Fernanda Bessani Leite Penteado

Método de Filtragem Fuzzy Para Avaliação de Bases de Dados Relacionais

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, sendo parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Ivan Nunes da Silva

São Carlos

2009

“Porque *dele* e por meio *dele*, e para *ele*, são todas as coisas. Glória, pois a *ele* eternamente. Amém.”

Romanos 11, 36

Agradecimentos

Agradeço primeiramente a Deus por me dar força e capacidade necessária para a realização desse trabalho.

Ao meu marido Marcos pelo suporte e carinho nos momentos mais difíceis, e por sempre me fazer sentir mais confiante. Também agradeço meu filho Daniel por me alegrar nos momentos de tensão.

Agradeço aos meus pais, João Marcos e Vilma, pela educação e investimento cedidos a mim e o carinho com que me aconselham e ajudam, e ao meu irmão Roman pelas conversas e companhia sempre presente.

Ao meu primo Valmir Ziolkowski por me incentivar, ajudar e abrir portas para que eu ingressasse nessa jornada.

Aos colegas, professores e funcionários da EESC que de alguma forma me auxiliaram.

Finalmente, um agradecimento especial ao meu orientador Prof. Dr. Ivan Nunes da Silva pelo tempo e paciência dedicados, o conhecimento fornecido e a confiança em mim depositada.

SUMÁRIO

Resumo

PENTEADO, F. B. L. (2009). *Método de Filtragem Fuzzy Para Avaliação de Bases de Dados Relacionais*. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, 2009.

As informações imprecisas e vagas, comumente encontradas na modelagem de problemas do mundo real, muitas vezes não são manipuladas de forma adequada por meio das consultas convencionais aos bancos de dados. Alternativamente, a teoria de conjuntos *fuzzy* tem sido considerada uma ferramenta bem promissora para tratamento destas informações consideradas imprecisas e, em determinados casos, até mesmo ambíguas. Esse trabalho utiliza a linguagem SQL padrão para apresentar uma abordagem *fuzzy* de consultas a bancos de dados relacionais. Estudos de casos referentes à aplicabilidade do método desenvolvido são apresentados a fim de mostrar as suas potencialidades em relação aos métodos tradicionais de consultas.

Palavras Chave: Banco de Dados Relacional, Lógica *Fuzzy*, Linguagem SQL.

Abstract

PENTEADO, F. B. L. (2009). *Fuzzy Filtering Method for Evaluation of Relational Databases.* Dissertation (Master's Degree) – Escola de Engenharia de São Carlos, Universidade de São Paulo, 2009.

Often, the imprecise and vague information, commonly found in the modeling of real world problems, are not dealt in an appropriate way through conventional queries used in databases. Alternatively, the fuzzy set theory has been considered a very promising tool to treat imprecise and ambiguous information. This work uses the standard SQL language and fuzzy set theory to develop a fuzzy query method for relational databases. Simulation examples are presented to illustrate its potentialities in relation to the traditional query methods.

Keywords: Relational Database, Fuzzy Logic, SQL Language.

Lista de Siglas e Abreviaturas

ANSI	*American National Standards Institute*
API	*Application Programming Interfaces*
CDA	Centro de Área
DDL	*Data Definition Language*
DML	*Data Manipulation Language*
IBGE	Instituto Brasileiro de Geografia e Estatística
ISO	*International Standards Organization*
PIB	Produto Interno Bruto
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	*Structured Query Language*
XML	e*Xtensible Markup Language*

Lista de Figuras

Lista de Quadros

1 Introdução

1.1 Motivação e Relevância do Trabalho

Desde o seu surgimento no final da década de 60, a tecnologia em bancos de dados vem sendo uma das áreas de maior crescimento na ciência da computação e na de informação. Com o surgimento do modelo relacional, implementado de forma comercial no início da década de 80, cada vez mais organizações se tornam dependentes do funcionamento contínuo e bem sucedido de seus Sistemas Gerenciadores de Bancos de Dados (SGBDs), dentro dos quais se encontram grande parte da informação significativa e necessária ao processo de administração e decisão da organização em questão (Date, 1984; Elmasri e Navathe, 2005).

Esses sistemas têm como objetivo registrar e manter os dados relevantes às organizações, proporcionando um ambiente eficiente na inserção e recuperação dos mesmos em bancos de dados (Silberschatz et al., 2006).

Os novos dados inseridos a cada momento nos SGBDs têm os tornado em poderosas ferramentas auxiliadoras nos processos gerenciais e decisórios, ao mesmo tempo em que se aumenta a dificuldade de transformar esses dados em informações úteis.

A informação é um dos recursos mais importantes e valiosos em uma empresa; porém, a existência da informação é freqüentemente confundida com a existência de dados.

Os dados são os fatos em sua forma primária, estando disponibilizados sem a devida organização ou arranjo significativo. Eles representam coisas do mundo real, contudo, tem poucos valores além de si mesmos (Stair, 1998).

Já as informações são conjuntos de dados organizados, de forma a adquirirem valor além de si mesmo, sendo considerado também um dado mais útil por intermédio da aplicação de conhecimento.

Stair (1998) afirma que a transformação de dados em informações é um processo, ou uma série de tarefas logicamente relacionadas, que são executadas para atingir um resultado definido.

Os computadores e os sistemas de informação estão constantemente transformando a maneira como as empresas conduzem seus negócios. A própria informação tem valor, e os sistemas de informação estão cada vez mais sendo utilizados como meio para criar, armazenar e transferir informações. A crescente competitividade em todos os setores de negócios dificulta o processo de gerenciamento e tomada de decisões empresariais, aumentando-se ainda mais a importância de um sistema de informação para auxiliar nesses processos (Stair, 1998).

Uma informação para ser valiosa necessita ser precisa e completa para que decisões ruins não sejam tomadas, causando enormes prejuízos empresariais. Também é necessário que as informações sejam pertinentes à situação, fornecida em tempo certo, e ser fornecida de forma compreensível ao usuário (Stair, 1998).

Os dados imprecisos, incertos e incompletos que representam o mundo real também estão presentes nos sistemas de informações que se propõem a modelá-los. Ao se recuperar esses dados, muitas vezes obtêm-se informações inapropriadas

e que não atenderão integralmente às expectativas do usuário (Peres e Boscarioli, 2002).

Com o crescimento da complexidade das aplicações, têm-se o surgimento de novos mecanismos de tratamento de dados, de forma a enriquecer a manipulação dos mesmos. Também, têm sido realizados estudos de novas metodologias de extração de dados com a intenção de minimizar a influência das incertezas e, ainda, de aproximar o resultado obtido nas consultas a bancos de dados às necessidades do usuário.

Uma abordagem que já é bem utilizada em sistemas de controle tem se mostrado uma possibilidade para contornar os problemas gerados pela complexidade e incerteza, que são passíveis de caracterizar os bancos de dados, isto é, a aplicação da teoria de conjuntos *fuzzy* (ou nebulosa) em ferramentas de SGBDs.

Aplicando essa metodologia a situações reais, obtêm-se soluções aproximadas e não soluções estritamente precisas para consultas a SGBDs, pois, em determinados casos, essas consultas não requerem que a resposta gerada atenda completamente às condições das mesmas, mas requerem apenas que as satisfaçam com certo grau de precisão. Mediante a teoria de conjuntos *fuzzy*, a análise dos registros não é feita de forma a considerar apenas Verdadeiro ou Falso, mas sim de forma a flexibilizar essa função, utilizando-se de infinitos graus de inclusão (Li e Liu, 1990).

A utilização da teoria de conjuntos *fuzzy* nas consultas a bancos de dados permite que se possam realizar buscas a dados numéricos que atendam a certos predicados nebulosos, tais como ‘alto’, ‘pouco’ ou ‘aceitável’, aproximando-se então a resposta gerada de um resultado humanamente inferido.

Alguns trabalhos já são encontrados nessa área, porém, poucas implementações práticas estão disponíveis. Essas condições incentivaram o desenvolvimento de um mecanismo de consulta *fuzzy* que não interfira nos dados originais armazenados no banco de dados, mas que possa auxiliar a recuperação das informações num formato mais próximo ao esperado pelo usuário.

1.2 Objetivo e Justificativa da Dissertação

Os dados que se obtém sobre o mundo real estão repletos de informações imprecisas e incompletas. Naturalmente, essa imprecisão e incompletude também estão presentes nos bancos de dados.

O conhecimento advindo de especialistas das diversas áreas, às vezes, tem um formato conceitual e qualitativo sobre o mundo real; contudo, os dados coletados possuem geralmente um formato numericamente preciso. Essa diferença faz com que, ao utilizar metodologias comuns de consulta a banco de dados, seja difícil obter resultados que retratem aquelas imprecisões e incompletudes inerentes a determinados dados.

Como exemplo, ao se realizar uma consulta convencional a um banco de dados contendo características de municípios do estado de São Paulo, buscando-se locais que ofereçam maior possibilidade de sucesso ao implantar uma empresa de turismo de negócios, obter-se-ão somente os municípios que se enquadram totalmente às condições selecionadas. Sendo assim, descartam-se municípios que poderiam se aproximar muito dessas condições ideais e que poderiam também potencialmente se enquadrar nas características desejadas.

Alguns estudos na área de manipulação e consulta de dados *fuzzy* já foram realizados, como em Rashidov (2004), que implementou uma ferramenta orientada para web com intuito de obter consultas inteligentes.

Entretanto, poucos trabalhos são totalmente implementados em linguagem SQL (*Structured Query Language*) padrão, nativa da maioria dos atuais SGBDs relacionais. Veryha (2005) realizou um trabalho nessa área, onde desenvolveu uma ferramenta de mineração de dados utilizando linguagem SQL, procedimentos armazenados e extensões de dados do Microsoft SQL Server 2000. Porém, esse trabalho não utiliza o modelo de sistemas *fuzzy* baseado em regras, sendo que o resultado é obtido por meio de um cálculo compensatório, considerando que o registro está incluído em uma classe de determinada variável e inversamente não incluído na outra classe da mesma variável.

O presente trabalho tem como objetivo a proposição de um método alternativo às pesquisas convencionais em bancos de dados, utilizando-se para tanto um sistema de inferência *fuzzy* totalmente desenvolvido em linguagem SQL padrão, permitindo ainda mostrar um contraste dessa metodologia perante as consultas tradicionais. Para propósitos de validação da abordagem proposta, esse trabalho utilizará dados do IBGE (Instituto Brasileiro de Geografia e Estatística) para ordenar as cidades com maior potencial à implantação de empresas de turismo de negócios; contudo, não tem por finalidade gerar resultados com viés comercial, mas sim demonstrar a utilidade das técnicas e ferramentas desenvolvidas, as quais podem ser facilmente adaptadas para os mais diversos tipos de aplicação.

O modelo proposto é desenvolvido na forma de uma Visão (ou *View)* armazenada no banco de dados. Nesse contexto, uma Visão é uma relação que não armazena dados, sendo então composta dinamicamente por uma consulta que é previamente analisada e otimizada. A mesma é tratada na realização de consultas como uma tabela regular.

Para realizar a inferência do sistema é necessário também que sejam armazenadas tabelas auxiliares, representando as variáveis lingüísticas em forma de funções, assim como regras advindas do conhecimento de especialistas.

1.3 Organização da Dissertação

Esse documento é composto de cinco capítulos, incluindo esta Introdução.

No Capítulo 2 são apresentadas características e modelos de bancos de dados, assim como a linguagem SQL utilizada para definir, manipular e acessar dados. Também são definidas nesse capítulo as características especificas do sistema gerenciador de bancos de dados selecionado para o desenvolvimento desse trabalho.

O Capítulo 3 mostra uma descrição da constituição dos sistemas *fuzzy*, apresentando-se também um resumo de algumas abordagens que utilizam esse sistema em associação a bancos de dados tradicionais.

No Capítulo 4 define-se a metodologia proposta para realização de consultas inteligentes em bancos de dados utilizando sistemas *fuzzy*. Ainda nesse capítulo, são exibidos os resultados obtidos, além da comparação do modelo desenvolvido em relação à consulta tradicional a bancos de dados.

Finalmente, o Capítulo 5 descreve as conclusões gerais obtidas com esse trabalho e apresenta propostas de trabalhos futuros.

1.4 Trabalhos Publicados em Eventos Científicos

No decorrer do período de mestrado, o seguinte trabalho foi publicado em anais de congresso científico:

- PENTEADO, F. B. L., SILVA, I. N. (2009). Abordagem inteligente usando filtragem fuzzy para base de dados relacionais. *Proceedings of the 8th Brazilian Conference on Dynamics, Control and Applications*, DINCON’2009, Bauru, 6 p.

2 Aspectos de Bancos de Dados Relacionais

2.1 Introdução

A origem dos bancos de dados data do final da década de 60, em que foram propostos alguns modelos de bancos de dados implementados na década seguinte. Porém, o grande marco da área dos bancos de dados é o artigo *'A Relational Model of Data for Large Shared Data Banks'* (Codd ,1970), no qual se propõe o modelo de bancos de dados relacionais.

O modelo relacional é menos ligado à estrutura do que seus precursores, pois permite um alto grau de independência dos dados, além de prover fundamentos substanciais para lidar com a semântica, a consistência e com os problemas de redundância. Outra característica é que as aplicações ligadas ao banco de dados não são afetadas por mudanças internas na organização dos dados, ordenação dos registros ou caminhos de acesso (Connoly e Begg, 2005).

Independente de sua modelagem, os bancos de dados possuem algumas características comuns que serão apresentadas na seção a seguir.

2.2 Características de Bancos de Dados

Os bancos de dados são conjuntos de registros organizados em uma estrutura regular, possibilitando-se a reorganização dos dados e a produção de informação.

Normalmente, os bancos de dados são acessados e mantidos por um SGBD, cujo principal objetivo é gerenciar o acesso, manipulação e organização dos

dados, retirando-se essa responsabilidade das aplicações finais (Elmasri e Navathe, 2005).

Um SGDB inclui alguns componentes funcionais como:

- Gerenciador de Arquivos: gerencia a alocação do espaço físico de armazenamento e as estruturas utilizadas para representar as informações armazenadas;
- Gerenciador de Bancos de Dados: Realiza a comunicação entre os dados de baixo nível com as aplicações e consultas solicitadas ao banco de dados;
- Processador de Consultas: Transforma as solicitações de alto nível feitas pelo usuário em instruções de baixo nível, para assim serem interpretadas pelo gerenciador de banco de dados, além de realizar otimizações nas consultas solicitadas;
- Pré-compilador DML: Converte a linguagem DML (*Data Manipulation Language*, abordada em detalhes na Seção 2.5), utilizada nas aplicações em linguagem do *host*, por meio da interação com o processador de consultas;
- Compilador DDL: Converte os comandos DDL (*Data Definition Language*, abordada em detalhes na Seção 2.5) em um conjunto de tabelas contendo metadados, e armazenadas nos dicionários de dados.

Esses componentes e suas relações podem ser observados na Figura 2.1.

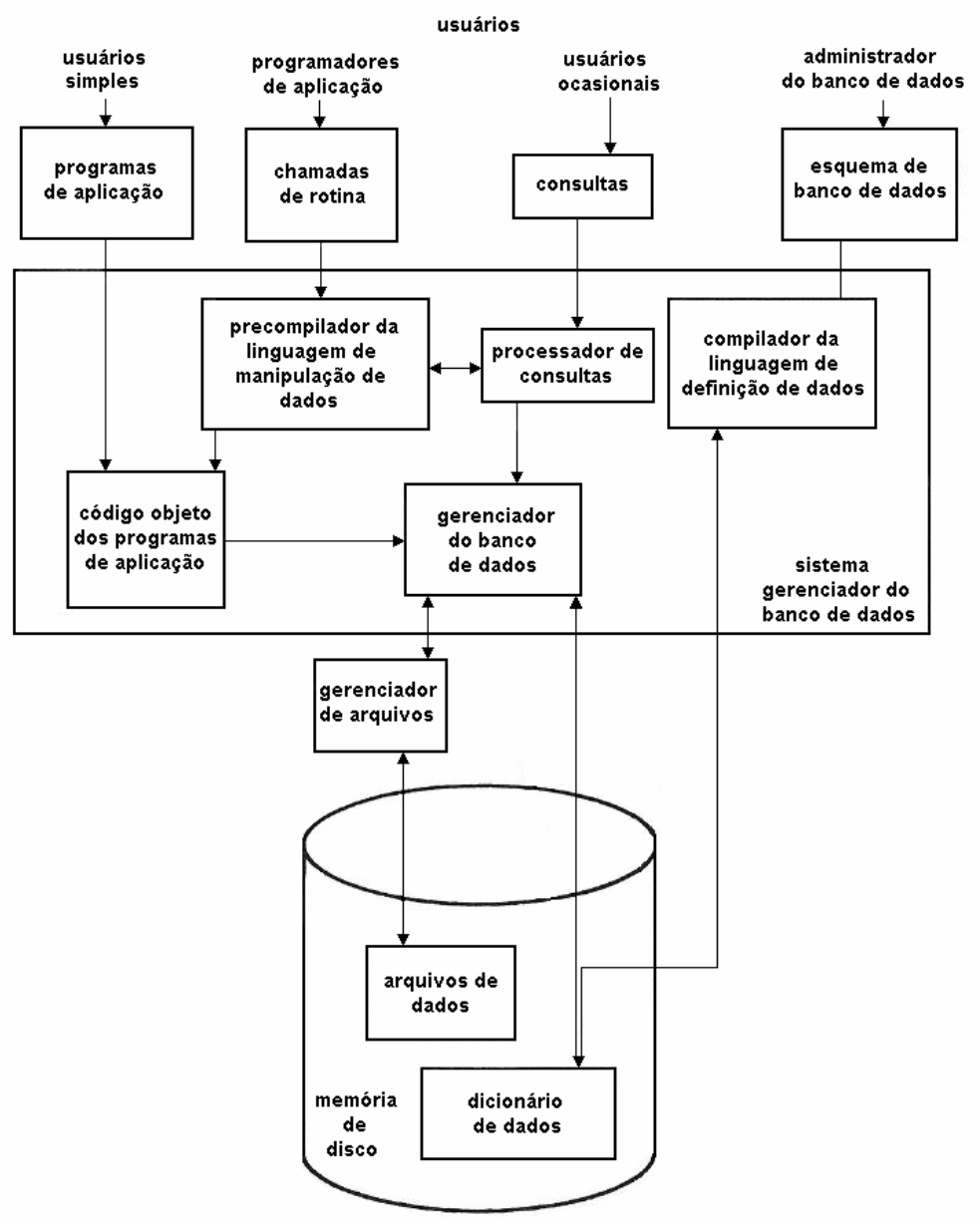


Figura 2.1 – Estrutura de um sistema gerenciador de bancos de dados.

Além dessas funcionalidades, muitas estruturas de dados são necessárias para implementação física do SGBD, dentre elas os arquivos de dados, dicionário de dados e os índices.

A interface para inclusão, alteração ou consulta aos dados é disponibilizada pelo SGBD. Considerando os bancos de dados relacionais, essa interface é

constituída pelas APIs ou drivers do SGBD, que utiliza a linguagem SQL para execução de comandos de manutenção da estrutura e dos dados.

No início da década de 60, iniciou-se o uso do termo ‘modelos de dados’ como sendo uma coleção de definições conceituais para semântica, descrição, relacionamento e restrições dos dados. No decorrer dos desenvolvimentos, muitos modelos de dados foram criados, sendo que apenas alguns foram realmente implementados.

2.3 Modelos de Dados

Segundo Elmasri e Navathe (2005), os modelos de dados provêm ao desenvolvedor e aos usuários finais uma forma única e precisa de se transcrever o conhecimento em bases de dados.

Um modelo de dados pode ser classificado em três categorias: Modelos lógicos baseados em objeto, Modelos lógicos baseados em registros e Modelos físicos.

Os modelos lógicos baseados em objeto são usados na descrição de dados nos níveis conceitual e visual. Eles fornecem capacidades de estruturação flexíveis e admitem restrições de dados para serem explicitamente especificados. Tem como principais representantes o modelo entidade-relacionamento, o funcional, o infológico e o orientado a objeto (Elmasri e Navathe, 2005).

Os modelos lógicos baseados em registros consistem de uma série de diferentes tipos de registros de formato fixo, tendo também um número invariável de campos, os quais normalmente possuem ainda um tamanho também fixo. Tem como principais exemplos o modelo em redes, o modelo hierárquico e o modelo relacional.

Os modelos físicos de dados são utilizados para descrever os dados em nível mais baixo, ou seja, como serão armazenados, como será a estrutura dos registros, ordenação dos dados e caminhos de acesso. Eles determinam aspectos de implementação do banco de dados. Esse modelo tem como representantes mais significantes o modelo unificador (*unifying model*) e o estrutura de memória (*frame memory*) (Elmasri e Navathe, 2005).

Muitos modelos de dados foram propostos na literatura; porém, os mais aceitos foram: modelo em redes, hierárquico, relacional, entidade-relacionamento e orientado a objetos.

2.3.1 Modelo em Redes

O Modelo em Redes representa os dados por intermédio de coleções de registros, e os relacionamentos entre os dados por meio de ligações. Cada registro é uma coleção de campos contendo somente uma informação, sendo que uma ligação é uma associação entre exatamente dois registros (Silberschatz et al., 2006).

Nesse modelo, todos os elos podem ser explícitos, pois não há restrição a um só tipo de relacionamento, e cada tipo de relacionamento corresponde a elos com nomes distintos. Para representar um projeto de um banco de dados no modelo de redes, utiliza-se um diagrama de estrutura de dados conforme ilustrado na Figura 2.2, sendo que as caixas correspondem aos registros e as linhas às ligações (Furtado e Santos, 1979).

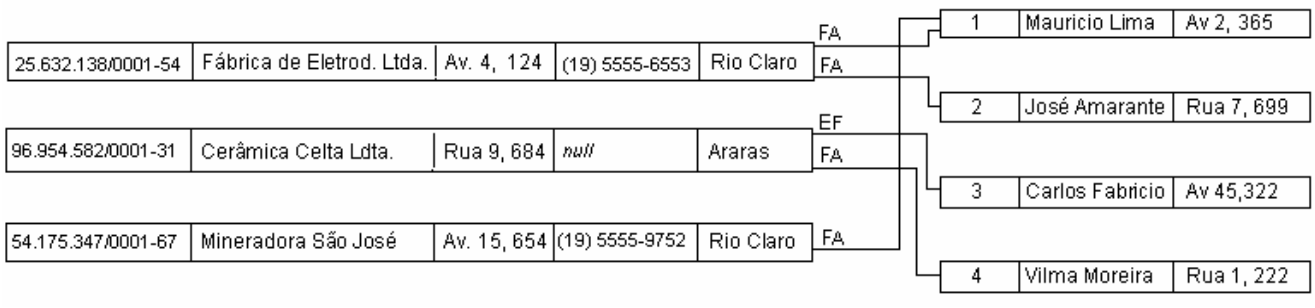


Figura 2.2 - Exemplo de banco de dados em redes.

A Figura 2.2 mostra dois tipos de rótulos: *FA* representando 'funcionário atual' e *EF* representando 'ex-funcionário'. O acesso por relacionamentos é conhecido como navegação, e é importante indicar por qual tipo de elo deseja-se navegar a cada passo.

A principal implementação desse modelo de dados foi o IDMS (*Integrated Database Management System*) que posteriormente foi estendido para IDMS/R (*Integrated Database Management System/Relational*), incorporando alguns conceitos do modelo relacional.

2.3.2 Modelo Hierárquico

O modelo hierárquico é definido como um conjunto de árvores; sendo assim, se dois registros são unidos por um elo, um dos registros é considerado ascendente e outro descendente. Os elos têm obrigatoriamente relacionamento 1:*N*, ou seja, um registro ascendente pode ter vários descendentes e um descendente pode ter somente um ascendente. São consideradas raízes os registros que não tem ascendente, podendo haver descendentes inexistentes, mas nunca ascendentes inexistentes (exceto os registros raízes), conforme ilustra a Figura 2.3.

Pode acontecer de o conteúdo de um registro em particular ser replicado em vários locais, no caso do mesmo ter ligações com mais de um registro. Essa réplica

possui duas grandes desvantagens, ou seja, pode causar inconsistência de dados quando acontecem atualizações, assim como o desperdício de espaço (Silberschatz et al., 2006).

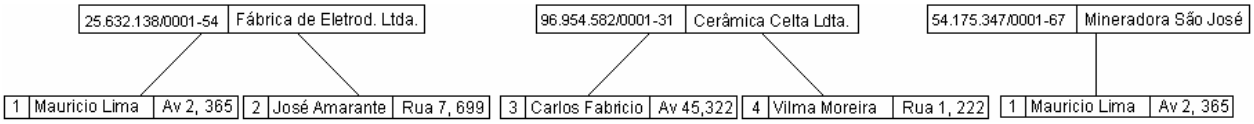


Figura 2.3 – Exemplo de banco de dados hierárquico.

O primeiro banco de dados comercialmente disponível chamava-se IMS (*Information Management System*), da empresa IBM. Ele utilizava-se do modelo de dados hierárquico, e foi lançado em 1968 (Kifer et al., 2006).

2.3.3 Modelo Relacional

O modelo relacional representa o banco de dados como uma coleção de relações que, informalmente, se parece com uma tabela de valores, similar às apresentadas no modelo entidade-relacionamento. Observando as relações como tabelas, cada linha representa uma entidade ou relacionamento do mundo real, sendo que o nome da tabela e suas colunas auxiliam na interpretação dos valores armazenados.

Utilizando a nomenclatura formal do modelo relacional, as tabelas são chamadas relações, as linhas são denominadas tuplas e o cabeçalho das colunas é nomeado de atributo. Por ser o modelo a ser empregado neste trabalho, detalhar-se-ão melhores essas e outras definições do modelo relacional na Seção 2.4.

2.3.4 Modelo Entidade-Relacionamento

O modelo entidade-relacionamento é constituído basicamente de um grupo básico de objetos, dentre eles as entidades e os relacionamentos.

Uma entidade identifica um indivíduo no universo e é constituída de um conjunto de atributos. Para cada atributo existe um conjunto de valores permissíveis denominado de Domínio. Alguns exemplos de entidade seriam os conjuntos de pares Atributos/Valores: {(Município, 'Rio Claro'), (População, 168.218), (PIB, 1.602.686)} e {(CNPJ, '01.235.654/0001-97'), (Razão, 'Fábrica de Eletrodomésticos Ltda.'), (Endereço, 'Av. 15, 324')}, representando respectivamente a cidade de Rio Claro e algumas de suas características, e uma Empresa e seu dados particulares.

As entidades podem ser agrupadas em um conjunto de entidades de um mesmo tipo; como exemplo, tem-se o conjunto nomeado MUNICIPIOS_SP, que contém dados sobre os municípios do estado de São Paulo.

Um relacionamento é definido como uma associação entre várias entidades. Pode-se, por exemplo, definir uma relação que associe a entidade 'Fábrica de Eletrodomésticos Ltda.' com a entidade 'Rio Claro', significando que a empresa citada está situada na cidade de Rio Claro.

Para cada relacionamento é definida uma cardinalidade, sendo as principais: 1:1 (um para um), 1:*N* (um para muitos) e *N*:*M* (muitos para muitos).

Este modelo entidade-relacionamento foi descrito em detalhes no artigo *'The Entity-Relationship Model-Toward a Unified View of Data'*, de Peter Chen (1976), o qual foi o seu idealizador.

2.3.5 Modelo Orientado a Objetos e Objeto-Relacional

Considerando que os sistemas de bancos de dados têm sido aplicados a uma faixa bem ampla de aplicações, como projeto assistido por computador e sistemas de informações geográficas, os modelos existentes apresentam algumas

limitações. Uma solução encontrada era introduzir bancos de dados baseados em objeto, que permitem lidar com tipos complexos de dados (Silberschatz et al., 2006).

Os tipos de dados simples têm domínios atômicos, ou seja, são consideradas unidades indivisíveis. Já os tipos de dados complexos surgiram com a necessidade de armazenar atributos compostos, multivalorados, generalização e especialização, que tem uma tradução complexa no modelo relacional (Kifer et al., 2006).

Como exemplo de dados complexos considera-se o campo Endereço, onde, sendo armazenado em uma cadeia simples de string, oculta detalhes como o logradouro ou o número do imóvel. Porém, ao solucionar esse problema por armazenar cada parte do endereço em um campo separado, aumenta-se a complexidade das consultas, já que é necessário mencionar cada um dos campos. Uma melhor alternativa seria permitir tipos de dados estruturados, em que se armazenaria o endereço com suas sub-partes em um único campo (Silberschatz et al., 2006).

O padrão SQL (ISO/IEC 9075, 1999) estende a definição de dados SQL e a linguagem de consulta para lidar com esses novos tipos de dados.

Em meados dos anos 90, foi criado o conceito de banco de dados objeto-relacional que busca combinar o melhor das abordagens relacional e orientada a objetos, possibilitando que dados relacionais sejam vistos como objetos e objetos sejam vistos como dados relacionais.

2.4 Organização de Bancos de Dados Relacionais

O modelo relacional é baseado no conceito matemático de relação, que é fisicamente representado por uma tabela. As estruturas que compõem esse modelo serão detalhadas a seguir.

2.4.1 Domínios e Atributos

Um domínio é um conjunto de valores atômicos, possuindo uma nomenclatura que deve definir seu conteúdo (Elmasri et al., 2005).

Comumente, um domínio é especificado por intermédio da definição de um tipo de dado, do qual pertencem valores que o comporão; por exemplo, NR_CNPJ armazenaria um conjunto de 14 dígitos válidos no Cadastro Nacional de Pessoa Jurídica e Nome_Cidade_Empresa conteria o conjunto de cidades onde a empresa poderia estar situada.

É necessário também especificar um tipo de dado e um formato para cada domínio. Como exemplo, o tipo de dado para o domínio NR_CNPJ são números inteiros, com o formato *nn.nnn.nnn/nnnn-nn*, onde cada *n* é também um número inteiro, sendo que tal cadeia de caracteres representa ainda um código válido no Cadastro Nacional de Pessoa Jurídica.

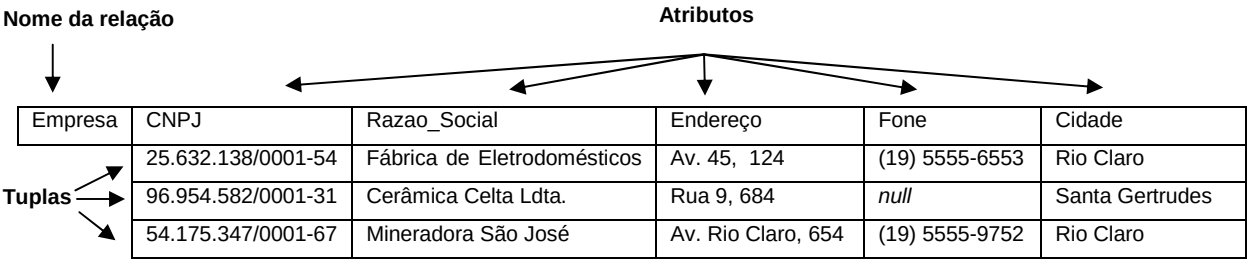
Um atributo representa o uso de um domínio dentro de uma relação. Ao considerarmos as relações como tabelas, os atributos são cabeçalhos de colunas que indicam a interpretação dos valores armazenados na mesma.

2.4.2 Relações

Na matemática, uma relação é um subconjunto de um produto cartesiano de uma lista de domínios. Considera-se uma relação como uma tabela de valores, em

que cada linha é chamada tupla e representa uma entidade no mundo real, e as colunas são os atributos da entidade (Elmasri et al., 2005).

Cada valor dentro de uma tupla é atômico, ou seja, não é divisível na estrutura do modelo relacional e, por isso, atributos compostos e multivalorados não são permitidos. Também é possivel armazenar valores especiais, chamados *nulls* ou nulos, que se referem a atributos desconhecidos ou não aplicáveis. Na Figura 2.4, tem-se um valor *null* para o atributo Fone, significando que a empresa referida não tem telefone, ou então, que o mesmo é desconhecido.



Empresa	CNPJ	Razao_Social	Endereço	Fone	Cidade
	25.632.138/0001-54	Fábrica de Eletrodomésticos	Av. 45, 124	(19) 5555-6553	Rio Claro
	96.954.582/0001-31	Cerâmica Celta Ldta.	Rua 9, 684	*null*	Santa Gertrudes
	54.175.347/0001-67	Mineradora São José	Av. Rio Claro, 654	(19) 5555-9752	Rio Claro

Figura 2.4 – Esquema de Relação 'Empresa'.

Um esquema de relação é composto do nome da relação e uma lista de atributos que auxiliam na interpretação dos dados armazenados, sendo que o grau de uma relação é o número de atributos que a compõem. Para o exemplo mostrado na Figura 2.4, tem-se uma relação de grau 5.

De acordo com Elmasri e Navathe (2005), a comparação das relações com tabelas é muito útil para a compreensão do conceito, mas existem algumas características que os tornam diferentes. Um subconjunto matemático não tem qualquer ordem entre eles, entretando, quando os registros são fisicamente armazenados em disco, sempre existe uma ordem lógica. A ordenação das tuplas não é parte da definição da relação, porém, quando uma relação é exibida como uma tabela, uma certa ordenação pode ser definida nas tuplas.

As relações podem armazenar dados sobre entidades ou relacionamentos. A relação Funcionario (Codigo, Nome, Endereco) estabelece que existem entidades relacionadas aos dados armazenados. Uma relação Contratados (Cod_Func, Cod_Emp) define que existem funcionários contratados por uma empresa, considerando que os dados dos mesmo estão armazenados na relação Funcionario e os dados da empresa estão inseridos na relação Empresa (Figura 2.5). Sendo assim, a relação Contratados representa o relacionamento entre as relações Empresa e Funcionario. No modelo relacional os dados sobre relacionamentos e entidades são representados indistintamente como relações, tornando-se, em certos casos, o entendimento do modelo mais difícil.

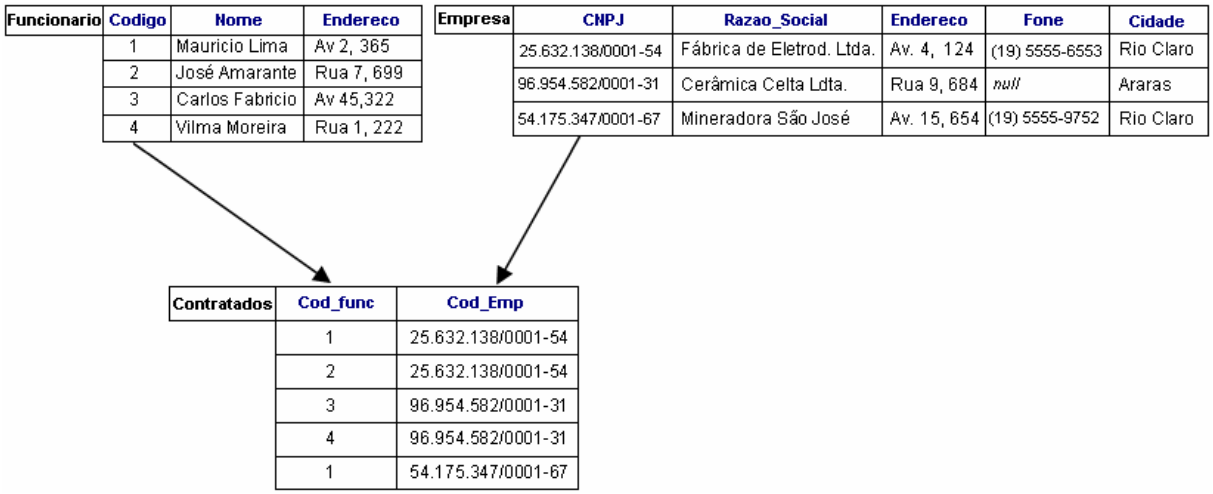


Figura 2.5 – Banco de dados Funcionários/Empresas – Visão relacional.

2.4.3 Chaves

É necessário que em todas as relações exista uma forma de identificar suas tuplas por meio de um valor único. Essa identificação única de cada tupla nos bancos de dados relacionais é chamada de chave primária.

Freqüentemente, a chave primária é formada de apenas um atributo, como no exemplo da Figura 2.4, em que a chave primária é o atributo CNPJ. Porém, nem todas as relações possuem como chave primária um atributo único, e sim uma composição de atributos, que, unidos, possuem a propriedade de identificação única (Silberschatz et al., 2006).

É garantida a existência de uma combinação de atributos que seja única no que tange àquela relação, pelo fato da relação ser um conjunto e o mesmo não possuir elementos duplicados. A chave primária poderá existir, nem que seja necessária a combinação de todos os atributos da relação, mas normalmente é utilizada uma combinação menor. É importante também que na chave primária não constem atributos redundantes, cuja participação não acrescente na unicidade da tupla. A Figura 2.5 contém um exemplo de chave primária composta, em que, na relação Contratados, a mesma é composta dos campos Cod_Func e Cod_Emp.

Considerando que cada tupla na relação representa uma entidade no mundo real, a chave primária serve como identificador único para essas entidades. Como todas as entidades devem ser distinguíveis, uma regra básica de integridade deve ser seguida, isto é, nenhum componente do valor de uma chave primária pode ser nulo. Isso porque se existir uma ou mais tuplas com valores de chave primária nulos significa que as entidades não são distinguíveis ente si, e se não são distinguíveis, segundo a definição dos conjuntos, não são várias entidades, mas somente uma.

Ocasionalmente, é possível encontrar relações em que exista mais de um atributo que tem a característica de identificador único da tupla. Esses atributos são chamados de chaves candidatas. Após a escolha arbitrária de um dos atributos, os outros recebem o nome de chave alternativa.

Nas relações em que existem relacionamentos, além das chaves primárias devem ser também definidas as chaves estrangeiras. As chaves estrangeiras são atributos que armazenam valores da chave primária de alguma tupla em alguma relação. Considerando na Figura 2.5 a relação Contratados, o campo Cod_Func, além de compor a chave primária dessa relação, é também uma chave estrangeira, significando que os valores armazenados nesse atributo representam entidades armazenadas na relação Funcionario.

Os atributos que são chaves estrangeiras também devem seguir algumas regras de integridade, chamadas de integridade referencial, em que o valor constante nesse campo pode ser somente nulo no caso de uma chave estrangeira não primária, ou ainda, no caso de estar contido em um conjunto de valores constantes na chave primária de alguma relação.

Nota-se então que as chaves, primária e estrangeira, fornecem meios para representar relacionamentos entre tuplas.

2.5 Linguagem Utilizada Para Banco de Dados Relacionais

Todos os SGBDs necessitam de uma linguagem de definição de dados (*Data Definition Language* – DDL) para definir um esquema de banco de dados relacional e de uma linguagem de manipulação de dados (*Data Manipulation Language* – DML) para acessar ou manipular os dados. Para tais objetivos, a maioria dos atuais SGBDs utiliza o SQL.

A origem do padrão SQL data da década de 70, quando um laboratório da empresa IBM lançou a linguagem chamada *Structured English QUery Language* (SEQUEL) e, também, um protótipo experimental de banco de dados relacional utilizando essa mesma linguagem, denominada de Sistema R (Astrahan et al.,

1976), ao mesmo tempo em que pesquisadores da universidade de Berkeley trabalhava no sistema INGRES, que utilizava a linguagem QUEL. A união desses conceitos de linguagem lançou os alicerces para a estruturação da linguagem SQL (Stonebraker et al., 1976).

Após a expansão da linguagem por meio de outros desenvolvedores, surgiu-se a necessidade de ser criado um padrão. Com esforços conjuntos da ANSI e da ISO foi criada a versão padrão da SQL, chamada SQL:86 ou SQL1 (Elmasri e Navathe, 2005).

A SQL teve revisões em 1992 (SQL:92) e em 1999 (SQL:99), em que foram introduzidos gatilhos, algumas características de orientação a objeto, dentre outras mudanças. A próxima revisão foi a SQL:2003, em que uma das principais mudanças referem-se às características relacionadas a XML, e a última revisão é a SQL:2006.

Apesar da padronização, na prática, existem muitas diferenças na linguagem utilizada pelos bancos de dados comerciais; porém, ao utilizar somente as funcionalidades abrangidas pelo padrão, a conversão entre os bancos de dados torna-se bem acessível.

A linguagem SQL padronizada define as estruturas de dados e operações básicas da mesma, especificando-se a semântica para criar e modificar as estruturas de dados, regras de integridade, procedimentos e consultas.

Parte da terminologia do padrão SQL difere do modelo relacional, não utilizando os termos formais para relação, atributo e tupla, mas sim os chamando respectivamente de tabela, coluna e linha (Connoly e Begg, 2005).

Alguns domínios básicos são definidos no padrão SQL, incluindo os seguintes:

- char(*n*): uma seqüência de caracteres de tamanho fixo *n*;

- varchar(*n*): seqüência de caracteres de tamanho variável, sendo o tamanho máximo *n*;
- int e smallint: são números inteiros, longo e pequeno, respectivamente;
- numeric(*n*,*d*): números de ponto fixo com precisão determinada, sendo *n* o número de dígitos acrescido do sinal e *d* é a quantidade dos *n* números que estão à direita da vírgula decimal;
- real: números decimais, com ponto flutuante.

Para o presente trabalho, outra funcionalidade relevante do padrão SQL é a estrutura denominada de v*iew* (ou visão). Essas estruturas são relações que não fazem parte do modelo lógico, mas se tornam visíveis aos usuários como relações virtuais. As *views* são definidas da seguinte forma:

CREATE view <nome da view> **as** <expressão de consulta>

onde <nome da view> é o nome atribuído a essa relação virtual e <expressão de consulta> é qualquer expressão de consulta válida.

Uma vez definida, a mesma pode ser chamada pelo seu nome em qualquer lugar permitido a uma relação regular, desde que não seja executado nenhum comando de atualização.

O padrão SQL é dividido em várias partes, entretanto, as mais relevantes para o desenvolvimento desse trabalho são a Linguagem de Definição de Dados e a Linguagem de Manipulação de Dados.

2.5.1 Linguagem de Definição de Dados (DDL)

O conjunto de tabelas em um banco de dados precisa ser expresso para o sistema por meio de uma linguagem de definição de dados.

Além de definir um conjunto de tabelas, a DDL também pode especificar outras informações, tais como um esquema para cada tabela, domínios associados aos atributos, restrições de integridade, índices, autorizações, informações de segurança e estrutura de armazenamento físico de cada relação no disco (Silberschatz et al., 2006).

O resultado da compilação de comandos DDL é um conjunto de tabelas armazenado em um dicionário de dados. O dicionário de dados integra os metadados, que são descrições dos objetos dentro do banco de dados, e facilita a manipulação e acesso a esses objetos.

A DDL é ainda usada para definir um esquema de dados ou modificar o esquema existente, não sendo possível manipular os dados por meio dela (Connoly et al., 2005).

São poucos os comandos que fazem parte da DDL, sendo os mesmos o comando CREATE (cria um objeto), o comando DROP (apaga um objeto) e o comando ALTER (altera um objeto).

Como exemplo, o código DDL de criação para o modelo exemplificado na Figura 2.5 é expresso como se segue:

```
CREATE table Empresa
(       CNPJ char(18),
        Razao_Social char(65),
        Endereco char(50),
        Fone char(35),
        Cidade char(35),
        primary key (CNPJ));

CREATE table Funcionario
(       Codigo int,
        Nome char(65),
        Endereco char(50),
        primary key (Codigo));

CREATE table Contratados
(       Cod_func int,
        Cod_emp char(18),
        primary key (Cod_func, Cod_emp),
        foreign key (Cod_func) references Funcionario (Codigo),
        foreign key (Cod_emp) references empresa (CNPJ));
```

2.5.2 Linguagem de Manipulação de Dados (DML)

O subconjunto da linguagem SQL que permite ao usuário acessar ou manipular dados armazenados no banco de dados é chamado DML. Por meio dela é possível recuperar informações armazenadas, inserir novas informações e excluir ou modificar informações existentes.

A parte da DML que envolve a recuperação de dados é chamada de linguagem de consulta, considerada um conjunto especial de linguagem de alto nível com o propósito específico de satisfazer diversos pedidos de recuperação de dados mantidos na base.

A DML pode ser basicamente dividida em dois tipos: DML procedural e DML declarativa. Os modelos de bancos de dados hierárquicos e em redes utilizam a DML procedural, que requer que o usuário especifique quais dados são necessários e como obtê-los. Para utilizá-la é preciso especificar todas as operações de acesso aos dados, chamando procedimentos específicos para se obter a informação desejada.

A linguagem SQL utiliza a DML declarativa (ou não-procedural), que requer somente que o usuário especifique quais dados são necessários, sem determinar como os mesmos serão obtidos. Essa linguagem facilita a sua utilização pelo usuário, porém, cabe ao banco de dados a tarefa de descobrir uma maneira eficiente de acessar os dados (Silberschatz et al., 2006).

Para manipular os dados, a SQL tem três comandos básicos: INSERT (para inserir um novo dado na tabela), UPDATE (para alterar) e DELETE (para excluir um dado existente).

Na linguagem de consulta utilizada pelo padrão SQL, toma-se como entrada uma ou várias tabelas, sendo obtida como resposta sempre uma única tabela. O comando fundamental para a recuperação de informações é o SELECT, cujo formato básico tem a seguinte estrutura:

SELECT <lista de atributos>
FROM <lista de tabelas>
WHERE <condições>
GROUP BY <lista de atributos de agrupamento> **HAVING** <condições de agrupamento>
ORDER BY <lista de atributos de ordenação>;

onde:

- <lista de atributos> é uma lista com os nomes dos atributos que se deseja recuperar, tendo-se como opção utilizar o símbolo de asterisco ‘∗’ para representar ‘todos os atributos’;
- <lista de tabelas> são as relações necessárias para o processamento da consulta;
- <condições> são as expressões condicionais que restringem as tuplas a serem recuperadas;

- <lista de atributos de agrupamento> são atributos que, contendo um mesmo valor, serão agrupados;
- <condições de agrupamento> são as condições as quais os grupos formados devem pertencer;
- <lista de atributos de ordenação> especifica os atributos que serão utilizados na ordenação das tuplas.

A ordem das cláusulas não pode ser modificada, contudo, excetuando-se SELECT e FROM, todas as outras cláusulas podem ser omitidas.

Para eliminar registros duplicados do resultado da consulta é necessário inserir a palavra-chave DISTINCT após o SELECT. A linguagem SQL define como padrão não remover registros duplicados, mas permite ao usuário explicitar essa condição usando-se a palavra-chave ALL após o SELECT.

A cláusula SELECT também pode conter operações com atributos e constantes utilizando operadores ‘+’, ‘–’, ‘∗’ e ‘/’, além das funções de agregação AVG (média), MIN (mínimo), MAX (máximo), SUM (somatório) e COUNT (contador).

Para definir as condições na cláusula WHERE utilizam-se expressões envolvendo os operadores de comparação ‘<’ (menor), ‘<=’ (menor ou igual), ‘>’ (maior), ‘>=’ (maior ou igual), ‘=’ (igual) e ‘<>’ (diferente), ao passo que para unir tais expressões usam-se os conectivos AND, OR e NOT.

É possível dentro de um SELECT renomear relações e atributos a fim de evitar ambigüidades, criar consultas recursivas e facilitar a compreensão e o desenvolvimento de consultas mais complexas. Para isso, utiliza-se a palavra-chave AS ligando o atributo ou relação ao seu pseudônimo, podendo também aparecer tanto na cláusula SELECT quanto na cláusula FROM.

A SQL fornece alguns mecanismos de junção de relações. Elas podem ser feitas por meio dos seguintes mecanismos: produto cartesiano, junções internas e junções externas (Silberschatz et al., 2006).

O produto cartesiano define os atributos a serem retornados após a cláusula SELECT, especificando-se a qual relação esses atributos pertencem. As relações envolvidas devem ser especificadas na cláusula FROM e a condição de ligação entre elas na cláusula WHERE.

As junções internas são caracterizadas por retornar apenas os dados que satisfazem as condições de junção. Considerando essa característica, é preciso atenção quanto aos valores nulos nos campos chaves, pois não é possível relacioná-los com valores da tabela unida, excluindo esses registros do resultado da consulta. As junções internas são definidas após a cláusula FROM e utilizam o seguinte bloco de comandos:

INNER JOIN <relação a ser unida>

ON <condição de união>;

Quanto às junções externas, não é necessário que os registros tenham correspondentes na tabela unida. Essas junções são subdivididas em três comandos: LEFT OUTER JOIN para trazer todos os registros da tabela à esquerda, mesmo que não tenham correspondências; RIGHT OUTER JOIN para trazer todos os registros da tabela à direita, mesmo que não tenham correspondências; e FULL OUTER JOIN para trazer todos os registros de ambas as tabelas, mesmo que não tenham correspondências. Essas junções são escritas da mesma forma que as junções interiores, somente substituindo-se a palavra-chave INNER pelo comando OUTER desejado.

A SQL fornece também um mecanismo para aninhar subconsultas, tendo como objetivo comum realizar testes de participação, comparação, cardinalidade entre conjuntos e cálculos complexos envolvendo várias tabelas. Como exemplo, uma consulta que retorne todos os campos da Tabela1, cujo Campo1 tenha valor maior que pelo menos uma referência no Campo2 da Tabela2 (Silberschatz et al., 2006):

```
SELECT * FROM Tabela1
WHERE Campo1 > SOME (SELECT Campo2  FROM Tabela2 );
```

2.6 Aspectos Principais do Microsoft SQL Server 2005

O sistema gerenciador de banco de dados Microsoft SQL Server 2005 foi selecionado para o desenvolvimento desse trabalho por realizar processamentos complexos, com elevado fluxo de dados, de forma eficiente e suportar grande parte dos comandos definidos no padrão SQL.

Esse trabalho considerou para realização do experimento somente os municípios do estado de São Paulo, que somam 645 municípios. A necessidade de suporte a um grande processamento de dados não se deve ao número de municípios que serão processados, e sim, aos dados referentes à base de conhecimento do sistema *fuzzy*, que totalizam cerca de 695 mil registros. Como exemplo, um município que ativa quatro regras do modelo *fuzzy*, realizará processamentos em consultas aninhadas, ordenações, agrupamentos e estruturas condicionais com cerca de 10 mil registros. Para gerar um ranking total dos municípios, o SGBD terá de processar mais de 4 milhões de registros, o que torna o processamento complexo, mesmo não tendo um número tão grande de municípios.

Algumas funcionalidades do padrão SQL que são fundamentais para o desenvolvimento desse trabalho, que são suportadas por essa ferramenta, são as

consultas aninhadas, blocos condicionais do tipo CASE, agrupamentos e funções de agregação.

O SQL Server é um SGBD baseado no modelo relacional e na linguagem SQL, utilizando uma extensão da linguagem chamada Transact-SQL (T-SQL). Dentre as funcionalidades incluídas nesta extensão estão o uso de variáveis globais, funções extras de processamento de string e de data e algumas funções matemáticas, porém, nenhuma dessas extensões é utilizada no desenvolvimento do modelo.

Os tipos de função suportados pela linguagem Transact-SQL são: agregativas, matemáticas, estatísticas, com data e hora, de segurança, com string, de sistema, dentre outras. Como exemplo tem-se:

- Funções de agregação: AVG (média), COUNT (contador), MAX (máximo), MIN (mínimo), SUM (somatório), STDEV (desvio padrão) e VAR (variância);
- Funções matemáticas: ABS (valor absoluto), POWER (Potenciação), RAND (número aleatório entre 0 e 1), ROUND (arredondamento), SQRT (raiz quadrada) e SQUARE (quadrado);
- Funções com string: LOWER (transforma em minúsculas), UPPER (transforma em maiúsculas), SUBSTRING (retorna parte de uma string) e LEN (retorna o número de caracteres da string);
- Funções de conversão: CAST e CONVERT (transformam um tipo de dados em outro).

Embora o SQL Server seja projetado para ser um servidor de banco de dados para muitos usuários, o mesmo também pode ser utilizado num único computador que tenha tanto o aplicativo como o banco de dados em si.

O SQL Server suporta também a linguagem XML, que é considerada um dos padrões para tratamento de dados na Internet. Normalmente, retorna seus resultados em forma de tabelas relacionais, mas o mesmo suporta uma cláusula que transforma seu resultado num documento XML. Também é possível armazenar documentos XML na base de dados e expor os dados por meio de tabelas relacionais.

Os componentes lógicos como tabelas, visões e procedimentos no SQL Server são visíveis aos usuários e a implementação física dos arquivos é bem transparente. É possível ao administrador do banco de dados programar a execução automática de tarefas repetitivas, programar o servidor para enviar e-mails em determinadas condições e utilizar as ferramentas gráficas disponíveis para executar as tarefas administrativas de forma mais fácil e eficiente.

O Microsoft SQL Server 2005 possui ainda um conjunto de ferramentas que também auxiliam no desenvolvimento e manutenção da base de dados, sendo:

- SQL Server Management Studio: Ferramenta para gestão da base de dados relacional utilizando linguagem Transact-SQL, MDX e XML;
- SQL Server Surface Area Configuration e Configuration Manager: Ferramenta para configurar inicialização automática, opções de conectividade e opções avançadas;
- Business Intelligence Development Studio: Utilizado para geração de relatórios e serviços de integração;
- SQL Server Profiler: Ferramenta para capturar e monitorar a atividade do servidor de banco de dados;
- Database Engine Tuning Advisor: Ferramenta para melhorar o desempenho do banco de dados;

- Command Prompt Utilities: Ferramentas e prompt de comandos para uso com o SQL Server;
- Books Online: Arquivo de ajuda do SQL Server.

Já as APIs são mecanismos usados pelas aplicações para se ter acesso a recursos no computador local ou disponível via rede. O Microsoft SQL Server 2005 suporta diversos tipos de APIs que as aplicações podem utilizar para acessar recursos no SQL Server.

O SQL Server 2005 também disponibiliza um algoritmo *fuzzy* para realizar uma busca aproximada de caracteres. Essa busca tenta encontrar registros similares a outros passados por parâmetro, retornando o valor mais próximo do registro selecionado e o valor de similaridade entre eles. Outra utilização do algoritmo *fuzzy* é o agrupamento em classes, considerando que os registros agrupados não são exatamente iguais, mas são muito semelhantes entre si.

Apesar da disponibilidade desses algoritmos *fuzzy* no SGBD, os mesmos somente são usados para comparação de caracteres, não podendo ser utilizados para manipular dados numéricos.

2.7 Considerações Parciais

Nesse capítulo foram destacados os principais aspectos relacionados aos bancos de dados e à linguagem SQL, sendo que os mesmos serão reportados quando da proposição da estratégia *fuzzy* (Capítulo 4). Algumas das considerações que devem ser enfatizadas são as seguintes:

- O modelo de dados mais utilizado na atualidade e mais adequado ao desenvolvimento desse trabalho é o modelo relacional. Nesse modelo, os

dados são armazenados em estruturas parecidas com tabelas de valores, em que cada linha representa uma entidade ou relacionamento do mundo real;

- A linguagem SQL, que é utilizada no SGBD selecionado, fornece meios para definir, manipular e acessar dados dentro do SGBD;
- O SGBD Microsoft SQL Server foi aquele especificado para o desenvolvimento desse trabalho em virtude de ser um sistema robusto, utilizar a linguagem SQL e aceitar várias funções matemáticas e de agregação que facilitam a integração do modelo tradicional de consultas com as técnicas dos sistemas *fuzzy*.

3 Aspectos de Sistemas de Inferência *Fuzzy*

3.1 Introdução

A maioria dos modelos computacionais utiliza conceitos da lógica clássica para a tomada de decisões, oferecendo-se respostas do tipo ‘sim’ ou ‘não‘. Porém, grande parte das situações reais não pode ser definida tão precisamente, pois é carregada de conceitos abstratos, vagos e imprecisos, que são inerentes da forma de processar informação, comunicar e raciocinar dos seres humanos (Pedrycz, 1993; Zimmermann, 1996).

A teoria de conjuntos *fuzzy* (também chamada de conjuntos nebulosos ou difusos), introduzida em Zadeh (1965), determinou uma inovadora forma de manusear informações imprecisas, traduzindo expressões verbais vagas e qualitativas, comuns da inferência humana, numa forma passível de implementação em computadores. O valor prático proporcionado por essa tecnologia advém da possibilidade de implantar predicados *fuzzy* em controles e processos, possibilitando-se então a resolução de problemas complexos.

Assim, o objetivo de sistemas *fuzzy* é aproximar a decisão computacional da decisão humana, representar sistemas não-lineares complexos por meio de regras lingüísticas, executar procedimentos de inferência de forma aproximada e obter conclusões sem a necessidade explícita de modelos matemáticos que descrevem o comportamento do processo.

3.2 Aspectos Envolvidos com a Lógica Fuzzy

Na teoria clássica, um conjunto é definido como uma coleção de elementos, podendo ser finito ou infinito. Sua propriedade fundamental é que sua função característica é bivalente, ou seja, um elemento de um universo está ou não em determinado conjunto. Assim, considerando um elemento em particular, pode-se então concluir de antemão que o mesmo irá pertencer ou não a um dado conjunto, bastando-se para tanto avaliar a função característica do conjunto.

Diferindo da teoria dos conjuntos clássicos, a teoria dos conjuntos *fuzzy* considera que os conjuntos do mundo real não possuem fronteiras tão precisas, o que leva à flexibilização de sua função de pertinência, possibilitando-se então concluir que um elemento possa pertencer parcialmente a um conjunto. Uma característica a ressaltar sobre esses sistemas é que sua função de pertinência não é agora mais bivalente, podendo assim levar a diversos valores que possibilitam quantificar o quão um elemento é compatível com um conjunto em particular.

3.2.1 Funções de Pertinência

A função de pertinência representa um aspecto fundamental dos sistemas *fuzzy*, pois a mesma especifica o grau de compatibilidade de determinado elemento do universo ao conjunto *fuzzy*.

As funções de pertinência são definidas dentro de um universo de discurso, que representa todos os possíveis valores que podem ocorrer para a variável especificada como argumento da função. Formalmente, tem-se que:

$$\mu_A : X \rightarrow [0,1] \qquad (3.1)$$

onde $\mu_A(x)$ retorna o grau de pertinência do elemento *x,* pertencente ao universo de discurso *X*, em relação ao conjunto *fuzzy A.* O valor de $\mu_A(x)$ está normalizado entre 0 e 1, sendo que 0 indica exclusão total e 1 pertinência total ao conjunto *A*.

As funções de pertinência podem ser contínuas ou discretas; entretanto, no contexto discutido aqui, os sistemas *fuzzy* serão implementados em computadores digitais que requerem valores finitos e discretos, implicando-se, para tanto, em funções também finitas e discretas.

Segundo Simões e Shaw (2007), funções triangulares e trapezoidais são os tipos de funções de pertinência mais utilizadas, devido à facilidade de sua geração; porém, em sistemas em que um desempenho suave é de importância crítica, outra alternativa possível seria o uso de funções do tipo gaussiana ou sigmoidal. Não é necessário que as funções sejam simétricas ou igualmente espaçadas, sendo que cada variável de entrada ou saída pode ter um conjunto de funções de pertinência diferentes, com formatos e distribuições próprias, definidas de acordo com as características da variável e do processo a ser mapeado. Essa especificação pode ser realizada por intermédio de especialistas ou por procedimentos automatizados.

No exemplo da Figura 3.1, tem-se definido uma função de pertinência triangular para o conjunto *fuzzy* de números reais próximos de 8, em que o eixo μ(*x*) representa os valores de pertinência no intervalo [0,1], ao passo que o eixo *x* representa o universo de discurso definido pelo intervalo [0,18].

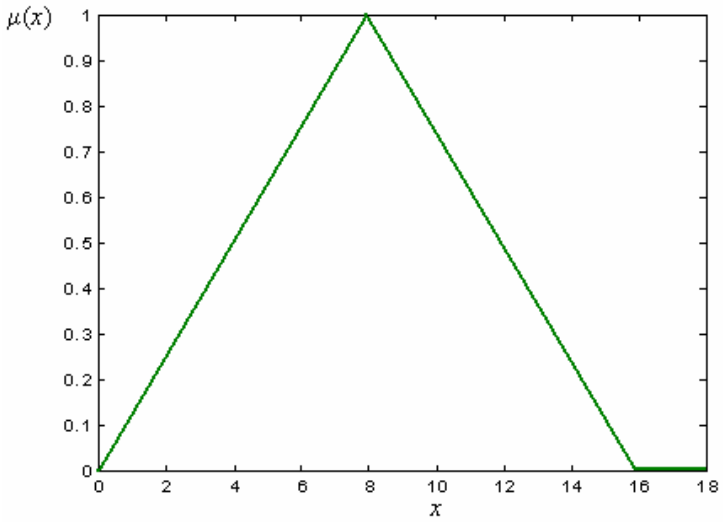


Figura 3.1 - Exemplo de conjunto fuzzy (*próximo de oito*).

3.2.2 Números *Fuzzy*

Um número *fuzzy* é um conjunto *fuzzy* que obedece a duas regras básicas, ou seja, normalidade e convexidade (Zimmermann, 1996).

Um conjunto *fuzzy* é considerado normal se pelo menos um de seus elementos possuírem grau de pertinência igual a 1. Formalmente, tem-se:

$$\sup(\mu_A(x)) = 1 \tag{3.2}$$

onde *A* representa o conjunto *fuzzy*, e o maior valor dentre todos os valores de pertinência do conjunto, ou a sua altura, deve ser igual a 1.

Um conjunto *fuzzy* não normal pode vir a ser. Para tanto, basta-se a utilização da seguinte expressão de conversão:

$$A_{NORM} = \frac{A}{Alt(A)} \tag{3.3}$$

sendo que *Alt*(*A*) representa o maior valor de pertinência encontrado no conjunto *fuzzy*, a altura de *A*.

Para ser convexo, um conjunto *fuzzy* deve obedecer a seguinte regra:

$$\mu_A(\lambda \cdot x_1 + (1-\lambda) \cdot x_2) \geq \min[\mu_A(x_1), \mu_A(x_2)], \ \textit{para} \ \forall \ x_1, x_2 \in X \qquad (3.4)$$

onde $\lambda \in$ [0,1].

A convexidade determina que não haja “buracos” no meio do conjunto e “baías” em seus limites, garantindo-se assim a unicidade na avaliação numérica do valor no eixo horizontal.

3.2.3 Operações com Conjuntos *Fuzzy*

Assim como na teoria clássica, as operações básicas de intersecção, união e complemento podem ser também executadas sobre os conjuntos *fuzzy*. Para realizar essas operações, utilizam-se basicamente duas classes de operadores: as normas triangulares (ou t-normas), como por exemplo o operador mínimo (‘$\wedge$’); e as s-normas ou co-normas triangulares (Pedrycz e Gomide, 1998), como por exemplo o operador máximo (‘$\vee$’).

Assim, assumindo-se dois conjuntos *fuzzy A* e *B*, definidos em um mesmo universo de discurso *X*, a união A$\cup$B tem como função de pertinência os valores máximos entre $\mu_A(x)$ e $\mu_B(x)$, tendo-se a seguinte notação:

$$\mu_{A \cup B}(x) = \max[\mu_A(x), \mu_B(x)], \ \forall x \in X \qquad (3.5)$$

O conjunto intersecção A$\cap$B, por sua vez, é constituído dos valores mínimos entre $\mu_A(x)$ e $\mu_B(x)$. Formalmente, tem-se:

$$\mu_{A\cap B}(x) = \min[\mu_A(x), \mu_B(x)],\ \forall x \in X \quad (3.6)$$

Finalmente, para se obter o complemento $\overline{A}$ de um conjunto *fuzzy A*, subtrai-se $\mu_A(x)$ da constante 1 (complemento de 1). Formalmente, tem-se:

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x),\ \forall x \in X \quad (3.7)$$

3.2.4 Variáveis lingüísticas

Segundo Pedrycz e Gomide (1998), as variáveis lingüísticas dentro dos sistemas *fuzzy* permitem a descrição da informação que é normalmente qualitativa. Elas são compostas por um nome, um conjunto de termos e um universo de discurso. Como exemplo, considera-se um conjunto de termos *T*(*x*), sendo *x* o PIB de um município, o qual poderia ser representado por:

T(*x*) = {*Baixo, Médio, Alto*}

onde os rótulos '*Baixo*', '*Médio*' e '*Alto*' são os termos ou valores lingüísticos da variável lingüística *x.*

3.3 Principais Componentes dos Sistemas de Inferência Fuzzy

Os sistemas de inferência *fuzzy* fornecem uma forma alternativa para modelagem de processos, que está apta no tratamento e na manipulação de informações imprecisas, as quais são expressas de forma qualitativa e representadas por uma série de conjuntos *fuzzy*.

O comportamento de um sistema de inferência *fuzzy* pode ser basicamente dividido em três etapas: *fuzzificação*, processo de inferência e *defuzzificação*. Essa estrutura representa a transformação que ocorre no domínio de aplicações do mundo real para o domínio *fuzzy*. Durante esse processo, um conjunto de

inferências *fuzzy* é utilizado para tomada de decisões, sendo que, em seguida, procede-se com uma transformação inversa do domínio *fuzzy* para o domínio do mundo real (Simões e Shaw, 2007).

3.3.1 Fuzzificação

Na fuzzificação o sistema toma as variáveis de entradas (reais), geralmente provenientes de sensores físicos ou dispositivos computadorizados, e as transformam em conjuntos *fuzzy*. Para isso, são utilizados os dados armazenados na base de conhecimento para identificar e determinar a relevância dessas entradas perante o sistema. A base de conhecimento é formada por uma base de dados e uma base de regras.

A base de regras representa o comportamento do sistema, mediante um conjunto de regras *fuzzy*. A base de dados contém as definições numéricas necessárias para definir as funções de pertinência usadas pelo conjunto de regras *fuzzy* e parâmetros do modelo.

3.3.2 Processo de Inferência

Para simular a tomada de decisão humana, o processo de inferência se utiliza de regras *fuzzy*, gerando-se resultados a partir das condições definidas na base de conhecimento.

O processo de inferência tem como objetivo final obter um conjunto *fuzzy* de saída. Para tal, o mesmo encontra todas as regras ativadas e determina suas individuais contribuições *fuzzy*, as quais serão então combinadas a fim de produzir a respectiva saída *fuzzy*.

3.3.3 Defuzzificação

Após o processo de inferência, realiza-se o procedimento de defuzzificação. Nessa etapa, o conjunto *fuzzy* de saída, inferido a partir das regras ativadas, será traduzido num único valor de saída (real), que representa as contribuições das regras ativadas. Apesar de esse ser um mecanismo para a maioria dos casos, em algumas situações particulares, a defuzzificação não é necessária desde que uma saída qualitativa seja aceitável.

Para realizar a defuzzificação, diversos métodos podem ser empregados, tais como o Centro de Área, a Média dos Máximos e o Primeiro Máximo.

O método utilizado nesse trabalho foi o Centro de Área (CDA), que calcula o centróide da área que representa o conjunto de saída *fuzzy*, o qual foi composto pela união de todas as contribuições de regras que foram ativadas. O cálculo do centro de área é realizado usando a seguinte expressão:

$$CDA = \frac{\sum(\mu_k \cdot x_k)}{\sum \mu_k} \tag{3.8}$$

onde x_k são os elementos discretizados do universo de discurso e μ_k são os seus respectivos valores de pertinência.

3.4 Processo de Inferência Associado a Sistemas de Tipo Mamdani

Para que o processo de inferência gere um conjunto *fuzzy* de saída, torna-se necessário realizar um processo de composição. Esse procedimento envolve a representação das regras e a atribuição de uma semântica a elas (Zimmermann, 1996).

O operador de implicação utilizado nesse trabalho é aquele proposto por Mamdani (King e Mamdani, 1977; Mamdani, 1977), que teve como objetivo inicial controlar um conjunto de motores a vapor por meio de regras lingüísticas obtidas a partir de especialistas. Formalmente, a semântica é definida pela seguinte expressão:

$$\mu_{R(A\rightarrow B)}(x,y) = \min[\mu_A(x), \mu_B(x)], \ \forall x \in X, \ \forall y \in Y \tag{3.9}$$

As regras devem descrever o comportamento desejado do sistema, e podem ser determinadas na forma de sentenças com o auxílio de um especialista, por meio de princípios físicos ou de variáveis inerentemente *fuzzy* (quando não há quantidades precisas reais que possam ser estabelecidas).

As regras lingüísticas que definem o comportamento do sistema possuem a seguinte forma: “SE *antecedente* ENTÃO *conseqüente*”, em que *antecedente* indica a condição e o *conseqüente* representa a saída (ação) produzida. Utilizando-se de um conjunto finito de regras desse tipo, os sistemas *fuzzy* determinam o comportamento das variáveis de saída por meio do processo de inferência.

No presente trabalho, utilizar-se-á a regra da Composição, isto é, dado que *x* é *A*’ e sabendo que se *x* é *A* então *y* é *B*, pode-se então deduzir que *y* é *B*’, onde $B' = A' \circ R_{(A\rightarrow B)}$, sendo que o símbolo “$\circ$” denota a composição max-min.

Por conseguinte, um estimador *fuzzy* pode ser então constituído por várias dessas regras, que podem ser ativadas em paralelo, com diferentes graus de ativação, sendo que cada uma delas produz um resultado de saída em particular.

Como ilustração deste processo de inferência, conforme ilustrado na Figura 3.2, considerando-se duas regras ativas R_1 e R_2, as entradas precisas (reais) representadas pelas variáveis *x* e *y* ativam duas regras *fuzzy*, cujos termos ativados são A_1 e B_1 (para a primeira regra) com A_2 e B_2 (para a segunda.) Os respectivos

resultados individuais produzidos em cada uma delas, após a aplicação do operador de implicação de Mamdani, são representados pelas regiões *fuzzy* destacadas nos seus conseqüentes C_1 e C_2.

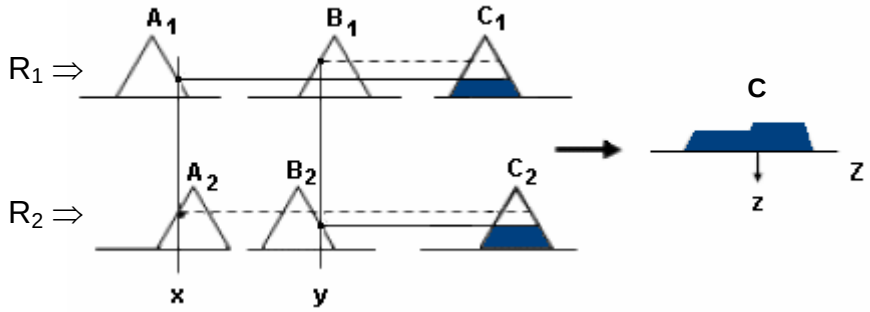


Figura 3.2 - Sistema de inferência utilizando o operador de implicação de Mamdani e o operador de agregação máximo.

Após a operação de composição, torna-se necessário utilizar um método de agregação para combinar todos os conjuntos *fuzzy* produzidos por cada regra, a fim de gerar o conjunto *fuzzy* de saída. Para tanto, emprega-se normalmente uma s-norma para realizar tal tarefa, sendo tipicamente assumido o operador máximo para efetuar este procedimento. Considerando a Figura 3.2, visualiza-se a região *fuzzy* de saída *C*, que foi produzida com a aplicação do operador máximo sobre as contribuições proporcionadas por C_1 e C_2, isto é, $C = C_1 \cup C_2$.

Ressaltam-se a seguir algumas vantagens em utilizar sistemas do tipo Mamdani:

- As regras são de fácil compreensão, sendo que o efeito de cada regra pode ser naturalmente interpretado;
- Os sistemas desse tipo executam processos de forma mais rápida, pois não envolvem cálculos muito complexos;
- São robustos, confiáveis e tolerantes às perturbações externas, visto que nesse tipo de sistema cada regra é processada de forma independente.

Assim, uma falha parcial do sistema pode não deteriorar significativamente o desempenho do estimador.

3.5 Resumo de Abordagens Fuzzy Aplicadas em Bancos de Dados Relacionais

Na literatura correlata, há diversos trabalhos relacionando sistemas *fuzzy* e sistemas de informações computacionais, tais como aqueles relatados em Chen e Gorla (1998) e em Satur e Liu (1999). Já no que tange ao contexto de bancos de dados, há também aplicações diversificadas.

No contexto de modelos orientados a objetos, destacam-se aqueles trabalhos registrados em Koyuncu e Yazici (2003), Majumdar et al. (2002) e Yazici et al. (2008). No que tange aos modelos em redes, algumas investigações foram relacionadas em Chen e Huang (2005) e em Saygin e Ulusoy (2001).

Entretanto, para o caso específico de aplicações envolvendo os modelos relacionais, observam-se poucos trabalhos diretamente correlacionados. Entre tais trabalhos, destaca-se um estudo realizado por Ma e Yan (2007), em que se propõem regras teóricas para traduzir consultas *fuzzy* em consultas SQL tradicionais, baseando-se em princípios de conjuntos *fuzzy* e em operações de α-cortes em números *fuzzy*.

Outra ferramenta, orientada para web, foi implementada por Rashidov (2004) para realizar consultas inteligentes no banco de dados da Universidade Técnica de Gabrovo (Bulgária), em que se utilizaram sistemas *fuzzy* a fim de expandir as consultas tradicionais utilizando sistemas *fuzzy*.

Em Branco (2004), apresenta-se uma metodologia para, a partir de uma base de regras *fuzzy* obtida por um algoritmo de aprendizado para aplicações de classificação, gerar de maneira semi-automática um conjunto de consultas *fuzzy*.

Essas consultas, se submetidas à base de dados, selecionam os registros pertencentes a cada classe. O autor também apresenta um método de avaliação de consultas *fuzzy* com pesos e um processo de simplificação das mesmas por intermédio de uma ferramenta comercial.

O tema também é abordado por Peres e Boscarioli (2002), os autores propõem uma aplicação de sistemas *fuzzy* para recuperação de informações textuais em bancos de dados relacionais. O sistema implementado permite observar as vantagens obtidas na recuperação de documentos textuais quando a imprecisão e a incompletude da informação são consideradas no processo de busca e de tomada de decisão.

Já Veryha (2005) propõe a utilização de sistemas *fuzzy*, da linguagem SQL tradicional e de extensões do Microsoft SQL Server 2000 como ferramenta de mineração de dados. O modelo proposto permite funções de pertinência regulares com apenas duas classes. Isso se deve ao fato do valor de saída ser calculado em função de um cálculo compensatório, onde os graus negativos (de não concordância) são tratados e exercem a mesma influência como se fossem positivos, objetivando-se para tanto chegar a um valor compensatório da contribuição de cada atributo no processo de agregação.

3.6 Considerações Parciais

Estruturas e características dos sistemas *fuzzy* foram abordados nesse capítulo. Os principais pontos a serem destacados são os seguintes:

- A teoria de conjuntos *fuzzy* surgiu com a necessidade de se desenvolver um método capaz de expressar, de maneira sistemática, quantidades que são geralmente disponibilizadas de forma imprecisa, vaga ou mal-definida;

- Expressões verbais, inerentes à linguagem de comunicação humana, e que possuem vários graus intrínsecos de incerteza, podem ser manipuladas por meio da teoria de conjuntos *fuzzy*;
- Os conjuntos *fuzzy* proporcionam uma técnica que realiza o gerenciamento de incertezas a partir da atribuição de graus de pertinência, no intervalo [0,1], a cada um dos elementos do universo de discurso do respectivo conjunto, em que a pertinência total é representada pelo valor 1 e a não pertinência pelo valor 0;
- Alguns estudos têm sido realizados com o intuito de incorporar a teoria de conjuntos *fuzzy* às consultas em bancos de dados relacionais, as quais consideram as possíveis vantagens de junção da linguagem SQL tradicional à flexibilidade dos sistemas *fuzzy*. Mesmo sendo alvo de crescentes investigações, poucas soluções implementadas e viáveis são encontradas nessa área.

4 Estratégia de Filtragem *Fuzzy* em Bancos de Dados Relacionais e Estudos de Aplicação

4.1 Introdução

Os bancos de dados relacionais disponíveis trabalham em suas consultas com a lógica clássica, reconhecendo-se apenas dois valores, ou seja, verdadeiro ou falso. Ao submeter uma consulta ao SGBD, o mesmo retornará os registros que correspondem totalmente às condições especificadas. Em contrapartida a este paradigma, tem-se a lógica *fuzzy* que, assim como o raciocínio humano, é multivalente, considerando-se então infinitos graus de inclusão a um conjunto e não somente seus extremos.

Em muitos casos, espera-se um resultado aproximado para uma consulta aos dados em um SGBD, onde os registros se enquadrem parcialmente nas condições formuladas. Utilizando-se somente a estrutura tradicional dos SGBDs relacionais, isto se torna uma tarefa impossível.

Assim, considerando a utilização dos SGBDs tradicionais para aplicações do mundo real, multivalorado e cheio de imprecisões, torna-se muito conveniente uni-los à lógica *fuzzy*, o quê possibilitaria que os resultados das consultas poderiam ser aquelas mais próximas do esperado pelo usuário, em relação a um contexto de aplicação com características *fuzzy*.

Esse conceito será aplicado nesse trabalho em um mecanismo que resultará na ordenação de municípios do estado de São Paulo, onde haveria maior viabilidade de implantação de uma empresa de turismo de negócios.

Segundo investigações em agências de turismo do interior de São Paulo, uma empresa de turismo de negócios auxilia executivos com a venda de passagens aéreas, hospedagem, aluguel de veículos e todas as outras facilidades necessárias para uma bem sucedida viagem de negócios. Algumas variáveis referentes aos municípios do estado de São Paulo foram selecionadas, a fim de fornecer indícios de que o município tenha demanda suficiente para implantação de uma empresa desse tipo.

A informação sobre os municípios aqui utilizada refere-se ao ano 2000 e foi obtida por meio da *homepage* do IBGE, sendo assim pública e confiável.

Apesar da aplicação do conceito de filtragem SQL utilizando lógica *fuzzy* no presente caso ser demonstrado por meio de uma aplicação específica, a mesma pode ser facilmente generalizada para solucionar outros tipos de problema.

Para o desenvolvimento de consultas SQL usando teoria de conjuntos *fuzzy* será utilizado o SGBD Microsoft SQL Server, sendo executado em uma máquina local e com linguagem SQL padrão.

4.2 Descrição dos Dados

A informação referente aos municípios do estado de São Paulo encontra-se em uma tabela denominada ‘Cidade’, em que estão armazenados o código e nome da cidade, população, PIB, escolaridade e quantidade de empresas locais (Figura 4.1).

Data in Table 'Cidade' in 'Projeto' on '(local)'

codigo	nome	populacao	qtdemp	pib	estudo
1	Adamantina	33000	1759	169000	6,7
2	Adolfo	3000	105	35000	5,5
3	Aguaí	28000	904	260000	5,6
4	Águas da Prata	7000	286	45000	6,3
5	Águas de Lindóia	16000	791	88000	5,9
6	Águas de Santa Bárbara	5000	273	38000	6
7	Águas de São Pedro	1000	274	15000	8,3
8	Agudos	32000	1001	517000	6,3
9	Alambari	3000	102	24000	5
10	Alfredo Marcondes	3000	111	26000	5,6
11	Altair	3000	122	55000	5,3
12	Altinópolis	15000	620	147000	5,7
13	Alto Alegre	4000	130	22000	5,3
14	Alumínio	15000	347	299000	6,7
15	Álvares Florence	4000	84	21000	5,6

Figura 4.1 – Tabela “Cidade”.

As variáveis foram selecionadas baseando-se em pesquisas com agências de turismo do interior de São Paulo, onde foi levantado o perfil do cliente do turismo de negócios.

A variável indicando a média de anos de Estudo foi selecionada por dar indícios de que existem executivos bem qualificados residindo no município. A variável População e a Quantidade de empresas locais foram selecionadas por determinar se existe demanda suficiente para a implantação de uma nova empresa do tipo selecionado. Finalmente, a variável PIB foi selecionada por indicar o poder financeiro do município, podendo dar indícios de que estão instaladas ali grandes empresas.

É importante ressaltar que os resultados obtidos não têm objetivo comercial, e sim, demonstrar a viabilidade do modelo para soluções de problemas desse tipo. Para que houvesse um resultado com validade comercial, seria então necessário avaliar outros tipos de variáveis adicionais, como, por exemplo, o número de concorrentes no ramo que já estão instalados no município.

Assim, sobre os dados armazenados, assinalam-se os seguintes aspectos: a população disponibilizada na base indica o número total de habitantes residentes no município citado; a escolaridade indica a média de anos de estudo da população

com dez anos ou mais de idade; o PIB representa o Produto Interno Bruto a preço de mercado corrente (em milhares de reais) e a quantidade de empresas locais representa o número de empresas que tenham endereço de atuação no referido município.

As informações sobre população e escolaridade fazem parte dos indicadores sociais do censo demográfico do ano 2000, as quais são disponibilizadas em IBGE (2000a). Em IBGE (2002) estão dispostas as informações do PIB de municípios paulistas, referentes aos períodos de 1999 a 2002. Finalmente, a informação sobre quantidade de empresas locais, referente ao ano 2000, advém do cadastro nacional das empresas, estando disponibilizada em IBGE (2000b). No caso da população e do PIB, os dados foram transformados para múltiplos de mil, observando-se que esse processo não afeta o resultado final.

4.3 Estruturação da Filtragem Fuzzy Usando a Linguagem SQL

Além de armazenar os dados sobre os municípios, a base de dados armazena algumas tabelas adicionais, formando-se então a base de conhecimento do sistema *fuzzy*. Uma tabela foi criada para representar cada função de pertinência das variáveis de entrada e saída, além de uma tabela para representar o conjunto de regras do sistema. Para normalização e melhor visualização do modelo, todas as tabelas envolvidas no processo *fuzzy* iniciam seu nome com os caracteres “*F_”*.

As funções de pertinência foram estruturadas de forma intuitiva, realizando ordenações simples sobre as variáveis e selecionando cortes nos conjuntos a fim de agrupar municípios com perfil semelhante em cada uma das variáveis.

O município de São Paulo tem um perfil muito destacado em relação aos outros municípios, contendo valores de PIB, População e Quantidade de empresas

locais praticamente dez vezes maiores que o próximo município da ordenação. Isso fez com que as funções de pertinência tivessem de ser ajustadas para contemplar tais aspectos, sendo que aquela que representa a variável PIB fora especialmente flexibilizada, para que os outros municípios se distribuíssem melhor em relação aos graus de inclusão na classe ‘Alto’.

Para cada variável de entrada e saída existe uma tabela em que as funções de pertinência são discretizadas de forma independente, armazenando-se somente o suporte (todos os elementos do conjunto com grau de pertinência maior que zero) dos conjuntos *fuzzy* (de forma seqüencial), ou seja, um conjunto após o outro. Essas tabelas possuem a seguinte estrutura (Figura 4.2):

- <nome da variável>_Conj: Armazena a variável representando o conjunto *fuzzy*, onde o registro selecionado está inserido;
- <nome da variável>_Atomic: Armazena o valor atômico (elemento do universo de discurso);
- <nome da variável>_Fuzzy: Armazena o valor *fuzzy* para o referido valor Atômico;
- <nome da variável>_CodTermo: Armazena o termo lingüístico para o conjunto determinado.

Data in Table 'F_Estudo' in 'Pr

E_Atomic	E_Conj	E_Fuzzy	E_CodTermo
5.3	A	.7000	6
5.4	A	.6000	6
5.5	A	.5000	6
5.6	A	.4000	6
5.7	A	.3000	6
5.8	A	.2000	6
5.9	A	.1000	6
5.1	B	.0500	7
5.2	B	.1000	7
5.3	B	.1500	7
5.4	B	.2000	7
5.5	B	.2500	7
5.6	B	.3000	7
5.7	B	.3500	7
5.8	B	.4000	7
5.9	B	.4500	7

Figura 4.2 – Função de pertinência da variável de entrada Estudo armazenada na tabela F_Estudo.

A função de pertinência da variável de entrada *P*(PIB) foi armazenada na tabela F_PIB, sendo que os valores atômicos são múltiplos de 1000, dentro de um universo de discurso U_P = [0, 130.000.000]. O conjunto de termos que compõe essa variável é definido como T_P = {Baixo (P_A), Médio (P_B), Alto (P_C)}, conforme função de pertinência mostrada na Figura 4.3.

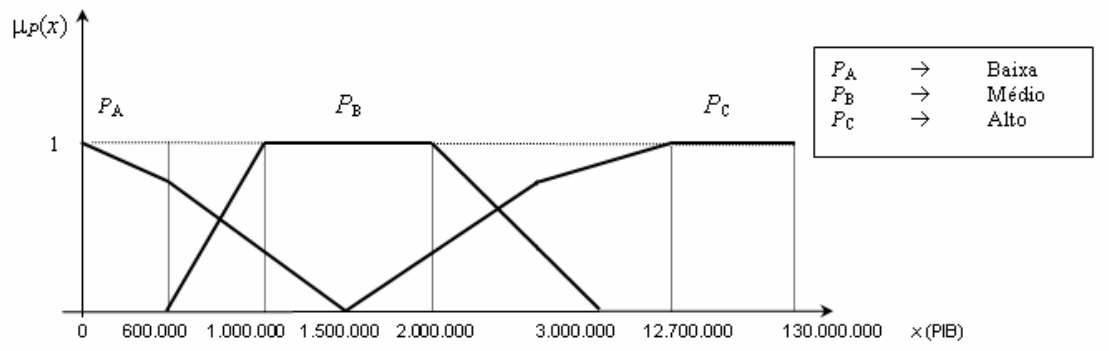


Figura 4.3 – Função de pertinência (μ_P) para a variável PIB.

Na tabela F_Estudo está armazenada a função de pertinência da variável de entrada *E*(Estudo). Os valores atômicos são múltiplos de 0,1, dentro de um universo de discurso U_E = [0, 10], cujos graus de pertinência associados são definidos em função de μ_E (Figura 4.4). O conjunto de termos da variável Estudo é definido como T_E = {Pouco (E_A), Aceitável (E_B)}.

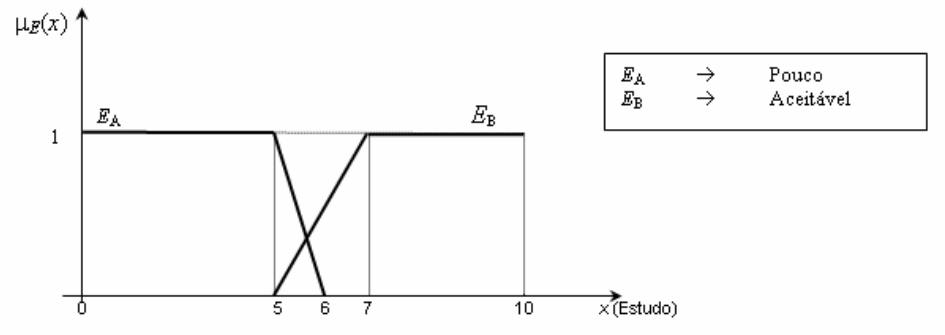


Figura 4.4 – Função de pertinência (μ_E) para a variável Estudo.

A quantidade de empresas locais representada pela variável de entrada *Q*(Qtd_Empresas) está armazenada na tabela F_Qtd, pertencendo ao universo de discurso U_Q=[0, 535.000], com valores atômicos múltiplos de 1. A função de pertinência dessa variável é representada por μ_Q (Figura 4.5) e seu conjunto de termos é T_Q = {Pequena(Q_A), Normal(Q_B)}.

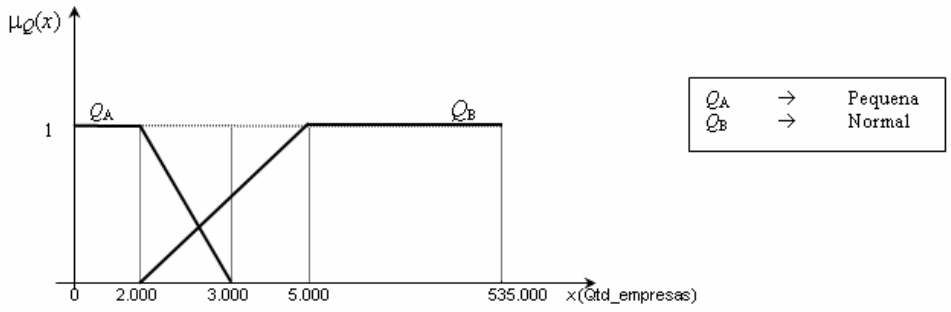


Figura 4.5 – Função de pertinência (μ_Q) para a variável Qtd_Empresas.

A variável de entrada *L*(População) representa a população residente no município, tendo sua função de pertinência μ_L representada pela Figura 4.6, dentro de um universo de discurso U_L = [0, 11.000.000]. Essa função está armazenada na tabela F_Populacao, em múltiplos de 1.000, e tem o conjunto de termos *fuzzy definido por* T_L = {Baixa(L_A), Média(L_B), Grande(L_C)}.

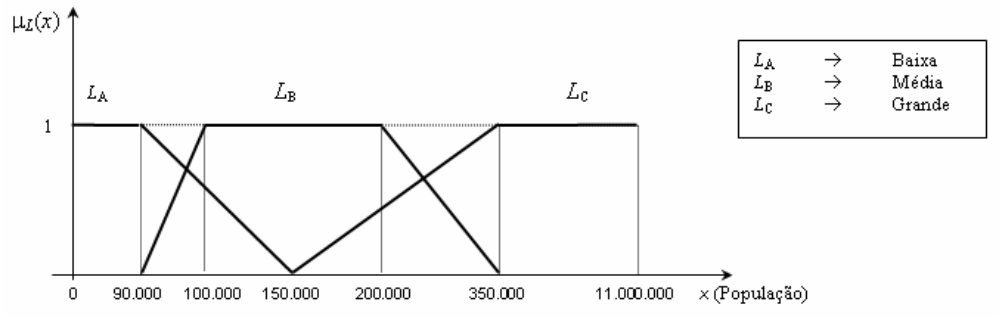


Figura 4.6 – Função de pertinência (μ_L) para a variável População.

A variável de saída *R*(Ranking) representa o ranking dos municípios propícios à implantação de uma empresa de turismo de negócios, de acordo com a inferência *fuzzy* realizada. Seu universo de discurso é representado por $U_R = [0, 1]$ e seu conjunto de termos T_R = {Ruim(R_A), Regular(R_B), Médio(R_C), Bom(R_D), Muito Bom(R_E)} é armazenado na tabela F_Rank em múltiplos de 0,001, segundo a função de pertinência mostrada na Figura 4.7.

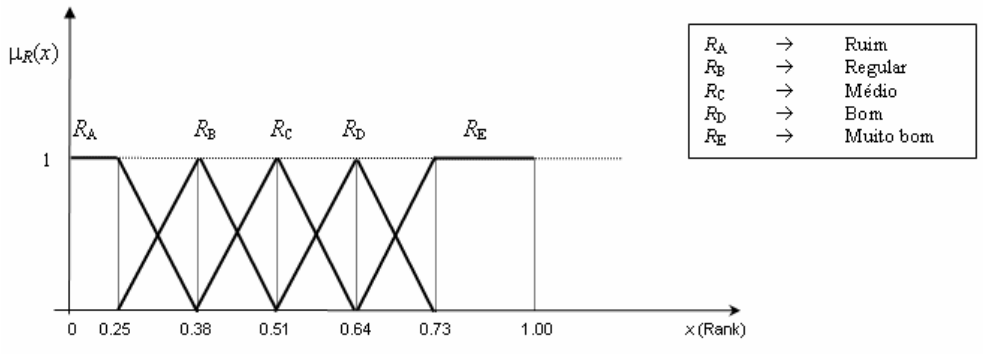


Figura 4.7 – Função de pertinência (μ_R) para a variável de saída Ranking.

O conjunto de regras do sistema *fuzzy* em questão foi armazenado em uma tabela chamada F_Regras (Figura 4.8). Nessa tabela estão armazenadas todas as possíveis combinações de variáveis de entrada e seu respectivo conjunto *fuzzy* de saída. Os atributos *Q*, *P*, *E* e *L* representam as variáveis de entrada e o atributo *R* o respectivo conjunto *fuzzy* de saída.

Data in Table 'F_Regras' in 'Projeto' on '(local)'

Q	P	E	L	R
A	A	A	A	A
A	A	A	B	A
A	A	A	C	B
A	A	B	A	A
A	A	B	B	B
A	A	B	C	B
A	B	A	A	B
A	B	A	B	C
A	B	A	C	C
A	B	B	A	B
A	B	B	B	C
A	B	B	C	D
A	C	A	A	B
A	C	A	B	D
A	C	A	C	C
A	C	B	A	C
A	C	B	B	E
A	C	B	C	E
B	A	A	A	A
B	A	A	B	B
B	A	A	C	B
B	A	B	A	B
B	A	B	B	B
B	A	B	C	C
B	B	A	A	B
B	B	A	B	C
B	B	A	C	C
B	B	B	A	C
B	B	B	B	D
B	B	B	C	D
B	C	A	A	C
B	C	A	B	E
B	C	A	C	D
B	C	B	A	D
B	C	B	B	D
B	C	B	C	E

Figura 4.8 – Tabela F_Regras representando todas as possíveis regras do sistema fuzzy.

Os termos referentes às variáveis lingüísticas estão armazenados em uma tabela chamada F_Termos, conforme a Figura 4.9. Essa tabela contém um atributo CodTermo que armazena um código único para cada termo dentro do sistema e um atributo Termo que guarda as descrições dos termos. A mesma se relaciona por meio do campo CodTermo com o campo <nome da váriavel>_CodTermo de todas as tabelas referentes às variáveis de entrada e saída.

Data in Table 'f_Termos' in 'Projeto' on '(local)'

CodTermo	Termo
1	Pequena
2	Normal
3	Baixa
4	Media
5	Grande
6	Pouco
7	Aceitavel
8	Baixo
9	Medio
10	Alto
11	Ruim
12	Regular
13	Medio
14	Bom
15	Muito Bom

Figura 4.9 – Tabela F_Termos representando os termos lingüísticos do sistema.

A inferência *fuzzy* é realizada por intemédio de uma *view* chamada Cidade_Fuzzy (Figura 4.10), onde o usuário tem a possibilidade de realizar diversas consultas, tratando-a como se fosse uma tabela do banco de dados. Essa inferência segue o padrão de sistemas *fuzzy* do tipo Mamdani, utilizando consultas aninhadas.

```
CREATE View Cidade_Fuzzy as
Select d.codigo, d.Nome, d.Cda Rank, f_rank.r_conj Classe, f_rank.r_fuzzy Fuzzy, F_Termos.Termo from
(select convert(numeric(15,4),SUM(c.Fuzzy * c.r_atomic) / SUM(c.Fuzzy)) CDA, c.nome, c.codigo from
(Select Max(b.Fuzzy) Fuzzy, b.r_atomic, b.nome, b.codigo from
(select CASE WHEN f_rank.r_Fuzzy <= a.Minimo THEN r_Fuzzy
        else a.Minimo
        end as Fuzzy,
f_rank.r_atomic, a.codigo, a.nome, f_regras.R from
(select l_conj, p_conj, e_conj, q_conj, q_fuzzy, p_fuzzy, e_fuzzy, l_fuzzy, nome, codigo,
        CASE WHEN q_fuzzy <= p_fuzzy AND q_fuzzy <= e_fuzzy AND q_fuzzy <= l_fuzzy THEN q_fuzzy
             WHEN p_fuzzy <= q_fuzzy AND p_fuzzy <= e_fuzzy AND p_fuzzy <= l_fuzzy THEN p_fuzzy
             WHEN e_fuzzy <= q_fuzzy AND e_fuzzy <= p_fuzzy AND e_fuzzy <= l_fuzzy THEN e_fuzzy
             WHEN l_fuzzy <= q_fuzzy AND l_fuzzy <= p_fuzzy AND l_fuzzy <= e_fuzzy THEN l_fuzzy
        end as Minimo
   from cidade
   inner join f_populacao
   on (f_populacao.l_atomic = cidade.populacao)
   inner join f_pib
   on (f_pib.p_atomic = cidade.pib)
   inner join f_estudo
   on (f_estudo.e_atomic = cidade.estudo)
   inner join f_qtd
   on (f_qtd.q_atomic = cidade.qtdemp)
) a
inner join f_Regras
on ( (f_Regras.Q = a.Q_conj) and (f_Regras.P = a.P_conj) and
     (f_Regras.L = a.L_conj) and (f_Regras.E = a.E_conj) )
inner join f_Rank
on ( F_regras.R = f_Rank.r_conj)
) b
group by b.r_atomic, b.codigo, b.nome
) c
group by c.codigo, c.nome
) d
inner join f_Rank
on (d.Cda = f_Rank.r_atomic)
inner join f_Termos
on (f_Rank.r_CodTermo = F_Termos.CodTermo)
```

Figura 4.10 – View Cidade_Fuzzy para realizar a inferência fuzzy.

A subconsulta denominada ‘**a**’ *r*ealiza a fuzzificação, obtendo os conjuntos *fuzzy* ativos a partir das variáveis de entrada (tabela Cidade) e determinando o mínimo valor *fuzzy* ativado. Para tal, a mesma utiliza junções internas (*inner join*) para relacionar as variáveis armazenadas na tabela Cidade com seus respectivos valores *fuzzy* (tabelas F_Populacao, F_Pib, F_Estudo e F_Qtd), obtendo-se a classe de cada conjunto *fuzzy* ativo para cada variável. Nesta subconsulta também é utilizada uma estrutura CASE a fim de obter o mínimo valor *fuzzy* dentre todos os conjuntos ativados. Nesse caso, é impossível utilizar a função MIN, pois a mesma retorna o valor mínimo em uma coluna na tabela de resultado, não sendo aplicável a retornar o mínimo valor dentre várias colunas.

A subconsulta ‘**b**’ realiza o processo de inferência, encontrando todas as regras ativadas e transformando cada uma delas em um conjunto *fuzzy* de saída.

Nessa parte da consulta é realizado o processo de inferência do tipo Mamdani, que utiliza as classes de entrada obtidas na subconsulta ‘**a**’ para determinar as classes de saída, por meio da junção interna das classes de entrada com a tabela F_Regras.

Para cada conjunto de saída ativo (podendo ser mais de um), utiliza-se uma junção interna com a tabela F_Rank para selecionar a função que representa o conjunto encontrado. Finalmente, a estrutura CASE monta o conjunto *fuzzy* de saída, mantendo-se da função de saída os valores *fuzzy* menores ou iguais ao mínimo encontrado na consulta ‘**a**’; caso contrário, o valor considerado é o próprio mínimo.

Complementando o processo de inferência, a subconsulta ‘**c’** realiza a agregação de todos os conjuntos de saída ativos, gerando-se um conjunto *fuzzy* de saída. Para tal, é utilizada a função MAX que realiza a união dos conjuntos *fuzzy*

ativos, mantendo-se para cada valor atômico do universo de discurso de saída somente o maior valor *fuzzy* encontrado.

A defuzzificação é realizada pela subconsulta '**d**', por meio do método Centro de Área, gerando-se como resultado, no presente caso, um valor de ranking para o município, considerando para tanto as cidades mais propícias ao investimento na área de turismo de negócios. A função CONVERT é utilizada para padronizar a visualização do resultado do Rank com quatro casas decimais.

Finalmente, a consulta '**e**' determina a classe de investimento do município (Ruim, Regular, Médio, Bom e Muito Bom), podendo o mesmo município ter participação em mais de uma classe de saída, com graus *fuzzy* variados.

Considerando o modelo de sistema *fuzzy* explicitado na Figura 3.2, uma representação gráfica do processamento da *view* é mostrada na Figura 4.11.

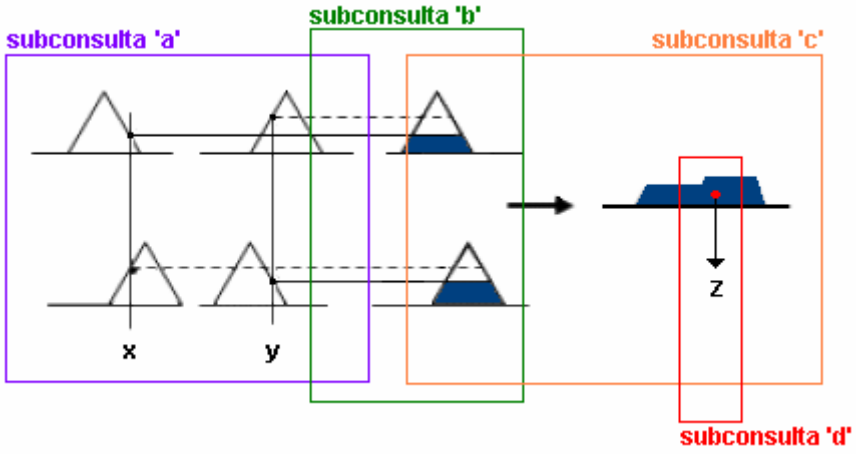


Figura 4.11 – Representação gráfica do processamento realizado pela view *Cidade_Fuzzy.*

4.4 Resultados de Aplicação da Filtragem Fuzzy em Problemas de Ordenação de Informações

Para obter os resultados da inferência sobre os dados, é necessário executar comandos sobre a *view* Cidade_Fuzzy.

Ao executar um bloco de comando básico, solicitando todos os dados da *view*, obtêm-se os seguintes campos (Figura 4.12):

- Codigo: Código do município, cadastrado no campo de mesmo nome dentro da tabela Cidade;
- Nome: Nome do município, cadastrado no campo de mesmo nome, dentro da tabela Cidade;
- Rank: Representa a posição do município selecionado dentro do ranking de cidades propícias à implantação da empresa de turismo de negócios. Fornece o valor encontrado pela inferência *fuzzy* realizada pelo modelo;
- Classe: Mostra a variável que representa o conjunto *fuzzy* de saída em que o município está incluído;
- Fuzzy: Mostra o grau de inclusão no conjunto;
- Termo: Mostra o valor lingüístico do conjunto selecionado.

Em casos em que o valor ‘Rank’ encontrado para um município possuir valores *fuzzy* maior que zero em mais de um conjunto (termo) de saída, a consulta retornará várias linhas para o mesmo município, mostrando o grau de pertinência em cada um dos conjuntos. Como exemplo desta situação, tem-se as linhas 20 e 21 da Figura 4.12.

```
select * from cidade_fuzzy
order by rank desc
```

	codigo	Nome	Rank	Classe	Fuzzy	Termo
1	565	São Paulo	.8415	E	1.0000	Muito Bom
2	560	São José dos Campos	.8411	E	1.0000	Muito Bom
3	547	São Bernardo do Campo	.8403	E	1.0000	Muito Bom
4	109	Campinas	.8399	E	1.0000	Muito Bom
5	537	Santo André	.8388	E	1.0000	Muito Bom
6	215	Guarulhos	.8384	E	1.0000	Muito Bom
7	390	Osasco	.8383	E	1.0000	Muito Bom
8	584	Sorocaba	.8379	E	1.0000	Muito Bom
9	490	Ribeirão Preto	.8378	E	1.0000	Muito Bom
10	545	Santos	.8375	E	1.0000	Muito Bom
11	154	Diadema	.8353	E	1.0000	Muito Bom
12	333	Mauá	.8353	E	1.0000	Muito Bom
13	436	Piracicaba	.8212	E	1.0000	Muito Bom
14	296	Jundiaí	.8187	E	1.0000	Muito Bom
15	559	São José do Rio Preto	.7730	E	1.0000	Muito Bom
16	306	Limeira	.7614	E	1.0000	Muito Bom
17	607	Taubaté	.7575	E	1.0000	Muito Bom
18	68	Bauru	.7537	E	1.0000	Muito Bom
19	588	Suzano	.7475	E	1.0000	Muito Bom
20	65	Barueri	.7278	D	.0244	Bom
21	65	Barueri	.7278	E	.9756	Muito Bom
22	586	Sumaré	.7244	D	.0622	Bom
23	586	Sumaré	.7244	E	.9378	Muito Bom

Figura 4.12 – Resultado de uma consulta total à view de inferência.

Para obter o resultado do conjunto global, em que o município tem maior grau de inclusão, é necessário agrupar os dados selecionando-se o registro que têm maior valor *fuzzy* dentre todos os encontrados para o determinado município. A Figura 4.13 mostra a consulta utilizada e o resultado obtido.

```
Select Nome, Rank, Termo, Fuzzy from Cidade_Fuzzy
inner join f_Rank
on (Rank = f_Rank.r_atomic)
where Fuzzy = (select max(r_fuzzy) from f_rank where r_atomic = Rank)
group by Nome, Rank, Termo, Fuzzy
order by Rank desc
```

	Nome	Rank	Termo	Fuzzy
1	São Paulo	.8415	Muito Bom	1.0000
2	São José dos Campos	.8411	Muito Bom	1.0000
3	São Bernardo do Campo	.8403	Muito Bom	1.0000
4	Campinas	.8399	Muito Bom	1.0000
5	Santo André	.8388	Muito Bom	1.0000
6	Guarulhos	.8384	Muito Bom	1.0000
7	Osasco	.8383	Muito Bom	1.0000
8	Sorocaba	.8379	Muito Bom	1.0000
9	Ribeirão Preto	.8378	Muito Bom	1.0000
10	Santos	.8375	Muito Bom	1.0000
11	Diadema	.8353	Muito Bom	1.0000
12	Mauá	.8353	Muito Bom	1.0000
13	Piracicaba	.8212	Muito Bom	1.0000
14	Jundiaí	.8187	Muito Bom	1.0000
15	São José do Rio Preto	.7730	Muito Bom	1.0000
16	Limeira	.7614	Muito Bom	1.0000
17	Taubaté	.7575	Muito Bom	1.0000
18	Bauru	.7537	Muito Bom	1.0000
19	Suzano	.7475	Muito Bom	1.0000
20	Barueri	.7278	Muito Bom	.9756
21	Sumaré	.7244	Muito Bom	.9378
22	Guarujá	.7208	Muito Bom	.8978
23	Cubatão	.7188	Muito Bom	.8756

Figura 4.13 – Resultado de consulta mostrando somente a classe com maior grau de inclusão do município.

Para selecionar os municípios que tem alguma participação no conjunto considerado com grau ‘Bom’ de investimento, utilizou-se uma consulta conforme mostrado na Figura 4.14. Nota-se que a primeira cidade do ranking tem um pequeno grau de participação nesse conjunto (termo ‘Bom’). Isso se deve ao fato de o valor discreto do ranking ser referente ao sistema como um todo, e não necessariamente a cidade que tem maior valor de ativação dentro de um conjunto (termo) específico será aquela com maior grau de inclusão no conjunto selecionado. De fato, se assim fosse, a cidade de Franca (linha 11) estaria no topo da lista, pois é aquela que possui a maior ativação quando se considera o termo ‘Bom’.

```
select * from cidade_fuzzy
where Termo = 'Bom'
order by rank desc
```

	codigo	Nome	Rank	Classe	Fuzzy	Termo
1	65	Barueri	.7278	D	.0244	Bom
2	586	Sumaré	.7244	D	.0622	Bom
3	214	Guarujá	.7208	D	.1022	Bom
4	151	Cubatão	.7188	D	.1244	Bom
5	280	Jacareí	.7085	D	.2389	Bom
6	549	São Carlos	.7080	D	.2444	Bom
7	591	Taboão da Serra	.7021	D	.3100	Bom
8	19	Americana	.6973	D	.3633	Bom
9	494	Rio Claro	.6700	D	.6667	Bom
10	37	Araraquara	.6675	D	.6944	Bom
11	188	Franca	.6441	D	.9544	Bom
12	220	Hortolândia	.6321	D	.9392	Bom
13	548	São Caetano do Sul	.6263	D	.8946	Bom
14	146	Cotia	.6192	D	.8400	Bom
15	413	Paulínia	.6186	D	.8354	Bom
16	238	Indaiatuba	.6163	D	.8177	Bom
17	346	Mogi Guaçu	.5708	D	.4677	Bom
18	122	Carapicuíba	.5672	D	.4400	Bom
19	275	Itu	.5663	D	.4331	Bom
20	347	Moji das Cruzes	.5652	D	.4246	Bom
21	467	Presidente Prudente	.5461	D	.2777	Bom
22	329	Marília	.5368	D	.2062	Bom
23	516	Santa Bárbara d'Oeste	.5329	D	.1762	Bom

Figura 4.14 - Resultado de consulta para Municípios com grau de investimento 'Bom'.

Observa-se também na Figura 4.14 que a cidade de 'Barueri' tem o maior valor de ranking dentro da classe 'Bom'; porém, ao observar a Figura 4.12 (linha 21) nota-se que a mesma está incluída em maior grau *fuzzy* na classe 'Muito Bom'. Tal fato comprova a coerência realizada pela filtragem *fuzzy* desenvolvida.

4.5 Análise Comparativa de Resultados Obtidos

4.5.1 Estudo de Caso 1

Para realizar uma busca tradicional à base de dados, visando-se obter as cidades que se enquadram na categoria de investimento 'Muito Bom', foi utilizada a seguinte consulta:

```
select Codigo, Nome, QtdEmp, Populacao, Estudo, Pib from Cidade
where QtdEmp > 2000
and Populacao > 150000
and Estudo > 5
and Pib > 1500000
```

Em comparação com as funções de pertinência definidas para o modelo *fuzzy*, foram consideradas incluídas na categoria ‘Muito Bom’ as cidades que tenham valor de pertinência maior que zero nos conjuntos Q_B (Qtd_Empresas = ‘Normal’), L_C (População = ‘Grande’), E_B (Estudo = ‘Aceitável’) e P_C (Pib = ‘Alto’), conforme destacado na Figura 4.15.

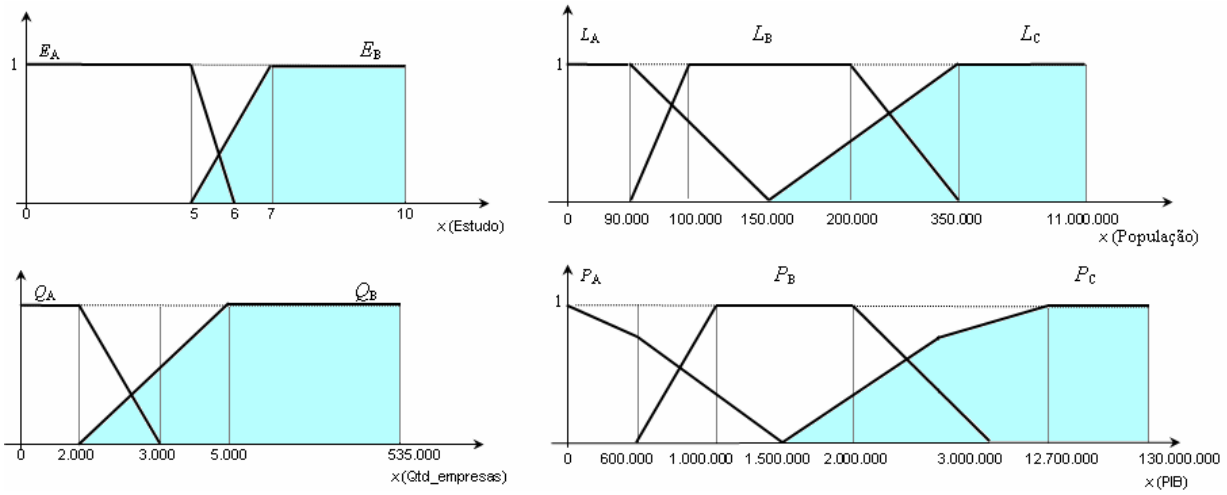


Figura 4.15 – Visualização gráfica da regra utilizada como base para as consultas tradicionais – Estudo de Caso 1.

Para gerar um ranking de cidades por meio do método tradicional, é necessário ordenar os registros por uma das quatro variáveis utilizadas, sendo possível considerar somente uma das variáveis a cada consulta, não se levando em conta as outras informações. Assim, para gerar um ranking unificado, seria necessário o conhecimento de um especialista para analisar todos os rankings gerados e, por intermédio de sua experiência, determinar a ordenação das informações. Além disso, os rankings produzidos podem variar de especialista para especialista.

O diferencial do modelo *fuzzy* é que o mesmo vai gerar um ranking único automaticamente, considerando de forma paralela e integrada todas as variáveis envolvidas no processo.

O Quadro 4.1 faz uma comparação entre os resultados obtidos pela forma tradicional e pela forma *fuzzy*, mostrando-se os quinze primeiros registros. O primeiro bloco do quadro mostra o resultado do ranking utilizando o modelo *fuzzy* e os outros blocos mostram o ranking utilizando o modelo convencional, com as quatro ordenações possíveis.

Quadro 4.1 – Comparação de resultados de consultas do Estudo de Caso 1: Modelo tradicional x Modelo Fuzzy.

	Consulta *Fuzzy*		**Consultas Tradicionais**							
	Rank *Fuzzy*		**Rank por População**		**Rank por PIB**		**Rank por Estudo**		**Rank por Qtd. Empresas**	
01	São Paulo	0.8415	São Paulo	10.435.000	São Paulo	127.437.000	Santos	8,5	São Paulo	534.258
02	São José dos Campos	0.8411	Guarulhos	1.072.000	São José dos Campos	13.496.000	Campinas	7,7	Campinas	46.435
03	São Bernardo do Campo	0.8403	Campinas	969.000	Guarulhos	12.238.000	Ribeirão Preto	7,7	Ribeirão Preto	30.084
04	Campinas	0.8399	São Bernardo do Campo	703.000	São Bernardo do Campo	11.129.000	Santo André	7,6	Guarulhos	28.830
05	Santo André	0.8388	Osasco	652.000	Campinas	10.010.000	São Bernardo do Campo	7,6	São Bernardo do Campo	26.667
06	Guarulhos	0.8384	Santo André	649.000	Santo André	6.828.000	São José dos Campos	7,6	Santos	25.698
07	Osasco	0.8383	São José dos Campos	539.000	Barueri	6.088.000	São Paulo	7,6	Santo André	23.902
08	Sorocaba	0.8379	Ribeirão Preto	504.000	Osasco	5.492.000	Taubaté	7,4	São José do Rio Preto	22.262
09	Ribeirão Preto	0.8378	Sorocaba	493.000	Jundiaí	5.289.000	São José do Rio Preto	7,4	São José dos Campos	20.887
10	Santos	0.8375	Santos	417.000	Sorocaba	4.384.000	Bauru	7,4	Sorocaba	19.556
11	Mauá	0.8353	Mauá	363.000	Ribeirão Preto	3.907.000	São Carlos	7,4	Osasco	18.092
12	Diadema	0.8353	São José do Rio Preto	358.000	Diadema	3.894.000	Araraquara	7,4	Bauru	15.086
13	Piracicaba	0.8212	Diadema	357.000	Taubaté	3.773.000	Jundiaí	7,2	Jundiaí	14.676
14	Jundiaí	0.8187	Piracicaba	329.000	Mauá	3.283.000	Americana	7,1	Barueri	14.167
15	São José do Rio Preto	0.7730	Jundiaí	323.000	Santos	3.202.000	Rio Claro	7,1	Piracicaba	13.509

É possível observar que a cidade de ‘São Paulo’ tem destaque em quase todas as formas de consulta, devido ao fato de ser muito maior do que todas as outras cidades do estado. Mesmo com essas características marcantes, a mesma não aparece em primeiro lugar se for considerado o fator ‘Estudo’, em que a mesma tem a terceira posição juntamente como outras três cidades.

Observa-se também que a cada ordenação solicitada, um novo grupo ordenado de cidades aparece, sendo apenas possível selecionar uma variável como determinante, deixando-se portanto de considerar as outras variáveis.

Ao tomar como exemplo a cidade de ‘Piracicaba’, que no ranking *fuzzy* é a décima terceira cidade com maior potencial de investimento, observa-se que a mesma não está no ranking das quinze melhores cidades quando se leva em conta as variáveis PIB e Estudo como critério para ranking.

Considerando agora a variável PIB, observa-se que na sétima posição do ranking se encontra a cidade de ‘Barueri’. Essa cidade, apesar de estar bem classificada nesse ranking, tem as outras variáveis mais fracas, resultando-se na inferência *fuzzy* uma classificação abaixo das quinze melhores (ocupa a vigésima posição).

Resultados como esses poderiam ser prejudiciais ao estudo como um todo, excluindo-se cidades potencialmente viáveis à implantação de uma empresa, ou ainda, selecionando-se cidades com baixo potencial.

4.5.2 Estudo de Caso 2

Com intuito de obter cidades com potencial de investimento classificado como ‘Ruim’, foram consideradas incluídas na categoria as cidades que tenham valor de pertinência maior que zero nos conjuntos *fuzzy* Q_A (Qtd_Empresas = ‘Pequena’), L_A (População = ‘Baixa’), E_A (Estudo = ‘Pouco’) e P_A (Pib = ‘Baixo’), conforme ilustrado na Figura 4.16. Para tal finalidade, utilizou-se a seguinte consulta ao modelo convencional:

```
select Codigo, Nome, QtdEmp, Populacao, Estudo, Pib from Cidade
where QtdEmp < 3000
and Populacao < 150000
and Estudo < 6
and Pib < 1500000
```

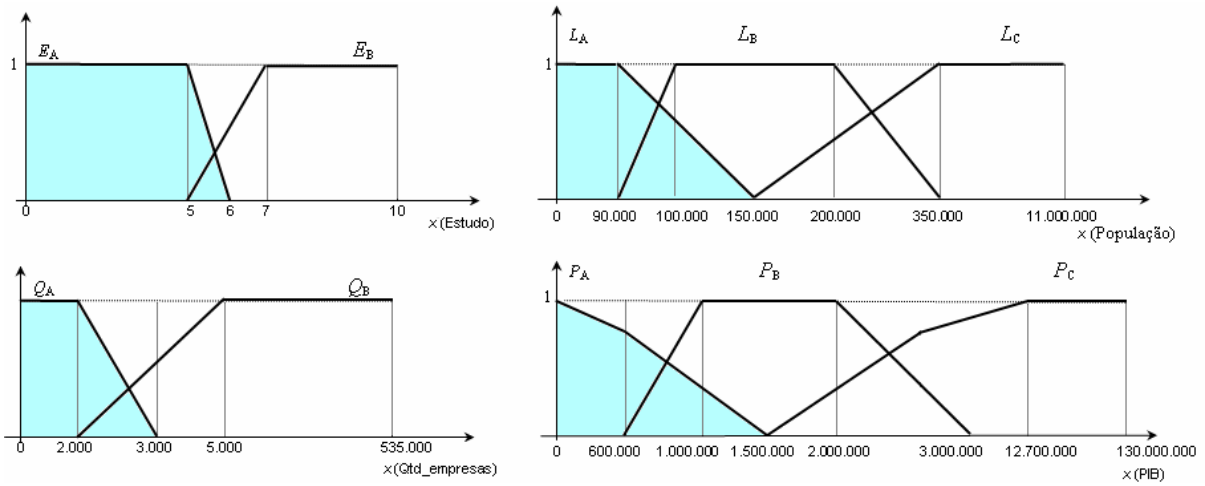


Figura 4.16 – Visualização gráfica da regra utilizada como base para as consultas tradicionais – Estudo de Caso 2.

Nota-se na Figura 4.8 que existem quatro regras que definem uma cidade com potencial de investimento ‘Ruim’ (Coluna R, com valores iguais a ‘A’). Para desenvolvimento dessa consulta foi selecionada apenas uma das regras, pelo fato de que, para cada regra existente, um bloco condicional contendo todas as variáveis seria necessário, o que seria inviável em um sistema com muitas regras.

Para obter os dados por meio da inferência *fuzzy*, utilizou-se o seguinte bloco de comandos:

```
Select * from Cidade_fuzzy
where termo = 'Ruim'
order by Rank desc
```

O Quadro 4.2 mostra o resultado da consulta utilizando a metodologia *fuzzy* e a metodologia padrão com todas as ordenações possíveis. A primeira linha do quadro representa as cidades com melhor situação de investimento dentro da classe selecionada, detectada por meio das diferentes metodologias de pesquisa.

Para o resultado da consulta tradicional organizada pela variável Estudo, observou-se que as quinze primeiras cidades possuem o mesmo valor de anos de estudo, sendo utilizado como critério de ordenação secundário a variável PIB.

Quadro 4.2 – Comparação de resultados de consultas do Estudo de Caso 2: Modelo tradicional x Modelo Fuzzy.

	Consulta *Fuzzy*		**Consultas Tradicionais**							
	Rank *Fuzzy*		**Rank por População**		**Rank por PIB**		**Rank por Estudo**		**Rank por Qtd. Empresas**	
01	Jaguariúna	0.3637	Francisco Morato	133.000	Cajamar	883.000	Cajamar	5.9	Itapeva	2.996
02	Itapecerica da Serra	0.3226	Itapecerica da Serra	129.000	Castilho	656.000	Itapecerica da Serra	5.9	Itapecerica da Serra	2.776
03	Ferraz de Vasconcelos	0.3174	Franco da Rocha	108.000	Itapecerica da Serra	649.000	Várzea Paulista	5.9	Campos do Jordão	2.618
04	Tupã	0.3114	Várzea Paulista	92.000	Louveira	566.000	Franco da Rocha	5.9	Juquitiba	2.478
05	Jaboticabal	0.3054	Itapeva	82.000	Várzea Paulista	545.000	Guaíra	5.9	Várzea Paulista	1.988
06	São Sebastião	0.2971	Ibiúna	64.000	Ouroeste	506.000	Bastos	5.9	Cajamar	1.981
07	Vinhedo	0.2958	Cajamar	50.000	Franco da Rocha	492.000	Novo Horizonte	5.9	Francisco Morato	1.841
08	Ilha Solteira	0.2801	Piedade	50.000	Itapeva	400.000	Campos do Jordão	5.9	Santa Isabel	1.830
09	Lorena	0.2798	Capão Bonito	46.000	Cabreúva	383.000	Artur Nogueira	5.9	Porto Feliz	1.817
10	Caçapava	0.2791	Itararé	46.000	Monte Mor	373.000	Palmital	5.9	São Lourenço da Serra	1.752
11	Itapeva	0.2751	Porto Feliz	45.000	Itápolis	367.000	Brotas	5.9	Itápolis	1.717
12	Lins	0.2675	Campos do Jordão	44.000	Salto de Pirapora	349.000	Colina	5.9	Socorro	1.605
13	Votorantim	0.2593	Santa Isabel	43.000	Morro Agudo	322.000	Tanabi	5.9	Novo Horizonte	1.598
14	Leme	0.2529	Itápolis	37.000	Guaíra	321.000	Lucélia	5.9	São Manuel	1.576
15	Mococa	0.2464	Monte Mor	37.000	Cândido Mota	314.000	Rafard	5.9	Ibiúna	1.555

Nesse caso, a análise decisória se torna ainda mais complicada, pois a cidade de ‘Jaguariúna’ encontra-se como a primeira cidade dentro da classe ‘Ruim’, não aparecendo em nenhuma das outras consultas tradicionais. Isso acontece pelo fato de essa cidade ativar as seguintes regras:

Qtd_Empresas	Pib	Estudo	Populacao	Rank
Q_A (Pequena)	P_A (Baixo)	E_B (Aceitável)	L_A (Baixa)	R_A (Ruim)
Q_A (Pequena)	P_B (Médio)	E_B (Aceitável)	L_A (Baixa)	R_B (Regular)

Considerando a análise inicial do problema, mesmo a cidade tendo a variável Estudo classificada como ‘Aceitável’, mas se possuir as outras variáveis fracas, a mesma é classificada dentro da classe de saída ‘Ruim’. Ao realizar a defuzzificação por meio do método CDA, considerando todas as regras ativas, a cidade pode se enquadrar na classe ‘Ruim’, mesmo que com um pequeno grau de pertinência, tornando-se uma situação não contemplada por essa consulta tradicional. A mesma situação acontece com a maioria das outras cidades do

ranking *fuzzy*, devido ao fato de que, ao estruturar a consulta tradicional, foi utilizada como critério uma única regra, sendo que a mesma contempla poucos municípios.

4.5.3 Estudo de Caso 3

Com o objetivo de obter as cidades com potencial de investimento classificado como ‘Médio’, será utilizada a seguinte consulta:

```
select Codigo, Nome, QtdEmp, Populacao, Estudo, Pib from Cidade
where (QtdEmp < 3000
and Populacao between 90000 and 350000
and Estudo > 5
and Pib between 60000 and 3000000)
or
(QtdEmp > 2000
and Populacao < 150000
and Estudo > 5
and Pib between 60000 and 3000000)
or
( QtdEmp > 2000
and Populacao < 150000
and Estudo < 6
and Pib > 1500000)
```

sendo consideradas incluídas as cidades que tenham valor de pertinência maior que zero nos seguintes conjuntos *fuzzy:*

Q_A (Qtd_Empresas = ‘Pequena’), L_B (População = ‘Média’), E_B (Estudo = ‘Aceitável’) e P_B (Pib = ‘Médio’), conforme ilustrado na Figura 4.17;

OU

Q_B (Qtd_Empresas = ‘Normal’), L_A (População = ‘Baixa’), E_B (Estudo = ‘Aceitável’) e P_B (Pib = ‘Médio’), conforme ilustrado na Figura 4.18;

OU

Q_B (Qtd_Empresas = ‘Normal’), L_A (População = ‘Baixa’), E_A (Estudo = ‘Pouco’) e P_C (Pib = ‘Alto’), conforme ilustrado na Figura 4.19.

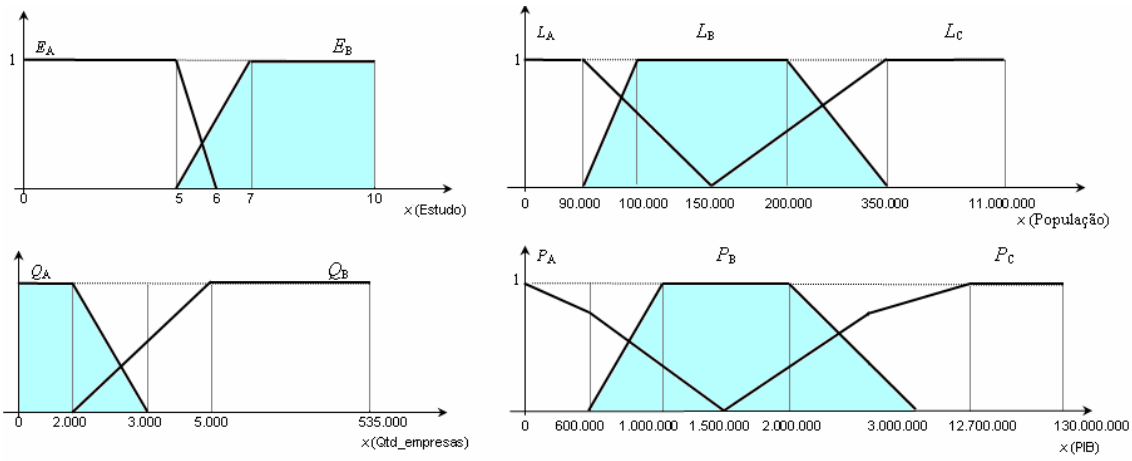


Figura 4.17 – Visualização gráfica da regra 1 utilizada como base para as consultas tradicionais – Estudo de Caso 3.

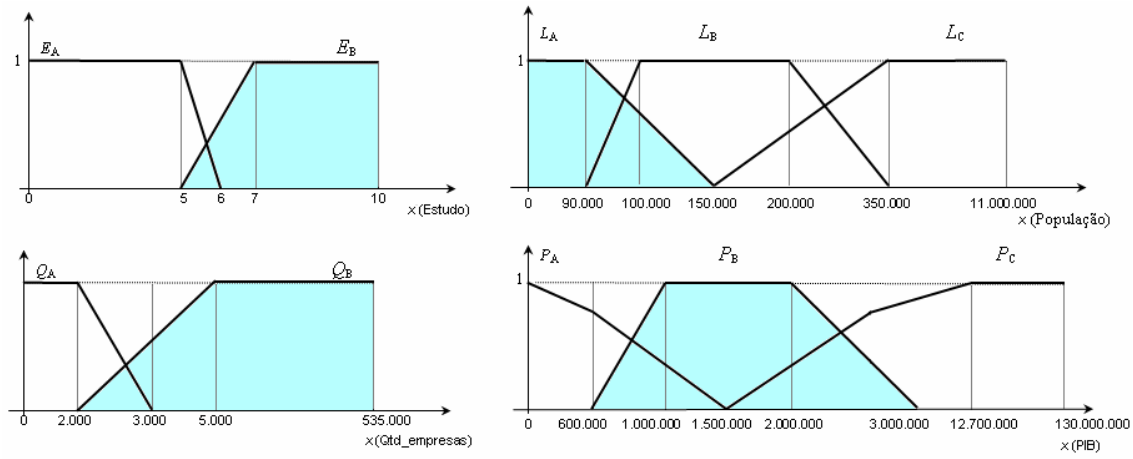


Figura 4.18 – Visualização gráfica da regra 2 utilizada como base para as consultas tradicionais – Estudo de Caso 3.

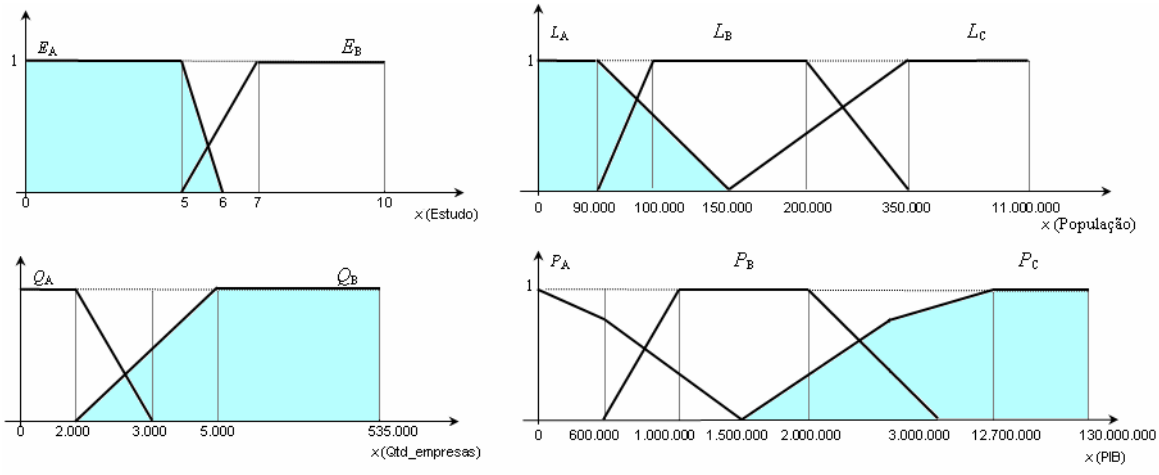


Figura 4.19 – Visualização gráfica da regra 3 utilizada como base para as consultas tradicionais – Estudo de Caso 3.

Nota-se na tabela de regras (Figura 4.8) que a classe selecionada é resultado de dez regras, sendo selecionadas três delas de forma intuitiva. É possível observar também que, ao aumentar o número de regras selecionadas na consulta tradicional, aumentam-se em muito as linhas de código necessárias (acrescentam-se ao menos quatro condições por regra).

O Quadro 4.3 mostra a comparação da consulta realizada pelo método *fuzzy*, em que é considerada a influência de todas as variáveis e regras, com o método tradicional, onde se considera a influência de apenas três das regras.

Quadro 4.3 – Comparação de resultados de consultas do Estudo de Caso 3: Modelo tradicional x Modelo Fuzzy.

	Consulta *Fuzzy*		**Consultas Tradicionais**							
	Rank *Fuzzy*		Rank por População		Rank por PIB		Rank por Estudo		Rank por Qtd. Empresas	
01	Hortolândia	0.6321	Itapevi	162.000	Mogi Guaçu	2.240.000	Assis	7.3	Poá	18.721
02	São Caetano do Sul	0.6263	Cotia	148.000	Indaiatuba	1.519.000	Botucatu	7.3	Santana de Parnaíba	18.012
03	Cotia	0.6192	Indaiatuba	147.000	Cotia	1.437.000	Guaratinguetá	7.3	Cotia	8.505
04	Paulínia	0.6186	Ferraz de Vasconcelos	142.000	Bebedouro	1.320.000	Lins	7.2	Indaiatuba	6.400
05	Indaiatuba	0.6163	Itu	135.000	Matão	1.320.000	Lorena	7.2	Bragança Paulista	6.266
06	Mogi Guaçu	0.5708	Francisco Morato	133.000	Itu	1.314.000	Vinhedo	7.2	Jaú	5.696
07	Carapicuíba	0.5672	Itapecerica da Serra	129.000	Valinhos	1.193.000	Valinhos	7.1	Catanduva	5.570
08	Itu	0.5663	Pindamonhanga-ba	126.000	Pindamo-nhangaba	1.173.000	Ribeirão Pires	7.1	Itu	5.474
09	Moji das Cruzes	0.5652	Itapetininga	125.000	Caçapava	988.000	Cruzeiro	7.1	Atibaia	5.320
10	Presidente Prudente	0.5461	Bragança Paulista	125.000	Araras	985.000	Pindamonhan-gaba	7.0	Botucatu	4.801
11	Marília	0.5368	Mogi Guaçu	124.000	Vinhedo	971.000	Pirassununga	7.0	Barretos	4.675
12	Santa Bárbara d'Oeste	0.5329	Jaú	112.000	Itatiba	960.000	Poá	6.9	Itapetininga	4.664
13	Araçatuba	0.5323	Atibaia	111.000	Itapevi	892.000	Santana de Parnaíba	6.9	Mogi Guaçu	4.648
14	Pindamonhan-gaba	0.5321	Botucatu	108.000	Sertãozinho	886.000	Moji-Mirim	6.9	Sertãozinho	4.531
15	São Vicente	0.5265	Franco da Rocha	108.000	Bragança Paulista	885.000	Caçapava	6.9	Assis	4.471

A cidade de ‘Hortolândia’, que ocupa a primeira posição no ranking dessa classe, ativou as seguintes regras:

Qtd_Empresas	Pib	Estudo	Populacao	Rank
Q_B (Normal)	P_B (Médio)	E_B (Aceitável)	L_B (Média)	R_D (Bom)
Q_B (Normal)	P_B(Médio)	E_B (Aceitável)	L_C (Grande)	R_D (Bom)
Q_B (Normal)	P_C (Alto)	E_B (Aceitável)	L_B (Média)	R_D (Muito Bom)
Q_B (Normal)	P_C (Alto)	E_B (Aceitável)	L_C (Grande)	R_E (Muito Bom)

Mesmo não ativando nenhuma regra com classe de saída ‘Médio’, a mesma aparece nesse ranking com um pequeno grau de pertinência (0.0608) devido ao cálculo de seu CDA (Figura 4.20), estando ainda incluída em maior grau de pertinência (0.9392) na classe ‘Bom’.

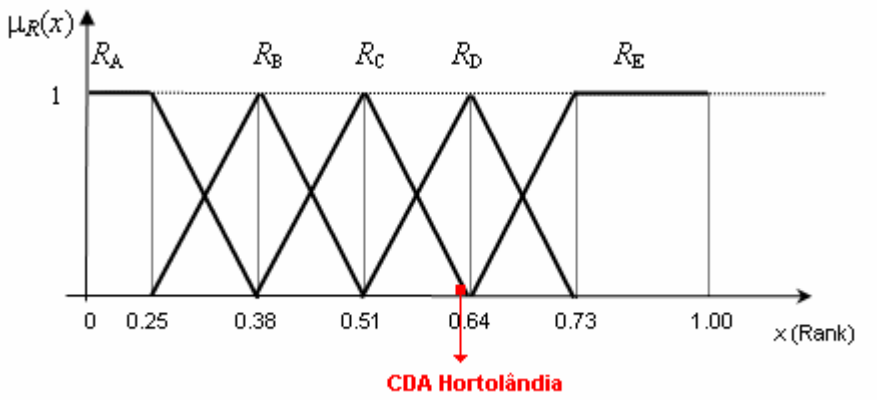


Figura 4.20 – Centro de área do município de Hortolândia.

Mesmo utilizando uma consulta tradicional, em que muitas das cidades do ranking *fuzzy* também aparecem nas listas das consultas ordenadas, nota-se que a variação entre os seus resultados ordenados é muito grande. Sendo assim, torna-se praticamente impossível a um especialista intuir um ranking único utilizando consultas convencionais, necessitando-se para tal o levantamento de muitos outros dados adicionais.

4.6 Considerações Parciais

As consultas realizadas utilizando o modelo *fuzzy* se mostraram bem apropriadas ao inferir a influência de todas as variáveis para a ordenação das cidades no ranking. Pode-se observar que, sem o modelo proposto, seria necessário realizar o processo de avaliação da capacidade dos municípios manualmente, por

meio do sentimento do analista sobre o conteúdo das variáveis, variando de acordo com a sua compreensão e com a ordem das consultas.

As consultas tradicionais se comportam de forma semelhante a sistemas especialistas convencionais, onde um analista humano com domínio na área do problema fornece dados para que um sistema computadorizado analise casos de testes. Porém, esses sistemas utilizam somente a lógica Verdadeiro/Falso, sendo muito bem vinda a agregação proporcionada pelo método de avaliação *fuzzy*.

Foi também possível observar que não é necessário utilizar outras ferramentas externas ao banco de dados para realizar a inferência *fuzzy*, e nem modificar os dados inseridos na base, bastando-se apenas inserir as tabelas referentes à base de conhecimento e adaptar a *view* de acordo com a aplicação.

5 Conclusões e Trabalhos Futuros

Esse trabalho se concentrou na aplicação da teoria de conjuntos *fuzzy* em consultas a bancos de dados. O modelo proposto buscou proporcionar ao especialista uma ferramenta que pudesse auxiliar tanto nas análises de suas buscas como nas tomadas de decisões, quando consultas a bancos de dados existentes for a forma disponível para se decidir ações a serem tomadas.

Foram realizadas comparações do modelo desenvolvido com consultas convencionais em bancos de dados. Os resultados obtidos durante os testes demonstram que o modelo é viável para a complementação de consultas a bancos de dados, obtendo-se resultados de ordenações coerentes com a realidade dos municípios, e realizando-se ainda a consideração das variáveis de forma simultânea sem a necessidade de uma análise posterior de um especialista.

Pode-se observar que essa abordagem apresenta diversas particularidades em relação às consultas tradicionais, destacando-se as seguintes:

- A obtenção de resultados utilizando conjuntos multivalentes flexibiliza a resposta retornada ao usuário, aproximando-o da realidade;
- A consulta analisa todas as variáveis de forma simultânea, considerando a influência de todas as variáveis ao resultado final, não realizando apenas uma ordenação simples;
- As soluções finais do modelo proposto tendem a facilitar as análises a serem efetuadas pelos especialistas frente às consultas realizadas, as quais

poderiam demandar esforços adicionais para avaliar os respectivos resultados.

Por meio do desenvolvimento desse trabalho, foi possível também observar que, ao estruturar as funções de pertinência, valores muito extremos (como no caso do município de São Paulo) distorcem os resultados da consulta *fuzzy.* Para que isso não ocorra é necessária a adaptação das funções, diminuindo-se as ambigüidades (municípios com mesmo valor de rank) na ordenação *fuzzy* e distribuindo-se melhor o resultado.

Foi possível observar ainda que se pode adaptar os sistemas existentes ao modelo proposto sem muita dificuldade e sem que ocorram alterações bruscas nos dados armazenados, mantendo-se intacta a integridade do banco de dados.

Considerando a implementação existente de sistemas *fuzzy* utilizando a linguagem SQL, definida por Veryha (2005), o presente trabalho proposto aqui apresenta algumas diferenças e melhorias.

O método desenvolvido por Veryha (2005) não utiliza o modelo *fuzzy* constituído por regras, sendo possível somente a definição de funções de pertinência regulares com duas classes, em que, considerando-se uma variável em particular, o valor positivo em uma classe representa um valor inversamente negativo na outra classe.

Diferentemente, o presente trabalho proposto aqui permite o desenvolvimento de funções de pertinência com várias classes, não necessariamente regulares, onde os valores de pertinência são definidos de forma totalmente independente para cada classe.

Adicionalmente, constata-se também que o método de Veryha (2005) utiliza algumas extensões da linguagem SQL que são proprietárias (Microsoft). Por outro

lado, o modelo apresentado nesta dissertação utiliza somente comandos definidos pelo padrão SQL:1999.

Em suma, com a realização do presente trabalho, conclui-se que a lógica *fuzzy* tem um potencial muito elevado em aplicações que envolvem a área de sistemas de informação e banco de dados.

Como projeto de trabalhos futuros está a possibilidade de se ampliar os retornos *fuzzy* que são passíveis de serem obtidos, podendo-se pesquisar os termos lingüísticos de qualquer uma das variáveis e obtendo-se resultados mais diversificados.

Propõe-se ainda que a junção dos valores discretos com o respectivo representante *fuzzy* seja feito por meio de uma busca por similaridade, ampliando-se a seleção por igualdade, conforme utilizada nesse trabalho.

Referências Bibliográficas

ASTRAHAN, M.M., BLASGEN, M.W., CHAMBERLIN, D.D., et al. (1976). System R: Relational approach to database management. *ACM Transactions on Database Systems*, vol. 2, no. 1, pp. 97-137.

BRANCO, A.C.S. (2004). *Geração de Fuzzy Queries para Mineração de Dados*. Tese de Doutorado, Universidade Federal do Rio de Janeiro.

CHEN, K., GORLA, N. (1998). Information system project selection using fuzzy logic. *IEEE Transactions on Systems, Man and Cybernetics – Part A*, vol. 28, no. 6, pp. 849-855.

CHEN, P.P.S. (1976). The entity-relationship model – Toward a unified view of data. *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9-36.

CHEN, Y.L., HUANG, T.C.K. (2005). Discovering fuzzy time-interval sequential patterns in sequence databases. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, vol. 35, no. 5, pp. 959-972.

CODD, E.F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, vol. 13, no. 6, pp. 377-387.

CONNOLY, T., BEGG, C. (2005). *Database Systems*. Addison Wesley.

DATE, C.J. (1984). *Introdução a Sistemas de Bancos de Dados*. Editora Campus.

ELMASRI, R., NAVATHE, S.B. (2005). *Sistemas de Banco de Dados*. Pearson Addison Wesley.

FURTADO, A.L., SANTOS, C.S. (1979). *Organização de Banco de Dados*. Editora Campus.

IBGE (2000a). ftp://ftp.ibge.gov.br/Censos/Censo_Demografico_2000/Indicadores_Sociais/UFs, Acessado em 15 de Novembro de 2008.

IBGE (2000b). ftp://ftp.ibge.gov.br/Economia_Cadastro_de_Empresas/2000/, Acessado em 20 de Novembro de 2008.

IBGE (2002). ftp://ftp.ibge.gov.br/Pib_Municipios/2002/, Acessado em 20 de Novembro de 2008.

ISO/IEC 9075 (1999). *Information Technology – Database Languages – SQL.*

KIFER, M., BERNSTEIN, A. LEWIS, P.M. (2006). *Database systems: an application-oriented approach.* Pearson Education.

KING, P.J., MAMDANI, E.H. (1977). The application of fuzzy control systems to industrial process. *Automatica*, vol. 13, pp. 235-242.

KOYUNCU, M., YAZICI, A. (2003). IFOOD: An intelligent fuzzy object-oriented database architecture. *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 5, pp. 1137-1154.

LI, D., LIU, D. (1990). *A Fuzzy PROLOG Database System*. John Wiley & Sons INC.

MA, Z.M., YAN, L. (2007). Generalization of strategies for fuzzy query translation in classical relational databases. *Information and Software Technology*, vol. 49, no. 2, pp. 172-180.

MAJUMDAR, A.K., BHATTACHARYA, I., SAHA, A.K. (2002). An object-oriented fuzzy data model for similarity detection in image databases. *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 5, pp. 1186-1189.

MAMDANI, E.H. (1977). Application of fuzzy logic to approximate reasoning using linguistic synthesis. *IEEE Transactions on Computers*, vol. 26, no. 12, pp. 1182-1191.

PEDRYCZ, W. (1993). *Fuzzy Control and Fuzzy Systems*. John Wiley & Sons Inc.

PEDRYCZ, W., GOMIDE, F.A.C. (1998). *An Introduction to Fuzzy Sets – Analysis and Design*. The MIT Press.

PERES, S.M., BOSCARIOLI, C. (2002). Sistemas gerenciadores de banco de dados relacionais Fuzzy: uma aplicação em recuperação de informação. *Acta Scientiarum*, vol. 24, no. 6, pp. 1733-1743.

RASHIDOV, A. (2004). Intelligent methods for data-processing and search in information systems. *Proceedings of the 2nd IEEE International Conference on Intelligent Systems*, vol. 3, pp. 129-134.

SATUR, R., LIU, Z.Q. (1999). A contextual fuzzy cognitive map framework for geographic information systems. *IEEE Transactions of Fuzzy Systems*, vol. 7, no. 5, pp. 481-494.

SAYGIN, Y., ULUSOY, O. (2001). Automated construction of fuzzy event sets and its application to active databases. *IEEE Transactions of Fuzzy Systems*, vol. 9, no. 3, pp. 450-460.

SILBERSCHATZ, A., KORTH, H.F., SUDARSHAN, S. (2006). Sistemas de Bancos de Dados. Elsevier.

SIMÕES, M.G., SHAW, I.S. (2007). *Controle e Modelagem Fuzzy*. Edgar Blucher.

SQL Server (2005). Books Online. http://www.microsoft.com/, acessado em 31 de Julho de 2008.

STAIR, R.M. (1998). *Princípios de Sistemas de Informação – Uma Abordagem Gerencial.* LTC (2ª Edição).

STONEBRAKER, M., WONG, E., KREPS, P. (1976). The design and implementation of INGRES. *ACM Transactions on Database Systems*, vol. 1, no. 3, pp. 189-222.

VERYHA, Y. (2005). Implementation of fuzzy classification in relational databases using conventional SQL querying. *Information and Software Technology*, vol. 47, no. 5, pp. 357-364.

YAZICI, A., INCE, C., KOYUNCU, M. (2008). FOOD Index: A multidimensional index structure for similarity-based fuzzy object oriented database models. *IEEE Transactions of Fuzzy Systems*, vol. 16, no. 4, pp. 942-957.

ZADEH, L. (1965). Fuzzy sets. *Information and Control*, vol. 8, no. 3, pp. 338-353.

ZIMMERMANN, H.J. (1996). *Fuzzy Set Theory and Its Applications*. Kluwer Academic Publishers.