e conhecimento explícito.

Segundo Cruz (2007), o conhecimento tácito é aquele que nós acumulamos dentro de nós mesmos, fruto do aprendizado, da educação, da cultura e da experiência de vida.

Por outro lado, ainda segundo Cruz (2007), o conhecimento explícito é aquele compartilhado, que passamos a outros para que estes, também, desenvolvam suas habilidades e possam gerar mais conhecimento.

O conhecimento tácito é o conhecimento "informal", de natureza subjetiva e intuitiva, o que torna muito difícil sua identificação, coleta e organização. Já o conhecimento explícito é o conhecimento "formal", que pode ser materializado, estruturado e disponibilizado em uma base de dados com maior facilidade.

2.1 Bases de Conhecimento

A Gestão do Conhecimento (*Knowledge Management*) trata da criação, identificação, integração, recuperação, compartilhamento e utilização do conhecimento dentro de uma organização. No contexto desse trabalho, os principais benefícios da Gestão do Conhecimento são: o melhor aproveitamento do conhecimento já existente na organização; a unificação e distribuição do conhecimento; a redução dos custos; o tempo de desenvolvimento de novos produtos e a melhor qualidade dos produtos gerados.

Uma base de conhecimento trata-se de um repositório centralizado de informação e conhecimento sobre um determinado domínio. Ela é uma importante ferramenta da Gestão do Conhecimento e tem como finalidade primordial o armazenamento e a difusão do conhecimento em uma organização. Algumas vezes esse conhecimento é incompleto ou inconsistente. Isso se deve ao fato que nem todo conhecimento se encontra na base ou é proveniente de fontes diversas e com padrões de avaliação diferentes. Outro fator importante é que o conhecimento armazenado oriunda da mente dos conhecedores (especialista) e por isso está passível a equívocos, assim como o especialista humano. As Bases de Conhecimento classificam-se em dois grandes tipos:

- **Bases de Conhecimento legíveis por máquinas**: projetadas para armazenar conhecimento de forma legível ao computador, geralmente com a finalidade de obter raciocínio dedutivo automático. Possui uma série de dados em forma de regras que descrevem o conhecimento de maneira logicamente consistente. Elementos lógicos como operadores e condições são utilizados ("E", "OU", "NOT", "IF", "CASE", etc).
- **Bases de Conhecimento legíveis por humanos**: projetadas para armazenar conhecimento de forma legível para as pessoas, principalmente para propósitos de aprendizagem. São usadas para obter conhecimento explícito sobre a organização, descrição de processos, artigos, documentos, manuais e outros. O principal benefício é proporcionar meios de descobrir soluções de problemas, os quais poderiam ser resolvidos baseados em outros problemas dentro ou fora da mesma área de conhecimento.

Na implementação de uma base de conhecimento, o conhecimento a ser armazenado pode ser obtido por meio da documentação disponível, análise dos processos e, principalmente, junto ao especialista humano. A forma como esse conhecimento é armazenado também é muito importante. Um exemplo que será usado nesse trabalho é a

adoção de padrões de nomenclaturas, que auxiliam o desenvolvimento de ferramentas de manipulação do conhecimento armazenado.

Uma característica desejável e recomendável em uma base de conhecimento é o uso de um mecanismo de pesquisa simples e intuitivo, com procura por palavras ou expressões. A busca textual é uma forma muito comum de pesquisa de informações em Bases de Conhecimento, muito popularizada na Internet devido a sites buscadores.

No contexto desse trabalho, o especialista deverá informar a palavra ou conjunto de palavras que deseja verificar. Cada demanda cadastrada na base de conhecimento possuirá palavras-chaves que irão caracterizá-la. O algoritmo realizará a busca sobre essas palavras-chaves, sendo que sempre analisará todas as demandas cadastradas, recolhendo todas as informações sobre sistemas e tabelas utilizados, a fim de verificar as coincidências entre as demandas. Esse tipo de busca é conhecida como linear, onde a pesquisa é feita percorrendo toda a base e trazendo todas as ocorrências do elemento desejado, quando houver.

3 DESENVOLVIMENTO

O estudo de caso foi realizado no setor de Inteligência de Negócios de uma instituição financeira. Foi implementado e preenchido um protótipo de base de conhecimento e desenvolvido um algoritmo de sugestão de origens de informações em bases de dados.

Esse capítulo irá apresentar o fluxo de desenvolvimento da solução, além de funcionalidades necessárias para o preenchimento da base de conhecimento e padronização do conhecimento armazenado, e finalmente, a lógica envolvida no algoritmo de sugestão de origens de informações. A figura 1 mostra o fluxo do desenvolvimento:

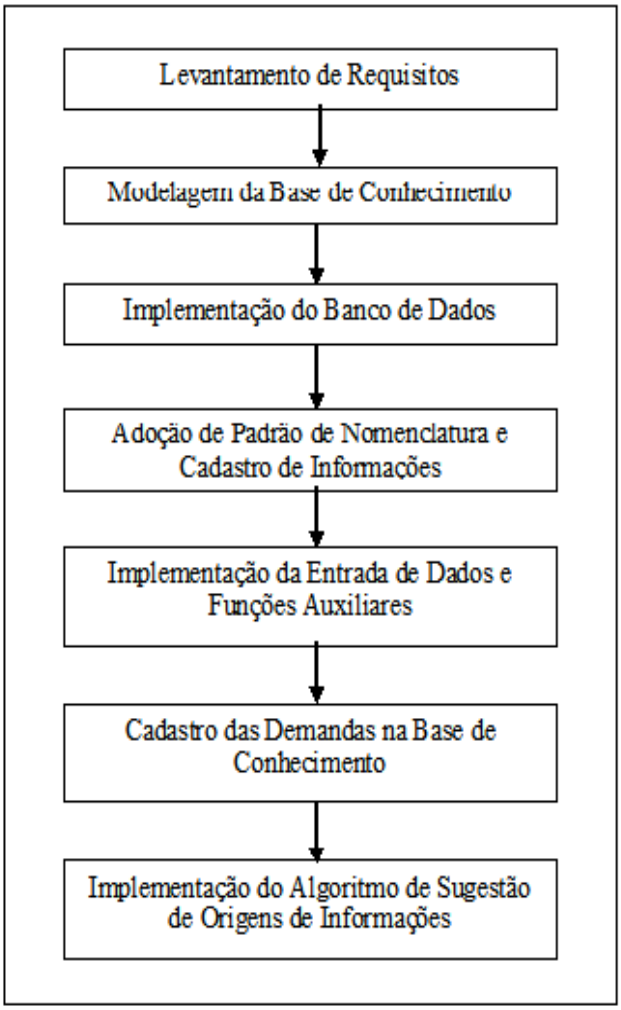

Figura 1 - Fluxo de desenvolvimento da solução
Fonte: própria, 2011

3.1 Levantamento de Requisitos

O levantamento de requisitos levou em consideração quais informações eram necessárias para a análise que seria feita pelo algoritmo, e quais informações seriam interessantes para o aprendizado dos colaboradores do setor de Inteligência de Negócios. Foram analisados a atual estrutura do cadastro de solicitações e os documentos e produtos que são gerados até a finalização da demanda.

3.2 Modelagem da Base de Conhecimento

Com base no levantamento de requisitos, foram definidas quais as principais entidades e atributos iriam compor o modelo físico da base de conhecimento. Além disso, outro fator levado em consideração era que a modelagem favorecesse a performance das pesquisas. A figura 2 mostra como ficou o modelo físico:

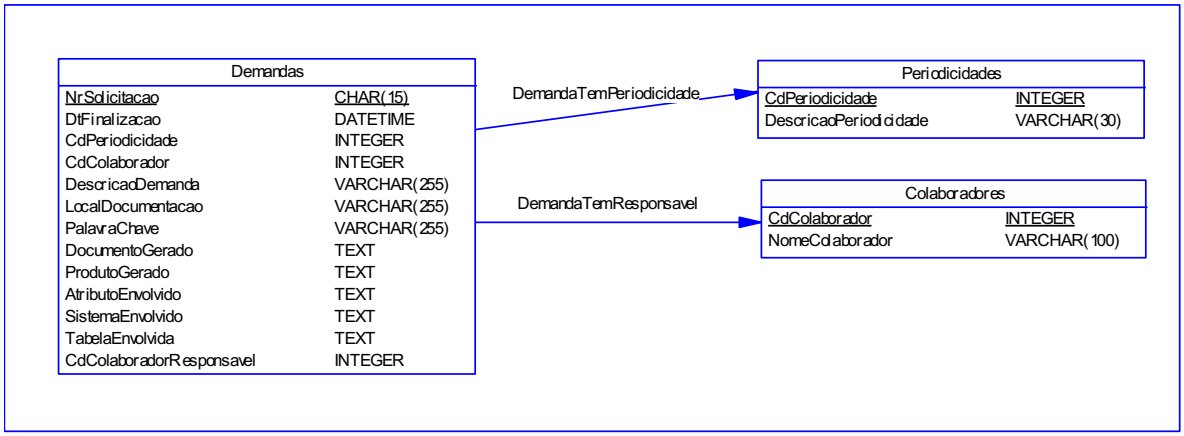

Figura 2 – Modelo físico inicial
Fonte: própria, 2011

Durante o processo de modelagem verificou-se que os atributos “PalavraChave”, “AtributoEnvolvido”, “SistemaEnvolvido”, e “TabelaEnvolvida” poderiam se tornar novas tabelas relacionadas com a tabela “Demandas” em um relacionamento de muitos para muitos. Esse tipo de relacionamento, por sua vez, proporcionaria o surgimento de uma tabela de ligação. Por exemplo, a tabela “Demandas” se relacionaria com a tabela “SistemasDaDemanda”, que se relacionaria com a tabela “Sistemas”. Essa modelagem normalizada, típica do modelo relacional, reduz a redundância e inconsistência dos dados. No entanto, elas têm mau desempenho nas pesquisas, devido ao número de ligações que precisam fazer para encontrar uma informação. Como o algoritmo faz uso direto desses atributos, optou-se por mantê-los desnormalizados com o intuito de melhorar a performance nas pesquisas.

Durante a fase de cadastro da base de conhecimento verificou-se a necessidade de outra tabela que auxiliaria na criação de um padrão para cadastro de palavras-chaves, onde existiriam cadastradas várias palavras relacionadas com uma determinada palavra padrão de mesmo significado semântico. Essa tabela não estaria relacionada diretamente com a tabela “Demandas”, funcionando apenas como dicionário de sinônimos. O item 3.4 irá explicar como é usado esse dicionário. Com isso, o modelo físico final ficou conforme a figura 3:

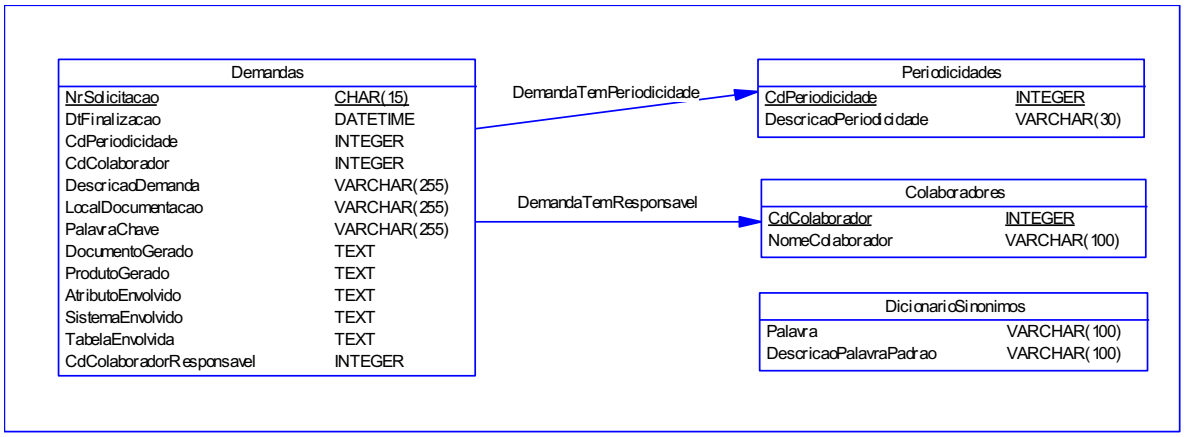

Figura 3 – Modelo físico final
Fonte: própria, 2011

3.3 Implementação do Banco de Dados

O Sistema Gerenciador de Banco de Dados (SGDB) usado para implementar o banco de dados foi o SQL Server 2000. O nome do banco criado foi "DICDEM", que é uma abreviação de "Dicionário de Demandas".

3.4 Adoção de Padrão de Nomenclatura e Cadastro de Informações

Devido aos atributos "AtributoEnvolvido", "TabelaEnvolvida" e "SistemaEnvolvido" serem desnormalizados e do tipo TEXT, foi necessária a adoção de um padrão de nomenclatura das informações, de maneira que a lógica do algoritmo se beneficiasse, facilitasse o cadastro pelo usuário e garantisse a integridade das informações. Para isto, adotou-se o mesmo padrão de nomenclatura de objetos de banco de dados utilizados pela instituição financeira onde se deu a pesquisa. Assim, o usuário só precisaria informar o atributo da mesma maneira que foi implementada nos scripts SQL: "Tabela.Atributo", onde "Tabela" é o nome da tabela existente no banco de dados e "Atributo" é o nome do campo dessa tabela. Com esse padrão também é possível saber de qual sistema se origina o atributo, pois essa informação está contida no nome da tabela. Dessa maneira, o usuário preenche apenas a informação de "AtributoEnvolvido" com o formato "Tabela.Atributo" e a lógica do

cadastro de demandas se encarrega de extrair as informações de “TabelaEnvolvida” e “SistemaEnvolvido”. Por exemplo: ao informar o atributo VSKAE177.CO1_GE8, VSKAE177 é a tabela e SKA é a sigla que identifica o sistema.

Outro atributo que também precisou ter sua informação padronizada foi “PalavraChave”, principalmente porque o algoritmo de sugestão de origens faz a pesquisa sobre ele. Assim como os atributos comentados anteriormente, ele é desnormalizado e do tipo TEXT. Esse atributo contém as palavras que caracterizam a demanda. Como uma palavra pode ter vários sinônimos, foi criado um dicionário de sinônimos onde os principais termos foram cadastrados e relacionados com um termo padrão. O usuário informa a palavra padrão e a lógica do cadastro verifica sua existência no dicionário. Se existir, troca o termo informado pelo usuário pelo termo padrão, senão mantém o do usuário. Em conjunto com esse dicionário foram desenvolvidas lógicas de conversão de sintaxe que serão comentadas no item 3.5.

Para todos os atributos do tipo TEXT, também se convencionou que se houvesse mais de uma informação, elas deveriam ser separadas pelo simbolo “;”. Por exemplo, para o atributo “PalavraChave” de uma determinada demanda foram atribuídas três (3) palavras-chaves: CREDITO; EFETIVO; 2010.

3.5 Implementação da Entrada de Dados e Funções Auxiliares

A linguagem usada na implementação da entrada de dados e funcionalidades auxiliares foi a *Visual Basic* do pacote *Microsoft Visual Studio 6.0*. As funcionalidades auxiliares foram desenvolvidas para serem usadas tanto no cadastro das demandas quanto na pesquisa por origens de informações, tendo maior destaque as funções abaixo:

- **TiraAcentos**: retira qualquer acentuação no texto informado pelo usuário. Por padrão, a base não considera as acentuações. Por exemplo: INFORMAÇÃO se torna INFORMACAO.
- **RetiraPalavrasRepetidas**: retira as palavras padrões repetidas que o usuário possa ter informado erroneamente.
- **PadronizaPalavras**: função que recebe as palavras padrões informadas pelo usuário e verifica se elas existem no dicionário de sinônimos. Se existir, troca pela palavra padrão, senão mantém a palavra fornecida pelo usuário.

- **RetiraPluralBasico**: função que converte as palavras no plural para o singular, segundo as regras básicas da língua portuguesa. Por padrão, a base não considera apenas palavras no singular. Por exemplo: INFORMAÇÕES se torna INFORMAÇÃO.
- **ConverteEspacoDaFrase**: função que recebe uma frase e converte os espaços em branco entre cada palavra por “;”. Se houver mais de um espaço entre as palavras, mesmo assim converte apenas para um “;”. É usada quando o usuário ao fornecer as palavras-chaves se esquece de separá-las por “;”. É usada tanto na entrada de dados quanto no algoritmo de sugestão de origens de informações.

Em alguns atributos também é padrão da base guardar a informação em caixa alta (maiúsculo), sendo que para isso é utilizada uma função própria da linguagem. A figura 4 mostra a tela de cadastro das demandas de informações. Nessa tela existe um botão ao lado do campo de palavras-chaves que dá acesso ao dicionário de sinônimos, caso o usuário não queria escrever as palavras-chaves. Ao clicar nesse botão, o dicionário é aberto e seleciona automaticamente as palavras existentes na descrição da demanda, convertendo-as para a palavra padrão. Essa seleção automática é feita como sugestão de palavras-chaves, sendo possível o usuário desmarcá-las ou não, assim como selecionar outras palavras que desejar.

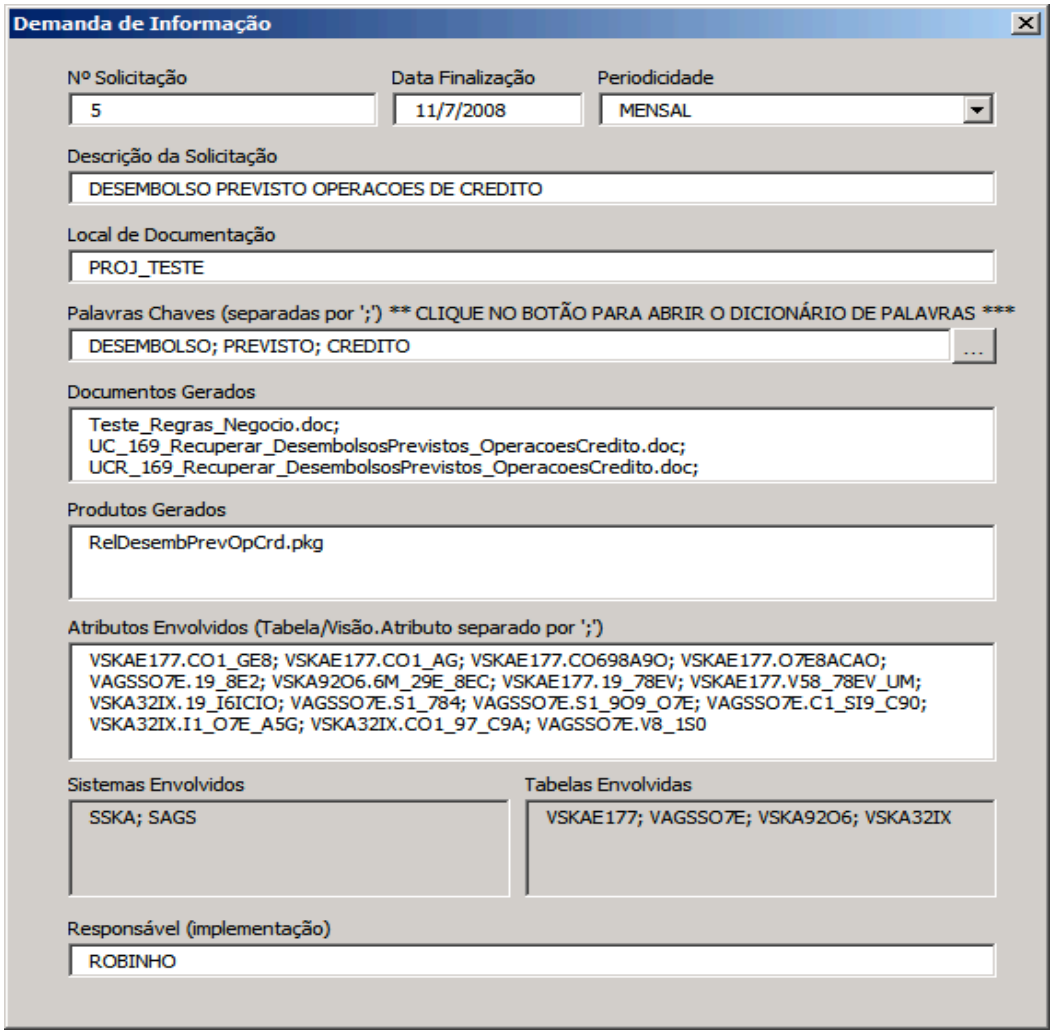

Figura 4 – Tela de cadastro de demandas de informações
Fonte: própria, 2011

3.6 Cadastro das Demandas na Base de Conhecimento

Foram cadastradas 50 demandas, que foram solicitadas à Célula de Recuperações no período de janeiro de 2009 até agosto de 2010. Elas fizeram parte do conjunto de treinamento que durante o processo de implementação do algoritmo auxiliaram na validação das respostas obtidas. As informações sobre o processo de implementação dessas solicitações foram extraídas dos artefatos RUP, scripts SQL e *jobs* de ETL (*Extract Transform Load*) gerados pelo setor de Inteligência de Negócios durante esse período.

3.7 Implementação do Algoritmo de Sugestão de Origens de Informação

Assim com a entrada de dados, a linguagem usada na implementação do algoritmo foi a *Visual Basic* do pacote *Microsoft Visual Studio 6.0*. Basicamente ele consiste em:

1) Entrada de dados onde o usuário informa as palavras-chaves que deseja analisar;
2) Busca textual na base de informações;
3) Armazenamento das ocorrências encontradas;
4) Análise sobre as ocorrências encontradas verificando as informações coincidentes entre elas;
5) Relatório final com as origens encontradas dispostas em percentual de coincidência. Por exemplo, se o usuário informar a palavra "CONTRATO", realizar a busca e o algoritmo encontrar dez demandas, e dessas, oito usam a tabela "VSKAE177", o algoritmo no relatório final irá informar que essa tabela tem 80% de coincidência entre as demandas analisadas.

A lógica detalhada do algoritmo de busca de origens pode ser visualizada no APÊNDICE A – Algoritmo de Sugestão de Origens de Informações em Bancos de Dados.

4 TESTES E RESULTADOS OBTIDOS

Finalizado o fluxo de desenvolvimento, os testes foram realizados em duas etapas distintas: testes com base nas palavras-chaves de demandas já cadastradas na base de conhecimento; e testes com base em palavras-chaves de demandas ainda não cadastradas na base de conhecimento.

O teste com palavras-chaves de demandas já cadastradas serviu para validar se as sugestões do algoritmo estavam corretas e se poderiam ser melhoradas. Foram submetidas palavras-chaves de dez (10) demandas e com base nos resultados obtidos foi possível verificar falhas no algoritmo inicial e fazer melhorias, sendo as principais:

- **Descartar da análise palavras com menos de 3 letras**: o uso de alguns artigos da língua portuguesa fazia o número de ocorrências de demandas aumentarem sem necessariamente elas terem alguma relação entre si.

- **Se forem informadas mais de uma palavra, não serão pesquisadas palavras isoladas**: quando o algoritmo recebe as palavras ele as visualiza como um número binário com combinações da ordem de 2^n, onde n é a quantidade de palavras. Por exemplo, se forem informadas as palavras CREDITO, RURAL e EFETIVO, as combinações seriam: CRÉDITO; RURAL; EFETIVO; CREDITO e RURAL; CREDITO e EFETIVO; RURAL e EFETIVO; CREDITO, RURAL e EFETIVO. Com a melhoria a pesquisa das combinações isoladas CRÉDITO; RURAL; EFETIVO seria desconsiderada. Se fosse informada apenas a palavra CREDITO é que seria feita a pesquisa isolada. Essa alteração foi feita pelo mesmo motivo da primeira.

Após a implementação dessas melhorias, os testes foram repetidos com o mesmo conjunto de exemplos. Foi verificado que ao submeter as palavras-chaves de uma determinada demanda ao algoritmo, o relatório final apresentou a própria demanda e outras que também faziam uso das mesmas palavras, sendo que dessa vez o número de ocorrências foi inferior. As sugestões de origens propostas no relatório também foram corretas para as demandas listadas, chegando em índices de coincidência de tabelas acima de 80%. As figuras 5 e 6 mostram uma das demandas e o relatório resultante da pesquisa de suas palavras-chaves usadas no teste após as melhorias.

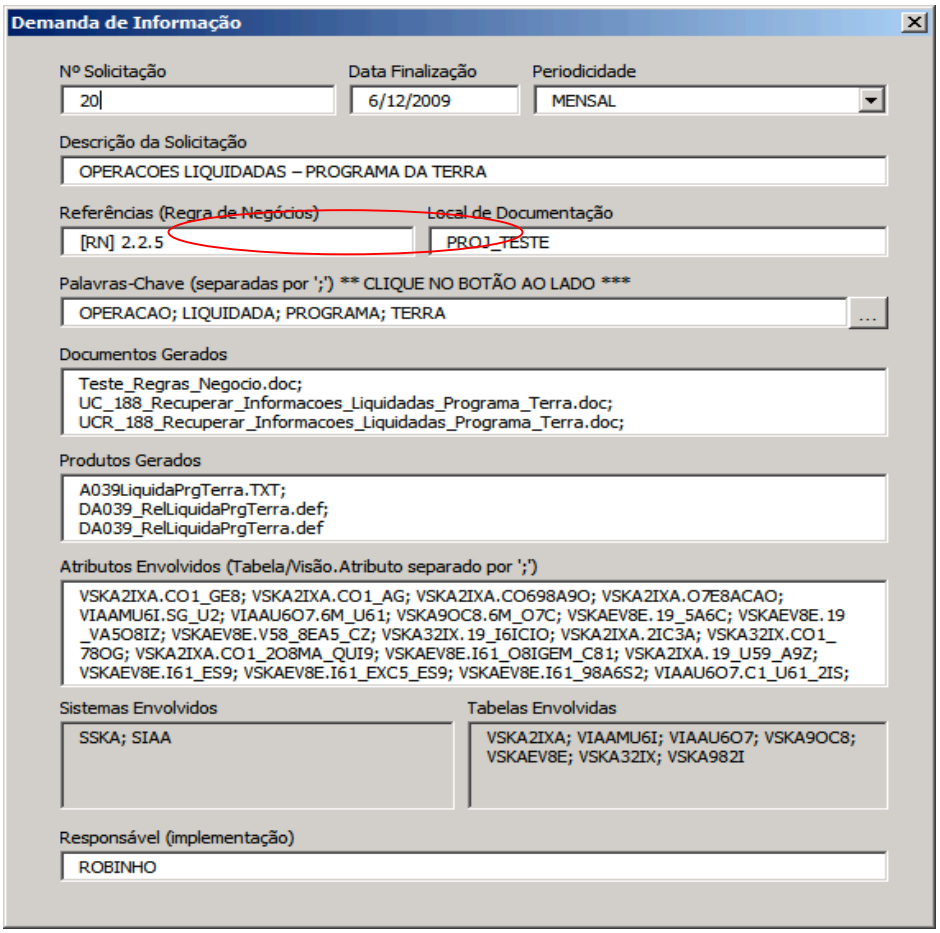

Figura 5 – Demanda "Operações Liquidadas Programa da Terra"
Fonte: própria, 2011

Palavra(s) Analisada(s)	*OPERACAO;LIQUIDADA;PROGRAMA;TERRA*

Todas as solicitações envolvidas:
29; 20; 6

Todos os sistemas envolvidos nas solicitações analisadas:
SSKA; SAGS; SIAA

Sistemas coincidentes em:	
Até 20% das solicitações	
Acima de 20% e até 40% das solicitações	
Acima de 40% e até 60% das solicitações	
Acima de 60% e até 80% das solicitações	
Acima de 80% e inferior a 100% das solicitações	*SIAA*
Em 100% das solicitações	*SSKA*

Todas as tabelas envolvidas nas solicitações analisadas:
VSKAO78A; VAGSSO7E; VIAAMU6I; VIAAU6O7; VIAA7ESS; VSKA92I6; VAGSC982; VSKA982I; VSKAEV8E; VSKA32IX; VSKA2IXA; VSKA9OC8

Tabelas coincidentes em:	
Até 20% das solicitações	
Acima de 20% e até 40% das solicitações	
Acima de 40% e até 60% das solicitações	
Acima de 60% e até 80% das solicitações	
Acima de 80% e inferior a 100% das solicitações	*VIAAMU6I; VIAAU6O7; VSKA2IXA; VSKA9OC8*
Em 100% das solicitações	*VSKA982I; VSKAEV8E; VSKA32IX*

Figura 6 – Relatório resultante da pesquisa com palavras-chaves de demanda "Operações Liquidadas – Programa da Terra"
Fonte: própria, 2011

Para o teste com palavras-chaves de demandas ainda não cadastradas foram consideradas dez (10) demandas que surgiram durante os meses de setembro e outubro de 2010. Para identificar que palavras-chaves seriam submetidas ao algoritmo foi feita uma leitura do pedido e destacadas as palavras que poderiam caracterizar a demanda. Para que o teste pudesse ser imparcial, essa etapa teve ajuda de um colaborador que não participou do processo de desenvolvimento e do primeiro teste.

Na primeira demanda, o colaborador usou apenas uma palavra para caracterizá-la, o que fez com que o algoritmo retornasse muitas ocorrências e as sugestões tivessem no máximo 20% de coincidência. O colaborador procurou então fazer uma análise mais detalhada do pedido e usou cerca de 4 palavras para caracterizar a demanda. Isso diminuiu o número de ocorrências e aumento o percentual de coincidências de algumas tabelas em até 100%. Nos testes seguintes o colaborador procurou usar pelo menos três (3) palavras-chaves.

No entanto, em algumas demandas houve um número de percentual de coincidências inferior a 60% e até a falta de ocorrências. Com base nesses testes, foi possível perceber:

- Análises com menos de três (3) palavras aumentam o número de ocorrências e diminuem o percentual de coincidências de tabelas entre as demandas;
- Análises com mais de dez (10) palavras comprometem a performance da pesquisa;
- Quanto mais demandas forem cadastradas na base de conhecimento, melhor será a resposta do algoritmo. O dicionário de sinônimos também deve ser constantemente atualizado, de maneira que ele possa prever cada vez mais palavras colocando-as no padrão que é usado pela base de conhecimento. Isso evita respostas com nenhuma ocorrência e baixos índices de coincidências;
- A escolha correta das palavras-chaves tanto no cadastro da demanda na base de conhecimento como no momento de fornecer as palavras para o algoritmo aumentam as chances de se obter sugestões compatíveis com a realidade.

Um ponto de destaque para o teste com demandas não cadastradas, foi o resultado positivo das sugestões de origens propostas pelo algoritmo. Dos dez (10) exemplos submetidos, apenas três (3) não obtiveram resultado, mas foi constatado que isso ocorreu devido às palavras fornecidas não terem sido usadas por nenhuma das demandas cadastradas na base de conhecimento. Dos sete (7) que tiveram resultado, apenas dois (2) tiverem índices de coincidência inferiores a 60%, devido a algumas palavras fornecidas não estarem cadastradas no dicionário de sinônimos, mas possuírem mesmo significado semântico de palavras-chaves cadastradas na base de conhecimento. Os cinco (5) exemplos restantes

tiveram índices de coincidência a partir de 80%. Isso mostra que mesmo para uma base de conhecimento com apenas cinquenta (50) demandas cadastradas, em um teste com dez (10) demandas nunca cadastradas, foi possível obter um resultado satisfatório em 50% dos casos. Esse teste confirmou a consistência do algoritmo desenvolvido, sendo as falhas encontradas ocasionadas pelo nível de qualidade e maturidade da base de conhecimento.

5 CONCLUSÃO

Esse trabalho se propunha desenvolver um algoritmo de sugestão de origens de informações em bases de dados com base na hipótese que determinadas palavras da linguagem natural usadas pelo usuário final têm relação direta com sistemas, tabelas e atributos em bases de dados.

O algoritmo foi desenvolvido e testado, sendo seu produto final considerado satisfatório, o que confirmou a hipótese para esse estudo de caso. Apesar disso, vale ressaltar a importância da qualidade das informações cadastradas e a maturidade da base de conhecimento, pois são de fundamental importância nas respostas obtidas com o algoritmo.

Uma melhoria que o algoritmo poderia ter diz respeito ao seu processo de busca. Assim como é feito em sites buscadores famosos, ao informar uma sequência de palavras entre de aspas (“”), só deverão ser listadas demandas em que todas as palavras da sequência apareçam. A implementação dessa melhoria bem como a aplicação desse algoritmo em outras instituições são desdobramentos para futuros trabalhos.

Baseado nos resultados obtidos, é possível supor que a aplicação dessa solução no setor de Inteligência de Negócios da instituição financeira onde se deu a pesquisa certamente unificaria o conhecimento, reduziria a dificuldade de colaboradores menos experientes e daria mais celeridade ao processo de geração das informações.

REFERÊNCIAS

CRUZ, Tadeu. **Gerência do Conhecimento**. 2.ed. Rio de Janeiro: E-papers, 2007.

DAVENPORT, Thomas; PRUSAK, Laurence. **Conhecimento Empresarial:** como as organizações gerenciam o seu capital intelectual. Rio de Janeiro: Campus, 1998.

PERILLO, Mara. **O Conceito de Gestão do Conhecimento**. Disponível em: <http://www.administradores.com.br/informe-se/artigos/o-conceito-de-gestao-do-conhecimento/32153/>. Acesso em: 01 out. 2010.

REZENDE, Solange Oliveira. **Sistemas Inteligentes:** fundamentos e aplicações. Barueri, SP: Manole, 2005.

SILBERSCHATZ, Abraham; KORTH, Henry; SUDARSHAN. **Sistema de Banco de Dados**. 5.ed. São Paulo: Campus, 2006.

WIKIPÉDIA. **Enciclopédia Livre**. Disponível em: <http://pt.wikipedia.org/wiki/P%C3%A1gina_principal>. Acesso em: 01 out. 2010.

APÊNDICE A – ALGORITMO DE SUGESTÃO DE ORIGENS DE INFORMAÇÕES EM BANCOS DE DADOS

ALGORITMO *SUGERE_ORIGENS_INFORMACOES*

Declaração de Variáveis

Variáveis do tipo TEXTO
strFrase, strCaracter, strPalavra, strCombinacao, strTodasSolicitacoes, strTodosSistemas, strTodasTabelas, strBinario, strAte20Sis, strAcima20Ate40Sis, strAcima40Ate60Sis, strAcima60Ate80Sis, strAcima80Inferior100Sis, strTotal100Sis, strAte20Tab, strAcima20Ate40Tab, strAcima40Ate60Tab, strAcima60Ate80Tab, strAcima80Inferior100Tab, strTotal100Tab

Variáveis do tipo TEXTO (vetores)
vetPalavras, vetPalavrasFinal, vetSolicitacoes, vetTabelas, vetSistemas, vetClassificacao

Variáveis do tipo NÚMERO
nrPos, nrTamanho, nrIndice, nrIndice2, nrPrxIndice, nrQtPalavras, nrQtCombinacoes, nrInicio, nrFim, nrPotencia, nrNumPalavras

Variáveis do tipo RECORDSET
qryDemandas

Inicio do procedimento

strFrase = *(Recebe palavras fornecidas pelo usuário)*

SE strFrase <> "" ENTÃO

  strFrase = PadronizaPalavras(1)(UCase(TiraAcentos(2)(strFrase))) //*UCase coloca as palavras em caixa alta*
  strFrase = Replace(strFrase, " ", "") //*Replace substitui os espaços em branco por vazio*

  nrTamanho = Len(strFrase) //*Retorna o tamanho da palavra*
  strPalavra = ""
  nrPos = 1
  nrIndice = 1

  //*Preenche vetor de palavras a serem analisadas*
  PARA nrPos = 1 ATÉ nrTamanho + 1 //*Repete a ação "nrTamanho + 1" número de vezes*

    strCaracter = Mid(strFrase, nrPos, 1) //*Mid retona a partir da posição nrPos um (1) caractere da palavra "strFrase".*
    SE strCaracter = ";" Or strCaracter = "" ENTÃO //*É espaço vazio ou final da frase*

```
        SE Len(strPalavra) > 2 ENTÃO //Não analisa palavras com menos de 3 caracteres
            vetPalavras(nrIndice) = strPalavra
        FINAL DO SE
        strPalavra = ""
        nrIndice = nrIndice + 1
    SENÃO
        strPalavra = strPalavra & strCaracter //"&"Concatena strPalavra e strCaracter
        strCaracter = ""
    FINAL DO SE

FINAL DO PARA

nrIndice = 1
nrIndice2 = 1
strPalavra = ""
nrQtPalavras = QuantidadePalavrasVetor(3)(vetPalavras)

SE nrQtPalavras <= 10 ENTÃO //O procedimento só analisa até 10 palavras

    nrPotencia = 2^nrQtPalavras // nrQtPalavras é o número de palavras
    PARA nrIndice = 1 ATÉ nrPotencia //Repete a ação "nrPotencia" número de vezes
        nrNumPalavras = 0
        strCombinacao = ""
        strBinario = Right("0000000000" & DecimalParaBinario(4)(nrIndice), nrQtPalavras)
        //Right retorna "nrQtPalavras" número de caracteres a direta da palavra resultante da concatenação de
        "0000000000" e DecimalParaBinario(4)(nrIndice)
        PARA nrCont = 1 ATÉ nrQtPalavras // Repete a ação "nrQtPalavras" número de vezes
            SE Mid(strBinario, nrCont, 1) <> 0 ENTÃO
                SE strCombinacao = "" ENTÃO
                    strCombinacao = vetPalavras(nrCont)
                SENÃO
                    strCombinacao = strCombinacao & ";" & vetPalavras(nrCont)
                FINAL DO SE
            FINAL DO SE
        FINAL DO PARA
        nrNumPalavras = ContaPalavrasItemVetor(5)(strCombinacao)
        SE nrNumPalavras > 1 ENTÃO //Grava no vetor apenas combinações com mais de uma palavra
            nrPrxIndice = QuantidadePalavrasVetor(vetPalavras) + 1
            vetPalavras(nrPrxIndice) = strCombinacao
        FINAL DO SE
    FINAL DO PARA

    //Preenche o vetor de palavras final (retira palavras isoladas se houver mais de uma palavra)
    nrIndice = 1
    nrQtCombinacoes = QuantidadePalavrasVetor(vetPalavras)
    SE nrQtPalavras > 1 ENTÃO
        PARA nrIndice = (nrQtPalavras + 1) ATÉ nrQtCombinacoes //Repete a ação de
        "nrQtPalavras + 1" até nrQtCombinacoes número de vezes
            vetPalavrasFinal(nrIndice - nrQtPalavras) = vetPalavras(nrIndice)
        FINAL DO PARA
    SENÃO //Se a pesquisa só tiver 1 palavra, o vetor é mantido igual ao inicial
```

```
   PARA nrIndice = 1 ATÉ nrQtCombinacoes
      vetPalavrasFinal(nrIndice) = vetPalavras(nrIndice)
   FINAL DO PARA
FINAL DO SE

//Inicia a localização de cada combinação gravada no vetor
nrQtCombinacoes = QuantidadePalavrasVetor(3)(vetPalavrasFinal)
nrIndice = 1
nrIndice2 = 1
strPalavra = vetPalavrasFinal(nrIndice)

FAÇA ENQUANTO strPalavra <> "" E  NÃO  IsNull(strPalavra) //IsNull diz se a variável está
  nula. Vai repetir a ação para cada combinação até o final do vetor.

    qryDemanda = AbreRecorset("Demandas")
    qryDemanda.MoveFirst //Move para a primeira demanda
    ENQUANTO NÃO qryDemanda.Eof //Eof diz se é o final da lista de demandas
        SE CombinacaoExisteNaDemanda(6)(strPalavra) ENTÃO
                vetSolicitacoes(nrIndice2) = qryDemanda("NumeroSolicitacao")
            vetSistemas(nrIndice2) = qryDemanda.("SistemasEnvolvidos")
            vetTabelas(nrIndice2) = qryDemanda.("TabelasEnvolvidas")
            nrIndice2 = nrIndice2 + 1
        FINAL DO SE
        qryDemanda.MoveNext // MoveNext move para a próxima demanda
    FINAL DO ENQUANTO

    nrIndice = nrIndice + 1
    strPalavra = vetPalavrasFinal(nrIndice)

FINAL DO FAÇA ENQUANTO

//Preenche as varáveis que irão compor a análise final do relatório
strTodasSolicitacoes = AnalisaOcorrencias(7)(vetSolicitacoes, 1) // Recupera todos as solicitações
  envolvidas
strTodosSistemas = AnalisaOcorrencias(7)(vetSistemas, 1) // Recupera todos os sistemas envolvidos
strTodasTabelas = AnalisaOcorrencias(7)(vetTabelas, 1) // Recupera todos as tabelas envolvidas

//Preenche vetor de classificação com as coincidências de SISTEMAS e depois cada variável de percentual de
coincidência
vetClassificacao = AnalisaOcorrencias(7)(vetSistemas, 2)
strAte20Sis = vetClassificacao(1)
strAcima20Ate40Sis = vetClassificacao(2)
strAcima40Ate60Sis = vetClassificacao(3)
strAcima60Ate80Sis = vetClassificacao (4)
strAcima80Inferior100Sis = vetClassificacao(5)
strTotal100Sis = vetClassificacao(6)

//Preenche vetor de classificação com as coincidências de TABELAS e depois cada variável de percentual de
coincidência
vetClassificacao = AnalisaOcorrencias(7)(vetTabelas, 2)
strAte20Tab = vetClassificacao(1)
```

strAcima20Ate40Tab = vetClassificacao(2)
strAcima40Ate60Tab = vetClassificacao(3)
strAcima60Ate80Tab = vetClassificacao (4)
strAcima80Inferior100Tab = vetClassificacao(5)
strTotal100Tab = vetClassificacao(6)

ImprimeRelatório(8)(strTodasSolicitacoes,
strTodosSistemas,
strTodasTabelas,
strAte20Sis,
strAcima20Ate40Sis,
strAcima40Ate60Sis,
strAcima60Ate80Sis,
strAcima80Inferior100Sis,
strTotal100Sis,
strAte20Tab,
strAcima20Ate40Tab,
strAcima40Ate60Tab,
strAcima60Ate80Tab,
strAcima80Inferior100Tab,
strTotal100Tab)

SENÃO

EXIBE A MENSAGEM "Só é permitido análise de até 10 palavras. Diminua a quantidade de palavras e tente novamente."

FINAL DO SE

FINAL DO SE

FINAL DO **ALGORITMO**

***(1) PadronizaPalavras**: Analisa cada palavra fornecida e verifica se está cadastrada no dicionário de sinônimos. Se existe, troca pela palavra padrão usada na base de conhecimento, senão mantém a atual.*

***(2) TiraAcentos**: Retira as acentuações de cada palavra.*

***(3) QuantidadePalavrasVetor**: Conta quantos itens tem um vetor.*

***(4) DecimalParaBinario**: Converte um número decimal para binário.*

***(5) ContaPalavrasItemVetor**: Conta quantas palavras tem um item de vetor.*

***(6) CombinacaoExisteNaDemanda**: Verifica se a combinação existe entre as palavras-chaves de uma determinada demanda, não importando a ordem onde elas se encontram.*

***(7) AnalisaOcorrencias**: Função que analisa as coincidências. No modo "1" são listadas todas as ocorrências, não importando se existem coincidências. No modo "2" as coincidências são listadas em 6 faixas de percentual.*

***(8) ImprimeRelatorio**:Imprime relatório com o resultado da análise.*