5
Busca

Este capítulo apresenta em detalhes a implementação da busca no sistema, mostrando todas as etapas desde o fornecimento e tratamento das palavras-chave de entrada, agrupamento, localização e mapeamento das referências e, por fim, a criação e execução das consultas.

5.1.
Tratamento das Palavras-Chave de Entrada

Quando o usuário submeter uma busca, após a seleção de uma aplicação que será utilizada e a inclusão de palavras-chave no campo texto, o sistema irá:

- Tratar as palavras-chave de entrada identificando se existe algum erro de entrada.
- Agrupar palavras que foram indicadas e que devem ser processadas conjuntamente.
- Encontrar referências destas palavras, ou agrupamento de palavras, nas tabelas internas do sistema
- Exibir uma tabela para que o usuário efetue a associação das palavras de entrada com a sua possível relação com o banco de dados.

O tratamento de erros de entrada é necessário para minimizar problemas lógicos quando se utiliza operadores. O tratamento efetuado foi apenas para identificar um número par de aspas (“) que são usadas para identificar palavras que devem ser processadas agrupadas semelhante ao que ocorre com os mecanismos de buscas na Internet, por exemplo, se a entrada for *marlim sul* o sistema irá tentar encontrar referências para a palavras *marlim* e para a palavra *sul*, porém se a entrada for *“marlim sul”* (agrupadamento) o sistema tentará encontrar referências para ambas as palavras juntas. Caso algum erro seja encontrado uma mensagem será exibida para o usuário efetuar a correção. O

trecho de pseudo-código exibido na Figura 11 mostra como é feito o tratamento da entrada dos dados.

```
/// Trata quando não tiver fechado aspas
protected bool trataEntradaBusca(ref string busca)
 {
    int i;
    int qtdAspas = 0;

    // verifica a quantidade de aspas que existem.
    // Se for impar retorna falso
    i = busca.IndexOf('"', 0);
    while (i != -1)
    {
      qtdAspas++;
      i = busca.IndexOf('"', i + 1);
   }

    // impar
    if (qtdAspas % 2 != 0)
       return false;
    else
       return true;
   }
```

Figura 11: Pseudo-código do Tratamento de Entrada

5.2. Agrupamento de Palavras-Chave

Após o tratamento de possíveis erros, o agrupamento de palavras é feito, juntando as palavras que serão processadas pelo sistema, fazendo que estas sempre trabalhem juntas, por exemplo, “Albacora Leste”. O pseudo-código da Figura 12 mostra a implementação do agrupamento das palavras que estão entre aspas.

```
// cria uma lista de palavras, agrupando as entre aspas
protected List<string> retornaPalavras(string palavras)
{
   // lista final, contendo já as agrupadas
   List<string> entradas = new List<string>();

   // primeiro retira as palavras entre aspas se houverem
```

```
    int posIni = palavras.IndexOf('"');
    int posFim;

    while (posIni != -1)
    {
      // tem que achar pq existe numero par de aspas
      posFim = palavras.IndexOf('"', posIni + 1);

      // se houver palavras entre aspas
   if ((palavras.Substring(posIni+1,posFim-posIni-1)).Trim()!= "")
      // guarda palavra
      entradas.Add(...);

    // remove a palavra encontrada
    palavras = palavras.Remove(posIni, posFim - posIni + 1);

    // encontra a proxima palavra
    palavras = palavras.Insert(posIni, " ");

    posIni = palavras.IndexOf('"');
  }

  // pega o restante das palavras
  string[] palavrasAux = palavras.Split(' ');

  // guarda as demais palavras
  for (int i = 0; i < palavrasAux.Length; i++)
     if (palavrasAux[i] != "")
       entradas.Add(palavrasAux[i]);

   return entradas;
 }
```

Figura 12: Pseudo-código do Agrupamento de Palavras entre Aspas

5.3. Localização e Mapeamento das Referências

Com as palavras definidas, a etapa de encontrar as referências é inicializada. Esta etapa é muito importante, pois será responsável por encontrar associações entre as palavras de entrada com as tabelas ou conteúdo do banco de dados. Uma palavra de entrada pode ser associada a uma tabela ou coluna ou a um conteúdo diretamente. Por exemplo, é possível buscar *produção do campo de marlim* que está associada à produção (tabela) do campo de petróleo (coluna da tabela) cujo nome é marlim (conteúdo da tabela). O sistema tenta identificar associações de três maneiras:

- Associação com o nome de uma coluna da base dados
    - Quando é possível encontrar uma palavra que seja igual ao nome ou parte do nome de uma coluna do banco de dados é identificada uma associação. Como muitos nomes de colunas de esquemas de bancos de dados procuram ter um significado real associado ao seu conteúdo, esta foi uma maneira simples e rápida de tentar encontrar uma associação. O problema é que muitas vezes a coluna da tabela é codificada e não reflete em nada o significado do seu conteúdo acarretando a não identificação de possíveis associações.

- Associação com a descrição de uma coluna do banco de dados
    - Visando contornar o problema encontrado na associação com o nome da coluna, é feita também uma busca na descrição das colunas no catálogo do banco de dados para tentar encontrar referências. No momento da inclusão das tabelas e colunas na parte administrativa, também é possível complementar esta descrição adicionando maiores informações além das existentes nos comentários. Desta maneira é possível muitas vezes encontrar nas descrições as palavras que o usuário forneceu na entrada, identificando assim uma associação. É muito importante colocar uma boa descrição nas colunas das tabelas para que esta associação seja feita de maneira correta.

- Associação com o conteúdo de uma coluna do banco de dados
    - Busca direta dentro do conteúdo da coluna. Como já informado na parte administrativa, o conteúdo das tabelas do usuário são copiados para as tabelas do sistema visando melhorar a velocidade de busca. Nesta associação, a palavra de entrada será comparada com o conteúdo das colunas das tabelas diretamente tentando encontrar uma possível associação. Quando esta for encontrada na tabela de possíveis associações será indicado como conteúdo. Contrastando com as duas associações anteriores que fazem referência a colunas das tabelas, esta é a única que faz associação a conteúdo.

A figura 13 mostra o exemplo de uma busca a uma base de publicações por livros de comédia escritos por um determinado autor. Foram colocadas no campo de busca as palavras: *livros comédia Leandro*. O sistema mostrará uma tela de possíveis associações com as palavras de entrada.

O usuário deve selecionar o que ele deseja associar, verificando se a associação deve ser feita ao conteúdo da coluna ou não. Neste caso (figura 13), a palavra *comédia* está associada ao conteúdo de uma tabela de categorias (observe coluna conteúdo marcado como *True*); igualmente a palavra *leandro* está associada ao nome do autor. Neste exemplo, apenas uma associação de cada palavra foi encontrada. Porém, é possível que sejam mostradas mais de uma associação.

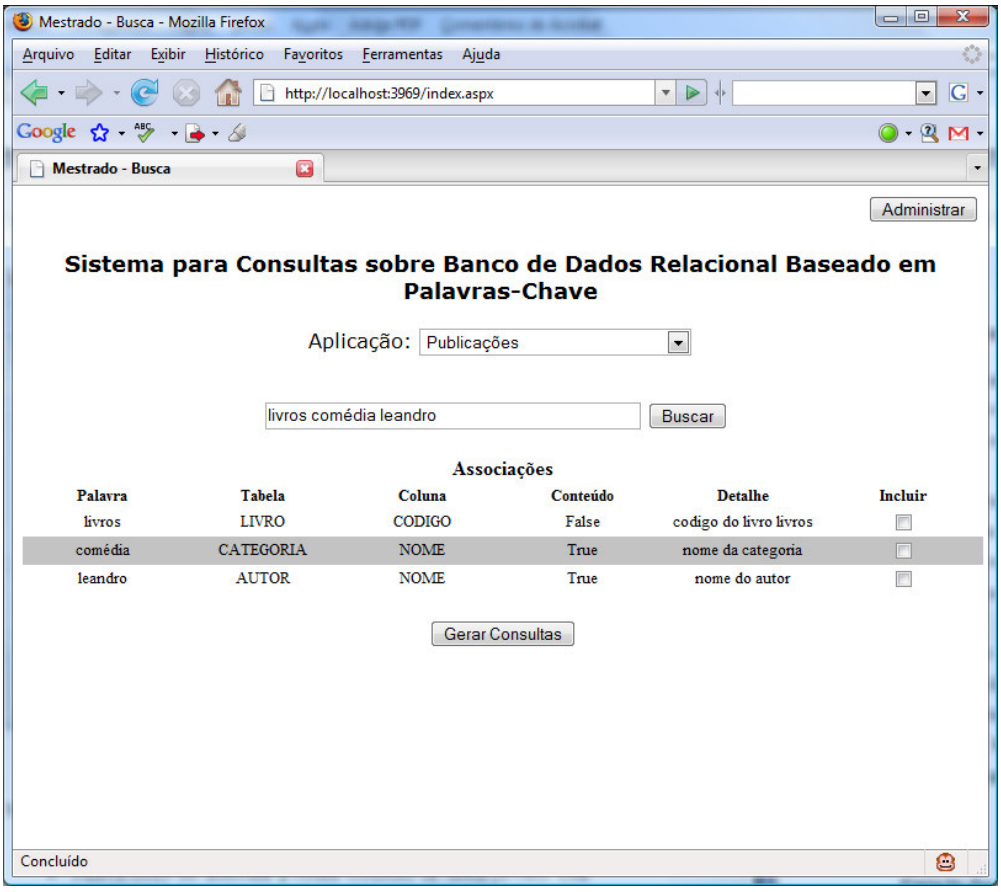

Figura 13: Tela de associação das palavras

Na tela da Figura 13, as seguintes associações são apresentadas:

- A palavra *livros* foi associada à coluna CODIGO da tabela LIVRO. Esta associação foi feita à coluna e não ao conteúdo da mesma.

- A palavra *comédia* foi associada à coluna NOME da tabela CATEGORIA. Esta associação foi feita com o conteúdo da tabela e não com a coluna diretamente. Neste caso realmente a palavra *comédia* é um valor da coluna.

- A palavra *leandro* foi associada à coluna NOME da tabela AUTOR. Esta associação foi feita com o conteúdo da tabela e não com a coluna diretamente. Neste caso, realmente a palavra *Leandro* é um valor da coluna.

O trecho de pseudo-código da figura 14 ilustra a implementação de como encontrar as referências para cada palavra, isto é, localizar a palavra na descrição, nome ou conteúdo das colunas (valor). Para cada entrada (palavras) são efetuadas consultas para tentar encontrar referências de nome ou valores de coluna, caso encontre alguma, adiciona em uma lista de referências.

```
/// Encontra referencias para as palavras entradas
/// Estas referencias podem ser na descrição das colunas, nome das
/// colunas ou conteudo das colunas (valor)
/// Para cada encontrada cria uma instancia da classe palavra e
/// adiciona suas referencias
void encontraReferencias(List<string> entradas)
{
  palavras.Clear();

  gerenteBD = new GerenteBD("oracle");

  DataSet ds;
  DataSet ds2;

  // para cada palavra
  for (int i = 0; i < entradas.Count; i++)
  {
    // ver se encontra alguma coluna de referencia
    ds = gerenteBD.executaConsulta
      ("..consulta encontrar referencia para coluna..");
```

```
    // ver se encontra algum VALOR de coluna de referencia
    ds2 = gerenteBD.executaConsulta
      ("..consulta encontrar referencia para valor coluna.. ");

      // se encontrou alguma cria a palavra
      if (ds.Tables[0].Rows.Count || ds2.Tables[0].Rows.Count)
      {
        palavras.Add(new Palavra(entradas[i]));

        // para cada referencia de coluna adiciona
        // false indica que é referencia de coluna
        for (int j = 0; j < ds.Tables[0].Rows.Count; j++)
          palavras[palavras.Count-1].referencias.Add(..., false));

        // para cada referencia de valor de coluna adiciona
        // true indica que é referencia de valor de coluna
        for (int j = 0; j < ds2.Tables[0].Rows.Count; j++)
          palavras[palavras.Count-1].referencias.Add(..., true));

        }
    }
}
```

Figura 14: Pseudo-código da Busca de Referências para as Palavras

5.4. Criação das Consultas

Após efetuar a seleção das referências é possível "Gerar Consultas" para que seja exibido, além do comando SQL, o conteúdo das dez primeiras linhas de execução, permitindo assim o usuário avaliar se existe algum resultado relevante. Neste momento, uma consulta é realizada no banco de dados de origem para retornar as primeiras linhas da consulta.

Para a geração da consulta, o sistema identifica as associações selecionadas pelo usuário e cria uma lista de palavras e suas associações com o banco de dados. Ao final deste processo, existirá uma lista de palavras e, para cada palavra, uma possível lista de associações marcadas. Para uma palavra pode haver mais de uma associação (significado). O trecho de pseudo-código apresentado na figura 15 apresenta como encontra a relação entre cada palavra e os objetos no banco de dados.

```
// lista de palavras selecionadas
palavrasSelecionadas = new List<Palavra>();

// verifica quais foram marcados
RepeaterItemCollection itens = rpAssociacao.Items;

// para cada item
for (int i = 0 ; i < itens.Count; i++)
{
  // recupera o item
  CheckBox chk = (CheckBox)(itens[i].FindControl("chkIncluir"));

  // se tiver sido marcado
  if (chk.Checked)
  {
    // encontra palavra e relacao
    // coloca em palavrasSelecionadas
    for (int k = 0; k < palavras.Count; k++)
    {
      // para cada associação da lista de palavra inicial
      for (int l = 0; l < palavras[k].referencias.Count; l++)
      {
        // verifica se é a associação marcada
        if (..se for a associacao..)
        {
          // caso tenha encontrado
          bool achou = false;

          // verifica se ja nao existe esta palavra (associação)
          ...

          // se nao houver adiciona em palavras selecionadas
          if (!achou)
          {
              // adiciona a palavra e a referencia
          }
          // caso contrario adiciona apenas a referencia
          else
          {
              // adiciona apenas a referencia

          }
        }
      }
    }
  }
}
```

Figura 15: Pseudo-código Cria Relação Palavras e Banco de Dados

Após a criação da lista com as relações, o sistema tentará gerar o maior número de consultas possível para todas as combinações de palavras e associações selecionadas pelo usuário. Assim, se houverem 3 palavras e cada palavra tiver 3 associações, será possível fazer até 27 combinações de palavras e associações,

pois a palavra 1 pode ter três possíveis resultados, a palavra 2 mais três possíveis resultados e a palavra 3 mais três possíveis resultados (3x3x3). O pseudo-código na figura 16 apresenta como o número possível de consultas é gerado.

```
// quantidade de palavras seleciopnadas
int qtdPalavras = palavrasSelecionadas.Count;

// calcula a quantidade de combinações possíveis
// multiplicando as referencias de cada palavra
for (int i = 0; i < qtdPalavras; i++)
{
  if (combinacoes != 0)
    combinacoes *= palavrasSelecionadas[i].referencias.Count;
  else
    combinacoes = palavrasSelecionadas[i].referencias.Count;
}
```

Figura 16: Pseudo-código Cria Quantidade de Possíveis Consultas

Vale lembrar que isto não quer dizer que existirão 27 consultas, mas sim possíveis consultas. Para cada combinação, será verificado se é possível gerar uma consulta, pois em muitos casos pode não haver relação entre as palavras selecionadas.

Para efetuar este processo, o sistema inicialmente irá gerar uma matriz contendo todas as associações possíveis. Cada linha desta matriz identifica uma palavra selecionada, cada coluna uma referência possível e o conteúdo de cada célula, o código das referências de cada palavra para formar uma associação.

Por exemplo, imagine a associação apresentada na figura 17:

Código	Palavra	Código	Referência
0	Palavra1	0	Tabela1.Coluna1
		1	Tabela2.Coluna1
1	Palavra2	0	Tabela1.Coluna1
		1	Tabela3.Coluna1
2	Palavra3	0	Tabela1.Coluna1

Figura 17: Associação entre palavras e referências

Será gerada uma matriz como ilustra as duas tabelas na figura 18:

		0	1	2	3
Código da Palavra	0	1	1	0	0
	1	1	0	1	0
	2	0	0	0	0

Palavra	**Referência**			
Palavra1 (0)	Tabela2. Coluna1 (1)	Tabela2. Coluna1 (1)	Tabela1. Coluna1 (0)	Tabela1. Coluna1 (0)
Palavra2 (1)	Tabela3. Coluna1 (1)	Tabela1. Coluna1 (0)	Tabela3. Coluna1 (1)	Tabela1. Coluna1 (0)
Palavra3 (2)	Tabela1. Coluna1 (0)	Tabela1. Coluna1 (0)	Tabela1. Coluna1 (0)	Tabela1. Coluna1 (0)

Figura 18: Matriz de associação

Como discutido anteriormente, cada coluna apresenta um conjunto de associações que pode gerar uma consulta. Neste caso, a matriz da Figura 18 permite tirar as seguintes associações:

1) “palavra1” (linha 0) associado a “tabela2.coluna1” (conteúdo de 0,0)
“palavra2” (linha 1) associado a “tabela3.coluna1” (conteúdo de 1,0)
“palavra3” (linha 2) associado a “tabela1.coluna1” (conteúdo de 2,0)

2) “palavra1” (linha 0) associado a “tabela2.coluna1” (conteúdo de 0,1)
“palavra2” (linha 1) associado a “tabela1.coluna1” (conteúdo de 1,1)
“palavra3” (linha 2) associado a “tabela1.coluna1” (conteúdo de 2,1)

3) “palavra1” (linha 0) associado a “tabela1.coluna1” (conteúdo de 0,2)
“palavra2” (linha 1) associado a “tabela3.coluna1” (conteúdo de 1,2)
“palavra3” (linha 2) associado a “tabela1.coluna1” (conteúdo de 2,2)

4) “palavra1” (linha 0) associado a “tabela1.coluna1” (conteúdo de 0,3)
“palavra2” (linha 1) associado a “tabela1.coluna1” (conteúdo de 1,3)
“palavra3” (linha 2) associado a “tabela1.coluna1” (conteúdo de 2,3)

O pseudo-código apresentado na figura 19 ilustra como a matriz de associações é montada a partir das palavras e suas referências.

```
// foi definido como frase uma possível sequencia de combinações
int[,] frases = new int[qtdPalavras,combinacoes];

// tamanho máximo de uma sequencia
int[] maxPalavra = new int[qtdPalavras];

// inicializa a primeira palavra com maximo dos indices
for ( int i = 0 ; i < qtdPalavras ; i++ )
{
  maxPalavra[i] = palavrasSelecionadas[i].referencias.Count-1;
  frases[i,0] = maxPalavra[i];
}
bool conseguiu = false;
// para cada combinação
for (int i = 1; i < combinacoes; i++)
{
  // copia a entrada anterior para a atual
  for ( int j = 0 ; j < qtdPalavras ; j++ )
    frases[j,i] = frases[j,i-1];
  frases[qtdPalavras - 1, i] = frases[qtdPalavras - 1, i] - 1;

  // se conseguir decrementar otimo, se nao, faz o "vai um"
  if ( frases[qtdPalavras-1,i] < 0 )
  {
    frases[qtdPalavras-1,i] = maxPalavra[qtdPalavras-1];

    conseguiu = false;

    for ( int j = qtdPalavras-2 ; j >=0 && !conseguiu ; j-- )
    {
      frases[j,i] = frases[j,i]-1;

      if ( frases[j,i] >= 0 )
        conseguiu = true;
      else
        frases[j,i] = maxPalavra[j];
    }
  }
}
```

Figura 19: Montagem da Matriz de Associação

A partir desta matriz, o sistema inicia uma busca nas tabelas internas para tentar fazer uma consulta que contemple cada caso. Para cada seqüência, o sistema identifica as referências entre as tabelas envolvidas na possível consulta, tentando relacionar todas as tabelas para que uma consulta possa ser válida.

Uma consulta será válida quando for possível relacionar todas as tabelas dos termos envolvidos. Caso não seja possível criar um caminho de associação entre as tabelas, o sistema indica que não foi possível gerar uma consulta com este conjunto de associações.

Um caminho é uma ligação entre todas as tabelas. Não é preciso ter ligação de todas as tabelas para todas as outras, mas ao menos que todas estejam interligadas para assim gerar uma consulta válida. O pseudo-código apresentado na figura 20 ilustra como é verificado se existe um "caminho" entre todas as tabelas envolvidas em uma possível consulta.

```
/// Verifica se existe ligacao entre todas as tabelas
/// Não pode haver tabelas isoladas, se houver, não existe caminho
bool encontraCaminho(int[,] encontrado, int[] passou, int x)
{
  int i;
  passou[x] = 1;

  // perorre ate encontrar uma relacao (1)
  for (i = 0; i < Math.Pow(encontrado.Length, 0.5); i++)
    if (encontrado[x, i] == 1 && passou[i] == 0)
      encontraCaminho(encontrado, passou, i);

  // verifica se passou por todos quando voltar
  if (x == 0)
  {
    for (i = 0; i < passou.Length && passou[i] == 1; i++);

    if (i == passou.Length)
      return true;
    else
      return false;
  }

  return false;
}
```

Figura 20: Verifica se Existe Caminho Entre Tabelas

O trecho de código apresentado na figura 21 verifica, para cada possível consulta, as relações entre tabelas e, para cada resultado, se existe um caminho entre tabelas (figura 20). Se houver este caminho, a consulta é considerada como válida, sendo executada e o seu resultado apresentado para avaliação do usuário. Todas estas informações são consultadas utilizando as tabelas do sistema, pois no momento da inclusão das tabelas as referências entre elas também são armazenadas, não sendo necessário consultar o banco de dados de origem.

```
string tabela;
string coluna;

// para cada "frase"
for (int i = 0; i < combinacoes; i++)
{
  int[,] encontrado = new int[qtdPalavras,qtdPalavras];
  string relacao = "";
  string associacao = "";
  string colunasSQL = "";
  string tabelasSQL = "";
  string tabelasUtilizadas = "";

  // para cada REFERENCIA ENCONTRADA daquela frase
  for (int j = 0; j < qtdPalavras; j++)
  {
    tabela =
palavrasSelecionadas[j].referencias[frases[j, i]].tabela;
    coluna =
palavrasSelecionadas[j].referencias[frases[j, i]].coluna;

    // verifica se tabela já encontrada, não precisa procurar
    if (tabelasUtilizadas.IndexOf(tabela) != -1)
    {
      // marca no vetor encontrado que guarda as relacoes
      // que esta tabela nao precisa ter relacao
      for (int k = 0; k < qtdPalavras; k++)
        encontrado[j, k] = -1;
    }
    else
    {
      // coloca a tabela como utilizada
      tabelasUtilizadas += tabela + " ";

      // procura as tabelas associadas a esta tabela
      // e verifica se as outras encontram-se ali
      ds = gerenteBD.executaConsulta("..consulta..");

     // verifica se as tabelas que fazem relacao
     // estao e marca em ENCONTRADO
     for (int k = 0; k < ds.Tables[0].Rows.Count; k++)
     {
       for (int l = 0; l < qtdPalavras; l++)
       {
         // encontrou uma referencia para uma tabela da frase
```

```
          if ( ..se houver referencia..)
          {
            encontrado[j, l] = 1;
            encontrado[l, j] = 1;
            relacao += (ds.Tables[0].Rows[k]["TAB1"].ToString() +
"." + ds.Tables[0].Rows[k]["COL1"].ToString() + " = " +
ds.Tables[0].Rows[k]["TAB2"].ToString() + "." +
ds.Tables[0].Rows[k]["COL2"].ToString()) + " AND ";
          }
        }
      }
    }

    // se a referencia for com o conteudo
    if (palavrasSelecionadas[j].referencias[xx].conteudo)
      relacao += "UPPER(" + tabela + "." + coluna + ") = UPPER('"
+ palavrasSelecionadas[j].palavra + "') AND ";

      // se não tiver adicionado a coluna na consulta
      if (colunasSQL.IndexOf(tabela + "." + coluna) == -1)
      {
        colunasSQL += tabela + "." + coluna + ", ";
        associacao += palavrasSelecionadas[j].palavra + " = "
           + tabela + "." + coluna + ", ";
      }
      // se não tiver adicionado a tabela
      if (tabelasSQL.IndexOf(tabela) == -1)
        tabelasSQL += tabela + ", ";
    }

    // se houver relacao
    if (relacao.Length > 0)
    {
      int[] passou = new int[qtdPalavras];

      // verificar se existe caminho, isto é relacao entre todos
      bool todos = encontraCaminho(encontrado,passou,0);

      // consulta valida, guarda consulta
      if (todos)
      {
        DataRow auxRow = dtResultado.NewRow();

        auxRow["Palavras"] =
             associacao.Substring(0, associacao.Length - 2);

      auxRow["Consulta"] =
      "SELECT " + colunasSQL.Substring(0, colunasSQL.Length - 2) +
       " FROM " + tabelasSQL.Substring(0, tabelasSQL.Length - 2) +
          " WHERE " + relacao.Substring(0, relacao.Length - 5) +
        " AND ROWNUM < 10 ";

         dtResultado.Rows.Add(auxRow);
      }
    }
  }
```

Figura 21: Geração das Consultas Válidas

5.5.
Execução das Consultas

Para geração da prévia do resultado da consulta será necessário efetuar uma consulta ao banco de dados de origem. Até agora apenas as tabelas do sistema foram consultadas para gerar as consultas. Porém para exibição do resultado, uma conexão com a base de origem torna-se necessária.

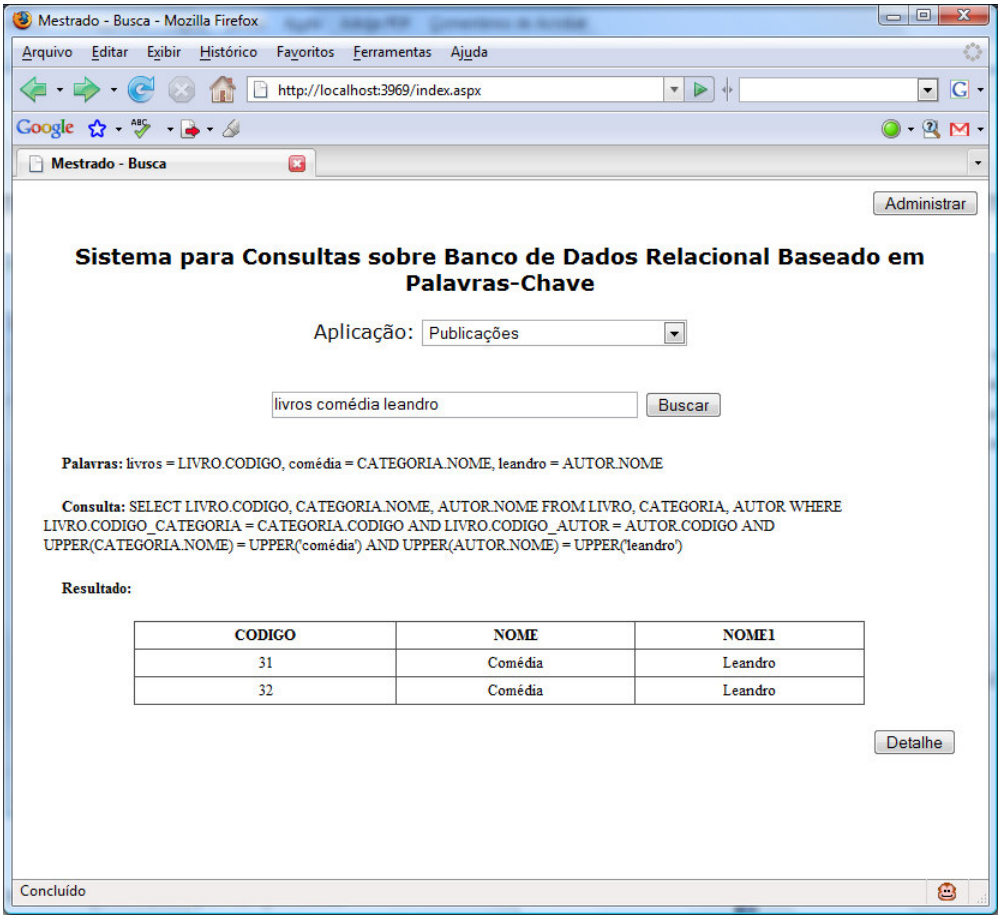

Figura 22: Interface de consultas geradas

A figura 22 mostra que foi gerada apenas uma consulta para as palavras e associações efetuadas pelo usuário. Uma prévia da consulta é apresentada mostrando que foram encontrados dois livros de comédia do autor Leandro. Pode-se notar que apenas o código do livro (31 e 32) foi apresentado, porém o seu nome não apareceu. Isto ocorre pois não foi feita nenhuma associação com o nome do livro.

Uma maneira de exibir o nome do livro seria incluir a palavra *nome* também na busca e associar a mesma com a tabela *livro* e com a coluna *nome*. O capítulo de Trabalhos Futuros inclui uma sugestão para permitir incluir colunas de tabelas que já fazem parte da consulta.

Para que seja executada integralmente uma consulta basta apertar no botão detalhes que a base de origem será acessada retornando assim toda a consulta.

Não está sendo utilizado nenhum critério para ordenar os resultados. Todas as possíveis consultas em função das palavras de entrada e das associações são apresentadas para que o usuário selecione o que deseja visualizar. Também não estão sendo computadas quais consultas foram selecionadas como corretas para um determinado conjunto de palavras de entrada. Estas poderiam ser possíveis futuras implementações abordadas, no capítulo 6, no tópico trabalhos futuros.