# Incerteza em Bancos de Dados: Tipo de Dados Nebulosos no GOA++

José Roberto de Freitas Boullosa
Flavia Cardoso de Almeida Cruz
Geraldo Xexéo
Programa de Engenharia de Sistemas e Computação - COPPE / UFRJ
{beto, fcruz, xexeo}@cos.ufrj.br

**Abstract**

*Uncertainty is certainly a main issue in database management. As information flows from real world into computer systems, points of doubt inevitably emerge. They can vary from possible mistakes in the process of collecting data and inserting / updating it in database, to vagueness in data itself, or uncertainty about what the user expects from a specific query. Many alternatives have been proposed to handle such uncertainties, from null values to more subtle concepts as "soundness", for example. Fuzzy logic is one of these alternatives. In this work, we show how fuzzy concepts can be used to handle some of these issues in an OODBMS, by implementing a fuzzy datatype that can be used to build queries.*

## 1. Introdução

Modelos de dados procuram representar a informação de parte do mundo real de uma maneira bem definida, a fim de disponibilizá-la para ser manipulada dentro da estrutura formal de um Sistema Gerenciador de Banco de Dados. Um modelo de dados pode ser definido como uma coleção de tipos de objetos, uma coleção de operadores e uma coleção de regras de integridade gerais [DATE95].

Visto de outra forma [MOTR95], um sistema de banco de dados modela uma parte do mundo real, incluindo um componente declarativo de descrição desse mundo, e um componente operacional que manipule (modifique ou transforme) esta descrição. O sistema é responsável por manter esta descrição confiável (íntegra) e por processar as modificações e transformações de maneira correta e eficiente.

Entretanto, ao se construir tal representação, deve-se levar em conta que o conhecimento que se tem do mundo real é permeado pela incerteza, expressa de diversas formas e em variados graus. O modelo de dados deve, portanto, refletir e lidar com esses diversos tipos de incerteza e, da melhor maneira possível, procurar reduzi-la.

Num sistema de banco de dados, a incerteza surge nos diversos níveis da representação. Assim, no nível da descrição, surge a questão de se os dados representados são ou não corretos, ou se a forma de representá-los é ou não adequada. No nível das transformações, pode-se questionar se as operações realizadas pelos usuários são mesmo as indicadas para se chegar ao resultado desejado, ou se elas levam ao resultado correto. Finalmente, no nível do processamento das transformações, a incerteza pode surgir como conseqüência do próprio funcionamento do sistema.

O tratamento da incerteza é tão importante que chega a ser um dos requisitos para um sistema ser definido como verdadeiramente relacional de acordo com as famosas "12 regras" de E.F.Codd [MCFA91]. Com efeito, a terceira regra estabelece que "valores nulos (distintos da cadeia vazia de caracteres, da cadeia de caracteres em branco, de zero e de qualquer outro

número) devem ser suportados pelo sistema para representar informações indisponíveis ou inaplicáveis, de maneira sistemática, independente do tipo de dados". Além disso, as duas regras de integridade do modelo relacional - integridade de entidade e integridade referencial - abordam o tema dos valores nulos.

A abordagem mais comumente desenvolvida e trabalhada ao se lidar com a incerteza em SGBDs tem sido o uso de valores nulos para representar tudo aquilo que for desconhecido, em termos de valores de atributo de uma tupla ou objeto, por exemplo. Especial cuidado, porém, deve ser tomado para se distinguir entre o uso de valores nulos como representantes de incerteza ou como um sinal da inaplicabilidade de um atributo (ou propriedade) a determinada tupla (ou objeto) do SGBD [DATE88].

Além dos valores nulos, outras técnicas têm sido utilizadas, tais como o uso de lógica nebulosa, probabilidade e possibilidade, fatores de certeza, ou noções como "soundness" e "completeness" [MOTR95]. O suporte à lógica nebulosa ("fuzzy" ou difusa) em SGBDs é outra das maneiras encontradas para lidar com a incerteza, especialmente para o caso de dados vagos ou imprecisos. Seu uso já tem sido algo difundido em pesquisas em SGBD's relacionais, e vem aos poucos também se incorporando à área de SGBDOO's.

A partir destas considerações, buscamos então abordar o tratamento de incerteza através da implementação de suporte nebuloso em um SGBDOO. Assim, implementamos um novo tipo de dado para tratamento de informações difusas no contexto do atributo. Um atributo classificado como difuso (fuzzy) terá associados a ele um valor base, do tipo float, e um ou mais termos e modificadores linguísticos que podem ser consultados pelo usuário. Este tipo de dado pode servir ainda como suporte para o tratamento de incerteza também no nível do objeto, na forma de uma função de pertinência à extensão. A implementação foi realizada no GOA++ (Gerente de Objetos Armazenados), gerenciador de objetos persistentes desenvolvido na UFRJ/COPPE [MAUR97][MAUR98].

O artigo está organizado como se segue: a seção 2 traz uma discussão sobre o tratamento de dados incertos em SGBDs, mostrando as suas dificuldades e algumas das principais soluções apresentadas; a seção 3 torna esta discussão mais específica, enfocando as abordagens orientadas ao uso de funções de distribuição de possibilidade/probabilidade e lógica nebulosa; a seção seguinte apresenta algumas implementações dessas abordagens; na seção 5, mostramos como foi implementado o tipo de dado nebuloso no GOA++; a seção 6 apresenta as conclusões do trabalho; finalmente, a seção 7 traz as referências bibliográficas.

## 2. Tratamento da incerteza em SGBDs

De forma geral, qualquer elemento de um modelo de dados que não tenha o seu valor estabelecido com total confiança terá um certo grau de incerteza. A incerteza propriamente dita é aquela em que não se pode distinguir se um determinado fato no banco de dados é ou não verdadeiro. Por exemplo, pode-se não ter certeza do fato "a aluna Maria nasceu em 01/01/1969".

A informação pode também sofrer de imprecisão, ou seja, pode faltar-lhe especificidade. Este fato pode ocorrer tanto através de predicados que envolvam intervalos ("a aluna Maria nasceu entre 01/01/1969 e 30/06/1969"), disjunções ("a aluna Maria nasceu em 01/01/1969 ou em 30/06/1969") ou negações ("a aluna Maria não nasceu em "10/03/1969"), como simplesmente através do desconhecimento ou indisponibilidade da informação - área onde se encaixa um dos usos dos valores nulos.

A informação do banco de dados pode ainda ser vaga. Este é um dos pontos mais convenientes para uma abordagem segundo a teoria dos conjuntos nebulosos ou segundo a

teoria das probabilidades. Assim, podem existir asserções como as seguintes: “a aluna Maria nasceu em meados de agosto de 1969” ou “a aluna Maria nasceu pouco depois da Páscoa de 1969”.

Podem ser encontradas, num sistema de banco de dados, informações inconsistentes entre si, ou seja, representadas em mais de um ponto do modelo, porém com valores diversos e incongruentes. Em um ponto do modelo, pode-se ter a informação de que “a aluna Maria nasceu em 18/05/1969”, e, em outro ponto, “a aluna Maria nasceu em 17/06/1969”.

A inconsistência é uma forma particular de falta de integridade. Ela pode surgir quando existe redundância na representação dos dados, sendo, portanto, um dos objetivos de um sistema de banco de dados, reduzir a redundância, a fim de reduzir a inconsistência, e consequentemente, aumentar a integridade, diminuindo também a incerteza.

Finalmente, tem-se a incerteza como conseqüência de uma descrição ambígua no modelo. Pode-se questionar, por exemplo, se um atributo “juros devidos” armazena o percentual de juros mensais ou anuais de um contrato.

Pode-se notar que os diversos tipos de incerteza podem ser agrupados da seguinte forma: num primeiro grupo, têm-se a incerteza propriamente dita, os dados imprecisos, desconhecidos e os expressos de forma vaga. Nesses casos, o problema está na captura dos dados do mundo real ou na disponibilidade desses dados. Num segundo grupo tem-se a incerteza causada pela inconsistência, no caso de dados redundantes. Aqui, o sistema é o responsável por ter permitido a entrada de dados inconsistentes entre si. Num terceiro grupo, fica a incerteza causada pelas ambigüidades do modelo.

A incerteza do terceiro grupo (dados expressos de forma vaga) deve ser tratada através de uma boa definição do modelo de dados, ou seja, é uma questão de se usarem corretamente as técnicas de modelagem de dados, normalização, etc., a fim de se ter um modelo enxuto, correto e não ambíguo.

A incerteza do segundo grupo (inconsistência) está no escopo de um problema maior: a integridade. É, desta forma, uma questão a ser tratada pelo subsistema de integridade do SGBD. Especial atenção deve ser tomada no caso dos sistemas distribuídos, onde a redundância normalmente não pode ser evitada.

Mas é no primeiro grupo (incerteza propriamente dita) que está a incerteza mais complexa de ser tratada, e é esta a que interessa quando se fala em tratamento de incerteza em bancos de dados, pois aí está o desafio duplo que se apresenta: como representar e como manipular esta incerteza, já que ela não pode ser completamente eliminada.

Num banco de dados relacional, a incerteza pode ocorrer tanto no nível dos atributos, quanto das tuplas ou das relações. Da mesma maneira, num banco de dados OO, a incerteza pode aparecer nas propriedades, nos objetos ou nas coleções de objetos, e assim por diante. A incerteza nos atributos ou propriedades é, de fato, incerteza no nível dos valores dos dados. Numa relação ALUNOS (nome, nascimento), pode-se não ter certeza sobre as datas de nascimento de alguns dos alunos.

Num sistema relacional, a incerteza no nível da tupla acontece quando não se sabe ao certo se determinada tupla faz parte ou não da relação. Por exemplo, na relação ALUNO_TURMA (aluno, turma), pode-se não ter certeza de que a tupla (aluno:”Maria”, turma:”T12”) deveria ou não estar na relação. Exemplos similares podem ser observados num banco OO, ao se avaliar o grau de pertinência de um objeto a uma coleção. Da mesma forma, encontram-se exemplos em sistemas de recuperação de informação, em sistemas especialistas, etc.

A incerteza pode estar presente em maior ou menor grau, sempre que a informação correta não está presente em qualquer elemento do modelo, em qualquer nível (por exemplo,

nas tuplas, ou nos atributos, ou nas coleções). Os maiores graus de incerteza são aqueles em que não se está certo nem mesmo da existência de um determinado objeto do mundo real. Caso a existência do elemento seja garantida, podem-se ter, contudo, algumas informações acerca dele desconhecidas, indisponíveis ou incompletas. Nestes casos, pode-se reduzir a incerteza quando estas informações vêm de um certo intervalo ou conjunto de valores. Tem-se aí uma informação do tipo disjuntiva. Caso este conjunto contenha todos os valores do domínio, a informação volta a ser desconhecida.

Quando se associam probabilidades aos valores alternativos para a descrição de um elemento, tem-se uma redução ainda maior na incerteza, sendo este tipo de informação probabilística. Neste caso, a soma das probabilidades de todas as alternativas deve ser 1. Caso as probabilidades não estejam disponíveis, a informação passa a ser disjuntiva.

O uso de fatores de certeza é uma outra abordagem para o tratamento de incerteza. Em Sistemas de Recuperação de Informações e em Sistemas Especialistas, tem sido aplicada esta técnica de se associar um grau de confiança à informação, para que ela tenha uma maior aproximação com o valor correto. Certos sistemas de recuperação de informação utilizam os conceitos de distância, métrica ou proximidade para lidar com a incerteza, especialmente os sistemas baseados em modelos vetoriais. Esta idéia de distância ou métrica acompanha o conceito de vizinhança de uma consulta ou de um resultado, significando uma aproximação maior de uma descrição em relação a outra.

Ainda em sistemas de recuperação de informações, há dois termos importantes que têm uma certa ligação com a incerteza: “precision” – precisão e “recall” [SALT83]. Consultas com grande precisão são aquelas que retornam um mínimo de informações irrelevantes. Consultas com grande recall são aquelas que retornam o máximo possível de informações relevantes. Esses conceitos são similares aos de “soundness” & “completeness”, definidos por Motro [MOTR95], para bancos de dados relacionais. Uma consulta é sólida (“sound”) se incluir somente informações que certamente ocorrem no mundo real. Consultas “completas” são as que incluem todas as informações verdadeiras que ocorrem no mundo real. São definidas então visões sobre as partes do banco de dados que estejam íntegras, a partir das quais novas visões possam ser inferidas para responder a consultas.

Ainda que o principal foco das pesquisas sobre a incerteza esteja na descrição do mundo real, deve-se atentar também para a incerteza nas transformações e modificações da descrição, bem como no processamento das operações e transações por parte do sistema de banco de dados. No caso das consultas, deve-se levar em conta o nível de conhecimento que os usuários têm do sistema, da descrição, da linguagem de manipulação e das ferramentas disponíveis. Em todos estes níveis, pode estar presente a incerteza.

Nem sempre se pode assumir que os usuários estejam certos quanto às consultas que estão fazendo. Isto dependerá muitas vezes do próprio nível de conhecimento das estruturas sintáticas da linguagem, bem como da semântica da descrição. Outras vezes, porém, mesmo que o usuário seja profundamente conhecedor da descrição e do modelo, ele pode simplesmente ter uma idéia vaga daquilo que está procurando. Interfaces tolerantes a erro, uso de navegadores, ferramentas interativas com realimentação através de diálogos, processadores de consultas vagas, uso de formalismos relaxados para interpretar as consultas são algumas das opções para tratar a incerteza nesses casos.

Em relação às operações de modificações, a incerteza pode surgir como conseqüência do conhecimento insuficiente do sistema, ou como produto de modificações vagas ou imprecisas. No primeiro caso, é necessário que, da mesma forma que nas consultas, as ferramentas de atualização sejam o mais simples possível para que os usuários se sintam mais seguros ao especificar as modificações. No segundo caso, a depender da abordagem tomada

para o tratamento da incerteza na descrição, o sistema deve procurar resolver as atualizações vagas e imprecisas de modo a acomodar os valores à estrutura usada para reduzir a incerteza na descrição.

## 3. Tratamento da incerteza através de possibilidades/probabilidades

Uma alternativa ao tratamento da incerteza no nível da descrição do SGBD que tem sido muito estudada é o uso de probabilidades ou de possibilidades. No primeiro caso, tem-se o uso dos modelos tradicionais de distribuição probabilística, e, no segundo, a aplicação da lógica difusa ou nebulosa ("fuzzy logic").

Os modelos de probabilidade utilizam-se de funções de distribuição de probabilidade [BARB92] de uma variável X sobre um domínio D, que atribui a cada valor $d \in D$ um valor entre 0 e 1, indicando a probabilidade de X ser igual a d. Nestes modelos, a soma das probabilidades deve ser igual a 1 (ao contrário dos modelos baseados em funções de distribuição de possibilidade).

As distribuições de probabilidade podem então ser associadas a domínios de um sistema relacional ou de um sistema OO, por exemplo, sendo estendidos os operadores do sistema para que manipulem tais variáveis probabilísticas. Desta forma, o usuário poderá determinar, em suas consultas, quais são os critérios que deseja para o seu conjunto resultado.

Cheeseman [CHEE93] apresentou um modelo probabilístico chamado AutoClass II, que induz classes em um banco de dados determinando automaticamente suas descrições probabilísticas e a probabilidade de cada objeto ser membro de uma classe. Para isso, o sistema utiliza-se de técnicas estatísticas Bayesianas, assumindo que as distribuições de prioridades associadas a priori aos atributos de uma classe são distribuições normais, podendo depois ser ajustadas pelo usuário.

Os modelos baseados em possibilidade utilizam-se da teoria dos conjuntos difusos ou nebulosos ("fuzzy sets"), aqueles onde cada elemento tem associado um valor no intervalo de 0 a 1, indicando o seu peso ou grau de pertinência ao conjunto. Assim, um conjunto fuzzy de alunos poderia ter os elementos "Maria", "Roberto" e "Flávia", com graus de pertinência 0.3, 1.0 e 0.75, respectivamente.

O uso de conjuntos difusos presta-se a dois propósitos diferentes, ainda que complementares: representar conceitos mal definidos em termos de relações difusas, onde as tuplas têm um peso que indique o grau de pertinência à relação, e representar informações incompletas dentro das próprias tuplas. Neste último caso, os domínios são conjuntos difusos e, assim, os valores dos atributos podem ser simples (sem incerteza), nulos, difusos ou ainda conjuntos ou intervalos.

O uso de lógica difusa implica na ampliação e modificação dos operadores da álgebra relacional (ou da álgebra de objetos, no caso de sistemas OO), para que reconheçam e manipulem corretamente os valores difusos, tanto no nível das relações (ou objetos), quanto no dos atributos. Assim, os operadores difusos permitirão a expressão de consultas difusas ou incertas.

A lógica nebulosa abrange muitos tópicos e tem sido aplicada a diferente áreas de conhecimento. Neste trabalho, especificamente, estamos interessados na noção de conjuntos nebulosos ou fuzzy, associado às idéias de termos, variáveis e modificadores linguísticos.

Uma variável linguística é um rótulo definido em termos de uma variável base, cujo estado é expresso por termos linguísticos, interpretados como números fuzzy específicos. Os números fuzzy são definidos dentro de um intervalo float de 0 a 1, expressando a possibilidade, não a probabilidade, de que determinado conceito seja verdadeiro.

Num SGBD, isso pode significar a possibilidade de que uma instância (ou tupla) pertença a uma extensão (ou relação). Por exemplo, uma instância de um país pertencer à extensão dos países. Mas, pode ainda expressar o quanto um país é, por exemplo, super-populoso, com base no número de habitantes que possui.

No primeiro caso, temos a lógica fuzzy expressa no nível da instância de um objeto (ou tupla). No segundo caso, ela está expressa no nível do propriedade ou atributo. O termo linguístico, nesse caso, seria o adjetivo "superpopuloso". Para associar valores de população ao número fuzzy que denota esse adjetivo, é necessário uma função de pertinência, outro elemento fundamental do conjunto fuzzy.

Assim, identificamos num conjunto fuzzy os termos e variáveis linguísticas associados a uma variável base, unidos por uma função de pertinência. Além disso, modificadores linguísticos, tais como "pouco", "muito", "razoavelmente", podem alterar o valor fuzzy de um termo linguístico, através de funções pre-definidas. Por exemplo, o termo "muito" é normalmente expresso elevando-se ao quadrado o valor fuzzy.

## 4. Algumas abordagens nebulosas em bancos de dados

Uma abordagem com o uso de sistema nebulosos em bancos de dados relacionais está presente na família de sistemas FQUERY [KACP89]. Desenvolvidos inicialmente para bancos de dados padrão X-base, o FQUERY suporta consultas do tipo "encontre (todos) os registros tais que *a maioria* (quase todos, mais da metade, ou outro quantificador) dos atributos *importantes* estejam como especificados (por exemplo, igual a 10, maior que 100, muito menor que 1000, etc.).

Kcprzyk e Zadrozny [KACP94], com base no FQUERY, implementaram facilidades nebulosas no Access v.2 para Windows, colocando os elementos do FQUERY como um módulo embutido no sistema, utilizando-se, para tanto, da possibilidade de construir e agregar uma biblioteca de elementos nebulosos em um arquivo de banco de dados específico, a ser instalado e carregado pelo usuário durante cada sessão.

Para as consultas, os elementos nebulosos são inseridos através de uma interface estilo QBE (Query-by-example), acessível através de uma barra de ferramentas específica do FQUERY. A composição dos elementos nebulosos nas consultas do Access é conseguida através do uso de uma convenção para nomes de parâmetros, aproveitando-se da possibilidade oferecida pelo Access de se colocarem parâmetros nas consultas.

A SQLf [BOSC95] é uma extensão à SQL que apresenta suporte a valores nebulosos, suportando uma grande variedade de consultas nebulosas, introduzindo predicados nebulosos sempre que possível, mas, ainda assim, mantendo-se fiel ao espírito original da SQL.

Seu princípio básico é a introdução de nebulosidade em dois níveis principais da cláusula WHERE do SELECT: nos predicados em si e na maneira como estes são conectados. A cláusula é alterada adicionando-se um mecanismo de ajuste do resultado, pelo qual se especifica o número N de respostas desejadas e/ou um limite ($T \in [0,1]$) a ser aplicado à relação resultante da seleção nebulosa.

O bloco SELECT resultante fica expresso da seguinte maneira:

```
SELECT    (N | T | N,T) (lista de atributos)
FROM      (lista de relações)
WHERE     (condição nebulosa)
```

Sua interpretação é como se segue: aplicar o predicado (condição nebulosa) ao produto cartesiano das relações envolvidas e projetar a relação nebulosa resultante sobre o conjunto de atributos, retornando as melhores N tuplas e/ou aquelas sobre o limite T.

O uso da lógica nebulosa adequa-se muito bem à incerteza nas transformações, ou seja, nas consultas realizadas pelo usuário. Diversos modelos de álgebras difusa têm sido propostos com vistas a facilitar para o usuário a definição e extração daquilo que ele realmente pretende obter do banco de dados. Berthier [BERT95] propôs uma álgebra relacional fuzzy estendida chamada F-G para classificar respostas aproximadas em relação a consultas com critérios múltiplos.

Um exemplo mais recente de tratamento de valores nebulosos em SGBDS é Braga, [BRAG98], que desenvolveu uma biblioteca OO para manipulação de sistemas nebulosos relacionais, formada por classes C++ que encapsulam os dados e propriedades dos conjuntos nebulosos. Através dessas classes, o usuário define, de maneira simples e transparente, um modelo nebuloso de um sistema, e pode, a partir daí, realizar consultas recuperações e avaliações destes dados nebulosos.

## 5. Implementação de suporte fuzzy no GOA++

O GOA++ [MAUR97] [MAUR98] é um gerenciador de objetos armazenáveis compatível com o modelo ODMG 2.0, com APIs disponíveis para serem utilizadas com C++, O2 e Java, e tendo como linguagens de definição e consulta a ODL e a OQL, respectivamente. Apesar de não oferecer suporte para o tratamento de dados incertos, mesmo que através de valores nulos, o GOA++ possui uma estrutura interna adaptável e flexível, que permite a adição de novos tipos de dados com relativa facilidade.

### 5.1 Tipo de dados fuzzy

Tendo em vista as diversas formas possíveis para o tratamento de incerteza a partir do suporte nebuloso, optamos por uma implementação que permitisse a definição de um atributo ou propriedade nebulosa na classe, estendendo, portanto, o conjunto dos tipos de dados do GOA++. Com isso, proporcionamos ao usuário do sistema a possibilidade de ter um atributo com valor base numérico, mas com um valor nebuloso correspondente.

Assim, tornam-se viáveis consultas sobre os atributos do tipo nebuloso, evocando tanto o seu valor base, quanto o valor "fuzzyficado" correspondente. Tal processo de "fuzzyficação" é feito a partir de uma função de pertinência obtida pela extrapolação de valores de uma tabela de correlação valor-base x valor fuzzy, tabela esta fornecida pelo usuário quando da definição do tipo fuzzy.

O tipo de dados fuzzy por nós implementado no GOA++ compreende pois a definição de uma variável linguística ou rótulo definido sobre uma variável base numérica (de tipo ponto flutuante), e o fornecimento das tabelas de correlação entre os valores base e fuzzy em relação a um ou mais termos linguísticos. A variável base é armazenada internamente como um valor do tipo float. Além disso, o sistema fornece alguns modificadores linguísticos pré-definidos e a possibilidade de implementação de novos modificadores. Por exemplo, o sistema permite a definição de uma variável fuzzy IDADE, com termos linguísticos VELHO e JOVEM, definido por tabelas de correlação, tais como nas tabelas 1 e 2. A definição dos valores dessas tabelas depende do projetista da aplicação, que normalmente se vale do conhecimento de especialistas para definí-las.

**Tabela 1: Valores fuzzy para o termo linguístico VELHO**

| **Base (idade)** | 10 | 20 | 30 | 40 | 50 | 60 | 65 |
|---|---|---|---|---|---|---|---|
| **Fuzzy** | 0,1 | 0,2 | 0,4 | 0,5 | 0,8 | 0,9 | 1,0 |

**Tabela 2: Valores fuzzy para o termo linguístico JOVEM**

| **Base (idade)** | 10 | 20 | 30 | 40 | 50 | 60 | 65 |
|---|---|---|---|---|---|---|---|
| **Fuzzy** | 1,0 | 0,7 | 0,4 | 0,2 | 0,1 | 0,0 | 0,0 |

O sistema se vale de uma função de extrapolação para calcular os demais valores fuzzy para os casos não incluídos nas tabelas. Com isso, são possíveis consultas aos valores base ou aos valores "fuzzyficados" quando necessário. Além disso, modificadores linguísticos como os implementados no sistema (p.ex., MUITO) podem ainda causar transformação nestes valores. Por exemplo, uma pessoa com idade de 50 anos seria "0,8 velho", porém "0,64 muito velho", já que o modificador MUITO foi implementado elevando-se ao quadrado o valor fuzzy.

Para a implementação, foram criadas internamente duas novas classes GOA: GoaFuzzy e GoaTermoLinguistico. A primeira armazena a variável fuzzy propriamente dita e possui os métodos e atributos necessários para ler, alterar e armazenar um valor num atributo fuzzy, além dos métodos específicos para manipulação e visualização do esquema. A segunda representa um termo linguístico, seja ele externo, definido pelo usuário. ou interno, definido no próprio GOA++.

**Definição das classes GoaFuzzy e GoaTermoLinguistico**

```
Class GoaFuzzy: public GoaType, public SchemaElement{
  public:
    static GoaFuzzy *getGoaFuzzy( istream &in );
    GoaFuzzy( istream &in );
    virtual ~GoaFuzzy();
    virtual int isGoaFuzzy();
    virtual void printODL( ostream &out );
    virtual void printODLFull( ostream &out );
private:
    MemberList  *atr_TermosLinguisticos;
    GoaType       *atr_VariavelBase;}

class GoaTermoLinguistico: public GoaType
{
  public:
    static GoaTermoLinguistico *getGoaTermoLinguistico( istream &in );
    GoaTermoLinguistico( istream &in );
    virtual ~GoaTermoLinguistico();
    virtual void printODL( ostream &out );
    virtual void printODLFull( ostream &out );
    void        ModificadorFuzzy( float ValorFuzzy, char *NomeModificador, FieldValue
*fv );
  private:
    Vector  *atr_ValoresBase,
              *atr_ValoresFuzzy;}
```

### 5.2 Modificadores padronizados

A implementação dos modificadores padronizados no sistema foi feita através de um método da classe genérica GoaTermoLinguístico. Neste métodos estão encapsulados todos os modificadores padrões do GOA++, e é nele onde devem ser inseridos os novos modificadores não sendo ainda possível a sua definição automática e on-line. Foram implementados, a título de experimento, dois modificadores linguísticos padronizados: "muito" e "razoavelmente". O primeiro foi calculado elevando-se o valor fuzzy ao quadrado e o segundo, tirando-se a raiz quadrada do valor.

**Método ModificadorFuzzy, que implementa os modificadores padrões**

```
void GoaTermoLinguistico::ModificadorFuzzy( float ValorFuzzy, char
*NomeModificador, FieldValue *fv )
{
if (!strcmp (NomeModificador, "muito"))
  {
    fv->setFloat (ValorFuzzy * ValorFuzzy);
    return;
  }
if (!strcmp (NomeModificador, "razoavelmente"))

{
     fv->setFloat (sqrt (ValorFuzzy) );
     return;
  }
}
```

### 5.3 Sintaxe proposta

A sintaxe para criação de uma variável fuzzy com seus termos linguísticos associados foi definida da seguinte forma na linguagem de definição GOA++:

**Z {NomeDaVariavelFuzzy} AG TermoLinguistico [AG TermoLinguistico …] .**

Onde **Z** indica tratar-se de uma variável fuzzy e **AG** indica o início da definição de um termo linguístico. Cada termo linguístico é definido da maneira seguinte:

**TermoLinguistico := TabelaPertinencia {NomeTermoLinguistico}**
**TabelaPertinencia := ValorBase ValorFuzzy [ValorBase ValorFuzzy …] G**

O **G** assinala o final da tabela de pertinência para este termo linguístico. Como um exemplo, para a variável fuzzy IDADE apresentada anteriormente (seção 5.1), com apenas o termo linguístico VELHO, teríamos:

**Z fzIdade AG 10 0.1 20 0.2 30 0.4 40 0.5 50 0.8 60 0.9 65 1.0 G Velho .**

Propusemos também uma sintaxe ODL no seguinte formato exemplificado abaixo:

```
fuzzytype fzIdade{
attribute langterm ([10,0.1]  [20,0.2]  [30,0.4]  [40,0.5] [50,0.8]  [60,0.9] [65 1.0])
Velho;}
```

Para se utilizar a variável fuzzy numa classe, deve-se utilizar o tipo Z na sintaxe de definição do GOA++. O tipo Z foi adicionado aos já existentes: U (varchar), L (long), F (float), etc.

Como exemplo temo a seguir a criação de uma classe ALUNO com campos NOME e ENDERECO, strings variáveis, e com um campo fuzzy IDADE , do tipo fuzzy FZIDADE definido anteriormente (os dois asteriscos indicam a herança e a extensão da classe, que neste caso, não foram definidos):

```
C Aluno * * AU Nome AU Endereco AZ fzIdade Idade.
```

Uma observação importante é que a criação de tipos fuzzy bem como de classes que contenham atributos desses tipos implicam em alteração do esquema da base de dados, o que pode ser feito dinamicamente, ao contrário da definição dos modificadores padronizados.

Para referenciar as variáveis fuzzy, tanto para consulta quanto para atualização, pode-se fazê-lo acessando-se o valor base, os valores fuzzy dos termos linguísticos, ou ainda os modificadores fuzzy padronizados no sistema., seja através das funções da API básica ou através da OQL.

Por exemplo, se acessarmos **ALUNO.IDADE**, estaremos nos referindo ao valor base (do tipo float), tanto para ler como para gravar. Para acessarmos o valor fuzzy do termo linguístico VELHO, usamos a notação **ALUNO.IDADE.VELHO.** Da mesma forma, se quisermos acessar este valor com o modificador MUITO, usamos **ALUNO.IDADE.VELHO.MUITO**. Assim, tem-se uma forma simples e flexível de acessar os valores fuzzy da maneira que for mais conveniente à aplicação.

Utilizando-se as funções da API do GOA++, poderíamos proceder às leituras dos valores base, fuzzy do termo linguístico e fuzzy modificado da seguinte maneira, respectivamente (a alteração dos valores seria de forma semelhante):

```
Aluno->readFloat ("Idade ");
Aluno->readFloat ("Idade.Velho");
Aluno->readFloat ("Idade.Velho.Muito");
```

Em OQL, as seguintes consultas seriam possíveis:

```
SELECT a FROM a IN Alunos WHERE a.Idade.Velho > 0.8;

SELECT a FROM a IN Alunos WHERE a.Idade.Velho.Muito < 0.2;
```

Observe-se que, como apresentado, o sistema dá suporte direto ao tipo de dados nebuloso, o que representa conceitualmente o tratamento do dado com incerteza no nível do atributo ou propriedade. Como dito, a nebulosidade no nível da instância (correspondente ao nível da tupla no modelo relacional) poderia ser implementado a partir daí, criando-se um

atributo especial para indicar o grau de pertinência do objeto à extensão, e todas as operações seriam similares.

### 5.4 Implementação de aplicação com dados fuzzy

Para testar a implementação do tipo fuzzy, modificamos uma aplicação usada em testes do GOA++, a fim de exemplificar propriamente a manipulação dos conceitos aqui explanados. A aplicação é um cadastro de países, continentes e cidades, cujo modelo de dados pode ser visto na figura 2, com o esquema de criação das classes descrito na tabela 4. Neste modelo, foi criado um tipo fuzzy "**fzPopulacao**", que representa a população de um país ou cidade. Associado a este tipo fuzzy, está um termo linguístico "**SuperPopuloso**", com sua correspondente tabela de valores, indicando o quão superpopuloso é o país / cidade (tabela 3).

**Tabela 3: Valores fuzzy para o termo linguístico "SuperPopuloso"**

| **População** | 1.000.000 | 10.000.000 | 50.000.000 | 100.000.000 | 500.000.000 |
|---|---|---|---|---|---|
| **SuperPopuloso** | 0,1 | 0,4 | 0,5 | 0,8 | 1,0 |

**Tabela 4: Esquema de criação da base de dados em ODL**

```
class Continente (  extent Continentes) {
  attribute string Nome;
  attribute long Area;
  relationship list<Pais> Paises
        inverse Pais::Continente;
  attribute long timestamp;}

class Pais (  extent Paises) {
  attribute string Nome;
  attribute fzPopulacao Populacao;
  attribute long Area;
  attribute string Lingua;
  relationship list<Cidade> Cidades
        inverse Cidade::Pais;
  relationship Cidade Capital
        inverse Cidade::Pais;
  relationship Continente Continente
        inverse  Continente::Paises;
  relationship list<Organizacao>
                  Organizacoes
        inverse Organizacao::Membros;}
```

```
class Cidade (  extent Cidades) {
  attribute string Nome;
  attribute fzPopulacao Populacao;
  relationship Pais Pais
        inverse Pais::Cidades;
  relationship list<Organizacao>
                  Organizacoes
        inverse Organizacao::Sede; }

class Organizacao (  extent Organizacoes)
{
  attribute string Nome;
  attribute string Descricao;
  relationship Cidade Sede
        inverse Cidade::Organizacoes;
  relationship list<Pais> Membros
        inverse Pais::Organizacoes; }
```

A definição da variável fuzzy "fzPopulacao" e do termo linguístico "SuperPopuloso" foi feita como abaixo, na linguagem de definição de esquema do GOA++:

| |
|---|
| **Z fzPopulacao AG 1000000 0.1 10000000 0.4 50000000 0.5 100000000 0.8 500000000 1.0 G SuperPopuloso .** |

A definição correspondente em ODL seria como se segue:

| |
|---|
| **fuzzytype fzPopulacao{**<br>**attribute langterm ([1e+06,0.1] [1e+07,0.4] [5e+07,0.5] [1e+08,0.8] [5e+08,1] )**<br>**SuperPopuloso;}** |

A figura 1 mostra uma das telas da aplicação, na qual estão ilustrados os valores base e os valores fuzzy correspondentes para as populações de diversos países. Além disso, para cada país, é apresentado também o valor fuzzy resultante da aplicação dos modificadores MUITO e RAZOAVELMENTE.

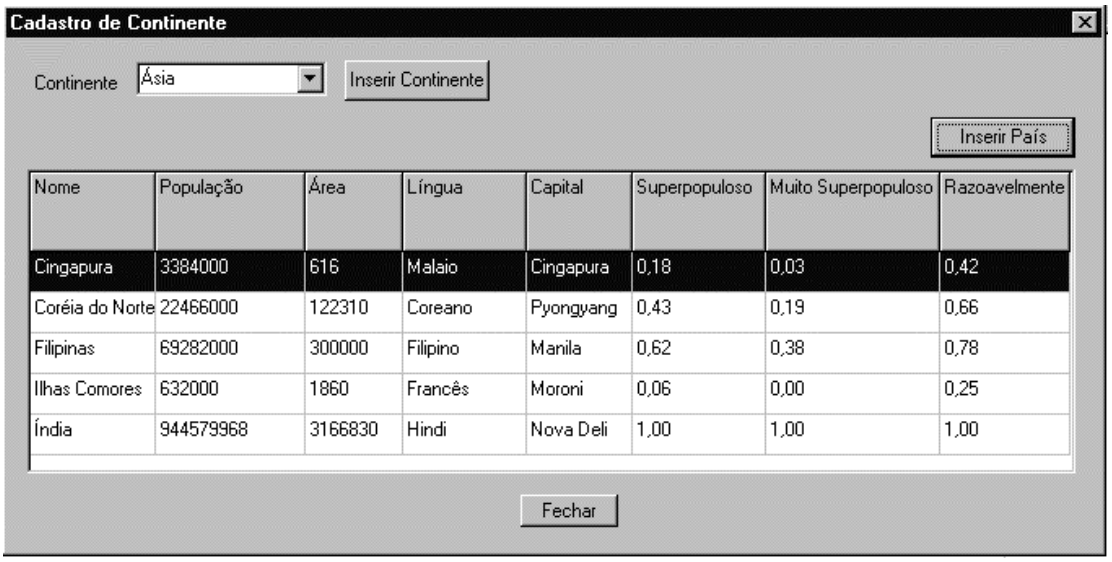


**Figura 1 : Cadastro de países mostrando valores fuzzy das populações**

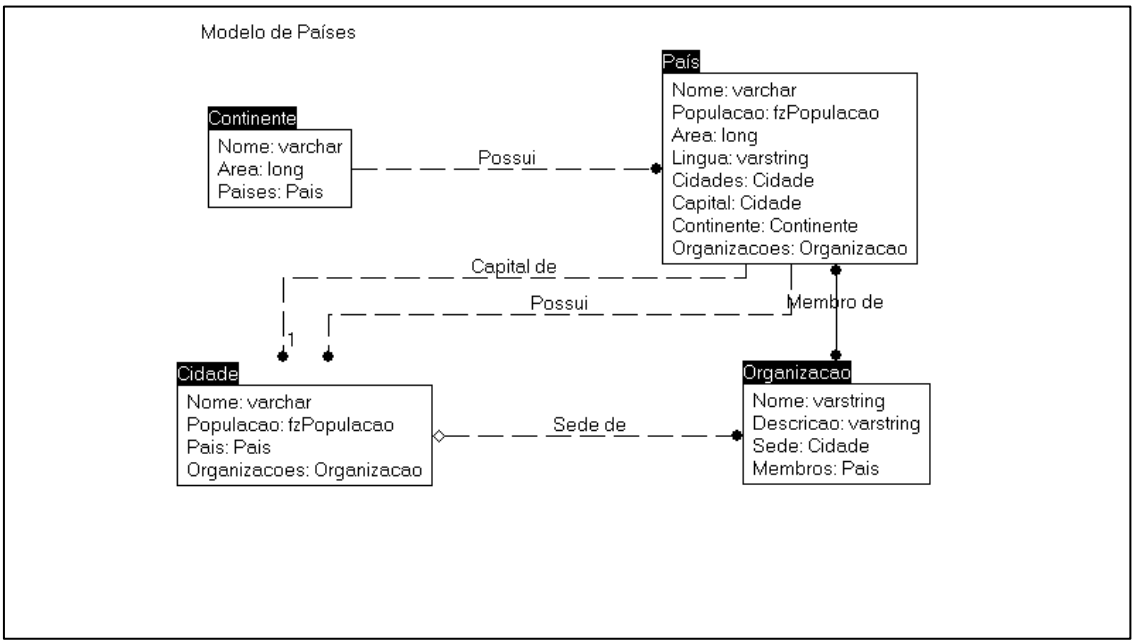


**Figura 2 : Modelo de dados da aplicação**

## 6. Conclusões

São múltiplas as abordagens para tratamento de valores imprecisos em bancos de dados, cada uma delas enfocando mais especificamente um aspecto da incerteza. Particularmente na área dos bancos de dados OO, o tópico não tem tido a mesma amplitude que em bancos de dados relacionais, até mesmo pela longo tempo de maturação que estes últimos já atingiram em relação aos primeiros. Desde as primeiras abordagens com valores nulos até as mais recentes implementações de bancos de dados nebulosos, evoluiu-se bastante na compreensão conceitual e na resolução dos problemas de implementação de valores incertos, mas vários pontos ainda estão em aberto.

De todas as abordagens revistas, apenas a de valores nulos já está razoavelmente bem definida e estabelecida, haja vista o seu relativamente longo tempo de estudos, testes e implementações. Na prática, os principais SGBDS relacionais dão um suporte completo e automático para valores nulos, como exigido pela terceira regra de Codd, assim como parte dos SGBDOO.

Um dos enfoques mais promissores é a utilização da abordagem nebulosa, utilizando-se conceitos emprestados à logica, a fim de se permitir, tanto no nível das consultas quanto no da representação dos dados, que sejam expressas as imprecisões inevitavelmente contidas em qualquer implementação de um modelo particular. Como as consultas nebulosas são mais complexas, o processamento e otimização delas torna-se mais trabalhoso, havendo duas possibilidades de desenvolvimento:

i) Uso de um SGBD convencional com uma camada adicional nebulosa que faça o papel de interface

ii) Construção de um novo sistema nebuloso completo que inclua no seu núcleo técnicas de processamento e otimização de consultas nebulosas.

Como visto, as implementações existentes, em sua maioria, inclusive a nossa, são do primeiro tipo. Nossa abordagem soma-se às tentativas de se implementar suporte a dados fuzzy em SGBDOOs, mostrando como isso pode ser feito tanto no contexto do atributo quanto da extensão, dando graus de pertinência aos objetos. Ela permite ao usuário a manipulação da incerteza através de nebulosidade no atributo, extensível ao objeto, possibilitando consultas e alterações sobre os dados fuzzy e definição dinâmica de novas variáveis e termos linguísticos.

Por outro lado, o suporte a dados nebulosos ainda exige uma série de estudos e evoluções. O tratamento de integridade, por exemplo, é um dos aspectos a serem mais estudados em sistemas nebulosos. A área de dependências funcionais e normalização em sistemas nebulosos começou a ser estudada recentemente [SAXE95], juntamente com as questões referentes ao processamento e implementação de consultas nebulosas, tendo muito pouco investido em técnicas de otimização.

Outra área de fundamental importância é quanto ao armazenamento interno dos dados nebulosos. Este problema, que já aparecia nas primeiras versões de sistemas relacionais que suportavam valores nulos, é ampliado com sistemas que proporcionem um grau mais alto de apoio à manipulação de incerteza. Em bancos de dados nebulosos, por exemplo, técnicas de indexação, construção de junções, uso de hash, etc. vêm sendo desenvolvidas a fim de proporcionar tempos de resposta aceitáveis [BOSC89].

Além disso, como uma das principais dificuldades, está o fato de que, quando um sistema lida com a incerteza, torna-se necessário que o usuário esteja atento e ciente para a existência da incerteza, saiba que existem operações que lidarão com esta incerteza, e, mais ainda, consiga interpretar o que significa a incerteza nos dados, na operação e no resultado da operação. Esta necessidade choca-se com a expectativa que o usuário normalmente possui de que o sistema fará suas consultas e retornará resultados sempre de maneira clara e não ambígua. Esta cultura precisa ser modificada, para que os sistemas que lidem com incerteza passem a ser parte integrante do dia-a-dia das corporações e empresas.

## 7. Referências


[BARB92] BARBARA, D. ET AL, 1992, “The Management of Probabilistic Data”, *IEEE Transactions on Knowledge and Data Engineering*, V. 4, n. 5, pp. 487-502.

[BELC97] BELCHIOR, A..D, XEXEO, G. B., ROCHA, A.R., “Enfoques Sobre a Teoria dos Conjuntos Fuzzy”. UFRJ – COPPE / Sistemas.

[BERT95] BERTHIER, A.N.R., 1995, "F-G: A Fuzzy Algebra for Approximate Answering in Databases", In: *Anais do X Simpósio Brasileiro de Banco de Dados,* pp. 365-376, Recife-PE.

[BOSC89] BOSC, P., GALIBOURG, M.., 1989, “Indexing Principles for a Fuzzy Database”, *Information Systems*, V. 14, , pp. 493-499.

[BOSC95] BOSC, P., PIVERT, O., 1995, “SQLf: A Relational Database Language for Fuzzy Querying”, *IEEE Transactions on Fuzzy Systems*, V. 3, , pp. 1-17.

[BRAG98] BRAGA, A.L., XEXEO, G.B, 1998, “Sistema de Manipulação Nebulosa de Banco de Dados”. In: *Anais do XIII Simpósio Brasileiro de Banco de Dados,* pp. 201-215, Maringá-PR.

[CHEE93] CHEESEMAN, P. ET AL, 1993, “AutoClass: A Bayesian Classification System”. In: *Readings in Knowledge Acquisition and Learning,* Buchanan, B. G., Wilkins, D. C. (eds), Morgan Kaufmann.

[DATE88] DATE, C.J., 1982, "Null Values in Database Management", In: *Proc. 2nd British National Conference on Databases (BNCOD-2)*, Briston, Inglaterra, Julho.

[DATE95] DATE, C.J., 1995, *"An Introduction to Database Systems"*, 6th ed., Reading, Massachusetts, Addison-Wesley.

[KACP89] KACPRZYK, J. AT AL, 1989, "FQUERY III+: A Human Consisten Database Querying System Based on Fuzzy Logic with Linguistic Quantifiers", *Information Systems*, V.6, pp 443-453.

[KACP94] KACPRZYK, J., ZADROZNY, S., 1994, "Fuzzy Querying for Microsoft Access". In: *Proceedings of 3rd International Conference on Fuzzy Systems*, V.1, pp. 167-171, Orlando-FA.

[MAUR97] MAURO, R.C. ET AL, 1997, "Goa ++: Tecnologia, Implementação e Extensões aos Serviços de Gerência de Objetos". In: *Anais do XII Simpósio Brasileiro de Banco de Dados*, pp.272-286, Fortaleza-CE.

[MAUR98] MAURO, R.C.,MATTOSO, M.L.Q, 1998, "Integração de LPOO e BDOO: Uma experiência com JAVA e GOA++". In: *Anais do XIII Simpósio Brasileiro de Banco de Dados*, pp.169-184, Maringá-PR.

[MCFA91] MCFADDEN, F.R., HOFFER, J.A., 1991, Moden Database Management, 4th ed, Redwood City, CA, Benjamin/Cummings Publishing.

[MOTR95] MOTRO, A., 1995, "Management of Uncertainty in Database Systems". In: Kim, W. (ed), *Modern Database Systems: The Object Model, Interoperability and Beyond*, chapter 22, New York, Addison-Wesley.

[SALT83] SALTON. G., MCGILL, M.J., 1983. *Introduction to Modern Information Retrieval*, New York, McGraw-Hill.

[SAXE95] SAXENA, P., TYAGI, B., 1995, "Fuzzy Functional Dependencies and Independencies in Extended Fuzzy Relational Database Models", *Fuzzy Sets and Systems*, N. 69, pp. 65-89.