Regras de Negócio para Análise em Ambientes OLAP

Pablo Lopes Alenquer

IM/NCE/UFRJ - Mestrado

Orientador: Maria Luiza Machado Campos
Ph.D. em Ciências da Computação

Universidade Federal do Rio de Janeiro
Instituto de Matemática
Núcleo de Computação Eletrônica

Julho/2002

Regras de Negócio para Análise em Ambientes OLAP

Pablo Lopes Alenquer

Dissertação submetida ao corpo docente do Instituto de Matemática e Núcleo de Computação Eletrônica (IM/NCE) – Universidade Federal do Rio de Janeiro – UFRJ, como parte dos requisitos necessários à obtenção do grau de Mestre.

Aprovada por:

_____ Orientador

Profª. Maria Luiza Campos – Ph.D.

_____

Prof. Geraldo Zimbrão – D.Sc.

_____

Prof. Ana Maria Moura – Dr.Ing.

Rio de Janeiro

2002

FICHA CATALOGRÁFICA

Alenquer, Pablo Lopes.

Regras de Negócio para Análise em Ambientes OLAP / Pablo Lopes Alenquer. – Rio de Janeiro, 2002.

viii, 146 f.: il.

Orientadora: Maria Luiza Machado Campos

Dissertação (Mestrado em Informática) - Universidade Federal do Rio de Janeiro – UFRJ, Instituto de Matemática – IM, Núcleo de Computação – NCE, 2002.

1. Regras de Negócio. 2. Análise em Ambiente OLAP.

RESUMO

ALENQUER, Pablo Lopes. **Regras de Negócio para Análise em Ambientes OLAP**. Orientadora: Maria Luiza Machado Campos. Rio de Janeiro: UFRJ, IM, 2002. Dissertação (Mestrado em Informática)

Ferramentas OLAP são exemplos de ambientes de suporte à tomada de decisão. Estas ferramentas facilitam o trabalho de análise sobre informações gerenciais por parte de analistas de negócios. Nestes processos de análise, os analistas fazem uso de conhecimentos que detêm acerca do comportamento do negócio. Este tipo de conhecimento é chamado de regra de negócio.

Porém, os ambientes de análise OLAP não evidenciam as regras de negócio que estão por trás dos processos de análise apoiados por eles. Normalmente, estes ambientes desconsideram a importância das regras de negócio nos processos de análise, pois esta seria uma atribuição do analista de negócios. Porém, isto pode comprometer a qualidade corporativa destes processos, pois, com o passar do tempo, o conhecimento que cada analista tem sobre as regras de negócio da corporação tende a se tornar mais informal, subjetivo, mal-documentado, ou mesmo equivocado.

Como solução para isto, esta dissertação propõe que o analista tenha acesso a regras de negócio. Além disto, ele deve poder interagir com estas regras no contexto de suas análises no ambiente OLAP. Para tanto, foi desenvolvido um protótipo de hipertexto que integra a apresentação regras de negócio para usuários e o ambiente OLAP utilizado.

ABSTRACT

ALENQUER, Pablo Lopes. **Regras de Negócio para Análise em Ambientes OLAP**. Orientadora: Maria Luiza Machado Campos. Rio de Janeiro: UFRJ, IM, 2002. Dissertação (Mestrado em Informática)

OLAP tools are examples of decision support environments. These tools facilitate the analysis process of management information for business analysts. On such analysis processes, analysts apply their knowledge about the business behavior. This kind of knowledge is known as business rules.

OLAP environments do not evidence business rules that are behind the supported analysis processes. Usually, these environments do not consider the importance of business rules on the analysis process, and count on the business analyst. However, this approach can jeopardize the quality of business processes. As time goes by, each analyst's knowledge about the organization business rules tends to become more and more informal, subjective, badly documented, or even mistaken.

To address this problem, this dissertation proposes that analysts should have access to business rules. Moreover, the analyst should be able to interact with these rules in the context of his/her analyses in the OLAP environment. To illustrate this approach a hypertext prototype was developed. This prototype provides business rules presentation both to users and to the OLAP environment.

CAPÍTULOS

SUMÁRIO

1. Introdução

No ambiente corporativo de empresas, é comum encontrar regras referentes ao comportamento dos negócios. Tais regras servem para que cada empresa tenha uma maneira padronizada de agir e reagir perante as situações cotidianas relacionadas ao negócio em que atuam. Estas regras são conhecidas como regras de negócio.

Freqüentemente, as regras de negócio têm que ser revistas, atualizadas e excluídas conforme a evolução dos conceitos que cercam a empresa. Embora muito regulares em décadas passadas, as regras de negócio têm se mostrado muito voláteis nos últimos anos por conta de mudanças nas condições do mercado, freqüentes fusões entre empresas e certas alterações legislativas. Em muitos casos, a capacidade de reação das empresas em relação a estas mudanças é crucial para sua sobrevivência no mercado.

Por estes motivos, é necessário que as empresas ampliem sua capacidade de adaptação a novas realidades de maneira rápida e eficiente. As tecnologias de informação têm desempenhado um papel importante no sentido de solucionar este tipo de problema. Os ambientes de apoio à decisão são sistemas de informação voltados para a simplificação da tarefa de análise de negócios em nível gerencial, e estão por trás de diversos casos de sucesso de gerência de informação em empresas.

Um exemplo deste tipo de ambiente corresponde às ferramentas OLAP. As ferramentas OLAP são sistemas voltados para a realização de consultas de usuários com perfil analítico e investigativo. O objetivo das ferramentas OLAP é colocar à disposição destes analistas as informações que são necessárias para que eles realizem suas análises da maneira mais livre possível. Em outras palavras, estas ferramentas provêm a possibilidade de um analista de negócio realizar consultas que julgue interessantes sem

a interferência de analistas de sistemas. Portanto, as ferramentas OLAP conseguem aliar simplicidade e força de expressão em benefício dos negócios.

Embora sejam necessárias para guiar processos de análise, regras de negócio não são explicitamente expostas em ambientes OLAP. Na verdade, tais ambientes têm como missão disponibilizar dados para consulta de usuários. As disciplinas de análise e interpretação dos dados são tidas como tarefas exclusivas dos usuários.

Realmente, é responsabilidade do analista de negócios emitir opinião sobre estas informações, mas existem regras de negócios que são aplicadas com maior freqüência, e por diversos usuários diferentes. Seria interessante que estas regras fossem difundidas e compartilhadas por estes usuários, pois isto melhoraria a qualidade corporativa das informações, facilitaria o aprendizado de analistas menos experientes, e também evitaria possíveis “esquecimentos” de algumas delas.

Outra razão para regras de negócio não aparecerem em ambientes OLAP (e talvez a mais determinante) é que elas não chegam a ser documentadas – quanto mais em formato digital. Desta forma, os ambientes OLAP não têm como prover o que sistematicamente não existe. Ou seja, antes de tudo, é necessário descobrir junto aos usuários quais são as regras de negócio que eles utilizam em seu dia-a-dia.

Além disto, mesmo que existissem regras à disposição, seria desejável que estas fossem expressas em linguagem formal, pois isto facilitaria seu tratamento computacional (por exemplo, em rotinas de recuperação de informação ou na confecção de agentes inteligentes). Para tanto, pressupõe-se a existência de linguagens (ou modelos) de definição de regras para expressá-las formalmente.

Embora existam iniciativas (ainda imaturas) neste sentido, a imensa maioria delas está direcionada para sistemas de informação tradicionais, conhecidos como sistemas OLTP, e não para sistemas de apoio à decisão, conhecidos como sistemas OLAP. A transição de um tipo de aplicação para o outro não é direta pois existem peculiaridades nas regras de negócio para OLAP que não são levadas em conta em sistemas OLTP – e vice-versa.

Apesar da expressão formal de regras de negócio trazer vantagens, este formalismo interessa muito mais aos desenvolvedores de sistemas de informação do que aos analistas de negócios. Na verdade, estes analistas necessitam que as regras sejam apresentadas da maneira mais amigável possível, de preferência usando termos de negócio que eles já conheçam. Em outras palavras, as linguagens formais carregam uma complexidade muitas vezes desnecessária aos analistas de negócios.

Enfim, a interação entre usuários e regras de negócios é uma qualidade muito desejada em ambientes OLAP, mas que, no entanto, é cercada de desafios do ponto de vista técnico e conceitual. Esta dissertação tem como objetivo propor uma solução para este dilema. Através de um estudo de caso sobre um ambiente OLAP específico, foi desenvolvido um protótipo de hipertexto visando apresentar regras de negócio para o usuário de maneira interativa com este ambiente.

O ambiente OLAP escolhido foi o ScanOne, uma aplicação OLAP voltada para análises na área do mercado varejista. Este domínio de aplicação é interessante pois contém diversos exemplos de regras de negócio utilizadas em práticas de análise.

Como estratégia de desenvolvimento da dissertação, foram estudadas tecnologias relacionadas a regra de negócio que vigoram hoje em dia. Como estas tecnologias são basicamente voltadas para aplicações OLTP, foi necessário adaptá-las

para aplicações OLAP. Foram estudadas as implicações destas adaptações. Em seguida, um exemplo de análise através do ScanOne foi discutido em detalhes, de modo a ilustrar como regras de negócio são utilizadas na prática. Por fim, é apresentado um protótipo de hipertexto alimentado por informações provenientes do próprio ScanOne, além, é claro, de regras de negócio associadas.

No capítulo 2, são apresentados conceitos relativos a regras de negócio: o que são, para que servem e como se expressam. Além disso, foi estudado como regras de negócio são normalmente definidas e tratadas em sistemas de informação típicos (OLTP). Além disso é apresentada a linguagem OCL (*Object Constraint Language*) como um formalismo suficientemente expressivo para a especificação de regras de negócio em linguagem formal.

Em seguida, no capítulo 3, apresentam-se aspectos relacionados ao ambiente analítico e ao processo de tomada de decisão. Em especial, foram abordados conceitos relacionados à ferramentas OLAP e os seus principais conceitos. Como exemplo e fundamento para a construção do protótipo, foi apresentado o modelo de dados que sustenta os conceitos do ScanOne. Além disso, foram discutidas algumas diferenças entre regras de negócio nos ambientes OLTP e OLAP.

Já o capítulo 4 contém um modelo de dados para regras de negócio que foi desenvolvido por um grupo de pessoas dedicadas à definições nesta área, o Business Rules Group (BRG). Este modelo apresenta fundamentos interessantes para a definição de regras, bem como idéias sobre como estas podem ser usadas em expressões em linguagem natural. Além disto, no final do capítulo, é realizada uma análise crítica em relação ao modelo.

No capítulo 5, é discutido um exemplo de análise OLAP e como regras de negócio são usadas na prática. O exemplo é detalhado passo a passo, de modo a identificar cada uma das regras de negócio de maneira cuidadosa.

Por fim, o capítulo 6 descreve o protótipo de hipertexto desenvolvido com o intuito de ilustrar como regras de negócio podem interagir com um ambiente OLAP de maneira interessante para usuário. Além disso, ilustra-se que regras de negócio podem aparecer tanto para analistas de sistemas quanto para analistas de negócios, se tratadas adequadamente para tal.

2. Regras de Negócio: Conhecimento X Tecnologia

Em qualquer empresa ou corporação, existem regras que definem como o seu negócio funciona. Essas regras podem abranger diversos assuntos pertinentes à empresa, como suas políticas, interesses, objetivos, compromissos éticos e sociais, entre outros (DAVENPORT, 1998). Estas regras são valiosas, pois de certa forma são uma representação da própria empresa. Na verdade, qualquer (re)formulação das estruturas e processos acarreta a (re)formulação de suas regras.

Uma correta avaliação da realidade que cerca a empresa é o que se deseja - e as regras fazem parte dessa realidade. Avaliar o desempenho do negócio requer o estabelecimento de objetivos a serem atingidos, estratégias para atingi-los, métricas para qualificar seu sucesso ou fracasso, planos de contingência, análise de riscos, entre diversos outros aspectos. Certamente, existem regras por trás dessas definições.

Porém, hoje em dia, os sistemas de informação utilizados como suporte a avaliação de desempenho de negócios não lidam com regras explicitamente. Isso significa que os usuários têm que tomar conhecimento das regras a partir de outro meio que não o sistema. Além disso, nenhuma ação que envolva o conhecimento das regras pode ser realizada automaticamente.

Que regras são essas, e como elas podem influir em processos decisórios? Como essas regras se expressam formalmente em sistemas de informação? Que tipo de extensão é necessária aos sistemas de suporte à decisão de hoje para lidarem com regras de maneira satisfatória? Essas são algumas questões que esse capítulo ajudará a elucidar.

2.1. As Regras do Ambiente Corporativo

Ambientes corporativos são repletos de regras. Por exemplo, existem regras para o comportamento individual das pessoas no local de trabalho – como proibição de fumo, trajes adequados, horários de almoço etc. Há também regras que dizem respeito ao comportamento do negócio propriamente dito – como regras de atendimento ao cliente, regras para aprovação de crédito etc. Há ainda regras para organizar a própria corporação – como regras de aposentadoria, regras para as férias e procedimentos para pedir reparos. E além desses, existem diversos outros tipos de regra dentro de ambientes corporativos DAVENPORT.

Independentemente do negócio em questão, de seu modelo de administração, das pessoas envolvidas, ou mesmo da tecnologia de suporte, sempre existem regras relevantes para o andamento do negócio. Com o tempo, acostumou-se denominar estas regras de “regras de negócio”, uma vez que elas contêm informações importantes para o andamento do negócio. As regras de negócio se confundem com a própria essência dos negócios, uma vez que elas são em si uma forma de expressão do negócio. Por causa disso, a existência das regras de negócio se confunde com a própria existência dos negócios. Logo, é errado pensar que algum novo modelo de negócio ou nova tecnologia possa substituir ou eliminar regras de negócio: elas existem desde que o Homem inventou seus primeiros negócios, existem até hoje e existirão para sempre.

Porém, as regras de negócio de um ambiente corporativo nem sempre são evidenciadas de maneira clara, formal ou são difundidas pela corporação de maneira homogênea. Existem regras formalmente definidas pela organização (como os horários de trabalho), mas existem também regras informais que podem surgir em resposta a

situações específicas. Em outras palavras, as regras podem ser explícitas ou tácitas, dependendo se a corporação tem conhecimento sobre estas regras.

Esta característica das regras de negócio é natural e inevitável, mas as corporações podem sofrer com o desconhecimento das regras de negócio tácitas. Na verdade, as regras de negócio fazem parte da memória organizacional da corporação (MALHOTRA, 1998), necessitando os mesmos cuidados que quaisquer informações dessa natureza.

Muitas corporações têm investido na padronização e integração de suas informações e processos. Com esse tipo de atividade, tem-se uma oportunidade interessante de evidenciar e reavaliar as regras de negócio da corporação. Essa tarefa não é trivial, visto que a documentação das regras de negócio não é uma prática generalizada nas corporações.

Assim sendo, não seria um absurdo imaginar que quanto mais formalizado for um processo corporativo, melhor será a documentação de suas regras de negócio. Dentre os processos corporativos, os mais formais são os representados em sistemas de informação, já que eles são definidos em linguagens formais rigorosas; e portanto menos suscetíveis a imprecisões e ambigüidades. Logo, os sistemas de informação deveriam fornecer regras de negócio com maior facilidade e naturalidade. Porém, a prática não confirma este raciocínio.

2.2. O Conflito Negócios versus Sistemas

Existe um conflito constante entre a gerência de um negócio e os sistemas de informação que suportam o negócio. Muitos desses problemas têm como ponto fundamental o tratamento computacional das regras de negócio.

O conflito mais comum é o descompasso nas velocidades de mudança das regras de negócio e suas representações computacionais. Hoje em dia, os negócios têm uma dinâmica muito maior do que tinham a 30 anos atrás. Existem diversas razões para os negócios mudarem. Por exemplo: lançamentos de novos produtos e serviços; parcerias, aquisições e fusões entre empresas; mudanças de legislação e/ou órgãos reguladores; alterações nas condições de mercado, como atividades da concorrência ou mudanças de hábito dos clientes (VON HALLE, 2002).

Com o aumento da concorrência, velocidade dos meios de comunicação e o maior grau de informação dos clientes, as corporações têm tido que fazer mudanças como essas em um ritmo muito mais acelerado que em décadas passadas. Em outras palavras, as regras de negócio têm que ser modificadas dentro de períodos de tempo relativamente curtos. Em conseqüência disso, tem-se que as soluções computacionais para os problemas dos negócios não conseguem acompanhar o ritmo de alterações a contento. Isso acontece principalmente por dois motivos (VON HALLE, 2002): 1) as regras de negócio não são separadas de aspectos de implementação dos sistemas; e 2) geralmente, os sistemas têm documentação muito deficiente.

O primeiro problema diz respeito à separação entre o problema e sua solução em termos do negócio, e os problemas e soluções tecnológicos associados (HARVARD BUSINESS REVIEW, 2001). Muitas vezes, soluções para problemas do negócio são empregadas por serem mais simples a curto prazo. Um analista de negócios deve se guiar pelas necessidades do negócio, não por conjunturas tecnológicas. Decisões mal pensadas podem gerar problemas no futuro que poderão ser tratados de maneira errada e provavelmente custarão mais caro do que poderiam. Enfim, se o analista de negócios não enxergar as regras de negócio separadas de suas implementações tecnológicas, ele possivelmente não tomará as melhores decisões para o negócio.

O segundo problema é que os sistemas têm documentação deficiente. Por causa disso, os sistemas de informação seguidamente se transformam em caixas-pretas cujo conteúdo é desconhecido. Como conseqüência, quando há necessidade de alterar alguma regra de negócio, os analistas de sistemas não têm condições de determinar precisamente quais são os impactos que essas mudanças causarão no sistema. Isso significa que existe um risco potencialmente alto de que as alterações gerem efeitos colaterais indesejados. Com o passar do tempo, após sucessivas mudanças (e muitos defeitos), é possível que os analistas de negócios comecem a desconfiar da acurácia dos sistemas de informação e a se perguntarem: será que o sistema está correto sob o ponto-de-vista do negócio? Essa crise de confiança pode levar ao surgimento de sistemas informais paralelos – o que é uma ameaça à qualidade corporativa das informações.

2.3. O Conceito de Regra de Negócio

> **regra**. [Do lat. *regula*, pela f. *regla*.] S. f. 1. Aquilo que regula, dirige, rege ou governa. 2. Fórmula que indica ou prescreve o modo correto de falar, de pensar, raciocinar, agir, num caso determinado: uma regra de gramática, de matemática; as regras de um jogo. 3. Aquilo que está determinado pela razão, pela lei ou pelo costume; preceito, princípio, lei, norma: as regras do bom senso, da boa educação; transgredir uma regra. 4. Estatutos de certas ordens religiosas. 5. Moderação; método, ordem: No trabalho ou no lazer, evita sair da regra.
>
> (FERREIRA, 1999)

A partir das definições acima, pode-se entender melhor a natureza da palavra regra. Pela definição nº 1, regra é “aquilo que regula, dirige, rege ou governa”. As definições subseqüentes indicam alguns exemplos do que pode ser “regulado, dirigido, regido ou governado” por uma regra: gramática, matemática, um jogo, o bom senso etc.

Disso conclui-se que o termo “regra” pode ser empregado em diversos contextos ou domínios diferentes, mas seu significado de “regular, dirigir, reger ou governar” permanece independente. É como se o conceito de regra fosse ortogonal aos diversos contextos em que ele pode ser empregado - e estes são muitos.

De maneira geral, diz-se que regras de negócio são “o que se usa para manter um negócio funcionando” (ROSS, 2000a). Embora correta, essa afirmação é imprecisa demais para ser utilizada de maneira prática – é necessária uma conceituação melhor. Isso, porém, não é tão simples quanto parece.

O conceito de regra de negócio tem sido definido e redefinido continuamente (ROSS, 2000a) (GOUGEON, 2001), ao mesmo tempo que o interesse sobre o assunto vem aumentando. Embora a maior parte dos autores da área apontem para uma direção comum, não existe consenso absoluto sobre o escopo, o público-alvo e a forma de expressão de regras de negócio. Essa seção visa esclarecer os pontos convergentes e divergentes sobre o assunto.

Entre os pontos convergentes (ROSS, 2000a) (DATE, 2000) (VON HALLE, 2002) (BUSINESS RULES GROUP, 2001) destacam-se os seguintes:

- **Regras de negócio são expressões declarativas**. Regras de negócio são expressas através de declarações. Declarações são expressões objetivas que visam comunicar a essência da regra da maneira mais clara possível. Dependendo do autor, o termo "declaração" é substituído por algum sinônimo: assertiva, proposição, sentença, frase etc - porém, trata-se do mesmo conceito.

- **Regras de negócio não são processos**. Conforme (DATE, 2000) resumiu muito apropriadamente em seu livro sobre o assunto, regras de negócio expressam ***o que*** deve ser feito, e não ***como*** deve ser feito (*what, not how*). Em outras palavras, a declaração de uma regra de negócio não deve conter informações sobre como a regra deve ser implementada ou executada. Um exemplo bem sucedido dessa estratégia de desenvolvimento é o SQL. A adoção do SQL como linguagem de definição e manipulação de sistemas gerenciadores de bancos de dados livrou os desenvolvedores de sistemas de preocupações sobre *como* as informações são armazenadas em disco, mas apenas sobre *o que* armazenar. Espera-se que benefícios semelhantes sejam obtidos utilizando o mesmo princípio em regras de negócio.

- **Regras de negócio não dependem de tecnologia**. Regras de negócio devem ser definidas sem comprometimento com tecnologias particulares. Dessa maneira, um analista de negócios pode observar as regras de maneira adequada às necessidades do negócio. Além disso, mudanças de tecnologia não afetam a definição da regra: uma implementação particular pode ser alterada sem prejuízo aos conceitos do negócio.

- **Regras de negócio são atômicas**. Uma regra de negócio não pode ser decomposta em mais regras de negócio. Isso significa que a informação presente em uma regra de negócio é uma informação essencial para o negócio, de modo que a falta de alguma parte da regra implica falta de informação.

Logo, ao lidar com regras de negócio, espera-se obter declarações explícitas sobre informações importantes para o funcionamento do negócio. Porém, como obter isso na prática? É preciso definir quais são os tipos de regras que interessam, quem exatamente está interessado nelas, como elas devem ser expressas, como podem ser modeladas. Essas questões são tratadas na subseções seguintes.

2.3.1. Regras de Negócio: O Que São?

O *Business Rules Group* (BRG) é uma organização formada com o objetivo de definir e padronizar os termos associados a regras de negócio. Esse grupo publicou um documento (BUSINESS RULES GROUP, 2001) no qual define-se regra de negócio em dois aspectos: sistemas de informação e o negócio propriamente dito.

Sob o ponto-de-vista de sistemas de informações, regra de negócio é “uma sentença que define ou restringe algum aspecto do negócio (...) sua intenção é manter a estrutura do negócio, ou controlar ou influenciar algum aspecto do negócio”. Nesse contexto, as regras de negócio dizem respeito aos “dados que podem ser cadastrados em um sistema de informação”.

Já sob o ponto-de-vista do negócio, regras de negócio são “diretivas que visam influenciar ou guiar o comportamento do negócio. Tais diretivas existem como suporte a políticas de negócio, formuladas em resposta a riscos, ameaças ou oportunidades”. Assim sendo, as regras atenderiam diretamente às pessoas que lidam com o negócio, servindo como guias em questões que possam aparecer.

Essas duas visões não são contraditórias, mas complementares. Na verdade, a visão de negócios e a de sistemas correspondem respectivamente às linhas 2 e 3 da Arquitetura de Sistemas de Informação criada por John Zachman ZACHMAN. Essa

arquitetura evidencia os diferentes níveis de comprometimento com tecnologia que as informações têm ao longo da construção de um sistema. A perspectiva de negócio (linha 2) indica como as regras de negócio são vistas sob a ótica do empreendimento, sem influência de sistemas ou tecnologias. Já sob a perspectiva de sistemas (linha 3), as regras são vistas sob o caráter dos sistemas, embora ainda sem compromissos com tecnologias ou implementações específicas.

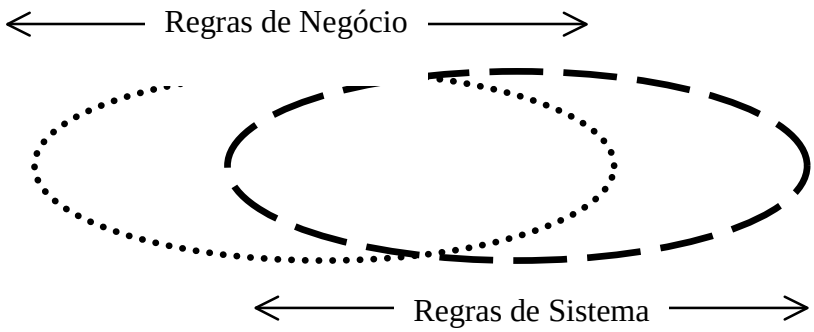

Figura 2.1. Relação entre regras de negócio e regras de sistema.

Uma razão pela qual existe confusão entre as duas perspectivas é que muitas regras interessam às duas perspectivas ao mesmo tempo, assim como há regras que interessam apenas a uma delas (Figura 2.1). Por exemplo, a regra que calcula o salário de vendedores comissionados interessa tanto à gerência de vendas quanto ao sistema de emissão de pagamentos: ambas as áreas precisam conhecê-la e concordar sobre ela. Já regras para otimização de consultas são foco exclusivo de sistemas de informação; enquanto regras para análise de eficiência promocional interessam apenas aos analistas de negócios.

Ross em (ROSS, 1998) propõe outra forma de entender regras de negócio: classificá-las segundo sua reação a algum evento - onde evento é uma alteração no

estado do sistema ou a execução de uma ação. Desse modo, regras de negócio são classificadas em:

- **Regras de Rejeição** (*Rejector*): são regras que rejeitam um evento porque ele conduz o sistema a um estado que não é permitido. As restrições de integridade em bancos de dados são exemplos desse tipo de regra.

  Ex1: “Uma conta não pode ter saldo negativo”.

  Ex2: “Um time não pode ter mais que 11 jogadores em campo”.

- **Regras de Projeção** (*Projector*): são regras que projetam o evento em outros eventos, ou seja, geram novas alterações no estado do sistema ou executam ações. Assim sendo, em sistemas de informação, essas regras alteram os fluxos de execução.

  Ex1: “Ao criar um cliente, deve ser criada uma nova conta para ele”.

  Ex2: “Se uma conta tiver saldo maior que 5.000, enviar folheto sobre investimentos”.

- **Regras de Produção** (*Productor*): são regras que não respondem a eventos, elas apenas definem informações do negócio. As Regras de Produção são sub-classificadas em Computações e Inferências.

  Ex1: “Saldo é igual a crédito menos débito”

  Ex2: “Uma conta é “Ouro Especial” se seu saldo atual é superior a 30.000 e seu médio saldo médio é superior a 20.000.”

Outras classificações bastante difundidas são usadas no modelo de dados do BRG (BUSINESS RULES GROUP, 2001), que é comentado em detalhes no Capítulo 4.

2.3.2. Regras de Negócio: Formas de Expressão

Como as regras de negócio devem ter expressão declarativa (no sentido de serem claras e objetivas), a forma de expressão das regras é fundamental para que elas sejam utilizadas devidamente. Existem dois pontos importantes a destacar sobre a forma de expressão de regras de negócio:

- Regras de negócio devem comunicar sua mensagem de maneira apropriada a seu público-alvo.

- Regras de negócio devem ser manipuláveis computacionalmente.

O primeiro ponto dita que uma regra de negócio deve ser clara e objetiva para aquele que vai usá-la. Isso é importante porque o usuário da regra de negócio pode ser um ser humano ou uma máquina – isso depende apenas da perspectiva da regra (conforme observado anteriormente). Logo, a comunicação adequada da regra de negócio depende de quão adequada é a linguagem de expressão da regra.

Por exemplo, analistas de negócios tendem a preferir frases em linguagem natural, usando termos conhecidos de seu dia-a-dia, ou ainda tabelas e gráficos que sintetizem facilmente a idéia da regra. Já analistas de sistemas tendem a usar linguagens formais, já que seu objetivo é chegar a estruturas e processos computacionais – muitas vezes a linguagem formal é a alternativa mais indicada

No fundo, essa classificação é muito simplista. Embora analistas de sistemas tenham capacidade de entender linguagens formais, isso não significa que eles não possam utilizar a linguagem natural. Por exemplo, é comum que diagramas de análise contenham notas de rodapé, e que linguagens de programação suportem comentários livres.

Em contrapartida, pode ser que o analista de negócios prefira utilizar linguagens formais em determinadas situações em que isso se mostre mais adequado. Por exemplo, ao definir uma regra de cálculo, uma fórmula matemática pode ser mais expressiva que a descrição da fórmula por extenso.

O importante é que as regras de negócio devem ser comunicadas de maneira simples, clara e objetiva. A forma de expressão mais adequada depende do tipo da regra, assim como do tipo de usuário que a utiliza. Além disso, uma alternativa não exclui a outra: linguagens naturais e formais podem ser combinadas, se isso facilita a comunicação da regra.

Dentre essas alternativas, a linguagem natural é a forma de expressão de regra de negócio considerada como a mais simples de todas. Isso se deve ao fato de que a linguagem natural é conhecida tanto pelos analistas de negócios, quanto pelos analistas de sistemas: logo, sua mensagem tem maior poder de difusão. Além disso, a linguagem natural tem grande amplitude de expressão. Porém, expressões em linguagem natural têm maior propensão a ambigüidades e imprecisões, ou seja, dão maior margem a interpretações alternativas, dúbias, ou até mesmo incorretas.

Por estes motivos, sistemas de informação são definidos em termos de linguagens formais: elas são precisas e objetivas. Porém, a capacidade de expressão dessas linguagens é muito mais limitada que a das linguagens naturais. Assim sendo,

muitas vezes frases que são simples em linguagem natural ficam complexas em linguagem formal. Por exemplo, a Figura 2.2 mostra a mesma expressão definida em uma frase em Português e em uma expressão de cálculo de predicados.

Ling. Natural	"Rui ganha um salário cujo valor é o mesmo do salário de Ana"
Ling. Formal	(∃ x, y: Ganha) (∃ z, w: Salário) (∃ s: Medida) (Pessoa(Rui) ∧ Pessoa(Ana) ∧ agente(x, Rui) ∧ tema(x, z) ∧ temValor(z, s) ∧ agente(y, Ana) ∧ tema(y, w) ∧ temValor(w, s))

Figura 2.2. Exemplo em que a linguagem natural é mais simples que a linguagem formal.

Em contrapartida, algumas vezes a linguagem formal comunica muito mais facilmente que a linguagem natural. A Figura 2.3 ilustra uma situação desse tipo.

Ling. Natural	"O salário de um vendedor é igual à soma de seu salário fixo com o total de vendas realizadas pelo vendedor no mês multiplicadas pelo percentual de comissão"
Ling. Formal	context Vendedor salário = salárioFixo + vendas(mês) * pctComissão

Figura 2.3. Exemplo em que a linguagem formal é mais simples que a linguagem natural.

Além disso, regras em linguagem natural podem ser ambíguas em algumas situações. Um exemplo disso está na Figura 2.4. A primeira frase não explicita que o que se requisita é um carro de um modelo, e não o modelo em si. Além disso, a frase é ambígua porque não se sabe ao certo se a data se refere a quando o carro deverá ser usado ou se é a data em que houve a requisição. A Figura 2.4 também ilustra uma versão mais clara para a mesma frase.

Frase com ambigüidade	"Um modelo de carro pode ser requisitado por um cliente em uma locadora de automóveis em uma data"
Frase sem ambigüidade	“Uma locadora de automóveis pode ser requisitada a alugar, em uma determinada data, um carro de um modelo particular para um cliente”

Figura 2.4. A mesma regra apresentada com e sem ambigüidades.

O segundo ponto sobre expressão de regras de negócio é que as regras devem ser manipuláveis computacionalmente. Isso significa que as regras de negócio devem ter uma estrutura computacional de modo a tornar possível armazenar e recuperar regras de maneira sistemática. Logo, é necessário que as regras de negócio sejam diferenciáveis umas das outras através de suas qualidades individuais, de modo que sejam classificadas dentro de um acervo de regras.

Por exemplo, dado um grande conjunto de regras de negócio devidamente documentadas, pode ser necessário encontrar “quais as regras que restringem o crédito dos clientes idosos?”. Se as regras estiverem em formato de texto livre (em linguagem natural), pode ser muito difícil dar uma resposta razoável a esta questão. Na verdade, isso é uma questão de recuperação de informação (*information retrieval*), que embora tenha avançado nas últimas décadas, pode não produzir resultados satisfatórios facilmente.

Dizer que as regras são manipuladas computacionalmente não significa que as regras serão necessariamente usadas para construir sistemas de informação, ou que as regras servirão a sistemas especialistas. Elas podem até ser usadas para esses propósitos, mas a questão aqui é prover utilidade prática a um grande conjunto de regras. Em outras palavras, as regras de negócio devem ter algum nível de transparência em relação a seu

conteúdo para que possam ser recuperadas sistematicamente. Isso passa pela exposição de metadados que descrevam formalmente as regras de negócio.

2.3.3. Templates para Regras de Negócio

Muitos autores (ROSS, 2000b) (VON HALLE, 2002) (MORIARTY, 2002) argumentam que uma boa maneira de lidar com regras de negócio é aliar a simplicidade das linguagens naturais à precisão das linguagens formais. Nesse sentido, uma solução interessante é o uso de *templates* para regras de negócio.

Um *template* é uma seqüência padronizada de termos usados para montar regras de negócio. Nesse padrão, existe uma certa estrutura de termos textuais que deve ser seguida. Alguns dos termos são fixos (não podem ser alterados) enquanto outros podem ser substituídos por outros termos quaisquer (seguindo alguma norma). Cada tipo de regra deve ter um *template* particular característico.

Ronald Ross e Barbara von Halle (BUSINESS RULE SOLUTION, 2001) (VON HALLE, 2002) definem diversos *templates* para cada tipo de regra. Seguindo a classificação de Ross (ROSS, 1998), esses são alguns exemplos de *template*:

- **Templates de Rejeição**
    - "Tem que" ("Must")
        - Um pagamento tem que ter um status
        - Um pedido tem que indicar seu cliente
    - "Não pode" ("Must Not")
        - Um novato não pode participar do clube honorário
        - O número de assentos de um curso não pode exceder 30

        -
        - Um funcionário aposentado não pode participar do conselho de funcionários

    - "Se ... Então ... Tem que ..." ("If ... Then ... Must ...")
        - Se um funcionário trabalha 12 meses, então ele tem que tirar férias

- **Templates de Produção**
    - " ... é igual a ..." ou "... = ..."
        - O custo de um produto é igual à soma dos custos de seus componentes
        - O salário de um vendedor é igual ao salário fixo mais o que ele conseguiu vender multiplicado pelo percentual de comissão
        - Salário = SalárioFixo + Vendas * Comissão

    - "... é considerado ... se ..."
        - Um cliente é considerado "de alto risco" se seu saldo médio é negativo nos últimos 2 anos
        - Um produto é considerado "caro" se seu preço é maior que R$1.000

- **Templates de Projeção**
    - "Se ... Então ..."

- Se um cliente apresentar cupom válido, então ele ganha 20% de desconto

2.4. Regras de Negócio e o Ciclo de Desenvolvimento

Todo processo de desenvolvimento de software segue um ciclo característico (PRESSMAN, 1995): inicia-se com idéias conceituais e abstratas sobre o que deve ser o sistema e, gradativamente, convertem-se estas idéias em estruturas e procedimentos computacionais que comporão o sistema propriamente dito.

Embora haja variações sobre a definição do ciclo de desenvolvimento, ele é classicamente definido como a seqüência das fases de análise, projeto e implementação. A fase de análise compreende definições puramente conceituais sobre o sistema. A fase de projeto serve para planejar a transformação dos conceitos definidos na fase de análise em estruturas computacionais de fato. Por fim, a fase de implementação compreende a construção efetiva dos componentes do sistema de informação previamente planejado.

As informações usadas no início de cada fase e as informações resultantes no final delas também é bastante diferente:

- **Análise**: os requisitos são definidos pelo usuário com ajuda do analista de sistemas. Muitas vezes, o resultado é definido em textos livres.

- **Projeto**: estes textos devem ser usados como fonte para o desenvolvimento de arquiteturas de software. O resultado desta arquitetura é muitas vezes representado através de linguagens gráficas formais ou semi-formais.

- **Implementação**: a partir das definições do projeto constróem-se estruturas e procedimentos reais usando linguagens de programação. O resultado é um sistema de informação completo.

O ciclo de desenvolvimento ajuda a entender o custo de manutenção dos sistemas. Sabe-se que quanto mais próximo da fase de implementação, mais custosas são as alterações, pois mais trabalho foi realizado sobre o conceito definido anteriormente. Logo, quanto mais cedo se identificam requisitos, menor é o custo de desenvolvimento.

O mesmo princípio vale para regras de negócio: quanto mais cedo elas são identificadas, menos custosa é a sua implementação. Uma vez que as regras de negócio são definições conceituais sobre o funcionamento do negócio, o momento mais adequado para sua captura é durante a fase de análise. Desta maneira, as regras de negócio podem evoluir seguindo o ciclo de desenvolvimento até se tornarem componentes do sistema de informação.

Extrapolando estes fatos, pode-se imaginar que o conflito entre analistas de negócio e analistas de sistemas descrito na seção 2.2 é, na verdade, um conflito entre a fase de análise e as fases de projeto e implementação do ciclo de desenvolvimento de software. Os analistas de negócios atuam na fase de análise, enquanto os analistas de sistemas atuam nas fases de projeto e implementação. O conflito entre negócios e sistemas vem justamente do alto custo de se descobrir e alterar requisitos conceituais quando já existem componentes de software construídos a partir de outros requisitos prévios.

Regras de negócio podem ser usadas para amenizar estes conflitos (VON HALLE, 2002). A razão disto é que regras podem ser expressas em linguagem natural e

também em linguagem formal . Isto é vantajoso porque, se os requisitos de sistema fossem definidos através de regras em linguagem natural, a transição destes requisitos para componentes em linguagem formal seria potencialmente mais suave do que são nos ciclos tradicionais.

Desta forma, os requisitos de sistema podem ser especificados pelos analistas de negócios em uma linguagem que lhes é familiar - garantindo-lhes expressão e documentação adequados. Em contrapartida, a transformação destas expressões de requisito em componentes reais de software seria mais suave: a distância entre as linguagens de especificação não seria tão grande, o que facilitaria não apenas o desenvolvimento, mas também a manutenção do sistema. Portanto, a tecnologia de regras de negócio pode trazer benefícios para o desenvolvimento de sistemas de informação.

Esta seção analisa diversos aspectos de regras de negócio em torno do ciclo de desenvolvimento de software. Inicialmente, serão abordadas questões relacionadas à regras de negócio na fase de análise, em seguida no projeto e, por fim, a implementação.

2.4.1. Regras de Negócio em Diagramas de Dados

Uma maneira usual de se definir um sistema de informação é através de diagramas. Por possuírem uma estrutura gráfica, os diagramas conseguem transmitir algumas idéias de maneira mais clara e direta do que através de textos ou linguagens formais. Além disso, em fase de análise e projeto de sistemas, a diagramação é uma forma de trabalhar os conceitos do sistema sem comprometimento com linguagens de programação ou SGBDs (Sistemas Gerenciadores de Bancos de Dados) em particular.

Existem diversos tipos de diagramas, cada um cobrindo um aspecto específico de sistemas. Por exemplo, diagramas entidade-relacionamento (E-R) são voltados para a definição de dados e suas associações; diagramas de transição de estados mostram como os valores dos dados podem ser modificados; diagramas de fluxo de dados mostram como as informações passam entre os processos realizados no sistema; diagramas de classe mostram como os objetos (dados + processos) se relacionam estaticamente; entre outros tipos de diagrama.

Porém, não há diagramas específicos para regras de negócio. Isso ocorre porque as regras de negócio podem se manifestar de diversas maneiras e em diferentes aspectos do sistema. Logo, é possível encontrar regras de negócio em diversos tipos de diagrama diferentes.

Porém, o maior problema que regras de negócio encontram em diagramas é que elas podem ser complexas demais para a sintaxe do diagrama. Por exemplo, a Figura 2.5 ilustra um pequeno diagrama E-R. O diagrama em si define as seguintes regras: “cada empregado trabalha em um único departamento” e “cada departamento tem pelo menos 1 empregado trabalhando”. Porém, um diagrama E-R não tem como expressar regras como: “se o departamento de sistemas tiver um empregado que seja analista, então este departamento tem que ter pelo menos outros dois empregados que sejam programadores, ou então um consultor externo”. No caso desta regra, a dificuldade de expressão em E-R está no fato dela conter restrições sobre instâncias de departamento e empregado: a sintaxe do E-R só permite expressar regras sobre as entidades e relacionamentos de maneira genérica.

Figura 2.5. Exemplo de modelo E-R.

Um outro problema desse tipo de diagrama é que eles não têm o mesmo poder de expressão da lógica de 1ª ordem. Assim sendo, regras que envolvam negação (não-lógico), o quantificador universal (para todo) ou a operação lógica de dijunção (ou-lógico) podem não ser expressáveis em E-R. Por exemplo, “se existir um gerente comercial no departamento de marketing então ou não pode haver analistas no departamento de mídia ou só pode haver gerentes plenos em finanças”.

Geralmente, os modelos de dados mais populares (como o E-R e as classes da UML) não expressam completamente a lógica de 1ª ordem, mas somente uma parte dela. Esta parte é conhecida como lógica existencial-conjuntiva, pois comporta apenas o quantificador existencial e a relação de conjunção (e-lógico).

Por causa desta limitação, diversas regras de negócio acabam se perdendo no momento da análise de um sistema simplesmente porque não podem ser capturadas adequadamente. No caso da UML, este problema é amenizado através de uma linguagem a parte especializada em descrever restrições: a Object Constraint Language (OCL). A OCL tem o poder da lógica de 1ª ordem, não é gráfica e serve de complemento a diagramas de classe UML para definir completamente a semântica das classes. Dessa maneira, as classes UML funcionam como um grande vocabulário a partir do qual frases em OCL podem ser escritas. A OCL é descrita em mais detalhes na seção 2.5.

2.4.2. Opções de Projeto para Regras de Negócio

Para implementar regras de negócio em sistemas de informação de maneira adequada, é preciso escolher o projeto mais conveniente para cada aplicação. Esta seção descreve algumas possíveis opções de escolha.

Uma escolha importante diz respeito à forma de implementação das regras de negócio. Segundo Ross, existem três formas de implementação de regras de negócio em sistemas de informação: **procedimentos manuais**, **procedimentos automáticos** e **bases de regras** (ROSS, 2000c).

Regras em procedimentos manuais são aquelas que necessitam intervenção humana para serem realizadas. Realmente, a prática demonstra que há muitas situações em que os procedimentos manuais são preferidos em relação aos automáticos (por exemplo, decisões sobre investimento financeiro). Outro exemplo disto aparece quando o procedimento manual é exigido pelos próprios requisitos da aplicação. A regra: "para aprovar um empréstimo muito alto é necessária a aprovação de dois gerentes". Neste caso, o procedimento automático é tecnicamente viável, mas opta-se pelo manual por uma questão de segurança.

Regras em procedimentos automáticos são aquelas que ficam programadas em procedimentos de sistemas de informação. Neste caso, as regras não têm limites definidos, sendo distribuídas ao longo dos procedimentos de maneira arbitrária. Esta distribuição desordenada é a principal causa dos problemas relacionados à manutenção de regras discutidos na seção 2.2 (regras em caixas-pretas, documentação deficiente etc). Esta é a alternativa de implementação de regras de negócio mais comumente encontrada em sistemas de informação do mercado de software.

Já o projeto de regras de negócio através de bases de regras explora a idéia de que regras de negócio devem ser tratadas como dados, e não como procedimentos. Seguindo este princípio, as regras têm estruturas de dados definidas a partir de um modelo ou linguagem de regras. Deste modo, as instâncias de regras de negócio podem ser mantidas em uma base de dados. As bases de dados que contêm regras de negócio são chamadas de bases de regras (LOSHIN, 2001).

Em contraste com os procedimentos automáticos, as regras em base de dados são explícitas e formalmente delimitadas, enquanto as dos procedimentos automáticos são implícitas e arbitrariamente distribuídas pelo sistema. Deste modo, regras em bases de regra apresentam custos de manutenção potencialmente mais baixos do que regras em procedimentos automáticos, conforme foi observado na seção (2.3).

Uma vez que existem bases de regras, elas podem ser processadas através de programas especializados, conhecidos como máquinas de regras (*rule engine*). Uma máquina de regras "recebe como entrada o conjunto de regras contidas na base de regras, cria um *framework* para a execução delas, e monitora estas execuções de modo a garantir que todas as regras sejam obedecidas" (LOSHIN, 2001).

Desta maneira, o procedimento de execução da máquina de regras é sempre o mesmo. Na verdade, o comportamento do sistema é determinado pelas regras contidas na base de regras, e não pelo procedimento da máquina. O algoritmo executado pelas máquinas de regras é sempre o seguinte:

1. Avaliar o estado corrente
2. Identificar as regras cujas condições são satisfeitas
3. Selecionar uma das regras para ser executada
4. Executar a ação desta regra
5. Voltar ao passo 1

Este procedimento de execução de regras é também é conhecido como Evento–Condição–Ação (ECA) (WIDOM, 1996). Neste procedimento, a ocorrência de um evento faz com que uma condição seja testada. Caso essa condição seja satisfeita, então uma ação é disparada em resposta ao evento correspondente.

Portanto, a opção de projeto através de bases de regras é mais próxima ao conceito de regra de negócio do que as outras opções. Por outro lado, há situações em que a máquina de regras não escolhe a melhor ordem para a execução das regras, resultando em performance insatisfatória. Se isto ocorrer, os desenvolvedores do sistema não têm como interferir diretamente na execução da máquina de regras para adequá-la aos seus interesses. Pode-se, no máximo, utilizar serviços de parametrização oferecidos pela própria máquina de regras, mas que não são obrigatórios.

Logo, a opção pelo desenvolvimento através de bases de regras tende a tornar os sistemas menos flexíveis a ajustes específicos que se mostrem necessários. Esta é uma desvantagem em relação aos procedimentos automáticos, nos quais a liberdade de interferência por parte dos desenvolvedores é total.

Date apresenta um outro critério de escolha para o projeto de regras de negócio: a camada de aplicação em que a regra deve se localizar. Segundo Date, qualquer sistema de informação tem que lidar com três aspectos distintos (DATE, 2000):

- **Aspectos de interface com o usuário** (camada de interface)
- **Aspectos do negócio propriamente dito** (camada de negócio)
- **Aspectos de banco de dados** (camada de banco de dados)

Cada dado ou procedimento que compõe o sistema de informação localiza-se em alguma destas camadas. Da mesma forma, cada implementação particular de uma regra de negócio deve estar contida em apenas uma destas camadas.

Deste modo, pode-se optar por colocar regras de negócio em qualquer uma destas camadas. Porém, cada opção pode levar a uma implementação diferente. Por exemplo, seja a seguinte regra de negócio: “Uma conta não pode ter saldo negativo”. Esta é uma regra de restrição, segundo a classificação de Ross (ROSS, 1998). Sua função é evitar que o valor de saldo de uma conta fique negativo quando este valor é alterado. Deste modo, o projeto desta regra pode ser realizado das seguintes maneiras:

- **Camada de interface**: quando o usuário modificar o valor de saldo de uma conta, um pequeno teste é realizado, validando o novo saldo segundo esta regra. Em outras palavras, a regra é implementada como resposta aos eventos relativos aos objetos de interface, e distribuída entre eles. Portanto, trata-se de um procedimento automático localizado na camada de interface.

- **Camada de negócio**: a própria estrutura de dados que mantém o valor de saldo não permite que ele seja negativo. Nesse caso, a regra faz parte da estrutura de dados e procedimentos que compõem o sistema propriamente dito, ficando distribuída entre estes elementos (dados e procedimentos). Portanto, trata-se de um procedimento automático localizado na camada de negócio.

- **Camada de banco de dados**: uma restrição no campo de saldo pode ser programada no banco de dados para que seja proibida a gravação de um valor de saldo quando este for negativo. Neste caso, a regra é declarada e explicitada como tal, sendo única e presente em um só lugar. Portanto, trata-

se de uma regra contida em uma base de regras e localizada na camada de banco de dados.

Logo, a mesma regra pode ter implementações formalmente diferentes, em cada um dos diferentes aspectos de aplicação: interface, negócio e banco de dados (DATE, 2000). Se uma implementação é melhor ou pior do que outra, isso depende do contexto da aplicação, mas o fato é que regras de negócio encontram-se comumente distribuídas entre os componentes do sistema, e muitas delas não podem ser identificadas e delimitadas no sistema.

2.4.3. Regras em Bancos de Dados

Bancos de dados são aplicações nas quais regras são muito utilizadas. Esta seção descreve algumas de suas características em relação a regras.

Um dos grandes objetivos de um sistema gerenciador de bancos de dados (SGBD) é garantir que as informações ali colocadas sejam íntegras em relação a alguma convenção. Geralmente, as convenções são determinadas através de regras definidas formalmente junto ao esquema do banco. Deste modo, o banco utiliza uma máquina de regras interna para garantir a execução das regras.

Todo SGBD baseia-se em um modelo de dados teórico. O modelo mais empregado por SGBDs hoje em dia é o modelo relacional. O modelo relacional tem este nome porque as informações são tratadas de maneira a evidenciarem as relações que trazem entre si. Os SGBDs relacionais obtiveram muito sucesso ao longo dos anos, entre outras razões, por oferecerem uma maneira consistente de fazer valer regras para garantir a integridade lógica das informações, e com uma performance razoável.

As regras de um SGBD relacional são geralmente definidas de maneira declarativa, usando SQL padrão ou uma variação especializada para o SGBD. Essa característica permite poupar o desenvolvedor de aplicações de manter as regras dentro de seus sistemas: elas são mantidas automaticamente pelo próprio SGBD.

As possíveis regras suportadas por um SGBD variam de um fornecedor para outro, mas em geral, pode-se dizer que os SGBDs têm restrições de integridade e gatilhos (triggers).

Restrições de integridade são regras que indicam ao banco de dados quais são as informações válidas em um determinado contexto. Existem diversos tipos de restrições de integridade, sendo que cada SGBD pode implementar alguns tipos e outros não. Gatilhos (*triggers*) são respostas programadas a eventos que ocorrem quando se atualiza alguma informação. Por exemplo, pode-se indicar ao banco de dados o que fazer quando se insere algum registro novo.

As restrições de integridade podem ser classificadas quanto ao escopos de atuação no banco (DATE, 2000):

- **Restrições de Domínio** (ou Tipo): especificam os valores permitidos para um determinado tipo de dado utilizado no banco de dados. Podem ser tipos definidos pelo sistema (como data e *string*), mas podem também ser tipos definidos pelos usuário. A maioria dos SGBDs relacionais do mercado suporta apenas a primeira alternativa.

- **Restrições de Coluna** (ou Atributo): indicam que valores são válidos para uma determinada coluna de um banco de dados. Por exemplo, a idade de uma pessoa não pode ser negativa.

- **Restrições de Transição**: indicam quais os valores de colunas válidas em uma transição de estados, ou seja, uma vez que uma coluna tenha um determinado valor, quais são os valores permitidos para ela.

- **Restrições de Tabela**: indicam restrições em um conjunto de relações como um todo. Por exemplo, pode ser proibido que salários caiam com o passar do tempo.

- **Restrições de Banco de Dados**: são a generalização das restrições de tabela sendo aplicadas no banco de dados como um todo.

Os gatilhos (*triggers*) são necessários para realizar ações quando ocorre alguma alteração no banco. Os gatilhos geralmente seguem a estrutura evento - condição - ação, onde a ocorrência de um evento (alteração no banco) causa a averiguação de uma condição que, se satisfeita, dispara uma ação correspondente. Neste contexto, os eventos são inserção, atualização e exclusão de informações.

2.4.4. Alternativas para Representação de Regras de Negócio

A representação computacional de regras de negócio é fundamental para o seu tratamento em sistemas de informação. Ainda não há uma forma de expressão padronizada para regras de negócio, mas há alternativas propostas. Segundo Ross (ROSS 2000b), alguns dos principais candidatos a formalismo para expressão de regras de negócio são os seguintes:

- **Lógica de Predicados**: é a base lógica para muitas linguagens formais comumente empregadas no mercado (SOWA, 2000). Sua origem vem da Matemática, tendo aplicações em diversas Ciências, além da Informática. Portanto, sua forma e poder de expressão são bastante difundidos. No

entanto, como sua origem é muito anterior ao surgimento do computador, sua forma de expressão não estimula sua utilização em sistemas de informação de maneira direta.

- **O Método Ross**: apresenta diversas classificações a partir de diversos critérios (BUSINESS RULE SOLUTION, 2001). A idéia é formar uma taxonomia abrangente para as regras de negócio, de modo a identificá-las mais precisamente. Esta alternativa pode ser interessante para uma definição inicial sobre as regras, mas pouco indica sobre a estrutura interna delas.

- ***Object Constraint Language*** **(OCL)**: linguagem de definição de restrições para diagramas UML (OBJECT MANAGEMENT GROUP, 1997). Trata-se de um complemento a UML gráfica, para aumentar seu poder de expressão. Utiliza estruturas de dados definidas em UML como vocabulário básico para então definir regras complexas.

Outra alternativa que merece destaque são as linguagens de expressão de regras em XML (THORPE, 2001). Estas linguagens utilizam um vocabulário XML para determinarem as regras de negócio. Desta forma, facilitam não apenas a definição de regras, mas também a sua capacidade exportação para outros sistemas. Thorpe destaca duas delas como sendo as mais significativas: a RuleML (*Rule Modeling Language*) e a BRML (*Business Rules Modeling Language*).

Porém, os modelos de dados nos quais estas linguagens foram baseadas é muito influenciado pela área de Inteligência Artificial, de modo que elas têm características que desestimulam seu uso em aplicações de negócios. Na verdade, o modelo de dados que está por trás das regras é mais importante que o vocabulário XML em si.

Para o desenvolvimento desta dissertação, optou-se pela OCL (*Object Constraint Language*). Esta decisão está baseada no fato desta linguagem ter um excelente poder de expressão, não necessitar definição de estrutura de dados (pois isto está a cargo da UML), e pelo fato dela ter expressão textual, e não apenas gráfica. A OCL é descrita em maiores detalhes na seção 2.5.

2.5. Object Constraint Language (OCL)

A *Object Constraint Language* (OBJECT MANAGEMENT GROUP, 1997) é uma linguagem formal usada para expressar restrições sobre diagramas de classes UML – *Unified Modeling Language*. A OCL surgiu como linguagem de modelagem de negócios na Divisão de Seguros da IBM. Mais tarde, ela foi adotada pelo OMG (OBJECT MANAGEMENT GROUP) para complementar a expressividade dos diagramas UML. Esta seção apresenta um resumo sobre a potencialidade e sintaxe da linguagem OCL.

2.5.1. Para que Serve a OCL?

A UML é uma linguagem gráfica criada para atender a descrição de sistemas de informação orientados a objetos em fase de análise e projeto (BOOCH, 2000). Para tanto, a UML contém uma série de diagramas diferentes, cada um especializado na descrição de um aspecto de sistema: há diagramas de classes, de estados, de interação, de colaboração, entre outros. Segundo (BOOCH, 2000), ao longo do processo de modelagem, a integração desses diagramas tende a prover qualidade e robustez ao sistemas construídos.

Porém, por ser uma linguagem gráfica, a UML apresenta limitações de expressividade. O diagrama de classes UML talvez seja aquele em que esse problema

mais se evidencie, já que esse é o principal modelo de dados da UML. Esse diagrama tem a função de determinar as relações lógicas entre as classes de objeto de maneira estática, ou seja, o que os objetos são e como se relacionam uns com os outros.

A questão é que a sintaxe dos diagramas de classe UML não é capaz de expressar certas relações que são importantes. Por não poderem ser expressas nesses diagramas, muitas dessas relações são perdidas ao longo do processo de modelagem. As que são documentadas muitas vezes são materializadas na forma de notas descritas em linguagem natural. Em conseqüência disso, muitas relações acabam sendo perdidas ou mal interpretadas na fase de projeto e implementação.

Para amenizar essa questão, foi criada a OCL. Ela é uma linguagem que especifica restrições que devem ser obedecidas pelas instâncias de um modelo de dados UML. A OCL tem poder de expressão equivalente à lógica de 1ª ordem, e por isso ela é mais expressiva que o diagrama de classes da UML, por exemplo.

2.5.2. Características Gerais da OCL

Uma expressão em OCL estipula uma relação que restringe os valores possíveis de objetos em modelos UML. Uma característica importante da OCL é que suas expressões não causam mudança de estado dos objetos, ou seja, uma expressão OCL não pode alterar valores de atributos de nenhuma instância de uma classe (*side-effect free*). No entanto, o OMG já admite incorporar esse tipo de característica na próxima versão da linguagem, ainda em discussão (OBJECT MANAGEMENT GROUP, 2001).

Uma expressão OCL pode ser de três tipos: invariante, pós-condição e pré-condição. Pré e pós-condições são restrições que se aplicam a métodos. Pré-condições são restrições que devem ser satisfeitas antes da execução do método. Pós-condições

são restrições que devem ser satisfeitas ao fim da execução do método. A restrição invariante vale para toda instância de um dado classificador, enquanto esta instância existir. Mais adiante veremos exemplos dos diferentes tipos de expressão OCL.

Cada expressão OCL é escrita no contexto de um esquema UML. Assim, uma expressão em OCL pode se referir a uma classe, tipo, interface ou associação. Uma expressão OCL pode também referir-se às propriedades de instâncias de classes: atributos, métodos, e ponta de associações.

Para fazer referência a uma instância deste contexto usa-se a palavra reservada *self*. No exemplo abaixo a classe *Companhia* é declarada como contexto de uma restrição do tipo invariante (*inv*), chamada *LimiteMinEmpregadosOk*, cuja expressão indica que o atributo *numeroEmpregados* de uma instância da classe *Companhia* deve exceder 50. Em outras palavras, uma companhia deve, invariavelmente, ter no mínimo 51 empregados.

```
context Companhia inv LimiteMinEmpregadosOk:
        self.numeroEmpregados > 50
```

Uma outra forma de expressar a mesma restrição é declará-la como pré-condição para a operação *demitir()*. No exemplo abaixo o método *demitir()* da classe *Companhia* é declarado como contexto de uma restrição do tipo pré-condição (*pre*), chamada *LimiteMinEmpregadosOk*, cuja expressão indica que o atributo *numeroEmpregados* de uma instância da classe *Companhia* tem que ter no mínimo 52 empregados. Em outras palavras, um desligamento da companhia só poderá ocorrer, se esta tiver no mínimo 52 empregados.

```
context Companhia::demitir(p : Pessoa) : Boolean
        pre LimiteMinEmpregadosOk: self.numeroEmpregados > 51
```

Ainda uma outra forma de expressar a mesma restrição é declará-la como pós-condição para a operação de desligamento da companhia. No exemplo abaixo o método *demitir()* da classe *Companhia* é declarado como contexto de uma restrição do tipo pós-condição (*post*), chamada *LimiteMinEmpregadosOk*, cuja expressão indica que o atributo *numeroEmpregados* de uma instância da classe *Companhia* tem que ter no mínimo 51 empregados. Note que a palavra reservada *result* é usada para representar o resultado da operação, que no exemplo é do tipo *Boolean*.

```
context Companhia::demitir(p : Pessoa) : Boolean
 post LimiteMinEmpregadosOk:
   result = self.numeroEmpregados > 50
```

É possível ter acesso ao valor anterior de uma propriedade ou atributo referenciado em uma restrição do tipo pós-condição. Para se referir ao valor anterior à execução da operação do contexto, usa-se a palavra reservada '@pre' como sufixo da propriedade ou atributo referenciado. O exemplo abaixo mostra a restrição *ficarMaisVelho,* onde a expressão compara o atributo *idade* com seu valor anterior em *idade@pre*.

```
context Pessoa::fazerAniversario()
   post ficarMaisVelho: idade = idade@pre + 1
```

2.5.3. Tipos Básicos da OCL

A OCL oferece alguns tipos básicos para compor expressões, bem como as operações básicas sobre eles. Entre os tipos básicos mais usados podemos citar: *Integer, Real, Boolean,* e *String*. Algumas das operações sobre os tipos básicos são: +, -, *,/, *abs(), floor(), toUpper(), concat()* etc. Por convenção, as classes (tipos) da OCL são iniciados com letra maiúscula, enquanto os atributos, métodos e associações são iniciados com letra minúscula.

Para facilitar a escrita de expressões mais complexas, a OCL permite nomear “pedaços” de expressão através da cláusula *let*. Esta cláusula pode estar inserida nos três tipos de restrição, podendo definir variáveis, métodos, etc., que são válidos apenas no escopo da restrição em que são definidos. Essas expressões nomeadas funcionam como “macros” substituíveis na expressão principal. O exemplo a seguir mostra a restrição *semDevolucao* que inclui a definição de uma variável chamada *impostoDevido*, usada neste caso para “explicar” o cálculo feito sobre o atributo *salário*. Embora o exemplo seja simples, em uma restrição mais complexa, o uso de variáveis pode permitir o reuso de expressões, facilitando o entendimento da restrição.

```
context Pessoa inv semDevolucao:
let pctImposto : Real = self.cargo.pctImposto
let impostoDevido : Real = self.salario * pctImposto
  self.impostoPago <= impostoDevido
```

Uma restrição pode conter tipos globais, que valem para um conjunto de outras restrições. Para isso é preciso declarar uma restrição do tipo definição, um quarto tipo de restrição. Este tipo é especial, pois além de estar associado a um contexto, pode conter somente cláusulas *let*. As definições contidas nesta restrição só serão válidas em restrições de mesmo contexto, funcionando como uma extensão deste contexto. No exemplo abaixo, a variável *impostoDevido* passa a ser vista como um atributo a mais da classe *Pessoa*. O método *salarioPeriodo* também estende o comportamento da classe *Pessoa*.

```
context Pessoa def:
let impostoDevido : Real = self.salario * 0,15
let salarioPeriodo (meses : Integer) : Real = self.salario * meses
```

Todo objeto referenciado pelas expressões OCL apresenta algumas propriedades comuns que facilitam a sua manipulação. Entre as operações mais usadas destacam-se *oclIsTypeOf* e *oclIsKindOf*. A operação *oclIsTypeOf* permite identificar o tipo de uma instância (*self*), enquanto a operação *oclIsKindOf* permite identificar o tipo ou supertipo

de uma instância. Estas operações são especialmente úteis quando o contexto da restrição é uma classe abstrata e o tratamento deve ser diferenciado para cada subtipo.

```
oclIsTypeOf(t : OclType) : Boolean
oclIsKindOf(t : OclType) : Boolean
```

O tipo *Collection* (coleção) é um dos tipos pré-definidos OCL mais úteis. Na verdade o tipo *Collection* é um tipo abstrato que é especializado em três subtipos: *Set*, *Sequence* e *Bag*. Um *Set* é uma coleção onde não ordenação e nem elementos duplicados. Um *Bag* é uma coleção que permite duplicatas, mas não tem ordenação. Já um *Sequence* é uma coleção ordenada que pode ter duplicação. Os exemplos abaixo mostram como declará-los:

```
Set { 1 , 2 , 5 , 88 }
Set { 'apple' , 'orange', 'strawberry' }
Bag { 1, 3, 45, 2, 3 }
Sequence { 'ape', 'nut' }
Sequence { 1..(6 + 4) }
Sequence { 1..10 }
Sequence{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }
```

2.5.4. Navegação em OCL

Expressões OCL podem se valer do poder de expressão do modelo OO. Assim, é possível navegar pelas classes de um diagrama de classes UML, usando seus atributos, métodos e associações entre classes. No exemplo abaixo, a restrição *gerenteVálido* mostra uma referência ao atributo *estáDesempregado* da classe *Pessoa* associada à classe *Companhia* através do atributo *gerente*. Neste caso, a multiplicidade da relação entre as classes *Companhia* e *Pessoa* é 0..1, por isso foi possível tratar uma única instância da classe *Pessoa*, e seus atributos e métodos. Já a restrição *companhiaVálida* mostra a referência ao atributo *empregado*. Neste caso, como a multiplicidade da associação entre as classes *Companhia* e *Pessoa* é 0..n, um *empregado* corresponde a uma coleção de instâncias da classes *Pessoa*. Uma coleção é um dos tipos pré-definidos

em OCL que possuem algumas propriedades que podem ser referenciadas através do símbolo ‘->’. A restrição *companhiaVálida* faz referência à propriedade *notEmpty()* da coleção *empregado* obtida na navegação a partir da classe *Companhia.*

```
context Companhia
  inv gerenteVálido: self.gerente.estáDesempregado = false
  inv companhiaVálida: self.empregado->notEmpty()
```

Um outro exemplo mostra o uso da propriedade *size()* que retorna o número de instâncias associadas à instância da classe declarada no contexto. Neste exemplo, a restrição expressa o número máximo de instâncias associadas a uma instância de *Pessoa*, através da associação *employer*. Em outras palavras, uma pessoa possui no máximo 2 empregadores.

```
context Person inv:
  self.employer->size() < 3
```

Em uma associação de multiplicidade 1 as associações também podem ser tratadas como conjuntos de um único elemento. Assim, as propriedades das coleções também podem ser referenciadas. No exemplo abaixo, o atributo *gerente*, que representa uma associação de multiplicidade 1, foi tratado como uma coleção, e associado à propriedade *size()* através do símbolo ‘->’.

```
context Compania inv:
  self.gerente->size() = 1
```

Esta flexibilidade quanto ao tratamento de associações de multiplicidade 0..1 como coleções é especialmente útil quando é preciso verificar a existência ou não de uma instância associada nas expressões OCL. Por exemplo, a seguinte expressão testa se uma Pessoa tem marido ou esposa, antes de verificar se este tem pelo menos 18 anos de idade.

```
context Pessoa inv:
  self.esposa->notEmpty() implies self.esposa.idade >= 18 and
```

```
self.marido->notEmpty() implies self.marido.idade >= 18
```

Ao navegar por uma auto-associação, é preciso indicar a direção em que se navega. Em OCL, esta indicação é feita através do papel origem da associação. No exemplo abaixo, o atributo *hierarquia* representa uma auto-associação entre instâncias da classe *Pessoa*. Para uma instância de *Pessoa*, a propriedade *sum()* sobre o atributo *hierarquia* pode informar a soma de seus empregados ou a soma de seus superiores. Para identificar a coleção que se deseja referenciar, foi preciso indicar a direção da associação, no exemplo, *empregados*.

```
context Pessoa inv:
  self.hierarquia[empregados]->sum() > 0
```

2.5.5. Operações sobre Coleções

A manipulação de coleções conta ainda com algumas operações: s*elect, reject, collect, forAll* e *exists*. A operação *select* permite selecionar as instâncias de uma coleção que satisfaçam uma dada condição. No exemplo abaixo, a coleção de empregados de uma companhia é referenciada através da navegação a partir do atributo *empregado* de uma instância da classe *Companhia*. Neste ponto, usa-se a operação *select* sobre a coleção de instâncias da classe *Pessoa* que atendem à condição '*idade > 50*'.

```
context Companhia inv:
self.empregado.select(p : Pessoa | p.idade > 50) ->notEmpty()
```

A operação *reject* é similar a operação *select*, selecionando somente as instâncias que não satisfazem a uma dada condição. No exemplo abaixo, ao trocar as operações, obtemos exatamente o conjunto de instâncias complementar ao conjunto obtido no exemplo anterior.

```
context Companhia inv:
```

```
self.empregado.reject(p : Pessoa | p.idade > 50) ->notEmpty()
```

A operação *collect* funciona como a operação algébrica de projeção, selecionando somente os atributos que são de interesse. No exemplo abaixo, as instâncias da classe *Pessoa* são reduzidas aos valores do atributo *aniversario*.

```
context Companhia inv:
  self.empregado.collect(aniversario)->notEmpty()
```

No exemplo anterior, a coleção resultante permite repetições, sendo portanto do tipo *Bag*. Para evitar repetições neste caso, basta acrescentar a propriedade *asSet()*, como mostra o exemplo que se segue.

```
context Companhia inv:
  self.empregado.collect(aniversario)->asSet()->notEmpty()
```

Ao expressar uma restrição, às vezes é necessário validar todas as instâncias de uma coleção, a respeito de uma dada condição. A operação *forAll* é usada nestes casos. Uma outra situação onde esta operação é útil ocorre quando os elementos de uma coleção precisam ser comparados aos pares, i.e., para cada elemento, este precisa ser comparado a todos os outros elementos da coleção. O exemplo abaixo mostra uma restrição onde a operação *forAll* é usada para comparar empregados de uma companhia e reforçar que não é permitido existirem homônimos.

```
context Companhia inv:
self.empregado->forAll( e1, e2 : Pessoa |
  e1 <> e2 implies e1.nome <> e2.nome)
```

Ao expressar uma restrição, às vezes é necessário verificar se existem instâncias que respeitam uma dada condição. A operação *exists* é usada nestes casos. O exemplo abaixo mostra uma restrição onde a operação *exists* é usada para reforçar que em uma companhia devem existir empregados estagiários.

```
context Companhia inv:
  self.empregado->exists( categoria = 'estagio' )
```

2.5.6. Organizando Expressões OCL

As restrições OCL podem estar organizadas por pacote UML (*package*). Para evitar ambigüidades com relação a classes de mesmo nome pertencentes a pacotes diferentes, é possível declarar restrições OCL de acordo com seu contexto. O exemplo abaixo mostra como declarar um *package*, e as restrições cujo contexto pertence a este *package*.

```
package Package::SubPackage

context X inv:
... alguma restrição invariante ...

context X::operationName(..)
  pre: ... alguma pré-condição ...

endpackage
```

3. Regras de Negócio para Tomada de Decisão

Um dos desdobramentos mais interessantes dos sistemas de informação corresponde aos sistemas de apoio à decisão. Os sistemas de apoio à decisão têm o objetivo de ajudar executivos em cargos de gerência a consultar, interpretar, compreender e até prever o comportamento do negócio em seus aspectos de interesse. Isto significa que os sistemas de apoio à decisão são ferramentas de análise gerencial voltadas para a realidade dos negócios. Enfim, são ferramentas utilizadas por analistas de negócios.

Não é difícil imaginar que regras de negócio tenham que fazer parte deste processo de tomada de decisão de alguma maneira. E realmente, as regras de negócio estão lá – o problema é que a tecnologia atual deste tipo de sistema não as usa ou as exibe explicitamente: o analista de negócio tem que conhecê-las de antemão. Isso acontece porque as tecnologias vigentes de sistemas de tomada à decisão são excessivamente focadas em fornecer dados para os usuários – e não as regras que controlam os dados.

Porém, uma vez que as regras de negócio fazem parte dos negócios é interessante que estas apareçam no contexto de tomada de decisão de alguma forma. Para tanto, é preciso adaptar as tecnologias de tomada de decisão para que as regras possam ser corretamente tratadas.

Este capítulo constrói os primeiros passos na direção de uma integração entre sistemas de apoio à decisão e regras de negócio. Para tanto, as principais tecnologias de apoio à decisão são estudas – especialmente as ferramentas OLAP. Em seguida, apresenta-se o modelo de dados da ferramenta OLAP Maestro, que fundamenta o estudo

de caso proposto mais adiante. Daí, discutem-se as diferenças entre as regras de negócio em sistemas de tomada de decisão e os sistemas tradicionais (OLAP e OLTP respectivamente), bem como enumeram-se as características típicas das regras de negócio em ambientes OLAP.

3.1. OLTP versus OLAP

Sistemas de apoio à decisão têm características muito diferentes de outros sistemas encontrados em corporações. O objetivo destes sistemas é facilitar a tarefa de gerência de negócios através de dados, enquanto os sistemas comuns da empresa têm a tarefa de manter o próprio negócio funcionando.

Os sistemas de informação voltados para a execução do negócio propriamente dito são conhecidos como sistemas transacionais (ou operacionais). Eles têm como foco a operação de transações sistemas específicos de interesse do negócio (operações de fábrica, distribuição em pontos de venda, controle financeiro, gerência de recursos humanos etc). Estes sistemas são conhecidos como *On Line Transaction Processing* (OLTP), pois sua execução atende a transações sobre as informações do negócio.

Já os sistemas de apoio à decisão são conhecidos como On Line Analytical Processing (OLAP) pois sua função não é executar transações sobre dados do negócio, mas analisá-los. Enquanto os sistemas OLTP têm seu fluxo determinado por procedimentos definidos computacionalmente, os sistemas OLAP têm seu fluxo definido pelo próprio analista de negócios; o fluxo depende de decisões que o analista toma durante sua análise. A Figura 3.1 resume as principais diferenças entre sistemas OLTP e OLAP (THOMSEN, 1997).

Característica	**OLTP**	**OLAP**
Freqüência de execução	Maior	Menor
Previsibilidade	Maior	Menor
Quantidade de dados por consulta	Pequena	Grande
Tempo do dado associado à consulta	Tempo corrente	Passado, presente e projeções do futuro
Derivações complexas	Poucas (se alguma)	Muitas

Figura 3.1. Características de OLTP versus OLAP.

3.2. Tecnologias para Apoio a Decisão

Os sistemas de apoio à decisão tendem a cruzar informações provenientes de diversos sistemas diferentes. Geralmente, estas informações são relacionadas através de cálculos complexos e, possivelmente, através de vários pontos no tempo. Os sistemas OLTP não possuem uma estrutura de dados que comporte este tipo de requisito: eles são projetados para atender às suas necessidades específicas. Destes requisitos, surgiram tecnologias como os *data warehouses* e as ferramentas OLAP.

3.2.1. Data Warehouse

O conceito de *data warehouse* (DW) surgiu como solução para suporte a sistemas de apoio à decisão. Segundo Hackartorn (HACKARTORN, 1997), o DW tem como objetivo prover "uma imagem única da realidade do negócio", ou seja, integrar os dados de uma empresa de modo a viabilizar a análise destes dados sob uma ótica administrativa. *Data warehouses* são construídos a partir de dados extraídos organizadamente de sistemas transacionais, de modo a comporem uma estrutura de informação mais apropriada para gerência e tomada de decisão.

Inmon é responsável pela definição clássica de *data warehouse* (INMON, 1992):

> *"Data warehouse é uma coleção de dados orientada por assunto, integrada, variante no tempo e não volátil que tem por objetivo dar suporte aos processos de tomada de decisão."*

Esta definição cobre diversas características dos ambientes de apoio à decisão que evidenciam as diferenças entre esses ambientes e os ambientes transacionais. A primeira delas indica que o DW é orientado por assunto. Isso significa que o DW deve ser projetado visando os aspectos mais gerais da empresa. Esta visão contrapõe-se à dos ambientes operacionais nos quais vigoram a visão funcional dos dados.

Além disso, o DW também deve ser integrado, ou seja, o DW não deve apresentar inconsistências na representação dos dados. Por exemplo, é possível que um mesmo nome de atributo seja utilizado em dois sistemas de produção dos quais são extraídos dados para o DW. Como cada sistema não conhece a existência do outro, a coincidência de nomes não causa problemas para eles. Porém, para formar o DW, é preciso que os dois atributos sejam corretamente diferenciados

Outro problema típico na integração de sistemas é a codificação dos dados. Por exemplo, campos que definem o sexo de uma pessoa podem ser codificados de várias maneiras: “M” e “F”, “m” e “f”, “0” e “1”, “true” e “false” etc. Existem infinitas possibilidades de codificação, e cada sistema pode usar uma própria. O DW deve conter apenas uma forma de codificação desses dados, independente de onde eles venham.

Por fim, um último problema típico da integração de dados é a diferença de unidades nos valores dos dados. Cada sistema transacional pode utilizar uma unidade de medida diferente para uma mesma variável que tenham em comum. Para integrar estas informações, o DW deve convencionar uma única unidade de medida para cada variável que integra.

O DW também deve ser variável no tempo. Isto significa que cada registro da base deve ser referente a um único momento no tempo. A motivação disto é fornecer uma visão histórica dos dados. É interessante notar que a historicidade dos dados não depende apenas da manutenção dos valores em relação ao tempo, mas também da manutenção do significado dos dados (ou seja, seus metadados). Isto é dito porque as regras de negócio podem (e devem) se alterar com o passar dos anos. É importante para o processo de análise de dados que o significado preciso dos dados seja preservado junto com eles.

A última característica do DW é a sua não volatilidade. Esta característica diz respeito ao fato do DW não possuir o mesmo caráter transacional dos sistemas operacionais. As informações armazenadas no DW não são alteradas ou destruídas com o passar do tempo. Esta característica resulta em diferenças no projeto físico do banco de dados em comparação aos sistemas tradicionais, pois o intenso controle de transações torna-se desnecessário. No DW, basta existir uma carga inicial de dados; não existem atualizações. Por este motivo, otimizações típicas de projetos transacionais - como a normalização de tabelas - perdem importância e dão lugar a questões mais críticas para o DW, como o desempenho das consultas.

3.2.2. Interfaces com Usuário Final

A definição de Inmon (INMON, 1992) dita que o *data warehouse* "tem por objetivo dar suporte aos processos de tomada de decisão". Embora a definição estipule esse objetivo, não há nada nela que indique como transformar as informações presentes no DW em soluções práticas para o dia a dia dos negócios.

A tecnologia de *data warehouse* não é capaz de realizar este objetivo sozinha. Ao observar a literatura e a cultura que cercam o conceito de *data warehouse*, vê-se que este está envolto numa atmosfera extremamente técnica. Assuntos comumente relacionados a DW são modelagem, projeto e implementação de grandes bases de dados, bem como os processos de extração e limpeza das informações oriundas dos sistemas de produção. Estes são assuntos muito interessantes para os analistas de sistemas que desenvolvem soluções de DW, mas estão fora do escopo de atuação dos analistas de negócios. A missão do analista de negócios é transformar as informações presentes no DW em benefícios concretos para a corporação, e não saber como a tecnologia de DW funciona.

Daí surge uma lacuna perturbadora entre o DW (processo técnico de integração de informações) e o analista de negócios (especialista na busca de estratégias e soluções práticas para o cotidiano empresarial). Preencher essa lacuna significa estabelecer uma ponte através da qual as informações presentes no DW façam sentido no contexto profissional do analista de negócios. Isto significa que é preciso criar uma **interface** entre o usuário e o DW que seja adequada aos interesses de análise de dados sob a ótica prática dos negócios.

Diferentes alternativas já foram desenvolvidas para solucionar esta questão. Duas das mais importantes são o **DSS (*Decision Support System*)** e o **EIS (*Executive Information System*)**. Os DSS são sistemas construídos para solucionar problemas de tomada de decisão bastante específicos. De maneira geral, o usuário DSS precisa alimentar o sistema com alguns parâmetros usados para a realização de cálculos. É necessário que o usuário possua conhecimento sobre o procedimento interno executado pelo sistema porque a escolha correta dos parâmetros depende disso. Portanto, o uso de DSS exige um alto grau de especialização do usuário em relação ao problema analisado e ao próprio DSS.

Já os sistemas EIS são relatórios dinâmicos gerados a partir de alguns parâmetros – como o DSS – mas sem exigir conhecimentos sobre o funcionamento interno do sistema. Os EIS são como visões parametrizadas através das quais os analistas podem comparar o comportamento dos diversos valores de parâmetros possíveis para um determinado aspecto do negócio. Os EIS são mais fáceis de utilizar que os DSS pois não desgastam os usuários com muitos detalhes técnicos, de modo que eles fiquem livres para a análise da informação propriamente dita.

Apesar de obterem performance satisfatória e atenderem a muitos problemas práticos, ambas as alternativas apresentam um grande problema: são inflexíveis a mudanças estruturais (THOMSEN, 1997). Tanto os DSS quanto os EIS são sistemas de informação nos quais as estruturas de dados e os processos executados são estáticos. Quando o analista deseja investigar algum aspecto do negócio que não tenha sido anteriormente estipulado, ou mesmo observar os dados em uma disposição diferente, é necessário alterar o sistema através de uma linguagem de programação – requisito extremamente técnico e obscuro para o analista de negócios. Assim, o analista tem seu desejo por informação tolhido e condicionado à intervenção de técnicos em Informática. Portanto, pode-se afirmar que DSS e EIS não dão plena autonomia para o usuário em relação ao seu processo de análise.

Como contraponto à inflexibilidade desses sistemas, tornou-se muito popular uma outra alternativa: a **planilha eletrônica**. As planilhas eletrônicas são tabelas bidimensionais nas quais pode-se inserir vários tipos de informação de maneira simples e conveniente aos mais diversos interesses. Através de planilhas, muitas atividades relacionadas à administração e cálculos matemáticos são bastante simplificadas. As planilhas permitem a realização de cálculos através da definição de fórmulas, bem como a construção de pequenos programas para a validação e consolidação de dados. As planilhas possuem uma linguagem simples através da qual o usuário final pode construir pequenos modelos de dados rapidamente. Por essas características, as planilhas se tornaram uma das interfaces homem-computador mais conhecidas do mundo: é virtualmente impossível encontrar um analista de negócios que não seja familiarizado com sua sintaxe e uso.

Apesar disto, existem diversas razões pelas quais as planilhas eletrônicas não são a interface analista-DW mais adequada para o processo de tomada de decisão. Uma

dessas razões é que as planilhas não têm conexão direta com o DW. Esta ligação exige um grande conhecimento técnico para ser estabelecida e o usuário final não tem condições de fazer isso sozinho. Um outro motivo é que muitos modelos para tomada de decisão são complexos demais para a linguagem da planilha. Isso faz com que as planilhas fiquem muito grandes e complexas, demandando grande esforço do usuário tanto para construir quanto para entender as planilhas durante sua análise.

Porém, o grande argumento contrário às planilhas eletrônicas como interface usuário-DW é que a flexibilidade das planilhas não é plena. A linguagem das planilhas facilita a construção de modelos pelo usuário final a partir do zero, mas ela não lida muito bem com mudanças de estrutura. Isso pode dificultar o processo de tomada de decisão, pois é muito comum o usuário querer observar dados em perspectivas diferentes daquelas que já estão ali. Neste sentido, as planilhas eletrônicas apresentam graus de inflexibilidade semelhantes aos de DSS e EIS.

Após estas considerações, pode-se vislumbrar algumas características desejáveis em relação ao processo de análise do usuário final. Em primeiro lugar, é importante que o analista de negócio tenha liberdade e facilidade para escolher os dados que ele deseja analisar bem como o formato no qual eles devem ser visualizados. Este requisito remete à necessidade de consultas ad hoc usando o DW como fonte de informação. Em segundo lugar é necessário que o usuário consiga realizar suas consultas de maneira simples e intuitiva, pois um usuário típico não tem o perfil de um técnico em Informática. Por este motivo, consultas através de linguagens formais no estilo de SQL não se constituem numa alternativa razoável. Em terceiro lugar, é preciso que a linguagem de consulta seja poderosa o suficiente para que o analista consiga extrair informações realmente úteis para o negócio. Isso é dito porque de nada adiantariam os esforços de construção do DW se a capacidade de análise de seus dados fosse limitada.

Existe um conflito quase contraditório entre a segunda e a terceira considerações: como aliar a simplicidade para o usuário à robustez para consultas e análises? A solução para esse dilema reside em um (meta)modelo de dados conhecido como **modelo dimensional**. Através do modelo dimensional, o usuário tem liberdade para visualizar diversos aspectos do negócio sob a perspectiva que entenda ser a mais interessante para sua análise. A simplicidade do modelo dimensional reside na nítida distinção entre dois conceitos básicos e complementares: as **dimensões** e as **variáveis**. A amplitude de análise deste modelo reside no fato de que um grande número de aspectos do negócio pode ser expresso em termos destes dois conceitos.

A utilização do modelo dimensional como linguagem fundamental para a comunicação entre o usuário e o DW é um dos principais alicerces de **OLAP (*On Line Analytical Processing*)**. A grande diferença entre OLAP e as outras alternativas de interface usuário-DW é que OLAP visa dar ampla autonomia para o usuário em relação ao processo de análise. O modelo dimensional é essencial para isto porque é através dele que o usuário consegue realizar análises complexas, mas de maneira simples.

3.2.3. O Modelo Dimensional

O modelo dimensional é o alicerce conceitual de OLAP. É ele que viabiliza o ideal de liberdade do usuário final em relação ao processo de análise. O modelo dimensional está baseado na idéia de que qualquer aspecto do negócio pode ser descrito em termos de alguns parâmetros independentes entre si. Em outras palavras, o modelo dimensional dita que todos os dados (**variáveis**) podem ser expressos em função de parâmetros (**dimensões**).

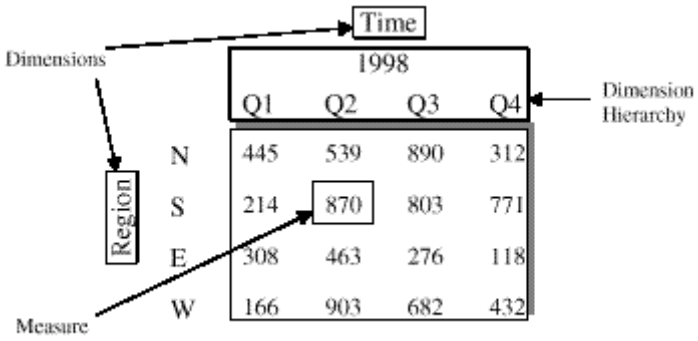

Figura 3.2. Exemplo de modelo dimensional.

Por exemplo, ao analisar o *Volume de Vendas* (variável) de uma loja, espera-se encontrar um número que indique quantas unidades foram vendidas. Porém, este número não pode ser obtido a menos que se indiquem algumas referências, por exemplo: qual é o *Produto* (dimensão) em questão; o período de *Tempo* (dimensão) em que esse valor está sendo medido; qual é o *Local* (dimensão) em que as vendas se realizaram. Portanto, as dimensões *Produto*, *Tempo* e *Local* podem ser usadas para construir um cubo com os valores da variável *Volume de Vendas*, conforme ilustrado na Figura 3.2.

No exemplo anterior, o desenho final é um cubo pois o número de dimensões utilizadas é 3, mas este número pode ser maior, dependendo do modelo. Por exemplo, pode-se desejar que, além das dimensões já citadas, o volume de vendas dependa ainda do *Canal de Distribuição* e do *Tipo de Cliente* - isto varia com a realidade que está sendo modelada. Quando o número de dimensões é maior que 3, não é mais possível obter um desenho de cubo representando o *Volume de Vendas* em relação a suas dimensões. Porém, é possível abstrair a idéia do desenho e “imaginar” as formas desse cubo. Este “desenho abstrato” recebe o nome de **hipercubo**, por se tratar de um cubo

que excede os limites espaciais. Porém, por conveniência, é comum as pessoas utilizarem o termo **cubo** como sinônimo de hipercubo (THOMSEN, 1997).

Um aspecto interessante do modelo dimensional é que (em geral) as variáveis contêm valores numéricos, em contraponto às dimensões, que (em geral) possuem valores não numéricos. Esta não é uma regra rígida, pois as variáveis não são obrigatoriamente numéricas, nem as dimensões são obrigatoriamente não numéricas. Porém, esta situação é bastante comum, e traz consigo algumas características muito úteis para o processo de análise.

Por exemplo, enquanto a variável *Volume de Vendas* corresponde a um número, a dimensão *Tempo* pode assumir valores como “Semana de 03/08/98”, “Jan/99” ou “13:45 AM”; a dimensão *Local* pode ser “Tijuca”, “Rio de Janeiro” ou “Brasil”; e a dimensão *Produto* pode conter os valores “Geladeira”, “Mesa”, “Guaraná”. De maneira geral, os valores que dimensão pode assumir são chamados de **categorias da dimensão** – ou melhor, as categorias são as instâncias das dimensões. Assim, cada combinação envolvendo uma categoria de cada dimensão corresponde a um determinado valor no cubo. A Figura 3.3 mostra um exemplo dessa situação.

Tempo	Local	Produto	Volume de Vendas
Semana de 03/01/99	Tijuca	Geladeira	5
Semana de 03/01/99	Tijuca	Mesa	2
Semana de 03/01/99	Tijuca	Guaraná	1.573
Semana de 03/01/99	Rio de Janeiro	Geladeira	21
Semana de 03/01/99	Rio de Janeiro	Mesa	12
Semana de 03/01/99	Rio de Janeiro	Guaraná	23.452
Jan/99	Tijuca	Geladeira	23
Jan/99	Tijuca	Mesa	9
Jan/99	Tijuca	Guaraná	6.017
Jan/99	Rio de Janeiro	Geladeira	54
Jan/99	Rio de Janeiro	Mesa	51
Jan/99	Rio de Janeiro	Guaraná	46.809

Figura 3.3. Os valores de uma variável a partir das categorias de cada dimensão.

Cada dimensão pode conter uma grande quantidade de categorias. Existem situações em que o número de categorias pode chegar à ordem de dezenas de milhares. A fim de facilitar a prática de análise, é comum que as categorias de cada dimensão sejam organizadas de forma hierárquica. Essas hierarquias são conhecidas como **hierarquias de dimensão**. Por exemplo, seja o seguinte conjunto de categorias da dimensão *Local*: “Tijuca”, “SP”, “Rio de Janeiro”, “Copacabana”, “MG”, “São Paulo”, “RJ”, “Brasil” e “Niterói”. A Figura 3.4 mostra uma possível hierarquia para essa dimensão.

Figura 3.4. Exemplo de hierarquia de uma dimensão.

Esta hierarquia organiza as diferentes categorias de *Local* através de uma relação de continência: a localidade de nível mais alto é aquela que contém as localidades de nível mais baixo. Implicitamente, esta hierarquia definiu uma organização para as categorias em que cada nível representa um tipo de detalhamento da informação (Figura 3.5).

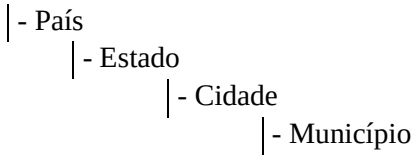

Figura 3.5. Exemplo de como a hierarquia se organiza.

Hierarquias de dimensão são um dos principais instrumentos de análise de dados dimensionais. Isto é dito porque as hierarquias permitem que o analista aprofunde a observação das informações, buscando informações mais detalhadas acerca de algum aspecto do negócio. Essa operação de detalhamento da informação é conhecida como drill-down.

Por exemplo, o analista pode estar interessado no *Volume de Vendas* obtido no "Ano de 1999". O número lhe parece pequeno demais, de modo que o analista decide fazer um drill-down na dimensão *Tempo*, para que possa observar o *Volume de Vendas* em cada mês contido no "Ano de 1999" – ou seja, as categorias de meses ("Jan de 1999", "Fev de 1999" etc) correspondem ao nível hierárquico imediatamente inferior ao nível de "Ano de 1999" na dimensão *Tempo*. Desse modo, o analista pode claramente observar que o Volume de Vendas foi menor nos primeiros meses do ano, e se lembra que houve muitos problemas na economia do País naquele período.

Muitas vezes, o valor de uma variável em um nível da hierarquia da dimensão corresponde à soma dos valores da variável nos níveis imediatamente abaixo. Quando isso acontece, o valor do nível superior é chamado de **agregação** dos valores do nível inferior. Por exemplo, o Volume de Vendas no ano de 1999 é uma agregação dos valores de Volume de Vendas de cada mês de 1999. Apesar de ser a situação mais comum, nem toda variável é agregável em uma dimensão. Por exemplo, se uma variável contém valores percentuais, sua adição pode não fazer qualquer sentido.

O fato do modelo dimensional possuir poucos elementos (variáveis, dimensões e hierarquias) o torna ideal para a realização de processos de tomada de decisão uma vez que o usuário não é sobrecarregado com muitos conceitos novos. Além disso, os conceitos do modelo são simples e aderem facilmente ao cotidiano do usuário. Em

contraposição a modelos de dados mais comumente empregados em sistemas de informação (como os modelos entidade-relacionamento e orientado a objeto), o modelo dimensional é mais simples porque apresenta uma estrutura uniforme. Dessa maneira, o modelo dimensional é mais adequado ao processo de análise de usuários finais que os modelos ER ou OO (KIMBALL, 1997).

3.2.4. Ferramentas OLAP

Ferramentas OLAP são sistemas que atuam como interfaces entre os analistas que realizam consultas e data warehouses. Como o objetivo das ferramentas OLAP é permitir análises para analistas de negócios, elas têm requisitos bem particulares se comparadas a outros sistemas de informação (THOMSEN, 1997):

- **Liberdade**. É importante que o analista de negócio tenha liberdade e facilidade para escolher os dados que ele deseja analisar bem como o formato no qual eles devem ser visualizados. Esse requisito remete à necessidade de consultas ad hoc usando o DW como fonte de informação.

- **Simplicidade**. É necessário que o usuário consiga realizar suas consultas de maneira simples e intuitiva, pois um usuário típico não tem o perfil técnico em Informática. Por esse motivo, consultas através de linguagens formais no estilo de SQL não se constituem numa alternativa razoável.

- **Expressividade**. É preciso que a linguagem de consulta seja poderosa o suficiente para que o analista consiga extrair informações realmente úteis para o negócio. Isso é dito porque de nada adiantariam os esforços de construção do DW se a capacidade de análise de seus dados fosse limitada.

- **Velocidade**. As consultas realizadas devem ter um tempo de resposta relativamente baixo. Isso significa que um usuário não deve esperar horas pelo resultado de uma consulta. Ferramentas OLAP são interativas, no sentido que elas respondem às consultas dos analistas de negócios, que usam seus resultados para decidir que consultas realizar adiante. Projetar sistemas para que essas consultas sejam rápidas não é uma tarefa simples uma vez já que as consultas são complexas e envolvem uma grande quantidade de dados.

Existe um conflito quase contraditório entre o segundo e o terceiro requisitos: como aliar a simplicidade para o usuário à expressividade nas consultas e análises? A solução para esse dilema reside em um (meta)modelo de dados conhecido como **modelo dimensional**. Através do modelo dimensional, o usuário tem liberdade para visualizar diversos aspectos do negócio sob a perspectiva que entenda ser a mais interessante para sua análise. A simplicidade do modelo dimensional reside na nítida distinção entre dois conceitos básicos e complementares: as **dimensões** e as **variáveis**. A amplitude de análise desse modelo reside no fato de que um grande número de aspectos do negócio pode ser expresso em termos desses dois conceitos.

Através do modelo dimensional, as ferramentas OLAP conseguem prover grande poder de autonomia para seus usuários; em outras palavras, as ferramentas OLAP oferecem maior independência dos analistas de negócios em relação aos analistas de sistemas. Antes das ferramentas OLAP, quando o analista de negócios desejava realizar alguma consulta nas bases de dados necessitava que o analista de sistemas construísse uma fórmula (*query*) para acessar a informação ali presente. Quando algumas consultas se tornavam muito comuns, elas podiam passar para dentro de

sistemas de informação construídos pelos analistas. Porém, qualquer modificação, ou pequena adaptação tinha que passar pelo analista de sistemas.

3.3. O Modelo Dimensional do Maestro

O estudo de caso proposto mais adiante visa ilustrar como regras de negócio podem ser úteis em análises OLAP. O caso em questão é fundamentado no modelo de dados de uma ferramenta OLAP em particular: o Maestro.

O Maestro é uma ferramenta para desenvolvimento rápido de EIS - *Executive Information Systems,* ou DSS - *Decision Support Systems* para ambientes cliente-servidor em Windows (HYPER CONSULTORIA). Além disso, o Maestro possibilita a investigação de dados multidimensionais através de consultas ad hoc, caracterizando-se também como uma ferramenta OLAP. Assim, o Maestro é uma ferramenta de apoio a decisão que atende a profissionais com perfis bem distintos: desde executivos sênior, que têm pouco tempo para investigar dados e desejam informações diretas e objetivas (perfil de usuários EIS); até gerentes operacionais e analistas de negócios, que têm maior disponibilidade e aptidão para investigações mais detalhadas dos dados (perfil de usuários OLAP). O Maestro é comercializado e desenvolvido pela empresa brasileira Hyper Consultoria em Informática.

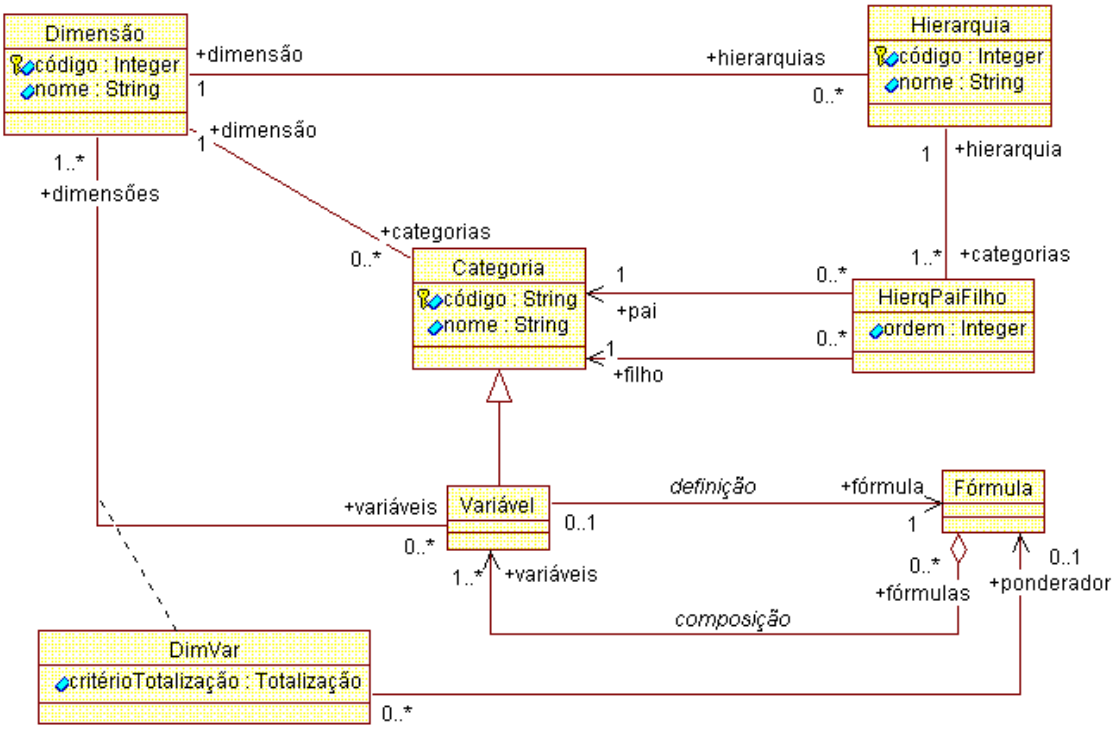

Figura 3.6. Modelo de dados do Maestro.

No modelo do Maestro (Figura 3.6), as dimensões são representadas pela classe Dimensão. Dimensões são conjuntos a partir dos quais os usuários determinam como será sua consulta em um Cubo. Cada Dimensão está associada a uma coleção de Categorias. Uma Categoria é um exemplo de ocorrência da Dimensão. Por exemplo, "Rio de Janeiro" e "Zona Norte" são Categorias da Dimensão "Geografia"; já "Sabonete", "Arroz" e "Biscoito" são Categorias da Dimensão "Produto".

As variáveis no Maestro são Categorias de uma Dimensão especial: a Dimensão Variável. Assim, pode-se dizer que uma Variável é uma especialização de Categoria. Cada Variável se relaciona com várias Dimensões, indicando quais são as Dimensões das quais a Variável depende. Da mesma maneira, uma Dimensão pode determinar o valor de várias Variáveis.

Cada combinação entre Dimensão e Variável (classe DimVar) deve indicar o Critério de Totalização da Variável em relação a Dimensão; ou seja, como essa Variável

deve ser agregada nessa Dimensão. O Maestro prevê os seguintes Critérios de Totalização: "Não Totaliza", "Somatório", "Média Simples", "Média Ponderada", "Máximo", "Mínimo", "Contador" e "Último".

Quando o Critério de Totalização é "Média Ponderada", é necessário indicar uma Fórmula que será usada como ponderador. Fórmulas também são usadas para definir como Variáveis que são calculadas pelo próprio Maestro. Assim sendo, uma Variável pode ser definida por apenas uma Fórmula, sendo que a Fórmula pode ser composta por diversas Variáveis.

O Maestro admite ainda a definição de Hierarquias. Para o Maestro, uma Hierarquia é apenas uma outra maneira de organizar as Categorias de uma Dimensão - isso porque as Categorias da Dimensão são definidas como uma lista simples. Cada Dimensão pode conter diversas Hierarquias. A classe HierqPaiFilho representa o relacionamento entre Hierarquia, Categoria (Pai) e Categoria (Filho). Dessa forma, cada instância de HierqPaiFilho indica que para uma determinada Hierarquia, uma Categoria faz o papel de Filho, enquanto outra faz papel de Pai. A coleção de Filhos de um Pai em uma Hierarquia deve seguir uma determinada ordem.

Uma vez determinados os dados que compõem o Maestro, é importante ver como eles são representados em cubos multidimensionais. A Figura 3.7 ilustra como são fatos e cubos do Maestro em relação às variáveis e categorias. A classe FatoMaestro representa os fatos multidimensionais contidos nas bases de dados. Um FatoMaestro é o resultado da associação entre um conjunto de Categorias e um conjunto de Variáveis. O conjunto de Categorias indica qual o valor que se deseja observar, enquanto o conjunto de Variáveis indica quais as variáveis que serão vistas. Um CuboMaestro é formado por um conjunto de FatoMaestro.

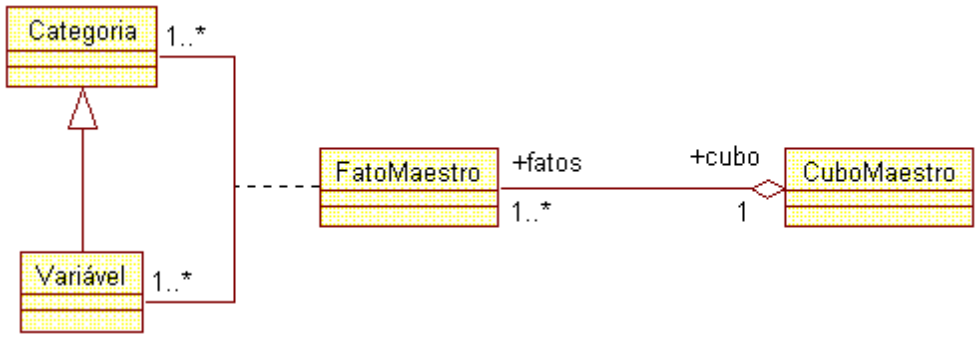

Figura 3.7. Cubos e Fatos do Maestro.

O modelo de dados do Maestro contém algumas regras que não podem ser evidenciadas apenas na sua estrutura de classes. É necessário adicionar regras através de OCL. Por exemplo, é necessário explicitar que a Dimensão associada à classe HierqPaiFilho tem que ser a mesma usada pelas classes Hierarquia e Categoria associadas. Em OCL, isso pode ser expresso da seguinte maneira:

```
context HierqPaiFilho inv:
  dimensão = hierarquia.dimensão and
  dimensão = pai.dimensão and
  dimensão = filho.dimensão
```

Segundo a definição da OCL (OBJECT MANAGEMENT GROUP, 1997), uma expressão OCL deve se referir a uma classe ou operação. No caso acima, a expressão se refere a classe HierqPaiFilho, e de maneira invariante (ou seja, para qualquer instância da classe). Definido o contexto, define-se a restrição desejada através de uma expressão lógica que usa os termos que são visíveis à classe (seus atributos, operações e relacionamentos). Assim sendo, nenhuma instância de HierqPaiFilho poderá ser criada sem que se atenda a esses requisitos.

Outra regra interessante é que uma associação entre Dimensão e Variável (classe DimVar) que tenha Critério de Totalização igual a "Média Ponderada" deve ter uma Fórmula como ponderador. Isso pode ser expresso em OCL da seguinte maneira:

```
context DimVar inv:
  if
    critérioTotalização = 'Média Ponderada'
  then
    ponderador->notEmpty()
```

Assim como essas, existem diversas outras informações que podem (e devem) ser expressas em OCL de modo que o modelo fique completo. A OCL também poderá ser utilizada no estudo de caso para outras relações entre entidades do modelo, bem como diversas regras de negócio.

3.4. Regras OLAP: Sistema versus Negócio

Ferramentas OLAP são sistemas de informação como quaisquer outros. Isso significa que, seguindo os conceitos colocados no capítulo anterior, existem regras de negócio nessas ferramentas que podem ser classificadas da seguinte maneira:

- **Regras para o sistema**: são regras que interessam à realização das tarefas do sistema.

- **Regras para o negócio**: são regras que existem para atender a requisitos particulares do negócio.

- **Regras para sistema e negócio**: são regras que atendem ao dois pólos ao mesmo tempo

Ferramentas OLAP são criadas de maneira independente do domínio de aplicação, o que significa que uma mesma ferramenta OLAP pode ser utilizada em diversos negócios diferentes – independente dos tipos de negócio. Isso significa que a ferramenta OLAP em si – enquanto sistema de informação – não contém regras para um negócio específico. Apesar disso, muitas ferramentas OLAP embutem algumas

capacidades de análise e cálculo que são comuns a muitos negócios, como contraste automático entre variáveis (diferença absoluta e diferença percentual).

No entanto, ferramentas OLAP contêm diversas regras que interessam tanto ao sistema quanto ao negócio. Um exemplo disto é a regra de agregação de uma variável em relação a uma dimensão. O sistema OLAP necessita esta informação para calcular os agregados corretamente, assim como o analista de negócios as necessita para entender o que é o agregado. Para fazer o juízo correto de valor de um agregado, o analista tem que ter em mente como ele foi calculado – especialmente se for uma média. No caso de médias ponderadas, a escolha de um ou outro ponderador pode mudar completamente o significado de um agregado. Se isto não estiver claro na mente do usuário, suas conclusões podem ser completamente distorcidas em relação à realidade.

Outro exemplo importante de regra que atende tanto ao sistema quanto ao analista de negócios: fórmulas de cálculo de variáveis. O sistema necessita destas fórmulas para produzir os valores sendo usados no sistema; o analista de negócios utiliza a fórmula para entender como as variáveis são calculadas, ou ainda que outras variáveis são influentes.

Além disso, estas regras não existem na forma de procedimentos programados – elas são definidas como dados (ou melhor, metadados) nos repositórios dessas ferramentas. Isso significa que, dependendo do grau de abertura dos metadados da ferramenta, estas regras podem ser consultadas e apresentadas para usuários com razoável facilidade.

No entanto, as regras de negócio que talvez mais interessem aos analistas de negócios não se encaixam nesta categoria. Estas são as regras de negócio que não são

interessantes para o sistema. Este tipo de regra pode servir de guia para o analista navegar pelas informações gerenciais, e assim tomar decisões com maior embasamento.

3.5. Características das Regras OLAP

As regras em OLAP tendem a ter características diferentes das encontradas em regras OLTP. Uma delas é que as regras OLTP seguem um modelo mandatário, ou seja, a regra deve ser cumprida sempre. Já as regras OLAP podem ter um caráter mais sugestivo ou menos mandatário que as regras em OLTP. Isso acontece porque as regras OLTP visam à definição de dados e processos para sistemas de informação, enquanto as regras OLAP visam apontar caminhos de análise. Na Língua Inglesa, esta diferença é marcada sutilmente pelo emprego dos verbos modais *must* e *should* (obrigatoriedade e probabilidade, respectivamente). A Figura 3.8 ilustra uma comparação entre *templates* simples de típicas regras de negócio OLTP e OLAP.

OLTP	se <antecedente> então **é obrigatório que** <conseqüente>
OLAP	se <antecedente> então **é possível que** <conseqüente>

Figura 3.8. *Templates* de regra de negócio OLTP e OLAP.

Outra diferença importante entre regras OLTP e OLAP está na temporalidade das informações. Sistemas OLAP não têm a volatilidade de dados que os sistemas OLTP têm, ou seja, as informações OLAP não mudam com o passar do tempo – o histórico é guardado. Isso também deveria valer para regras de negócio OLAP, ou seja, as regras de negócio OLAP deveriam ser mantidas historicamente, de maneira análoga aos dados. Dessa maneira, o analista de negócios pode analisar informações sob o ponto de vista das mudanças de regras de negócio que foram acontecendo. Essa característica poderia aumentar bastante a qualidade das análises.

Em termos do modelo dimensional, se há um histórico de regras de negócio, então regras de negócio são variáveis que dependem da dimensão tempo. Em outras palavras, as regras de negócio podem apresentar dimensionalidade. Por exemplo, há regras que valem apenas para um determinado lugar geográfico, ou apenas para um produto, ou não valem para uma embalagem específica; enfim, a dimensionalidade das informações OLAP abre espaço para uma exploração dimensional de regras de negócio. Por exemplo, os seguintes fatos e regras fazem referência a alguma dimensão específica:

> "Vendem-se mais refrigerantes durante o verão".
> "Coca-Cola é o produto mais importante da categoria de refrigerantes".
> "Mineirinho é uma marca importante em Niterói".

Alguns conceitos típicos de OLAP não se aplicam diretamente em regras de negócio. Por exemplo, regras não são informação numérica: o que significaria um agregado de regras de negócio? O texto das regras cabe em células de planilha: como apresentar as regras para o usuário final? Será que esse tipo de visão é realmente útil para o usuário?

Enfim, regras de negócio no universo OLAP abrem espaço para diversas possibilidades que podem ser interessantes no que diz respeito à modelagem, aplicação e exibição dessas regras.

4. Modelagem de Regras de Negócio

Conforme observado no capítulo 2, regras de negócio devem ser declarativas e manipuláveis computacionalmente e, para tanto, é preciso que elas sejam expressáveis em um modelo de dados. Porém, criar um modelo de dados adequado às necessidades de regras de negócio não é uma tarefa simples.

Isto é dito porque é desejável que as regras de negócio sejam expressáveis em linguagem natural. Logo, um modelo de dados para regras de negócio deve definir mecanismos que permitam ou facilitem a tradução de seus elementos formais em frases em linguagem natural.

Além disto, é desejável que o modelo de dados para regra de negócio tenha poder de expressão suficiente para exprimir tais regras. Esta questão deve ser considerada porque os modelos de dados mais populares e difundidos da atualidade (como o E-R e a UML) têm poder de expressão inferior à lógica de 1ª ordem (DATE, 2000) (SOWA, 2000), o que é insuficiente para a expressão de muitas regras de negócio.

Este capítulo descreve o modelo para regras de negócio proposto pelo *Business Rules Group* (BUSINESS RULES GROUP, 2001), uma comunidade que se dedica à temática de regras de negócio em diversos aspectos, além de ser mundialmente reconhecida e referenciada. Embora o modelo apresente alguns problemas estruturais, ele discute idéias importantes sobre regras de negócio que podem ser adaptadas a uma solução mais adequada para o problema desta dissertação. Uma análise crítica sobre o modelo é desenvolvida ao final do capítulo.

4.1. Escopo e Objetivos do Projeto GUIDE

O Projeto GUIDE (BUSINESS RULES GROUP, 2001) foi uma iniciativa para determinar um modelo conceitual para regras de negócio em sistemas de informação. Iniciado em 1993, o projeto definiu seu modelo em 1997, sendo revisado em 2000. Hoje em dia, o grupo que compôs o GUIDE se denomina *Business Rules Group*, e conta com algumas figuras de peso na comunidade científica como Edgar F. Codd, John Zachman e Ronald Ross.

Os quatro grandes objetivos do Projeto GUIDE são:

- Definir e descrever regras de negócio, bem como alguns conceitos associados. Uma importante conseqüência disto é que se determina claramente o que é (e o que não é) uma regra negócio.

- Definir um modelo conceitual para regras de negócio em termos inteligíveis para profissionais de tecnologia da informação. Este modelo deve definir como são as regras de negócio e como elas se aplicam nos sistemas de informação. Porém, estas definições não devem envolver questões sobre a implementação de regras nos sistemas de informação – o importante é “o que”, e não “como”.

- Prover uma base rigorosa para a engenharia reversa de regras de negócio em sistemas de informação já existentes.

- Prover uma base rigorosa para a engenharia de regras de negócio em novos sistemas de informação.

Até o momento, apenas os dois primeiros objetivos foram alcançados de maneira satisfatória (na visão dos autores do modelo). O grande feito do Projeto GUIDE foi

conseguir definir regras de negócio de maneira declarativa, e não de maneira procedimental – como normalmente se faz em linguagens de programação. Outro ponto importante é que regras de negócio são diretivas atômicas, ou seja, uma regra de negócio não pode ser subdividida em mais regras de negócio.

Um ponto que também merece atenção é que o foco do modelo está nas regras de negócio para sistemas de informação – linha 3 do *Zachman Framework* (ZACHMAN, 1987). Deste modo, as regras definidas pelo modelo não atendem completamente aos requisitos de um sistema OLAP – que é o foco de interesse desta dissertação. Embora haja no modelo inserções para extensões desta natureza, elas não são muito detalhadas porque estavam originalmente fora do escopo do Projeto.

Segundo o BRG (BUSINESS RULES GROUP, 2001), a razão de se atuar na perspectiva de sistemas de informação é fazer com que regras de negócio se tornem um "problema tratável". No entanto, o grupo reconhece a importância da visão de negócios para a aplicação prática de regras de negócio, e vêm desenvolvendo trabalhos nesse sentido.

Mais recentemente, o grupo definiu um modelo de planejamento de negócio (BUSINESS RULES GROUP, 2000) que tem relação com o modelo de regras de negócio elaborado anteriormente. No entanto, o próprio grupo reconhece a necessidade de um modelo para regras de negócio referente à ótica do negócio propriamente dito. Este modelo deverá existir no futuro, mas hoje encontra-se ainda em discussão.

4.2. A Origem das Regras de Negócio

O processo de identificação de regras de negócio é iterativo e heurístico. Segundo o GUIDE, as regras são originadas a partir de diretivas gerais do negócio

(também chamadas de políticas do negócio). O GUIDE não determina uma metodologia para especificar regras de negócio, mas modela como se dá o relacionamento entre as regras e as políticas do negócio. A Figura 4.1 ilustra esses conceitos.

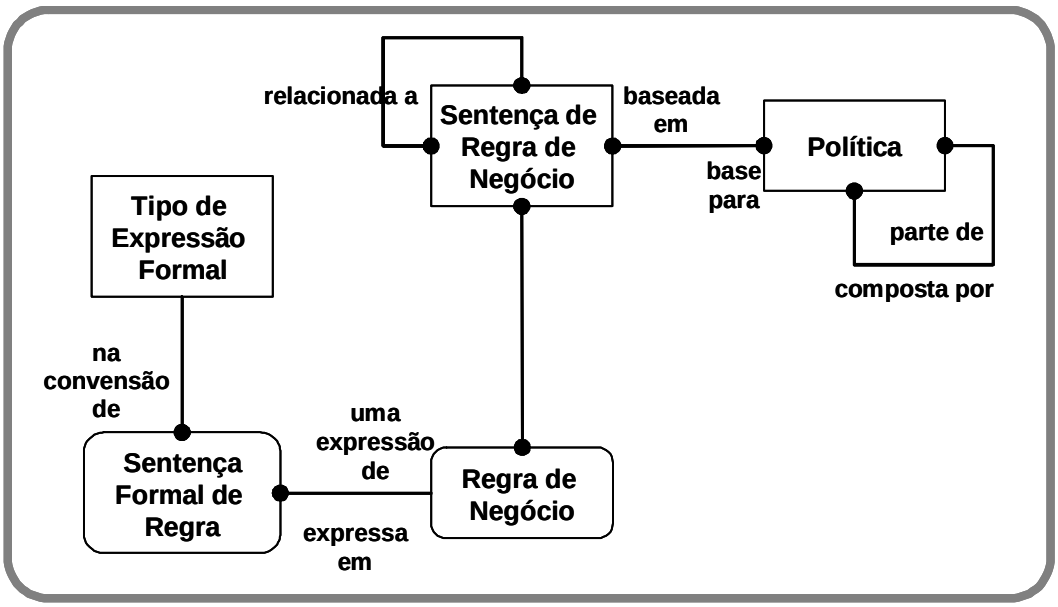

Figura 4.1. A origem das regras de negócio.

Segundo o modelo, as regras começam a existir a partir de Políticas (*Policy*). Essas Políticas são assertivas gerais sobre a direção do negócio. Um exemplo de Política para um supermercado seria a seguinte: “Os alimentos que vendemos para nossos clientes têm que estar em bom estado para consumo dentro de um período de tempo razoável”.

Ainda segundo o modelo, uma Política pode ser a base para uma ou mais Sentenças de Regra de Negócio (*Business Rule Statement*). Uma Sentença de Regra de Negócio é uma declaração sobre estruturas ou restrições colocadas sobre o negócio. Sentenças de Regra de Negócio podem estar relacionadas com várias outras. Um exemplo de Sentença de Regra de Negócio (novamente no contexto de supermercados) é o seguinte: “Pelo menos uma vez por dia, as prateleiras e geladeiras devem ser

averiguadas para procurar artigos que estejam fora ou próximos do fim do prazo de validade. Aqueles que não estiverem mais em condições de consumo devem ser retirados e jogados fora. Aqueles que estão próximos da data de validade mas que ainda podem ser consumidos devem ser colocados em promoção".

Uma Sentença de Regra de Negócio pode originar diversas Regras de Negócio (*Business Rule*). Assim como as Sentenças de Regra de Negócio, as Regras de Negócio definem estruturas ou restrições sobre o negócio, mas elas guardam uma importante diferença: uma Regra de Negócio é atômica, ou seja, não pode ser decomposta em mais Regras de Negócio com maior detalhe. Dessa maneira, pode-se dizer que a decomposição de uma Regra de Negócio implica perda de informação para o negócio. Cada Regra de Negócio é baseada em uma ou mais Sentenças de Regra de Negócio. Um exemplo de Regra de Negócio: "Se um artigo está fora de seu prazo de validade, então ele deve ser retirado de oferta ao público".

Outro ponto importante do modelo é a distinção entre Regras de Negócio e Sentenças Formais de Regra (*Formal Rule Statement*). Uma Sentença Formal de Regra corresponde a uma expressão de uma Regra de Negócio em linguagem formal. Exemplos de linguagens formais seriam: inglês (ou português) estruturado, IDEF1X, ORM, entre outras. No modelo, essas linguagens são representas como Tipos de Expressão Formal (*Formal Expression Type*).

4.3. A Classificação de Regras de Negócio

O modelo do GUIDE classifica regras de negócio (*Bussiness Rule*) em 3 grandes categorias: Assertivas de Estrutura, Assertivas de Ação e Derivações (Figura 4.2).

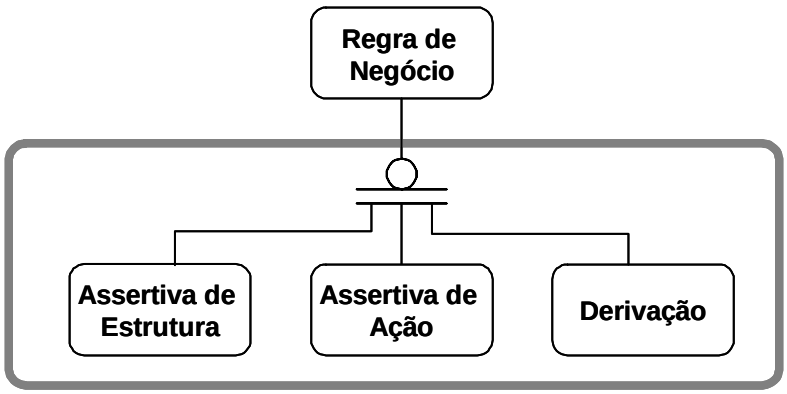

Figura 4.2. A classificação de Regra de Negócio.

- **Assertivas de Estrutura** (*Structural Assertion*): formam o vocabulário e a estrutura básica a partir das quais as Regras de Negócio são expressas. As Assertivas de Estrutura formam a ontologia do negócio (SOWA, 2000).

- **Assertivas de Ação** (*Action Assertion*): são regras que restringem o alcance das estruturas gerais definidas pelas Assertivas de Estrutura. Tratam do estabelecimento de limites, exceções e regulamentos para as estruturas.

- **Derivações** (*Derivation*): são conhecimentos obtidos a partir de outros. Em outras palavras, as Derivações servem para indicar como as Regras de Negócio são obtidas a partir de outras.

4.3.1. Assertivas de Estrutura

As Assertivas de Estrutura definem o vocabulário básico das regras de negócio, bem como as relações lógicas que existem entre os termos deste vocabulário. Em outras palavras, as Assertivas de Estrutura definem a ontologia do negócio (SOWA, 2000).

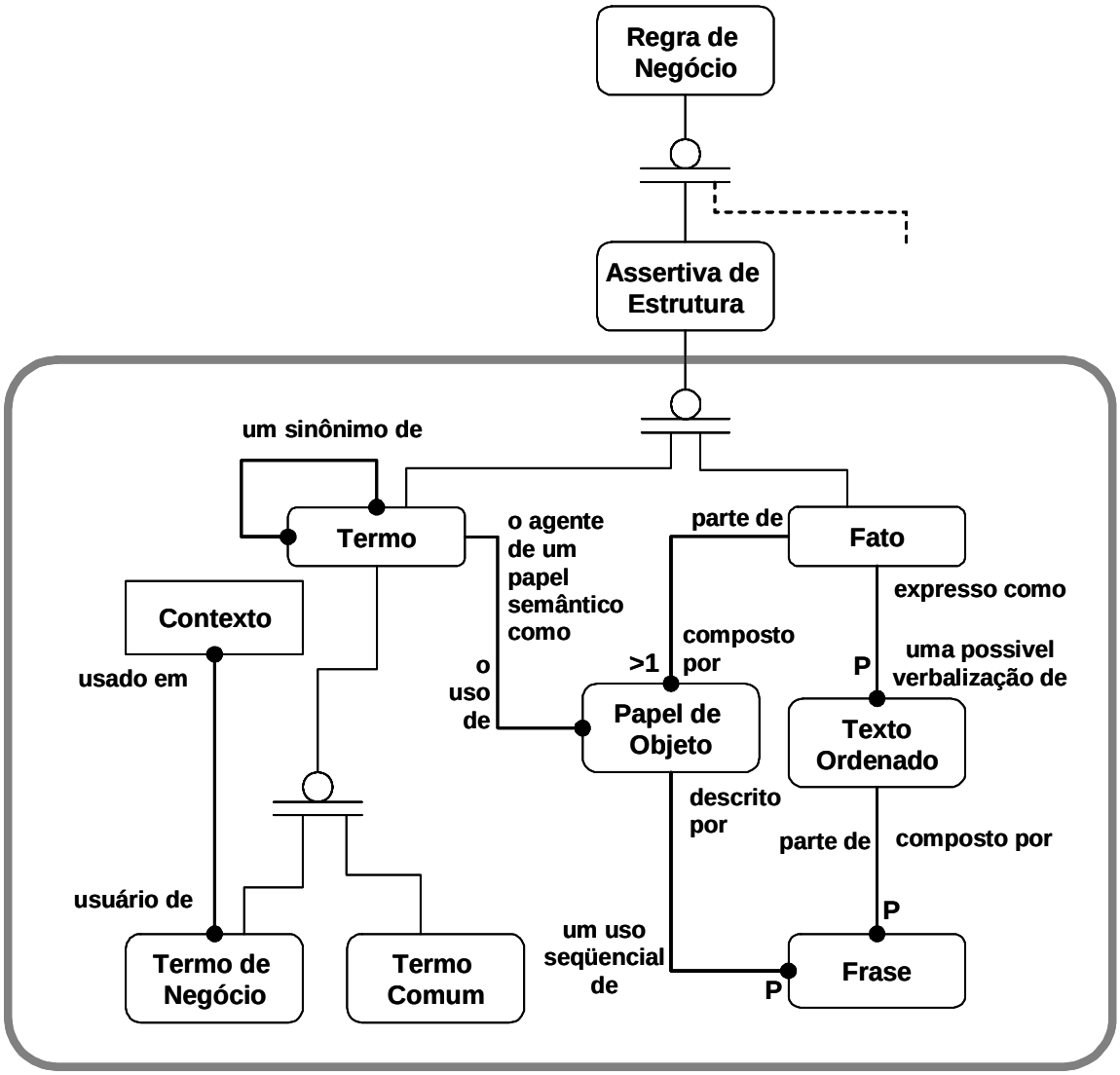

Figura 4.3. As assertivas de estrutura.

A Figura 4.3 mostra que as Assertivas de Estrutura são classificadas em Termos (*Term*) e Fatos (*Fact*). Termos são os elementos que compõem o vocabulário do negócio. Os Termos podem ser Comuns (*Common Term*) ou de Negócio (*Business Term*). Um Termo Comum diz respeito àqueles cujo significado é de conhecimento

geral, independente do domínio em questão. Por exemplo, "supermercado", "preço" e "embalagem" são exemplos de Termos Comuns. Já os Termos de Negócio têm um sentido próprio dentro do negócio. Este sentido é determinado através de um Contexto (*Context*). Por exemplo, "volume base", "multiplicador promocional" e "índice-loja" são Termos que só fazem sentido dentro do Contexto “ScanOne” HYPER CONSULTORIA.

Uma diferença importante entre os Termos Comuns e os Termos de Negócio é que todo Termo de Negócio tem que ser definido a partir de um ou mais Fatos. Por exemplo, o Termo de Negócio “venda” é definido a partir do Fato “um cliente pode levar um artigo de uma loja ao pagar o preço desse artigo”.

Fatos são Assertivas de Estrutura que relacionam dois ou mais Termos. É importante notar que um Fato pode relacionar não apenas Termos particulares, mas também Termos genéricos. O exemplo anterior (“um cliente pode levar um artigo de uma loja ao pagar o preço deste artigo”) diz respeito a qualquer cliente, qualquer artigo e qualquer loja; e não uma instância em particular – embora um Fato também possa descrever situações particulares.

A relação entre cada Termo e Fato se dá através de um Papel de Objeto (*Object Role*). Dentro de um Fato, cada Termo tem um Papel de Objeto associado que representa o papel que este Termo desempenha em relação a este Fato. Cada Fato *é composto por* um ou mais Papéis de Objeto, enquanto cada Papel de Objeto tem que ser *parte de* um Fato. Por outro lado, cada Termo pode ser *agente de um papel semântico como* um ou mais Papéis de Objeto, assim como cada Papel de Objeto tem que ser *o uso de* um Termo em um Fato.

Por exemplo, a Figura 4.4 ilustra o Fato de haver um relacionamento entre os Termos “Cliente” e “Compra”. Há dois Papéis de Objeto associados a este Fato – um indicando que o Termo “Cliente” pode ser *representante de um papel semântico como* um Papel de Objeto que *é parte desse* Fato; e outro indicando que o Termo “Compra” pode ser *representante de um papel semântico como* um Papel de Objeto que *é parte desse* Fato.

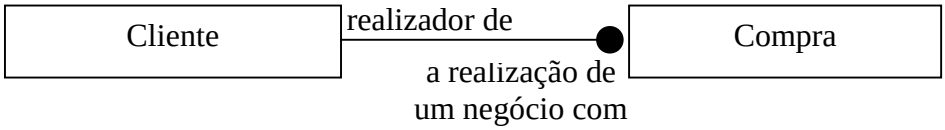

Figura 4.4. Exemplo de Fato relacionando dois Termos.

Uma (pequena) crítica à descrição do modelo BRG

Segundo o BRG (BUSINESS RULES GROUP, 2001), um mesmo Fato pode ser expresso de diversas maneiras. No exemplo da Figura 4.4, existiriam pelo menos duas leituras (expressões) para aquele Fato: uma para cada direção do relacionamento – pois este é um Fato binário (isto é, relaciona apenas dois Termos). Uma possível leitura deste Fato seria “cada Cliente pode ser *realizador de* várias Compras”; enquanto na outra direção seria “cada Compra pode ser *a realização de um negócio com* um Cliente”.

Porém, é importante notar que esta afirmação apresenta inconsistências:

1. As duas leituras não contêm a mesma informação: a 1ª leitura contém informações de cardinalidade que a 2ª não contém (“... várias *Compras”*). Logo, as duas leituras não expressam o mesmo Fato, como foi afirmado.

2. Em conseqüência de (1), cada leitura expressa a leitura de um Fato

diferente. Que relação existe entre os Fatos descritos por cada leitura? E com o Fato originalmente elaborado?

3. A existência de cardinalidades nas descrições dos Fatos expressa restrições ao Fato, de modo que este não é um Fato, e sim uma Assertiva de Ação do tipo Restrição (conforme será observado mais adiante).

Embora a idéia do modelo seja interessante, o exemplo do documento causa confusão quando se deseja utilizar o modelo de maneira prática – o que é o caso desta dissertação. Logo, é importante ter cuidado ao manipular as informações do modelo, pois elas não parecem ter sido testadas de maneira completa. Outras críticas ao modelo são discutidas na seção 4.4

A transcrição de um Fato em sentenças em linguagem natural é uma maneira prática de lidar com eles pois isto permite a comunicação entre usuários comuns e os analistas de sistemas. Desta maneira, as Regras de Negócio formalizadas podem ser transformadas em frases comuns e postas à prova dos usuários para validação.

No GUIDE, representações textuais para Fatos são modeladas através do objeto Texto Ordenado (*Text Ordering*). Cada Fato pode ser *expresso através de* vários Textos Ordenados, e cada Texto Ordenado é *uma possível expressão textual de* um Fato.

Para relacionar os Papéis de Objeto que existem em um Fato com suas representações em Texto Ordenado, usa-se o objeto Frase (*Phrase*). Cada Texto Ordenado tem que ser *composto por* uma ou mais Frases, enquanto cada Frase tem que ser *parte de* um Texto Ordenado. Além disto, cada Frase é *o uso seqüencial de* um dos Papéis de Objeto do Fato, enquanto cada Papel de Objeto tem que ser *descrito por* uma

ou mais Frases. Logo, a Frase identifica para o Texto Ordenado qual é a seqüência de Termos sendo empregada na representação textual do Fato. Frases também provêm o papel sintático do Papel de Objeto (ou seja, se a Frase é sujeito ou objeto do Texto Ordenado) e um texto com marcação que representa o Papel de Objeto textualmente.

Por exemplo, “cada Cliente pode ser realizador de várias Compras” é um Texto Ordenado que representa o Fato da Figura 4.4 sendo lido da esquerda para a direita. Neste exemplo, existem duas Frases. Na primeira, o Papel de Objeto relacionado com o Termo “Cliente” é sujeito da Frase, ocupa a posição 1 do Texto Ordenado e está associado ao texto “cada <>” (onde <> representa a marcação para o Termo). Já a segunda Frase diz respeito ao Termo “Compra” e indica que ele é objeto, ocupa a posição 2 do Texto Ordenado e está associado ao texto “pode ser realizador de várias <>”.

Outro Texto Ordenado para o Fato da Figura 4.4 (da direita para a esquerda) é “cada Compra pode ser a realização de um negócio com um Cliente” também tem duas Frases, sendo que os Papéis de Objeto de cada Termo trocam a função sintática entre si (um passa a ser objeto e o outro passa a ser sujeito) e de posição no Texto Ordenado.

Embora o modelo não esclareça formalmente, existe uma simetria entre as relações (Fato – Termo) e (Texto Ordenado – Frase). Deste modo deve sempre existir uma correspondência de um para um entre as Frases de um Texto Ordenado e os Papéis de Objeto do Fato representado pelo Texto Ordenado.

O modelo também prevê classificações para os Fatos (Figura 4.5). Uma destas classificações divide os Fatos entre Fatos Básicos e Fatos Derivados. Os Fatos Básicos são simples associações entre Termos ou Fatos. Já os Fatos Derivados são resultantes de

cálculos matemáticos ou inferências lógicas a partir de outros Fatos. Os Fatos Derivados são usados para se definirem Regras de Negócio do tipo Derivação (*Derivation*).

Uma outra classificação divide os Fatos em três tipos: Atributo, Participação e Generalização. Um Fato é um Atributo quando um de seus Termos é um Atributo de outro. Um exemplo disto é o Fato "capacidade é um atributo de embalagem". Um Fato também pode representar uma relação de Generalização: "refrigerante é uma especialização de bebida". Uma Participação indica que dois Termos estão associados em algum sentido no negócio.

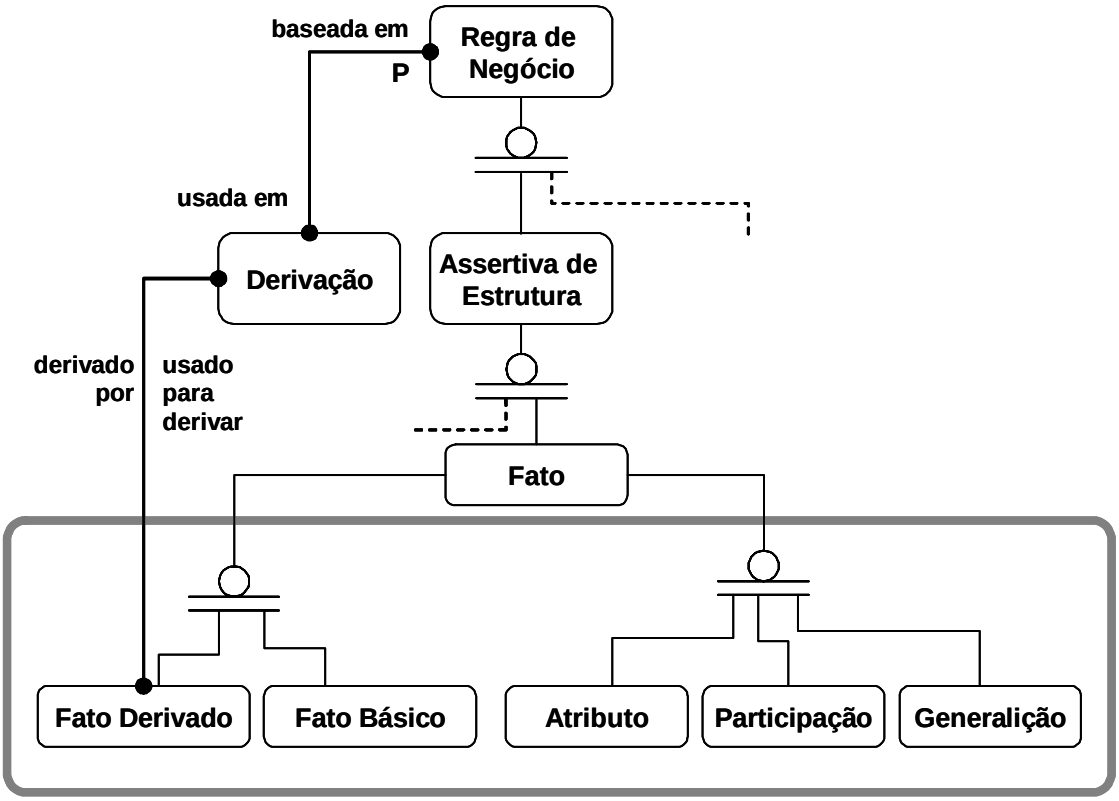

Figura 4.5. As classificações para o objeto Fato.

Um Fato do tipo Participação pode ser ainda sub-classificado em Associação, Agregação e Papel. Uma Agregação indica que um Termo faz parte de outro ("pneu faz parte de automóvel"). Um Fato do tipo Papel descreve o papel que um Termo desempenha em relação a outro em uma interação do negócio ("uma pessoa pode ser

cliente em uma venda"). Já as Participações que são do tipo Associação são quaisquer outro tipo de associação entre termos. É interessante notar que estes tipos de relacionamentos são muito semelhantes aos que compõem os modelos de dados populares, como o E-R e a UML.

4.3.2. Assertivas de Ação

As Assertivas de Ação (Figura 4.6) são declarações que se referem a aspectos dinâmicos do negócio. Elas especificam restrições sobre os resultados que uma ação pode produzir. Em outras palavras, as Assertivas de Ação impõem restrições às Assertivas de Estrutura.

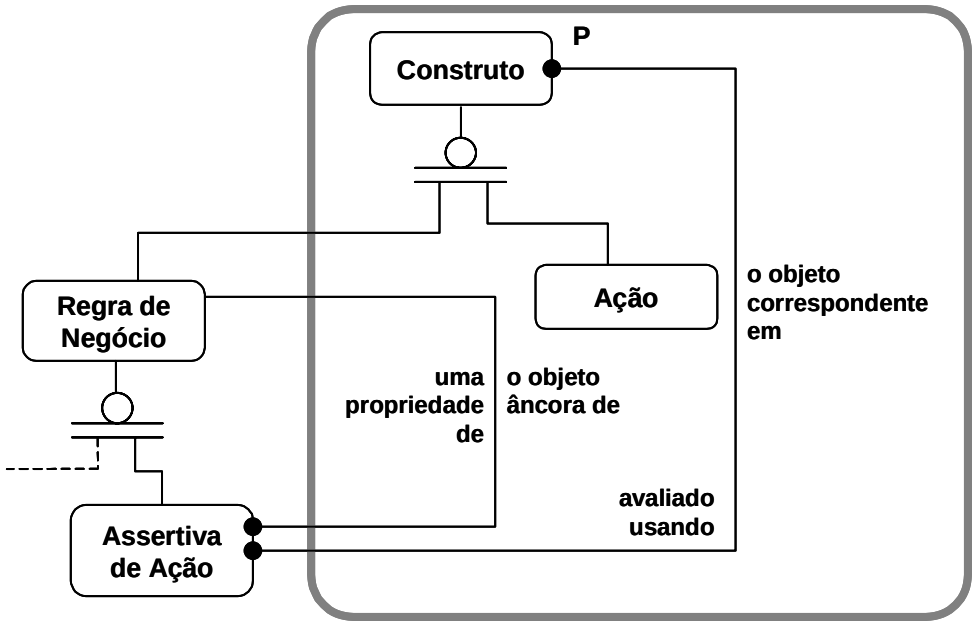

Figura 4.6. Assertivas de Ação.

Uma Assertiva de Ação é composta por um objeto âncora e um ou mais objetos correspondentes. Em uma regra da forma "Se ... Então ...", o objeto âncora seria a parte logo após o "Se", e os objetos correspondentes ficariam logo após o "Então".

Embora o modelo não demonstre explicitamente, o texto do GUIDE afirma que as Assertivas de Ação podem utilizar modificadores para obterem o efeito restritivo da maneira adequada. Modificadores são expressões que alteram o significado de frases; por exemplo, "tem que ser", "pode ser", "pode ser apenas", entre outras. Na verdade, os modificadores são traduções dos verbos modais da Língua Inglesa (como *must*, *should*, *could* etc).

O objeto âncora pode ser qualquer tipo de Regra de Negócio (Assertiva de Estrutura, Assertiva de Ação ou Derivação). Em contrapartida, toda Assertiva de Ação tem que ser uma propriedade da Regra de Negócio associada. Geralmente, o objeto âncora é uma Assertiva de Estrutura, embora outras especializações de Regra de Negócio sejam permitidas.

O objeto correspondente pode ser ou uma Regra de Negócio ou uma Ação (*Action*). O objeto Construto (*Construct*) é a generalização dos dois. No contexto do ScanOne, uma Ação pode ser a venda de um artigo, o aumento de um preço, ou o promocionamento de um artigo.

O modelo GUIDE prevê uma classificação para as Assertivas de Ação (Figura 4.7). As especializações de Assertiva de Ação são Condição (*Condition*), Restrição de Integridade (*Integrity Constraint*) e Autorização (*Authorization*). Segundo o GUIDE, esta classificação é completa, no sentido que este conjunto de classes cobre todas as possibilidades para Assertivas de Ação. Além disto, a classificação é exclusiva: cada instância de Assertiva de Ação só pode pertencer a uma única classe.

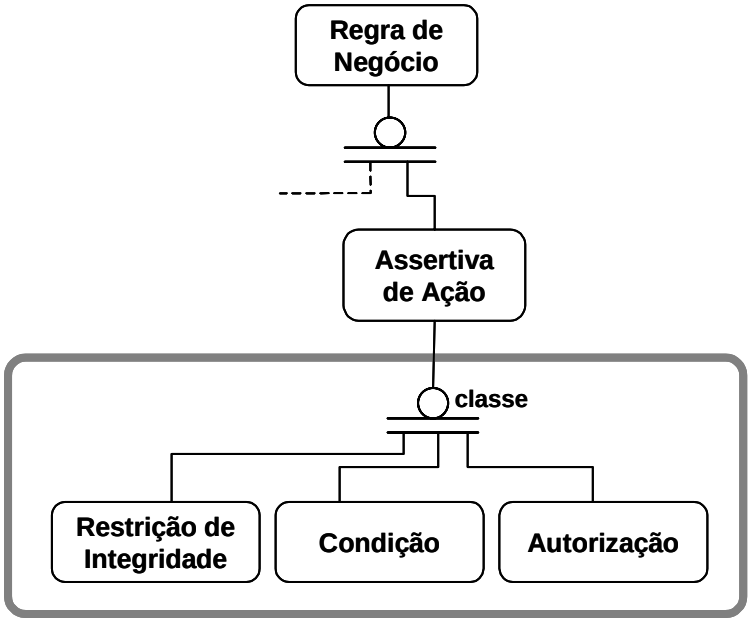

Figura 4.7. Classificação das Assertivas de Ação.

Uma Condição é uma assertiva que deve ser testada para saber se alguma outra regra deve ser aplicada. Por exemplo: “o preço caiu mais de 20%”, “acabou o artigo no estoque”, “houve aumento de imposto”.

Uma Restrição de Integridade é uma assertiva que sempre representa uma verdade. Enquanto uma Condição pode ou não ser verdadeira, uma Restrição de Integridade sempre o é, não importa a situação. Por exemplo “todo artigo tem que ter um preço”, “o volume de vendas é sempre um número positivo ou nulo”.

Uma Autorização define uma prerrogativa ou privilégio a respeito de um ou mais Construtos. As Autorizações seguem a forma: “Apenas x pode fazer y”, onde geralmente x é um usuário e y é uma ação a ser executada. Por exemplo: “apenas o gerente do supermercado pode aceitar compras a prazo”.

Além destas classificações, o GUIDE também prevê uma tipificação para as Assertivas de Ação. Diferente das classificações, a tipificação não determina um conjunto fechado de possibilidades para as Assertivas de Ação, de modo que este

conjunto pode crescer conforme a necessidade de quem modela as regras. O modelo define alguns tipos úteis: Enabler, Timer e Executive (Figura 4.8).

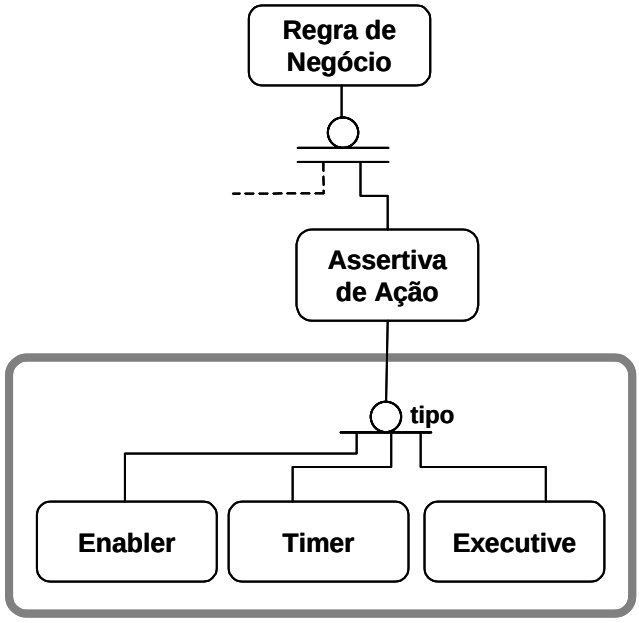

Figura 4.8. Tipos de Assertiva de Ação.

Um Enabler é um tipo de Assertiva de Ação que, se verdadeira, permite ou lida com a existência do objeto correspondente. A assertiva é verdadeira se o objeto âncora existe. A interpretação do objeto varia de acordo com a natureza do objeto correspondente:

- Se o objeto correspondente é uma Assertiva de Estrutura, o Enabler permite a criação de uma nova instância.

- Se o objeto correspondente é uma Assertiva de Ação, o Enabler permite outra Assertiva de Ação.

- Se o objeto correspondente é uma Ação, o Enabler permite a sua execução. Caso contrário, a execução é bloqueada ou desabilitada.

Mais precisamente, a descrição acima é a de um Enabler que é uma Restrição de Integridade. Quando o Enabler é uma Condição, ele simplesmente testa se o objeto correspondente está habilitado, ao invés de diretamente habilitá-lo.

Timer é um tipo de Assertiva de Ação que testa, habilita (ou desabilita), ou cria (ou destrói) caso um limite especificado tenha sido alcançado. Um Timer pode ser entendido como uma “contagem regressiva”, ou um “alarme de relógio”. Por exemplo, se o que é controlado pelo Timer é outra Assertiva de Ação, então o Timer vai “ligar” esta assertiva quando seu limite for alcançado.

Um Executive é um tipo de Assertiva de Ação que causa a execução de várias Ações. O seguinte exemplo ilustra como estes tipos podem ser usados em conjunto: “se um cliente passar três meses sem pagar, então apreender a mercadoria”. A parte que conta “três meses sem pagar” e requer uma ação em resposta é uma Condição do tipo Timer. A segunda Assertiva de Ação “apreender a mercadoria” é uma Restrição de Integridade do tipo Executive.

Existe uma última classificação para as Assertivas de Ação. Nesta classificação, dividem-se as Assertivas de Ação entre as que indicam obrigatoriedade (*must*) e as que indicam possibilidade (*should*). A primeira classe é chamada de Assertiva de Controle da Ação (*Action Controlling Assertion*) e a segunda classe é Assertiva de Influência na Ação (*Action Influencing Assertion*). A Figura 4.9 ilustra essa classificação.

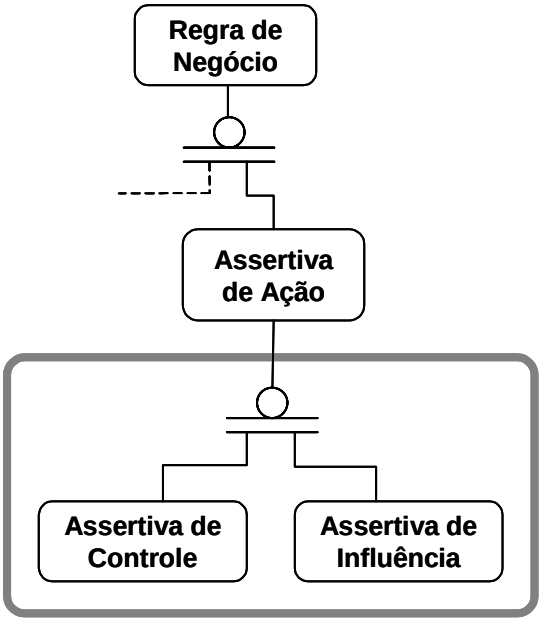

Figura 4.9. Assertivas de Controle versus Assertivas de Influência.

A grande motivação desta classificação é separar as regras tipicamente usadas em sistemas OLTP (Assertivas de Controle) daquelas usadas em sistemas OLAP (Assertivas de Influência). As Assertivas de Influência são úteis para as empresas determinarem guias para tomada de decisão. Segundo o GUIDE, estes guias devem indicar quais são as situações esperadas ou normais, bem como qual ação deve ser tomada quando uma exceção acontece. Por exemplo, seja a Restrição de Integridade “menos do que 10% do estoque deve ter a validade expirada na semana”. Caso essa regra seja quebrada, então executa-se a seguinte ação de contingência “fazer uma promoção leve 2 pague 1 até o final da semana”.

Embora identifique a existência e a importância das Assertivas de Influência, o Projeto GUIDE não modelou detalhadamente como esta influência acontece no nível das regras de negócio: isto é indicado como um trabalho que o grupo deve realizar no futuro.

4.3.3. Derivações

Uma das especializações de Regra de Negócio é Derivação (*Derivation*). Derivações são Regras de Negócio que indicam como um conhecimento pode ser obtido a partir de outro. Desta maneira, novos Fatos podem ser originados como conseqüências de uma Derivação. Estes novos Fatos são chamados de Fatos Derivados. A Figura 4.10 mostra como as Derivações se relacionam com os outros elementos do modelo.

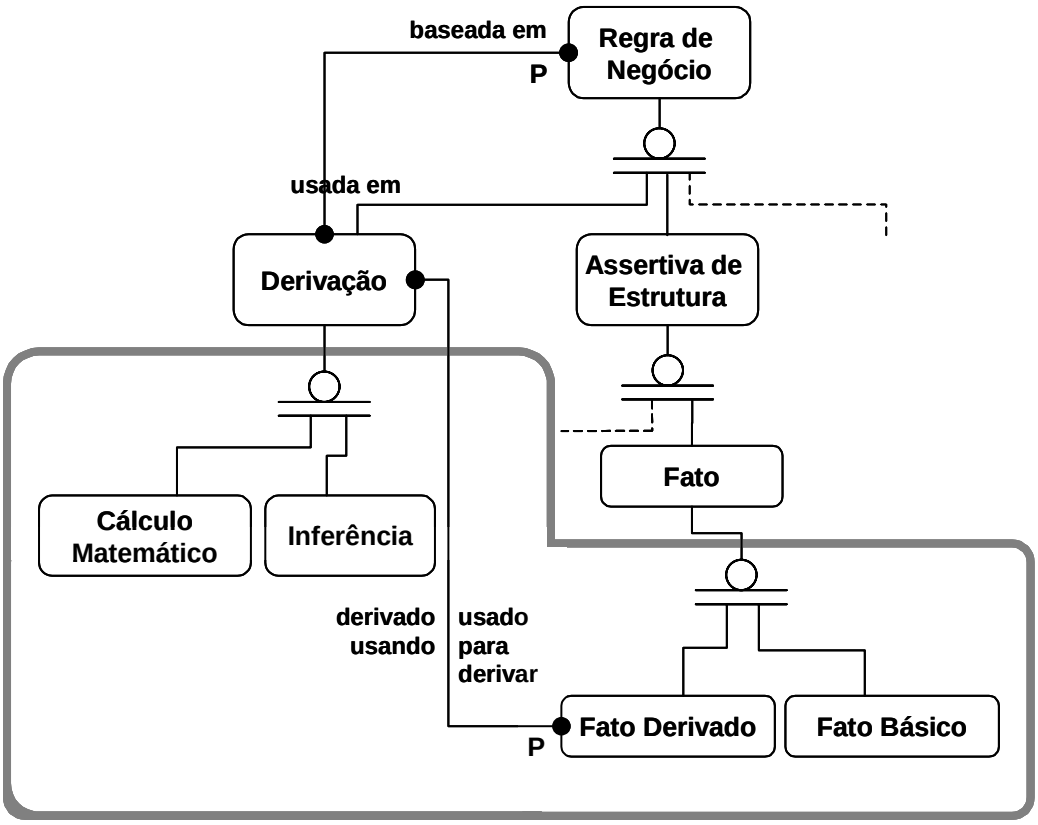

Figura 4.10. Derivações e Fatos Derivados.

De acordo com o modelo, uma Derivação pode gerar vários Fatos Derivados. Além disso, cada Derivação tem que estar baseada em uma ou mais Regras de Negócio (Assertivas de Estrutura, Assertivas de Ação ou outras Derivações).

As Derivações são classificadas em Cálculos Matemáticos e Inferências. Cálculos Matemáticos são Derivações que envolvem operações aritméticas, por

exemplo “volume de vendas multiplicado pelo preço”. Inferências são Derivações que utilizam dedução ou indução lógicas para gerar informações novas. Um exemplo de Inferência é “como o preço subiu e os custos de produção são os mesmos, então houve aumento na margem de lucro”.

4.4. Uma Análise Crítica sobre o Modelo GUIDE

O modelo GUIDE compreende algumas idéias interessantes sobre o universo de regras de negócio. Dentre estas idéias, destaca-se a decomposição de regras em termos do negócio, que devem ser previamente definidos antes de se construir qualquer fato ou regra. Desta forma, as regras de negócio de uma corporação podem ser sempre construídas a partir de um vocabulário conhecido e bem definido, evitando ambigüidades e mal-entendidos.

Outra idéia interessante proveniente do modelo é a relação existente entre os objetos que definem a estrutura de dados (Termo e Fato) e os objetos que constroem as frases em linguagem natural (Frase e Texto Ordenado). Esta relação indica que há possibilidade de se passar de linguagem formal para linguagem natural (e vice-versa) com relativa facilidade.

Embora apresente estes pontos positivos, o modelo é criticável em diversos outros. O principal problema do GUIDE está na definição das Assertivas de Estrutura como uma especialização de Regra de Negócio. Na verdade, as Assertivas de Estrutura formam a ontologia do negócio, que não são em si regras do negócio. O motivo desta confusão é que existem no modelo situações em que se faz referência a qualquer tipo de Assertiva (de Estrutura, de Ação ou Derivação), e daí refere-se a Regra de Negócio pois este objeto incorporaria esta generalidade. Na realidade, algum outro objeto deveria ter

sido criado para capturar a generalidade das Assertivas e outro nome deveria lhe ser dado: como foi feito para o objeto Construto das Assertivas de Ação – ele está lá apenas para generalizar as classes Regra de Negócio e Ação.

Mas porque as Assertivas de Estrutura não seriam especializações de Regra de Negócio? Seja o seguinte contra-exemplo: o Termo “geografia” é uma Assertiva de Estrutura, mas ele é uma Regra de Negócio? Creio que não. De maneira geral, pode-se dizer que nenhum Termo pode ser considerado, por si só, como uma Regra de Negócio. Assim sendo, a generalização do modelo não é correta.

Porém, o argumento é até aceitável quando se trata de Fatos. Uma vez que fatos expressam regularidades, ou seja, relações entre entidades que ocorrem com freqüência, esta “regularidade” pode levar a entender este tipo de informação como sendo uma Regra. Porém, esta não parece ser a forma mais adequada de se modelar estas informações.

Há também problemas na definição de Texto Ordenado e Frases na representação de Fatos. Um problema é que existem expressões necessárias para dar o sentido aos Fatos não aparecem no modelo mas aparecem nas Frases dos Textos Ordenados. Por exemplo, o Texto Ordenado “cada Compra tem que ser a realização de um negócio com um Cliente”, utiliza expressões como “cada” e “tem que” para compor suas Frases. Certamente, diversos outros Textos Ordenados farão uso das mesmas expressões, mas elas não são modeladas em lugar nenhum. Isto é um problema porque pode dificultar a padronização das expressões das regras.

Outra questão importante é que o modelo GUIDE classifica Fatos de maneira semelhante a modelos de dados populares, como o E-R e a UML. Isto tem vantagens (lidar com conceitos amplamente conhecidos), mas também traz desvantagens

relevantes. A principal desvantagem é que estes modelos têm expressividade lógica inferior à lógica de 1ª ordem (não têm variáveis, quantificadores universais e operador de negação). Deste modo, diversos Fatos e Regras de Negócio podem não ser expressáveis através do modelo GUIDE.

Ainda sobre as especializações de Fato, faz falta o relacionamento de instanciação, ou seja, um relacionamento que associe um tipo a uma instância. Embora implicitamente o GUIDE entenda que existem tipos e instâncias, este relacionamento não é modelado, podendo fazer falta para diversas aplicações. Nesta dissertação em particular, este relacionamento é interessante para distinguir as dimensões de suas categorias, já que as categorias são instâncias de dimensão. Por exemplo, a categoria “Rio de Janeiro” é uma instância da dimensão “Geografia”.

Outra lacuna do modelo GUIDE é a conciliação entre as Assertivas de Estrutura e construções matemáticas usadas para definir Derivações. Por exemplo, como ficaria a regra: "Se há promoção, então o volume base deve ser maior que o volume observado"? Teria que haver um sistema de tipos que suportasse números reais e a relação "maior que" para estes números.

Além disto, como tornar esta relação geral a ponto de se poder exprimi-la em termos de outros fatos? Por exemplo, a Regra de Negócio "se há promoção, então o volume observado é igual ao volume base vezes o multiplicador da promoção". Na verdade, este tópico deverá ser abordado ao se definir uma linguagem de expressão para o modelo, pois as expressões matemáticas são regras importantes, mas o formato termo-fato não é o mais adequado para exprimi-las.

Outro problema é que o modelo não lida bem com o conceito de conjunto e coleção, e nem com quantidades e unidades de medida. Por exemplo, através do modelo

pode-se definir individualmente que as geografias “RJ”, “SP”, “MG” e “ES” são filhas da geografia “Sudeste”, mas não existe a noção explícita que elas formam um conjunto, e que poderiam ser tratadas como tal. Assim sendo, muitas expressões possíveis (em SQL, por exemplo) não podem ser realizadas no modelo – mais uma vez por falta de equivalência com a lógica de 1ª ordem.

Ainda nesta questão, existe o problema de não se tratarem informações de quantidade e unidade de medida. Por exemplo, Assertivas de Ação do tipo Timer necessitam um limite para marcar o momento que devem ser ativadas. Este limite muitas vezes será um número, possivelmente seguido de uma unidade de medida. Por exemplo, o Timer “quando o estoque de feijão for menor que 300 caixas, deve ser feito um novo pedido de 500 caixas ao fornecedor”. Como o modelo expressaria a Frase “300 caixas de feijão”? E como fica a expressão se for mudada a unidade de medida? Por exemplo “562,60 kg de feijão” poderia ser a mesma quantidade expressa em “kg” ao invés de “caixa”. Como se dá a conversão de uma unidade para outra? Seria uma outra regra de negócio?

Por fim, para esta dissertação em particular, seria interessante que o modelo detalhasse melhor como se dão as Assertivas de Influência nas Ações pois estas assertivas são interessantes para sistemas OLAP. Como esta definição ainda não existe, esta dissertação desenvolve algumas maneiras de expressar isto através de um estudo de caso. Esta é a temática do próximo capítulo: como aplicar regras de negócio em aplicações OLAP.

5. Aplicando Regras de Negócio em OLAP

Este capítulo se dedica a aplicar regras de negócio em análises OLAP. Para tanto, será realizado um estudo de caso no comércio varejista, especialmente para supermercados. Como auxílio às análises, será utilizada uma aplicação OLAP especializada em varejo chamada ScanOne. O ScanOne é voltado para profissionais do mercado de produtos de consumo – fabricantes e varejistas. O objetivo do ScanOne é tornar mais eficiente e embasado o processo de tomada de decisão destes profissionais.

Inicialmente, o capítulo tratará de algumas definições e conceitos importantes para que se possa entender o mercado varejista e o próprio ScanOne. Um exemplo de análise será discutido para que se entenda melhor o escopo e as regras de negócio que existem. Em seguida, um modelo de dados será esboçado para abrigar os conceitos do ScanOne e possíveis regras de negócio. Enfim, estas regras serão descritas formalmente através da OCL.

5.1. Mercado Varejista, ScanOne e Regras de Negócio

Tradicionalmente, o comércio varejista é um dos domínios de negócio que mais utiliza ferramentas de apoio a decisão. Uma razão para isto é que estas aplicações têm que lidar com uma enorme quantidade de dados: dezenas (ou centenas) de lojas vendendo milhares de artigos todos os dias. Para analisar esta massa de dados, é necessário usar variáveis que meçam aspectos específicos desta realidade complexa. A quantidade de variáveis também tende a ser grande: só o ScanOne disponibiliza mais de 200. Além disso, estas variáveis podem se influenciar mutuamente e de maneira muito complexa –até mesmo imprevisível.

Embora as informações de varejo não sejam simples de serem manipuladas, elas são preciosas e necessárias para que a realidade do negócio seja entendida corretamente. Daí a necessidade de um ambiente com as características do ScanOne: facilitar o acesso a informações gerenciais. A maneira com que o ScanOne realiza isso é através de interfaces OLAP, que inclui planilhas e gráficos multidimensionais, *drill-down*, *slice-and-dice*, *ranking* etc.

Porém, “facilitar o acesso a informações gerenciais” não significa apenas disponibilizá-las – é preciso também filtrar o que é relevante. No contexto de uma análise, nem todas as informações disponíveis têm influência decisiva. Poupar o usuário de se preocupar com o que não tem relevância também é uma maneira de “facilitar o acesso a informações gerenciais”.

Regras de negócio podem orientar a determinação desta relevância. As regras de negócio podem guiar o usuário mostrando como as variáveis se influenciam

mutuamente. Ao saber o que realmente interessa – e porque interessa – o analista pode avaliar o que é importante nas situações sendo analisadas.

5.2. Os Dados de Scanner

O termo *scan* refere-se ao processo de leitura ótica dos códigos de barras que acompanham os artigos vendidos em supermercados. Chama-se de dados de scanner as informações obtidas sobre a venda dos artigos adquiridos pelos clientes ao passarem pelo caixa. Mais precisamente os dados de scanner são: 1) o volume de vendas e 2) o faturamento dos artigos. O volume de vendas é a quantidade de artigos vendidos. O faturamento é a quantidade de dinheiro obtido com as vendas. Quando se divide o faturamento de um artigo pelo seu volume de vendas, obtém-se o preço deste artigo.

Assim sendo, cada passagem de um artigo pelo caixa de supermercado inclui uma informação multidimensional na base de dados do supermercado. Estas informações têm pelo menos as seguintes dimensões: loja (geografia), artigo (também chamado de SKU, ou *Stock Keeping Unit*) e tempo (data-hora em que ocorreu a venda). A partir destas informações, sistemas de suporte à decisão podem então ser alimentados – o ScanOne é um exemplo deste tipo de sistema.

Embora muito relevantes, as informações de "scanner" não são as únicas formas de se acompanhar o desempenho do negócio varejista. Informações de estoque, por exemplo, podem influenciar a eficiência com que uma loja vende (pode haver falta de artigos). Informações sobre a disposição de artigos nas prateleiras também podem explicar o comportamento dos clientes perante as vendas. A veiculação de propaganda na mídia sobre ofertas e vantagens de uma loja e/ou artigo também pode influir no

negócio. Em sua versão atual, o ScanOne disponibiliza variáveis sobre informações de estoque e de veiculação de mídia, além daquelas provenientes de "scanner".

Apesar de possibilitarem a construção de ricos cenários de análise, os analistas de negócio têm que lidar com informações faltosas ou insuficientes. Por exemplo, informações sobre a posição dos artigos nas prateleiras não são fáceis de serem obtidas (ao menos nas redes de supermercado brasileiras). Outra ausência importante é a possibilidade de identificar unicamente os clientes: nem todos os clientes usam cartões de crédito ou outras formas de identificação única. Esta característica limita a atuação de práticas de CRM (*Customer Relationship Management*), como a análise de cestas, identificação de perfis e estratégias de fidelidade.

5.3. Variáveis do ScanOne

O Scan One possibilita que os analistas de negócios obtenham “fotografias” de seu negócio no passado (recente e remoto), de modo que eles possam construir estratégias e tomem decisões sobre o negócio fundamentadas em argumentos sólidos, não apenas palpites ou sentimentos. Enfim, as análises via ScanOne servem para observar a realidade de um varejo.

No ScanOne, a realidade se reflete em variáveis. O ScanOne utiliza basicamente variáveis de scanner (volume e preço), a partir das quais deriva outras (como faturamento médio e preço regular). Há também outras variáveis que não são fruto apenas dos dados de scanner (como o número de dias com estoque zero).

A partir da interpretação das variáveis, pode-se determinar estratégias de atuação para influenciar o mercado de maneira adequada aos interesses do negócio. Estratégias são implementadas através de ações sobre o mercado. Exemplos de ações no mercado

são: mudanças de preços, promoções, propagandas no jornal, rádio ou TV, lançamentos de novos produtos, entre outras. As variáveis do ScanOne que expressam ações no mercado são chamadas de variáveis de controle. As outras variáveis têm seus valores determinados (ou apenas influenciados) por variáveis de controle.

O ScanOne também disponibiliza variáveis especiais que expressam situações hipotéticas úteis para análises. Por exemplo, caso tenha havido uma promoção de um artigo durante uma semana, os dados de scanner conterão o volume de vendas daquele artigo durante aquela semana. Muito provavelmente, este valor será superior ao que seria se esta promoção não tivesse acontecido. Mas será que isto foi verdade desta vez? De quanto seria o volume se não houvesse esta promoção? Valeu a pena fazer essa promoção? Sem ter uma idéia a respeito do valor deste volume sem promoção (que é hipotético), não se pode analisar o desempenho da promoção (que é real).

O ScanOne utiliza métodos estatísticos para construir informações hipotéticas a partir dos dados reais. Ou seja, ao analisar extensas séries de tempo dos dados disponíveis, o ScanOne pode indicar valores prováveis para situações que não ocorreram realmente. Por exemplo, para o caso da promoção do artigo, a variável Volume Baseline mostra o volume de vendas caso promoções não tivessem acontecido.

Além do Volume Baseline, existem outras variáveis de caráter hipotético, como o Volume Ajustado. O Volume Ajustado indica quanto seria o volume de vendas sem distorções eventuais, como variações ocasionais de temperatura, festas e feriados, tendências decorrentes de variações no nível de atividade econômica, entre outras. Através do Volume Ajustado, as ações praticadas podem ser comparadas e analisadas de maneira mais justa.

5.4. Exemplo de Análise com ScanOne

Neste exemplo, serão ilustrados alguns conceitos do negócio de supermercados – que na verdade valem para grande parte dos negócios – além de algumas regras de negócio úteis para o usuário em suas análises. Este é um exemplo de análise realizada através do ScanOne.

O exemplo é sobre a categoria de sabonetes de uma grande rede de supermercados. A rede tem lojas no Brasil inteiro, vendendo todos os principais fabricantes de sabonetes do País. No caso, a análise visa comparar o desempenho de vendas entre dois meses consecutivos. A Visão 5.1 ilustra o resultado de uma consulta ao ScanOne realizada com este intuito. Esta consulta ilustra o total de volume para toda a categoria de sabonetes entre os meses de julho e agosto de 2000.

	Volume de Vendas			
	Jul/2000	Ago/2000	DifABS	Dif%
Total Sabonetes	161,568	156,202	-5,366	-3.3%

Visão 5.1. Comparação de volume de vendas entre dois meses.

Para analisar o desempenho destas vendas, é necessário comparar os volumes entre estes dois meses. Para tanto, duas medidas foram utilizadas: a diferença absoluta (DifABS) e a diferença percentual (Dif%) entre os dois meses. A diferença absoluta é a subtração simples do volume do mês de agosto sobre o mês de julho (Regra 5.1). Assim sendo, diz-se que o mês de julho é a referência da diferença.

Regra: Diferença Absoluta

DifABS = Valor – ValorRef

Regra 5.1. Fórmula da diferença absoluta.

Já a diferença percentual mede quanto variou percentualmente o volume a partir do mês de julho (Regra 5.2). Mais uma vez, o mês de julho é tomado como referência.

Variáveis como diferença absoluta e diferença percentual são chamadas de variáveis de contraste, uma vez que servem para contrastar fatos semelhantes.

Regra: Diferença Percentual

Dif% = (Valor – ValorRef) / ValorRef

Regra 5.2. Fórmula da diferença percentual.

Analisando a diferença percentual (Dif%), vê-se que houve uma baixa de –3,3% no volume total da categoria. Este valor de perda é considerado muito alto pelo analista de negócios. Por que uma perda como esta aconteceria? Uma relação muito presente neste tipo de negócio é a relação inversa entre preço e volume. Em outras palavras, quando o preço sobe, o volume tende a cair; e quando o preço cai, o volume tende a subir. Estas são regras de negócio importantes, e podem ser enunciadas segundo a Regra 5.3.

Regra: Preço-Volume

“Se há alta de preço, é provável que haja baixa de volume”.

“Se há baixa de preço, é provável que haja alta de volume”.

Regra 5.3. Relações inversas entre volume e preço.

É importante notar que estas regras não são mandatárias, no sentido que a relação não é uma verdade absoluta. Outro ponto importante é que as regras são enunciadas sobre comparações de preço e volume (sem maiores restrições sobre como se dão estas comparações). No entanto, fica implícito que o critério de comparação de preços tem que ser o mesmo critério de comparação de volumes. Por fim, também cabe notar que estas regras indicam que preço é uma variável que afeta volume – mas a recíproca não é verdadeira.

Dando continuidade ao exemplo, uma vez que houve baixa de volume, isto pode ser explicado por uma alta de preços (segundo a Regra 5.3). Porém, a regra não é mandatária, ela apenas indica uma tendência esperada. Desta maneira, é preciso verificar se isto realmente ocorre na prática. A Visão 5.1 ilustra a variação de preço médio dentro do mesmo período para a categoria de sabonetes. Realmente, a diferença percentual do preço médio entre os meses de julho e agosto foi de +4,2%, mostrando que houve uma alta significativa de preços na categoria como um todo.

	Preço Médio			
	Jul/2000	Ago/2000	DifABS	Dif%
Total Sabonetes	0.58	0.60	0.02	4.2%

Visão 5.2. Comparação entre os preços médios entre dois meses.

Logo, houve uma baixa de volume em função de uma alta de preços. Por que os preços subiram? Existem diversas razões para isto. Na verdade, a análise da categoria como um todo não é muito adequada para visualizar a questão: cada fabricante pode ter adotado uma política de preços diferente e que, somadas, levaram a estes resultados. Assim sendo, é interessante analisar os fabricantes individualmente. Isso também pode ser entendido como uma regra de negócio (Regra 5.4). Para esta análise, a Regra 5.5 é um caso particular da Regra 5.4.

Regra: Drill-Down

"Todo valor de variável em uma categoria depende dos valores desta mesma variável nas categorias inferiores àquela categoria, independente da hierarquia".

Regra 5.4. Regra OLAP que define a capacidade de drill-down.

Regra: Drill-Down Fabricantes

"O volume total de sabonete depende dos volumes de cada fabricante de sabonete".

Regra 5.5. Caso particular de drill-down para fabricantes de sabonete.

Na verdade, esta regra deriva do modelo dimensional, que define que as variáveis computadas em categorias agregadas são definidas a partir das categorias inferiores. Em outras palavras, esta regra indica que pode-se fazer a operação de drill-down para ver a informação em mais detalhes. No caso do exemplo, o drill-down seria sobre a dimensão produto, usando a hierarquia de fabricantes (Visão 5.3).

	Volume de Vendas			
	Jul/2000	Ago/2000	DifABS	Dif%
Total Sabonetes	161,568	156,202	-5,366	-3.3%
Amaro	541	506	-35	-6.5%
Brazuca	23,283	27,387	4,104	17.6%
Constantino	15,271	15,732	461	3.0%
Genaro	95,544	84,151	-11,393	-11.9%
João Antônio	1,457	2,081	624	42.8%
Montes Claros	3,088	6,076	2,988	96.8%
Outros	1,465	1,990	525	35.8%
Princestone	684	652	-32	-4.7%
Queen	4,031	4,054	23	0.6%
Serrinha	678	928	250	36.9%
Soap-Soap	15,526	12,645	-2,881	-18.6%

Visão 5.3. Contraste entre os volumes dos fabricantes em dois meses.

A partir desta visão, nota-se que alguns fabricantes ganharam, enquanto outros perderam neste processo. Quem ganhou e perdeu mais? Isto pode ser facilmente verificado através de um ranking entre as variações de volume, ou seja, visualizar os volumes ordenados pela diferença de volume. A Visão 5.4 ilustra o resultado desta ordenação.

	Volume de Vendas			
	Jul/2000	Ago/2000	DifABS	Dif%
Total Sabonetes	161,568	156,202	-5,366	-3.3%
Brazuca	23,283	27,387	4,104	17.6%
Montes Claros	3,088	6,076	2,988	96.8%
João Antônio	1,457	2,081	624	42.8%
Outros	1,465	1,990	525	35.8%
Constantino	15,271	15,732	461	3.0%
Serrinha	678	928	250	36.9%
Queen	4,031	4,054	23	0.6%
Princestone	684	652	-32	-4.7%
Amaro	541	506	-35	-6.5%
Soap-Soap	15,526	12,645	-2,881	-18.6%
Genaro	95,544	84,151	-11,393	-11.9%

Visão 5.4. Contraste entre volumes dos fabricantes ordenados pela diferença absoluta.

Vê-se então que o fabricante que mais ganhou volume foi “Brazuca” (+4.010), enquanto o fabricante que mais perdeu foi “Genaro”.(-11.393). É importante notar que a variável de contraste usada na ordenação foi a diferença absoluta, e não a diferença percentual. A razão disto é que a diferença percentual causaria distorções na comparação entre volumes. Por exemplo, o fabricante “Montes Claros” teve um acréscimo de volume de +96,8%, mas sua diferença absoluta de volume foi de apenas +2.988, enquanto o fabricante “Brazuca” teve aumento percentual de +17,6%, mas seu volume absoluta cresceu +4.104. Logo, “Brazuca” cresceu mais que “Montes Claros” mas a diferença percentual não registra isto.

Porém, a categoria de sabonetes perdeu –3,3% de volume, de modo que esta comparação pode conter distorções. Na verdade, a forma mais correta de avaliar a redistribuição de volumes entre os fabricantes é através do *market share*. *Market share* é o percentual de participação de um fabricante dentro do mercado em termos de volume de vendas. Em outras palavras, é o percentual do volume de cada fabricante dentro do volume total. A Visão 5.5 ilustra o contraste de *market share* entre os fabricantes. É

interessante notar que o *market share* é um valor percentual, de modo que sua diferença absoluta também resulta em um valor percentual.

	Market Share			
	Jul/2000	Ago/2000	DifABS	Média
Total	100.0%	100.0%	0.0%	100.0%
Brazuca	14.4%	17.5%	3.1%	16.0%
Montes Claros	1.9%	3.9%	2.0%	2.9%
Constantino	9.5%	10.1%	0.6%	9.8%
João Antônio	0.9%	1.3%	0.4%	1.1%
Outros	0.9%	1.3%	0.4%	1.1%
Serrinha	0.4%	0.6%	0.2%	0.5%
Queen	2.5%	2.6%	0.1%	2.5%
Princestone	0.4%	0.4%	0.0%	0.4%
Amaro	0.3%	0.3%	0.0%	0.3%
Soap-Soap	9.6%	8.1%	-1.5%	8.9%
Genaro	59.1%	53.9%	-5.3%	56.5%

Visão 5.5. Contraste de fabricantes pelo *market share* (ordenados pela diferença absoluta).

O contraste de *market share* confirma que o fabricante que mais ganhou volume entre estes meses foi “Brazuca”, enquanto o que mais perdeu foi “Genaro”. É importante notar que estes são justamente os fabricantes com maior *market share* do mercado: o *market share* médio de “Genaro” entre estes 2 meses é igual a 56,5% e o *market share* médio de “Brazuca” é 16%. Deste modo, apenas estes dois fabricantes acumulam mais de 70% de todo o mercado. Depois de “Genaro” e “Brazuca”, os fabricantes que mais perderam e ganharam volume nestes dois meses foram “Soap-Soap” e “Montes Claros” (-1,5% e +2,0% respectivamente); mas estes dois fabricantes juntos somam pouco mais de 10% de *market share* médio apenas. Isto pode ser formalizado através da Regra 5.6.

Regra: Fabricantes mais importantes

"A influência de cada fabricante sobre a categoria de sabonetes é proporcional ao *market share* do fabricante".

Regra 5.6. Como identificar os fabricantes de maior influência.

Desta maneira, pode-se afirmar que a grande influência no comportamento da categoria se deve aos fabricantes "Genaro" e "Brazuca". Embora os outros fabricantes tenham sua influência, eles podem ser deixados de lado – pelo menos em uma análise inicial do quadro. Ou seja, por simplicidade, a análise pode prosseguir considerando apenas os fabricantes "Genaro" e "Brazuca". Isto pode ser obtido através de uma operação de *slice* nas visões. A Visão 5.6 ilustra o resultado do *slice* no contraste de volume.

	Volume de Vendas			
	Jul/2000	Ago/2000	DifABS	Dif%
Total Sabonetes	161,568	156,202	-5,366	-3.3%
Genaro	95,544	84,151	-11,393	-11.9%
Brazuca	23,283	27,387	4,104	17.6%

Visão 5.6. Contraste de volume entre os fabricantes que mais interessam.

Uma vez que a análise está mais focada, pode-se procurar entender melhor o comportamento dos preços de cada fabricante. A Visão 5.7 ilustra o contraste de preços praticados por "Genaro" e "Brazuca" entre estes dois meses. Enquanto "Genaro" aumentou seu preço médio em +8,6% (uma alta considerável), "Brazuca" manteve-se estável. Logo, a Regra 5.3 confirma-se em "Genaro", pois ele teve aumento de preço e perda de volume. Porém, como explicar o aumento de volume de "Brazuca"? Ele manteve o preço estável, mas seu volume subiu.

	Preço Médio			
	Jul/2000	Ago/2000	DifABS	Dif%
Total Sabonetes	0.58	0.60	0.02	4.2%
Genaro	0.58	0.63	0.05	8.6%
Brazuca	0.29	0.29	0.00	0.0%

Visão 5.7. Contraste de preço médio entre os fabricantes mais influentes.

Na verdade, a comparação do preço praticado pode esconder o efeito de promoções realizadas durante estes períodos de tempo. Isto significa que o aumento de preços entre os meses de julho e agosto pode ser decorrente de o mês de julho ter tido mais promoções que o mês de agosto. Deste modo, o preço "oficial" dos produtos não teria variado. Este preço "oficial" é conhecido como preço regular. O preço regular é um preço estimado como sendo aquele que o consumidor percebe como sendo o preço normal do produto, ou seja, o preço do produto fora de promoção. Assim sendo, o preço praticado de um artigo pode ser entendido como a composição de preços regulares e preços eventuais (promoções ou aumentos) ao longo de um mês. Logo, a comparação entre o preço praticado e o preço regular pode indicar intenções ou motivos por trás das variações de preço. A Visão 5.8 ilustra os preços regulares médios de "Genaro" e "Brazuca".

	Preço Regular Médio			
	Jul/2000	Ago/2000	DifABS	Dif%
Total Sabonetes	0.58	0.60	0.02	3.7%
Genaro	0.59	0.63	0.04	6.8%
Brazuca	0.29	0.30	0.01	3.4%

Visão 5.8. Contraste de preço entre os fabricantes mais influentes.

Por esta visão, nota-se que os preços regulares médios de "Genaro" e "Brazuca" subiram, sendo que "Genaro" aumentou consideravelmente o seu (+6,8%) enquanto "Brazuca" teve um aumento menor (+3,4%). O aumento de "Genaro" foi novamente

bem maior que o de “Brazuca”, indicando mais uma vez que o aumento de preço é a provável causa da queda de volume.

Ao analisar a diferença entre os preços praticados e preços regulares de cada fabricante, notam-se as diferentes estratégias de cada um. “Brazuca” teve aumento no preço regular, mas o preço praticado permaneceu quase o mesmo. Isto significa que “Brazuca” optou por fazer muitas promoções no mês de agosto – possivelmente atrás do aumento de *market share* que acabou obtendo.

Já “Genaro” tomou a direção oposta: mesmo com o preço regular já aumentado, subiu ainda mais seus preços. Qual seria a razão disto? Um analista experiente diria que existem duas causas mais prováveis: um aumento no preço de custo sendo repassado para os consumidores, ou então a necessidade de aumentar as margens de lucro (ou seja, ganhar mais dinheiro). Qual destes foi o motivo real?

O ScanOne disponibiliza variáveis para análise financeira. As principais delas são faturamento, margem e margem percentual. O faturamento é a quantidade de dinheiro creditada ao supermercado mediante a venda dos artigos. Em outras palavras, é o total de dinheiro pago pelos clientes das lojas da rede de supermercados. Assim sendo, o faturamento é igual ao preço de um artigo multiplicado pela quantidade de artigos vendidos, conforme indica a Regra 5.7.

Regra: Faturamento

Faturamento = Preço * Volume

Regra 5.7. Fórmula de faturamento.

A margem é a quantidade de dinheiro que realmente fica com o supermercado, já que este tem que pagar aos fabricantes pelos artigos que disponibiliza a seus clientes.

Ou seja, a margem é igual ao faturamento menos o preço de custo dos artigos para o supermercado (Regra 5.8). É importante não confundir margem com lucro. O lucro é a margem menos os custos fixos do supermercado (salários, luz, aluguel etc). A margem lida apenas com os custos variáveis dos artigos. Estes custos são estipulados principalmente pelos fabricantes e distribuidores de produtos.

Regra: Margem

Margem = Faturamento – Custo

Regra 5.8. Fórmula da margem.

Outra variável importante para análise é a margem percentual. A margem percentual mede o quanto rende a venda de cada artigo. Em outras palavras, a margem percentual indica o quão vantajoso foi vender um determinado artigo. A margem percentual é igual à margem dividida pela faturamento (Regra 5.9).

Regra: Margem Percentual

Margem% = Margem / Faturamento

Regra 5.9. Fórmula da margem percentual.

Uma vez que estas variáveis estão definidas, pode-se prosseguir com a análise. É desejado encontrar as razões que levaram "Genaro" a elevar tão fortemente seu preço, prejudicando seu volume de vendas.

	Faturamento			
	Jul/2000	Ago/2000	DifABS	Dif%
Total Sabonetes	1,007,703	1,011,530	3,827	0.4%
Genaro	601,485	571,990	-29,495	-4.9%
Brazuca	75,281	88,636	13,355	17.7%

Visão 5.9. Contraste de faturamento.

Ao observar o faturamento dos fabricantes (Visão 5.9), nota-se que “Genaro” caiu –4,9%, enquanto “Brazuca” subiu +17,7%. Logo, embora “Genaro” tenha aumentado consideravelmente seu preço, a perda de volume foi tão grande que fez com menos dinheiro entrasse na companhia no 2º mês. Já o aumento de faturamento de “Brazuca” foi bastante alto (+17,7%).

Porém, ao analisar a margem de cada fabricante (Visão 5.10), vê-se que, na verdade, “Genaro” conseguiu colocar mais 3,0% de dinheiro em caixa; “Brazuca” também, sendo que seu aumento de receita foi de +15,8%. Logo, isso indica que talvez “Genaro” tenha mesmo aumentado seus preços em busca de mais margem.

	Margem			
	Jul/2000	Ago/2000	DifABS	Dif%
Total Sabonetes	167,317	183,157	15,840	9.5%
Genaro	85,984	88,593	2,609	3.0%
Brazuca	15,536	17,984	2,448	15.8%

Visão 5.10. Contraste de margem.

Outra forma de analisar a margem é através da margem percentual, conforme ilustrado na Visão 5.11. Nesta visão, vê-se que na verdade “Brazuca” perdeu (–1,7%), enquanto “Genaro” ganhou (+8,3%). Logo, os artigos de “Genaro” ficaram muito mais lucrativos, enquanto os artigos de “Brazuca” ficaram um pouco menos.

	Margem %			
	Jul/2000	Ago/2000	DifABS	Dif%
Total Sabonetes	16.6%	18.1%	1.5%	9.1%
Genaro	14.3%	15.5%	1.2%	8.3%
Brazuca	20.6%	20.3%	-0.3%	-1.7%

Visão 5.11. Contraste de margem percentual.

Porém, ainda não se sabe se este aumento de preços está em parte repassando custos ao consumidor. A margem percentual ajuda a esclarecer isso. Como a diferença

percentual de preço de “Genaro” (+8,6%) está muito parecida com sua diferença percentual da margem percentual (+8,3%), então pode-se dizer que praticamente não houve repasse de custos para o consumidor. Logo, “Genaro” teve grande aumento de preços para ter aumento de margem.

5.5. Modelo de Dados do ScanOne

O ScanOne é uma aplicação OLAP construída sobre o Maestro. Deste modo, o modelo de dados do ScanOne é uma especialização do modelo de dados do Maestro. A Figura 5.1 indica como é este modelo. FatoScanOne e CuboScanOne são ocorrências especiais de FatoMaestro e CuboMaestro respectivamente. A diferença é que no ScanOne apenas algumas Dimensões, Categorias e Variáveis existem.

As Dimensões que existem no ScanOne são: Geografia, Produto, Embalagem, Data e Periodicidade. Por se tratar de um universo de possibilidades bem determinadas, estas Dimensões foram definidas em classes específicas. De maneira semelhante, as Categorias de cada Dimensão do ScanOne também têm uma classe própria: CategoriaGeografia, CategoriaProduto, CategoriaEmbalagem, CategoriaData e CategoriaPeriodicidade. Desta maneira, um FatoScanOne contém um exemplo de cada Categoria. A união de todas estas Categorias forma uma chave para o valor de uma Variável.

As Variáveis são modeladas como operações de FatoScanOne. Estas operações são funções, isto é, retornam valores de dados. Portanto, para cada instância de FatoScanOne, tem-se a combinação das Categorias do ScanOne e um valor para cada Variável determinada. Um CuboScanOne é uma coleção de várias instâncias de FatoScanOne.

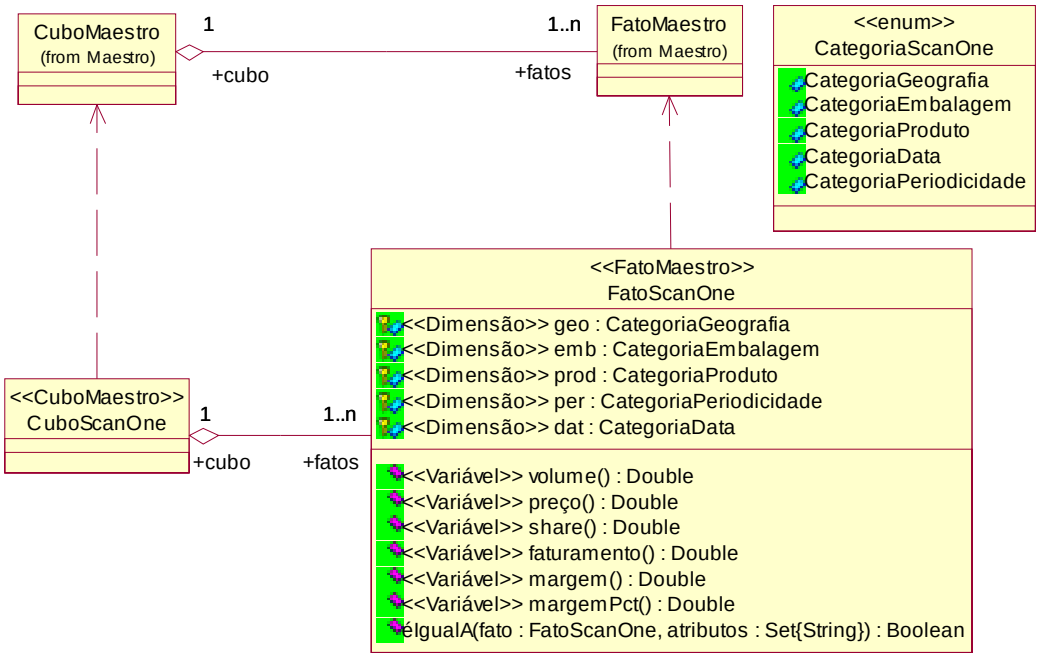

Figura 5.1. Relacionamento entre os modelos do Maestro e do ScanOne.

5.6. *Um Modelo de Regras de Negócio para ScanOne*

Escrever regras de negócio para o ScanOne não é uma tarefa simples pois as regras devem ser expostas, no sentido de serem manipuláveis computacionalmente (conforme foi discutido na seção 2.3.2). Uma forma de se obter esta "exposição" é definir as regras através de um modelo de dados. Desta maneira, as regras de negócio teriam atributos com os quais sistemas de informação ou *parsers* poderiam interagir.

Esta seção define um modelo de dados para isto. Com base neste modelo, regras de negócio terão atributos definidos explicitamente. Porém, conforme discutido na seção 2.4.1, o modelo de classes UML não tem poder de expressão suficiente para definir regras de maneira satisfatória. Portanto, as regras propriamente ditas serão definidas em expressões OCL, sendo que estas expressões farão referência aos atributos das regras de negócio.

A Figura 5.2 ilustra a classe Regra, usada para representar regras de negócio. Como as expressões em OCL de um diagrama UML devem ser sempre relacionadas a um contexto específico (ver seção 2.5), convencionou-se que o contexto das regras de negócio em OCL seja a própria classe Regra – ou alguma classe descendente.

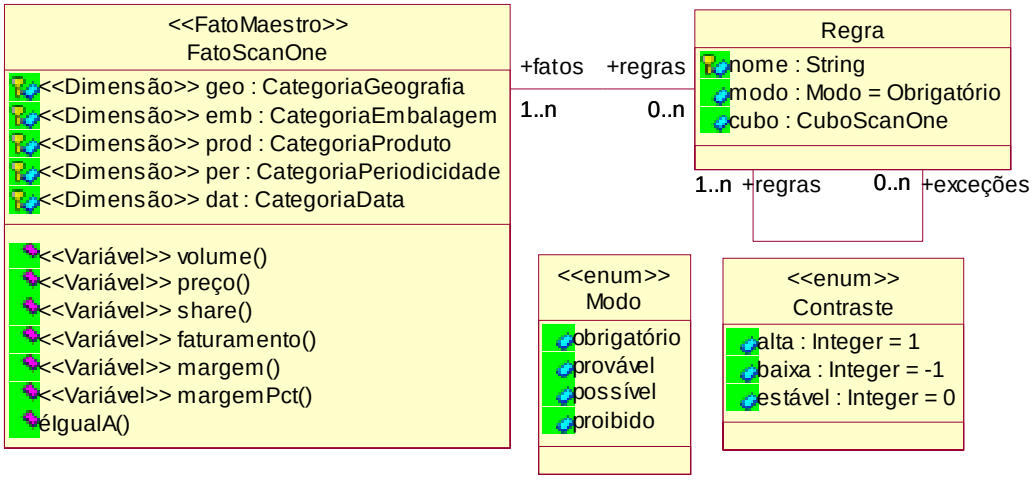

Figura 5.2. Fatos do ScanOne e regras de negócio.

Assim sendo, é necessário diferenciar as diversas regras de negócio que estão no contexto da classe Regra. Para tanto, usa-se o atributo 'nome' como identificador único das instâncias de Regra. Por exemplo, seja a seguinte Regra: "o valor de volume de vendas nunca é negativo". Esta Regra pode ser expressa em OCL da seguinte maneira:

```
context Regra inv:
  if
    self.Nome = 'volume-não-negativo'
  then
    FatoScanOne.allInstances->forAll( F | F.Volume() >= 0 )
  endif
```

Além disso, a classe Regra tem um atributo chamado modo, que indica o grau de obrigatoriedade da regra. O modo de uma regra admite os seguintes valores: obrigatório, provável, possível e proibido. Os modos obrigatório e proibido denotam que a regra é mandatária, isso é, a regra tem que ser cumprida. Nestes modos, a regra tem semântica semelhante ao verbo modal *must* da Língua Inglesa. Já os modos provável e possível

denotam tendências ou incertezas, sendo mais próximos aos verbos modais *should* e *may* respectivamente.

Na verdade, FatoScanOne funciona como uma tabela de fatos em um esquema estrela, em que as categorias formam o conjunto de chaves e as operações são as variáveis que dependem das dimensões. Porém, os cubos construídos a partir desta estrutura contêm somente aquelas dimensões mencionadas. Para cada combinação de dimensionalidade que aparecer, terá de haver uma outra classe que componha as dimensões adequadamente. Um exemplo disso aparece quando há fatos derivados de análises de contraste entre dois outros fatos. A Figura 5.3 é um exemplo de fato derivado por contraste a partir de dois outros.

F1: "O volume de janeiro é igual a 200"

F2: "O volume de fevereiro é igual a 300"

Deriv [F1, F2]: "A diferença percentual entre janeiro e fevereiro é de +50%"

Figura 5.3. Exemplo de fato sendo derivado por contraste.

Este fato derivado tem a mesma dimensionalidade dos dois fatos anteriores, acrescido de mais uma dimensão de Data, indicando o tempo de referência para o qual a diferença percentual está sendo medida. Para capturar informações como esta, é necessário criar uma novo tipo de fato. A Figura 5.4 ilustra como este caso foi representado no modelo de dados do ScanOne.

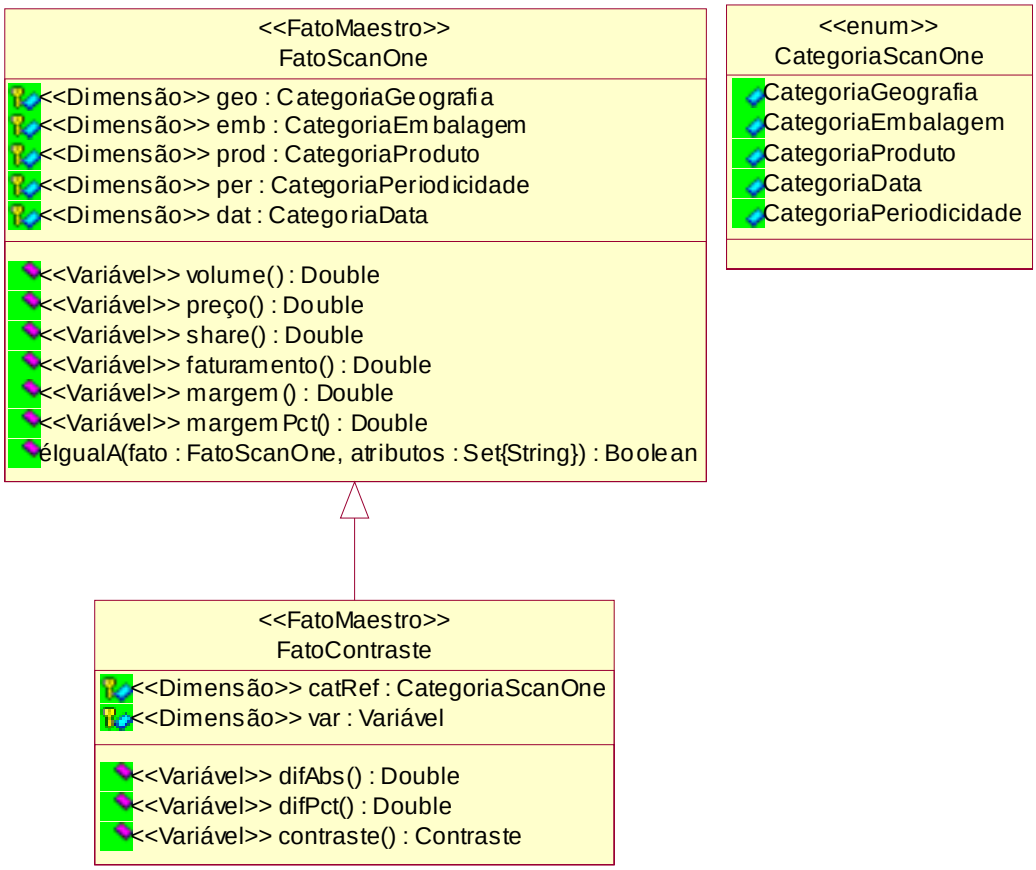

Figura 5.4. Fatos simples e fatos de contraste.

Os fatos que têm essa característica serão chamados de fatos de contraste, e são representados no modelo através da classe FatoContraste. Esta classe herda de FatoScanOne todas as suas dimensões (na forma de atributos de categoria), mas também acrescenta duas outras categorias a sua chave:

- a categoria 'catRef', da classe CategoriaScanOne, representando a categoria usada como referência no contraste.
- a variável 'var', da classe Variável, que indica a grandeza sendo contrastada.

Em outras palavras, esta classe mantém fatos que são a comparação (ou derivação) de dois outros fatos muito parecidos, variando apenas o valor de uma das

dimensões. O valor "antigo" é o que fica na categoria de referência (catRef), enquanto o valor "novo" fica em alguma das categorias herdadas de FatoScanOne.

FatoContraste tem outras Variáveis que têm os valores expostos através de operações. Elas são diferença absoluta, diferença percentual e contraste - difAbs(), difPct() e contraste(). Cada uma das operações recebe como parâmetro uma instância de Variável. Isso acontece por que estes contrastes podem ocorrer em qualquer Variável. Por exemplo, pode-se calcular a diferença absoluta de volume, preço, *market share* etc.

Dentre as Variáveis de FatoContraste, é interessante notar a operação contraste(). Ela é um tipo de comparação entre Variáveis que, ao contrário de difAbs() e difPct(), não realiza cálculos matemáticos, e sim mantém informação qualitativa sobre a variação das Variáveis. Por exemplo, tomando os mesmos dois fatos sobre volume da Figura 5.3, pode-se dizer que: "volume teve alta entre janeiro e fevereiro".

Este valor de alta é uma interpretação obtida a partir da análise dos dois fatos anteriores - não é o resultado de uma computação automática. Isto deve ser observado porque nem sempre uma variação deve ser levada em conta – isto depende da escala que os números têm em sua aplicação prática. Um exemplo disso está na Figura 5.5. Embora tenha havido uma variação positiva entre fevereiro e março, ela foi muito pouco significativa para fins práticos do negócio. Por isto o analista de negócios optou por classificar este contraste como “estável”. É claro que há meios de automatizar certas classificações; mas isso certamente passa pela definição de novas regras de negócio especializadas em colocar valores-limite para as Variáveis em determinadas combinações de categoria.

F2: "O volume de fevereiro é igual a 300"

F3: "O volume de março é 300,01"

Deriv [F2, F3]: "O volume entre fevereiro e março se manteve estável"

Figura 5.5. Exemplo de fato de contraste dependente de interpretação.

5.7. Extraindo Regras do Exemplo de Análise

Conforme o exemplo da seção 5.4 demonstrou, durante o processo de análise, o analista de negócios combina regras de negócio e conhecimentos próprios para interpretar e julgar as informações apresentadas pelo ScanOne. O problema disto é que as regras de negócio usadas fazem parte desse conhecimento próprio do analista já que a maior parte das regras não é descrita e/ou formalizada adequadamente. Esta seção visa explicitar as regras de negócio usadas no exemplo em linguagem formal OCL.

5.7.1. Exemplo: Fatos Simples

Cada célula multidimensional expressa um fato. Por exemplo, a Visão 5.1 contém o fato: "o volume de vendas da categoria de sabonetes em julho de 2000 foi de 161.566". Na verdade, este fato contém outras dimensões que estão implícitas: "o volume de vendas da categoria de sabonetes, em todas as embalagens, em todo o Brasil, em junho de 2001 foi de 161.566". Estas dimensões não aparecem antes porque seus valores são totais de hierarquia; ou seja, estas dimensões não são importantes (para este fato em particular). Usando o modelo de dados do ScanOne, pode-se expressar este fato em OCL da seguinte maneira:

```
context FatoScanOne inv:
  self.prod.código = 'Sabonete' and
  self.emb.código = 'Total' and
  self.geo.código = 'Brasil' and
  self.per.código = 'Mensal' and
  self.dat.código = '01/07/2000' and
  self.volume() = 161566
```

Assim como este fato, existem diversos outros que podem ser expressos de maneira semelhante. Por exemplo, o *market share* para estas mesmas categorias de dimensão seria:

```
context FatoScanOne inv:
  self.prod.código = 'Sabonete' and
  self.emb.código = 'Total' and
  self.geo.código = 'Brasil' and
  self.per.código = 'Mensal' and
  self.dat.código = '01/07/2000' and
  self.share() = 100%
```

Ainda na Visão 5.1, é interessante notar que os fatos expressos nas colunas “Dif ABS” e "Dif%" indicam o resultado da comparação de volume entre os meses de julho e agosto. Estes fatos não são como os fatos de volume e *market share* por que eles são resultado da comparação de dois fatos diferentes. Portanto, eles são fatos de contraste. A seguinte expressão em OCL indica qual é a diferença absoluta de volume entre os meses de julho e agosto de 2000:

```
context FatoContraste inv:
  self.prod.código = 'Sabonete' and
  self.emb.código = 'Total' and
  self.geo.código = 'Brasil' and
  self.per.código = 'Mensal' and
  self.dat.código = '01/08/2000' and
  self var.nome = 'volume' and
  self.catRef.oclIsTypeOf(CategoriaData) and
  self.catRef.código = '01/07/2000'
  self.difAbs() = -5366
```

A expressão usa o atributo catRef para determinar a referência usada para calcular a diferença absoluta. Neste caso, o fato é referente ao mês de julho/2000; e o fato de contraste em si é datado de agosto/2000. Além disso, a expressão faz o nome do

atributo ‘var’ ser igual a ‘volume’, indicando que a diferença se dá sobre valores de volume. A diferença absoluta tem seu valor em difAbs().

Assim como existe a diferença absoluta, pode-se verificar também a diferença percentual entre os dois fatos. A diferença percentual também é uma variável de contraste. A expressão em OCL de uma diferença percentual de volume seria a seguinte:

```
context FatoContraste inv:
  self.prod.código = 'Sabonete' and
  self.emb.código = 'Total' and
  self.geo.código = 'Brasil' and
  self.per.código = 'Mensal' and
  self.dat.código = '01/08/2000' and
  self var.nome = 'volume' and
  self.catRef.oclIsTypeOf(CategoriaData) and
  self.catRef.código = '01/07/2000'
  self.difPct(Volume) = -5.6%
```

A diferença percentual é muito útil para a realização de análises comparativas, uma vez que ela normaliza os concorrentes em função de seu desempenho anterior.

Generalizando as variáveis de contraste, existe a operação contraste(), que permite indicar a qualidade de uma variação, e não sua quantidade. Por exemplo, para os contrastes observados até aqui, pode-se fazer expressar o seguinte:

```
context FatoContraste inv:
  self.prod.código = 'Sabonete' and
  self.emb.código = 'Total' and
  self.geo.código = 'Brasil' and
  self.per.código = 'Mensal' and
  self.dat.código = '01/08/2000' and
  self var.nome = 'volume' and
  self.catRef.oclIsTypeOf(CategoriaData) and
  self.catRef.código = '01/07/2000'
  self.contraste() = Contraste::baixa
```

Esta expressão indica que houve baixa de volume entre os meses de julho e agosto de 2000. A operação contraste() retorna uma instância da classe Contraste. as instâncias da classe Contraste são: alta, baixa e estável. É interessante notar que o valor designado para Contraste pode não ser computável automaticamente, já que podem necessitar a interpretação de um analista. Por exemplo, se uma diferença absoluta for

igual +500, isto não significa necessariamente uma “alta” porque o valor de +500 pode ser insignificante dentro da escala (o analista pode considerar esse contraste como “estável”). Logo, trata-se de uma variável dependente do processo OLAP.

5.7.2. Exemplo: Regra “Preço-Volume”

Conhecidas estas relações entre os objetos, pode-se expressar algumas regras de negócio. Neste modelo, as regras de negócio são expressões em OCL pertencentes à classe Regra. Cada expressão OCL expressa no contexto da classe Regra corresponde a uma regra de negócio. Por exemplo, seja a seguinte regra de negócio:

Regra: Preço-Volume
“Se houver baixa de preço entre datas então é provável que haja alta de volume entre estas mesmas datas”.

Regra 5.10. Definição da regra “Preço-Volume” em linguagem natural.

Esta regra está expressa em linguagem natural, mas pode também ser expressa em linguagem formal. Uma vez que existe o modelo de dados em UML definindo a estrutura OLAP do ScanOne, é possível usar a OCL como lógica de expressão das regras de negócio que atuam sobre estas informações. No caso desta regra, sua expressão em OCL seria a seguinte:

```
context Restrição inv:
  if
    self.nome = 'Preço-Volume'
  then
    FatoContraste.allInstances()->forAll( f1, f2 : FatoContraste |
      if
        f1.éIgualA(f2, FatoContraste.attributes() - Set{'var'}) and
        f1.catRef.oclIsTypeOf(CategoriaData) and
        f2.catRef.oclIsTypeOf(CategoriaData) and
      then self.modo = Modo::provável or (
        (  f1.var.nome = 'preço' and
           f1.contraste() = Contraste::baixa )
        implies
        (  f2.var.nome = 'volume' and
           f2.contraste() = Contraste::alta )
      )
      else Boolean::false
      endif
    )
  else Boolean::false
  endif
```

Esta expressão OCL é uma invariante da classe Restrição, uma especialização da classe Regra – seguindo a classificação de Ross (ROSS, 1998). Esta regra foi considerada uma restrição porque estabelece uma simples relação entre dois fatos de contraste: um de preço, e outro de volume. Ela não produz fatos novos (como fazem as regras de produção), e nem gera novos eventos (como fazem as regras de projeção). Portanto, por eliminação, ela é uma regra de restrição.

Analisando a expressão desta regra em OCL, vê-se que, inicialmente, testa-se se o nome seu é igual a ‘Preço-Volume’. Em outras palavras, testa-se se a instância de regra em questão tem aquele nome (o atributo nome identifica unicamente as instâncias de Regra). Por convenção, todas as expressões de Regra em OCL terão esta forma. Assim sendo, a regra pode ser decomposta da seguinte forma:

```
context Restrição inv:
  if self.nome = <nome-da-regra>
  then <regra-propriamente-dita>
  else Boolean::false
  endif
```

Sendo satisfeita esta condição, inicia-se a definição da regra propriamente dita. Esta regra vale para todos os pares de instâncias de FatoContraste, simbolizados pelas

variáveis auxiliares f1 e f2. O conjunto de todas as instâncias de um tipo é obtido através do método allInstances(), que é definido pela própria OCL como válido para qualquer tipo. Na prática, este teste de ocorrência não varrerá todas as instâncias de FatoContraste pois isso significaria varrer um cubo histórico inteiro. Na verdade, estas regras de negócio são expressões conceituais, ou seja, estão no nível do negócio propriamente dito: seu objetivo não é a implementação de sistemas, mas sim definir formalmente o significado de uma regra de negócio.

```
context Restrição inv:
  if
    self.nome = 'Preço-Volume'
  then
    FatoContraste.allInstances()->forAll( f1, f2 : FatoContraste |
      if
         <restrições sobre f1 e f2>
      then
         <relação de causalidade f1 => f2>
      else Boolean::false
      endif
    )
  else Boolean::false
  endif
```

O método forAll faz com que se desloque o foco para dois fatos quaisquer f1 e f2. Dados estes fatos, o restante da regra é um grande teste sobre f1 e f2. Neste teste, verifica-se se f1 e f2 obedecem a algumas restrições que envolvem os dois fatos. Se estas restrições forem obedecidas, então o resultado da expressão OCL passa a ser uma relação de causalidade entre f1 e f2, ou seja, uma expressão f1 implica logicamente uma expressão em f2.

Estas restrições sobre f1 e f2 aparecem no texto da regra de negócio em linguagem natural de maneira muito sutil. Por exemplo, embora não esteja explicitamente escrito, as geografias, embalagens, produtos e periodicidades dos dois fatos devem ser os mesmos: caso contrário, a comparação entre os dois contrastes não faz sentido. Esta informação é omitida para que a frase não fique muito “carregada” para o leitor. Na verdade, o leitor necessita apenas saber que os contrastes entre fatos

são referentes a datas, pois isto não pode ser subentendido. Logo, diferente das outras dimensões, a informação de data é explicitamente citada como sendo a mesma para os dois fatos: “se (...) entre datas então (...) entre estas mesmas datas”. A expressão “estas mesmas” faz a conexão entre as datas de cada oração.

Para formalizar esta relação entre as categorias de dimensão de cada fato, bastaria criar uma expressão de junção entre cada categoria de dimensão dos fatos, por exemplo:

```
...
f1.geo = f2.geo and
f1.emb = f2.emb and
f1.prod = f2.prod and
f1.per = f2.per and
f1.catRef = f2.catRef and
...
```

No entanto, ao ser expressa desta maneira, a frase em OCL faria menção às informações omitidas na regra em linguagem natural. Isto poderia dificultar uma futura tradução entre linguagens natural e formal, pois não haveria correspondência entre os termos utilizados em cada uma das formas de expressão.

Pensando nisto, buscou-se uma maneira de expressar a regra de negócio em OCL de modo a torná-la mais próxima à expressão da regra em linguagem natural. Assim sendo, as expressões de junção entre categorias foram substituídas por uma expressão equivalente logicamente, mas que não exibe as categorias omitidas na frase original.

Para ajudar nisto, utilizou-se o método éIgualA() da classe FatoScanOne e, por herança, também pertencente à classe FatoContraste. Este método retorna verdadeiro se dois fatos têm os mesmos valores para determinadas categorias, e falso caso contrário. A assinatura de éIgualA() é a seguinte:

```
FatoScanOne::éIgualA(fato: FatoScanOne, atributos: Set(String)) : Boolean
```

O método depende de dois parâmetros. O primeiro é a instância de FatoScanOne (ou FatoContraste) que está sendo comparada. O segundo é um conjunto de nomes de atributos que se deseja comparar. Internamente, o método constrói uma expressão de junção entre os atributos de categoria contidos no conjunto ‘atributos’ e retorna se há ou não coincidência em seus valores. Portanto, pode-se substituir aquela expressão de junção descrita anteriormente pela seguinte:

```
...
f1.éIgualA(f2, Set{'geo', 'emb', 'prod', 'per', 'catRef'}) and
...
```

No entanto, os termos originalmente omitidos ainda aparecem na expressão OCL. Para omiti-los, pode-se usar o método attributes(), definido pelo padrão OCL (OBJECT MANAGEMENT GROUP, 1997). Este método pode ser invocado por qualquer classe, e retorna o conjunto dos nomes dos atributos desta classe. Assim, pode-se fazer a mesma chamada ao método éIgualA(), mas mencionando apenas os atributos de categoria que não são iguais:

```
...
f1.éIgualA(f2, FatoContraste.attributes() - Set{'var'}) and
...
```

Outra condição que não é explicitamente colocada na frase em linguagem natural é que os contrastes f1 e f2 têm a categoria data como referência: diz-se sutilmente que o contraste é “entre datas”. Em OCL isto pode ser formalmente descrito com auxílio do método oclIsTypeOf(), que indica se um objeto pertence a uma determinada classe:

```
  ...
  f1.catRef.oclIsTypeOf(CategoriaData) and
  f2.catRef.oclIsTypeOf(CategoriaData) and
  ...
```

Portanto, a expressão em OCL para a regra de negócio teria o seguinte aspecto:

```
context Restrição inv:
  if
    self.nome = 'Preço-Volume'
  then
    FatoContraste.allInstances()->forAll( f1, f2 : FatoContraste |
      if
        f1.éIgualA(f2, FatoContraste.attributes() - Set{'var'}) and
        f1.catRef.oclIsTypeOf(CategoriaData) and
        f2.catRef.oclIsTypeOf(CategoriaData) and
      then
        <relação de causalidade f1 => f2>
      else Boolean::false
      endif
    )
  else Boolean::false
  endif
```

Uma vez formalizadas as restrições sobre f1 e f2, pode-se então descrever a relação de causalidade que existe entre os dois fatos. No caso desta regra, a relação é: baixa de preço implica alta de volume. Isto pode ser facilmente expresso em OCL da seguinte maneira:

```
  ...
  (  f1.var.nome = 'preço' and
     f1.contraste() = Contraste::baixa )
  implies
  (  f2.var.nome = 'volume' and
     f2.contraste() = Contraste::alta )
  ...
```

Assim sendo, a expressão em OCL ficaria assim:

```
context Restrição inv:
  if
    self.nome = 'Preço-Volume'
  then
    FatoContraste.allInstances()->forAll( f1, f2 : FatoContraste |
      if
        f1.éIgualA(f2, FatoContraste.attributes() - Set{'var'}) and
        f1.catRef.oclIsTypeOf(CategoriaData) and
        f2.catRef.oclIsTypeOf(CategoriaData) and
      then
        (  f1.var.nome = 'preço' and
           f1.contraste() = Contraste::baixa )
        implies
        (  f2.var.nome = 'volume' and
           f2.contraste() = Contraste::alta )
      else Boolean::false
      endif
    )
  else Boolean::false
  endif
```

Porém, falta ainda um detalhe que é tecnicamente importante. Esta regra não é mandatária pois seu modo é igual a Contraste::provável. Ou seja, ela pode não ser verdadeira para todos os FatosContraste de um cubo. Uma vez que a OCL não permite que existam instâncias que firam uma expressão invariante, é necessário adicionar uma cláusula a respeito do modo da regra.

Para tanto, será adicionada uma expressão sempre verdadeira à relação de causalidade, sendo conectada a esta através de um ou-lógico. Assim, mesmo que a relação de causalidade não se confirme, a expressão sempre será verdadeira. Para aumentar a similaridade entre a OCL e a linguagem natural, a expressão verdadeira é a seguinte:

```
...
if
  <restrições sobre f1 e f2>
then self.modo = Modo::provável or (
   <relação de causalidade f1 => f2>
)
else Boolean::false
endif
...
```

Deste modo, a expressão final em OCL para a regra de negócio 'Preço-Volume' é a seguinte:

```
context Restrição inv:
  if
    self.nome = 'Preço-Volume'
  then
    FatoContraste.allInstances()->forAll( f1, f2 : FatoContraste |
      if
        f1.éIgualA(f2, FatoContraste.attributes() - Set{'var'}) and
        f1.catRef.oclIsTypeOf(CategoriaData) and
        f2.catRef.oclIsTypeOf(CategoriaData) and
      then self.modo = Modo::provável or (
        (  f1.var.nome = 'preço' and
           f1.contraste() = Contraste::baixa )
        implies
        (  f2.var.nome = 'volume' and
           f2.contraste() = Contraste::alta )
      )
      else Boolean::false
      endif
    )
  else Boolean::false
  endif
```

5.7.3. Exemplo: Regra “Preço-Volume-Concorrente”

Uma outra regra que é bastante semelhante a anterior é a seguinte:

Regra: Preço-Volume-Concorrente

“Se houver baixa de preço para um produto entre datas então é provável que haja baixa de volume para um concorrente desse mesmo produto entre estas mesmas datas”.

Regra 5.11. Definição da regra “Preço-Volume-Concorrente” em linguagem natural.

Seguindo os mesmos princípios abordados no exemplo anterior, esta regra de negócio também pode ser expressa em OCL da seguinte maneira:

```
context Restrição inv:
  if
    self.nome = 'Preço-Volume-Concorrente'
  then
    FatoContraste.allInstances()->forAll( f1, f2 : FatoContraste |
      if
        f1.éIgualA(f2, FatoContraste.attributes() - Set{'prod', 'var'}) and
        f1.prod.concorrentes()->includes(f2.prod) and
      then self.modo = Modo::provável or (
        ( f1.contraste() = Contraste::baixa and
          f1.var.nome = 'preço' )
        implies
        ( f2.contraste() = Contraste::baixa and
          f2.var.nome = 'volume' )
      )
      else Boolean::false
      endif
    )
  else Boolean::false
  endif
```

Neste caso, a regra se refere explicitamente aos produtos comparados como sendo uma informação relevante porque os produtos dos fatos de contraste têm que ser concorrentes um do outro. Assim sendo, o atributo 'prod' também é colocado na lista de atributos usados no método éIgualA(). A relação entre os dois produtos é expressa da seguinte maneira:

```
...
f1.prod.concorrentes()->includes(f2.prod) and
...
```

O atributo 'prod' pertence à classe CategoriaProduto. Esta classe possui uma função chamada concorrentes() que retorna o conjunto de instâncias de CategoriaProduto que são concorrentes do produto. Assim, pode-se usar o método includes() da classe Set para saber se o outro produto pertence ao conjunto de concorrentes do primeiro.

5.7.4. Um Template para Regras em OCL

As duas regras apresentadas têm muitas semelhanças, pois ambas seguem a mesma estrutura de formação. Na seção 2.3.3, referiu-se a este tipo de estrutura como

sendo *templates* para regra de negócio. A diferença é que, naquele contexto, *templates* foram definidos para regras em linguagem natural, e aqui, foram definidos para OCL.

Deste modo, pode-se imaginar que existe uma família de regras semelhantes, sendo que seus membros se diferenciam através de alguns parâmetros pré-definidos. Em outras palavras, trata-se de uma nova classe de regras de negócio. No caso destes exemplos, esta classe é uma especialização da classe Restrição, que será chamada de classe RegraContrasteSeEntão (Figura 5.6). Esta classe tem a missão de representar todas as regras de restrição sobre fatos de contraste que expressam causalidade entre eles, ou seja, têm a forma “se-então”.

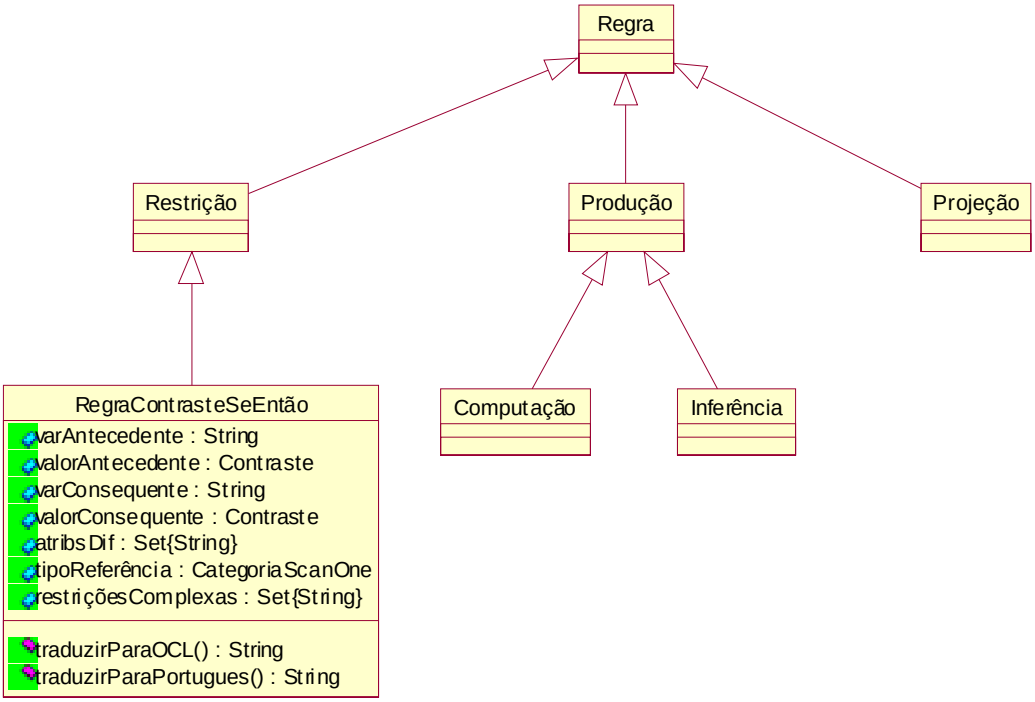

Figura 5.6. A classe especializada RegraContrasteSeEntão.

Cada instância da classe RegraContrasteSeEntão se diferencia das outras através dos valores de seus atributos, ilustrados na Figura 5.6. Deste modo, pode-se diferenciar estas regras de negócio dentro de um acervo de regras, facilitando seu tratamento

computacional. Portanto, estas regras de negócio são definidas segundo os preceitos apresentados na seção 2.3.2: são declarativas e manipuláveis computacionalmente.

Os nomes das variáveis antecedentes e conseqüentes são definidos pelos atributos 'varAntecedente' e 'varConsequente'. Já os valores de contraste que elas deverão assumir ficam em 'valorAntecedente' e 'valorConsequente'. O conjunto de nomes de categorias que devem ser diferenciados na junção entre os fatos é definido no atributo 'atribsDif'. O tipo da categoria de referência, que tem que ser o mesmo para ambos os fatos de contraste, é determinado no atributo 'tipoReferencia'. Restrições mais complexas entre os fatos de contraste devem ser expressas no atributo 'restriçõesComplexas'.

Para exemplificar isto, a instância da classe RegraContrasteSeEntão que representa a regra 'Preço-Volume' tem os seguintes valores em seus atributos:

```
nome = 'Preço-Volume'
modo = Modo::provável
varAntecedente = 'preço'
valorAntecedente = Contraste::baixa
varConsequente = 'volume'
valorConsequente = Contraste::alta
atribsDif = Set{'var'}
tipoReferencia = CategoriaData
restriçõesComplexas = Set{}
```

Já a instância da regra 'Preço-Volume-Concorrente' teria os seguintes valores:

```
nome = 'Preço-Volume-Concorrente'
modo = Modo::provável
varAntecedente = 'preço'
valorAntecedente = Contraste::baixa
varConsequente = 'volume'
valorConsequente = Contraste::baixa
atribsDif = Set{'var', 'prod'}
tipoReferencia = CategoriaData
restriçõesComplexas = Set{'<a>.prod.concorrentes()->includes(<c>.prod)'}
```

Deve-se chamar atenção para a diferença do atributo 'restriçõesComplexas' em relação aos outros. A função deste atributo é listar as restrições entre os fatos de contraste cuja semântica não é previamente conhecida. Logo, estas restrições deverão

ser expressas diretamente em OCL. Por este motivo, o atributo 'restriçõesComplexas' recebe um conjunto de *strings*, cada um contendo uma expressão OCL.

Um exemplo disto está na regra 'Preço-Volume-Concorrente'. Neste caso, existe apenas uma restrição, na qual se indica que os produtos antecedentes são concorrentes dos produtos conseqüentes. Para se referir aos fatos antecedentes e conseqüentes dentro da expressão OCL, usam-se marcações dentro do texto. Assim, foi convencionado que as marcações <a> e <c> serão usadas para indicar os objetos FatoContraste antecedentes e conseqüentes, respectivamente.

5.7.5. Traduzindo Regras para OCL e Linguagem Natural

Uma vez que os atributos das regras são devidamente definidos, podem-se obter suas expressões em linguagem formal e em linguagem natural. Para tanto, a classe RegraContrasteSeEntão utiliza o métodos traduzirParaOCL() e traduzirParaPortugues(), respectivamente. Assim sendo, os atributos das regras funcionam como expressão intermediária entre a linguagem formal e a linguagem natural. Outras abordagens têm comportamento semelhante, porém usando OCL como linguagem intemediária entre a linguagem natural e o SQL (ZIMBRÃO, 2002); ou então, usando linguagem natural estruturada como ponto de partida para a obter regras em linguagem formal (GOMES, 2002).

O método traduzirParaOCL() parte de um *template* para regra de negócio em OCL, substituindo os atributos da instâncias de maneira conveniente. Neste caso, o *template* para regras OCL seria o seguinte:

```
context Restrição inv:
  if
    self.nome = <nome>
  then
    FatoContraste.allInstances()->forAll( <a>, <c> : FatoContraste |
      if
        <a>.éIgualA(<c>, FatoContraste.attributes() - <atribsDif> ) and
        <restriçõesComplexas>
      then self.modo = <modo> or (
        ( f1.contraste() = <valorAntecedente> and
          f1.var.nome = <varAntecedente> )
        implies
        ( f2.contraste() = <valorConsequente> and
          f2.var.nome = <varConsequente> )
      )
      else Boolean::false
      endif
    )
  else Boolean::false
  endif
```

Dentro do *template*, há marcações entre os símbolos “<” e “>”. Cada marcação se refere a um atributo da classe. Deste modo, o método traduzirParaOCL() sabe onde substituir os valores dos atributos da classe. Cabe notar que as variáveis auxiliares usadas na chamada do método forAll() são <a> e <c>, de modo que as expressões em OCL contidas no atributo 'restriçõesComplexas' mantenham relação coerente com o restante da expressão.

De maneira análoga, pode-se também obter a expressão em linguagem natural destas regras através do método traduzirParaPortugues(). Para realizar isto, usa-se basicamente a mesma solução empregada em traduzirParaOCL(): um *template* contendo marcações a serem substituídas por valores de atributos. Um *template* para linguagem natural poderia ter sua estrutura semelhante a esta:

```
se houver
  <valorAntecedente> de <varAntecedente>
  entre <tipoReferencia.dimensão>
  (para um <atribsDif + restriçõesComplexas>)*
então é <modo> que haja
  <valorConsequente> de <varConsequente>
  entre estes mesmos <tipoReferencia>
  (para estes mesmos <atribsDif + restriçõesComplexas>)*
```

A estrutura deste *template* para linguagem natural é semelhante à do *template* para linguagem formal: intercalam-se termos do *template* com marcações correspondentes a atributos de regra.

No entanto, há diferenças que devem ser observadas. Elas se dão principalmente nos últimos trechos de cada oração antecedente e conseqüente. Estes trechos têm a seguinte forma:

```
...
(para um <atribsDif + restriçõesComplexas>)*
...
```

Uma diferença é que se convencionou usar a expressão "`( ... )*`" para representar a repetição de um trecho de *template*. Isto é necessário porque cada atributo do conjunto 'atribsDif' deve ter uma expressão da forma "... para um <categoria> ...", mas não se sabe a priori quais e quantos são os elementos do conjunto.

Além disto, o *template* contém a combinação dos atributos 'atribsDif' e 'restriçõesComplexas' dentro da mesma marcação, usando o sinal "+" para conectá-los. Esta marcação indica que a tradução para Português pode depender de uma combinação dos dois atributos. Por exemplo, a regra 'Preço-Volume-Concorrente' em linguagem natural contém os seguintes trechos:

```
se houver
  ...
  para um produto
  ...
então é provável que
  ...
  para um produto concorrente deste mesmo produto
  ...
```

Ou seja, a segunda oração tem que fazer referência ao produto citado na primeira (função do atributo 'atribsDif') e, além disto, também deve indicar que eles são produtos concorrentes (função do atributo 'restriçõesComplexas'). Como o significado das

expressões contidas em 'restriçõesComplexas' é desconhecido, pode ser muito difícil obter esta tradução de maneira automática. Portanto, a tradução para linguagem natural, quando possível, tende a ser bem mais complexa do que a tradução para OCL.

Outra diferença entre as duas modalidades de tradução é que a substituição dos valores dos atributos da regra deve resultar em termos de linguagem natural, e não em termos da OCL. Ou seja, o método tem que saber que 'prod' se traduz em “produto”, que 'geo' se traduz em “geografia” etc. Em outras palavras, é necessário que existam mecanismos para traduzir termos do modelo de dados UML em termos de linguagem natural.

Uma outra dificuldade é que a linguagem natural exige que se respeitem algumas normas de concordância da Língua Portuguesa. Portanto, a tradução não deve apenas substituir termos por outros: é preciso ainda analisar como eles se relacionam sintaticamente nas frases, e adaptar os termos resultantes neste contexto. Por exemplo, uma tradução direta da regra 'Preço-Volume' resultaria o seguinte:

```
se houver
  baixa de preço
  entre data
então é provável que haja
  alta de volume
  entre estes mesmos data
```

Ao ler a frase, nota-se que ela apresenta erros de concordância em relação ao termo “data”: ele deveria estar no plural, e seu gênero é feminino. Assim sendo, a frase deveria ser escrita da seguinte forma:

```
se houver
  baixa de preço
  entre datas
então é provável que haja
  alta de volume
  entre estas mesmas datas
```

Vê-se, portanto, que a tradução de regras de negócio em linguagem natural não é uma tarefa simples. Na verdade, o tratamento de linguagem natural em computador é ainda um problema muito complexo para as tecnologias atuais. Este tema ainda é estudado em diversas iniciativas de pesquisa (SOWA, 2000). Portanto, é de se esperar que processos de tradução automática como o desta experiência conduzam a resultados ainda modestos, ou que não possam ser automatizados completamente.

No entanto, os resultados obtidos até aqui contribuem positivamente para que se consiga traduzir regras de negócio para linguagem natural. Resumindo, estes resultados são os seguintes:

- A decomposição das regras de negócio em atributos com semântica definida.
- A determinação de *templates* para linguagem natural para guiarem a tradução das regras.
- A tradução individual de cada termo do modelo de dados em termos da linguagem natural.

A partir destes insumos, futuros trabalhos deverão encontrar menos dificuldades neste tipo de tradução. Sugestões para trabalhos futuros são discutidos em mais detalhes no capítulo 7.

6. Protótipo de Ambiente OLAP para Regra de Negócio

Este capítulo descreve um protótipo experimental que visa facilitar a interação entre analistas e regras de negócio. O protótipo consiste em um hipertexto contendo informações conceituais sobre o ScanOne (variáveis, dimensões etc), além de regras de negócio relevantes.

O hipertexto é alimentado dinamicamente pelo próprio dicionário de metadados do ScanOne. Portanto, regras de negócio serão incluídas no dicionário como uma extensão. A principal vantagem disso é que as regras de negócio podem ser construídas usando dimensões, variáveis, categorias e hierarquias como vocabulário básico.

As seções seguintes apresentam os requisitos imaginados para o protótipo, bem como alguns exemplos de navegação.

6.1. Requisitos do Protótipo

O grande requisito do protótipo é permitir que o usuário tenha acesso a regras de negócio de maneira amigável. Isto significa visualizar as regras de negócio que existem, navegar por elas e observar suas composições. Uma vez que as regras de negócio se referem a outros elementos do modelo dimensional (dimensões, variáveis etc), pode-se chegar a estas informações a partir das regras que as compõem (e vice-versa).

Inicialmente, foram previstas três formas principais de navegação: navegação pelas variáveis; navegação pelas regras de negócio; e navegação pelo contexto da visão ScanOne. A navegação pelas variáveis serve para o usuário observar as variáveis e entender o seu significado isoladamente. Em outras palavras, é a possibilidade de entender o que as variáveis significam.

A outra forma de navegação é através das regras de negócio. Cada regra de negócio tem um nome e uma expressão textual. Dependendo do tipo de regra, esta expressão textual pode ser na forma de linguagem natural ou em linguagem formal (OCL, neste caso). Regras de negócio que interessem tanto aos analistas de negócios quanto aos analistas de sistemas devem ter as duas formas de expressão.

A última forma de navegação é pelo contexto do ScanOne. Esta forma serve para que o usuário acesse as regras de negócio relacionadas com alguma visão que o ScanOne esteja apresentando em um dado momento. Desta maneira, o analista de negócio que esteja fazendo análises via o ScanOne pode pedir auxílio ao hipertexto do protótipo para saber as regras de negócio que têm relação com aquele contexto de análise.

6.2. Recursos de Navegação pelo Hipertexto

O hipertexto foi desenhado para ser acessado via Internet através de um browser. A Figura 6.1 ilustra a página inicial do hipertexto.

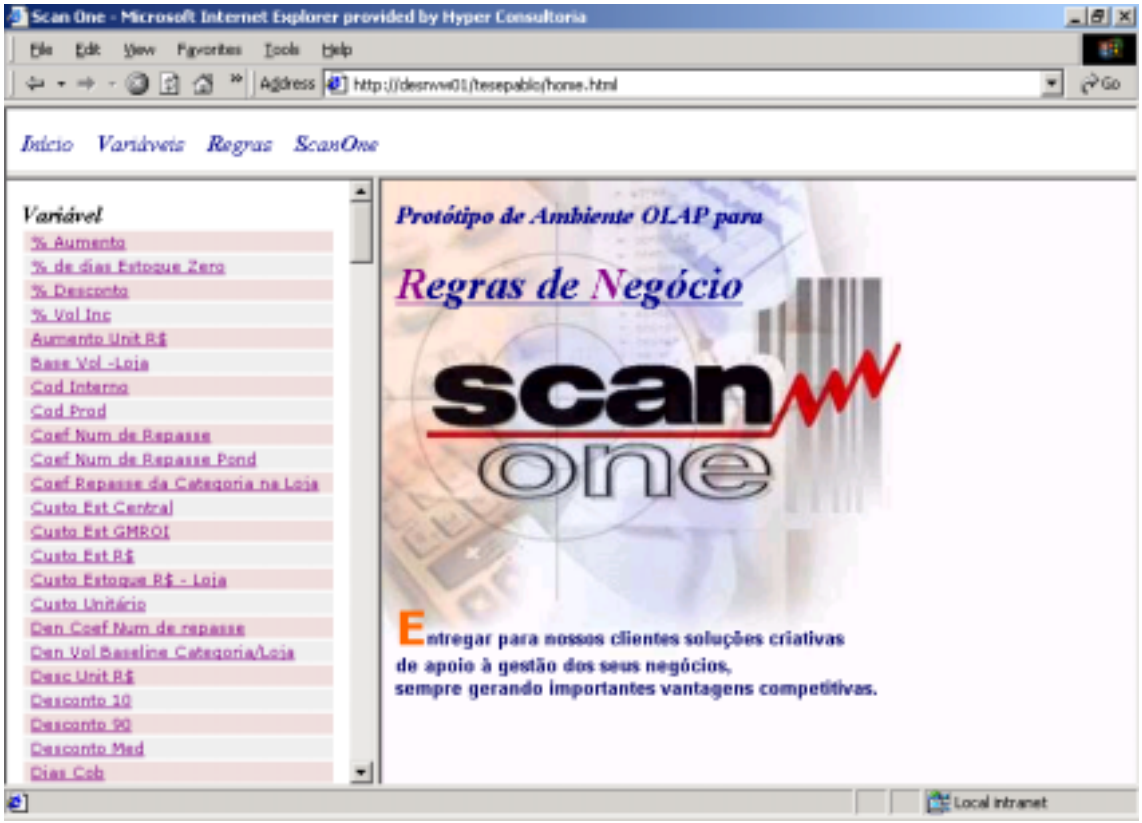

Figura 6.1. Página inicial do protótipo.

Selecionando a opção “Variável” no alto da tela, tem-se como resultado o conjunto de todas as variáveis do ScanOne sendo exibidas à esquerda. Os elementos desta também são ligações, de modo que – quando selecionados – exibem os detalhes da variável à direita (Figura 6.2). A partir da lista, vê-se que é razoavelmente extensa a quantidade de variáveis mantidas pelo ScanOne. Futuramente, esta lista pode ser transformada em uma hierarquia de variáveis organizada por temas (variáveis de volume, de preço, de faturamento etc).

Figura 6.2. Exemplo de uma variável descrita em detalhes.

Uma vez que se tenha selecionado uma variável em particular (no caso da Figura 6.2 é o “Preço”), pode-se descobrir quais são as regras de negócio que têm alguma relação com esta variável. Para tanto, basta usar a ligação “Regra” no alto da janela. Esta ligação faz com que surja a relação de regras de negócio desejada, já descritas em linguagem natural. Desta forma, o analista pode ter idéia do peso que a variável tem perante o resto do sistema: quais são as variáveis que ela influencia? Que outras variáveis têm influência sobre ela? O resultado desta consulta é ilustrado na Figura 6.3 .

É interessante notar que as descrições em linguagem natural das regras listadas exibem os termos do negócio como ligações. Estas ligações levam às definições destes termos, de modo que o usuário pode facilmente encontrar o significado de um termo que apareça na regra.

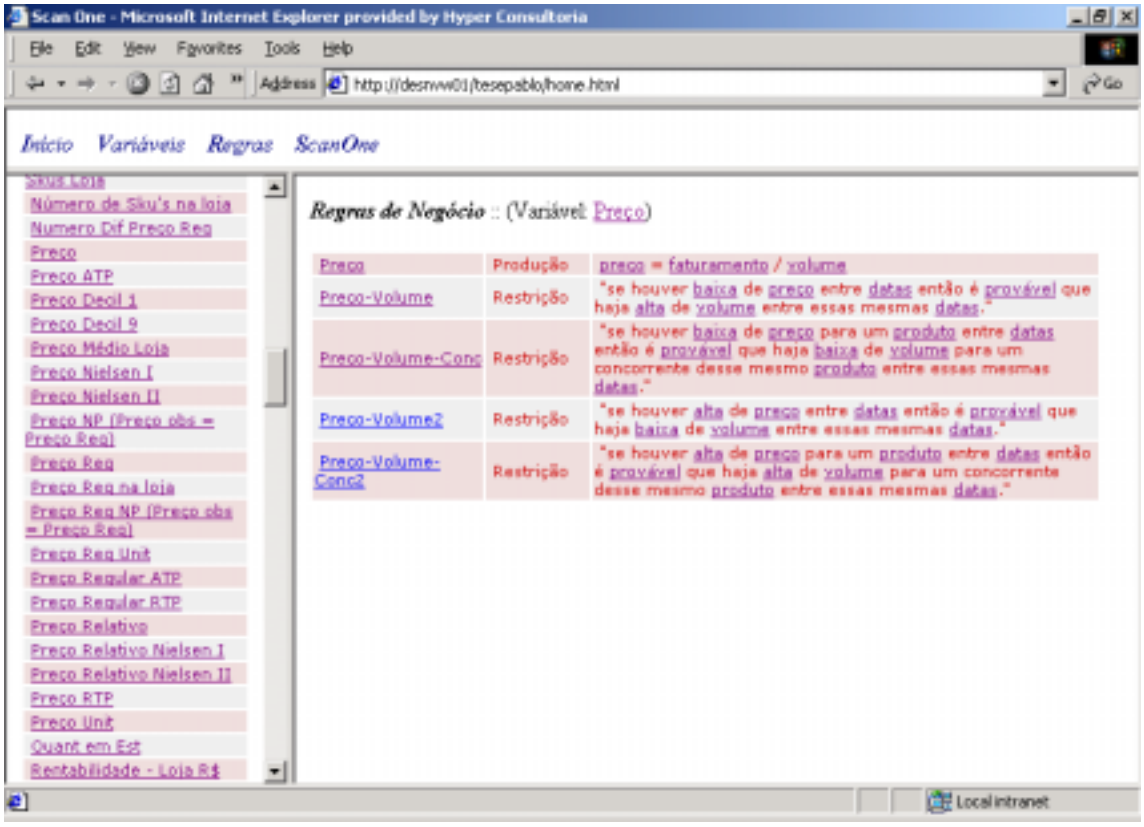

Figura 6.3. Regras de negócio relacionadas à variável “Preço”.

A partir daí o analista pode escolher uma regra de negócio e observá-la em maiores detalhes. Por exemplo, a Figura 6.4 ilustra os detalhes da regra “Preço-Volume-Conc”. É interessante notar que esta é uma regra de negócio que admite expressão em linguagem natural (“Descrição Informal”) e também em OCL (“Descrição Formal”). Esta funcionalidade é especialmente interessante para defrontar as regras de negócio que são interessantes para o negócio e para o sistema.

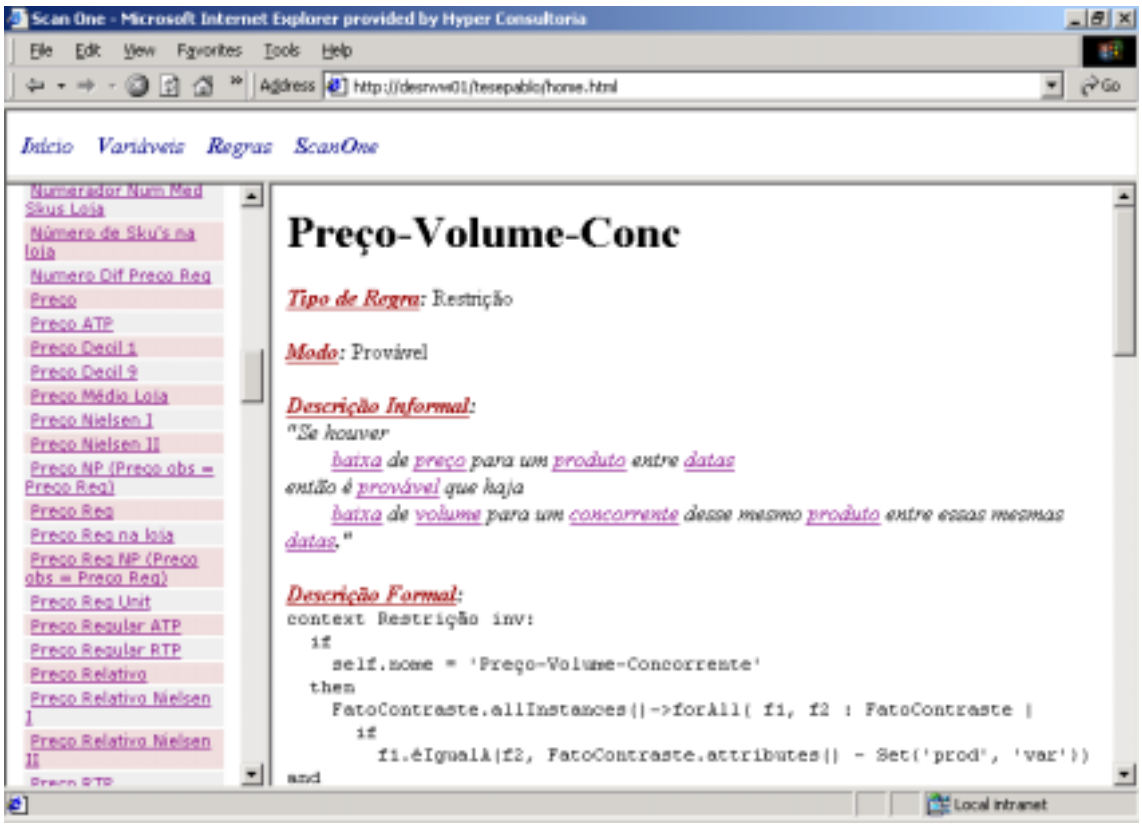

Figura 6.4. Regra de negócio descrita em detalhes.

O recurso mais interessante do protótipo talvez seja o seguinte: a capacidade de interação direta com o próprio ScanOne. A ligação “ScanOne” no alto da janela exibe uma instância do ScanOne no próprio browser (Figura 6.5). Esta instância possui todos os recursos que o ScanOne tem no ambiente desktop, permitindo que o usuário realize análises como ele faz normalmente.

Porém, se o usuário selecionar novamente a ligação “ScanOne”, o protótipo exibirá as regras de negócio que têm relação com a visão ScanOne construída neste momento. A parte esquerda da janela exibe as categorias de cada dimensão que estão selecionadas, e a parte da direita lista as regras de negócio associadas (de alguma forma) a estas categorias. Desta maneira, o analista pode usar o protótipo como fonte de sugestões para o prosseguimento da análise, ou ainda encontrar razões conhecidas para fenômenos cujas causas ele desconhece.

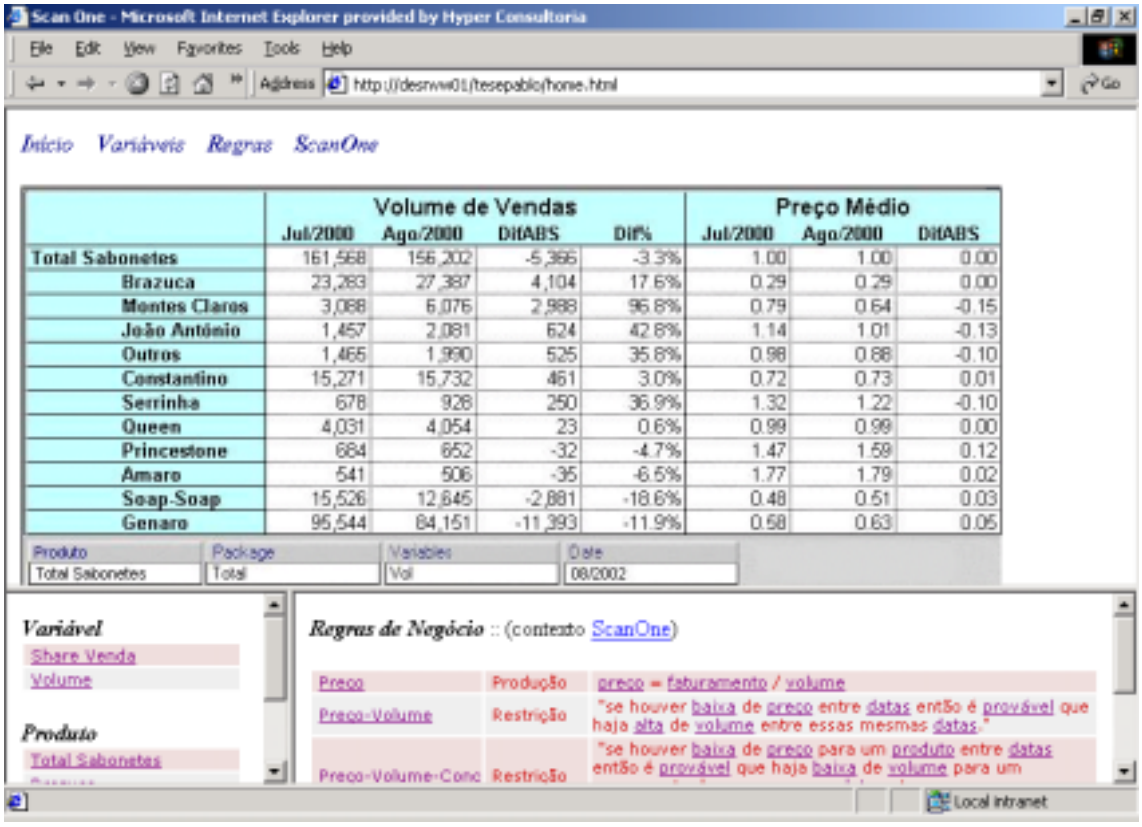

Figura 6.5. Interação entre ScanOne e ambiente do protótipo.

Experimentos deste protótipo junto a usuários reais estão sendo planejados para os próximos meses. Desta forma, será possível avaliar na prática se as informações apresentadas surtem efeito na qualidade das análises.

7. Conclusão

A competitividade sempre presente no mercado tem gerado uma demanda cada vez maior por gerência de conhecimento nas empresas. Esta demanda tem evidenciado que, al~em da manipulação de dados, é necessário manipular as regras de negócio associadas a estes dados. Esta dissertação explorou conceitos associados a regras de negócio, e ilustrou como estas regras são utilizadas para tomada de decisão em ambientes OLAP. Entretanto, os ambientes OLAP atuais ainda não incluem facilidades para a manipulação de regras de negócio. Neste sentido, este trabalho apresentou uma maneira de integrar ambientes OLAP às definições de termos e regras do negócio com o objetivo de melhorar a qualidade de processos decisórios nestes ambientes.

A estratégia de desenvolvimento desta dissertação consistiu em estudar os conceitos relativos a regras de negócio em OLTP e adaptá-los a questões típicas de ambientes OLAP. Para tanto, as tecnologias de regras de negócio e ambientes OLAP foram descritas e analisadas com o intuito de serem posteriormente integradas. Um exemplo de análise OLAP foi descrito de modo a materializar melhor as idéias apresentadas. Por fim, um protótipo de hipertexto foi construído para demonstrar os potenciais benefícios desta integração entre ambientes OLAP e regras de negócio.

7.1. Contribuições

Uma das contribuições desta disstertação foi a integração entre ambientes OLAP e regras de negócio. Na verdade, estudos neste sentido são praticamente inexistentes, de modo que a discussão sobre este tema tem potencial para se aprofundar muito mais. Em contrapartida, as iniciativas para o uso de regras de negócio em OLTP (BUSINESS RULES GROUP, 2002) (ROSS, 1998) apontam na direção de regras de negócio para o negócio propriamente dito, mostrando que há uma certa tendência de aproximação entre estes dois tópicos.

Porém, a maior contribuição desta dissertação se deu no sentido de aproximar os processos decisórios apoiados por ambientes OLAP e as regras de negócio que sustentam estes processos. Esta contribuição é interessante pois os analistas de negócios não precisam recorrer a fontes externas ao ambiente OLAP para obter insumos para suas análises. Deste modo, os processos decisórios tendem a apresentar maior uniformidade e integridade entre si – o que é positivo no contexto corporativo.

Outra contribuição interessante foi a possibilidade de apresentar regras de negócio em formato adequado, ou para analistas de sistemas, ou para analistas de negócios, ou para ambos. Esta é uma contribuição interessante porque geralmente os trabalhos desta área têm uma postura excludente em relação a estas formas de expressão, oferecendo solução para apenas um perfil de usuário (BUSINESS RULES GROUP, 2002). Porém, conforme colocado no capítulo 2, há regras de negócio que interessam a usuários de ambos os perfis, variando somente a forma de expressão da regra de negócio. Portanto, embora uma mesma regra de negócio pode ser apresentada de maneiras diferentes, tratam-se apenas de formas diferentes de representação de uma

mesma regra, e portanto, deve haver uma expressão conceitual da regra que seja mais geral que suas representações particulares.

Neste sentido, o capítulo 5 apresentou um metamodelo de dados para regras de negócio que representa mais claramente esta definição conceitual de regra de negócio. Na verdade, trata-se de um metamodelo para regras de negócio, através do qual podem-se definir regras de negócio a partir da instanciação de atributos do modelo. Isto ajuda a organizar acervos de regras de negócio, uma vez que o metamodelo propõe uma natural categorização das regras de negócio. Na prática, as regras de negócio do metamodelo podem ser expressas através de *templates* genéricos, que orientam a definição das regras de negócio finais. Nesta dissertação, foram apresentados exemplos de *templates* para regras de negócio em linguagem formal (OCL), e também em linguagem natural.

Além disto, outra característica do metamodelo que merece destaque é o de que sua estrutura visa expressar informações multidimensionais. Desta maneira, as regras de negócio podem ser expressas usando informações provenientes de ambientes OLAP – o que, em síntese, correponde ao objetivo maior da dissertação.

7.2. Melhorias e Trabalhos Futuros

Como melhoria possível, pode-se citar o uso de XML como linguagem de expressão de *templates*. Além de ser uma linguagem padronizada e computacionalmente tratável, os esquemas XML possibilitariam a construção de verdadeiras gramáticas de regras de negócio, acomodando possibilidades interessantes, como recursividade e combinações de atributos.

Como trabalho futuro, seria interessante estender o exemplo do protótipo para que ele exiba todos os conceitos presentes no dicionário de dados do ScanOne, pois no momento ele contempla apenas alguns deles.

Uma outra continuação do trabalho atual envolveria o desenvolvimento de outras formas de consulta por tipo de entidade. Além das consultas atuais por variável e regras de negócio, poderiam ser acrescentadas dimensões, hierarquias e categorias.

Um outro trabalho futuro interessante seria completar o ciclo de desenvolvimento de software que, nesta dissertação, inicia-se com regras de negócio em linguagem natural, e chega até a sua expressão das regras em OCL. Para o ciclo ficar completo, pode-se transformar as regras em OCL em linguagem de programação ou regras de bancos de dados, já que existem iniciativas de pesquisa neste sentido (GOMES, 2002) (ZIMBRÃO, 2002). Desta maneira, um ambiente completo de desenvolvimento de sistemas através de regras de negócio poderia ser futuramente desenvolvido.

8. Bibliografia

BOOCH, G.; RUMBAUGH J.; JACOBSON I. **UML Guia do Usuário**. Rio de Janeiro : Campus, 2000. 472 p.

BUSINESS RULES GROUP. **Defining Business Rules - What Are They Really**. Jul 2001. Disponível em: http://www.businessrulesgroup.org/first_paper/br01c0.htm. Acesso em: 22 Out 2002.

_____. **Organizing Business Plans: The Standard Model for Business Rule Motivation**. Nov 2000. Disponível em: http://www.businessrulesgroup.org/second_paper/BRG-BRMM.pdf. Acesso em: 11 fev. 2002.

BUSINESS RULE SOLUTION. **RuleSpeak Sentence Templates**. 2001. Disponível em: http://www.brsolutions.com/rulespeak_download.shtml. Acesso em: 15 mar. 2002.

DATE, C J. **What Not How**. Addison Wesley Longman, Inc., 2000. 132 p.

DAVENPORT, T. H.; PRUSAK, L. **Conhecimento Empresarial: Como as organizações gerenciam seu capital intelectual**. Rio de Janeiro : Campus, 1998. 237p.

ELSMARI, R. S.; NAVATHE, S. **Fundamentals of Database Systems**. Addison-Wesley Publishing, 1999. 960 p.

FERREIRA, A. B. H. **Novo Aurélio Século XXI – O Dicionário da Língua Portuguesa**. Rio de Janeiro : Nova Fronteira, 1999. 2128 p.

GOMES, M. **Uma Abordagem de Regras de Negócio Baseada em Linguagem Natural Estruturada**. Dissertação de Mestrado. UFRJ, Rio de Janeiro, 2002 a defender.

GOUGEON, A. **Everything You Ever Wanted to Know about Business Rules**, Proyecto ILACO II, Bolívia, 2001. Disponível em: http://www.dulcian.com/papers/BR%20Symposium%202001/Gougeon_BR.htm. Acesso em: 23 Mar 2002.

HACKARTORN, R. **Data Warehousing Energizes Your Enterprise**. Datamation, New York, v.41, nº 02, p.38-43, Fev 1997.

HARVARD BUSINESS REVIEW. **Tomada de Decisão**, Rio de Janeiro : Campus, 2001. 182p.

HYPER CONSULTORIA. **Scan One**. Disponível em: http://www.hyperinf.com.br. Acesso em: 17 Ago 2001.

INMON, W. H. **Building the Data Warehouse**. New York : John Wiley & Sons, Inc., 1992. 298p.

KIMBALL, R. **The Dimensional Model Manifesto**, DBMS Magazine, Ago 1997.

LOSHIN, D. **Enterprise Knowledge Management: The Data Quality Approach**, Morgan Kaufmann, 2001. 494p.

MALHOTRA, Y. **Knowledge Management for the New World of Business**. 1998. Disponível em: http://www.brint.com/km/whatis.htm. Acesso em: 2 Jul 2002.

MORIARTY, T. **Business Grammar Rules Part 1**, 2002. Disponível em: http://www.tdan.com/i020fe03.htm. Acesso em: 10 Abr 2002.

OBJECT MANAGEMENT GROUP. **Object Constraint Language Specification**. Set 1997. Disponível em: http://www.omg.org/cgi-bin/doc?formal/01-09-77. Acesso em: 03 Mar 2002.

_____. **OCL 2.0 Request for Proposal Submission Objectives**. Jul 2001. Disponível em: http://www.klasse.nl/ocl/subm-objectives-text.html. Acesso em: 04 Mar 2002.

_____. **Object Management Group**. Disponível em: http://www.omg.org. Acesso em: 11 Abr 2001.

PRESSMAN, R. **Engenharia de Software**. São Paulo : Makron Books, 1995. 1056 p.

ROSS, R. G. **Business Rules Concepts**, Business Rules Solutions Inc., 1998.

_____. **Current Thoughts On Expressing Business Rules**. Jan 2000. Disponível em: http://www.brcommunity.com/cgi-local/x.pl/commentary/a453.html. Acesso em: 15 Dez 2001.

_____. **Expressing Business Rules**. SIGMOD Record, Volume 29, Number 2, Jun 2000. Disponível em: http://www.acm.org/sigmod/sigmod00/eproceedings/papers/ronross.pdf. Acesso em: 12 Set 2001.

_____. **What Is A 'Business Rule'?**. Business Rule Community, Mar 2000. Disponível em: http://www.brcommunity.com/cgi-local/x.pl/commentary/b005.html. Acesso em: 25 Out 2001.

SOWA, J. **Knowledge Representation, and Computational, Foundations**. Brooks/Cole, 2000.

THOMSEN, E. **OLAP Solutions: Building Multidimensional Information Systems**. John Wiley & Sons, 1997.

THORPE, M. **Business Rules Modeling Language**. Internationales Congress Centrum, Berlin, Alemanha, Mai 2001. Disponível em: http://www.gca.org/papers/xmleurope2001/papers/html/s15-2.html. Acesso em: 05 Nov 2002.

VON HALLE, B. **Business Rules Applied**, John Wiley & Sons, Inc., New York, 2002. 546p.

WORLD WIDE WEB CONSORTIUM. **Resource Definition Framework (RDF) Model and Syntax Specification**, 1999. Disponível em: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/. Acesso em: 12 Abr 2001.

WIDOM, J.; Ceri, S. **Active Database Systems Triggers and Rules for Advanced Database Processing**, Morgan Kaufmann, 1996.

ZACHMAN, J.A., **A framework for information systems architecture**. IBM Systems Journal, 26(3):276-292, 1987.

ZIMBRÃO, G.; MIRANDA, R. A.; SOUZA, J. M.; NETO, F. P. **FalaOCL: Uma Ferramenta para Parafrasear OCL**. SBES - Sociedade Brasileira de Engenharia de Software, 2002.