Representação de Conhecimentos em um Modelo de Dados Orientado a Objetos

Olga Maria Lopes Miranda
Lin Tse Min
Sonia Schechtman Sette
José Sergio Antunes Sette

Departamento de Informática-UFPE
Av. Prof. Luiz Freire, s/n - Cidade Universitária
Caixa Postal 7851
50.739 - Recife/PE - BRAZIL
e-mail: omlm@di.ufpe.br

Abstract

This work describes a formal approach to the expression of behavioural and deductive features in an Object-Oriented Data Model. The Object Orientation is preserved and the semantics is improved. The paradigms of Object-Orientation, Logics and Databases are integrated by formal specification to provide new resources.

Keywords

Behavioural and deductive knowledge representation, data models, object-orientation

1 Introdução

As características das aplicações ditas não convencionais estimularam a pesquisa de Modelos de Dados Orientados a Objetos (MDOOs), que se propõem a estreitar a diferença semântica entre a aplicação a ser modelada e a sua representação no modelo de dados ([3, 8, 10, 1]).

No entanto, além de permitir uma expressão mais fidedigna da realidade em questão, o Paradigma de Orientação a Objetos (POO) tornou-se de especial interesse por revelar-se como um paradigma unificador em projetos envolvendo Sistemas Baseados em Conhecimento, Bancos de Dados (BDs) e Linguagens de Programação.

Esta unificação, na realidade um desafio de enormes proporções, tem motivado a pesquisa anunciadora de uma nova geração de Sistemas Gerenciadores de Bancos de Dados (SGBDs) . Esta nova geração pretende permitir a expressão de conhecimentos procedimentais através do modelo de dados, além das características estruturais e organizacionais da aplicação modelada. Em [5] foram abordadas características requeridas por esta próxima

geração de SGBDs, que propõe "serviços" em três dimensões: gerenciamento de objetos, dados e conhecimento. Em [17] projetos de sistemas integrados que unifiquem BDs, o POO e Linguagens Lógicas para aplicações inteligentes são considerados futurísticos, ou seja, também prenunciadores de uma nova geração de SGBDs.

Este trabalho foi motivado por esta última corrente de pesquisa em BDs e compreende a especificação formal de tipos de objetos que ampliam a expressão de conhecimentos em um MDOO [14], caracterizando uma extensão do Modelo E/D [2, 15].

Na seção seguinte, a caracterização da proposta sob o enfoque de representação de conhecimentos é resumidamente apresentada. O modelo E/D é brevemente descrito na seção 3, utilizando a notação Zc [16]. Em seguida, os tipos definidos são especificados formalmente na seção 4. Finalmente, a seção 5 contém as conclusões e direções futuras.

2 Representação de Conhecimentos

Vários esquemas foram desenvolvidos para a representação do conhecimento humano de forma a obter uma definição mais natural do universo de discurso em Sistemas Baseados em Conhecimento, Sistemas Especialistas e Sistemas de Bancos de Dados Dedutivos.

Alguns destes esquemas enfatizam as características declarativas do conhecimento, outros a representação de aspectos procedimentais, enquanto outros baseiam-se nos aspectos estruturais do conhecimento a ser representado.

Sistemas de representação híbridos, que combinam as vantagens das técnicas de representação, tornaram-se então atrativos constituindo uma abordagem com alto poder de expressão de conhecimentos [9, 11].

O modelo de dados a ser apresentado pode ser classificado como um Modelo de Representação Híbrido, uma vez que: é fundamentado na filosofia de Orientação a Objetos; através do conceito de regras de produção se propõe a exprimir deduções e características comportamentais das entidades modeladas; a linguagem utilizada para a expressão de condições lógicas é baseada no Cálculo de Predicados [7].

A ênfase especial concedida ao suporte completo de diversos tipos de conhecimento permite uma modelagem precisa e natural dos aspectos do mundo da aplicação. Nesta proposta, dados e meta-dados são suportados sob um mesmo conceito: objetos. Assim, mais um passo é alcançado na unificação dos paradigmas.

3 O Modelo E/D

O Modelo Estático/Dinâmico (E/D) [2, 15] é uma extensão do Modelo Entidade/Relacionamento e caracteriza-se por ter uma visão uniforme de objetos com identificadores que são independentes de seus valores, por diferenciar entre as noções de tipos e classes e por prover uma semântica de inclusão de conjuntos, registros e seqüências para o mecanismo de generalização/especialização. Os objetos básicos da especificação do Modelo E/D [15] estão relacionados na Figura 1.

```
> GIVEN
>         NomeTipo,          // Nome de Tipo
>         NomeClasse,        // Nome de Classe
>         ValorInteiro,      // Conjunto de valores inteiros
>         ValorReal,         // Conjunto de valores reais
>         ValorString,       // Conjunto de valores string
>         INDEF,             // Valor indefinido
>         Mensagem,          // Texto
>         NomeOp,            // Nome de operação
>         IdObj,             // Identificador de objeto
>         Rótulo             // Identificador de campo de um registro
```

Figura 1

O Modelo E/D possui tipos simples pré-definidos também chamados de atômicos (Integer, Real, Bool, String) e construtores de tipos que permitem a definição de tipos estruturados. Estão disponíveis os construtores de Conjuntos (Conj), Seqüências (Seq) e Registros (Reg).

Cada tipo é caracterizado pela sua estrutura e pelas restrições que deve verificar, que são especificadas para cada um dos tipos do modelo. Além destas características próprias, cada tipo possui um nome (único entre todos os tipos que o modelo manipula), uma mensagem de descrição, os nomes dos tipos que são seus supertipos, uma indicação que informa se o tipo é excludente com algum outro tipo do modelo e um conjunto de operações que define sua interface e manipulação. Estas características comuns a todos os tipos do modelo compõem o esquema InfTipo, permitindo a modularização da especificação.

```
> SCHEMA InfTipo;
>   VAR
>       descrição      : Mensagem,
>       supertipos     : SET NomeTipo,
>       excludente     : SET NomeTipo,
>       operações      : Ops;
> END InfTipo;
```

A definição de um "Ambiente de Tipos" reúne as definições de todos os tipos que o modelo manipula em uma aplicação (tipos pré-definidos do modelo e tipos definidos pelo usuário). A identificação de cada tipo é única e corresponde ao nome do tipo. Cada tipo referenciado por outro tipo no conjunto de seus supertipos ou no conjunto dos tipos a ele excludentes, deve estar previamente definido no ambiente. Não podem existir também tipos com supertipos indicados como sendo mutuamente excludentes.

4 O Conhecimento Dedutivo e Comportamental

O Modelo E/D através do tipo Regra se propunha a tratar ações condicionais. O conceito de regras de produção ([13, 17, 11]) era utilizado, mas sem o propósito de expressar regras de dedução e reações.

Para dotar o Modelo E/D com as características requisitadas por uma nova geração de BDs, novos tipos foram definidos e outros sofreram alterações na especificação.

Através destes tipos, a capacidade de dedução e princípios de Bancos de Dados Ativos [12] são suportados no modelo. A seguir a extensão é especificada.

4.1 O tipo REGRA-DEDUÇÃO

O tipo Regra-Dedução expressa conhecimentos sob a forma de *objetos deduzidos*, constituindo a *base de dados intencional ou implícita*, enquanto os objetos persistentes constituem a *base de dados extensional ou explícita.*

A definição do tipo Regra-Dedução utilizando a notação Zc é a seguinte:

```
> SCHEMA   RegraDed;
>   INCLUDE InfTipo,
>   VAR
>       estrut : Estrutura[CorpoRegraDed],
> END   RegraDed;
```

A cláusula INCLUDE InfTipo copia todas as definições de InfTipo no esquema.

```
> TYPE
>         CorpoRegraDed   =   (sobre   : NomeEntRelaGen,
>                              se      : CondLog,
>                              entao   : SET AçãoDed);
> ALGTYPE   NomeEntRelaGen :: er NomeEntRela | erded NomeEntRelaDed;
> ALGTYPE   NomeEntRela :: ent NomeEntidade | rela NomeRela;
> ALGTYPE   NomeEntRelaDed   ::   entded NomeEntidadeDeduzida
                                | reladed NomeRelaDeduzido;
```

NomeEntidade corresponde ao nome de um tipo Entidade (entidade no ambiente de tipos). Analogamente, NomeRela, NomeEntidadeDeduzida e NomeRelaDeduzido correspondem respectivamente a nomes de tipos Rela, EntidadeDed e RelaDed no ambiente de tipos.

Assim, as características do tipo Regra-Dedução descrevem os tipos de objetos (sejam eles persistentes ou não) sobre os quais a regra está definida (aqueles presentes na parte SE), uma condição lógica a ser verificada e as ações dedutivas que devem ser executadas quando da verificação da condição.

A estrutura de uma CondLog especificada na parte SE do objeto Regra-Dedução foi inspirada naquela definida em [4]. No entanto, o POO é marcante na proposta, permitindo que condições lógicas sejam definidas sobre tipos/variáveis-objs.

A proposta aqui descrita assume que, ou uma linguagem baseada no Cálculo de Predicados [7] é fornecida pelo modelo (analogamente ao Cálculo Relacional de Tuplas no Modelo Relacional [6]) ou um mecanismo mais fraco, porém permitindo a expressão de fórmulas com um conjunto mínimo de conectivos, é suportado.

Admitindo que a primeira alternativa seja a considerada, uma condição lógica consiste em uma expressão do Cálculo de Predicados, onde os predicados são considerados os tipos que definem os objetos. Ou seja, um *predicado n-ário* representa um tipo entidade ou relacionamento com n atributos.

```
> ALGTYPE   CondLog :: ev EscopoVar | and (EscopoVar, Critério);
```

Uma CondLog compreende uma "parte de definição de escopo", que associa variáveis a

tipos de objetos e, opcionalmente, uma "subfórmula", que corresponde a uma expressão lógica sobre as variáveis-objs definidas na primeira parte.

```
> ALGTYPE  EscopoVar  ::  and (EscopoVar, EscopoVar)
>                         | and (EscopoVar, not(EscopoVar))
>                         | nerg NomeEntRelaGen (var IdObj);
```

Um EscopoVar pode ser negativo desde que haja ao menos um EscopoVar positivo na mesma regra. Isto significa que não existe no tipo o objeto especificado no EscopoVar positivo.

```
> ALGTYPE  Critério  ::  not Critério | and (Critério, Critério)
>                        | or (Critério, Critério)
>                        | aplopb (NomeOp, SEQ Expr)
>                        | aplqt (Quantif, Critério);
```

O tipo aplopb corresponde a aplicar qualquer operação de resultado booleano, de nome NomeOp à seqüência de expressões (SEQ Expr) correspondente a objetos do tipo dos parâmetros da operação e o tipo aplqt corresponde a aplicar um quantificador (universal ou existencial) a um critério. Uma expressão pode ser uma constante, uma variável ou a aplicação de operações a expressões.

Uma AçãoDed corresponde à especificação de um objeto deduzido através da definição de um predicado n-ário (correspondendo a um tipo n-ário). Ações dedutivas parametrizadas podem ser especificadas utilizando variáveis-objs (definidas na condição lógica) como argumentos das ações, sendo que uma variável pode estar associada a um tipo como um todo ou a uma propriedade de um tipo.

```
> TYPE   AçãoDed = nomeEntRelaDed (Variável);
> ALGTYPE  Variável :: var IdObj.Rótulo | var IdObj;
```

O tipo Unidade-Dedutiva consiste em uma meta-regra de dedução, no sentido em que é definido com o objetivo de agrupar objetos do tipo Regra-Dedução que definem um mesmo tipo de "objeto deduzido". Dessa forma, todos os objetos Regra-Dedução que possuírem (na parte ENTÃO do objeto) ações dedutivas a serem realizadas sobre um mesmo objeto deduzido, devem ser reunidos em um único objeto Unidade-Dedutiva.

O objeto Unidade-Dedutiva é definido como se segue:

```
> SCHEMA   UnidadeDed;
>   INCLUDE InfTipo,
>   VAR
>       estrut : Estrutura[CorpoUnidadeDed],
> END   UnidadeDed;
> TYPE   CorpoUnidadeDed =  (base      : NomeEsquemaE/D,
>                            obj_ded   : NomeEntRelaDed (SEQ Rótulo),
>                            regras    : (SEQ NomeRegraDed));
```

O esquema do Modelo E/D é especificado como o par contendo o ambiente de tipos definidos dentro da aplicação e as classes definidas para cada tipo.

O domínio de valores para cada rótulo da estrutura de um objeto deduzido não precisa ser especificado, uma vez que pode ser inferido através das regras que o definem (SEQ NomeRegraDed). As outras características dos objetos deduzidos (restrições, relacionamentos totais, etc) também são inferidas a partir dos objetos que as regras utilizam para

compor o objeto deduzido (aqueles especificados no campo "sobre" das regras utilizadas).

Para a definição de um tipo deduzido pode ser necessária a definição (também virtual) de tipos intermediários, que também devem constar na Unidade-Dedutiva relacionada.

4.1.1 A Semântica de Avaliação

A Semântica de avaliação de objetos do tipo Regra-Dedução é definida a partir da interpretação das partes SE e ENTÃO. Para a definição semântica das partes, é necessária a noção de *instanciação*.

Uma instanciação de uma Regra-Dedução consiste na substituição de cada variável livre por um objeto de uma classe associada a um tipo. Para relacionar instâncias de uma mesma classe, basta que sejam definidas variáveis-objs, quantas forem necessárias. Dessa forma, cada variável é substituída eventualmente por instâncias diferentes, conforme requerido.

A parte SE de uma Regra-Dedução retorna o conjunto de objetos que são obtidos através da instanciação das variáveis-livres presentes na condição lógica e que satisfazem aos critérios definidos. Se o conjunto vazio é retornado, a não-avaliação da parte AÇÃO da Regra-Dedução é determinada. Este conjunto só existe em tempo de avaliação da regra e é utilizado na parte ENTÃO.

Para cada instância pertencente ao conjunto resultante da avaliação da parte SE, as ações dedutivas são executadas.

4.2 O tipo REGRA-COMPORTAMENTO

A definição do tipo Regra-Comportamento utilizando a notação Zc é a seguinte:

```
> SCHEMA   RegraComp;
>   INCLUDE InfTipo,
>   VAR
>       estrut : Estrutura[CorpoRegraComp],
> END   RegraComp;
> TYPE   CorpoRegraComp   =   (se      : CondLogPer,
>                              então   : NomeAção);
> ALGTYPE   CondLogPer :: evp EscVarPer | and (EscVarPer,Critério);
> ALGTYPE   EscVarPer   ::   and (EscVarPer, EscVarPer)
>                          | and (EscVarPer, not(EscVarPer))
>                          | ner NomeEntRela (var IdObj);
```

Uma CondLogPer traduz um CondLog que não pode ser aplicada a NomeEntRelaDed, ou seja, uma RegraComp pode ser definida apenas para Entidades e Relacionamentos (persistentes).

Uma Ação corresponde a qualquer conjunto de operações especificado para objetos do tipo AÇÃO do Modelo E/D. Ações parametrizadas podem ser definidas utilizando variáveis-objs como argumentos das ações.

```
> SCHEMA   Ação;
>   INCLUDE InfTipo,
>   VAR
>       estrut : Estrutura[CorpoAção],
> END   Ação;
```

```
> TYPE  CorpoAção  =  (sobre  : SET NomeEntRela,
>                      act    : SET OpAct);

> ALGTYPE  OpAcT :: opd (OpDDL, Arg) | nop (NomeOp, Arg) | na NomeAção;
```

As operações do tipo OpDDL correspondem às operações da linguagem de definição do modelo. NomeOp corresponde ao nome das operações específicas definidas para cada tipo. Arg deve corresponder a tipos da seqüência de argumentos da operação.

```
> ALGTYPE  Arg :: var IdObj.Rótulo | cte Objeto | SEQ Arg;
```

4.2.1 O tipo REGRA-GATILHO

O tipo Regra-Gatilho contém as mesmas características do tipo Regra-Comportamento, permitindo no entanto, a definição de uma pré-condição para a execução da regra, assim como o tipo de gatilho pretendido e o estado momentâneo do mesmo.

A definição do tipo Regra-Gatilho utilizando a notação Zc é a seguinte:

```
> SCHEMA  RegraGat;
>   INCLUDE RegraComp,
>   VAR
>       estrut : Estrutura[CorpoRegraGat],
> END  RegraGat;
> TYPE  CorpoRegraGat  =  (pre-cond  : Evento,
>                          estado    : Status,
>                          tipo      : TipGat);
```

A pré-condição pode ser especificada como um evento operatório ou temporal. Um evento operatório especifica a ocorrência de operações sobre objetos ou classes, enquanto um evento temporal diz respeito à passagem do tempo. Ou seja, como em [12], um evento não se restringe a ocorrências de modificações de propriedades dos objetos manipulados no Modelo E/D. Assim, a especificação de um evento explicita quando uma determinada Regra-Gatilho deve ser avaliada.

```
> ALGTYPE  Evento :: eop EventoOp | etp EventoTp;
> TYPE  EventoOp  ::  (sobre      : NomeEntRela,
                       ocorrencia : NomeAção);
> ALGTYPE  EventoTp  ::  and (Critério, CondTemp)
                         | or (Critério, CondTemp)
                         | not (CondTemp) | ct CondTemp;
```

Uma CondTemp é uma condição que retorna um Bool e utiliza uma Classe Sistema que responde operações (mensagens) como *Datasis, DiaSis, MesSis, AnoSis.*

O estado da regra (se habilitada ou não) determina se a ocorrência da pré-condição especificada deve ou não engatilhar a regra, enquanto o tipo especifica se um gatilho é temporal ou operatório (de acordo com o evento especificado). Se o gatilho é operatório, a especificação corresponde a determinar se o mesmo deve ser avaliado antes (PRÉ) ou depois (PÓS) da ocorrência do evento.

```
> ALGTYPE  Status :: 'habilitada' | 'desabilitada';
> ALGTYPE  TipGat :: 'PRE' | 'POS' | 'TEMP';
```

O tipo Meta-Regra é o responsável pelo controle de execução de regras de comporta-

mento e gatilho, permitindo que prioridades e regras excludentes sejam expressas.

```
>  SCHEMA   MetaRegra;
>    INCLUDE InfTipo,
>    VAR
>        estrut : Estrutura[CorpoMetaRegra],
>  END   MetaRegra;
>  TYPE   CorpoMetaRegra  =  (quando        : OpAct,
>                             sobre         : NomeEn ,
>                             exclud-priori : SET (SEQ Regra));
>  ALGTYPE   Regra :: RegraComp | RegraGat;
```

Todas as regras que podem ser executadas quando uma OpAct for enviada a um determinado NomeEntRela devem ser explicitadas em "SET (SEQ Regra)", que corresponde a um conjunto de seqüências de regras. Cada seqüência dentro do conjunto determina a ordem de prioridade em que as regras devem ser executadas e diferentes seqüências indicam que a execução de uma lista é excludente à de outra.

O tipo Transação foi modificado apenas para agrupar regras de comportamento, gatilho e ações.

4.2.2 Semântica de Avaliação

Uma regra (gatilho ou comportamento) pode engatilhar a execução de outras, ocorrendo um aninhamento. Por outro lado, uma regra apenas *dispara/ativa* outra, jamais *chama* outra, ou seja, o que causa o aninhamento é a modificação de estados dos objetos.

Em objetos do tipo RegraGat, a avaliação de Evento retorna um objeto do tipo Bool. Caso o evento avalie para verdadeiro, a condição é avaliada (de modo semelhante àquele de objetos do tipo RegraDed). Se não, o par (SE, ENTÃO) não é avaliado.

Em qualquer regra (gatilho ou comportamento), se a condição se verifica, as ações são executadas (parte ENTÃO). Caso contrário, a avaliação da regra termina com a avaliação da condição.

5 Conclusões e Direções Futuras

Foram formalmente especificados os tipos que introduziram no Modelo E/D conceitos de BDs Dedutivos e Ativos compondo uma arquitetura integrada (o suporte às novas características não se deu através de camadas adicionais e interfaces, proporcionando um modelo de dados robusto e contido).

Através da definição do tipo Regra-Dedução, há uma economia de espaço de armazenamento, uma vez que os objetos deduzidos são objetos virtuais e não estão efetivamente armazenados na Base. O poder de expressão de objetos do tipo Regra-Dedução permite ainda que relações de cardinalidade arbitrária sejam definidas. Por outro lado, objetos do tipo Regra-Comportamento e Regra-Gatilho incorporaram ao modelo uma representação semântica mais fiel ao comportamento das aplicações modeladas.

Dessa forma, características de Orientação a Objetos, Lógica de Primeira Ordem e Bancos de Dados foram integradas e especificadas formalmente.

O tratamento formal ocorreu sobre as características estruturais dos tipos que permitem

a expressão de *deduções, comportamentos e reações* no modelo. A continuidade deste trabalho visa à especificação das operações destes tipos e à prototipagem da proposta, facilitada pela abordagem rigorosa do método de especificação formal utilizado.

Referências

[1] **M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier e S. Zdonik**: "THE OBJECT-ORIENTED DATABASE SYSTEM MANIFESTO", *Proceedings of the 1989, International Workshop on Object-Oriented Database Systems*, 1989.

[2] **M. Bandeira**: "MODELAGEM ESTÁTICA/DINÂMICA DE SISTEMAS DE INFORMAÇÕES", *Dissertação de Mestrado*, UFPE, Recife, 1989.

[3] **J. Banerjee, H. Chou, J. Garza, W. Kim et al**: "DATA MODEL ISSUES FOR OBJECT ORIENTED APPLICATIONS", *ACM Transactions on Office Information System*, 5(1), 1987.

[4] **M. Barthez**: "INTEGRATION D'UN LANGAGE DE REGLES DE PRODUCTION DANS UN SGBD RELATIONNEL: PRESENTATION DU LANGAGE ET DU CONTROLE DE L'EXECUTION", *These de Docteur en Informatique*, Conservatoire National des Arts et Metiers, Paris, 1990.

[5] **D. Beech, P. Bernstein, M. Brodie, M. Carey, B. Lindsay, L. Rowe, M. Stonebraker**: "THIRD-GENERATION DATABASE SYSTEM MANIFESTO". *Proc. IFIP Conference on Object-Oriented Database Systems - Analysis, Design, Construction*, 1990.

[6] **E. F. Codd**: "A RELATIONAL MODEL FOR LARGE SHARED DATA BANKS", *Communications of the ACM*, 13(6), 1970.

[7] **H.B. Enderton**: "A MATHEMATICAL INTRODUCTION TO LOGIC", Academic Press, 1972.

[8] **D. Fishman et al.**: "IRIS: AN OBJECT-ORIENTED DATABASE MANAGEMENT SYSTEM", *ACM Transactions on Office Information Systems*, 5(1), 1987.

[9] **R. Fikes, T. Kehler**: "THE ROLE OF FRAME-BASED REPRESENTATION IN REASONING", *Communications of the ACM*, 28(9), 1985

[10] **C. Lécluse, P. Richard e F. Velez**: "O_2, AN OBJECT ORIENTED DATA MODEL", *Proc. SIGMOD International Conference*, Chicago, 1988.

[11] **N. M. Mattos**: "AN APPROACH TO KNOWLEDGE BASE MANAGEMENT", *Lecture Notes in Artificiall Intelligence, Springer-Verlag*, 1991.

[12] **C. Medeiros, P. Pfeffer**: "TRANSFORMAÇÃO DE UM BANCO DE DADOS ORIENTADO A OBJETOS EM UM BD ATIVO": *Anais do VI Simpósio Brasileiro de Bancos de Dados*, Manaus - AM, 22 a 24 de Maio de 1991.

[13] **R. N. Melo** "BANCO DE DADOS NÃO CONVENCIONAIS: A TECNOLOGIA DE BD E SUAS NOVAS ÁREAS DE APLICAÇÃO": *VI Escola de Computação*, Campinas - SP, 1988.

[14] **O. Miranda, S. Sette, J. Sette**: "ESTENDENDO UM MODELO DE DADOS ORIENTADO A OBJETOS PARA EXPRESSÃO DE CONHECIMENTO DEDUTIVO E COMPORTAMENTAL", *VII SBBD*, Porto Alegre - RS, 1992.

[15] **R. Motz**: "ESTUDO FORMAL DE UM MODELO DE DADOS ORIENTADO A OBJETOS", *Dissertação de Mestrado*, UFPE, Recife, 1990.

[16] **A. C. A. Sampaio**: "ZC: UMA NOTAÇÃO PARA ESPECIFICAÇÃO DE SISTEMAS COMPLEXOS", *Dissertação de mestrado*, UFPE, Recife, 1988. 201, 1985.

[17] **C. Zaniolo, H. Ait-Kaci, D. Beech, S. Cammarata, L. Kershberg, D. Maier**: "OBJECT ORIENTED DATABASE SYSTEMS AND KNOWLEDGE SYSTEMS", *Expert Database Systems*, 1986.