Integração de Regras Ativas e Dedutivas no Paradigma de Orientação a Objetos

Fernanda Lima
e-mail: ferlima@inf.puc-rio.br

Astério Tanaka
Centro de Ciências Exatas e de Tecnologia, Universidade do Rio de Janeiro, UNI-RIO
Rio de Janeiro, Brasil
e-mail: tanaka@uniriotec.br

Rubens Nascimento Melo
e-mail: rubens@inf.puc-rio.br

PUC-RioInf.MCC 16/99 August, 1999

Abstract

The use of rules in Database Systems increases language expressiveness and consequently, increases the processing power of Databases. The main objective of this work is to investigate different possibilities of integrating active and deductive rules in databases. Another approach is also presented where the Object Oriented (OO) paradigm is used initially as data model and language and later is enriched with active and deductive rules. Therefore, this work does not intend to evaluate an isolated aspect of using active rules and deductive rules, or even the utilization of OO paradigm in databases, but mainly to explore the combinations of these approaches and the possible advantages.

Keywords: Active rules, Deductive Rules, Databases, Object-Orientation

Resumo

O uso de regras em Sistemas de Bancos de Dados (SBDs) aumenta o poder de expressividade das linguagens, e conseqüentemente, aumenta a capacidade de processamento dos SBDs. O objetivo principal deste estudo é investigar propostas de integração de regras ativas e dedutivas em geral, e uma proposta em particular onde o paradigma de orientação a objetos (OO) é utilizado inicialmente como modelo de dados e linguagem para posteriormente ser enriquecido com regras ativas e dedutivas. Este estudo visa, portanto, não avaliar apenas um aspecto isolado do uso de regras ativas ou de regras dedutivas, ou ainda do paradigma OO isoladamente, e sim investigar a combinação possível destas abordagens e as vantagens oferecidas.

Palavras-chave: Regras Ativas, Regras Dedutivas, Bancos de Dados, Orientação a Objetos

1. Introdução

Conforme pode ser encontrado na literatura [CR96], estender Sistemas de Bancos de Dados (SBDs) com regras ativas e dedutivas é um dos maiores passos na evolução da tecnologia de Banco de Dados (BD). Através de regras, os projetistas podem expressar conhecimento declarativo e procedural de suas aplicações, pois estas regras aumentam o poder de expressividade das linguagens de BD, resultando no aumento da capacidade de processamento.

Regras ativas especificam o comportamento ativo desejado em Sistemas de Bancos de Dados. Uma vez que um conjunto de regras ativas é definido, o SBD ativo monitora os eventos relevantes. O paradigma geral utilizado por SBDs ativos pode ser descrito da seguinte forma: para cada regra, se o evento declarado ocorre, então o SBD ativo avalia a condição estabelecida na regra e se esta condição for verdadeira, a ação descrita na regra é executada.

Regras dedutivas são declarativas ao estilo de Programação em Lógica. Estas regras fornecem uma interface ao usuário onde novos fatos são deduzidos da base de dados através de mecanismos de inferência [EN94].

Neste contexto de regras ativas e dedutivas reside a questão de como usufruir dos benefícios proporcionados pelo poder declarativo das regras dedutivas e também contar com a execução automática de operações em resposta a eventos e condições, definidos por regras ativas. Propostas foram realizados com o intuito de tratar regras ativas e dedutivas em um mesmo ambiente de BD. Em [Wido93], Widom apresenta uma discussão do relacionamento entre SBDs ativos e dedutivos. Em [CW94], Ceri e Widom descrevem SBDs dedutivos implementados usando SBDs ativos. Já a proposta de Bayer e Jonker [BJ93] é estender os SBDs dedutivos com funcionalidade ativas. A integração dos dois tipos de regras em SBDs estendidos é descrita por Harrison e Dietrich em [HD93] e por Zaniolo em [Zani93].

Estas diferentes propostas comentadas neste trabalho abordam problemas específicos em sua maioria associados ao modelo relacional. Entretanto, o paradigma OO pode fornecer contribuições.

A ênfase da orientação a objetos tem sido fornecer um mecanismo de modelagem natural para objetos do mundo real, encapsulando suas estruturas e métodos, de modo a ampliar a capacidade de modelagem. Esta vantagem pode ser estendida a ambientes de SBDs orientados a objetos ou pode ser utilizada em SBDs relacionais, através de mapeamentos.

Projetos de pesquisa realizaram implementações de BDs utilizando funcionalidades genéricas de processamento de regras para dar suporte a SBDs simultaneamente orientados a objetos, dedutivos e ativos, como OSAM* da Universidade da Flórida [CHS92], A DOOD RANCH da Universidade do Arizona [DUHK92], LOGRES da Universidade de Milão [C+90], antecessor do Projeto Chimera [CM93].

O modelo de dados e a linguagem Chimera são apresentados como exemplo do uso de regras ativas e dedutivas dentro do contexto de Orientação a Objetos. É importante comentar

que a metodologia IDEA para aplicações com objetos e regras foi desenvolvida [CF97a, CF97b, CFP98, CBFP95], visando preencher a lacuna constatada por diversos autores [CR96, SK95, Wido94]. Estes autores afirmam que, apesar da evolução das pesquisas em regras e da incorporação de algumas destas funcionalidades em produtos comerciais, regras continuam sendo pouco usadas. Pesquisadores e desenvolvedores concordam que a falta de metodologias para projeto e de ferramentas é um dos maiores dificultadores para o desenvolvimento de aplicações com uso de regras.

Vale ressaltar que em [TNCK91, NTC93, TN94 e CTC98], os autores apresentam extensões do modelo E-R para incorporar comportamento ativo ao modelo conceitual, bem como um conjunto de ferramentas de projeto de SBDs ativos.

O objetivo deste trabalho é avaliar o uso do paradigma de orientação a objetos em conjunto com regras ativas e dedutivas. Um maior poder de expressividade é alcançado em diversos aspectos de SBDs. Em um mesmo contexto, é possível haver eventos mais complexos envolvendo detecção de comportamento além da possibilidade de navegação por hierarquia de classes (inexistentes no modelo relacional). As condições podem ser expressas declarativamente (somente possível com regras dedutivas). A linguagem declarativa pode ainda expressar consultas, restrições de integridade, atributos derivados representando relacionamento entre classes ou até mesmo classes derivadas e visões (tema ainda não resolvido em OO). Esta integração permite que regras ativas monitorem o banco de dados e garantam as derivações especificadas.

É importante ressaltar que o objetivo deste trabalho não é restringir o uso de regras ativas e dedutivas a SBD orientados a objetos e sim investigar uma direção de pesquisa interessante que inicia de um modelo de dados orientado a objetos e enriquece-o com regras ativas e dedutivas para oferecer suporte adicionais, como visões e restrições de integridade.

Este trabalho está estruturado da seguinte forma: a seção 2 apresenta os conceitos de regras ativas e dedutivas e suas utilizações em SBDs, enquanto que a seção 3 descreve conceitos básicos de orientação a objetos. Na seção 4 são comentadas diferentes propostas para a integração das regras ativas e dedutivas. Em particular, a derivação de regras ativas a partir de dados dedutivos é exemplificada na seção 5. A aplicação de diversos destes conceitos no modelo de dados e linguagem Chimera são descritos na seção 6, destacando suas funcionalidades ativas e dedutivas. E, finalmente, na seção 7 podem ser encontradas as conclusões deste trabalho.

2. Regras em Bancos de Dados

Regras ativas e dedutivas são importantes pois podem expressar grande parte da semântica codificada em aplicações. A representação do conhecimento através de regras pode ser encarado como um nível adicional de independência das aplicações, conhecido como independência de conhecimento (*knowledge independence)* [Tana95, CR96]. Assim como os níveis de independência física e lógica, o nível de independência de conhecimento é também compartilhado por todas as aplicações.

A seguir, um resumo dos conceitos envolvidos no uso de regras ativas e dedutivas em SBDs é realizado.

2.1 Regras Ativas

Regras ativas foram influenciadas por trabalhos na área de Inteligência Artificial, descrevendo reações computacionais automáticas que ocorrem em resposta a eventos de manipulação de dados [CR96, MST97, Tana95]. Em SBDs convencionais passivos, dados são criados, consultados, alterados e removidos em resposta a operações explícitas de usuários ou aplicações. Já os SBDs ativos são capazes de responder de forma autônoma a operações, quando determinados estados ou transições ocorrem.

Regras ativas em SBDs são importantes para preservar restrições de integridade, para materialização de dados derivados, suporte a regras de negócio, dentre outros exemplos. As pesquisas na área de Bancos de Dados Ativos[1] iniciaram em Bancos de Dados relacionais, contudo, regras ativas podem ser inseridas em SBDs independente do modelo de dados utilizado, como demonstram diversas pesquisas em Bancos de Dados Orientados a Objetos ativos [MP91, SMT97, MST98].

Regras em Bancos de Dados Ativos especificam o comportamento ativo desejado e de forma genérica consistem de três partes:

- Eventos que fazem com que a regra seja disparada,
- Condições que são verificadas quando as regras são disparadas e
- Ações que são executadas quando a regra é disparada e a condição é verdadeira.

Outros termos têm sido utilizados como sinônimos para regras em SBDs ativos, como regras em produção (como em linguagens de Inteligência Artificial), regras ECA (para Evento-Condição-Ação), *triggers*, monitores e alertas. Entretanto, a idéia básica pode ser descrita da seguinte forma: após a definição de um conjunto de regras, o SBD ativo executará a monitoração dos eventos relevantes. Para cada regra, se o evento declarado ocorre, então o SBD ativo avalia a condição estabelecida na regra e se esta condição for verdadeira, a ação descrita na regra é executada.

Uma vez que o comportamento desejado de um SBD ativo é especificado através de uma linguagem de BDs de regras ativas, a expressividade da linguagem de regras tem impacto direto no poder e na complexidade do SBD ativo [Tana95, WC96]. Como exemplo, pode-se citar eventos de consulta em SBDs relacionais (SBDRs) como operações de seleção em uma tabela, enquanto que em SBDOOs estes eventos podem ser tanto a seleção de um objeto quanto uma chamada a um método que recupera objetos. Já eventos de modificação de dados em SBDRs podem ser operações de inserção, remoção e atualização em uma determinada tabela, e em SBDOOs, podem ser criação, remoção e atualização de um objeto

[1] A expressão "Banco de Dados Ativo", amplamente utilizada em publicações atuais, foi cunhada por Mathew Morgenstein em [Morg83].

específico, ou de qualquer objeto em uma determinada classe, ou ainda uma chamada a método que modifica objetos.

2.2 Regras Dedutivas

Regras dedutivas originam-se dos estudos de Programação em Lógica, oferecendo uma linguagem poderosa em diversos aspectos como suporte à agregação, negação e recursão [CR96]. Este tipo de regra expressa conhecimento declarativo (*declarative knowledge*) pois pode representar consultas de forma a refletir o seu significado sem depender da estratégia de avaliação da consulta.

SBDs dedutivos utilizam especificações de fatos e regras. Fatos são especificados de forma semelhante à especificação de relações, enquanto que regras podem ser comparadas a visões. Através do uso de regras dedutivas, relações virtuais não armazenadas na base podem ser formadas a partir de fatos. O mecanismo de inferência do sistema pode deduzir estes novos fatos da base de dados através da interpretação das regras. A principal diferença entre regras e visões é que as regras dedutivas podem envolver recursão e produzir visões que não podem ser definidas em termos de visões relacionais tradicionais [EN94].

Existem diferentes formas de inserir regras dedutivas em SBDs, como povoar classes derivadas ou definir atributos derivados. Além disto, estas regras podem expressar restrições de integridade, restrições de domínio ou relacionamentos entre classes. A linguagem contendo regras dedutivas pode ser também usada como linguagem de consulta declarativa.

Muitos trabalhos têm sido realizados visando integrar regras dedutivas e SBDs, independente do modelo de dados utilizado.

3. Paradigma de Orientação a Objetos

O paradigma de orientação a objetos pode ser caracterizado pelos conceitos descritos a seguir. Cada entidade é modelada como um objeto, sendo que cada objeto está associado a um identificador único. Objetos possuem atributos e métodos. O valor dos atributos pode ser um valor atômico, outro objeto ou também um conjunto de objetos. Métodos são usados para acessar ou modificar valores e estados em objetos. Classes representam o agrupamento de objetos que possuem a mesma estrutura e comportamento. Uma classe pode ser definida como especialização de uma ou mais classes. A classe definida como especialização é chamada sub-classe e, através do conceito de herança, herda atributos e métodos de suas super-classes.

Neste trabalho os conceitos de orientação a objetos não estão necessariamente associados a implementações em SBDOOs. A idéia aqui é que o paradigma OO pode ser utilizado como modelagem de dados e linguagem, independente da implementação física a ser utilizada. Se a opção for um SBD relacional, mapeamentos serão realizados. Se a opção for um SBDOO, o esquema poderá ser utilizado diretamente.

Em SBDOOs ativos, regras podem ser tratadas como objetos de primeira ordem [WC96]. Isto significa que algumas propriedades são obtidas automaticamente. Como exemplo, pode-se citar a criação e remoção de regras eqüivalendo a inserção e remoção de instâncias em extensões, ao invés das codificações necessárias das mesmas propriedades no modelo relacional. Outras propriedades são controle de concorrência e autorização. Em OO, classes, coleções e herança permitem estruturar e modularizar conjuntos de regras.

4. Propostas de Integração de Regras

Alguns trabalhos foram realizados com o intuito de tratar regras ativas e dedutivas em um mesmo ambiente de BD. Para facilitar a compreensão das propostas encontradas da literatura, os estudos foram agrupados da seguinte maneira:

Inicialmente é apresentada uma discussão do relacionamento entre os SBDs ativos e dedutivos na sub-seção 4.1. Em seguida, são enumeradas as propostas de:

- Regras dedutivas implementadas usando SBDs ativos;
- SBDs dedutivos estendidos com funcionalidades ativas, significando o oposto da proposta anterior;
- SBDs integrados com propriedades ativas e dedutivas.

Os conceitos apresentados em 4.1 e 4.2 foram utilizados no restante deste trabalho.

4.1 Discussão sobre relacionamento de SBDs ativos e dedutivos

Em [Wido93], Widom classifica BDs ativos e dedutivos relacionais em um espectro de linguagens de regras de BDs. Este espectro corresponde a uma noção de nível de abstração, com as linguagens de regras dedutivas em um nível maior e as linguagens de regras ativas em um nível menor.

A autora introduz uma notação genérica para regras e classifica as linguagens Datalog, RDL, A-RDL, Ariel, Starbust e Postgres da seguinte forma:

Dedutivo — Ativo

Nível mais alto — Nível mais baixo

Datalog... RDL... A-RDL... Ariel... Starbust... Postgres

A interpretação genérica das regras é: [quando um Ativador], se um Antecedente, então um Conseqüente. E a notação utilizada é :

[Ativador~>] Antecedente ~> Conseqüente

Ao longo do espectro partindo das linguagens dedutivas para ativas, Widom observa que são adicionados aspectos importantes, como: noção de estados de transições, operações mais poderosas para conseqüentes e ativadores mais complexos.

A principal conclusão obtida em [Wido93] é que deve ser possível utilizar linguagens de baixo nível de abstração (linguagens ativas) como alvo de compiladores para linguagens de mais alto nível, como por exemplo um compilador de Datalog para Starbust.

A classificação, conforme mencionado, foi realizada para linguagens de BDs relacionais. No caso de SBDOOs, um estudo interessante seria estudar cada uma das linguagens propostas na literatura e posicioná-las em um espectro semelhante, se possível.

A linguagem Chimera, segundo esta classificação, pode ser dividida em dois pontos do eixo. A parte dedutiva de Chimera é proveniente de Datalog, portanto ocupa a mesma posição desta linguagem. Já a parte ativa possui mais recursos que todas as outras linguagens ativas apresentadas, podendo ser classificada à direita de todas as linguagens analisadas. Além de referenciar estados anteriores como Postgres, Chimera possui ativadores mais complexos. Em Chimera, é utilizado o conceito de compilador de linguagem de alto nível para linguagem de baixo nível proposta em [Wido93]. A descrição detalhada da linguagem pode ser encontrada na seção 6 deste documento.

Esta classificação, embora interessante, não trata aspectos importantes como relacionamento entre transações. Em [FWP97], os autores comentam que a semântica operacional dos SBDs ativos é apresentada informalmente e com alto nível de abstração. Críticas são feitas em relação a linguagem Datalog ser apresentada de forma simplificada como uma linguagem de definição de visões, ignorando sua expressividade para domínios de aplicação, como restrições de integridade e regras de negócio. A resposta a estes comentários é esclarecida ao longo deste trabalho, uma vez que a linguagem Chimera utiliza Datalog nas funções mencionadas e também utiliza regras ativas.

4.2 Regras dedutivas implementadas usando SBDs ativos

Em [CW94], Ceri e Widom demonstram como regras dedutivas podem ser suportadas usando facilidades existentes em SBDs ativos. Como motivação inicial para a escolha desta abordagem, os autores afirmam que o crescimento das funcionalidades de regras ativas, tanto em protótipos de pesquisa, SBDs comercias e no padrão SQL3, apontam para um grande uso de regras na prática.

Outro fator motivador desta proposta tem como base o estudo apresentado na sub-seção acima. Conseqüentemente, em [CW94], as regras ativas são tratadas como mecanismo de baixo nível, usadas para implementar funcionalidades de mais alto nível, como a interface

fornecida por SBDs dedutivos. Regras ativas são derivadas com o objetivo de manter dados intensionais quando os dados extensionais são atualizados.

No artigo mencionado, os autores apresentam estudos realizados no contexto do Starbust (paradigma relacional), porém comentam que a mesma abordagem pode ser utilizada em outras linguagens de regras, como é o caso da linguagem Chimera a ser descrita.

Esta proposta será apresentada com maiores detalhes na seção 5.

4.3 SBDs dedutivos estendidos com funcionalidades ativas

Em [BJ93], a proposta de Bayer e Jonker é estender os SBDs dedutivos com funcionalidade ativas. Os autores descrevem um *framework* para suportar *triggers* no contexto de BDs dedutivos baseado na definição de eventos primitivos, eventos compostos e ocorrência de eventos.

Neste artigo, o suporte a regras ativas no contexto de BDs dedutivos pode se beneficiar da expressividade destes BDs dedutivos oferecendo uma definição declarativa da condição, em termos de predicados da base (extensionais) ou virtuais (intensionais).

Eventos, condições e ações são especificados formalmente, juntamente com a semântica de execução. Entretanto, o ambiente de banco de dados é estritamente dedutivo, não permitindo considerar orientação a objetos.

Os autores comentam que esta integração com OO é uma investigação interessante a ser realizada.

4.4 SBDs integrados com propriedades ativas e dedutivas

A integração dos dois tipos de SBDs estendidos é descrita por Harrison e Dietrich em [HD93] e por Zaniolo em [Zan93].

Em [HD93], a proposta de Harrison e Dietrich é aumentar a expressividade das regras ativas do tipo E-C-A para detectar eventos complexos e expressar condições relacionadas a atualização de dados armazenados e derivados. As atualizações podem ser detectadas sem materializar as relações derivadas.

No artigo em questão, é proposto um SBD ativo que permite ao usuário especificar eventos em relações derivadas (definidas em regras de Datalog) e condições complexas envolvendo tanto relações armazenadas quando derivadas. A extensão da semântica operacional de SBDs dedutivos é utilizada para capturar alguma forma de comportamento ativo. Neste caso, os eventos que podem ser especificados são operações de atualização. Entretanto, não há indicação que o formalismo apresentado pode incorporar interações com um gerenciador de transações.

Zaniolo [Zani93] formaliza uma nova semântica operacional para SBDs dedutivos, baseada em estratificação. Em seu estudo, é proposta uma extensão que permite a integração de SBDs ativos e SBDs dedutivos. Para tal, é usada uma extensão não monotônica de

cláusulas lógicas, incluindo negação e agregação sob uma semântica de estratificação chamada X-Y.

Neste ambiente, regras ativas são expressas através de predicados embutidos que implementam operações básicas de atualização. Uma semântica uniforme, essencialmente declarativa, é fornecida através de técnicas de transformação. Esta proposta pressupõe uma arquitetura básica que possa processar regras dedutivas puras e também simular o efeito do comportamento ativo.

Em [FWP97], os autores afirmam que a elegância alcançada na abordagem apresentada por Zaniolo é proveniente de uma visão restrita do comportamento ativo de regras, onde não há uso de eventos complexos nem interseção de execução de regras com modelos de transação suportados por SBDs.

5. Especificação Declarativa de Regras Ativas

Em [Ceri92, CFPW94], os autores apresentam, em uma visão geral, como diversas aplicações de Bancos de Dados Ativos podem ser especificadas de forma declarativa. A Figura 5.1 mostra uma arquitetura de referência para geração semi-automática de regras, iniciando com especificações de alto nível. Esta abordagem foi utilizada para restauração de restrições de integridade após violações [CFPT94], manutenção incremental de visões convencionais (SQL) [CW91] e para manutenção incremental de visões no contexto de visões definidas por regras dedutivas (com recursão e negação estratificada) [CW94].

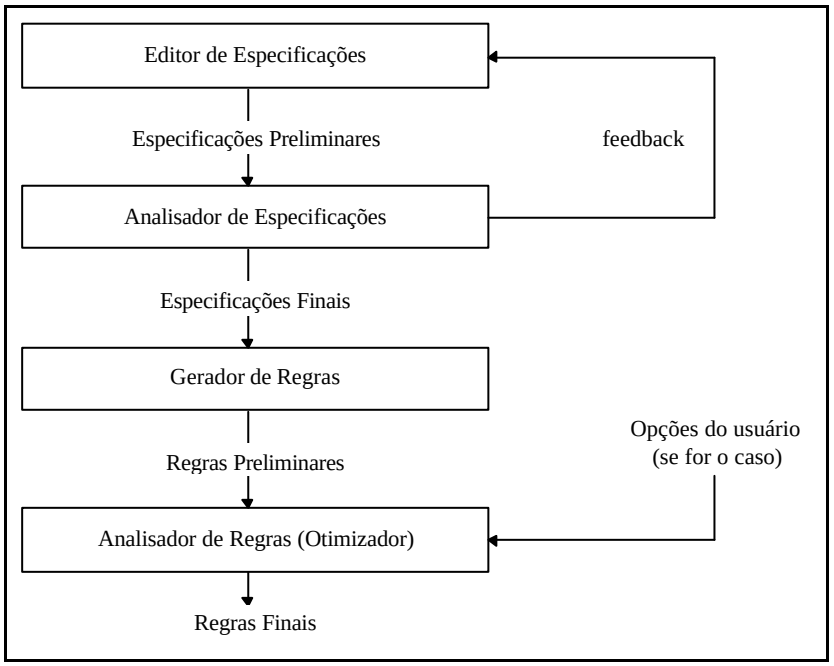

Figura 5.1 - Ambiente para geração de regras ativas a partir de especificações declarativas

De forma geral, a motivação desta abordagem é que regras ativas são difíceis de compreender e manter, devido a sua interação complexa, porém uma ferramenta pode ser capaz de gerar regras que satisfaçam critérios específicos, como terminação e confluência.

No ambiente proposto, especificações declarativas, que podem ser fornecidas por um projetista, são traduzidas para regras ativas a serem incorporadas em um Banco de Dados Ativo. Para alguns casos, a derivação de regras a partir das especificações pode ser completamente automática. Porém existem situações onde diferentes comportamentos ativos são possíveis. Nestes casos, o projetista deve fornecer as diretivas para o analisador de regras e a derivação será semi-automática.

As técnicas de análise de regras podem ser usadas para validar regras produzidas pelo gerador. Por exemplo, pode ser identificada uma coleção de regras com potencial de não terminação (regras que podem disparar umas as outras indefinidamente) [BCP95]. Neste contexto, o projetista pode validar um conjunto de regras e assegurar que a execução sempre terminará para o conjunto de dados existente, ou também pode alterar as regras para que a terminação seja alcançada .

Uma vantagem de utilizar mecanismos de geração automáticos para regras ativas com estruturas conhecidas é que a otimização de regras pode ser aplicada de forma bem sucedida. Uma variedade de técnicas pode ser utilizada, como substituir diversas regras por uma equivalente *(net effect)*, ou substituir as referências a tabelas inteiras por referências a estruturas de dados menores descrevendo as mudanças geradas pelas transações *(delta relations*).

Na sub-seção a seguir é apresentado o trabalho desenvolvido para a geração automática ou semi-automática de regras ativas.

5.1 Derivação Incremental de Regras Ativas para Manutenção de Dados Derivados

A partir de regras dedutivas definidas em Datalog (com recursão, predicados e negação estratificada), pode-se gerar um conjunto de regras para manutenção de dados derivados materializados ou não materializados. Para os dados derivados materializados, sempre que modificações na base afetam o valor de dados derivados, regras são disparadas e suas ações modificam as derivações, conforme necessário. Para os dados não materializados, regras executam uma avaliação iterativa usada em SDBs dedutivos.

Aqui, regras ativas podem ser usadas de forma natural para manter visões materializadas, no contexto de visões definidas por regras dedutivas. Esta é uma aplicação clássica de geradores de regras. O problema é manter a atualização das visões de forma a acompanhar as modificações realizadas na base. Uma solução simples e ineficiente é resubmeter a visão a cada modificação. Outra solução mais complexa é a manutenção incremental, que consiste em derivar das atualizações realizadas na base, as atualizações que

devem ser aplicadas a visão materializada, para manter a consistência com o novo conteúdo da base.

Nesta seção, a tradução de regras ativas a partir de regras dedutivas será apresentada através de exemplo. Em [CW94], a prova de corretude do método e a extensão dos resultados para dados derivados não materializadas podem ser encontradas. O exemplo a ser utilizado é composto de três tabelas:

Pedidos (N_pedido, Cliente, Status, Qtde)
Aviso_Bancário (N_pedido, Banco, Estado, Data)
Conta (Cliente, Banco)

Considerando a seguinte restrição de integridade: Não deve haver pedido acima de 1000 reais, atendido por um banco no Rio de Janeiro onde o cliente não possua conta.

Esta restrição é especificada formalmente em uma linguagem declarativa Datalog:

Pedido (N_pedido, Cliente, Status, Qtde),
Qtde > 1000,
Aviso_Bancário (N_pedido, Banco, Estado, Data),
Estado = "Rio de Janeiro",
not Conta (Cliente, Banco)

Intuitivamente, a especificação da restrição é avaliada contra a base de dados, de modo que as condições sejam satisfeitas. Conforme descrito em [CW94], a linguagem dedutiva utilizada obedece a interpretação dedutiva padrão [Ullm89].

Regras ativas podem ser utilizadas para manter visões materializadas no contexto de regras dedutivas. A visão Pedido_Crítico pode ser definida declarativamente:

Pedido_Crítico (N_pedido, Cliente, Qtde) <- Pedido (N_pedido, Cliente, Status, Qtde),
not Conta (Cliente, Banco),
Status != aprovado,
Qtde > 1000

Como exemplo do modo de operação deste método, considere a visão como materializada e verifique como mudanças em Pedido e Cliente podem ser propagadas para a visão.

Para propagar alterações, as regras ativas devem executar e aplicar visões *delta*: instâncias que devem ser inseridas ou removidas da visão. Visões *delta* são baseadas apenas nas instâncias inseridas ou removidas, fazendo com que seja possível lidar com uma quantidade de dados muito menor que a base total. Modificar visões a partir de *deltas* é chamado de manutenção incremental de visões [WC96].

Deltas são representados por termos como *inserted*, para instâncias inseridas na tabela/classe e *deleted*, para instâncias removidas na tabela/classe. Novas instâncias podem ser inseridas na visão quando pedidos são inseridos ou quando contas são removidas. As regras ativas a seguir lidam com estes casos:

```
create rule R1 for Pedido
 event insert
 condition inserted [Pedido (N_pedido, Cliente, Status, Qtde),],
            not Conta (Cliente, Banco),
            Status != aprovado,
            Qtde > 1000
 action insert into Pedido_Crítico: (N_pedido, Cliente, Qtde)
```

```
create rule R1 for Pedido
 event delete
 condition [Pedido (N_pedido, Cliente, Status, Qtde),],
            deleted [Conta (Cliente, Banco)],
            Status != aprovado,
            Qtde > 1000
 action insert into Pedido_Crítico: (N_pedido, Cliente, Qtde)
```

Estes exemplos ilustram a abordagem genérica de realizar manutenção incremental de visões através de regras ativas. Para este tipo de problema, as regras ativas relevantes podem ser geradas de forma automática a partir da definição da visão. Deve destacar que o exemplo aqui apresentado é mais complexo do que uma simples visão tradicional, pois inclui negação na definição da visão. Com isto, remoções na base são propagadas como inserções e vice-versa). Estas regras se tornam mais complexas com duplicidades de dados e visões recursivas conforme apresentado em [CW94].

5.2 Geração Automática de Regras Ativas para Manutenção de Integridade

A partir de especificações declarativas de restrições de integridade, uma ferramenta pode gerar regras ativas capazes de garantir a consistência da base de dados. Entretanto, a ação de *rollback* é uma solução simplificada que não utiliza completamente a potencialidade das regras ativas, uma vez que sua reação consiste apenas em descartar todo o trabalho executado durante uma transação. Em [BCP94] é descrita uma proposta para especificação declarativa de múltiplas estratégias de reparo. Em [CFPT94] a proposta é diminuir a quantidade de transações sofrendo *abort* através de geração automática de ações para um conjunto de restrições de integridade, a partir da definições destas restrições.

6. Um exemplo de uso de Regras Ativas e Dedutivas no Paradigma de Orientação a Objetos

Chimera é uma linguagem de BDs que integra um modelo de dados orientado a objetos, uma linguagem declarativa baseada em regras dedutivas e uma linguagem de regras ativas para processamento reativo [CM94, CFPT96]. Neste contexto, as regras dedutivas são utilizadas para definir os conceitos derivados, enquanto que as regras ativas definem o comportamento reativo do sistema além de métodos do modelo de dados.

Expressões declarativas são utilizadas em primitivas de consulta, regras dedutivas, restrições de integridade e declaração de visões. Já expressões procedurais são usadas em transações e (operações) métodos. Nas regras ativas Chimera, condições são expressões declarativas e ações são expressões procedurais.

A semântica de execução de regras utilizada em Chimera é orientada a conjuntos, onde regras são ativadas como efeito de detecção de eventos afetando múltiplas instâncias de objetos.

6.1 O Esquema Orientado a Objetos

O modelo de dados Chimera é um modelo orientado a objetos clássico com operações e herança, onde a assinatura de objetos não descreve apenas o estado (atributos[2]) e operações (métodos) para acessar e manipular instâncias de objeto, mas também restrições de integridade e regras ativas (*triggers*).

Um esquema pode ter visões, restrições e regras ativas que não fazem parte do contexto local de uma determinada classe. Para o escopo deste trabalho é relevante destacar que a definição do esquema na linguagem Chimera é uma coleção de definições locais (*targeted)* ou globais (*untargeted).* As primeiras são atributos e operações definidas em um determinado contexto possuindo escopo limitado, como uma classe. Já as definições derivadas são aquelas informações que não podem ser expressas no contexto de um tipo ou uma classe, como visões combinando informações de diversas classes, *triggers* afetando múltiplas classes e restrições relativas a estados de objetos de diversas classes.

A descrição detalhada do modelo de dados orientado a objetos do Chimera pode ser encontrada em [CM93]. De forma sucinta, pode-se dizer que objetos possuem identificadores únicos, há suporte para tipos e classes, sendo que a definição de classe com seus atributos e operações (métodos) é realizada junto com seus domínios.

Dois outros conceitos são relevantes e podem ser descritos tanto como locais ou globais: restrições de integridade e regras ativas. As restrições são condições declarativas que, quando associadas a uma classe podem restringir sua extensão ou os possíveis valores de seus atributos. Já quando são associadas a derivações, restringem o conjunto de estados válidos do BD. As restrições são verificadas após o comando de validação (*commit*), disparado em um transação. Usualmente, se elas forem violadas, a transação é desfeita. Porém, em [CFP94], os autores descrevem uma abordagem mais flexível onde restrições são codificadas por regras ativas cuja parte de ação é responsável por reparar as violações da restrição.

[2] Tanto extensionais quanto intensionais

6.2 Regras Dedutivas

Em Chimera, regras dedutivas são do tipo Datalog. Cada regra possui um termo atômico no lado direito e uma expressão conjuntiva no lado esquerdo. expressões declarativas podem apresentar negações, navegações e agregações; e regras devem ser seguras e estratificadas (com relação a negação e agregação). Regras dedutivas podem ser aplicadas nas seguintes situações:

- Atributos derivados: podem ter valores definidos por regras dedutivas ao invés de atualizações individuais de tempos em tempos;
- Classes derivadas: são as sub-classes no contexto de hierarquia. A população de uma sub-classe pode ser definida implicitamente através de propriedades da super-classe;
- Visões: unicamente definidas através de regras dedutivas combinando informação de uma ou mais classes Chimera.

A sintaxe básica das regras dedutivas é:

```
passive_rule ::= head "<-" body
head ::= atomic_formula
body ::= formula
formula ::= pos_or_neg_atomic_formula {"," pos_or_neg_atomic_formula }
pos_or_neg_atomic_formula ::= atomic_formula | "not" atomic_formula
```

A seguir, exemplos auto explicativos de regras dedutivas para:

- atributo:

```
Self.salary = 2000 <- engenheiro (Self), Self.idade < 35
```

- sub-classe:

```
engenheiro (X) <- empregado (X), X.profissão = "engenheiro"
```

- visão:

```
trabalha_para ((X,Y)) <- empregado (X), empregado (X),
                         Y = X.departmento.chefe
```

6.3 Regras Ativas

A seguir, as principais características das regras ativas Chimera estão resumidas. A completa especificação juntamente com o modelo de execução pode ser encontrado em [CM93].

Uma regra ativa pode ser dividida em quatro componentes: evento, condição, ação e prioridade. Além disto, as regras possuem características de destaque a serem detalhadas, como diferentes modos de processamento de eventos (*event consumption modes*), suporte opcional a composição de eventos (*net effect computation*) e acesso a estados intermediários durante uma transação. A definição de regras ativas Chimera (chamadas de *triggers*) possui a seguinte sintaxe [CM96]:

trigger_rule ::=
“define” [trig_option] “trigger” trig_name [“for”class_name]
“events” trig_events
“condition” condition_formula
“actions” reactions
[priority_option]
“end”

Conforme mencionado anteriormente, quando as regras ativas são definidas no contexto de uma única classe (com a cláusula “for”), elas são chamadas regras locais. Já no contexto de múltiplas classes, as regras são denominadas globais. Esta distinção é relevante para o projeto do esquema e para a modularização das regras, mas sintaticamente e semanticamente os dois tipos de regras são idênticos.

Eventos

A parte de eventos da regra ativa indica quais as operações primitivas serão monitoradas. Estas operações podem ser do tipo consulta, criação de objetos, atualização, remoção, migração na hierarquia de generalização, mudança no estado de persistência de objetos e qualquer mudança de estado em uma determinada classe. Além disto, estas regras podem monitorar operações (métodos) mesmo com a execução de regras sendo sempre orientada a conjuntos.

trig_events ::= event {“,” event}

event ::= “create” | ”create_tmp” | ”delete” | ”make_persistent” | ”generalize” “(“class_name”)” |”especialize” “(“class_name”)” | ”modify” [“(“attribute_name”)”] | ”query” [“(“attribute_name”)”] | “change” | operation_name

Condições

Condições são expressões declarativas compostas de fórmulas. Além de fórmulas atômicas (ex.: comparação), a parte de condições das regras ativas Chimera pode conter fórmulas de eventos e referências a estados antigos (*old*).

condition_formula ::= simple_condition_formula { “,”simple_condition_formula } | “true”

simple_condition_formula ::= [“not”] atomic_formula | [“not”] event_formula

atomic_formula ::= class_formula | type_formula | comparison_formula | membership__formula | choose_formula

Fórmulas de eventos são fórmulas especiais suportadas pela linguagem declarativa Chimera para inspecionar eventos que ocorreram durante uma avaliação e evitar avaliações desnecessárias.

event_formula ::= event_token “(“ event_list “,” variable_name[3] “)”

event_token ::= “holds” | “ocurred”

event_list ::= t_event { “,” t_event }

[3] Aqui, a variável de eventos possui valores re identificadores de objetos {IDOs} das instâncias afetadas por pelo menos um evento.

Estes predicados especiais são avaliados durante o teste da condição. Um predicado do tipo *ocurred* associa a variável de evento a todos os objetos afetados pelos eventos, enquanto que o predicado *holds* associa a variável ao sub-conjunto de objetos afetados. Este sub-conjunto é baseado na avaliação do *net effect*, a ser descrito na semântica de execução das regras.

Ações

Ações são seqüências de operações de BD, incluindo primitivas de atualização ou *display*, métodos ou comandos de transação. condições e ações podem compartilhar variáveis.

reactions ::= reaction {connector reaction }

reaction ::= display_cmd | for_each_cmd | select_cmd |create_cmd | create_tmp_cmd | delete_cmd | change_persistence_cmd | generalize_cmd | specialize_cmd | modify_cmd | population_cmd | operation_cmd | procedure_cmd | savepoint_cmd | rollback_cmd

Prioridades

Uma ordenação ente as regras pode ser definida para controlar sua seleção a tempo de execução. As especificações de precedência definem a ordem parcial das regras ativas, com a busca de não formação de ciclos a cada nova regra criada.

priority_option ::= [“before” trigger_list] | [“after” trigger_list]

trigger_list ::= trigger_name {“,” trigger_name}

Modos de processamento

As opções associadas a cada regra ativa indicam modos de consumo da regra e modo de execução da regra.

trig_option ::= [trigger_consumption] [trigger_execution]

trigger_consumption ::= “event_consuming” | “event_preserving”

trigger_execution ::= “deferred” | “immediate”

O “modo de consumo” da regra é relevante quando uma determinada regra ativa é considerada várias vezes em uma transação. Eventos podem ser consumidos depois da avaliação da regra, onde cada instância de um evento é considerada pela regra somente na sua próxima execução para em seguida ser descartada. Ou então, eventos podem ser preservados, isto é, todos os eventos desde o início da transação são considerados a cada execução da regra.

Para cada regra ativa é definido um modo de processamento, que pode ser do tipo imediato ou adiado. No primeiro, as regras são processadas ao fim da cada linha da transação, enquanto que o tipo adiado só é considerado após um *commit* ou *savepoint*.

Referência a Estados Prévios

A funcionalidade de referência a estados passados é definida pela função old, que pode ser aplicada a termos da condição, indicando que estes termos devem ser avaliados em estados anteriores da base de dados.

old_term ::= “old” “(“ var_or_iod_name [opt_attrib] “)”

O estado prévio é diretamente ligado ao “modo de consumo” da regra. Se for utilizado o modo de preservação, o estado prévio é referente ao início da transação. Todavia, se o modo for de consumo de eventos, o mesmo só vale para a primeira avaliação regra. Quanto as demais execuções, o estado prévio é o referente a última avaliação da regra.

Semântica de Execução das Regras

Enquanto a linguagem de regras Chimera define o que pode ser especificado em cada regra de BD, a semântica de execução de regras define como o SBD se comportará uma vez que um conjunto de regras foi definido.

Uma regra é ativada pela ocorrência de qualquer dos eventos definidos. O mecanismo de processamento de regras, executado ao final das linhas de uma transação e também através de comandos (commit e savepoint), selecionam regras com base sem suas prioridades. A regra escolhida é avaliada e se a condição for satisfeita, a reação definida será executada. Este processo se repete até um ponto fixo ser alcançado, o que acontece com o fim das transações.

A semântica de Chimera utiliza uma abordagem de dois passos: inicialmente regras são traduzidas para um formato interno e em seguida uma semântica operacional é fornecida através de algoritmos de execução de regras. A estratégia é codificar as diferentes opções semânticas das regras em sintaxe de baixo nível, permitindo modularidade.

A execução dos net effects é realizada da seguinte maneira. Uma seqüência que envolva primitivas de criação e remoção do mesmo objeto, com possíveis outras primitivas de modificação intermediárias, resulta em um efeito final nulo. Já uma seqüência de criação e modificação tem o efeito de uma única operação de criação. Várias modificações seguidas de uma remoção em um mesmo objeto removem o objeto.

Existem muitas possibilidades para execução semântica de regras e algumas vezes uma determinada semântica pode ser inconveniente. Em Chimera, o sistema de regras ativo trata este problema permitindo que o usuário faça escolhas dentre alternativas semânticas [CM93]. A semântica da avaliação dos *net effects* pode ser redefinida, devido a generalidade e extensibilidade da abordagem.

Regras e Transações

A execução das regras interage com as operações de transações da seguinte forma. Cada transação é subdividida em linhas. A execução inicia com as instruções da primeira linha e em seguida, uma primeira rodada de processamento da regra é executada (limitada a apenas

regras de modo imediato). Se o processamento de regras atinge um estado de quietude, então a segunda transação é aplicada a este novo estado. Este procedimento continua até a última linha de transação, quando finalmente a rodada de execução das regras adiadas é executada.

7. Conclusões

Este estudo apresentou a investigação de questões relevantes no contexto de regras em Bancos de Dados. Inicialmente, foram descritas propostas de integração de regras ativas e dedutivas em contextos genéricos. Em seguida, o paradigma de orientação a objetos foi utilizado como modelo de dados e linguagem, enriquecendo o problema em um ambiente de regras ativas e dedutivas.

Diversos outros trabalhos podem se beneficiar deste estudo. Destacam-se oportunidades de avaliações de desempenho mediante instalação de ambiente apropriado para uso da metodologia IDEA, disponível na Internet [4]. A utilização da linguagem Chimera em conjunto com um SGBDOO, permitirá avaliar as diferentes abordagens de utilização de regras e métodos em um mesmo ambiente.

Uma ferramenta que permitisse avaliar o impacto de extensões nas tabelas de *net effect* também seria bastante útil, pois permitiria que um projetista experiente definisse quais as operações não precisariam ser avaliadas. Uma abordagem interessante poderia seria considerar tabelas de *net effect* como locais às aplicações. Desta forma, até mesmo descrições de anulações semânticas poderiam ser feitas, como métodos com ações opostas.

Por fim, este estudo também é relevante para a importante tarefa de construção de ferramentas de auxílio ao projetista da aplicação, como por exemplo, uma ferramenta para avaliação dinâmica de regras.

Referências Bibliográficas

[BCP94] Baralis, E. Ceri, S. Paraboschi "Declarative Specification of Constraint Maintenance", *Proceedings of the International Conference on Entity-Relationship Approach*, Benjamin Cummings, Redwood City, California, 1992.

[BCP95] Baralis, E. Ceri, S. Paraboschi "Run-Time Detection of Non-Terminating Active Rule Systems", *Proceedings of the 4th International Conference on Deductive and Object-Oriented Databases*, Singapore, Springer-Verlag, pp. 38-54, dezembro 1995.

[BJ93] Bayer, P. Jonker, W. "A Framework for Supporting Triggers in Deductive Database", *Proceedings of the 1st International Workshop on Rules in Database Systems*, Edinburg, Scotland, Springer-Verlag, pp. 316-330, setembro 1993.

[4] http://www.txt.it/idea/

[C+90] Cacace, F. Ceri, S. Crespi-Reghizzi, S. Tanca, L. Zicari, R. "Integrating Object-Oriented Data Modeling with a Rule-based Programming Paradigm", *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 225-236, 1990.

[CBFP95] Ceri, S. Baralis, E. Fraternali, P. Paraboschi, S. " Design of Active Rule Application: Issues and Approaches", *Proceedings of the 4th International Conference on Deductive and Object-Oriented Databases*, Singapore, Springer-Verlag, pp. 1-18, dezembro 1995.

[Ceri92] Ceri, S. "A Declarative Approach to Active Databases", *Proceedings of the 8th International Conference on Data Engineering*, Arizona, pp. 452-456, 1992.

[CF97a] Ceri, S. Fraternali, P. Editores, "*Designing Database Applications with Objects and Rules - The IDEA Methodology*", Addison-Wesley, 1997.

[CF97b] Ceri, S. Fraternali, P. "The Story of the IDEA Methodology", *Proceedings of the 9th International Conference on Advanced Information Systems Engineering (CAiSE)*, pp. 1-17, 1997.

[CFP94] Ceri, S. Fraternali, P. Paraboschi, S. "Constraint Management in Chimera", *Bulletin of the Technical Committee on Data Engineering*, vol. 17, n. 3, pp. 4-8, junho 1994.

[CFP98] Ceri, S. Fraternali, P. Paraboschi, S. "The IDEA Web Lab", *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 587-589, 1998.

[CFPT94] Ceri, S. Fraternali, P. Paraboschi, S. Tanca, L. "Automatic Generation of Production Rules for Integrity Maintenance", *ACM Transactions on Database Systems*, vol. 19, n. 3, pp. 367-422, setembro 1994.

[CFPT96] Ceri, S. . Fraternali, P. Paraboschi, S. Tanca L. "Active Rule Management in Chimera" *em Widom, J. Ceri, S., editores, Active Database Systems - Triggers and Rules for Advances Database Processing*, Morgan Kaufmann Publishers, Inc., pp. 151-176, 1996.

[CFPW94] Ceri, S. Fraternali, P. Paraboschi, S. Widom, J. "Active Database Systems", *PUC-Rio DB Workshop on New Database Research Challenges*, pp. 35-52, setembro 1994.

[CHS92] Chakravarthy, S. Hanson, E. Su, S. "Active Data/Knowledge Bases Research At the University of Florida", ", *Data Engineering Bulletin, Special Issue on Active Databases*, v. 15, n. 1-4, pp. 35-39, dezembro 1992.

[CM93] Ceri. S. Manthey, R.. "Consolidated Specification of Chimera (CM and CL)", *Relatório Técnico IDEA.DE.2P.006.01*, ESPRIT Project 6333 (IDEA), Politecnico de Milano, Italy, novembro 1993.

[CM94] Ceri. S. Manthey, R. "Chimera: A Model and Language for Active DOOD Systems", *Proceedings of the 2nd East-West Database Workshop*, Klagenfurt, Austria, J. Eder Ed., pp. 3-16, setembro 1994.

[CR96] Ceri, S. Ramakrishnan, R. "Rules in Database Systems", *ACM Computing Surveys*, vol. 28, n. 1, pp. 109-111, março 1996.

[CTC98] Costa, G. Tanaka, A. Campos, M. "Ambiente de Projeto de Regras de Negócio em Bancos de Dados", *Anais do 13º Simpósio Brasileiro de Banco de Dados*, 1998.

[CW91] Ceri. S. Widom, J. "Deriving Incremental Production Rules for Incremental View Maintenance", *Proceedings of the 17th International Conference on Very Large Data Bases, Barcelona*, pp. 577-589, setembro 1991.

[CW94] Ceri. S. Widom, J. "Deriving Incremental Production Rules for Deductive Data", *Information Systems*, vol. 19, n. 6, pp. 467-490, 1994.

[DUHK92] Dietrich, S. Urban, S. Harrison, J. Karadimce, A. "A DOOD RANCH at ASU: Integrating Active, Deductive and Object-Oriented Databases", *Data Engineering Bulletin, Special Issue on Active Databases*, v. 15, n. 1-4, pp.40-43, dezembro 1992.

[EN94] Elmasri, R. Navathe, S. "*Fundamentals of Databases Systems*", Addison-Wesley, 1994.

[FWP97] Fernandes, A. Williams, M. Paton, N. "A Logic-Based Integration of Active and Deductive Databases", *Journal of New Generation Computing*, vol.15, n. 2, pp.205-244, 1997.

[HD93] Harrison, J. Dietrich, S. "Integrating Active and Deductive Rules", *Proceedings of the 1st International Workshop on Rules in Database Systems*, Edinburg, Scotland, Springer-Verlag, pp. 288-305, setembro 1993.

[Morg83] Morgenstein, M. "Active databases as a paradigm for enhanced computing environments", *Proceedings of the 9th International Conference on Very Large Data Bases*, 1983.

[MP91] Medeiros, C. Pfeffer, P. "Transformação de um banco de dados orientado a objeto em um banco de dados ativo", *Anais do 6º Simpósio Brasileiro de Bancos de Dados*, 1991.

[MST97] Melo, R. Silva, S. Tanaka, A. *"Banco de Dados em Aplicações Cliente-Servidor"*, Editora Infobook, Rio de Janeiro, 1997.

[MST98] Moura, A. Silva, A. Tanaka, A. "An Active Rule Language for an Object Management System", *Anais do 13º Simpósio Brasileiro de Banco de Dados*, 1998.

[NTC93] Navathe, S. Tanaka, A. Chakravarthy, S. "Active database modeling and design tools: issues, approach, and architecture", *IEEE Database Engineering (Quarterly), Special Issue on Active Databases*, janeiro 1993.

[SC93] Schiel, U. Carvalho, A. "TOM rules: A uniform and flexible approach to events, constraints and derived information", *Anais do 8º Simpósio Brasileiro de Bancos de Dados*, 1993.

[SK95] Simon, E. Kotz-Dittrich, A. "Promises and Realities of Active Database Systems", *Proceedings of the 21st International Conference on Very Large Data Bases*, Zurich, Switzerland, pp. 642-653, setembro 1995.

[SMT97] Silva, A. Moura, A. Tanaka, A. "Integrando o Comportamento Ativo a um Sistema de Banco de Dados Orientado a Objeto", *Anais da 23 ª Conferência Latinoamericana de Informática*, Valparaiso, Chile, 1997.

[Tana95] Tanaka, A. "Banco de Dados Ativos", *Anais do 10° Simpósio Brasileiro de Bancos de Dados*, 1995.

[TN94] Tanaka, A. Navathe, S. "Modeling and design of events and rules in databases", *Anais do 9° Simpósio Brasileiro de Bancos de Dados"*, 1994.

[TNCK91] Tanaka, A. Navathe, S. Chakravarthy, S. Karlapalem, K. "ER-R: an enhanced ER model with situation action rules to capture application semantics", *Proceedings of the International Conference on the Entity-Relationship Approach*, 1991.

[Ullm89] Ullman, J. "Principles of Database and Knowledge-Base Systems", Computer Science Press, Rockville, Maryland, 1989.

[WC96] Widom, J. Ceri, S. "Introduction to Active Database Systems*" em Widom, J. Ceri, S., editores, Active Database Systems - Triggers and Rules for Advances Database Processing*, Morgan Kaufmann Publishers, Inc., pp. 1-41, 1996.

[Wido93] Widom, J. "Deductive and Active Database Systems: Two Paradigms or Ends of a Spectrum?", *Proceedings of the 1st International Workshop on Rules in Database Systems*, Edinburg, Scotland, Springer-Verlag, pp. 306-315, setembro 1993.

[Wido94] Widom, J. "Research Issues in Active Database Systems", *Report from the Closing Panel at the Proceeding of the 4th International Workshop on Research Issues in Data Engineering: Active Database Systems*, Houston, Texas, pp. 30-38, fevereiro 1994..

[Zani93] Zaniolo, C. "A Unified Semantics for Active and Deductive Databases", *Proceedings of the 1st International Workshop on Rules in Database Systems*, Edinburg, Scotland, Springer-Verlag, pp. 271-287, setembro 1993.