UNIVERSIDADE FEDERAL DA PARAÍBA - UFPB
CENTRO DE CIÊNCIAS E TECNOLOGIA - CCT
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO - DSC
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA - COPIN

DISSERTAÇÃO DE MESTRADO

UMA INTERFACE PARA CONSULTAS A SISTEMAS DE SUPORTE À DECISÃO BASEADA EM LINGUAGEM NATURAL, CASAMENTO DE PADRÕES E METADADOS

André Vinicius Rodrigues Passos Nascimento

(Mestrando)

Ulrich Schiel

(Orientador)

CAMPINA GRANDE, NOVEMBRO DE 2001

NASCIMENTO, André Vinicius Rodrigues Passos Nascimento

N244I

Uma Interface para Consultas a Sistemas de Suporte à Decisão baseada em Linguagem Natural, Casamento de Padrões e Metadados.

Dissertação de Mestrado, Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande, Paraíba, Novembro de 2001.

119p. Il.

Orientador: Ulrich Schiel

1. Banco de Dados
2. Data Warehousing
3. Interfaces em Linguagem Natural

CDU - 681.3.07B

Resumo

O crescimento da Internet e da *World Wide Web* veio acampanhado pelo desenvolvimento de tecnologias que permitiram o surgimento dos sistemas de suporte à decisão baseados na *Web*. Esse novo paradigma reduz barreiras tecnológicas e torna mais fácil e menos onerosa a tarefa de difundir informações para diversos tipos de usuários dentro de uma organização. Contudo, as interfaces tradicionais, desenvolvidas especialmente para usuários profissionais, nem sempre são adequadas para usuários casuais e não-especialistas. Esta dissertação apresenta uma proposta de interface para sistemas de suporte à decisão baseada em linguagem natural restrita, comparação de padrões e metadados. Essa nova arquitetura pretende auxiliar usuários sem o devido treinamento técnico a recuperar informações em sistemas de suporte à decisão. A solução proposta mostra que uma abordagem alternativa e mais simples para o problema de interfaces em linguagem natural para banco de dados pode alcançar bons resultados. A arquitetura apoia-se no modelo multdimensional e na modelagem dimensional para oferecer um acesso transparente e natural a banco de dados e solucionar o problema de transportabilidade entre aplicações.

Abstract

The widespread use of the Word Wide Web and Internet has been accompanied by the development of technologies that enabled the emergence of web-based decision support systems. This new paradigm has reduced technological barriers and made it easier and less costly to make decision-relevant information available to all types of users in a company. However, traditional user interfaces specially designed for professional users will not work well for casual and non-specialist users. This dissertation presents a new interface to web-based decision support systems which is based on restricted natural language, pattern matching and metadata. This new architecture was designed with the aim of assisting users without technical instruction with the task of retrieving decision-relevant information. The proposed solution shows that an alternative and simpler approach to the problem of natural language interfaces to databases can achieve useful results. The architecture relies on the multidimensional model and dimensional modeling to provide a transparent and natural database access for non-specialist end-users, and address the domain portability problem.

"... As invenções são sobretudo o resultado de um trabalho teimoso..."

Alberto Santos Dumont

Dedicatória

Dedico esse trabalho às pessoas que estiveram, estão e sempre estarão presentes.

Aos meus pais, Ana Lúcia e César Augusto, pelo apoio constante e momentos de felicidade que sempre fizeram parte da minha vida.

A minha irmã, Luciana, pelo carinho, incentivo e compreensão nos momentos de ausência.

A minha noiva e eterna namorada, Silmara, que se fez presente em todas as etapas desta longa jornada. Seu carinho, apoio e compreensão jamais serão esquecidos.

Ao meu avô, José Rodrigues Pereira, que, mais uma vez, não pôde estar presente no fim de mais uma etapa de minha vida. Seu exemplo de pessoa, pai e avô estarão sempre em minha memória.

Aos meus avós, Cecília, Zeneide e Mário, meus eternos pais, que sempre me incentivaram e compreenderam minha ausência nos últimos anos.

Agradecimentos

Agradeço primeiramente a Deus pela saúde e força espiritual durante os anos dedicados ao Mestrado.

Ao meu amigo e orientador Ulrich Schiel, pelas lições de vida, ensinamentos, paciência, confiança e amizade responsáveis pela conclusão deste trabalho. Agradeço também a Ulrich pelo apoio prestado durante a minha permanência em Campina Grande.

Aos professores e amigos Marcus Costa Sampaio e Jacques Philippe Sauvé pelo apoio, incentivo e ensinamentos que foram cruciais para a realização deste trabalho e para minha formação.

Aos meus eternos orientadores, professor Cláudio Andrade Macêdo e professor Methanias C. R. Júnior, pelo imenso incentivo à minha formação.

Aos amigos Alberto, Edeyson, Eduardo Jorge, Eliane, Lady Jane, Hilmer, Rodrigo, Carlinhos e Benitz que sempre estiveram presentes e fizeram de Campina Grande meu segundo lar.

Aos eternos amigos Alexandre Michael e Dannie Rosie pela amizade e constante incentivo.

A todos os professores e funcionários da UFPB.

Aos outros colegas do Mestrado.

À CAPES pelo suporte financeiro durante todo o trabalho de Mestrado.

Sumário

Índice de Figuras

Capítulo 1

Introdução

A globalização da economia e o conseqüente surgimento de um ambiente de negócios mutável e altamente competitivo levaram as organizações a exigirem mais eficiência dos seus sistemas de suporte à decisão a fim de que seus membros pudessem tomar decisões de nível estratégico e tático de maneira rápida e segura. No início dos anos 90, como resposta a essa necessidade, tecnologias como *Data Warehouse* e OLAP (*On-Line Analytical Processing*) surgiram para melhorar o acesso e a qualidade da informação utilizada pelos sistemas de suporte à decisão [Barquin *et al*., 1997].

O enfoque *Data Warehouse* possibilita a existência de um repositório de informações integradas, provenientes de sistemas de informações operacionais e sistemas de informações legados [Wu & Buchmann, 1997]. Uma vez obtida essa integração, interfaces OLAP para *Data Warehouse* auxiliam os analistas e executivos a sintetizarem informações sobre a empresa através de comparações, visões personalizadas, análise histórica e projeção dos dados em vários cenários [Dinter *et al*., 1998].

A grande popularidade e a disseminação do uso da Internet vêm sendo acompanhadas pelo desenvolvimento de uma variedade de tecnologias que tornam possível o surgimento de uma nova classe de sistemas de suporte à decisão. Os Sistemas de Suporte à Decisão baseados na *Web* representam sistemas que fornecem informações e ferramentas de apoio à decisão para gerentes e analistas de negócios através do uso de um *browser* para *Web* como o Netscape Navigator ou Internet Explorer [Power, 1998]. Essa nova tendência, que se aproveita da infra-estrutura da *World Wide Web*, tem-se desenvolvido com grande rapidez em virtude das vantagens trazidas por esse novo paradigma. Possibilidade de atender a um maior número de usuários, redução de custos na aquisição de software e instalação são, entre outras, vantagens dessa nova abordagem em relação à tradicional arquitetura cliente/servidor.

Aplicações para suporte à decisão têm sido desenvolvidas, ao longo dos anos, principalmente para usuários de indústrias e organizações, que ocupam posições nos níveis estratégico e tático de uma empresa. A conseqüência direta dessa prática é a retenção de informação nas mãos de uma minoria privilegiada, diminuindo, dessa forma, as chances de um maior sucesso corporativo e individual. Na era da democracia da informação, o conhecimento deve, também, estar nas mãos daqueles que lidam diretamente com o cliente e que podem, mais efetivamente, promover sua satisfação [Pilot Software].

Embora o desenvolvimento de aplicações de suporte à decisão direcionadas para a Internet tenha reduzido as barreiras tecnológicas e financeiras que impediam a difusão da informação através dos diferentes níveis de pessoal e departamentos existentes em uma empresa, ainda não foram apresentadas soluções para o problema da diversidade de níveis de conhecimento técnico entre grupos de usuários. A continuidade do desenvolvimento de ferramentas de consulta que se baseiam no mesmo paradigma de utilização das ferramentas cliente/servidor, desenvolvidas especificamente para gerentes e executivos, impede que grupos de usuários, não familiarizados com tais tecnologias, recuperem informações relevantes às suas atividades.

A utilização de linguagens de consulta de alto nível como *SQL (Structured Query Language*) [Elmasri & Navathe, 1994] [Ullman & Widom, 1997] também não representa uma solução, pois embora tenham seu mérito ao abstrair a estrutura física dos dados e constituir-se de uma linguagem fácil e declarativa, exige de seus usuários um conhecimento mínimo sobre esquema de banco de dados e álgebra relacional para que as consultas sejam efetivamente realizadas.

Alguém poderia questionar se as atuais interfaces em linguagem natural para acesso a banco de dados representariam uma solução. A existência de interfaces comerciais dessa natureza (Elf da Elf Software, EnglishQuery da Microsoft e Q&A da Symantec) [About.com, 1999] subtraem a necessidade de aprendizado de uma linguagem de consulta formal, porém tais interfaces são desenvolvidas para esquemas de banco de dados em geral, o que as tornam ferramentas com expressiva necessidade de configuração antes de serem usadas. Para cada novo esquema a ser utilizado, todo um conjunto de entidades e relacionamentos linguísticos precisam ser definidos.

Surge, então, a seguinte questão: "Como disponibilizar uma interface de consulta para sistemas de apoio à tomada de decisões para usuários com pouco ou nenhum conhecimento técnico em ferramentas ou linguagens de consulta a banco de dados, e que atenda ao requisito de transportabilidade entre domínios, além de apresentar uma arquitetura que possa ser facilmente estendida ou modificada ?"

O presente trabalho procura mostrar uma solução para o problema apresentado através da definição de uma nova arquitetura de interface que tenha como meio de interação usuário-sistema, uma interface em linguagem natural restrita que atenda requisitos de *habitability*[1] e transportabilidade entre domínios, oferecendo acesso natural e transparente ao banco de dados através de uma camada conceitual multidimensional.

1.1 Objetivos da Dissertação

O objetivo fundamental deste trabalho é a apresentação de uma arquitetura de interface em linguagem natural restrita para acesso a banco de dados relacionais, que sirva como meio de obtenção de informações a níveis de pesquisa ou gerencial para usuários casuais que não tenham conhecimento de linguagens como *SQL* e que não sejam familiarizados com as atuais interfaces visuais de consulta. Através de uma abordagem com linguagem natural restrita, apoiada por metadados e diretamente relacionada com uma camada conceitual multidimensional, a arquitetura visa atender, também, a requisitos como *habitability* e transportabilidade entre domínios. Ao combinarmos comparação de padrões, linguagem natural restrita e metadados, buscamos explorar as vantagens sobre a linguagem natural pura, interfaces visuais e linguagens tipo *SQL* na interação com usuários casuais.

Um outro objetivo é apresentar uma especificação de projeto e implementação para validar a arquitetura proposta. O projeto desenvolvido possui dois principais objetivos: a) confirmar nossas expectativas a respeito da arquitetura proposta; b) direcionar futuras implementações para aspectos relevantes à arquitetura. De fato, não é objetivo desta dissertação prover uma interface para o usuário final, mas sim apresentar uma nova arquitetura, confirmar sua validade quanto à possibilidade de implementação e prover uma infra-estrutura que possa servir de ponto de partida para interfaces robustas o suficiente que

[1] O termo *Habitability* refere-se à medida da eficiência com a qual o usuário pode reconhecer e adaptar-se às limitações do sistema [Epstein, 1985] [Sethi, 1986].

possam ser testadas e utilizadas por usuários casuais. Logo, não apresentaremos um projeto que contemple todos os módulos da arquitetura proposta, mas restringiremos nossa abordagem aos módulos que acreditamos serem os mais relevantes para os objetivos a serem alcançados (ver capítulo 4).

1.2 Relevância da Dissertação

O sucesso e o conseqüente crescimento de um Sistema de Suporte à Decisão dentro de uma empresa é o reflexo direto do uso contínuo e da busca de novos requisitos por parte dos seus usuários. Para alcançar esse cenário, é necessário possuir ferramentas que auxiliem os "tomadores de decisões" na busca por informações que possam favorecer suas atividades. Como reflexo do atual desenvolvimento de sistemas de suporte à decisão voltados para a *Web* e de uma política de difusão da informação por toda a empresa, surge a necessidade de atender aos requisitos de uma nova classe de usuários não-especialistas, desconhecedores das atuais tecnologias para acesso a dados.

Em virtude do exposto acima, atribuímos a relevância do nosso trabalho à solução apresentada para agregar valor aos novos sistemas de suporte à decisão baseados na Web. O projeto desenvolvido, ao criar uma camada conceitual multidimensional como forma de abstração para a representação do banco de dados, também especifica modelos de metadados nos níveis técnico e semântico que podem ser de grande valia para futuras implementações de interfaces de acesso a *Data Warehouses*.

1.3 Estrutura da Dissertação

No capítulo 2, introduzimos a área de pesquisa "Interfaces em Linguagem Natural para Banco de Dados", apresentando seu histórico, sua evolução como conseqüência do surgimento de novas interfaces, as arquiteturas envolvidas nos projetos de tais interfaces e os principais problemas lingüísticos confrontados por sistemas que utilizam o processamento de linguagem natural. Ao final, detalhamos projetos e implementações que, restringindo o domínio de conhecimento ou limitando as interações entre consultas e respostas, tentam abordar o problema de acesso a banco de dados.

No capítulo 3, apresentamos o domínio dos Sistemas de Suporte à Decisão, suas principais características e tecnologias. Introduzimos a disciplina "Modelagem Dimensional"

e analisamos sua importância. O esquema Estrela e sua maior variação, o esquema *Snowflake* são explorados com o objetivo de fundamentar conceitos necessários aos capítulos subseqüentes.

No capítulo 4, apresentamos uma arquitetura de interface como solução para o problema de acesso a dados em Sistemas de Suporte à Decisão baseados na *Web* por usuários casuais e não-especialistas. Posteriormente, especificamos o projeto e implementação de uma interface baseada na solução descrita com a finalidade de validar a arquitetura apresentada e prover uma infra-estrutura a ser reutilizada por futuras implementações.

No capítulo 5, concluímos a especificação do projeto e implementação através da apresentação do mecanismo de extensão utilizado pela interface. Ao final, fazemos comparações entre a solução apresentada e outras interfaces que apresentam arquiteturas semelhantes.

Finalmente, no capítulo 6, concluímos o trabalho de dissertação e identificamos possíveis trabalhos futuros.

Capítulo 2

Interfaces em Linguagem Natural para Banco de Dados

"A linguagem é um dos aspectos fundamentais do comportamento humano e um componente crucial de nossas vidas" [Allen, 1995]. Esta afirmação reflete a importância da linguagem para a sociedade, em particular para a interação entre seres humanos, pois representa a forma mais importante de expressão e comunicação. O estudo da compreensão e produção da linguagem é uma área de pesquisa que tem alcançado grande sucesso e cuja relevância atual e futura é inquestionável. Como exemplo de aplicações nessa área podemos citar: tradução de linguagens naturais, recuperação de informação, processamento de textos para categorização e extração de informações, reconhecimento de voz, interfaces em linguagem natural[1] para banco de dados, sistemas especialistas, sistemas tutores, sistemas de ajuda *on-line* e geradores de resumos.

Interface em linguagem natural para banco de dados (ILNBD) é um sistema que permite ao usuário obter informações armazenadas em banco de dados através do uso de comandos ou perguntas escritos em linguagem natural, como por exemplo português. Atualmente esses comandos são traduzidos em alguma linguagem formal de acesso a banco de dados, sendo *SQL* a mais utilizada. A área de acesso a banco de dados foi o primeiro grande sucesso no desenvolvimento de aplicações que utilizam o processamento automático de linguagem natural [Russell & Norvig, 1998]. Naquela época, final dos anos 60, pesquisadores acreditavam que a tarefa de desenvolver tais interfaces era plausível e que o esforço despendido em pesquisas iria trazer grandes benefícios ao demonstrar a utilidade das interfaces como ferramenta para usuários em geral, e acumular experiência para aplicações

[1] Linguagem Natural representa a definição para linguagens que evoluíram naturalmente como resultado da comunicação entre seres humanos. São exemplos de linguagens naturais: Português, Inglês e o Espanhol.

mais complexas que fizessem uso de processamento de linguagem [Copestake & Jones, 1989].

Embora o escopo das ILNBDs seja limitado pelas restrições de linguagem e de domínio de discurso, ainda faz-se necessário confrontar questões como transportabilidade entre domínios, identificação das limitações do sistema por parte dos usuários e problemas lingüísticos que normalmente ocorrem na maioria dos sistemas que trabalham na interpretação de linguagem natural. Dentre esses problemas, encontramos várias formas de ambigüidade introduzidas por modificadores e conjunções, utilização de sentenças incompletas e mal formuladas [Androutsopoulos *et al*., 1995].

Neste capítulo, apresentamos um breve histórico da área de Interfaces em Linguagem Natural para Banco de Dados, identificamos as vantagens e desvantagens das ILNBDs em relação a outros tipos de interfaces, mostramos as principais arquiteturas utilizadas pelas interfaces e introduzimos os principais problemas confrontados durante o desenvolvimento de tais sistemas. Ao final, analisamos e tecemos comentários sobre projetos de interfaces cujas arquiteturas não seguem padrões geralmente utilizados e cujas linguagens são subconjuntos explicitamente restritos, mas que apresentam-se como solução para problemas como transportabilidade entre domínios, facilidade de extensão e desenvolvimento.

2.1 Histórico

Os primeiros protótipos de ILNBDs apareceram no final dos anos 60, mas foi no início dos anos 70, com o desenvolvimento do LUNAR (*the Lunar Science Natural Language Information System*) para a NASA (*National Aeronautics and Space Administration*), que deu-se o primeiro grande marco nessa área de pesquisa. LUNAR era um sistema experimental desenvolvido para ajudar geólogos a acessar, comparar e avaliar dados sobre análises químicas das rochas lunares e composição do solo lunar obtidos da missão Apollo-11. O principal objetivo dos projetistas era pesquisar os problemas que deveriam ser confrontados no desenvolvimento de um sistema que utilizasse uma linguagem natural como forma de interação homem-máquina. Embora o LUNAR não tenha sido utilizado em um ambiente operacional real, suas pesquisas tiveram grande influência na maioria dos sistemas subseqüentes [Russell & Norvig, 1998] [Allen, 1995].

Na década de 70, muitos outros protótipos apareceram. No RENDEZVOUS [Copestake & Jones, 1989], o usuário era encorajado a formular suas perguntas a partir de um diálogo com o sistema. LADDER [Copestake & Jones, 1989], cuja arquitetura era baseada em gramáticas semânticas, propiciava o acesso a grandes bancos de dados e podia ser configurado para acessar informações em diferentes tipos de sistemas gerenciadores de bancos de dados. Outros exemplos de interfaces dessa década podem ser encontrados em [Copestake & Jones, 1989].

O melhor exemplo de interface do início dos anos 80 é o CHAT80 [Russell & Norvig, 1998], cuja implementação serviu de base para muitos outros sistemas dessa época. Implementado inteiramente em prolog, transformava as perguntas em linguagem natural para expressões em prolog que posteriormente eram utilizadas para consultas a um banco de dados. CHAT80 tinha como um dos seus principais objetivos tratar o problema da transportabilidade de interfaces entre domínios, tema muito explorado na década de 80 pelas pesquisas e protótipos na área de interfaces em linguagem natural para banco de dados. Outro exemplo interessante da década de 80 foi o sistema de gerenciamento de informações ASK [Androutsopoulos *et al*., 1995], que tentava prover interações com outros tipos de aplicações, como programas de e-mail, além de permitir consultas a seu banco de dados proprietário.

Ainda na década de 80, surgiram sistemas comerciais em linguagem natural [Copestake & Jones, 1989] para acesso a banco de dados de propósito geral, porém, em virtude de algumas peculiaridades, tiveram uma aceitação limitada. INTELLECT da AI Corp. , apesar de ter apresentado um sucesso razoável em relação ao processamento de linguagem natural, possuía um alto custo de aquisição e necessitava de um grande esforço por parte dos administradores para ser configurado. Q&A da Symantec foi outro sistema comercial; embora tenha apresentado um grande volume de vendas, seu sucesso não foi conseqüência direta da interface em linguagem natural, mas sim de suas características e facilidades como ferramenta de acesso a dados, que não tinham relação com linguagem natural, frente à concorrência [Copestake & Jones, 1989]. As pesquisas e o desenvolvimento de protótipos continuaram durante a década de 90 e, através de avanços no campo de processamento de linguagem, evoluíram para novos sistemas comercias.

Embora muitas ILNBDs desenvolvidas a partir dos anos 80 tenham demostrado características importantes em relação ao processamento lingüístico, elas não ganharam a aceitação comercial tão esperada. Apesar da atual existência de interfaces comerciais de

propósito geral (Elf da Elf Software e EnglishQuery da Microsoft), ILNBDs ainda são tratadas como sistemas exóticos ou alvo de pesquisas. Provavelmente as principais razões para a falta de aceitação desses sistemas sejam o desenvolvimento de ferramentas gráficas de sucesso e os problemas intrínsecos das ILNBDs, apontados na seção 2.4.

ILNBDs são projetadas para trabalhar com modelos de dados específicos. O desenvolvimento do modelo relacional causou um fortíssimo impacto nas ILNBDs. A existência de uma única estrutura de armazenamento, a relação, e o surgimento de uma linguagem declarativa de alto nível para consultas, como a *SQL*, possibilitaram um grande desenvolvimento no campo das ILNBDs. Porém, novas tendências na tecnologia de bancos de dados como a extensão objeto-relacional [Ullman & Widom, 1997] [Braker & Moore, 1996] podem causar um impacto negativo nas futuras interfaces em linguagem natural. A existência de novas estruturas para capturar mais significado e possibilitar uma modelagem de dados mais avançada (*nested tables*[2], tipos abstratos de dados, apontadores, coleções) tornam mais árdua a tarefa de recuperação de dados. Além disso, a falta de padrões aceitos pelo mercado faz surgir inúmeras extensões proprietárias. Contudo, acreditamos que esses problemas são frutos das atuais implementações de SGBDs e não refletem os propósitos da extensão objeto-relacional.

2.2 Vantagens e Desvantagens das ILNBDs

A maior motivação para a pesquisa e o desenvolvimento de ILNBDs como ferramentas de consulta são as vantagens, embora ainda não alcançadas por completo na prática, em relação às linguagens como *SQL*, interfaces baseadas em formulários[3] e interfaces

[2] *Nested Table* representa um novo tipo de tabela criada como extensão objeto-relacional por alguns sistemas gerenciadores de bancos de dados. Esse novo tipo de tabela é logicamente armazenado em uma coluna de uma outra tabela. Tal característica oferece novas possibilidades para modelagem de objetos complexos que utilizam agregação.

[3] Interfaces baseadas em formulários são sistemas que apresentam formulários pré-definidos contendo campos a serem preenchidos para a realização de consultas. Após a especificação de um dos campos, a interface encarrega-se de preencher o resto do formulário através de consultas em um banco de dados [Elmasri & Navathe, 1994].

visuais[4]. A principal vantagem é que usuários não necessitam aprender uma linguagem de comunicação artificial ou conhecer modelos lógicos dos sistemas gerenciadores de banco de dados a fim de elaborar suas consultas. A ILNBD ideal seria aquela que pudesse oferecer o uso de linguagem natural sem restrições, porém, no atual estado da arte, quando utilizamos a expressão "linguagem natural", estamos fazendo referência a dialetos que restringem a linguagem natural livre [Savadovsky, 1988]. Logo, ainda existe a necessidade de conhecer as funcionalidades e limitações de uma interface dessa natureza. Outras características que tornam uma ILNBD a interface perfeita para usuários em geral são: a facilidade de formulação de perguntas que denotam negação ou quantificação e a utilização de expressões resumidas ou incompletas cujo significado é extraído do contexto do discurso [Reis *et al*.,1997].

ILNBDs, todavia, apresentam certas desvantagens em relação aos outros tipos de interfaces mencionados anteriormente. Dentre elas, as mais apontadas são: o usuário não possui plena compreensão das limitações lingüísticas e semânticas impostas a suas consultas; quando uma pergunta é rejeitada, não é exposto com clareza se a mesma está fora do âmbito lingüístico do sistema ou fora do modelo conceitual [Androutsopoulos *et al*., 1995]; as interfaces que apresentam-se como de propósito geral requerem longas etapas de configuração antes de serem utilizadas para uma aplicação particular [Copestake & Jones, 1989]; alto custo de desenvolvimento em virtude da escassez de ferramentas profissionais robustas e integradas aos ambientes computacionais, e desenvolvedores qualificados; alto custo de evolução e extensão desses sistemas ao longo do seu ciclo de vida.

2.3 Arquiteturas

Existem, basicamente, quatro modelos de arquiteturas para ILNBDs, que possuem estreita relação com as arquiteturas utilizadas por outros tipos de aplicação que fazem uso do processamento de linguagem natural. Essencialmente, as diferenças entre as interfaces para banco de dados e outras aplicações encontram-se nos módulos de tradução e acesso a banco de dados. As arquiteturas apresentadas nessa seção não correspondem exatamente, em tipos de módulos e funções, às utilizadas pelos protótipos e sistemas comercias, mas representam a

[4] Sistemas ou Interfaces Visuais de Consultas (SVCs) representam sistemas de consulta baseados em representações visuais. A idéia principal desses sistemas é prover ao usuário uma nova representação visual para modelos de banco de dados. TVQE é um exemplo dessa categoria de interface [Silva *et al*., 1997].

base para o surgimento de inúmeras variações. A seguinte apresentação tem por objetivo introduzir as principais arquiteturas abstratas utilizadas por ILNBDs e os princípios em que se baseiam os métodos de processamento utilizados. Logo, não discutiremos os diferentes aspectos envolvidos no processamento de linguagem natural, como tipos de analisadores sintáticos, categorias de gramáticas ou linguagens de representação intermediária. Discussões detalhadas sobre esses tópicos são encontradas em [Allen, 1995], [Savadovsky, 1988] e [Russel & Norvig, 1998].

2.3.1 Sistemas baseados em Comparação de Padrões

A arquitetura baseada em comparação de padrões, cuja forma é representada na figura 2.1, foi utilizada por alguns dos primeiros sistemas que faziam uso de uma interface em linguagem natural. Destinada a aplicações que possuem um conjunto pequeno de intenções, não utiliza analisadores sintáticos e gramáticas durante a interpretação de uma sentença. O processamento de uma sentença é feito através do uso de um conjunto de padrões. O sistema, segundo algum critério, compara a entrada de um usuário com os padrões disponíveis, e se algum padrão for identificado, a sentença é dita passível de interpretação. O programa ELIZA, desenvolvido em meados da década de 60 no MIT (*Massachusetts Institute of Technology*) e ainda utilizado em pesquisas de Inteligência Artificial, faz uso dessa arquitetura para simular um diálogo no qual o sistema desempenha o papel de um terapeuta durante uma seção com um paciente, representado pelo usuário [Allen, 1995].

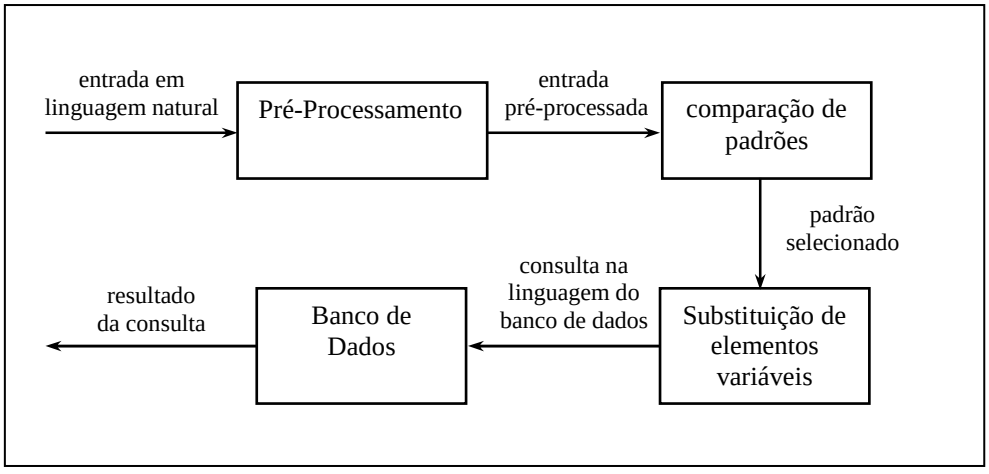


Figura 2.1: Arquitetura abstrata dos sistemas baseados em comparação de padrões

O exemplo a seguir ilustra essa abordagem no desenvolvimento de interfaces para bancos de dados. Considere as seguintes relações como parte de um esquema de dados para uma livraria:

```
Livro  (Cod_Livro, Titulo, ISBN, Cod_Editora, Preço)
Editora(Cod_Editora, Nome, Endereço, Telefone)
```

Agora, considere um sistema simples baseado na arquitetura da figura 2.1 e que utilize os seguintes padrões para interpretar as perguntas de um usuário:

Padrão 1.
(a) padrão: ... "preço | quanto custa" .... "livro"... <Título do Livro>
(b) ação: Selecionar o preço do livro cujo Titulo = <Titulo do Livro>

Padrão 2.
(a) padrão: ..."editora | nome da editora" ..."publicou"... <Título do Livro>
(b) ação: Selecionar a editora que publicou o livro cujo Titulo = <Titulo do Livro>

Em (a) encontramos as definições dos formatos dos padrões utilizados por nossa interface a ser considerada e em (b) as ações a serem executadas caso algum padrão seja identificado. O primeiro padrão especifica que se um usuário fizer uma pergunta que contenha a palavra "preço" ou a expressão "quanto custa" seguida da palavra "livro" mais o título de um livro que exista na relação Livro, o sistema deve localizar o preço do livro cujo nome foi mencionado na pergunta. O segundo padrão segue o mesmo raciocínio. Logo, o nosso sistema será capaz de responder perguntas como: "Qual o preço do livro O caminho da tranqüilidade ?", "Qual editora publicou o livro Diário de um Mago ?".

O exemplo anterior utiliza a arquitetura na sua forma mais simples, dando ao tratamento de linguagem natural um caráter superficial. Porém, muitos sistemas atuais utilizam variações do uso dessa técnica, tornando a abordagem menos superficial e alcançado bons resultados. gNarLI [Shankar & Yung, 2000] utiliza expressões regulares para definir os formatos dos padrões e a definição de regras para auxiliar no processamento. NLBean [Watson, 1999] também compara perguntas de usuários com combinações de padrões que são passíveis de processamento. Nesses sistemas, variações da técnica de comparação de padrões

são empregadas de modo a oferecer flexibilidade sintática à interface e facilitar seu desenvolvimento.

2.3.2 Sistemas baseados em Sintaxe

Em sistemas baseados em sintaxe, a pergunta em linguagem natural é analisada sintaticamente com o auxílio de uma gramática e um léxico. Posteriormente, a estrutura resultante é mapeada diretamente para uma linguagem de consulta a banco de dados. A figura 2.2 exemplifica os principais módulos dessa arquitetura e o fluxo de informação durante o processamento.

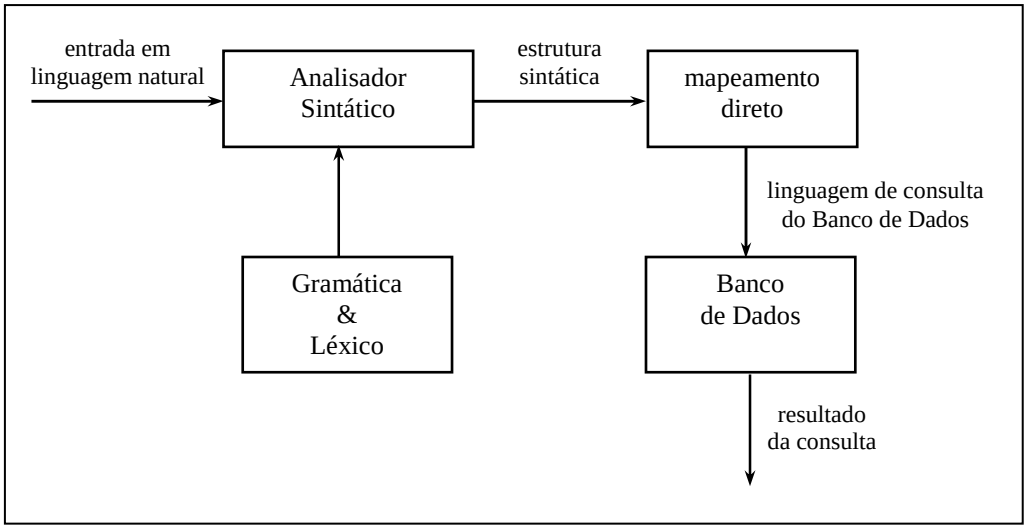


Figura 2.2: Arquitetura abstrata dos sistemas baseados em sintaxe

A gramática utilizada nessa arquitetura descreve as possíveis estruturas sintáticas que podem existir em uma pergunta do usuário, e o léxico encarrega-se de classificar as palavras em categorias gramaticais e atribuir-lhes algum significado a mais. Na figura 2.3 temos um exemplo simples de uma gramática e suas regras de produção. A estrutura resultante da análise sintática, geralmente uma árvore sintática, tende a dois propósitos: verificar se a pergunta do usuário possui uma estrutura válida; fornecer informações ao módulo de mapeamento para que produza uma expressão correspondente na linguagem de acesso a banco de dados utilizada.

S	⇦	*SN SV*
SN	⇦	*Det N*
Det	⇦	*"que" | "qual"*
N	⇦	*"rocha" | "espécime" | "magnésio" | "radiação" | "luminosidade"*
SV	⇦	*V N*
V	⇦	*"contém" | "emite"*

Figura 2.3: Exemplo de gramática simples para um sistema baseado em sintaxe

As interfaces desenvolvidas com base na arquitetura da figura 2.2 e utilizando gramáticas semelhantes à apresentada na figura 2.3 destinam-se a aplicações específicas e bancos de dados que possuem uma linguagem de consulta especialmente projetada para facilitar o mapeamento entre a estrutura resultante da análise sintática e uma sentença válida para a linguagem do banco de dados [Androutsopoulos *et al.*, 1995]. Logo, questões como transportabilidade entre domínios e utilização de diferentes tipos de bancos de dados não podem ser contempladas através do uso dessa arquitetura.

2.3.3 Sistemas baseados em Gramática Semântica

A arquitetura abstrata dos sistemas baseados em gramática semântica (figura 2.4) [Allen, 1995] é uma extensão da arquitetura dos sistemas baseados em sintaxe (figura 2.2).

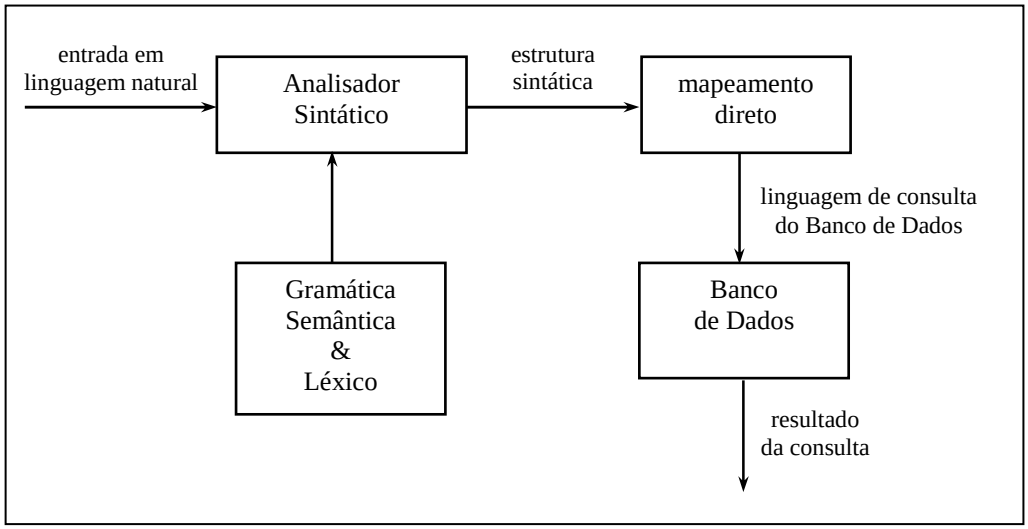


Figura 2.4: Arquitetura abstrata dos sistemas baseados em gramática semântica

Basicamente, a diferença está na estrutura da gramática utilizada. Essa nova gramática, conhecida como gramática semântica, possui em sua estrutura elementos que não

correspondem necessariamente a componentes sintáticos como verbos, nomes, sintagmas[5]. Um exemplo simples desse tipo de gramática pode ser observado na figura 2.5.

A vantagem da utilização desse tipo de gramática é que sua estrutura é concebida a partir de restrições semânticas. Dessa forma, o sistema pode assegurar que quando uma consulta for declarada válida pelo analisador sintático, uma sentença equivalente na linguagem de consulta do banco de dados poderá ser gerada. Considerando a gramática da figura 2.5 pode-se observar que a seguinte sentença: "Que rocha contém luminosidade ?", embora sintaticamente correta em relação à gramática da língua portuguesa, não é aceita como válida, pois a gramática semântica impõe a restrição que o verbo *conter* deve ser seguido por nome que denomine uma substância e não uma radiação.

S	⇦	*Espécime_pergunta*
Espécime_pergunta	⇦	*Espécime Emite_Info* | *Espécime Contém_Info*
Espécime	⇦	*"que rocha"* | *"que espécime"*
Emite_Info	⇦	*"emite" Radiação*
Radiação	⇦	*"radiação"* | *"luminosidade"*
Contém_Info	⇦	*"contém" Substância*
Substância	⇦	*"magnésio"* | *"calcium"*

Figura 2.5: Exemplo simples de gramática semântica

Essa abordagem foi adotada pelo sistema LADDER e seus sucessores, como o Q&A da Symantec [Copestake & Jones, 1989]. Um outro tipo de interface que também faz uso de gramáticas semânticas são os sistemas em linguagem natural baseados em menu. Nessas interfaces, o usuário seleciona sentenças a partir de menus a fim de formar uma consulta completa. Quando o usuário seleciona uma nova palavra ou sentença em um determinado menu, o sistema analisa a frase ou sentença utilizando a gramática semântica para determinar quais palavras ou sentenças podem seguir a sentença parcial a fim de formar uma consulta que possa ser entendida pelo sistema. Sistemas baseados nessa arquitetura alcançam bons resultados quando o domínio da aplicação é relativamente limitado. Todavia, reutilizar a interface para outra aplicação exige a definição de uma nova gramática semântica.

[5] Sintagmas são subdivisões intuitivas de orações de uma linguagem natural em que se percebe um significado claro [Savadovsky, 1988].

2.3.4 Sistemas baseados em Representação Intermediária

Atualmente, as pesquisas na área de Interfaces em Linguagem Natural para Banco de Dados estão dirigidas para uma arquitetura que adota o uso de uma linguagem de representação de conhecimento intermediária, geralmente alguma forma de lógica, que não é dependente de um banco de dados particular e não está diretamente relacionada ao acesso a banco de dados. A expressão em linguagem intermediária, gerada no módulo de análise semântica, expressa formalmente o significado que o sistema atribui à pergunta em linguagem natural [Androutsopoulos *et al*., 1995] .

Na figura 2.6, encontra-se a arquitetura abstrata dessa abordagem. O processamento da linguagem ocorre da seguinte forma: Primeiramente, a consulta do usuário é analisada sintaticamente com o auxílio de um léxico e regras sintáticas fornecidas por uma gramática. O resultado dessa análise é uma estrutura sintática que servirá de entrada para o analisador semântico que, através de regras semânticas, transformará a estrutura sintática em uma expressão da linguagem de representação intermediária. Os detalhes da representação produzida variam consideravelmente entre diferentes sistemas. Antes de enviar a expressão em linguagem intermediária para o tradutor, o módulo de análise semântica verifica, com o auxílio do modelo de domínio, a validade da expressão, uma vez que a representação intermediária utilizada é independente de contexto. Finalmente, o tradutor executa o mapeamento da representação intermediária para uma linguagem de consulta a banco de dados. Geralmente, as regras de mapeamento ficam localizadas em um módulo a parte, garantido, dessa forma, uma fácil transportabilidade entre bancos de dados.

É interessante observar que a grande vantagem dessa abordagem é a modularidade empregada. Nessa arquitetura, podemos destacar dois módulos principais. O primeiro corresponde à parte lingüística e o segundo à interface com banco de dados. Essa característica favorece a transportabilidade da interface em diferentes níveis. Edite [Reis *et al*., 1997], um sistema que responde perguntas sobre informações turísticas, é uma exemplo atual de ILNBD que utiliza essa arquitetura. O leitor interessado em uma visão mais detalhada dessa abordagem deve consultar [Allen, 1995] [Savadovsky, 1988] [Copestake & Jones, 1989].

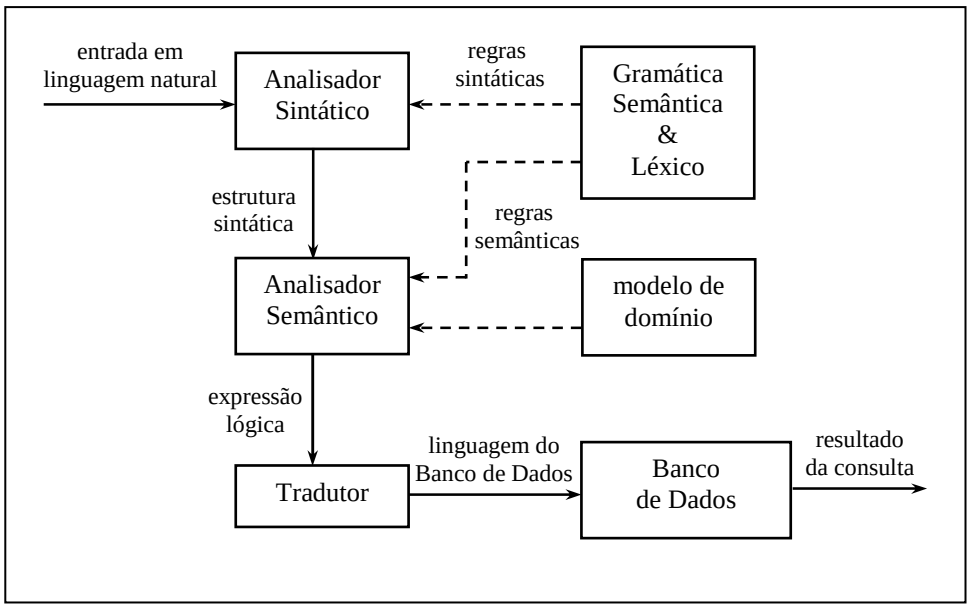


Figura 2.6: Arquitetura abstrata para sistemas baseados em representação intermediária

2.4 Desafios no Desenvolvimento de ILNBDs

Independentemente da arquitetura utilizada por um sistema que faça uso de uma interface em linguagem natural, inúmeros problemas e desafios são encontrados durante o seu desenvolvimento. Uma análise rigorosa desses problemas e a adoção de soluções são tarefas cruciais para tornar uma interface em linguagem natural um sistema realmente útil. Nessa seção, concentraremos nossa discussão nos aspectos mais importantes para ILNBDs.

2.4.1 Problemas lingüísticos

ILNBDs sofrem dos mesmos problemas lingüísticos comuns a outros tipos de interfaces em linguagem natural, todavia, como resultado da utilização de subconjuntos de uma linguagem natural e do uso de domínios de discurso geralmente limitados, esses problemas apresentam-se de forma mais amena. Nosso objetivo não é abordar todos os problemas nem todas as soluções possíveis, mas apresentar aqueles que acreditamos serem os mais relevantes ao desenvolvimento de uma ILNBD. O leitor que tenha interesse em uma lista mais completa e uma discussão mais profunda deve consultar [Allen, 1995] [Copestake & Jones, 1989].

O maior dos problemas lingüísticos que uma ILNBD pode enfrentar é a ambigüidade. Proveniente de inúmeras fontes, tende a se tornar mais grave à medida que a interface

aumenta o seu grau de expressividade através do uso de gramáticas e léxicos mais extensos. Ambigüidade léxica é decorrente da utilização de uma palavra que possui diversos significados, o que ocasiona dúvidas de interpretação de uma sentença como um todo [Russell & Norvig, 1998]. Geralmente, a análise do contexto é suficiente para que as interfaces possam determinar a melhor interpretação. Outro tipo de ambigüidade é a sintática ou estrutural, que ocorre quando a sintaxe de uma sentença possui diferentes interpretações semânticas [Savadovsky, 1988]. A presença de modificadores ou restrições em uma frase pode ocasionar esse tipo de ambigüidade: "Liste os alunos e professores do curso de Ciências Exatas com idade superior a 23 anos". Na pergunta anterior, por exemplo, a restrição "com idade superior a 23 anos" pode gerar dúvidas em relação à entidade a que se refere. Uma outra fonte de ambigüidade estrutural é o uso da palavra "e" que pode, a depender da sentença, denotar conjunção ou disjunção. Alguns sistemas empregam o uso de conhecimento sobre o domínio como solução para o problema de ambigüidade [Copestake & Jones, 1989], outros, a exemplo do PRE [Epstein, 1985], utilizam alguma forma de heurística.

Embora existam diversas soluções para os vários tipos de ambigüidade, podem surgir questões e contextos nos quais ela não pode ser resolvida. Nesses casos, é importante consultar o usuário para que ele possa esclarecer ou indicar o melhor significado. Tal procedimento impede que o sistema escolha, de maneira aleatória, um significado e retorne resultados que não possam ser identificados pelo usuário como resultantes de uma pergunta mal interpretada.

Quando as interfaces oferecem a possibilidade de diálogos como forma de interação, surgem alguns problemas que devem ser observados. Pronomes são geralmente usados em diálogos para referenciar entidades mencionadas anteriormente, fenômeno lingüístico conhecido por anáfora pronominal. Considere o seguinte diálogo entre um usuário e o gNarLI [Shankar & Yung, 2000], quando configurado para responder perguntas sobre cinema:

```
> tell me about "tom hanks"
(fale-me sobre "tom hanks")
29 Tom Hanks 1956-07-09  Concord, California, USA

> when was he born ?
(Em que data ele nasceu ?)
1956-07-09
```

No exemplo anterior, gNarLI entende que o pronome *he* (ele), utilizado na segunda pergunta do diálogo, refere-se ao ator Tom Hanks. Uma solução simples para tratar o problema da anáfora pronominal é guardar uma lista de todas as entidades mencionadas no discurso. Quando um pronome for encontrado, o sistema examina a lista, geralmente começando pelas entradas mais recentes, e associa a entidade referente ao pronome. A escolha é feita considerando a entidade mencionada mais recentemente e que satisfaça às restrições impostas pela forma da anáfora (número, gênero, pessoa) e as restrições de seleção impostas pela sentença que contém a anáfora [Allen, 1995].

Outro fenômeno lingüístico muito comum em diálogos é a elipse, omissão de palavras que se subentendem. A elipse, entre frases, ocorre quando um usuário utiliza sentenças incompletas, ficando subentendido o sentido real pelo contexto do diálogo criado pelas sentenças anteriores. O exemplo que segue, extraído de um diálogo que mantivemos com o NLBean [Watson, 1999], ilustra a ocorrência e tratamento desse fenômeno.

```
> list empname where salary IS greater than 12000
(Liste o empname onde o salário é maior que 12000)

Query results:
The value of EmpName is mark.
The value of EmpName is carol.

> and hiredate ?
(e a data de contratação ?)

Query results:
The value of EmpName is mark and HireDate is 1994/1/1.
The value of EmpName is carol and HireDate is 1994/2/10.
```

O tratamento desse fenômeno em diálogos exige que o sistema preserve a análise sintática das últimas sentenças mencionadas a fim de deduzir o significado da sentença incompleta [Allen, 1995]. O suporte à elipse é uma característica bastante desejada por usuários de interfaces em linguagem natural, pois permite o desenvolvimento de um diálogo sem a necessidade de repetir frases completas mencionadas anteriormente [Androutsopoulos *et al*., 1995].

2.4.2 Limitações e Restrições Ocultas

O objetivo do uso de uma interface em linguagem natural é subtrair dos usuários a necessidade de aprender uma linguagem e vocabulário especializados ou conhecer a estrutura interna de um banco de dados. Porém, sistemas atuais trabalham com subconjuntos de uma linguagem natural, criando, dessa forma, limitações e restrições que devem ser compreendidas pelos usuários a fim de que a interação com sistema alcance o sucesso esperado. O termo *Habitability,* utilizado na literatura com característica para categorizar interfaces, refere-se à medida da eficiência com a qual o usuário pode reconhecer e adaptar-se às limitações do sistema [Sethi, 1986].

Existem duas abordagens para minimizar o impacto de restrições desconhecidas pelo usuário. A primeira é customizar a linguagem utilizada para uma aplicação específica, ou seja, tornar a interface dependente de um determinado domínio. Embora tal procedimento possa diminuir o número de perguntas que não podem ser processadas pelo sistema, torna-se imprescindível a existência de especialistas para extrair conhecimento de um domínio e configurar a interface. Além disso, tal solução vai de encontro à tentativa de reutilizar uma interface em diferentes aplicações. A outra abordagem é definir explicitamente um subconjunto restrito de uma linguagem natural a fim de expor naturalmente aos usuários as limitações e restrições do sistema. A interface PRE [Epstein, 1985] e sistemas baseados em menu adotaram essa postura e alcançaram sucesso em relação à *habitability*.

2.4.3 Transportabilidade

Um dos temas mais pesquisados, a partir da década de 80, na área de ILNBDs foi a questão da transportabilidade das interfaces, que pode ser considerada sob três diferentes perspectivas: domínio, banco de dados e linguagem natural. A transportabilidade de domínio, ainda alvo de pesquisas, refere-se à capacidade e facilidade de adaptação para diferentes aplicações. As primeiras interfaces, a exemplo do LUNAR, foram desenvolvidas para aplicações e estruturas de dados específicas. Modificar tais sistemas significava rescrever todo o módulo de processamento de linguagem natural, modificar o mapeamento para novas estruturas de dados e criar um novo léxico. As pesquisas sobre transportabilidade intensificaram-se a partir do surgimento do CHAT80, cujo projeto objetivava alcançar um alto grau de transportabilidade. Porém, segundo [Androutsopoulos *et al*., 1995], tal característica dependia de esforço árduo para efetivar-se na prática. O uso de linguagem de

representação de significado intermediária nas arquiteturas dos sistemas mais recentes provê uma modularidade que favorece a transportabilidade em diferentes níveis.

Porém, a realidade atual é que nenhum sistema pode ser construído de modo a poder adaptar-se a um novo domínio sem a intervenção humana. Logo, a questão mais relevante é quanto treinamento e experiência são necessários e quão extenso é o processo para realizar tal tarefa. O English Query da Microsof, componente do SQL Server, é um exemplo de sistema comercial de propósito geral que busca atender à transportabilidade entre esquemas quaisquer em um banco de dados. A fim de disponibilizar um esquema para consultas em Inglês, um administrador define um modelo conceitual composto de entidades e relacionamentos, e explicita o mapeamento desses elementos para objetos do banco de dados. Uma vez que o modelo tenha sido definido, o sistema converte as perguntas em Inglês, que possuem relação com as entidades e relacionamentos especificados, em *SQL*.

A transportabilidade a nível de bancos de dados deixou de ser alvo de pesquisa a partir da consolidação do esquema relacional e do uso generalizado da linguagem *SQL*, utilizada pela maioria dos sistemas atuais como produto do processamento da linguagem natural.

Desenvolver uma interface que possa oferecer transportabilidade entre linguagens naturais é, provavelmente, o maior desafio a ser enfrentado, pois os atuais sistemas, salvo aqueles que não utilizam o processamento de linguagem natural padrão, são baseados em gramáticas e léxicos extensos diretamente relacionados com uma linguagem natural específica.

2.5 Trabalhos Relacionados à Restrição de Linguagem Natural

Nessa seção, analisaremos propostas, projetos e implementações, quando disponíveis, de interfaces em linguagem natural que, através de restrições de domínio ou limitando o conjunto de perguntas que são passíveis de processamento, tentam abordar o problema de acesso a banco de dados. Os projetos discutidos não constituem apenas uma abordagem simplificada para essa área de pesquisa, mas representam alternativas que, do ponto de vista prático, podem solucionar questões intrínsecas ao processamento de linguagem natural.

A análise que segue tem como objetivo introduzir soluções alternativas que possuam um enfoque mais prático que teórico e em cujo princípio baseia-se nosso trabalho. Muitas das soluções aqui apresentadas serão referenciadas quando tratarmos da nossa própria Interface.

2.5.1 O Sistema PRE

O sistema PRE (*Purposefully Restricted English*) [Epstein, 1985] representa uma linguagem de acesso a banco de dados que restringe explicitamente o conjunto de expressões em inglês que um usuário pode utilizar como forma de interação com o sistema. Embora, como o próprio autor explicita, "sua abordagem não produza uma análise lingüistica teoricamente motivadora como resultado do processamento de suas perguntas" [Epstein, 1985], questões relevantes como ambigüidade, *habitability* e transportabilidade entre aplicações são contempladas em sua implementação.

Primeiramente, mostraremos exemplos de perguntas que podem ser utilizadas, "bem formuladas", e exemplos de perguntas consideradas "mal formuladas" para interagir com o sistema. Posteriormente, analisaremos a arquitetura de informação e discutiremos pontos fracos e fortes dessa abordagem.

Perguntas feitas ao PRE devem obedecer a um determinado padrão sintático. Um exemplo de padrão utilizado pelo sistema é:

```
Qual
     frases nominais coordenadas
     cláusulas relativas[6] aninhadas
          cláusulas relativas coordenadas
```

Embora o PRE apresente outros padrões sintáticos, o exemplo dado anteriormente é o que apresenta maior flexibilidade de interpretações. Para melhor compreendermos como as perguntas estão relacionadas com um determinado padrão, tomemos como exemplo o esquema da figura 2.7.

[6] Do inglês *relative clauses*. Parte de uma sentença que possui um verbo e está ligada ao resto da sentença por um pronome relativo como *who, where, which, etc* [Longman, 1990]. Na gramática da língua portuguesa, as cláusulas relativas são conhecidas como orações subordinadas adjetivas restritivas [Nicola & Infante, 1991].

A pergunta a seguir é um exemplo de pergunta "bem formulada" em relação ao esquema da figura 2.7 e tendo como meio de interpretação o padrão apresentado anteriormente:

```
Quais são                                  (1)
os nomes e codigos dos alunos              (2)
     que estão em  grupos                  (3)
     que pertencem a disciplinas           (4)
     que são ensinadas por professores     (5)
          cujo nome é 'Pedro' e            (6)
          cuja idade é maior que 25 ?      (7)
```

O exemplo anterior representa uma pergunta "bem formulada", pois as linhas de (2) a (7) seguem rigorosamente as exigências do padrão. Em (2) temos nomes de atributos de entidades coordenados. Nas linhas (3) a (5) observamos cláusulas relativas aninhadas referindo-se às diversas associações do esquema da Figura 2.7. Finalmente, (6) e (7) correspondem às cláusulas relativas coordenadas que restringem a última entidade mencionada. A seguir, apresentamos uma variação da pergunta anterior que não é suportada pelo sistema e portanto considerada "mal formulada".

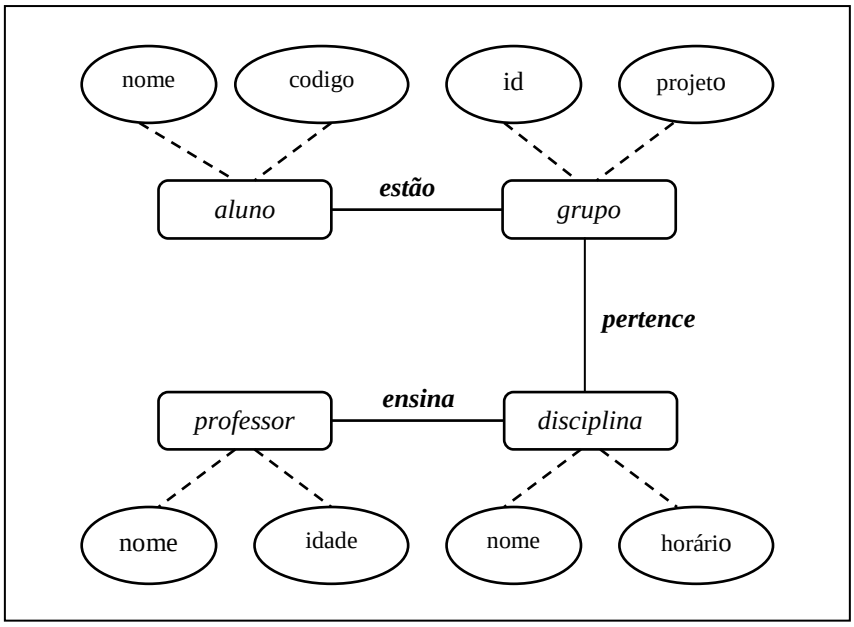


Figura 2.7: Um esquema de banco de dados ao estilo PRE.

```
Quais são                                  (1)
os nomes e codigos dos alunos              (2)
     que estão em  grupos                  (3)
     que pertencem a disciplinas           (4)
          cujo horário é 15:00             (5)
     que são ensinadas por professores     (6)
          cujo nome é 'Pedro' e            (7)
          cuja idade é maior que 25 ?      (8)
```

O exemplo acima apresenta na linha (5) uma violação à estrutura de sentença que a interface é capaz de reconhecer. Em (5) temos uma cláusula relativa que refere-se à entidade disciplina, seguida de uma cláusula relativa aninhada, transgredindo, dessa forma, a exigência do padrão de que as cláusulas relativas coordenadas sejam as últimas sentenças da pergunta. Em [Epstein, 1985], o autor alega que os padrões utilizados pelo PRE fornecem uma visão clara das exigências e limitações da interface e, portanto, minimizam erros dessa natureza.

A arquitetura do sistema PRE compreende quatro módulos: *pré-processamento de string*, *construção do grafo de navegação*, *validação e tradução do grafo de navegação*, e *recuperação de informação no banco de dados*. O *pré-processamento de string* é responsável por substituir as expressões originais por um vocabulário que possa ser reconhecido pelos módulos subseqüentes. Para isso, o sistema faz uso de tabelas de tradução que contêm informações relativas ao domínio da aplicação e de tabelas de tradução para vocabulário independente de aplicação. Os "grafos de navegação" construídos no segundo módulo especificam as dependências entre as diversas entidades, condições que devem ser impostas a registros, e atributos a serem selecionados para projeção. O módulo de *validação e tradução* assegura que as informações obtidas no processamento anterior estejam corretas e as converte para uma estrutura acessível ao módulo de *recuperação*.

O sistema aborda a questão de ambigüidade limitando a abrangência semântica das perguntas que podem ser processadas pela interface. Ao estabelecer, por exemplo, que as cláusulas relativas modificam a última entidade mencionada, uma expressão contendo restrições é traduzida em uma única especificação de navegação para o banco de dados. A implementação apresenta flexibilidade sintática através do suporte de sinônimos, especificação de navegação padrão e variação sintática através do uso de tabelas de tradução para vocabulário independente de aplicação. A transportabilidade entre aplicações, embora apontada pelo autor com uma das características, não foi demonstrada. Apesar do

processador da interface não estar associado a uma aplicação específica, é necessário que o projeto lógico da aplicação seja tal que possa adaptar-se ao modelo de dados utilizado pela interface a fim de que os padrões possam ser interpretados.

Apesar da interface, ao apresentar um conjunto de padrões sintáticos que facilite a adaptação do usuário e identifique claramente os limites da linguagem utilizada, ter alcançado sucesso com relação à *Habitability*, acreditamos que a mesma não atingiu o seu maior propósito: ser útil a usuários casuais. Para ser capaz de formular uma pergunta no ambiente PRE, o usuário é obrigado a tomar conhecimento das diversas associações existentes no esquema do banco de dados a fim de especificar com exatidão a navegabilidade da sua consulta.

2.5.2 A interface gNarLI

A interface em linguagem natural gNarLI[7], descrita no artigo titulado "A Practical Approach to Natural Language Interfaces to Databases" [Shankar & Yung, 2000], representa um sistema que utiliza no processamento de linguagem natural uma técnica de comparação de padrões baseado em regras para responder a perguntas sobre um determinado domínio. Ao utilizar um arquivo independente como fonte de informações sobre metadados e regras sobre uma determinada aplicação, o gNarLI tenta alcançar transportabilidade entre domínios com um tempo mínimo de configuração.

O sistema apresenta em sua arquitetura dois componentes principais: um arquivo de sintaxe e a máquina de processamento. O arquivo de sintaxe, que retém todo o conhecimento sobre o domínio e cuja disponibilidade é indispensável ao funcionamento da interface, é composto pelas seções *Metadados* e *Regras*. A seção *Metadados* fornece à máquina de processamento informações como localização e parâmetros de acesso para o banco de dados, definição de pronomes, consultas em linguagem *SQL* para futuras substituições e dados a respeito das associações entre as diversas tabelas que compõe o esquema da aplicação. As regras que constituem a seção *Regras* são identificadas por expressões regulares e possuem uma combinação de campos cujos valores indicam ações a serem executadas ou elementos que devem compor a expressão *SQL* gerada como resultado do processamento.

[7] o nome gNarLI é uma gíria em inglês que significa "really cool" [email de Ajeet Shankar em resposta à nossa curiosidade]

A máquina de processamento é composta pelos módulos: análise pronominal, pré-processador, aplicação de regras, análise de associações, e construção da consulta. A análise pronominal tem por objetivo identificar e resolver ocorrências de anáfora durante a utilização da interface por um usuário. Tal processo utiliza a seção de metadados do arquivo de sintaxe para verificar a existência de relacionamentos entre pronomes e nomes mencionados na última interação. Como forma de valer-se da flexibilidade sintática oferecida pela técnica de comparação de padrões e ao mesmo tempo evitar um crescimento exagerado no número de regras, a máquina de processamento faz uso de um pré-processador para converter diversas expressões em um único formato. Sua utilidade pode ser melhor verificada quando exemplificamos os inúmeros formatos de datas e horas que podem ser utilizados: 12/01/1987, 12 de Janeiro de 1997, 12 de Janeiro 97, 19:20, 7:20pm, etc. No módulo de aplicação de regras, um algoritmo que utiliza comparação de padrões tenta encontrar a melhor correspondência entre a pergunta feita ao sistema e as expressões regulares que definem as regras pertencentes a um domínio, elegendo, assim, as regras e a ordem em que as mesmas devem ser aplicadas. Ao final desse módulo, todos os componentes principais de uma consulta *SQL* já foram determinados. A análise de associações é responsável por identificar as restrições que faltam para uma junção correta das tabelas envolvidas. Finalmente, no módulo de construção da consulta, ocorre a concatenação de todas as informações necessárias para a produção da consulta final.

A solução apresentada, ao utilizar a comparação de padrões para verificar correspondências entre as perguntas passíveis de processamento e as expressões regulares que definem as regras, elimina a necessidade do usuário de ter conhecimento sobre o esquema de banco de dados da aplicação. A utilização do pré-processador mostrou-se eficaz ao oferecer uma maior liberdade de expressão para a formulação de perguntas. Embora o projeto não aborde a questão da *Habitability*, a produção de regras a partir da análise de um conjunto exemplo de perguntas reais diminui, consideravelmente, o número de questões que ultrapassam os limites da semântica que um determinado domínio pode depreender. Um dos problemas que encontramos nessa abordagem é que a transportabilidade entre domínios diz respeito apenas à máquina de processamento, pois nenhuma informação sobre metadados ou regras pode ser reutilizada por aplicações distintas.

2.6 Conclusões

Neste capítulo, introduzimos a área de pesquisa Interface em Linguagem Natural para Banco de Dados com o objetivo de apresentar a problemática por trás da tentativa de desenvolver uma ILNBD. Salientamos que, independentemente da arquitetura utilizada, existem inúmeros problemas relativos ao processamento de linguagem natural em interfaces para banco de dados. Observamos também que alguns desses problemas poderiam ser agravados a depender da arquitetura escolhida. Porém, em nossa discussão não apontamos a arquitetura ideal, pois acreditamos que esse julgamento depende diretamente da utilização, objetivos e domínio de aplicação de uma interface.

Ao final do capítulo apresentamos projetos de desenvolvimento de ILNBDs que influenciaram nossa interface. A adoção de uma variação da arquitetura de comparação de padrões, utilizada nos sistemas PRE e gNarLI, é diretamente responsável por projetos de fácil extensão e desenvolvimento. Além disso, as restrições semânticas explicitas exigidas por esse tipo de arquitetura contribuem para o esclarecimento das restrições e limitações de uma ILNBD, maior obstáculo à usabilidade dessa categoria de ferramenta.

Capítulo 3

Sistemas de Suporte à Decisão e "Data Warehouse"

Informação constitui o elemento indispensável para o sucesso e crescimento de uma organização. Durante anos, as aplicações do nível operacional e outras fontes externas serviram como repositório de dados que era utilizado por analistas e executivos na busca de material relevante que pudesse prover um melhor entendimento do negócio e auxiliar na tomada de decisões. O tempo para a concretização dessas atividades era relativamente alto, pois dependia da disponibilidade de pessoal técnico para a elaboração de programas específicos cuja principal função era unificar dados provenientes de fontes diversas. Porém, o cenário atual do mundo dos negócios exige que as empresas reajam mais rapidamente às mudanças do mercado a fim de acompanhar a evolução global e garantir vantagem competitiva frente à concorrência. Logo, a aquisição de informações e sua conseqüente circulação representam parte de um processo fundamental para a sobrevivência de uma empresa.

As atividades de aquisição de dados provenientes de diversas fontes e sua disseminação, como exposto anteriormente, constituem o cerne dos Sistemas de Suporte à Decisão[1] (SSD) que possuem como objetivo maior prover informações relevantes e integradas necessárias na execução de atividades por parte de analistas e executivos. Os primeiros arquétipos de SSDs surgiram e evoluíram com os primeiros sistemas de informações utilizados no processamento transacional. Durante sua evolução até os dias atuais, notou-se o aparecimento de inúmeras denominações na tentativa de classificar as diferentes categorias de SSDs que diferem em arquitetura, organização da informação, propósito específico e usuários potenciais [Inmon, 1997] [Power, 1999].

[1] Nesta dissertação, o termo Sistema de Suporte à Decisão possui uma conotação mais abrangente, referindo-se a todo e qualquer sistema de informação ou grupo de sistemas que tenham por objetivo prover informações que auxiliem pessoas envolvidas no processo de tomada de decisões.

A falta de consenso na categorização dos sistemas de suporte à decisão, segundo [Power, 2000b], dificulta a avaliação, integração e seleção de meios apropriados para suportar pessoas ligadas à tomada de decisões; e a proliferação de termos relativos aos diferentes tipos de SSDs geram problemas na condução de pesquisas e na discussão sobre SSDs dentro de uma organização. Para oferecer um melhor entendimento nessa área de pesquisa, [Power, 2000b] propõe uma classificação baseada no componente de tecnologia dominante, no propósito específico do sistema, nas classes de usuários que utilizarão o sistema e na tecnologia utilizada para implantação. O trabalho de Power [Power, 2000b] não é o primeiro a tentar identificar macro categorias para os diversos tipos de SSD, mas representa uma compilação de classificações anteriores juntamente com novas categorias que surgem em virtude do desenvolvimento tecnológico.

O primeiro tipo genérico de SSD, sob a dimensão do componente de tecnologia dominante, descrito por [Power, 2000b] e foco dessa dissertação é o SSD baseado em dados cuja ênfase é o acesso a dados estruturados e em particular séries temporais de dados internos e externos a uma organização. *Data Warehouse*, sistemas OLAP (*On-Line Analytical Processing*) e sistemas de informações executivas são exemplos de SSD que estão incluídos nessa categoria. Um outro tipo de SSD descrito em [Power, 2000b], agora sob a dimensão da tecnologia de implantação, é o SSD baseado na *Web* que é definido como um sistema que distribui informações e ferramentas de apoio à decisão através da infra-estrutura computacional da Internet. Vale ressaltar que os sistemas podem apresentar características híbridas, mesmo quando analisados sob uma mesma dimensão. Um *Data Warehouse*, por exemplo, pode servir a aplicações cliente/servidor em uma rede local e a aplicações distribuídas através da *World Wide Web*.

SSDs baseados em dados, em especial *Data Warehouse*, e SSD baseados na *Web* serão explorados com o objetivo de introduzir conceitos necessários ao entendimento dos próximos capítulos. O objetivo dessa dissertação é apresentar um projeto de arquitetura de interface de acesso a dados que utilize linguagem natural restrita como forma de interação com o usuário. A arquitetura utilizada pressupõe que os dados possam ser acessados através de um esquema relacional geralmente utilizado por *Data Warehouses*, o esquema em Estrela. Logo, faz-se necessário o conhecimento dessa categoria de SSD, assim como o entendimento detalhado da organização de suas informações.

Neste capítulo, apresentaremos um breve histórico da evolução dos Sistemas de Suporte à Decisão sob uma perspectiva tecnológica para o aparecimento do ambiente projetado para *Data Warehouse* e SSD baseados na *Web*. Exploraremos o ambiente de *Data Warehouse* e tecnologias, como OLAP, que utilizam dados derivados desse ambiente. A disciplina "Modelagem Dimensional" é introduzida como técnica de modelagem de dados para repositórios em banco de dados relacionais. Nesse contexto, apresentamos o esquema em Estrela e suas variações.

3.1 Histórico

O surgimento dos sistemas de suporte à decisão foi conseqüência direta do desenvolvimento das tecnologias de processamento e armazenamento de informações. Posteriormente, a evolução das tecnologias de transmissão de dados também dariam sua contribuição. No início da década de 60, os sistemas computacionais existentes em uma organização restringiam-se a aplicações específicas, geralmente escritas em COBOL, que manipulavam dados relativos a transações em arquivos mestres armazenados em fita magnética. Como conseqüência do uso desse meio de armazenamento, que tinha na busca seqüencial sua única forma de acesso aos dados, produzir relatórios repletos de informações operacionais detalhadas de interesse à gerência poderia representar uma tarefa árdua cujo tempo de execução era indeterminado.

O primeiro grande impulso na área de processamento de informações gerenciais foi o surgimento do armazenamento em disco por volta de 1970, acompanhado pelo aparecimento dos primeiros sistemas gerenciadores de banco de dados (SGBD) que mostravam-se como a solução para os problemas de manipulação inerentes ao armazenamento em fita magnética. Caracterizados como "uma única fonte de dados para todo o processamento" [Inmon, 1997], os SGBDs facilitavam o armazenamento de dados e o desenvolvimento de aplicações. Em meados da década de 70, o processamento em lote começou a dar lugar ao processamento *on-line*[2] que logo se propagaria por todas as organizações.

No final dos anos 70 e início dos anos 80, com a exploração comercial dos computadores pessoais e das linguagens de quarta geração, o papel do usuário final no

[2] O processamento *on-line*, ao contrário do processamento em lote, reflete imediatamente no banco de dados os efeitos do processamento sobre os dados, além de garantir uma resposta imediata ao usuário.

ambiente computacional começou a ganhar importância. Em meados da década de 80, a indústria de computadores voltava-se para o usuário final e inúmeros aplicativos para computadores pessoais podiam ser encontrados no mercado. Diante desse contexto, analistas e executivos começaram a exigir a disponibilidade de dados provenientes do ambiente operacional para suas estações de trabalho, originando, dessa forma, novas atribuições para os dados que serviam apenas ao processamento de transações *on-line*.

Para atender às exigências de analistas e executivos, programas de extração começaram a ser desenvolvidos e utilizados pelos departamentos dentro de uma empresa. Esse programas eram responsáveis por varrer arquivos e banco de dados em busca de dados que atendessem a critérios definidos *a priori* [Inmon, 1997]. Posteriormente, os dados selecionados eram transportados para outros bancos de dados ou arquivos que seriam diretamente manipulados por analistas de negócio. Se observarmos a nossa definição abrangente sobre sistemas de suporte à decisão, podemos considerar os programas de extração como seus precursores. A popularidade desses programas foi conseqüência de duas características principais: a possibilidade de retirar dados do caminho do processamento *on-line* sem que haja alta degradação de performance no processamento; mudança no controle dos dados, permitindo que o novo possuinte manipule-os de acordo com suas necessidades [Inmon, 1997].

Porém, o crescimento do ambiente operacional das empresas e a conseqüente explosão no número de programas de extração deu origem ao que pode ser chamado de "ambiente de desenvolvimento espontâneo", ou seja, um ambiente de apoio à decisão formado por subconjuntos de dados herdados do ambiente operacional que cresceu sem controle (Figura.3.1). Esse tipo de ambiente traz sérios problemas à gerência, tais como: falta de credibilidade dos dados, problemas de produtividade e impossibilidade de transformar dados em informações.

No ambiente da figura 3.1, quando bem evoluído, elaborar um relatório corporativo requisitado pela gerência representava uma tarefa árdua cujo tempo de conclusão não poderia ser antecipado e dependia diretamente da disponibilidade de pessoal técnico para localizar e analisar os dados do relatório, e desenvolver programas específicos para sua compilação.

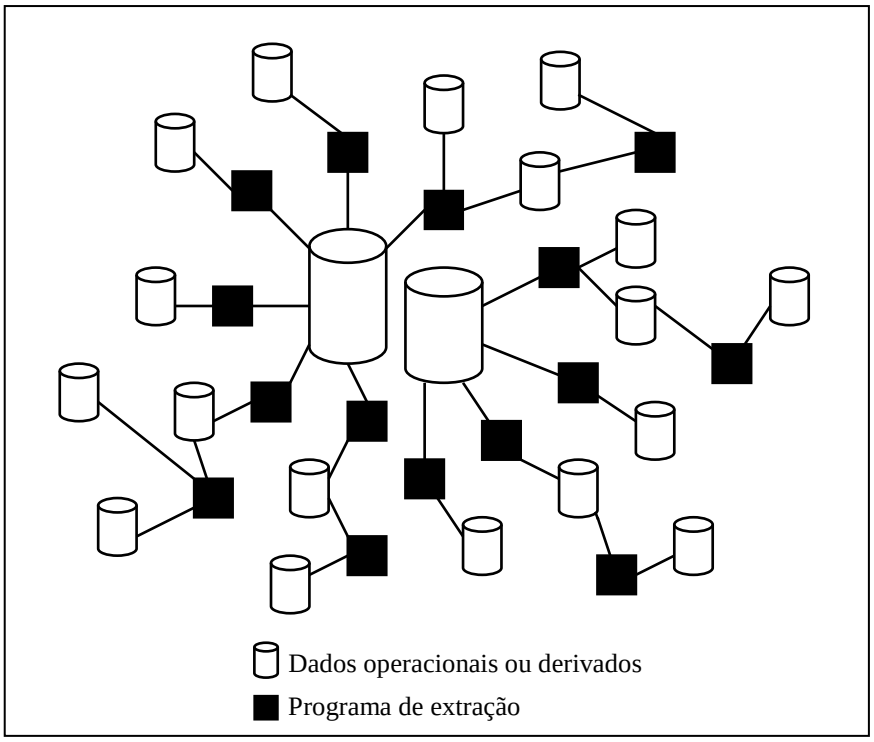


Figura 3.1: Ambiente de desenvolvimento espontâneo [Inmon, 1997]

Um outro problema é a falta de dados históricos armazenados, sem os quais é impossível para um analista observar tendências e anomalias no negócio a fim de tomar decisões precisas e de grande valia para a empresa. Desde o início dos anos 90, mudanças na economia global e o acirramento da competitividade entre as empresas forçaram uma mudança no ambiente computacional com o objetivo de melhorar a performance e a qualidade das informações obtidas a partir dos sistemas de suporte à decisão. Surge, então, o ambiente projetado de *Data Warehouse* que busca oferecer uma maior integração dos dados operacionais além de apresentar o histórico de dados necessários ao processamento analítico[3]. A próxima seção analisa em detalhes o ambiente de *Data Warehouse*.

No início da década de 80, as redes de computadores começaram a ganhar popularidade a partir do momento em que as redes de computadores pessoais passaram a oferecer uma grande vantagem de custo/desempenho em relação aos *mainframes*[4] [Tanenbaum, 1997]. Essa popularidade deu origem, em meados dos anos 80, ao que

[3] Processamento analítico representa a produção de análises de suporte a decisões gerenciais. A análise de tendências e o levantamento de perfis são exemplos de produções do processamento analítico.

[4] O termo *mainframe* é utilizado coloquialmente para denominar grandes computadores, em oposição aos microcomputadores.

chamamos hoje de Internet. A explosão do uso da Internet e de sua mais importante aplicação, a *World Wide Web*, nos anos 90, veio acompanhada pelo desenvolvimento de tecnologias que transformaram todo o processo de projeto, desenvolvimento e implementação de aplicações. Ao lado das tradicionais arquiteturas cliente/servidor, surge o desenvolvimento em n-camadas para a *Web* que oferece mecanismos para suportar uma maior comunidade de usuários a partir da solução de problemas como recursos de hardware, implantação e manutenção de aplicativos.

Ao acompanhar esse desenvolvimento, empresas e pesquisadores começaram a estudar o impacto da distribuição de informações e ferramentas para suporte à decisão utilizando a infra-estrutura da Internet [Bhargava & Power, 2001]. O resultado desse estudo foi o aparecimento dos SSDs baseados na *Web*. [Power, 1998] define essa nova categoria de SSD como sistemas que fornecem informações e ferramentas de apoio à decisão para gerentes e analistas de negócio através do uso de um aplicativo para navegação na Internet.

O uso desse paradigma, além de prover acesso a informações de apoio à decisão entre localidades geograficamente distantes, pode reduzir alguns problemas associados com o emprego da arquitetura cliente/servidor em aplicações para acesso a SSDs baseados em dados. Em sua maioria, as aplicações que utilizam a arquitetura cliente/servidor constituem-se de programas proprietários que necessitam de um suporte técnico para instalação nas estações clientes, e uma licença para cada máquina utilizada. A tecnologia da *Web* já é suficientemente madura para distribuir aplicações de acesso a dados tão poderosas quanto as aplicações cliente/servidor com a vantagem de não apresentar problemas em relação à instalação e licença de uso.

De modo geral, SSDs baseados na *Web* reduzem barreiras tecnológicas e tornam mais fácil e menos onerosa a disseminação de informações para gerentes e analistas, além de poder servir a outras classes de usuários dentro e fora de uma organização [Power, 2000a].

3.2 "Data Warehouse" e "Data Warehousing"

Por muito tempo, as organizações implementaram seus sistemas de suporte à decisão em um ambiente projetado para suportar as atividades operacionais da empresa. Nesse ambiente, a elaboração de relatórios corporativos, como apresentado na seção anterior, poderia representar uma tarefa difícil a depender do número e conteúdo das fontes de dados

que deveriam ser acessadas e integradas a fim de produzir e revelar informações necessárias a gerentes e analistas. Porém, mudanças na economia e o conseqüente aumento da competitividade nos negócios começaram a exigir um ambiente de suporte à decisão que pudesse oferecer uma visão mais inclusiva de toda a empresa. Nesse novo contexto, decisões de nível estratégico e tático exigem um conteúdo mais rico do que aquele encontrado no ambiente operacional, o qual apresenta inúmeros obstáculos para o processamento analítico. A seguir, apontamos as principais dificuldades apresentadas pelo ambiente operacional referentes à atividade de tomada de decisões [Kimball, 1996] [Gupta, 1997] [Inmon, 1997] [Chaudhuri & Dayal, 1997]:

- Trabalho difícil e tedioso para coletar dados que se encontram em diferentes locais e formatos, o que impede a determinação do espaço de tempo necessário para a aquisição de relatórios por parte da gerência;

- Modelagem dos dados não apropriada para análise dos negócios da empresa. As bases operacionais são compostas por aplicações desenvolvidas a nível departamental, não oferecendo uma visão no contexto da empresa;

- Baixo desempenho de acesso aos dados. As bases operacionais são gerenciadas por sistemas que privilegiam o processamento de transações *on-line* (OLTP). Os sistemas OLTP são eficientes e seguros para atualizações e inserções no tocante a pequenas transações. No entanto, não apresentam bom desempenho para a tarefa de análise, que, geralmente, requer consultas intensas;

- Ausência de contexto histórico dos dados disponíveis. Dados históricos são importantes para identificar tendências. Nas bases operacionais apenas o valor mais recente do dado é mantido a fim de garantir melhor performance de processamento;

- Ferramentas inadequadas para a tarefa de análise, restritas a consultas que foram antecipadas e que exigem contínua assistência de pessoal técnico para elaboração e manutenção.

Como solução para as questões apresentadas, surgiu um ambiente paralelo ao ambiente operacional, embora dependente, chamado de "ambiente projetado de *Data Warehouse*" [Inmon, 1997] [Kimball, 1996]. O componente de maior importância nesse

ambiente é o *Data Warehouse* (DW), um repositório de informações integradas provenientes dos sistemas operacionais e sistemas legados que provê dados para processamento analítico e tomada de decisões [*Wu & Buchmann, 1997*]. A idéia por trás dessa abordagem é extrair, filtrar e integrar os dados internos e externos de uma organização em uma estrutura única, permitindo uma melhor utilização dos dados por analistas, gerentes e executivos. O DW fornece uma única imagem da realidade dos negócios para toda a empresa, além de melhorar a qualidade dos dados e o tempo de acesso a informações.

O termo "*Data Warehousing"*, muitas vezes usado como sinônimo de *Data Warehouse*, compreende o conjunto de etapas necessárias para povoar o DW, o DW em si e as ferramentas utilizadas para consultar, analisar e apresentar informações armazenadas em um DW. *Data Warehousing* não é a única abordagem para integração de dados provenientes de fontes heterogêneas e distribuídas. O uso de mediadores [Wiederhold, 1991], conhecido como "abordagem por demanda" , também provê integração de informações. A característica principal dessa abordagem é a inexistência de um repositório físico integrado de informações. Para responder às consultas que necessitam de dados provenientes de diferentes fontes, o componente principal dessa arquitetura, o *Mediador*, decompõe a consulta original e redireciona as consultas resultantes para as fontes operacionais apropriadas. Após o retorno das respostas, o resultado da consulta original é produzido e enviado ao consulente. Contudo, o atual processo de tomada de decisões exige eficiência no tempo de respostas, característica que pode não ser alcançada pelo uso de mediadores quando as fontes de informações apresentam-se em grande quantidade e dispersas [Hammer *et al*., 1995].

Existem muitos desafios na construção e manutenção de um grande DW. A primeira atividade no desenvolvimento de um DW é o projeto de um esquema de dados que possa armazenar dados integrados de diversas fontes e aplicações. Antes do povoamento do DW, os dados extraídos do ambiente operacional e de fontes externas devem ser limpos, para assegurar que os dados extraídos contenham informações válidas, e transformados para reconciliar diferenças semânticas e entrar em conformidade com o novo esquema de dados. Essa integração, que lida com a heterogeneidade de fontes de informações, é responsável por um grande número de insucessos na construção de DWs [Srivastava &, Chen, 1999]. O povoamento ou carga do DW a partir dos dados limpos e transformados é acompanhado de atividades como divisão dos dados em partições, criação de índices e geração de informações sumariadas, que possuem papel crucial na performance da execução de consultas [Kimball,

1998]. Após a carga inicial do DW, é necessário que o mesmo seja continuamente monitorado para que permaneça consistente, ao longo do tempo, com as fontes de informações primárias.

A descrição das atividades relativas ao processo de *Data Warehousing*, apesar de apresentada de forma sintética, revela a complexidade envolvida na construção de uma ambiente projetado de DW. Essa complexidade é diretamente refletida nos custos necessários para o desenvolvimento, que devem ser exaustivamente justificados com base no retorno de investimentos [Power, 1997]. O valor de um DW é resultado direto do uso que gerentes e analistas fazem de suas informações. Logo, as ferramentas de acesso ao DW são elementos de extrema importância para seu sucesso e evolução. A seguir, apresentamos o conceito de interfaces OLAP, como representante das ferramentas mais predominantes em *Data Warehousing.*

3.3 Interfaces OLAP (*On-Line Analytical Processing*)

O principal objetivo de um DW é prover informações integradas que atendam aos requisitos da atividade de tomada de decisões. OLAP (*On-Line Analytical Processing*) representa um conjunto de tecnologias projetadas para suportar análise e consultas *ad-hoc* em um ambiente de *Data Warehouse*. Sistemas OLAP ou interfaces OLAP ajudam analistas e executivos a sintetizarem informações sobre a empresa, através de comparações, visões personalizadas, análise histórica e projeção de dados em vários cenários [Dinter *et al*., 1998]. Em sua maioria, esses sistemas fazem uso do modelo conceitual multidimensional [*Wu & Buchmann, 1997*] [Jarke *et al*., 1999] como meio de apresentação e organização de informações, provendo uma visão dos dados corporativos mais intuitiva e natural para as atividades de navegação e análise.

Ao contrário das técnicas tradicionalmente utilizadas no projeto de dados para aplicações do ambiente OLTP, cujo foco está voltado para entidades, relacionamentos, decomposição funcional e análises de transição de estado, o modelo multidimensional apoia-se em torno de fatos, geralmente medidas numéricas do negócio que serão objetos de análise, e um conjunto de dimensões que provêem um contexto para os fatos citados anteriormente. Grande parte do sucesso desse modelo é resultado da sua simplicidade. Executar análises ou navegar através dos dados corporativos em um projeto de dados OLTP totalmente normalizado, sendo provavelmente composto por uma grande quantidade de relações, não

representa uma tarefa trivial para o "homem de negócios", agente principal desse processo. Em contrapartida, o projeto conceitual multidimensional, além da sua forma mais simples, expressa os dados de um maneira quase natural para analistas do negócio.

A figura 3.2 é um exemplo de representação de dados no modelo multidimensional. Considere que os fatos ou medidas sejam representados pelas vendas, e as possíveis dimensões que caracterizam os fatos sejam Região, Produto e Tempo. A utilização de apenas três dimensões nesse exemplo permite a visualização de um array multidimensional, aqui em forma de cubo, como esquema do modelo multidimensional.

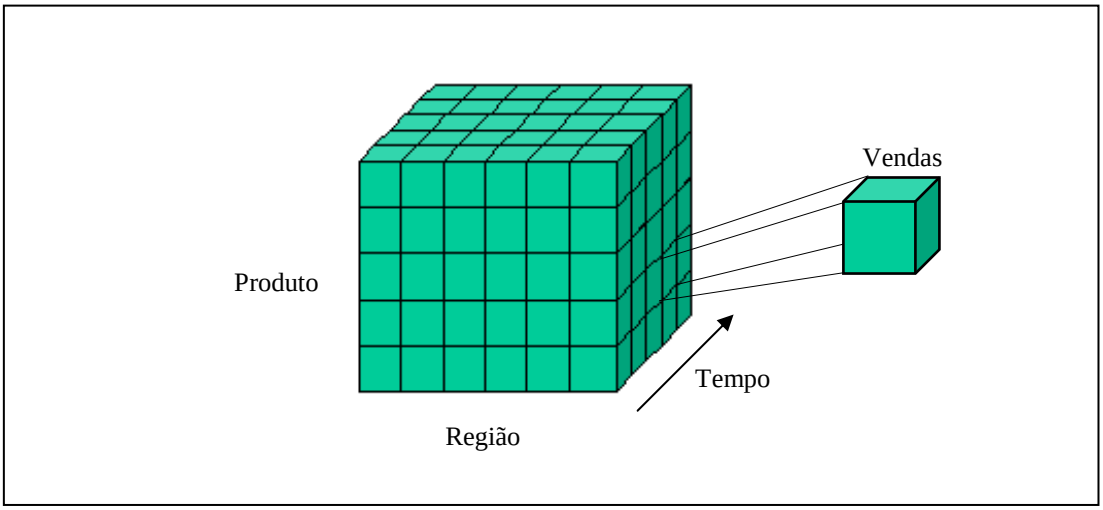


Figura 3.2: Representação de dados no modelo multidimensional

Na figura 3.2, os eixos do cubo representam as dimensões através das quais as medidas, células do cubo, podem ser analisadas. Nesse modelo, as vendas são vistas como um valor do espaço multidimensional criado pelas dimensões. Embora o exemplo seja apresentado com um número restrito de dimensões, a visão dos dados como um array multidimensional pode ser rapidamente generalizada para mais de três dimensões, derivando o conceito de um hipercubo. Cada dimensão é caracterizada por um conjunto de atributos cujas restrições determinam os fatos a serem recuperados. No esquema da figura 3.2, a dimensão Produto poderia ser caracterizada por atributos como nome, categoria, tipo da embalagem, marca, tamanho da embalagem e muitos outros. A restrição de uma categoria específica produzirá uma análise das vendas de todos os produtos que pertencem àquela categoria. Logo, a combinação de dimensões e restrições para determinar os fatos permite que os dados operacionais possam ser visualizados sob diferentes perspectivas (Figura 3.3).

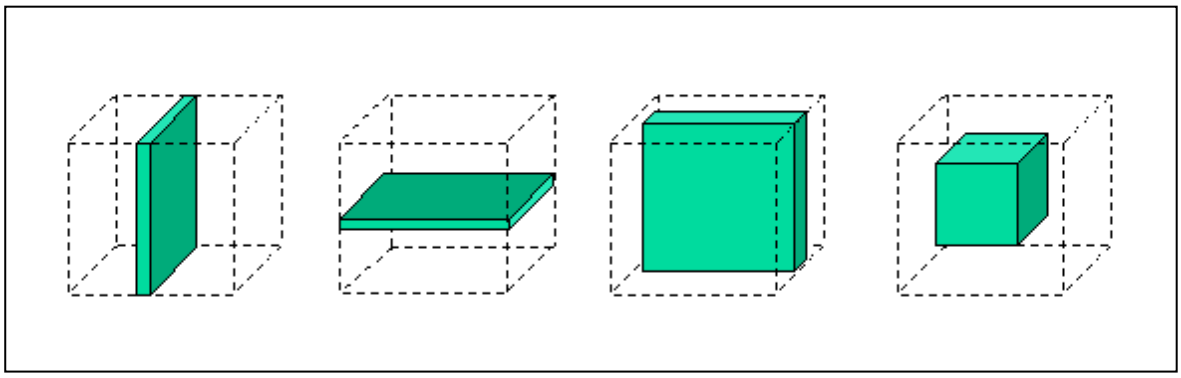


Figura 3.3: Visualização dos dados multidimensionais através de diferentes perspectivas

O modelo multidimensional, apesar de ter no *array* multidimensional sua base conceitual, pode ser implementado utilizando diversos paradigmas para o armazenamento físico dos dados, os quais determinam a classificação e as características de uma interface OLAP. Entenda-se paradigma para o armazenamento físico como o projeto físico da base de dados do *Data Warehouse*, que pode ser relacional, multidimensional ou orientado a objetos. Atualmente, existem duas principais arquiteturas de interfaces OLAP, também conhecidas como servidores OLAP, diferenciadas pelo projeto físico dos dados: MOLAP (*Multidimensional* OLAP) e ROLAP (*Relational* OLAP).

Interfaces MOLAP foram os primeiros tipos de interfaces OLAP a surgir no mercado. Essa classe de interface provê o gerenciamento de dados multidimensionais através de um sistema gerenciador de banco de dado multidimensional, o qual utiliza estruturas proprietárias, geralmente *arrays* multidimensionais, como unidade de armazenamento de dados. Bancos de dados multidimensionais são capazes de oferecer alta performance nas consultas, característica alcançada através da antecipação e restrições das diferentes formas que os dados podem ser acessados [Jarke *et al*., 1999]. A grande vantagem dessa abordagem é que as operações e consultas realizadas na visão conceitual multidimensional da interface são mapeadas diretamente para o modelo físico do banco de dados. Entretanto, a falta de padronização dos modelos lógicos utilizados e das operações e interfaces disponíveis para acesso aos dados, torna as interfaces totalmente dependentes do banco de dados utilizado. Essbase da Arbor Software, LightShip Server da Pilot Software e o Express da Oracle são exemplos de produtos que se enquadram nessa categoria.

A popularidade dos SGBDs relacionais fez surgir uma classe de sistemas OLAP que utilizasse o paradigma relacional como meio físico de armazenamento dos dados. Nessas interfaces, o uso de uma camada semântica acima do esquema relacional permite que os dados sejam apresentados ao usuário através do modelo multidimensional. Logo, Interfaces ou servidores ROLAP são definidos como sistemas OLAP que traduzem operações e consultas realizadas em um esquema multidimensional para um esquema relacional [Dinter *et al*., 1998] [Chaudhuri & Dayal, 1997]. Contudo, algumas ferramentas exigem que os dados estejam estruturados em um esquema relacional particular que facilite a tradução de consultas entre os dois modelos. Atualmente, a maioria dos *Data Warehouses* armazena suas informações em SGBDs relacionais, o que torna mais natural o uso de interfaces ROLAP como mecanismo de acesso aos dados. DSS server da MicroStrategy e seus produtos relacionados, Informix-MetaCube da Informix e Decision Suite da Information Advantage são exemplos de ferramentas ROLAP encontradas no mercado atualmente.

As duas arquiteturas possuem vantagens e desvantagens, o que impede a caracterização de uma melhor opção através de uma análise isolada. A escolha de uma solução deve ser baseada em parâmetros que definem cada interface em particular, e no ambiente projetado no qual a interface será utilizada. Interfaces MOLAP manipulam quantidades limitadas de dados em função da solução proprietária para armazenamento físico. Logo, a mesma deve ser aplicada para subconjuntos de dados provenientes de um repositório mais geral. Esses subconjuntos de dados, geralmente destinados a um departamento específico, representam *Data Warehouses* departamentais, conhecidos como *DataMarts*. Contudo, muitas empresas fazem uso de uma arquitetura mais centralizada, armazenando todos os dados destinados à tomada de decisões em um único *Data Warehouse*. Nesse ambiente, o grande volume de dados exige a maturidade dos sistemas gerenciadores de banco de dados relacionais e a conseqüente utilização de interfaces ROLAP como mecanismo de acesso aos dados.

3.4 Modelagem Dimensional

Sistemas de suporte à decisão, *em particular Data Warehouses*, são destinados a uma classe de usuários que necessita de uma visão simples e clara dos dados corporativos e que, ao mesmo tempo, apresente os objetos do negócio mais relevantes à tomada de decisões. As tradicionais técnicas de modelagem de dados para sistemas do ambiente operacional têm como foco principal a remoção de redundância nos dados e o conseqüente aumento da

performance na execução de transações. Porém, o uso de tais técnicas mostra-se inadequado para a concepção de uma visão corporativa destinada a gerentes e executivos, em função da complexidade dos esquemas gerados que, em sua maioria, englobam um grande número de entidades relacionadas entre si.

A complexidade descrita anteriormente somente pode ser contornada por usuários especialistas que entendam perfeitamente as entidades que constituem um esquema, assim como os relacionamentos necessários à navegação e execução de consultas. O homem de negócios, usuário de um sistema de suporte à decisão, dificilmente apresentará um perfil técnico necessário à elaboração de consultas que envolvam inúmeras entidades e junções. Vale ressaltar que o emprego das técnicas tradicionais de modelagem não apenas contribui para o aumento da complexidade dos esquemas de dados, mas degradam o desempenho de consultas necessárias à tomada de decisões. Na seção anterior, o modelo multidimensional foi apresentado como solução para a modelagem conceitual de dados em um *Data Warehouse* por prover esquemas corporativos simples, intuitivos e naturais em relação à maneira que usuários, em geral, observam os negócios da empresa.

Entretanto, pode-se inferir, também da seção anterior, que o modelo multidimensional só pode ser usado como modelo lógico de dados quando da utilização de um banco de dados multidimensional. É notório que, atualmente, ocorre uma predominância dos sistemas gerenciadores de banco de dados relacionais nas organizações de todo o mundo, seja no ambiente de processamento de transações ou em um ambiente projetado para a tomada de decisões. Desse modo, faz-se necessário a utilização de uma técnica de modelagem para repositórios relacionais que atenda aos requisitos de naturalidade, simplicidade e desempenho através da propagação das características do modelo conceitual multidimensional.

As dificuldades encontradas com a utilização de esquemas que seguem características operacionais para servir à SSDs sempre foram observadas por aqueles que visavam um ambiente mais compreensível e que pudesse oferecer uma melhor performance. Segundo [Kimball, *et al.*,1998], a compilação dessas observações deu origem ao conceito de

Modelagem Dimensional que, possivelmente, surgiu antes da abordagem entidade-relacionamento[5].

Modelagem Dimensional é uma técnica de modelagem lógica que apresenta os dados em um padrão intuitivo capaz de balancear performance e volume de dados [Kimball, 1996] [Kimball, *et al.*,1998] [Jarke *et al*., 1999]. Em um esquema do modelo dimensional podemos observar uma tabela dominante, chamada de tabela de fatos, cujos atributos representam medidas numéricas associadas ao negócio da empresa. Relacionando-se com a tabela de fatos, encontramos as tabelas de dimensões, que caracterizam as medidas numéricas citadas anteriormente. O esquema composto pela tabela de fatos e suas dimensões possui uma estrutura semelhante a uma estrela, daí a famosa denominação do esquema Estrela ou esquema em Estrela [Kimball, 1996] [Peterson, 1994] (figura 3.4) .

O esquema da figura 3.4 é o esquema relacional mais utilizado para modelagem de *Data Warehouses*, pois além de apresentar as característica já mencionadas, é exigido por muitas interfaces ROLAP como esquema relacional do repositório de dados [Dinter *et al*., 1998]. Tal exigência é resultado da perfeita correlação existente entre os modelos conceitual e lógico, o que permite uma tradução, não tão direta caso fosse utilizado um banco multidimensional, equivalente de operações.

A tabela de fatos no esquema Estrela é responsável por guardar o histórico das transações associadas com as atividades do negócio a serem modeladas [Peterson, 1994]. Geralmente, os atributos dessa tabela representam medidas numéricas ou fatos associados às transações e determinados pelas dimensões. No esquema da figura 3.4 podemos identificar a *Quantidade* e o *Valor_Reais* como medidas ou fatos que caracterizam uma transação. Na verdade, a tabela de fatos representa os relacionamentos muitos-para-muitos entre as tabelas de dimensões. Desse modo, cada tupla da tabela de fatos é unicamente identificada pelo conjunto das chaves primárias de cada dimensão.

[5] entidade-relacionamento é, provavelmente, a técnica mais conhecida para modelagem de banco de dados. É baseada no modelo entidade-relacionamento proposto por Peter Chen em 1976 [Chen, 1976] e, ao longo do tempo, evolui como resultado do trabalho de outros pesquisadores. O maior objetivo dessa abordagem é remover redundância nos dados e oferecer esquemas mais apropriados para o processamento transacional.

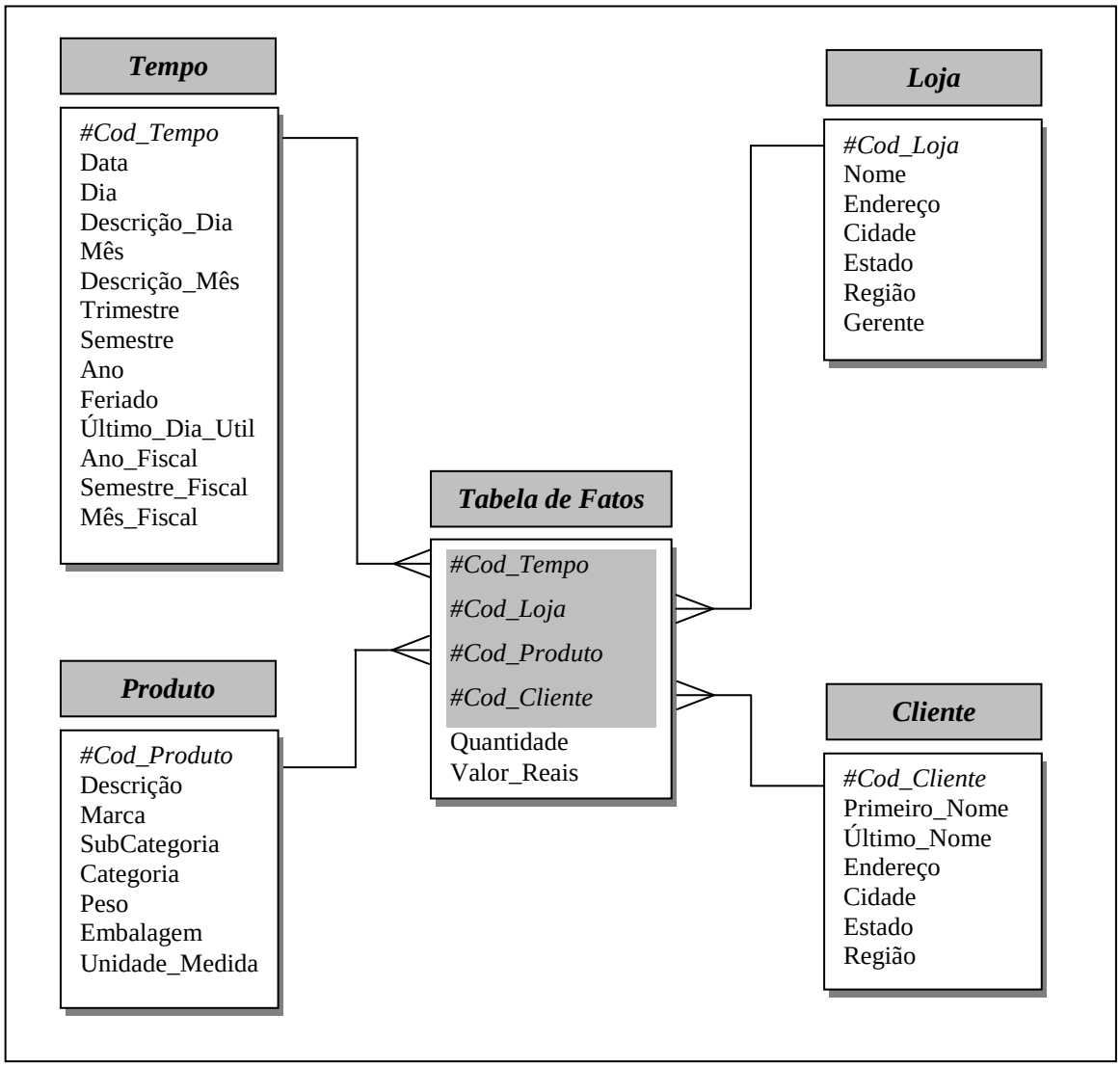


Figura 3.4: O esquema Estrela como estrutura relacional para *Data Warehouses*

Dois aspectos importantes da tabela de fatos que possuem grande influência no projeto de um esquema Estrela são a granularidade e a *esparsidade*[6]. A primeira, que representa o aspecto mais importante no projeto de um DW, refere-se ao nível de detalhe das transações a serem armazenadas. Quanto mais detalhes contiver uma transação armazenada, mais baixo será o nível da granularidade [Inmon, 1997]. A granularidade afeta diretamente o volume de dados na tabela de fatos e as classes de consultas que podem ser atendidas. *Esparsidade* é uma das principais características apresentadas pelos dados em um *Data Warehouse*. Os fatos ou medidas numéricas que caracterizam os negócios de uma empresa somente existirão para uma pequena fração do produto cartesiano das dimensões. Logo, a depender do paradigma de armazenamento físico utilizado, esse aspecto será de grande

[6] *Esparsidade*, do inglês *sparcity*.

influência para questões como volume e performance nas consultas dos dados armazenados. Em bancos de dados multidimensionais, essa questão é de extrema importância visto que as estruturas de dados proprietárias representam combinações de *arrays* multidimensionais que implementam diretamente o modelo conceitual multidimensional [Dinter *et al*., 1998]. A esparsidade na tabela de fatos é tratada de forma implícita seguindo uma regra crucial: as transações armazenadas só devem representar tuplas válidas para a combinação das dimensões [Raden, 1996] [Kimball *et al*., 1998].

Atributos da tabela de fatos podem ser caracterizados como aditivos, semi-aditivos e não-aditivos [Kimball, 1996]. Atributos aditivos, e por conseqüência numéricos, são os mais valiosos e importantes nas atividades de acesso a um esquema estrela, pois as atividades de análise sobre dados em um *Data Warehouse* tendem a utilizar, principalmente, a agregação de fatos para compor relatórios corporativos. Porém, o termo aditivo ou perfeitamente aditivo, nesse contexto, não traduz apenas a natureza numérica do atributo, mas caracteriza todos os atributos que podem ser somados ao longo de todas as dimensões relacionadas com a tabela de fatos. Geralmente, atributos que representam quantidades, vendas e ocorrências são perfeitamente aditivos.

São chamados de semi-aditivos os atributos cuja soma ao longo de uma ou mais dimensões não possui qualquer significado. Saldos de contas bancárias e número de clientes que compraram um determinado produto são exemplos dessa categoria. No primeiro caso, somar os saldos de uma conta bancária ao longo do tempo produzirá um valor acumulado sem significado, porém o valor obtido ainda pode ser utilizado na elaboração de uma média em um certo intervalo de tempo. Em um esquema que registre o histórico de transações para a compra de produtos, o número de clientes que compraram um determinado produto não é aditivo em relação à dimensão produto. Para melhor visualizar o segundo exemplo, considere dois registros que representam as venda de computadores e impressoras em uma loja particular, em um determinado dia. Considere, também, que o registro das vendas de computadores tenha armazenado 10 clientes e o segundo registro, o das impressoras, 20 clientes. Pode-se observar que a soma do número de clientes que compraram computadores ou impressoras não pode ser obtido através de uma simples agregação, pois os valores dos registros não representam, necessariamente, conjuntos disjuntos. Logo, o número real de clientes poderia ser qualquer valor entre 20 e 30.

Finalmente, os atributos não-aditivos caracterizam os atributos cuja soma dos valores não possui significado ao longo das dimensões do esquema. Alguns atributos cujos valores expressam intensidade, a exemplo da temperatura, são exemplos para essa categoria. Contudo, a média de atributos numéricos não-aditivos ao longo das dimensões pode representar medidas com algum significado. Fatos textuais também são exemplos de atributos não-aditivos. Geralmente, são descrições associadas aos fatos numéricos, porém com campos não suficientes para formar uma outra dimensão [Peterson, 1994].

Associadas à tabela de fatos, encontramos as tabelas de dimensões, responsáveis por descrever as transações armazenadas na tabela de fatos. As tabelas de dimensões, ou apenas dimensões como são usualmente referenciadas, representam as principais classes que definem os fatos [Raden, 1996]. Na figura 3.4, pode-se observar quatro dimensões: *Produto*, *Loja*, *Tempo* e *Cliente*. Nas tabelas de dimensões, são encontrados atributos que descrevem a classe de uma dimensão particular. A dimensão *Produto*, por exemplo, pode conter atributos como *nome*, *categoria e tamanho_da_embalagem*. Esses atributos, que apresentam um caráter textual independente do tipo de dado [Kimball, 1996], são utilizados por usuários como meio de seleção e agregação de fatos. Segundo [Kimball, 1996], 80% das operações realizadas em um esquema Estrela por um analista durante um dia de interação usuário-sistema são análises a respeito do conteúdo de dimensões. Essa observação reflete a importância da dimensão e de seus atributos para a atividade de tomada de decisões. Com base na observação anterior, Kimball em [Kimball, 1996] e [Kimball, 1998] também sugere que as tabelas de dimensões não sejam normalizadas a fim de facilitar a exploração de informações por parte dos usuários.

No projeto de uma dimensão, é interessante que seus atributos possam formar estruturas hierárquicas onde os diferentes níveis sejam traduzidos como níveis de agregação sobre os quais o usuário possa explorar os fatos. Na dimensão *Loja* da figura 3.4, os atributos *cidade*, *estado* e *região* formam uma hierarquia que pode ser utilizada como forma de analisar os fatos em relação à localidade. A figura 3.5 apresenta instâncias de hierarquias encontradas nas dimensões Loja e Tempo da figura 3.4. Uma boa prática no projeto das dimensões é tentar reutilizar dimensões para diferentes esquemas dentro de uma mesma organização a fim de que diferentes departamentos tenham a mesma perspectiva com relação aos fatos [Kimball, 1998].

Embora, de maneira geral, não exista uma conformidade nos tipos de dimensões utilizadas entre diferentes esquemas para domínios distintos, uma dimensão particular,

chamada dimensão Tempo, está quase sempre presente na modelagem do esquema em Estrela.

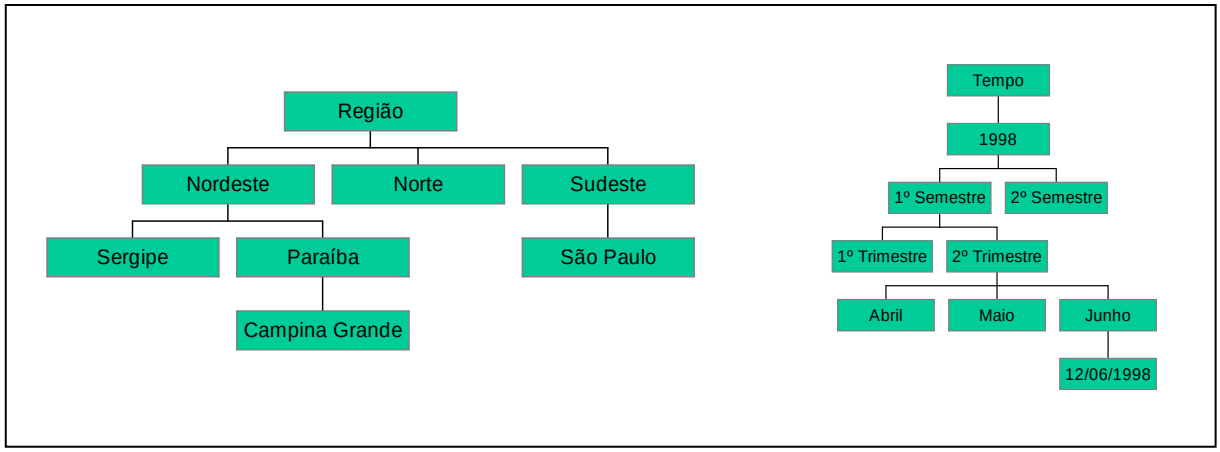


Figura 3.5: Exemplos de instâncias de hierarquias encontradas em tabelas de dimensões

A tabela de fatos é responsável por guardar o histórico das transações, que normalmente é analisado em relação ao tempo. Logo, todos as transações, seja qual for a granularidade, devem estar relacionadas com algum atributo que possa determinar o exato instante de sua ocorrência. Em um primeiro contato com essa necessidade, muitos projetistas tendem a sugerir a adição de um atributo do tipo *SQL Date* na tabela de fatos, o que parece ser uma solução correta visto que as transações estarão intimamente ligadas a um atributo do tempo. Porém, para efeito de análise em alguns domínios, um atributo *SQL Date* não é adequado pelo fato de obscurecer períodos de tempo como trimestres, semestres e finais de semana. Além disso, existem calendários particulares, como períodos fiscais e feriados, que não podem ser inferidos a partir de um simples atributo do tipo *Date*. A solução real para esse problema é a construção de uma dimensão Tempo, a exemplo da figura 3.4, que descreva um atributo do tipo *Date* a partir de dependências como trimestre, semestre, período fiscal, feriado, final de semana e último dia útil do mês. Uma lista mais completa de atributos para a construção de uma dimensão Tempo pode ser encontrada em [Kimball, 1996].

A mais importante variação do esquema em Estrela, o esquema *Snowflake* (figura 3.6) , remove atributos com baixa cardinalidade das dimensões principais[7] e os transfere para

[7] Utilizamos o termo dimensões principais para referenciar as tabelas de dimensões que estão diretamente relacionadas com a tabela de fatos.

tabelas de dimensões secundárias. Na verdade, a atividade que acabamos de descrever baseia-se no processo de normalização de esquemas [Elmasri & Navathe, 1994]. Embora exista um consenso geral [Kimball, 1997] [Poe *et al*., 1998] [Chaudhuri & Dayal, 1997] de que uma estrutura não normalizada das tabelas de dimensões pode ser mais apropriada para a exploração de atributos, muitos projetistas exploram o esquema *Snowflake* na tentativa de diminuir redundâncias nos dados das tabelas de dimensões, favorecendo e simplificando manutenções futuras. Porém, essa justificativa não é aceita por [Kimball, 1997], quando argumenta que manutenções intensas são realizadas no ambiente operacional antes dos dados serem carregados para o esquema dimensional. Talvez, o resultado mais característico dessa abordagem seja a representação explícita de hierarquias de atributos dentro das dimensões. No esquema da figura 3.6 podemos observar a hierarquia explícita que surge a partir da dimensão *Loja.*

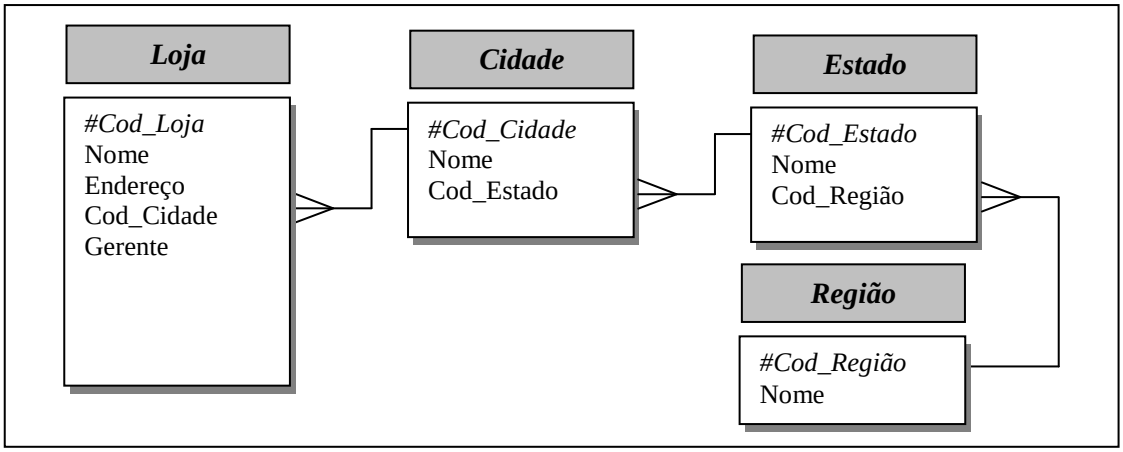


Figura 3.6: Hierarquia explícita no esquema *Snowflake*

3.5 Conclusões

Neste capítulo, introduzimos o conceito de Sistemas de Suporte à Decisão e suas principais tecnologias. Apresentamos a motivação por trás do surgimento do *Data Warehouse*, maior destaque dos SSD baseados em dados, a partir das principais dificuldades encontradas no ambiente operacional para a atividade de tomada de decisões. O modelo Multidimensional, largamente explorado pelas interfaces OLAP, foi apresentado como solução para a apresentação e organização de informações relevantes a usuários que não possuem um perfil técnico para a elaboração de consultas em modelos de dados tradicionais. O esquema em Estrela, principal estrutura relacional utilizada nos projetos de *Data*

Warehouses, foi explorado com o objetivo de apresentar a solução encontrada para oferecer um caráter dimensional a esquemas relacionais. O próximo capítulo, foco dessa dissertação, fará uso dos conceitos aqui introduzidos, em especial o esquema em Estrela e sua principal variação, o esquema *Snowflake*, como base para o projeto de nossa interface que visa propiciar um mecanismo mais natural de acesso a dados para os novos usuários[8] dos atuais Sistemas de Suporte à Decisão.

[8] O termo novos usuários refere-se a todos os usuários que serão contemplados com o acesso a Sistemas de Suporte à Decisão baseados na *Web*. Essa nova classe não está restrita a gerentes e analistas, pois a infra-estrutura da *Web* permite a difusão de informação por toda a empresa, de maneira fácil e pouco onerosa.

Capítulo 4

Arquitetura de Interface para SSDs baseados na *Web*

O capítulo 3 introduziu o conceito de Sistemas de Suporte à Decisão, com ênfase no ambiente projetado para *Data Warehouses*, e mostrou sua relevância para a evolução e sobrevivência das atuais empresas participantes em mercados competitivos e globalizados. Também no capítulo 3, foi apresentado o novo conceito de Sistemas de Suporte à Decisão baseados na *Web* [Power, 1998] como evolução dos tradicionais SSDs através do paradigma de implantação.

Essa nova categoria de SSD, que reduz barreiras tecnológicas no tocante à distribuição de informação, faz surgir no ambiente de suporte à decisão uma nova classe de usuários, antes restrita ao ambiente operacional, cujos perfil e conhecimento técnicos diferem daqueles encontrados em analistas e gerentes envolvidos diretamente na atividade de tomada de decisão. Como conseqüência, desenvolve-se uma democracia de informação corporativa, citada em [Power, 2000a] e [Pilot Software], que é fruto direto da utilização da infra-estrutura da Internet como meio de propagação de conhecimento. Em [Pilot Software], a informação nas mãos daqueles que lidam mais diretamente com o cliente é vista como um alicerce para o sucesso corporativo e individual. [Power, 2000a] acredita que esse novo conceito, ao atender novos consulentes, provenientes de dentro da organização ou fora dela, trará benefícios na utilização e desenvolvimento dos futuros SSDs.

No presente momento, vale ressaltar uma questão importante sobre Sistemas de Suporte à Decisão que não foi explorada no capítulo 3: Sistemas de Suporte à Decisão não são restritos ao ambiente corporativo de bancos, empresas de atacado, supermercados e comércio em geral, apesar da motivação apresentada no capítulo anterior da utilização nesses contextos. Um sistema de informação que apoie, informe ou esclareça questões de um usuário pode ser utilizado ou empregado em ambientes como bibliotecas, hospitais, páginas eletrônicas na Internet que versem sobre diversas áreas de conhecimento, centros de lazer e

turismo. Tomemos como exemplo um shopping virtual. Nele, usuários procuram por produtos com base em características, preços, localidade para compra, custo de frete, opiniões de outros usuários. Logo, podemos considerar que um usuário, ao interagir com sites dessa natureza, está fazendo uso de um Sistema de Suporte à Decisão e, por conseguinte, necessita de uma interface amigável para suas consultas. Desse modo, surgem, também, classes de usuários que assemelham-se em perfil técnico aos novos usuários do meio empresarial citados anteriormente.

O leitor atencioso depreenderá, também do capítulo 3, a importância das ferramentas de acesso a dados em um SSD. Responsáveis diretamente pelo sucesso e evolução de um SSD, as interfaces de interação usuário-sistema utilizam abstrações e modelos de dados conceituais com o objetivo de tornar mais intuitivo e natural a exploração e aquisição de informações. Some-se a isso o emprego das tradicionais metáforas de apontar, clicar e arrastar presentes nos atuais sistemas gráficos. Porém, todo o esforço empregado na concepção de tais interfaces não é suficiente para subtrair, por completo, a necessidade de treinamento do usuário final. Além disso, existem usuários que mostram-se resistentes ao aprendizado dos diversos "sabores" de ferramentas existentes no mercado. Contudo, essas barreiras ainda conseguem ser ultrapassadas pelo interesse profissional no uso das aplicações mencionadas.

Porém, quão útil são essas interfaces para os *novos usuários*[1] ? Será que usuários desprovidos de interesse profissional mostrar-se-ão motivados a explorar as funcionalidades de um sistema a fim de obter alguma interação ? E quanto à casualidade da Internet, que nunca será benevolente com tempo despendido em aprendizado ? Essas são, entre muitas outras, questões de comportamento que devem ser apontadas como desafios a serem superados para que os SSDs baseados na *Web* atingiam seus propósitos. Portanto, apesar da visão promissora desse novo paradigma, ainda há muito por fazer para que os benefícios da *Web* no desenvolvimento de SSDs sejam efetivamente percebidos [Bhargava & Power, 2001].

[1] No contexto desse capítulo, a expressão *novos usuários* compreende todos os indivíduos que nunca tiveram experiências anteriores com interfaces para Sistemas de Suporte à Decisão, seja em um ambiente empresarial ou fora dele, e que apresentam-se, em sua maioria, como usuários casuais.

A discussão do parágrafo anterior representa a motivação para o nosso trabalho de dissertação. A observação da impossibilidade da utilização das tradicionais ferramentas de acesso a dados para SSDs, em especial *Data Warehouse*, em combinação com a infra-estrutura da Internet para atender aos *novos usuários* mostrou-se uma área de pesquisa interessante e de grande relevância. A partir daquele momento, restava-nos sugerir uma solução viável para o problema e apresentar meios para comprovar ou contestar a utilidade da nossa proposta.

Neste capítulo, apresentamos uma arquitetura de interface como solução para o problema em questão e especificamos um projeto e implementação com a finalidade de validar essa arquitetura. A validação dessa arquitetura é restrita à sua implementação. Para realmente comprovar que a nova arquitetura atende aos objetivos da pesquisa e contribui para o avanço do estado da arte em intefaces para SSDs, faz-se necessário uma avalição empírica, com experimentos com usuários, e uma comparação com outras interfaces em contextos semelhantes. Essa avaliação será sugerida como trabalhos futuros dessa dissertação (ver capítulo 6). O projeto a ser detalhado nas próximas seções tem forte relação com as atuais ferramentas de acesso a *Data Warehouses* e destina-se à recuperação de dados presentes em sistemas gerenciadores de banco de dados relacionais. Essas duas características foram ditadas pela maturidade e predominância do modelo relacional, assim como pela necessidade de apresentar uma solução que possa ser reutilizada pelas atuais interfaces presentes no mercado.

Este capítulo está estruturado como segue. A primeira seção apresenta a nossa solução e as justificativas para sua adoção. A seção 4.2 exemplifica a utilização de uma interface baseada na solução proposta. A seção 4.3 apresenta a arquitetura abstrata representante de nossa solução e descreve seus principais módulos. Dedicamos a seção 4.4 ao desenvolvimento de uma infra-estrutura básica para a arquitetura descrita. Essa infra-estrutura é especificada através das fases de análise, projeto e implementação. Durante a descrição do projeto, detalhamos e justificamos as soluções adotadas para modelagem e implementação.

4.1 A Solução Proposta

Linguagens de consulta de alto nível como *SQL* foram concebidas com o objetivo de abstrair a estrutura física dos dados e, dessa forma, permitir a manipulação de dados em

SGBDs. Por ser uma linguagem declarativa, *SQL* apresenta-se como uma linguagem de fácil aprendizado. Porém, essa facilidade só é observada por profissionais da área de informática, particularmente, desenvolvedores de aplicações. Em aplicações do ambiente operacional, usuários finais fazem uso intensivo da linguagem *SQL*, embora de forma indireta através de comandos e menus [van der Lans, 1993]. Essa forma estática de apresentação de informação através de consultas pré-definidas, apesar da facilidade de manipulação e aprendizado, não pode ser a única forma de interação utilizada em Sistemas de Suporte à Decisão, os quais apresentam espaços de conhecimento dinâmico com grande número de variáveis, geralmente utilizadas na elaboração de consultas.

Ferramentas visuais de consulta de propósito geral, a exemplo do *Paradox* e do *Microsoft Access*, utilizam representações gráficas de conceitos do modelo relacional como forma de apresentar, de forma mais amigável, esquemas de dados para usuários casuais [Ramakrishnan, 1998]. A principal característica dessa categoria de interface, fortemente influenciada pela linguagem *QBE* (*Query-By-Example*) [Elmasri & Navathe, 1994], é a formulação automática de consultas em *SQL* através da manipulação de elementos como entidades, atributos e relacionamentos. Esse mecanismo de interação elimina a dependência de consultas pré-definidas e subtrai a necessidade do aprendizado da sintaxe da linguagem *SQL*. Contudo, o usuário precisa ter um perfeito entendimento do esquema de banco de dados a fim de elaborar suas consultas. Com isso, fica claro que esse tipo de interface não é adequada aos *novos usuários*, desconhecedores de modelos de dados e conceitos como chaves primárias, chaves estrangeiras e junções.

As atuais ferramentas especialmente projetadas para sistemas de suporte à decisão, a exemplo das interfaces ROLAP e das interfaces de consulta *ad-hoc*[2] , são compostas por uma grande variedade de mecanismos de interação usuário-sistema. Dentre os mais comuns, estão: linguagem *SQL*, interfaces gráficas influenciadas pela linguagem *QBE*, interfaces gráficas baseadas no modelo multidimensional, consultas pré-definidas ou baseadas em formulários. Interfaces ROLAP, em particular, utilizam uma camada conceitual acima da estrutura física do banco de dados, que representa uma abstração de dados mais natural para o usuário final. Essa representação abstrata do banco de dados pode ser customizada para diferentes tipos de

[2] Nesse contexto, interfaces de consulta *ad-hoc* representam as ferramentas de consultas especialmente projetadas para ambientes de suporte à decisão, mas que não apresentam características suficientes para serem categorizadas como interfaces OLAP.

usuários, com restrições de visões e terminologia apropriada [Rudloff, 1996]. O modelo conceitual utilizado nessas ferramentas possui o mérito de ser facilmente reconhecido como a tradução perfeita para modelos de negócios do usuário final. Apesar dessa característica e da extensa variedade de mecanismos de consulta, o treinamento específico é parte integrante do processo de utilização dessa categoria de interface, pois a falta de padrões origina diversos "sabores" de aplicações que diferem em terminologia, funcionalidades e passos necessários à formulação de consultas. Com efeito, somente gerentes e analistas com um histórico de envolvimento em SSDs e com um mínimo perfil técnico estarão aptos a fazer uso dessas ferramentas. Logo, podemos inferir que a falta de motivação profissional, a heterogeneidade de conhecimento técnico e a impossibilidade de treinamento de usuários na Internet impedem a adoção dessas interfaces para os *novos usuários*.

O segundo capitulo desta dissertação foi dedicado às Interfaces em Linguagem Natural para Banco de Dados. A principal vantagem dessa abordagem em relação às apresentadas anteriormente é a promessa de um sistema no qual o usuário não precise aprender uma linguagem artificial, nem conhecer estruturas de dados internas aos SGBDs a fim de elaborar consultas [Androutsopoulos *et al*., 1995] [Copestake & Jones, 1989] [Savadovsky, 1988]. Com isso, essa categoria, à primeira vista, parece apresentar parte da funcionalidade que estamos procurando para uma interface que sirva aos *novos usuários.* Entretanto, evidenciamos no capítulo 2 que essa característica, apresentar uso natural e transparente, ainda não foi alcançada por completo na prática em função dos problemas intrínsecos das ILNBDs apresentados na seção 2.2. Dentre esses problemas, aquele que mostra-se como maior obstáculo a uma grande aceitação das ILNBDs como ferramentas de acesso a dados é a necessidade do perfeito entendimento das limitações lingüísticas e semânticas impostas às consultas [Androutsopoulos *et al*., 1995] [Copestake & Jones, 1989] [Ogden, 1985]. Uma outra questão importante, alvo de pesquisas durante anos e que ainda não foi resolvida por completo, é a transportabilidade para diferentes aplicações. As atuais ILNBDs de propósito geral necessitam de uma extensa camada semântica que deve ser construída para cada novo esquema de dados a ser utilizado.

Apesar dos obstáculos apresentados pelas ILNBDs, acreditamos que uma abordagem baseada em linguagem natural ainda pode ser a mais apropriada para nosso problema em questão, principalmente, se combinada com outras abordagens que minimizam os problemas inerentes às ILNBDs. Acreditamos que esses obstáculos podem ser contornados através da

escolha de uma arquitetura que procure minimizar questões como limitações e restrições ocultas, e transportabilidade entre domínios. Nas próximas seções, procuraremos esclarecer as principais característica da nossa solução.

Na seção 2.4.2 apresentamos duas abordagens para minimizar o impacto das restrições e limitações desconhecidas em uma ILNBD. Uma delas, cuja idéia principal é definir explicitamente um subconjunto restrito[3] da linguagem natural, constitui a base de nossa solução. As atuais ILNBDs de propósito geral, principalmente as que utilizam a arquitetura da figura 2.6, não explicitam as construções passíveis de processamento, o que gera dúvida e insegurança na elaboração de consultas por parte de usuários casuais. Por esse motivo, nossa interface é baseada no conceito de padrões sintáticos que contribuem para a *habitability* do sistema. Essa abordagem foi utilizada em [Epstein, 1985], [Shankar & Yung, 2000] e [Watson, 1999], embora com particularidades distintas. A pesquisa realizada em [Ogden, 1985], ao analisar os fatores humanos envolvidos nos sistemas de consulta em linguagem natural, revela que usuários casuais estariam aptos a aprender um conjunto mínimo de regras para a elaboração de perguntas válidas. Esse resultado sugere que o uso de um pequeno conjunto de padrões sintáticos, ainda que flexíveis, pode prover uma visão clara das limitações e restrições de uma interface e encorajar a formulação de perguntas por usuários ocasionais.

A principal característica dos sistemas que empregam o uso de padrões sintáticos no processamento de linguagem natural é a ausência de uma gramática, a nível sintático e semântico, que dite regras na construção de sentenças válidas. O processamento de linguagem natural nessas interfaces baseia-se na extração de palavras chaves e na semântica dos padrões utilizados. Existem duas conseqüências diretas dessa característica. A primeira, de caráter positivo, é que os sistemas oferecem uma maior flexibilidade sintática e regras gramaticais não invalidam uma pergunta. A outra conseqüência, de caráter negativo, é que o sistema pode processar perguntas que, apesar de não apresentarem sentido, estão em conformidade com algum padrão definido *a priori*. Em resumo, parte da responsabilidade do processamento correto de sentenças está nas mãos dos usuários.

[3] No desenvolvimento do segundo capítulo, esclarecemos que as atuais Interfaces em Linguagem Natural para Banco de Dados utilizam, na verdade, um dialeto restrito de uma linguagem natural. Nesse momento, quando utilizamos o termo subconjunto restrito, estamos nos referindo a um subconjunto ainda menor e com restrições explícitas.

Nossa solução têm por objetivo traduzir para *SQL* uma pergunta em linguagem natural. Logo, o paradigma de armazenamento que pretendemos atender é o ROLAP. Essa características foi ditada pela maturidade e predominância do modelo relacional, assim como pela ausência de uma linguagem padrão para bancos de dados multidimensionais. Talvez, a linguagem MDX (Multidimensional Expression) da Microsoft torne-se um padrão de fato para consultas OLAP.

A linguagem MDX, definida nas extensões OLAP do OLE DB, é utilizada para manipular informações multidimensionais no Microsoft SQL Server Analysis Services. MDX é similar, em muitos aspectos, à linguagem SQL, porém é muito mais eficiente e intuitiva para consultas OLAP. Contudo, tal linguagem não constitui, atualmente, um padrão e não é suportada pelos principais fornecedores de bancos de dados multidimensionais e relacionais.

A outra principal característica da nossa solução é a utilização do modelo conceitual multidimensional, largamente utilizado nas interfaces OLAP, como forma de abstração e representação de informação para o usuário final. A facilidade e transparência desse modelo torna-o adequado aos nossos *novos usuários,* pois a existência de apenas dois grandes conceitos, fatos e dimensões, não apresenta obstáculos à compreensão das informações que podem ser manipuladas. Em resumo, o modelo multidimensional subtrai a necessidade do conhecimento de conceitos de caráter técnico como esquemas relacionais, junções, chaves primárias e estrangeiras. Como resultado dessa escolha, apoiaremo-nos na modelagem dimensional para prover um melhor mapeamento entre esquemas conceituais e esquemas lógicos, uma que vez nossa solução busca oferecer uma interface para SGBDs relacionais.

A apresentação da última característica pode inflamar o leitor a levantar questões como: "A utilização da modelagem dimensional não irá restringir as aplicações que podem ser contempladas pela arquitetura ?"; "Caso não haja restrição para as aplicações, existirá uma correspondência entre os esquemas atuais, geralmente modelados através da abordagem entidade-relacionamento (ER), e algum esquema dimensional ?".

A primeira questão pode ser analisada sob dois diferentes aspectos: esquema de dados e domínio de aplicação. Em relação ao primeiro aspecto, a primeira questão terá uma resposta afirmativa. Somente as aplicações que apresentarem esquemas de dados suportados pela arquitetura da interface poderão ser contempladas. Essa limitação é intencional e tem por objetivo reduzir a problemática de ILNBDs genéricas. Contudo, os esquemas de dados

resultantes da modelagem dimensional não restringem domínios de aplicações. Logo, o esquema Estrela e suas variações podem modelar, entre muitas outras, aplicações relacionadas com seguros, sistemas hospitalares, meteorologia e sistemas de informações de saúde. Dúvidas em relação à utilização da modelagem dimensional para sistemas não relacionados com vendas constitui um dos maiores mitos a respeito dessa disciplina [Kimball, 1997].

A segunda pergunta vem geralmente acompanhada por falsas afirmações como: "O esquema em Estrela não parece conter todas as informações do correspondente modelo ER" ou "O esquema em Estrela só é utilizado para armazenar dados sumariados" [Kimball, 1997]. Mais uma vez, salientamos que dúvidas dessa natureza representam mitos em relação à disciplina modelagem dimensional. Todo esquema ER pode possuir um esquema dimensional correspondente. A chave para o perfeito entendimento dessa afirmação é compreender que um esquema ER pode estar relacionado como um ou mais esquemas dimensionais. Sendo assim, o leitor pode esperar o aparecimento de diversas estruturas em estrela, representantes dos principais processos de uma aplicação, para um único diagrama ER.

A utilização do esquema Estrela e de sua mais importante variação, o esquema *Snowflake*, como esquemas relacionais para nossa solução não constitui somente um resultado da discussão anterior, mas representa uma exigência para que nossa arquitetura atinja seus propósitos. Essa exigência irá contribuir em dois aspectos importantes: a) através dela, poderemos fazer fortes suposições a respeito dos esquemas de dados utilizados e com isso facilitar o processamento de linguagem natural; b) a utilização de um único esquema de dados permite a criação de padrões sintáticos que independem do domínio da aplicação, o que ocasionaria uma transportabilidade a nível de aplicação.

Da reunião das características examinadas anteriormente, podemos inferir outros aspectos importantes da nossa solução. Como não utilizaremos gramáticas sintáticas ou semânticas para o processamento em linguagem natural, aplicaremos o conhecimento extraído de três principais fontes: padrões sintáticos, modelo de metadados semânticos e modelo de metadados técnicos. Os dois últimos conceitos serão esclarecidos e detalhados mais tarde neste capítulo. Por ora, podemos adiantar que esses dois modelos representam informações a respeito dos esquemas de dados conceituais e lógicos, respectivamente, utilizados por uma determinada aplicação.

4.2 Exemplo de Funcionamento da Interface

Como forma de tornar concretas as características descritas na seção anterior, e prover uma base para um melhor entendimento das seções subseqüentes, apresentaremos um pequeno estudo de caso que exemplificará o uso da nossa interface. Os exemplos aqui tratados são resultados do projeto e implementação da infra-estrutura básica para a arquitetura proposta.

Nossa solução exige que o esquema de dados da aplicação a ser contemplada pela interface possua uma das seguintes estruturas: Estrela ou *Snowflake*. Para os exemplos que seguem, consideremos o esquema lógico *Snowflake* da figura 4.1. Nele, um esquema *Snowflake*, encontramos uma tabela de fatos, *Fatos*, responsável por armazenar transações relacionadas com a venda de produtos em uma determinada loja, em um dado momento. Por conseqüência, as tabelas de dimensões presentes são: *Tempo*, *Produto* e *Loja*. A última encontra-se normalizada através dos relacionamentos entre as tabelas *Regional* e *Nacional*. O esquema lógico da figura 4.1 deve ser totalmente especificado na Interface, provavelmente por um Administrador de dados, a fim de que consultas a nível conceitual possam ser traduzidas para o nível lógico. Essa especificação deve englobar a definição das entidades, seus atributos e relacionamentos. O conjunto dessas informações será utilizado, posteriormente, para a criação do esquema conceitual multidimensional a ser apresentado ao usuário final.

Uma vez que o esquema lógico tenha sido especificado na Interface, o Administrador poderá construir um esquema conceitual multidimensional correspondente. Esse processo compreende a definição de entidades do esquema conceitual seguida pela definição do mapeamento entre elementos conceituais e lógicos. A figura 4.2 representa um esquema conceitual criado a partir do esquema lógico da figura 4.1. O esquema da figura 4.2 pode ser visto como um hipercubo no qual as células representam as medidas *Vendas* e *Valor Vendido*, que são unicamente determinadas pelas dimensões do negócio: Produto, Tempo e Loja. Deve-se observar que o esquema conceitual apresentado ao usuário final (figura 4.2) não apresenta conceitos como chaves primárias, chaves estrangeiras e junções. As tabelas Loja, Regional e Nacional do esquema lógico (figura 4.1) são vistas, conceitualmente, como uma única entidade: a dimensão Loja. Essa característica subtrai a necessidade de conhecimento de relacionamentos entre entidades do esquema de dados lógico.

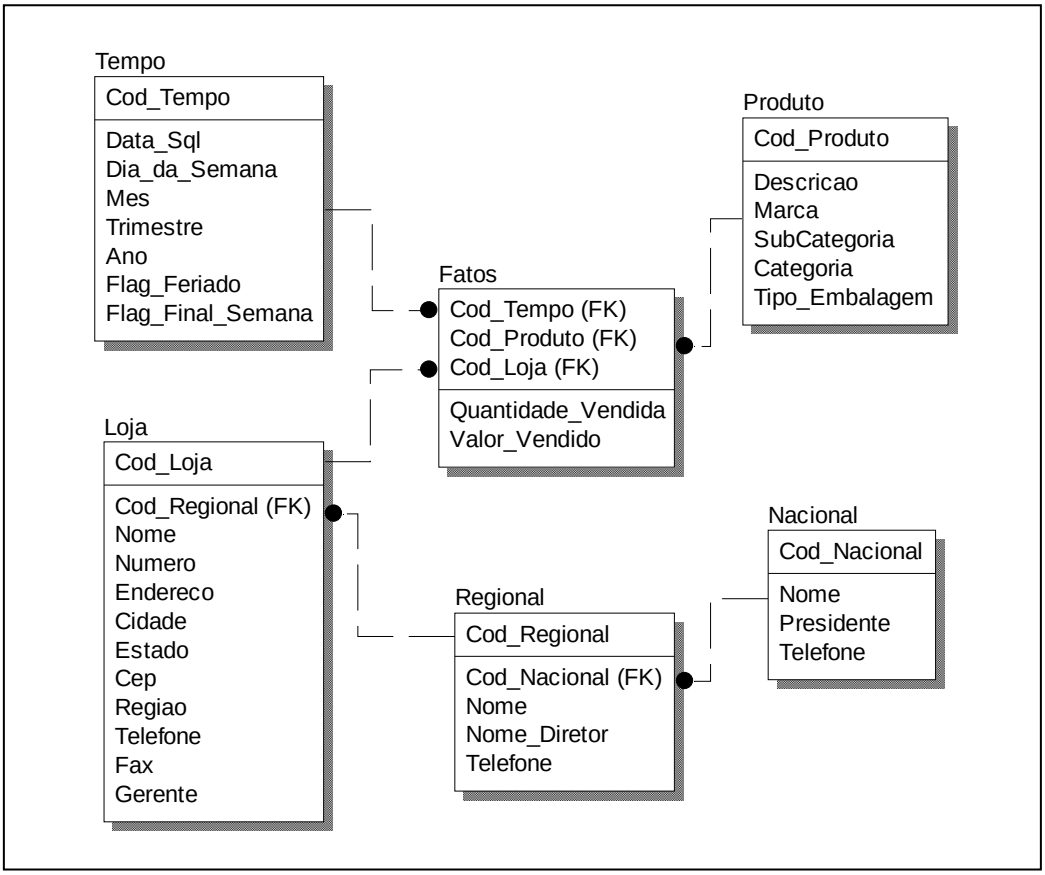


Figura 4.1: Esquema lógico de dados suportado pela Interface.

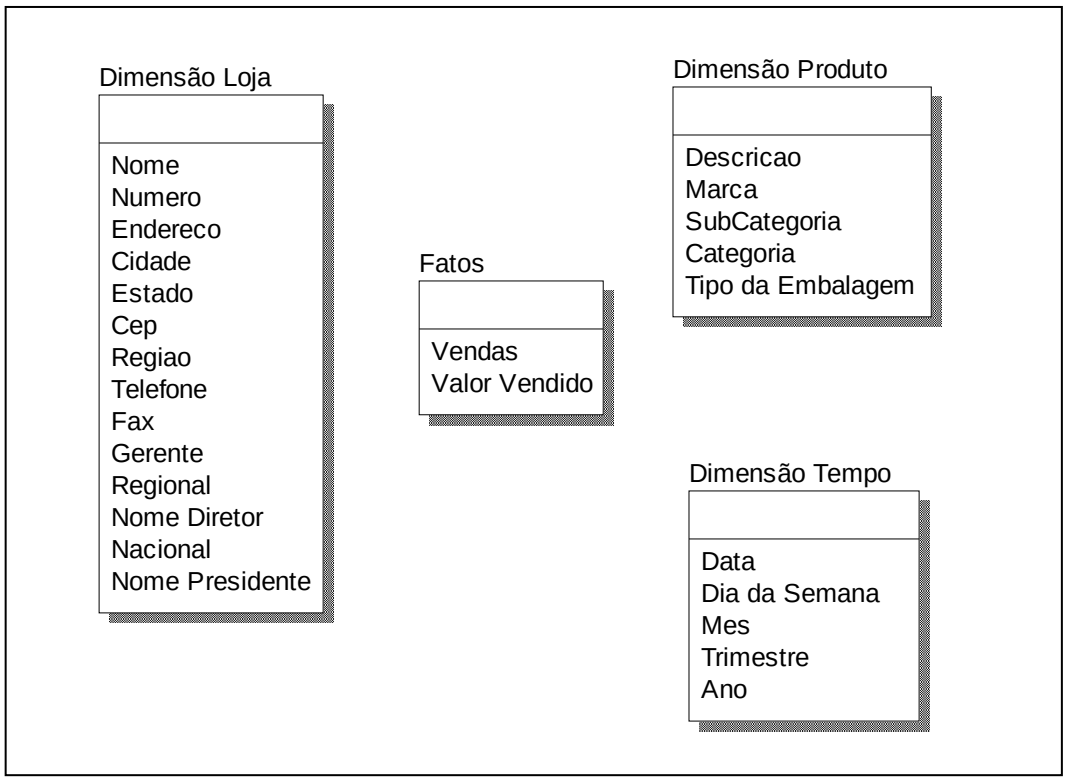


Figura 4.2: Esquema conceitual multidimensional.

O esquema apresentado na Figura 4.2 não captura toda a semântica necessária para uma camada semântica multidimensional. Contudo, atende ao papel ilustrativo de exemplificar a ausência de conceitos técnicos ao usuário final. Na seção 4.4, será apresentado um modelo de metados semântico responsável por capturar toda a semântica de um esquema conceitual manipulado pela interface. Nele, pode-se encontrar conceitos como hierarquias, níveis de hierarquias, visibilidade de atributos e caracterização de atributos quanto ao grau de agregação: aditivos, semi-aditivos e não aditivos.

Após a definição dos esquemas lógico e conceitual, a Interface estará apta a processar consultas em linguagem natural restrita que obedeçam aos padrões sintáticos suportados. A seguir, apresentaremos dois padrões sintáticos existentes na implementação atual da Interface e exemplos de consultas que são passíveis de processamento. Os padrões serão apresentados através de uma descrição sucinta de seu objetivo e estrutura sintática. Essa estrutura é descrita através da metalinguagem que encontra-se no Apêndice A dessa dissertação. Posteriormente, são apresentados exemplos de consultas em linguagem natural que podem ser processadas e os correspondentes comandos *SQL* gerados como resultado do processamento da Interface. Ao longo dos exemplos, serão esclarecidas características e limitações da implementação atual.

4.2.1 O padrão sintático DimensionPattern

Definição do Padrão

Nome: DimensionPattern

Objetivo: Recuperação de informações a respeito de uma dimensão.

Intenção: Conjunto de nomes que representam atributos de uma dimensão seguido de restrições envolvendo quaisquer atributos da dimensão.

Estrutura sintática:

: <;><*Nome de atributo*> <,>
: $ (
: [$ (<*Nome de atributo*><;>)] | |
: [$ (<*Restrição de atributo*><;>)]
:)

Exemplos de perguntas suportadas pelo padrão DimensionPattern

P1. Qual a descrição, marca e categoria dos produtos ?

```
SELECT
     PRODUTO.DESCRICAO, PRODUTO.MARCA, PRODUTO.CATEGORIA
FROM
     PRODUTO
```

No primeiro exemplo, podemos observar a flexibilidade para a seleção de atributos de uma dimensão. O padrão não impõe restrições no número e ordem sintática para os atributos a serem selecionados.

P2. Quais as lojas e seus respectivos gerentes que encontram-se na região igual a 'Nordeste' ?

```
SELECT
     LOJA.NOME, LOJA.GERENTE
FROM
     LOJA
WHERE
     LOJA.REGIAO = 'Nordeste'
```

O segundo exemplo ilustra a utilização de restrição para os atributos de uma dimensão. A implementação específica desse padrão exige que as restrições tenham a seguinte estrutura: *<Nome de atributo> <;> <Operador> <;> <Constante>*. Segundo os resultados da pesquisa em [Ogden, 1985], restrições explícitas dessa natureza podem, facilmente, ser manipuladas por usuários casuais. Por conseqüência, ambigüidades para valores de restrições são automaticamente solucionadas.

P3. Liste os nomes das lojas, endereço, estado, cep e nome do diretor da regional ?

```
SELECT
     LOJA.NOME, LOJA.ENDERECO, LOJA.ESTADO, LOJA.CEP,
     REGIONAL.NOME_DIRETOR
FROM
     LOJA, REGIONAL
WHERE
     LOJA.REGIONAL_KEY = REGIONAL.REGIONAL_KEY
```

Nesse exemplo, o usuário, por estar visualizando um esquema conceitual multidimensional, não toma conhecimento das junções necessárias a nível lógico para que a consulta possa ser realizada.

P4. Quais os produtos que têm marca igual a 'Nestlê' e categoria igual a 'Leite' ?

```
SELECT
     PRODUTO.DESCRICAO
FROM
     PRODUTO
WHERE
     PRODUTO.MARCA='Nestlê' AND
     PRODUTO.CATEGORIA = 'LEITE'
```

No exemplo *P4*, pode-se observar o uso natural da combinação de restrições para a seleção de atributos. Contudo, a atual implementação limita a combinação de restrições ao operador AND. Nenhum processamento é realizado no sentido de identificar semânticas de conjunção ou disjunção entre as restrições.

Como resultado da flexibilidade sintática dos padrões e da abordagem baseada em palavras chaves, a interface poderá processar perguntas que não possuem semântica para o esquema conceitual apresentado:

P5. Quais os produtos que têm peso superior a 50kg ?

Como peso não é um atributo da dimensão produto, não existirá restrição reconhecida pelo sistema e a pergunta produzirá um resultado não esperado:

```
SELECT
     PRODUTO.DESCRICAO
FROM
     PRODUTO
```

4.2.2 O padrão sintático StarPattern

O padrão StarPattern representa uma extensão do padrão DimensionPattern. Com isso, todas as características do padrão DimensionPattern podem ser observadas nesse novo padrão.

Definição do Padrão

Padrão 2: StarPattern

Objetivo: Seleção simples de fatos

Intenção: Conjunto de fatos e atributos de dimensão seguidos por conjunto de restrições envolvendo atributos de dimensão ou fatos

Estrutura sintática:

: *<;>*$(*<Nome de atributo>* || *<Nome de medida>*) *<,>*
: $ (
: [$ (*<Nome de atributo><;>*)] ||
: [$ (*<Nome de medida><;>*)] ||
: [$ (*<Restrição de atributo>*<;>)] ||
: [$ (*<Restrição de medida><;>*)]
:)

Exemplo de pergunta suportada pelo padrão StarPattern

P6. Qual o total de vendas por estado, nome da loja, categoria e subcategoria para o ano igual a '1994' onde estado seja diferente de 'Sergipe' ?

```
SELECT
     TEMPO.ANO, LOJA.ESTADO, LOJA.NOME,
     PRODUTO.CATEGORIA, PRODUTO.SUBCATEGORIA,
     SUM(FACT_TABLE.VALOR_VENDIDO)
FROM
     TEMPO, LOJA, PRODUTO, FACT_TABLE
WHERE
     TEMPO.ANO='1994' AND
     LOJA.ESTADO <> 'Sergipe' AND
     TEMPO.TEMPO_KEY=FACT_TABLE.TEMPO_KEY AND
     LOJA.LOJA_KEY=FACT_TABLE.LOJA_KEY AND
     PRODUTO.PROD_KEY=FACT_TABLE.PROD_KEY
GROUP BY
     TEMPO.ANO, LOJA.ESTADO, LOJA.NOME,
     PRODUTO.CATEGORIA, PRODUTO.SUBCATEGORIA
```

No exemplo *P6*, temos a seleção de fatos através da restrição em objetos de dimensão. A partir da pergunta do usuário, pode-se observar o papel crucial de informações a respeito dos esquemas conceituais e lógicos necessárias à tradução da consulta: definição de fatos e suas descrições a nível de negócio (total de vendas caracterizando *Valor_Vendido*), especificação de fórmula de agregação e dimensões participantes.

P7. Qual o total de vendas para produtos de marca igual 'Nestlê' no ano igual a '1994' quando o total de vendas é superior a '30000' ?

```
SELECT
     PRODUTO.MARCA, TEMPO.ANO,
     SUM(FACT_TABLE.VALOR_VENDIDO)
FROM
     TEMPO, PRODUTO, FACT_TABLE
WHERE
     PRODUTO.MARCA='Nestlê' AND
     TEMPO.ANO='1994' AND
     PRODUTO.PROD_KEY=FACT_TABLE.PROD_KEY AND
     TEMPO.TEMPO_KEY=FACT_TABLE.TEMPO_KEY
GROUP BY
     PRODUTO.MARCA, TEMPO.ANO
HAVING
     SUM(FACT_TABLE.VALOR_VENDIDO) > '30.0000'
```

O exemplo *P7* demonstra a utilzição transparente de restrições em diferentes objetos: fatos e objetos de dimensão.

4.3 Arquitetura

Nesta seção apresentaremos a arquitetura de um sistema que é capaz de oferecer as características mencionadas na seção 4.1 como solução para o problema de interface a dados para usuários casuais. Embora a arquitetura não especifique como seus principais módulos devem ser implementados, a mesma pressupõe que a utilização de padrões sintáticos e do modelo multidimensional sejam efetivamente tomados como parte da solução com o objetivo de atender aos requisitos de *habitability* e transportabilidade entre domínios, principais questões relacionadas à aceitação das ILNBDs como ferramentas de acesso a dados.

Na figura 4.3, encontra-se a arquitetura do sistema proposto. O princípio básico dessa arquitetura tem sua origem nas interfaces baseadas em comparação de padrões (ver seção 2.3.1). Porém, sua abordagem para o processamento de linguagem natural, baseada em modelos de informações, supera em muito a superficialidade de tratamento de linguagem empregada pela arquitetura da figura 2.1. Os principais módulos dessa arquitetura e suas respectivas funções traduzem a compilação de características observadas nas interfaces encontradas em [Epstein, 1985], [Embley & Kimbrell, 1985], [Meng & Chu, 1999], [Watson, 1999] e [Shankar & Yung, 2000] somadas aos aspectos particulares da solução apresentada.

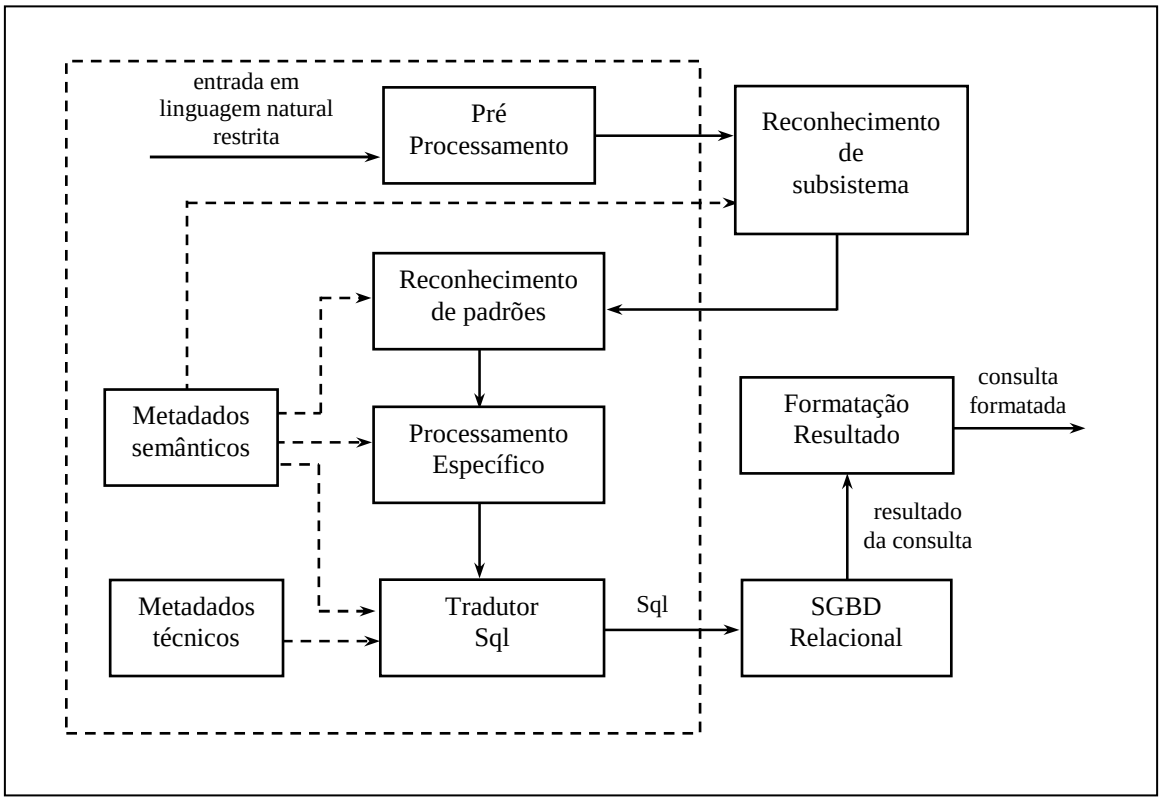


Figura 4.3: Arquitetura abstrata para interfaces em linguagem natural baseada em comparação de padrões e apoiada por metadados

A seguir, descrevemos as etapas necessárias ao processamento de consultas. Primeiramente, a consulta em linguagem natural restrita é pré-processada a fim de entrar em conformidade com os módulos subseqüentes. Comum à maioria das arquiteturas de interfaces em linguagem natural, esse módulo é responsável pelo sucesso da extração de informações. Como exemplo de processos aos quais a cadeia de caracteres que representa a pergunta de um usuário pode ser submetida temos: remoção de caracteres irrelevantes, separação de sinais de pontuação, substituição de expressões que representam operadores e datas, e identificação de constantes numéricas.

Posteriormente, a pergunta é processada pelo módulo de reconhecimento de subsistema a fim de revelar o esquema conceitual multidimensional ao qual pertence. Com isso, pode-se identificar o esquema dimensional que está diretamente relacionado com a pergunta do usuário. Esse processo, de caráter fundamental nessa arquitetura, reduz o escopo da semântica de uma pergunta através da solução de ambigüidades que podem existir entre esquemas dimensionais de uma mesma aplicação.

Reconhecido o esquema dimensional, o módulo de reconhecimento de padrões encarrega-se de escolher o melhor padrão sintático que atenda aos requisitos da pergunta. Nesse momento, vale ressaltar que a escolha do número e características dos padrões sintáticos irá influenciar diretamente o uso e sucesso da interface. A adoção de um pequeno número de padrões que possam ser facilmente reconhecidos e utilizados contribuirá para a *habitability* do sistema. Cada padrão sintático deve possuir um processamento específico que será utilizado para extrair da pergunta original objetos necessários à tradução da consulta. Esse módulo, o Processamento Específico, é responsável por validar a consulta original, resolver ambigüidades, com ou sem a ajuda do usuário, e prover informações suficientes e necessárias ao módulo de tradução.

O módulo denominado Tradutor tem a responsabilidade de construir uma expressão na linguagem *SQL* a partir das informações geradas pelo processamento específico. Finalizada essa atividade, a consulta em *SQL* é submetida ao SGBD relacional para que seja efetivamente realizada. Após o retorno do resultado, o módulo Tradutor o encaminha para o módulo de Formatação de Resultado, daí a explicação para o fluxo de informação que existe entre o SGBD e o módulo de Formatação. A formatação do resultado da consulta é uma tarefa de extrema relevância para o usuário final, pois o simples resultado de uma consulta *SQL* pode não traduzir, com clareza, as expectativas de uma consulta. Muitas interfaces em linguagem natural utilizam módulos semelhantes para a geração de resultados em linguagem natural.

Na arquitetura da figura 4.3, pode-se observar que o processamento como um todo é dependente de informações providas por dois módulos: Metadados Técnicos e Metadados Semânticos. Esses módulos constituem as principais fontes de informações necessárias ao reconhecimento de subsistemas, reconhecimento de padrões, processamento específico e tradução. Daí, a afirmação que essa arquitetura é apoiada por metadados.

Metadados são informações descritivas sobre a estrutura e significado dos dados, das aplicações e dos processos que manipulam os dados [Coalition, 1999] [Sherman, 1997]. Em SGBDs relacionais, os metadados constituem a descrição dos objetos definidos no banco de dados, ou seja, a definição de tabelas, colunas, visões, integridade referencial. Em outras palavras, representam o conteúdo de um catálogo. A natureza estática das aplicações do ambiente operacional, principalmente aquelas voltadas para o usuário final, exige apenas um conjunto limitado de metadados, geralmente manipulados por desenvolvedores, necessários

às atividades de acesso a dados. Entretanto, a complexidade do ambiente dos sistemas de suporte à decisão fez surgir um interesse particular na manutenção e gerenciamento de metadados [Sachdeva, 1999][Stöhr, *et al*.,1999]. Nesse contexto, os metadados não estão restritos ao conteúdo de catálogos, mas descrevem todo um ambiente onde os dados são extraídos, transformados e manipulados. As interfaces de acesso a SSDs, ao contrário dos sistemas encontrados no ambiente operacional, necessitam de um rico conjunto de metadados que possa descrever conceitos dos modelos conceituais utilizados e suas dependências com elementos de esquemas lógicos. A composição desses metadados é imprescindível à atividade de exploração de informação através de interfaces OLAP e ferramentas de consulta *ad-hoc*.

Um modelo de metadados ou modelo de informação é uma descrição dos tipos de objetos de dados que podem ser utilizados por uma aplicação e os relacionamentos que existem entre esses tipos de objetos. Os dois módulos de nossa arquitetura, Metadados Técnicos e Metadados Semânticos, contêm, respectivamente, modelos de informações que descrevem esquemas lógicos e esquemas conceituais para uma aplicação particular. Um outro modelo de informação existente na arquitetura, embora não explícito na figura 4.3, é o modelo que especifica as dependências entre os esquemas conceituais e esquemas lógicos. A arquitetura como um todo utiliza esses modelos de informações como forma de apresentar os dados ao usuário final e como modelo de domínio [Copestake & Jones, 1989] para o processamento de linguagem natural. Logo, pode-se inferir que as características finais de uma interface que seja baseada nessa arquitetura serão, principalmente, determinadas pelos modelos de informações implementados.

4.4 Proposta de projeto e implementação

Nesta seção apresentaremos a especificação do desenvolvimento de uma infra-estrutura de software básica para a arquitetura da figura 4.3. Essa infra-estrutrua, descrita através das fases de análise, projeto e implementação, possui dois principais objetivos: a) confirmar nossas expectativas a respeito da arquitetura proposta; b) direcionar futuras implementações para aspectos relevantes à arquitetura. De fato, não é objetivo desta dissertação prover uma interface para o usuário final, mas sim apresentar uma nova arquitetura, confirmar sua validade e prover uma infra-estrutura que possa servir de ponto de partida para interfaces robustas o suficiente que possam ser testadas e utilizadas por usuários casuais. Logo, não apresentaremos um projeto que contemple todos os módulos da arquitetura proposta, mas restringiremos nossa abordagem aos módulos que acreditamos

serem os mais relevantes para os objetivos a serem alcançados. Na figura 4.3, a área pontilhada delimita os módulos que serão contemplados nesta seção.

Utilizaremos a linguagem UML (*Unified Modeling Language*) [Booch *et al*., 1999] [Fowler & Scott, 1999] como mecanismo de representação para especificar e documentar as diversas fases do desenvolvimento desse projeto. Não adotamos nenhum processo de desenvolvimento particular, mas seguimos uma abordagem iterativa e incremental. Ao longo do desenvolvimento, houve a preocupação de sincronizar o projeto e a implementação a fim de produzir uma documentação coesa com o código gerado. A linguagem de programação Java [Flanagan, 1997] foi escolhida para a fase de implementação em função de sua simplicidade e crescente aceitação, características que tornam o código fonte em mais um artefato de documentação.

4.4.1 Análise de Requisitos

Durante a atividade de pesquisa bibliográfica desta dissertação, particularmente durante a leitura de artigos sobre trabalhos relacionados à construção de interfaces em linguagem natural, observamos muito pouca preocupação na definição de requisitos funcionais para os sistemas apresentados. Em sua maioria, os autores preocuparam-se apenas em ressaltar as vantagens de uma interface em linguagem natural, tema já bastante conhecido e de pouca contribuição para o desenvolvimento de sistemas. Em resumo, não existe uma preocupação formal com a finalidade do sistema para o usuário final ou para desenvolvedores responsáveis por manter e estender a aplicação. Essa atitude, além de dificultar o entendimento dos verdadeiros propósitos do sistema, impede a realização de testes funcionais. Por isso, resolvemos adotar uma postura diferenciada e especificar, através do uso de cenários, os principais requisitos de nossa interface.

Podemos identificar três principais atores com relação ao sistema: a) o usuário casual, b) o administrador da aplicação, c) o programador de padrões. O último não interage diretamente com o sistema, mas é responsável por estender a funcionalidade da aplicação. Cada um desses atores possui cenários particulares de interação com a interface que serão detalhados a seguir.

A.1 O Usuário Casual

O ator Usuário Casual representa todos os *novos usuários* dos sistemas de suporte à decisão baseados na *Web*. Proprietário de um perfil técnico incapaz de lidar com as atuais interfaces de acesso a SSDs, recorre a uma abordagem mais simples, baseada em linguagem natural restrita, que apresenta as informações de maneira mais natural e intuitiva para as atividades de consulta.

U.1 Navegação no Modelo Conceitual

A primeira atividade de interação realizada pelo Usuário Casual é a exploração do esquema conceitual de sua aplicação. Com o auxílio de algum mecanismo, seja ele gráfico ou textual, ao usuário devem ser apresentados os principais conceitos relativos à elaboração de consultas: os fatos e os objetos que definem os fatos. Dessa forma, o mesmo estará apto a utilizar as diversas variáveis como forma de seleção ou restrição em suas consultas.

U.2 Elaboração de Consultas

A atividade de elaboração de consulta é precedida pela apresentação dos padrões sintáticos que podem ser utilizados como forma de interação com o sistema. Os padrões sintáticos devem expressar, com relativa simplicidade, seus objetivos e suas restrições. De posse dessas informações, o usuário pode submeter perguntas à interface e esperar por uma resposta ou indicação de erro na elaboração da pergunta.

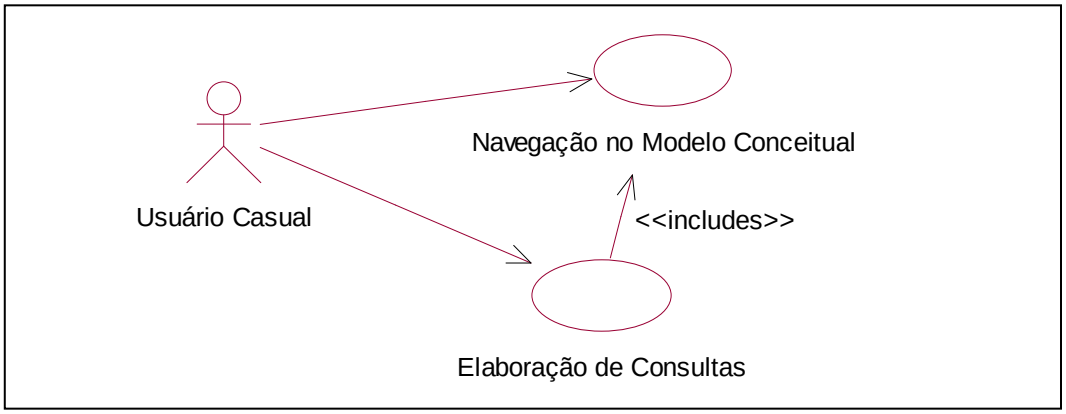


Figura 4.4: Cenários de interação para o Usuário Casual

A.2 O Administrador

O ator Administrador representa todos os indivíduos responsáveis pela configuração da interface. Essa configuração engloba as tarefas de manipulação de metadados técnicos e metadados semânticos. Essas atividades precedem a utilização da interface pelo ator Usuário Casual.

U.3 Definição de Metadados Técnicos

O ator Administrador é responsável por definir e manter o conjunto de metadados técnicos pertencentes a uma aplicação. Os metadados técnicos representam as informações relativas a um esquema relacional: esquema, tabelas, visões, atributos, restrições de integridade e relacionamentos.

U.4 Criação do Universo para o Usuário Casual

O ator Administrador também é responsável pela definição do Universo para o Usuário Casual. Nesse contexto, o termo Universo representa o conjunto de metadados semânticos que definem o esquema conceitual multidimensional utilizado por uma aplicação. Fatos, dimensões, objetos de dimensões e hierarquias de dimensões são exemplos de metadados semânticos que devem ser definidos e manipulados pelo ator Administrador.

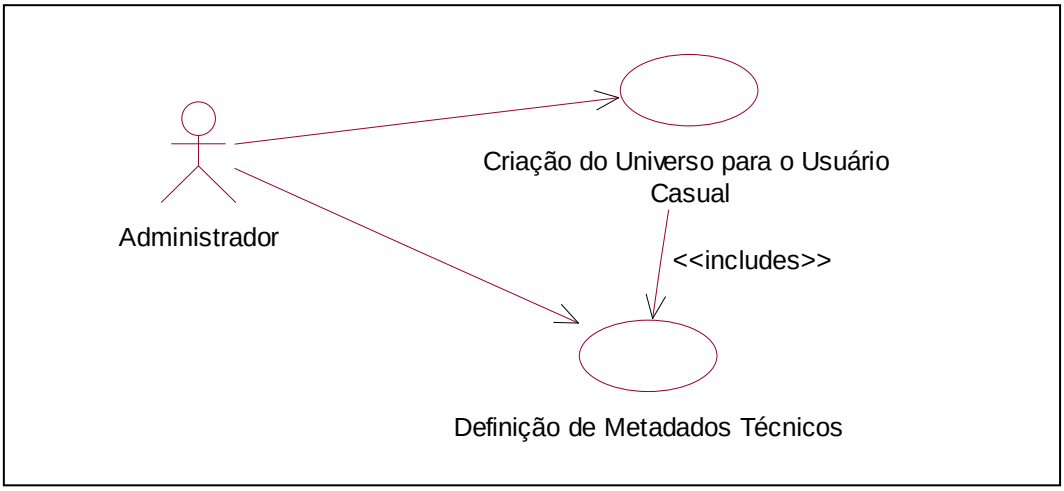


Figura 4.5: Cenários de interação para o Administrador

A.3 O Programador de Padrões

O ator Programador de Padrões representa os desenvolvedores responsáveis por estender a funcionalidade da interface através da implementação de novos padrões sintáticos.

U.5 Definição de novos padrões sintáticos

A atividade de definição de novos padrões sintáticos, principal requisito não-funcional da nossa interface, engloba uma série de tarefas necessárias à sua execução: a) análise dos novos requisitos junto aos usuários finais; b) definição e teste dos padrões sintáticos junto aos usuários finais, c) implementação do processamento específico para o novo padrão a ser definido. Embora essa atividade não represente uma interação com o sistema, nem mesmo um requisito funcional, optamos por seguir a notação de *Use Cases* numa tentativa de padronizar a documentação para os requisitos globais da aplicação.

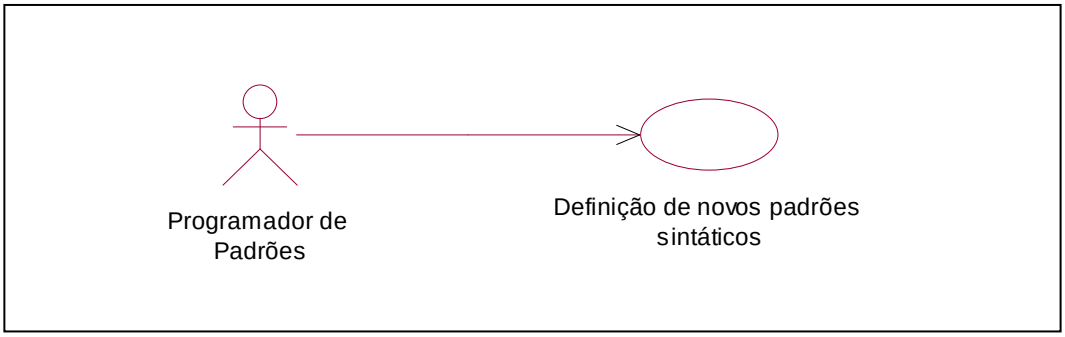


Figura 4.6: Cenário de interação para o Programador de Padrões

4.4.2 Visão Arquitetural

A solução para o desenvolvimento de sistemas que apresentam um certo grau de complexidade é projetar a partir de abstrações. O verdadeiro significado da palavra abstração é: "o processo mental de separar um ou mais elementos de uma totalidade complexa" [Ferreira, 1993]. Esse termo tem sido utilizado extensivamente para definir atividades relativas ao desenvolvimento de sistemas de software. Contudo, as novas definições herdam a principal idéia desse vocábulo: "A abstração consiste na concentração dos aspectos essenciais de uma entidade e em ignorar suas propriedades acidentais" [Rumbaugh *et al.*, 1991]; "A abstração é o ato de determinar a estrutura essencial de um sistema, ignorando seus detalhes funcionais" [Cantor, 1998]. De modo geral, a abstração representa o mecanismo utilizado para visualizar a estrutura de um sistema através de pequenos blocos funcionais, utilizados

como meio de decompor um grande problema em pequenas partes passíveis de entendimento e solução.

O mecanismo descrito acima é utilizado na concepção da visão arquitetural de um sistema. Essa visão, também conhecida como visão abstrata, é composta pelos diversos módulos funcionais que compõem um sistema, e seus relacionamentos. Em um projeto orientado a objetos que utilize a linguagem UML pode-se especificar a visão arquitetural através de diagramas de pacotes. Utilizaremos esse artefato com o objetivo de modularizar a nossa interface e prover uma visão mais clara dos relacionamentos existentes entre suas partes constituintes e os módulos da arquitetura abstrata que se pretende implementar. Na figura 4.7, pode-se observar o diagrama de pacotes como a visão abstrata do nosso sistema. A seguir, faremos uma descrição sucinta de cada pacote, definindo sua principal funcionalidade e seu papel na arquitetura da figura 4.3.

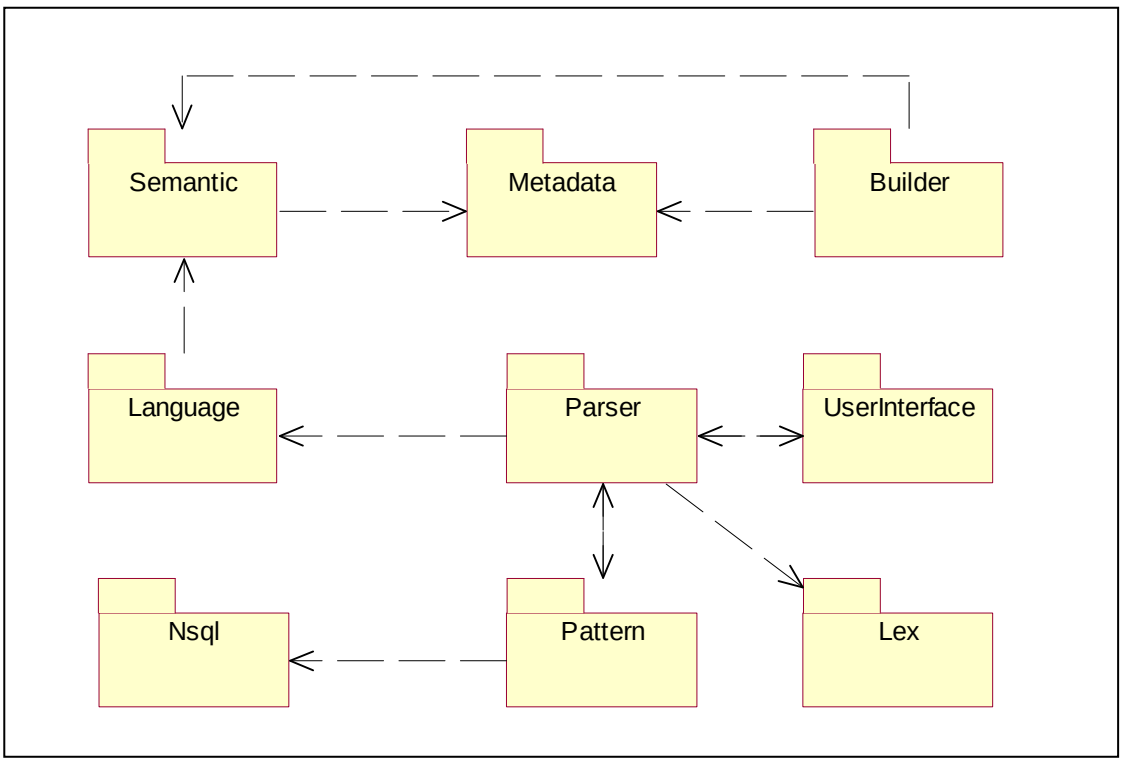


Figura 4.7: Visão arquitetural da interface

P.1 Metadata

O pacote Metadata é formado pelas classes que compõem o modelo de informação técnico. Seu principal objetivo é fornecer uma interface de acesso a informações contidas em

catálogos de SGBDs relacionais. Representa o módulo Metadados Técnicos da arquitetura abstrata.

P.2 Semantic

O pacote Semantic engloba as classes que formam o modelo de informação semântico. Seu principal objetivo é fornecer uma visão conceitual multidimensional de esquemas relacionais através da modelagem de conceitos como fatos, dimensões e hierarquias. Representa o módulo Metadados Semânticos da arquitetura abstrata.

P.3 Builder

O pacote Builder contém classes responsáveis pela definição automática de elementos do pacote Semantic e Metadata.

P.4 Language

O pacote Language não possui uma correspondência na arquitetura abstrata, pois representa um conjunto de classes pertinentes à nossa solução particular para o processamento de linguagem natural. Suas responsabilidades serão melhor compreendidas a partir do momento em que os outros pacotes forem sendo mais detalhadamente especificados.

P.5 Lex

O pacote Lex provê uma interface comum para informações contidas nos pacotes Metadata e Semantic, além de servir como um repositório para informações que independem de uma aplicação particular.

P.6 Parser

O pacote Parser representa o núcleo de processamento da interface. Dentre suas principais responsabilidades estão: pré-processamento da pergunta original em linguagem natural, análise da pergunta pré-processada e reconhecimento de padrões sintáticos. Esse pacote realiza as atividades dos seguintes módulos na arquitetura abstrata: Pré-Processamento e Reconhecimento de Padrões.

P.7 Pattern

O pacote Pattern contém as classes que representam os padrões sintáticos suportados pelo sistema. A responsabilidade desse pacote é fornecer os padrões sintáticos e seus respectivos processamentos específicos. Embora seja o representante direto do módulo Processamento Específico na arquitetura da figura 4.3, suas responsabilidades abrangem questões relativas aos módulos Reconhecimento de Padrões e Tradutor.

P.8 Nsql

O pacote Nsql fornece uma visão orientada a objetos dos principais componentes da linguagem *SQL* relacionados com consultas. Seu objetivo é fornecer um modelo genérico de consulta *SQL* que possa ser customizado pelo processamento específico dos padrões sintáticos. Em função de sua funcionalidade, é caracterizado como integrante do módulo Tradutor da arquitetura abstrata.

P.9 UserInterface

O pacote UserInterface contém as classes que representam a aplicação. Seu principal objetivo é servir de interface entre o usuário final e as classes do pacote Parser. No atual estágio de desenvolvimento do projeto, esse pacote contém classes responsáveis pela instanciação dos outros pacotes e serve, exclusivamente, para testes de integração. Por esse motivo, o mesmo não será contemplado com detalhes de projeto e implementação.

4.4.3 Análise, Projeto e Implementação

Deste ponto em diante, apresentaremos os detalhes do desenvolvimento de nossa proposta de projeto e implementação. Tentaremos ao máximo seguir uma ordem de dependência na explanação dos módulos. Os pacotes e os respectivos diagramas que serão detalhados representam modelos que se encontram entre as fases de análise e implementação. Vale ressaltar que omitiremos a maioria das interfaces utilizadas com o objetivo de facilitar o entendimento dos diagramas utilizados.

4.4.3.1 Pacote Metadata

O pacote Metadata representa a camada de software necessária para a tradução de consultas realizadas em um esquema conceitual e que necessitam ser transformadas para um

esquema lógico correspondente a fim de serem efetivamente processadas. Quando o processamento de consultas é realizado por um SGBD relacional, nosso caso particular, essa camada deve conter informações técnicas a respeito da estrutura dos objetos que compõe um esquema relacional. O conjunto dessas informações descritivas recebe o nome de metadados ou, mais precisamente, metadados técnicos. Para uma perfeita compreensão e manipulação de metadados a partir de uma aplicação faz-se necessário a definição de um modelo de metadados ou modelo de informação que descreva formalmente os tipos de metadados e seus relacionamentos. Além disso, deve-se prover uma API (*Application Program Interface*) para que os mesmos possam ser acessados.

A globalização das corporações em um ambiente de negócios altamente competitivo exige um fluxo de informações rápido e eficiente. Para isso, é necessário a integração de informações e seu conseqüente gerenciamento como forma de diminuir o custo e tempo despendido na implementação de novas soluções. Contudo, esse gerenciamento integrado encontra obstáculos na proliferação de ferramentas de gerenciamento e manipulação de dados que, em sua maioria, processam metadados de maneira proprietária. A solução para esse problema apresenta-se na forma de padrões para definição e troca de metadados, o que permite a reutilização de dados entre diferentes aplicações. Podemos citar dois principais padrões que possuem esse objetivo: MDIS (*Metadata Interchange Specification*) [Coalition, 1998] e OIM (*Open Information Model*) [Coalition, 1999] ambos pertencentes à Metadata Coalition, um consórcio não-lucrativo de vendedores e usuários finais que tem como objetivo prover uma especificação não proprietária dos metadados corporativos. O MDIS provê conceitos para suportar a troca de metadados relacionados com diferentes tipos de repositórios de dados: relacionais, multidimensionais, em rede, hierárquicos e baseados em arquivos. Contudo, sua especificação restringe-se aos relacionamentos existentes em um esquema de dados. Em contrapartida, o OIM apresenta modelos de informações que englobam todas as fases do desenvolvimento de sistemas de informação. Um desses modelos é o modelo de banco de dados, que fornece conceitos relativos a provedores de dados relacionais.

Dado a relevância da discussão anterior, seria sensato que o leitor esperasse a utilização do OIM como modelo de informação técnico para nossa aplicação. Porém, a generalidade de sua especificação traz consigo uma complexidade desnecessária para os requisitos do projeto. Logo, embora tenhamos utilizado alguns conceitos presentes no OIM,

descartamos sua generalidade na busca de um modelo centrado na nossa solução. A discussão que segue apresentará nosso modelo de informação para metadados técnicos e, ao mesmo tempo, revelará a API necessária a sua utilização.

Por questões de espaço e simplicidade de explicação, o pacote Metadata será divido em quatro diagramas: Elementos Básicos, Junções Conceituais, Gerenciamento de Junções, Carga de Metadados. Essa separação não limitará o entendimento do pacote como um todo uma vez que cada diagrama apresentará todos os elementos externos[4], embora com uma notação simplificada, que participem em associações.

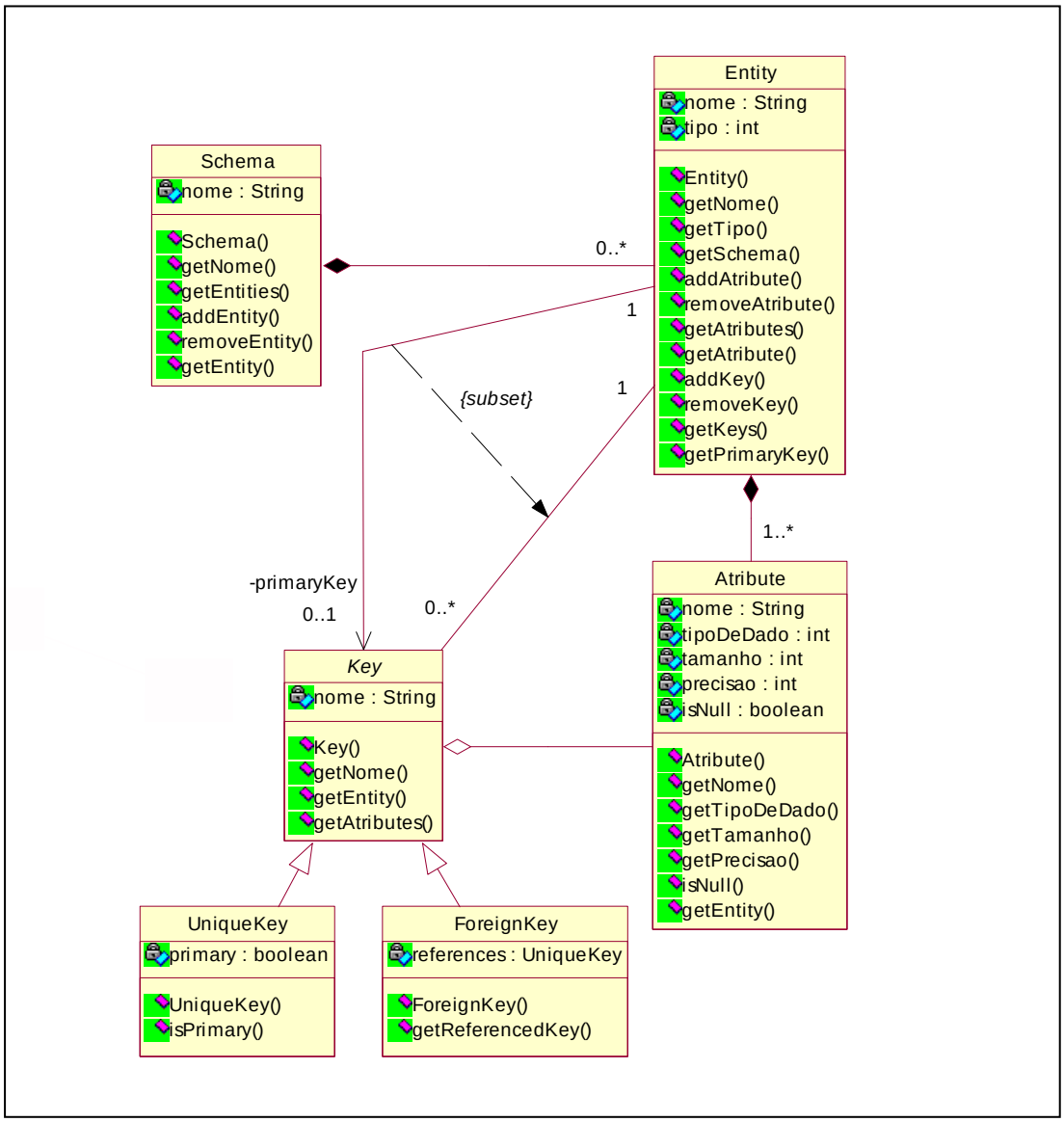


Figura 4.8: Diagrama Elementos Básicos do pacote Metadata

[4] Elementos representados em outro diagrama.

A classe Schema é a ponte de acesso ao conjunto de metadados técnicos. Uma instância dessa classe descreve um esquema relacional. Logo, é utilizada para organizar logicamente outros descritores de objetos encontrados em um banco de dados. Embora todos os descritores de objetos desse modelo (tabelas, atributos, restrições de integridade) estejam contidos em um Schema, a existência de uma hierarquia de agregação limita os objetos diretamente relacionados a essa classe. No modelo de informação da figura 4.8 pode-se observar, por exemplo, que os elementos Atribute e Key estão indiretamente ligados à classe Schema e só poderão ser manipulados através da navegação em seus atributos. Nos próximos diagramas desse pacote, veremos que a semântica de uma instância dessa classe suplanta a de um esquema relacional. Essa característica é reflexo da necessidade de informações de cunho conceitual diretamente relacionadas com os descritores técnicos.

Uma instância da classe Entity descreve a estrutura de dados básica em qualquer banco de dados relacional: a tabela. Objetos que indiretamente representam uma tabela, visões e sinônimos, também podem ser descritos por uma instância dessa classe. No modelo de informação, uma instância dessa classe só é significativa quando associada a uma instância da classe Schema.

Instâncias da classe Entity são descritas por instâncias da classe Atribute e da classe *Key*. Instâncias da classe Atribute representam colunas de uma tabela relacional. Os atributos que caracterizam instâncias dessa classe possuem semântica dependente do atributo *tipoDeDado* que, em nossa implementação, pode conter um dos seguintes valores: VARCHAR, DATE, DATE_TIME e NUMBER. O valor VARCHAR indica que a instância representa um atributo cujo tipo de dado é qualquer cadeia de caracteres, seja de tamanho fixo ou variável. O valores DATE e DATE_TIME especificam tipos de dados que descrevem datas do calendário Gregoriano. NUMBER representa o tipo de dado utilizado por atributos que possuem valores numéricos. Para os últimos, o atributo *precisao* da classe Atribute possui semântica bem definida.

Cada instância da classe *Key* descreve uma coleção ordenada de colunas em uma tabela ou visão, ou seja, representa um descritor para o objeto chave em um banco de dados relacional. Logo, é uma agregação de objetos do tipo Atribute. Uma *Key*, por ser uma classe abstrata, não possui instâncias diretas, mas delega suas responsabilidades para seus subtipos: UniqueKey e ForeignKey. O primeiro tipo descreve chaves primárias ou chaves candidatas, diferenciadas a partir do atributo *primary*. As instâncias do segundo tipo descrevem uma

coleção ordenada de colunas que podem ser utilizadas para referenciar uma outra instância da classe Key desde que seja do tipo UniqueKey.

As associações presentes no diagrama Elementos Básicos refletem as restrições impostas pelo modelo de informação. Dentre elas, podemos citar a composição entre instâncias da classe Schema e instâncias da classe Entity, assim como a composição entre Entity e Atribute. O relacionamento com a restrição {*subset*} indica que instâncias da classe Entity possuem como atributo uma instância da classe *Key* que está contida na associação entre Entity e *Key*.

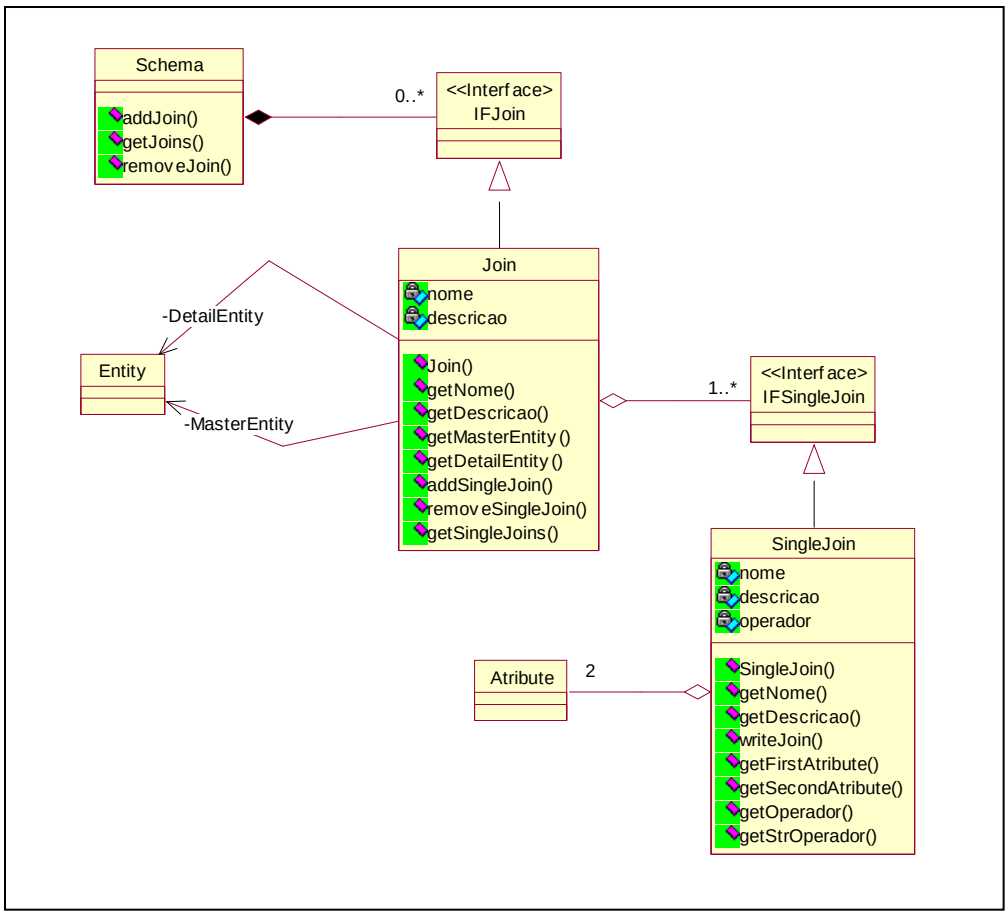


Figura 4.9: Diagrama Junções Conceituais do pacote Metadata

O diagrama Junções Conceituais (figura 4.9) é a base de restrições para o esquema conceitual. A principal responsabilidade de suas classes é prover informações a respeito dos relacionamentos lógicos existentes entre elementos descritos por instâncias da classe Entity. As duas classes principais desse diagrama são SingleJoin e Join. Instâncias da classe SingleJoin especificam a unidade básica de uma junção, ou seja, uma expressão entre

atributos de uma tabela. A agregação de instâncias da classe SingleJoin dá origem a uma instância da classe Join, uma junção propriamente dita. Ao contrário da especificação do pacote banco de dados no OIM, instâncias da nossa classe Join não se limitam a apresentar relacionamentos envolvendo restrições de integridade. Como resultado da definição desses novos elementos, a classe Schema passa o oferecer novas operações como definidas no diagrama da figura 4.9.

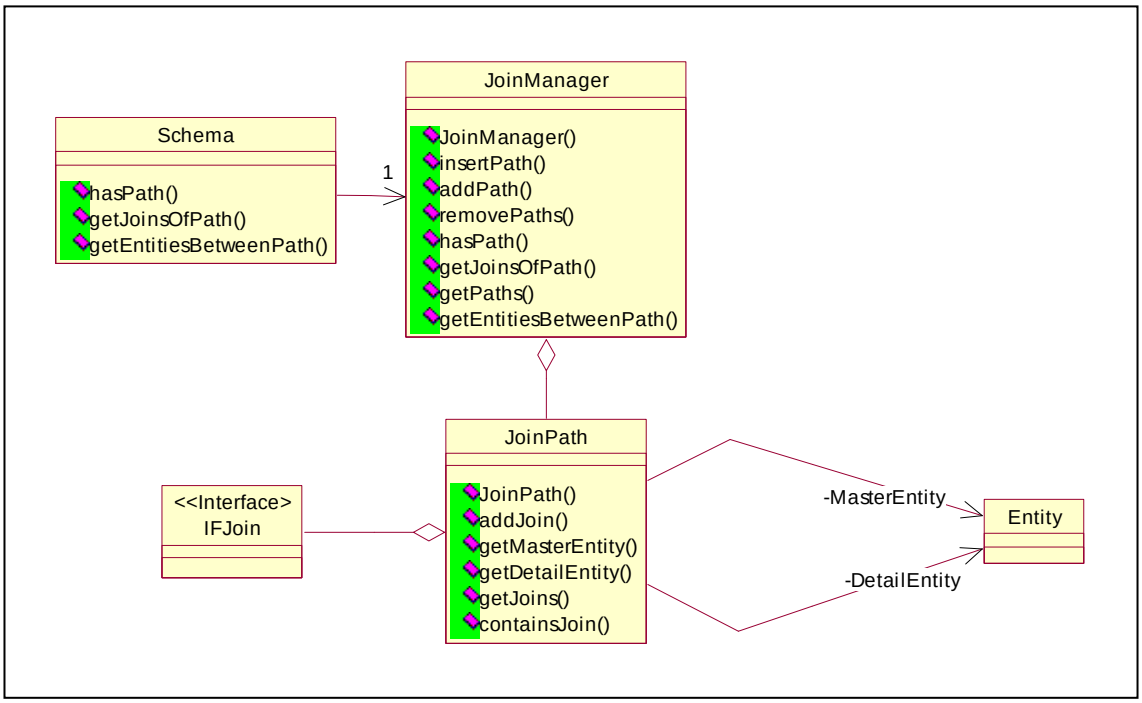


Figura 4.10: Diagrama Gerenciamento de Junções do pacote Metadata

Embora uma instância da classe Schema contivesse instâncias da classe Join, muito pouca informação poderia ser extraída através de sua API. Ao longo do desenvolvimento dos demais pacotes, surgiu a necessidade de obter informações a respeito dos relacionamentos indiretos entre instâncias da classe Entity. Logo, precisaríamos de uma funcionalidade adicional na classe Schema. Porém, acreditávamos que essa responsabilidade deveria pertencer a outras classes. Decidimos criar um novo diagrama (figura 4.10) com novos elementos e delegar as novas responsabilidades para os mesmos. A classe JoinPath generaliza o conceito de junção para relacionamentos diretos ou indiretos.

Uma instância da classe JoinManager é responsável pela API necessária à manipulação de junções indiretas. Durante a inserção de instâncias da classe Join, instâncias

do tipo JoinPath são automaticamente criadas: uma para representar o último objeto do tipo Join inserido no Schema e outras representando as junções indiretas que são derivadas a partir da inserção do novo objeto JoinPath. Assim como no último diagrama, a figura 4.10 adiciona operações à API da classe Schema.

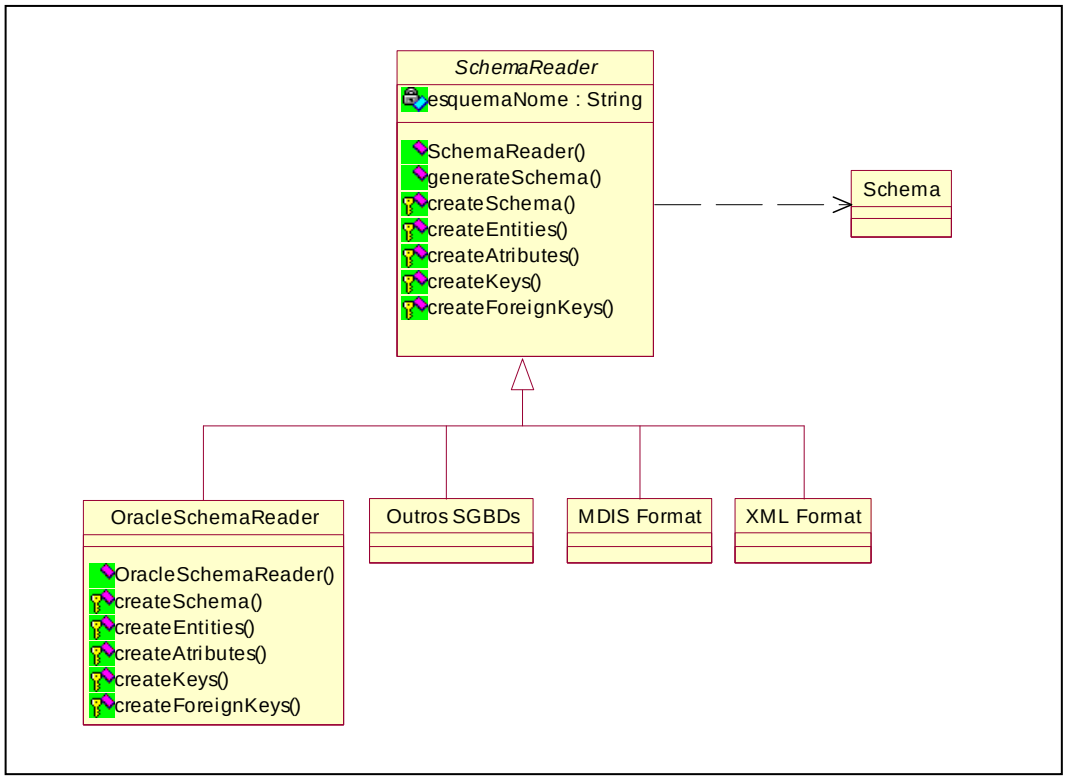


Figura 4.11: Diagrama Carga de Metadados do pacote Metadata

O último diagrama do pacote Metadata, Carga de Metadatos, tem como responsabilidade a importação de metadados provenientes de diversos meios e formatos. A classe *SchemaReader* representa uma interface comum a ser utilizada pela aplicação como provedor de metadados. Na atual iteração, foi implementada uma classe concreta para extrair os metadados técnicos necessários a partir do católogo do SGBD Oracle8.

Chegamos ao final da especificação do pacote Metadata. Como foi visto, esse pacote fornece o modelo de informação técnico necessário à arquitetura da aplicação. Além disso, o mesmo provê uma API para manipulação do conjunto de metadados técnicos e uma API adicional para manipulação de objetos de caráter conceitual, junções. Ao final do projeto e implementação, podemos afirmar que esse pacote, embora não aborde o modelo de

informação de forma genérica, pode ser facilmente reutilizado por outros projetos que apresentem necessidade de acesso e manipulação de metadados dessa natureza.

4.4.3.2 Pacote Semantic

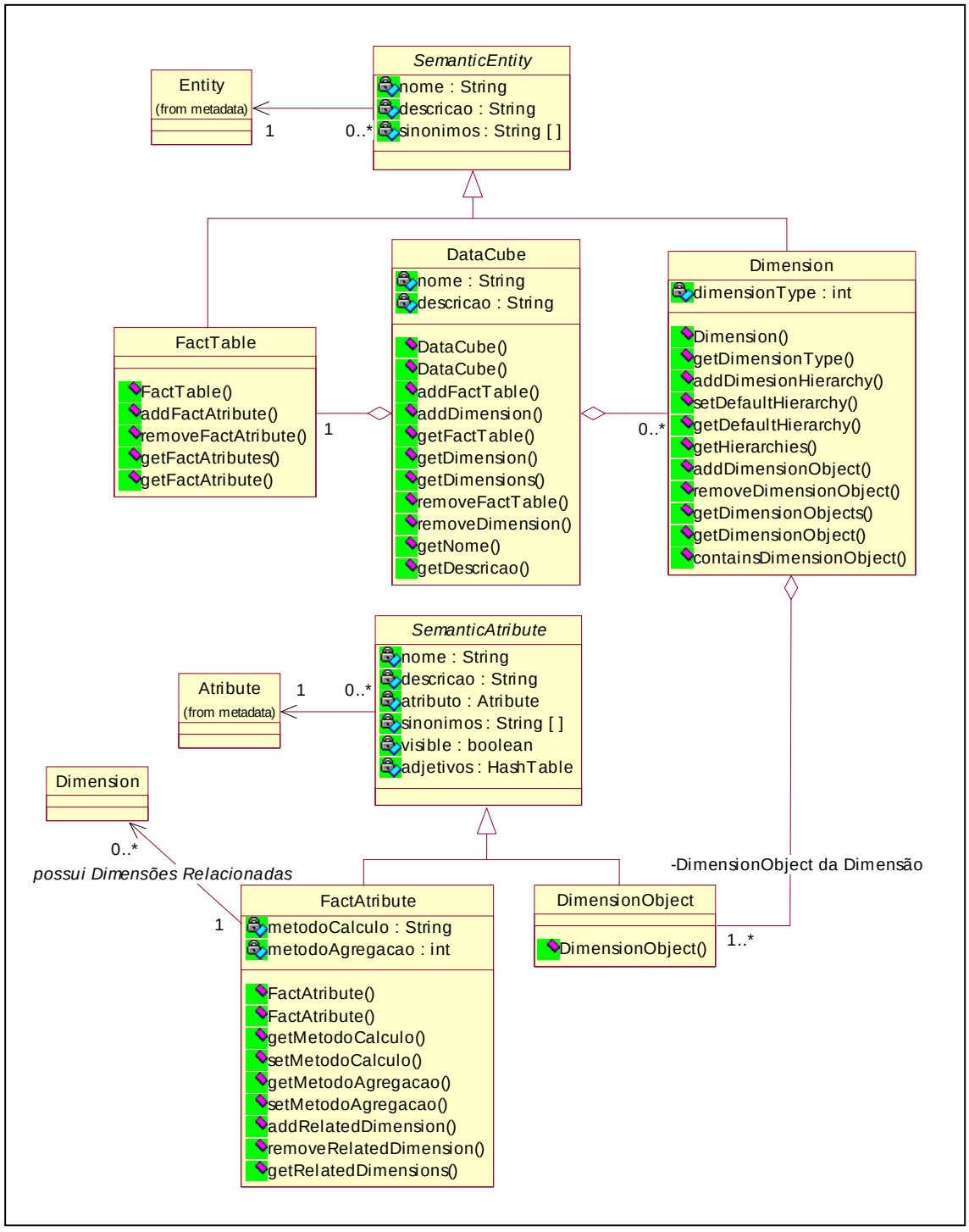


Figura 4.12: Diagrama Elementos Básicos do pacote Semantic

O pacote Semantic representa a camada de software da aplicação responsável por modelar uma visão conceitual multidimensional dos dados para o usuário final. Dentre suas principais funções, podemos destacar: a) apresentar um modelo de informação semântico para a interface; b) revelar as dependências existentes com metadados do modelo de informação técnico.

O modelo de informação semântico descrito através do diagrama de classes do pacote Semantic está diretamente relacionado aos conceitos da técnica de modelagem dimensional. Logo, observar-se-á descritores para elementos como tabela de fatos, fatos, dimensões, atributos de dimensões, hierarquias de dimensões e níveis de hierarquias. Além desses conceitos, o modelo de informação semântico inclui elementos que descrevem a visão de negócios para as entidades de uma determinada aplicação: descrição, sinônimos, adjetivos, relacionamentos possíveis. Tais atributos, que fazem parte de um cojunto maior de dados denominado metados do negócio, são cruciais para a aceitação e recuperação eficiente de informações em um *Data Warehouse* [Lehmann & Jaszewski, 1999]. A definição do conjunto de metadados para esse modelo de informação pressupõe a existência de um conjunto de metadados técnico previamente definido e do qual é dependente. Mais uma vez, em função da falta de espaço necessário, o diagrama de classes desse pacote será decomposto em três partes: Elementos Básicos, Hierarquias e Dimensão Tempo.

A primeira parte do pacote (figura 4.12), Elementos Básicos, introduz as principais classes do modelo de informação. O ponto de acesso para o conjunto de metadados semânticos é a classe DataCube, cujas instâncias são responsáveis pela manipulação direta e indireta dos outros objetos pertencentes ao modelo. Uma instância do tipo DataCube é descrita por dois principais elementos: FactTable e Dimension, descritores para uma tabela de fatos e tabelas de dimensões, respectivamente. Essas duas classes são especializações de uma classe superior, SemanticEntity, que provê uma estrutura comum para elementos conceituais derivados da classe Entity (ver figura 4.8). De modo semelhante, as classes FactAtribute e DimensionObject a serem detalhadas posteriormente, são especializações da classe SemanticAtribute, responsável por prover uma estrutura comum para elementos conceituais derivados da classe Atribute.

Instâncias da classe FactTable descrevem as tabelas de fatos em um esquema dimensional. Sua principal responsabilidade é manipular instâncias do tipo FactAtribute que descrevem as medidas encontradas em uma tabela de fatos. No diagrama da figura 4.12, uma

associação envolvendo a classe FactAtribute merece destaque. A associação *possuiDimensõesRelacionadas* é responsável pela caracterização de fatos como perfeitamente aditivos, semi-aditivos e não-aditivos.

Dimensões de um esquema dimensional são descritas por instâncias da classe Dimension, a qual representa uma agregação de DimensionObjects, descritores para os atributos de uma tabela de dimensão. Como visto anteriormente, cada instância da classe Dimension possui um relacionamento direto com uma única instância da classe Entity. Entretanto, instâncias do tipo DimensionObject relacionadas com uma instância do tipo Dimension não precisam, necessariamente, ser derivadas da mesma instância do tipo Entity. Porém, faz-se necessário a existência de junções diretas ou indiretas entre a instância do tipo Entity à qual o DimensionObject está relacionado e a instância do tipo Entity da qual a instância do tipo Dimension é derivada. Essa restrição limita o esquema dimensional utilizado em um esquema Estrela simples ou um esquema *Snowflake*.

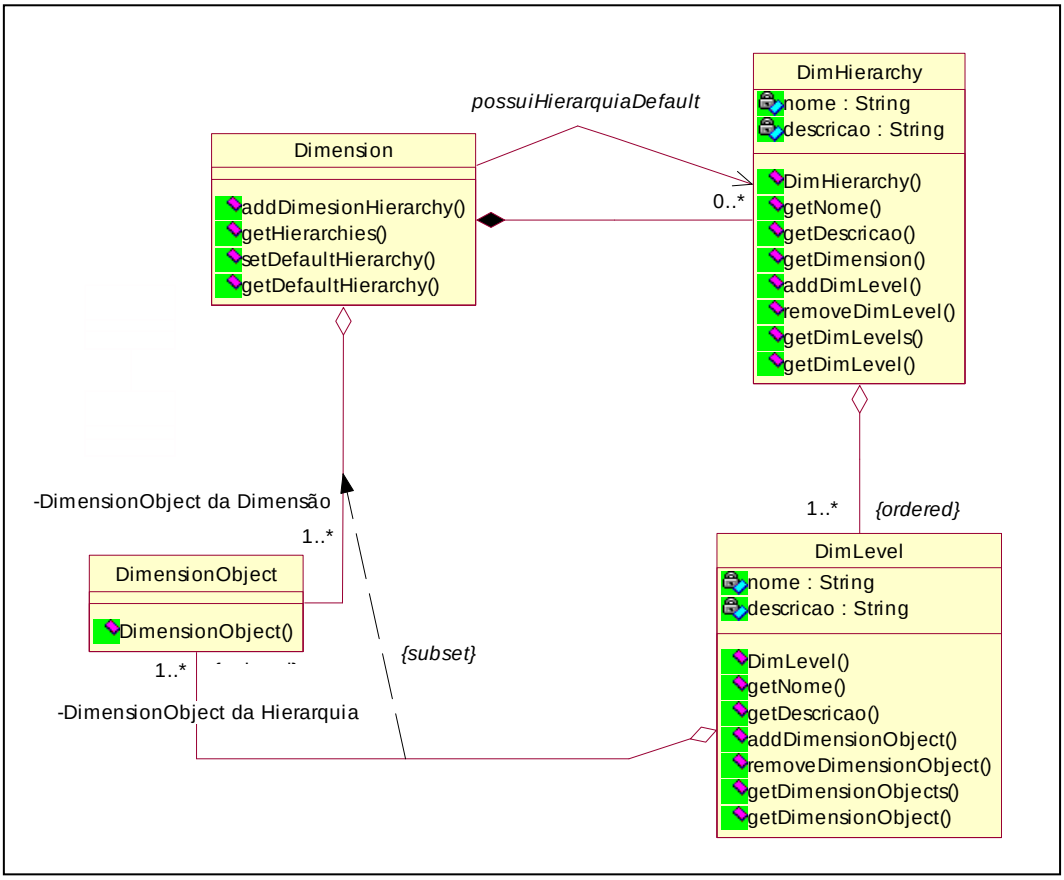
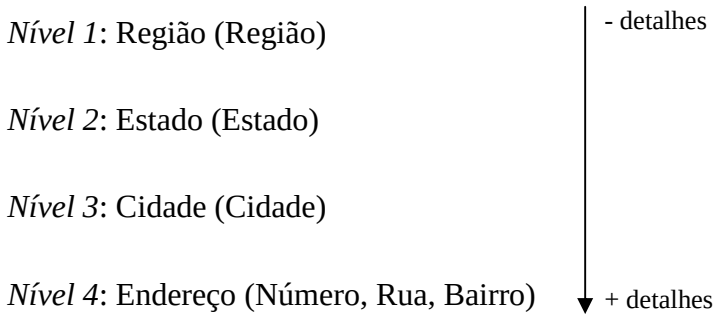


Figura 4.13: Diagrama Hierarquias do pacote Semantic

A segunda parte do pacote (figura 4.13) explora o conceito de hierarquias em uma dimensão (ver capítulo 3). Instâncias da classe DimHierarchy, definidas por instâncias da classe DimLevel, representam as hierarquias que podem estar presentes em uma tabela de dimensão de um esquema dimensional. Hierarquias são compostas de níveis, representados no nosso modelo por instâncias da classe DimLevel. Cada instância da classe DimLevel pode conter qualquer número de atributos de uma dimensão, DimensionObject, que representam informações a respeito do nível que os contêm. Para exemplificar, considere uma dimensão Cliente contendo os seguintes atributos: Número, Rua, Bairro, Cidade, Estado e Região. Poderíamos construir uma hierarquia de localidade a partir dos seguintes níveis:



Logo, teríamos uma instância da classe DimHierarchy formada por quatro instâncias da classe DimLevel: Região, Estado, Cidade e Endereço. Os níveis Região, Estado e Cidade são determinados por um único atributo de dimensão. Por consequência, instâncias que representam esses níveis serão compostas por um único objeto de dimensão. O nível Endereço compreende o conjunto de informações encontradas nos atributos Número, Rua e Bairro. Nesse caso, a instância da classe DimLevel será uma agregação dos DimensionObjects representantes desse atributos.

Ao contrário da agregação existente entre a classe Dimension e a classe DimensionObject, a associação entre DimHierarchy e DimLevel impõe uma restrição de ordenamento, {*ordered*}, responsável por modelar os níveis de detalhes existentes em uma hierarquia. No diagrama, também pode ser observado a restrição de subconjunto {*subset*} que especifica a relação de subconjunto de instâncias da classe DimensionObject pertecentes à associação com a classe DimLevel. Essa restrição é necessária para garantir que o conjunto de objetos de um nível de uma hierarquia de uma dimensão seja um subconjunto dos objetos da dimensão.

A última parte do pacote Semantic (figura 4.14) modela a existência de uma dimensão tempo explícita na qual os atributos são descritos por instâncias da classe TimeObject, especialização de DimensionObject. A diferença básica entre um DimensionObject e um TimeObject é o relacionamento existente com a classe TimeComponent. A idéia geral desse modelo é permitir que uma dimensão tempo possa ser definida a partir de elementos pré-definidos (TimeComponent). Com isso, pode-se criar um vocabulário comum para a dimensão tempo, que poderá ser reutilizado em diferentes aplicações. Todavia, essa característica só poderá ser alcançada se os valore reais para atributos da dimensão tempo forem mapeados para valores das instâncias de TimeComponent.

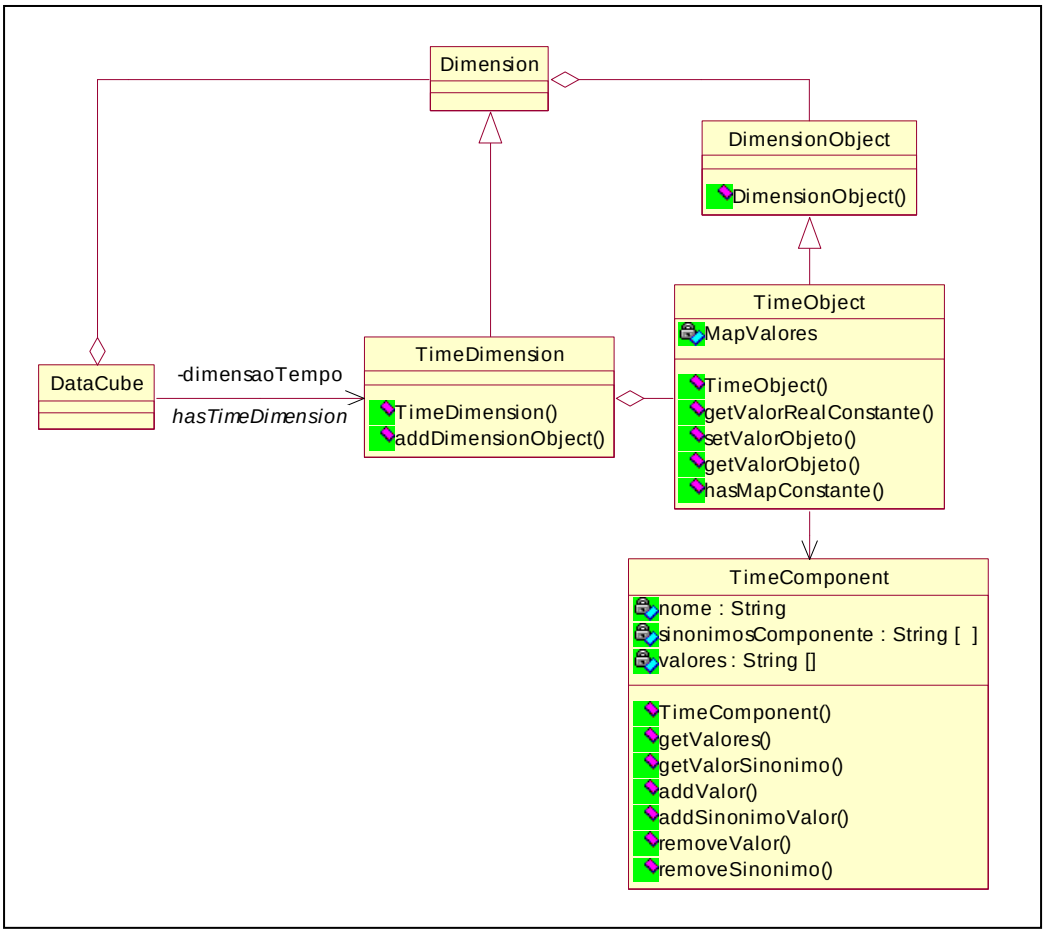


Figura 4.14: Diagrama Dimensão Tempo do pacote Semantic

Após a apresentação do pacote Semantic como modelo de informação semântico para a aplicação, o leitor pode, e com toda razão, questionar a ausência do tratamento de

agregados[5] [Kimball, 1996] [Kimball *et al*., 1998]. Justificamos essa ausência a partir da afirmação que performance na execução de consultas não representa, no momento, um requisito não-funcional para a Interface. Contudo, se tal requisito for imprescindível, o módulo de navegação em agregados [Kimball, 1996b] pode utilizar a saída do módulo Tradutor sem grandes impactos no projeto.

4.4.3.3 Pacote Lex

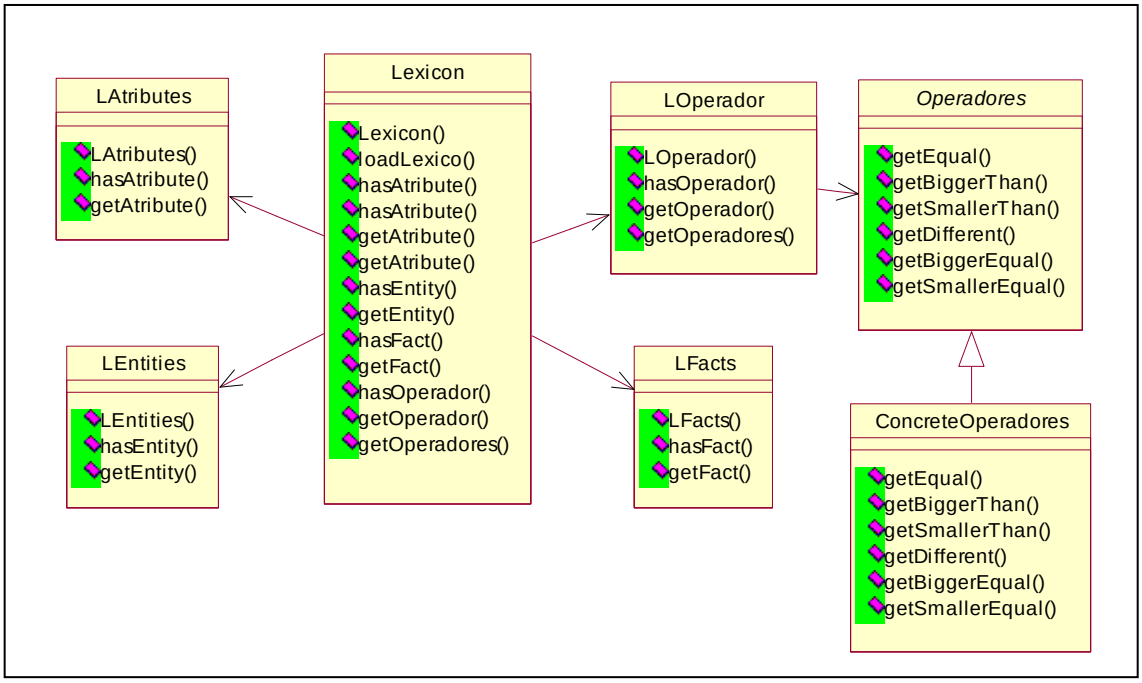


Figura 4.15: Diagrama de Classes do pacote Lex

Durante o processamento de linguagem natural, é necessário verificar a existência de relações entre nomes contidos na pergunta em linguagem natural e os objetos definidos no esquema conceitual. Realizar uma iteração em todos os objetos do esquema conceitual mostrou-se uma atividade dispendiosa. Nesse caso, a performance requerida diz respeito ao processamento de liguagem natural. Por isso, sentimos a necessidade de prover uma estrutura indexada para sinônimos dos objetos conceituais. Desse modo, foram definidas estruturas

[5] Agregados são tabelas de fatos pré-computados derivados de uma tabela de fatos com menor nível de granularidade. A principal motivação para a construção de agregados é a otimização de consultas. Contudo, agregados devem ser transparentes para o usuário final, que continuará elaborando suas perguntas em relação à tabela de fatos primária. Logo, faz-se necessário a utilização de um programa especial, o navegador em agregados, responsável por redirecionar uma consulta destinada à tabela de fatos primária para um agregado que possa atendê-la.

particulares para cada elemento do esquema conceitual: tabela de fatos, dimensões, fatos e atributos de dimensões. Contudo, resolvemos criar uma interface única para essas estruturas, a classe Lexicon, a fim de oferecer um único ponto de acesso para as informações indexadas. Uma outra responsabilidade desse pacote é servir de repositório para informações que independem de um esquema particular. No atual estágio de desenvolvimento da Interface, uma instância da classe LOperador é responsável por especificar os diferentes tipos de operadores, juntamente com seus sinônimos, que podem ser utilizados por usuários na formulação de perguntas.

4.4.3.4 Pacote Language

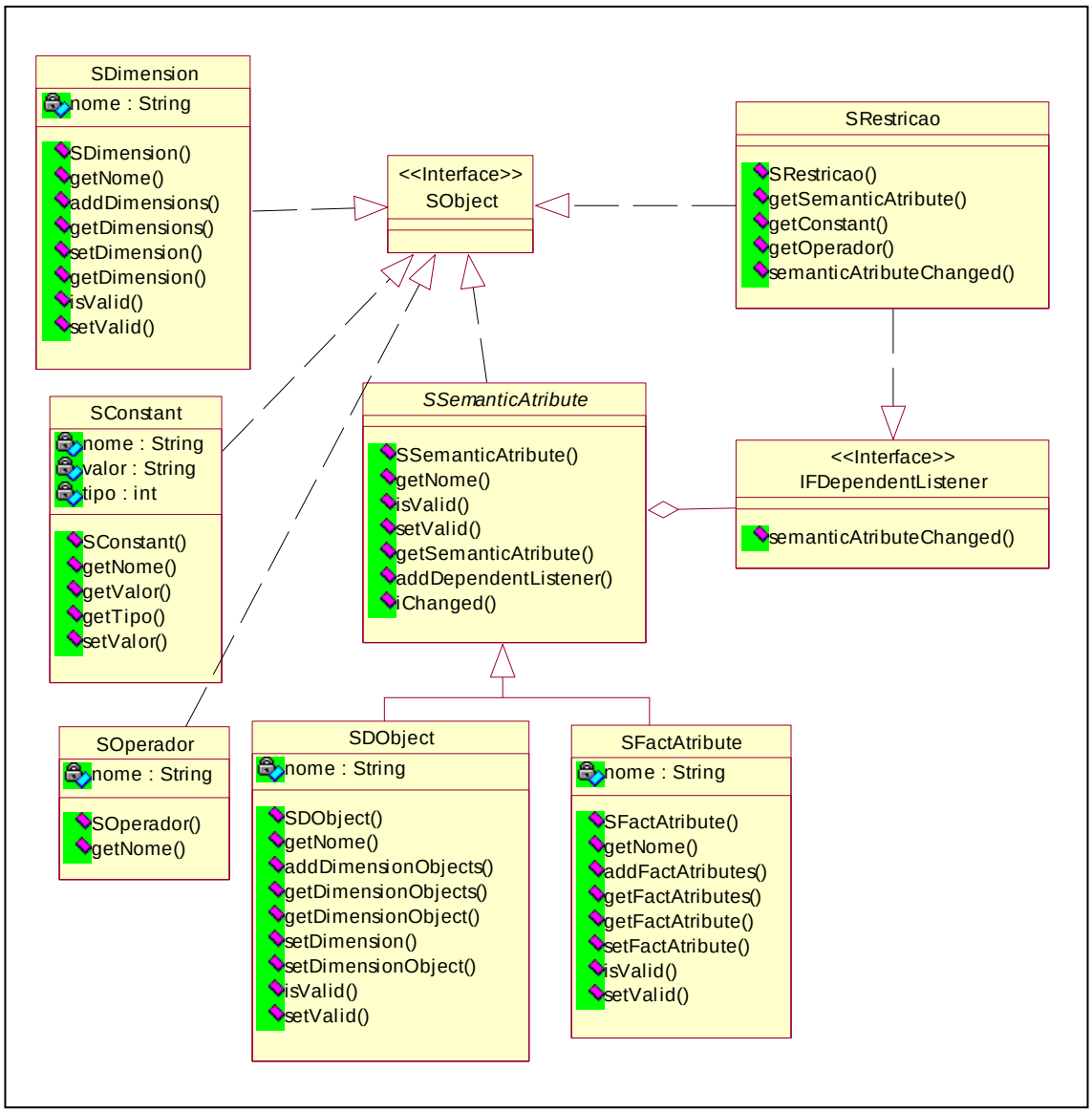


Figura 4.16: Diagrama de Classes do pacote Language

O pacote Language (figura 4.16) não possui uma correspondência na arquitetura abstrata da figura 4.3. Compreende um conjunto de classes pertinentes à nossa solução particular para o processamento de linguagem natural. Todas as instâncias de classes do pacote Language que implementam a interface SObject são reconhecidas como objetos semânticos. No contexto da nossa interface, objeto semântico é todo elemento resultante da primeira etapa do processamento de linguagem natural, ou seja, são os elementos que são fornecidos aos padrões sintáticos para a execução do processamento específico. Objetos semânticos podem ser de dois tipos: primitivos e derivados. Objetos semânticos primitivos são aqueles construídos a partir do conteúdo da pergunta original em linguagem natural. Objetos semânticos derivados são construídos a partir de objetos primitivos. Ao analisarmos o pacote Parser, veremos que existem classes especializadas na instanciação dos objetos primitivos e derivados definidos no pacote Language.

A implementação atual do pacote Language especifica os seguintes tipos de objetos primitivos: SConstant, SOperador, SDimension, SDObject e SFactAtribute. SConstant representa uma constante encontrada na pergunta em linguagem natural. Em nossa implementação, todas as constantes da pergunta do usuário devem ser delimitadas por apóstrofos. Essa é uma restrição comum em interfaces que utilizam a abordagem de linguagem natural restrita. Além disso, essa prática, que acreditamos não trazer grandes dificuldades para o usuário final, elimina a questão de ambigüidade para valores de atributos. Instâncias da classe SOperador descrevem os operadores existentes na pergunta do usuário. Uma instância da classe SDimension indica a presença de um nome que possa representar um ou mais objetos do tipo Dimension. Ao ser instanciado, um objeto do tipo SDimension conterá um conjunto de referências para todos os objetos do tipo Dimension que o mesmo possa representar. Instâncias da classe SDObject, que representam instâncias do tipo DimensionObject, e da classe SFactAtribute, relacionadas com objetos do tipo FactAtribute, comportam-se de maneira semelhante à classe SDimension e dispensam maiores detalhes.

A partir da discussão anterior, pode-se inferir que os objetos semânticos resultantes da primeira etapa do processamento de linguagem natural são ambíguos por natureza e, caso possuam mais de uma representação, serão marcados como inválidos. Caberá ao

processamento específico dos padrões sintáticos validar os objetos semânticos necessários à realização das consultas antes de passá-los para o módulo de tradução[6].

SRestricao é um objeto semântico derivado formado a partir de instâncias das classes SSemanticAtribute, SOperador e SConstant. Como conseqüência da composição de um objeto do tipo SSemanticAtribute, objetos do tipo SRestricao apresentam os mesmos problemas de resolução de ambigüidade. Porém, através da implementação do padrão *Observer*[7] [Gamma *et.al*, 1994] asseguramos que as mudanças de estado ocorridas no objeto primitivo do tipo SSemanticAtribute sejam propagadas para os objetos cadastrados como *Listeners*[8]. Logo, as questões de ambigüidade para objetos do tipo SRestricao serão resolvidas no momento em que as ambigüidades de seus constituintes forem solucionadas.

4.4.3.5 Pacote Parser

O pacote Parser (figura 4.17) representa o núcleo do processamento de linguagem natural. Dentre suas principais responsabilidades, estão: pré-processamento da pergunta original em linguagem natural, análise da pergunta pré-processada e reconhecimento de padrões sintáticos .

NLInterface representa o coordenador para o processamento de linguagem natural. Essa classe contém os padrões sintáticos utilizados pelo sistema. A instância única dessa classe é responsável por receber a consulta em linguagem natural, delegar a primeira etapa do processamento para uma instância da classe NLEngine, a qual retornará uma estrutura contendo objetos semânticos, selecionar o padrão sintático que melhor se relaciona com a pergunta do usuário e executar seu respectivo processamento específico. Para escolher o melhor padrão sintático, o objeto NLInterface compara as propriedades de cada padrão sintático com as propriedades da estrutura retornada pelo objeto NLEngine.

---

[6] A nossa implementação não possui um pacote ou módulo específico para tradução. O termo aqui utilizado diz respeito às instâncias de classes que cooperam para produção de uma consulta na linguagem *SQL* a partir de objetos semânticos validados pelo processamento específico dos padrões utilizados. Dentre essas classes, está o próprio padrão sintático.

[7] *Observer* representa um dos padrões de projeto de comportamento definidos em [Gamma *et al*., 1994].

[8] *Listener* representa um componente do padrão *Observer*.

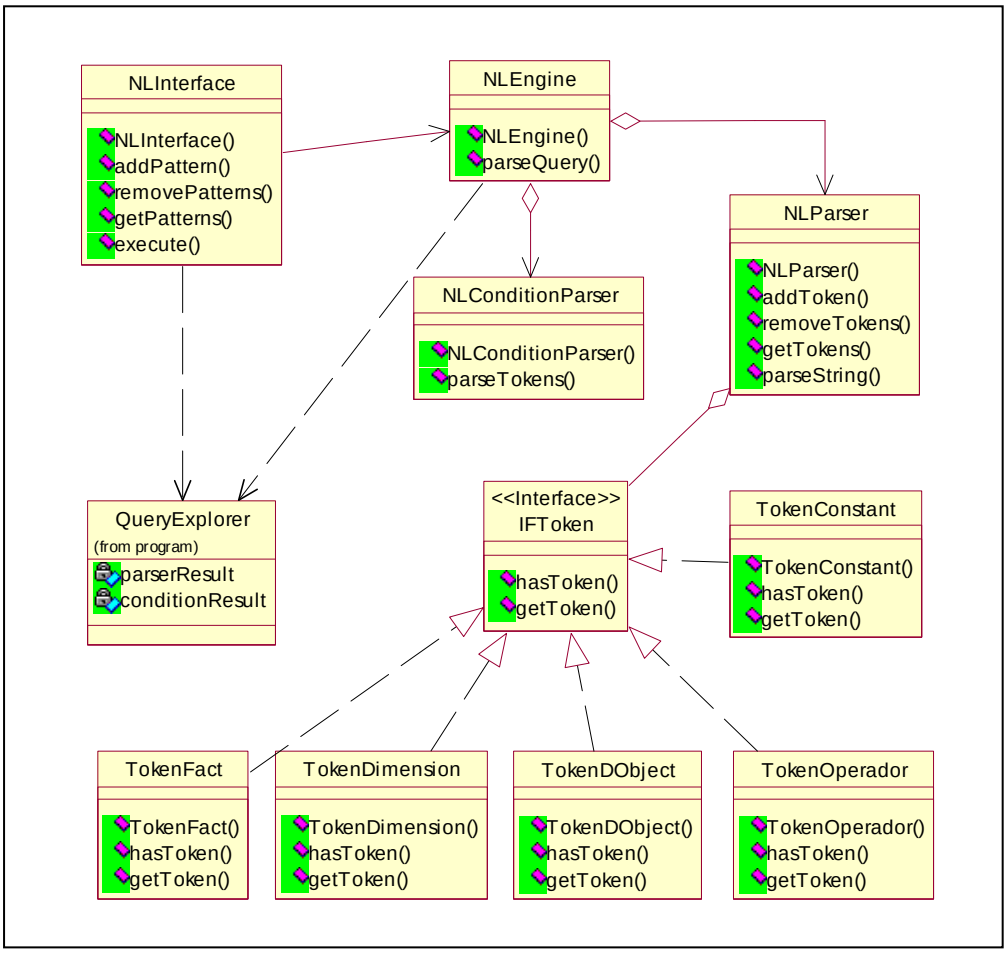


Figura 4.17: Diagrama de Classes do pacote Parser

NLEngine é responsável pelo pré-processamento da pergunta original e pelo povoamento da estrutura QueryExplorer com objetos semânticos primitivos e derivados. Essa estrutura será passada para uma instância da classe NLInterface, que se encarregará de repassá-la para o padrão sintático que melhor atenda à consulta do usuário. A implementação presente do NLEngine possui duas classes responsáveis pela instanciação de objetos semânticos: NLParser e NLCondition.

A classe NLParser representa um extrator genérico de objetos semânticos primitivos. O algoritmo utilizado pelo NLParser extrai subconjuntos de palavras da pergunta em linguagem natural e executa iterações nos objetos do tipo IFToken a fim de identificar objetos semânticos. Logo, todo objeto semântico primitivo que precise ser identificado na pergunta original deve possuir uma classe que implemente a interface IFToken. No diagrama da figura 4.17 podemos identificar as seguintes classes responsáveis pela instanciação de objetos primitivos: TokenFact, TokenDimension, TokenDObject, TokenOperador e TokenConstant.

A classe NLCondition é um extrator de objetos semânticos derivados. A entrada para seu processamento é a saída da execução do NLParser. Sua implementação atual extrai do resultado do NLParser objetos do tipo SRestricao.

4.4.3.6 Pacote Pattern

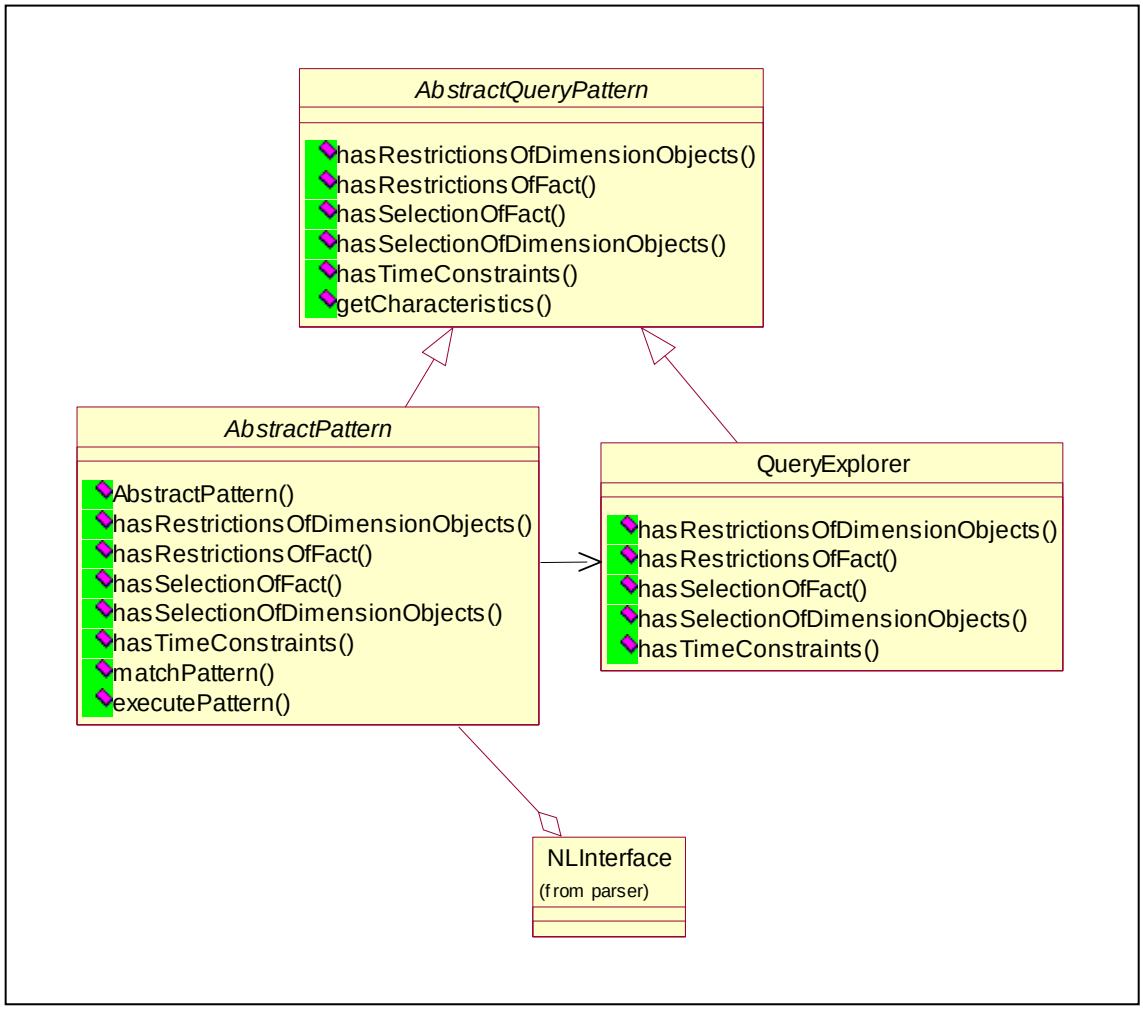


Figura 4.18: Diagrama de Classes do pacote Pattern

O pacote Pattern é responsável pela implementação dos padrões sintáticos utilizados pelo sistema. No diagrama da figura 4.18 apresentamos apenas a interface que deve ser implementada por classes que representam padrões sintáticos concretos. A classe abstrata *AbstractPattern* provê, basicamente, as operações que serão utilizadas por uma instância da classse NLInterface durante o processamento de linguagem natural.

A classe AbstractQueryPattern serve unicamente como mecanismo para garantir que os padrões sintáticos e uma instância da classe QueryExplorer possam ser comparados

utilizando a mesma interface. A conseqüência do uso desse padrão de projeto é a garantia de que as classes implementarão todas as operações necessárias.

4.4.3.7 Pacote Builder

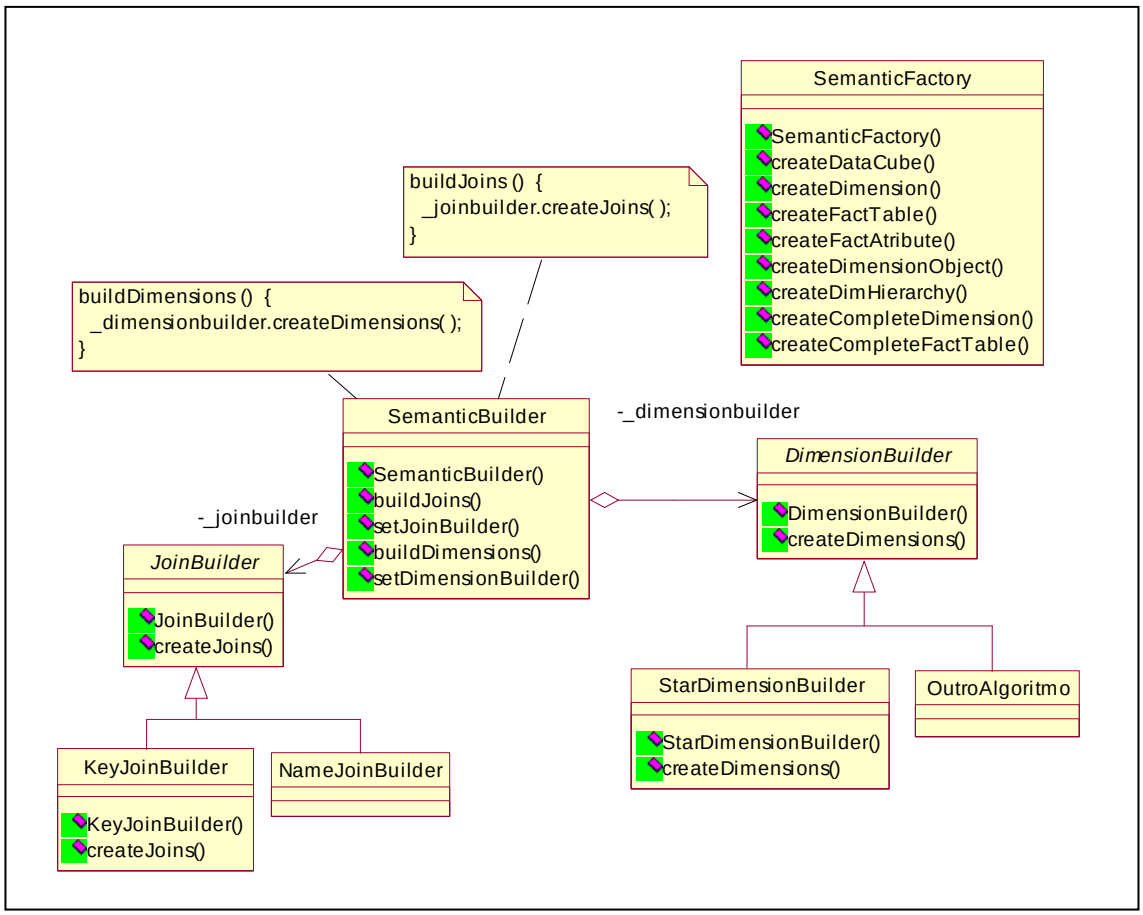


Figura 4.19: Diagrama de Classes do pacote Builder

O pacote Builder (figura 4.19) é responsável por prover mecanismos automáticos para a definição de metadados para os modelos de informação técnico e semântico. Sua utilidade é melhor percebida quando analisamos o cenário de interação do ator Administrador. Dentre suas atividades, estão a definição dos conjuntos de metadados técnicos e semânticos necessários à utilização da interface. Logo, seria apreciado que o mesmo pudesse criar objetos do tipo Dimension e instâncias da classe DimensionObject correspondentes a partir de instâncias da classe Entity e Atribute, respectivamente, de maneira automática, sem a necessidade de definir manualmente um mapeamento para todos os atributos. Outra utilidade seria a definição automática de junções a partir de restrições de integridade encontradas no conjunto de metadados técnicos. Essas características podem ser encontradas na maioria das interfaces de acesso a dados para SSDs. Além disso, as classes desse pacote foram de

extrema importância para a realização de testes durante o desenvolvimento do projeto, visto que ainda não dispomos de uma ferramenta administrativa para definição de metadados.

4.4.3.8 Pacote Nsql

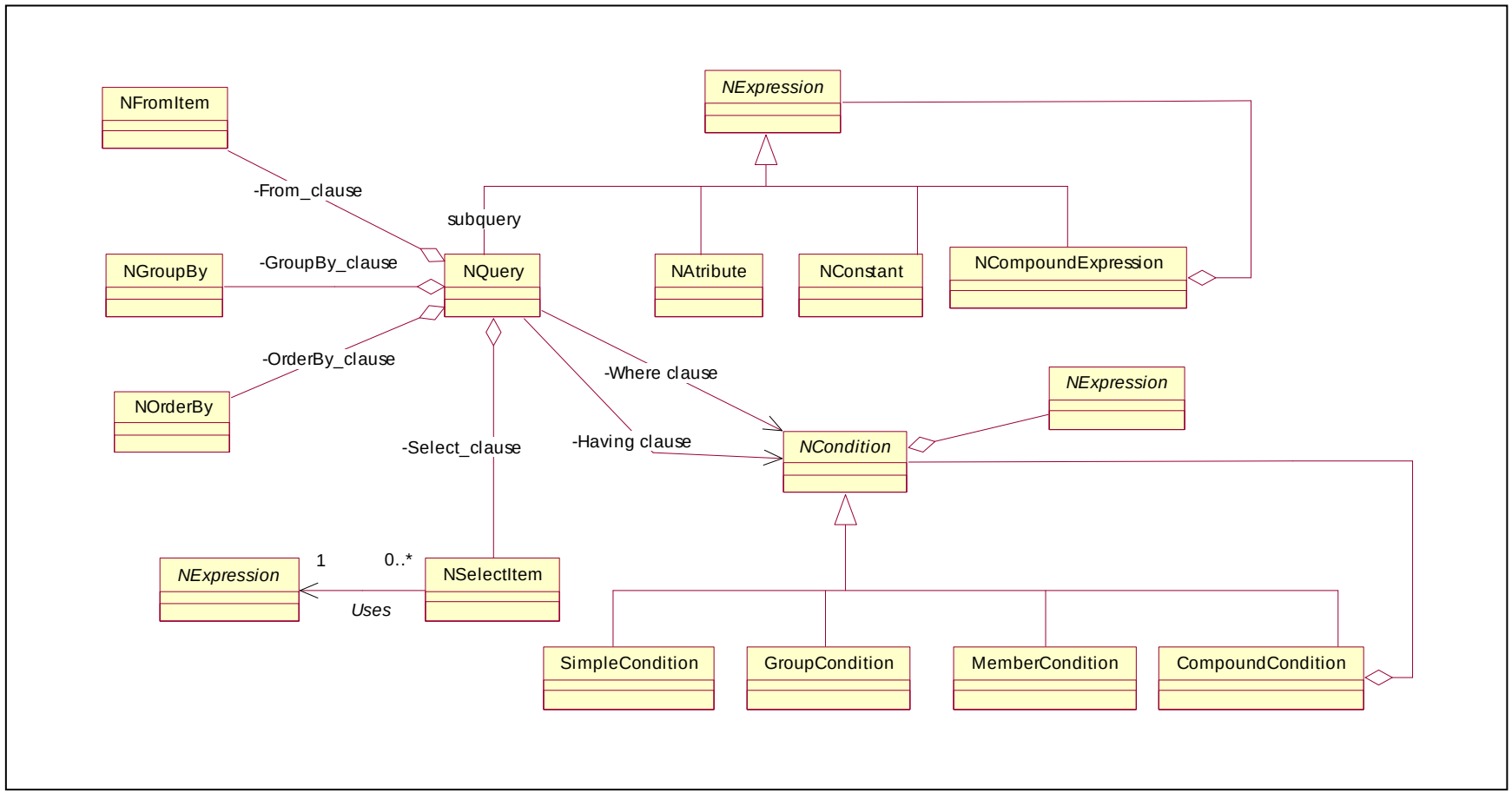


Figura 4.20: Diagrama de Classes do pacote Nsql

O pacote Nsql (figura 4.20) fornece uma visão orientada a objetos dos principais componentes da linguagem *SQL* relacionados com consultas. O diagrama da figura 4.20 não traduz a especificação do padrão *SQL*, mas apresenta estreita relação como a extensão *SQL* apresentada pelo SGBD Oracle 8. Um dos seus objetivos é fornecer um modelo genérico de consulta *SQL* que possa ser customizado pelo processamento específico dos padrões sintáticos. Outro objetivo importante desse pacote é subtrair a responsabilidade do programador de padrões de formular corretamente uma expressão na linguagem *SQL* sem erros de sintaxe. A manipulação de objetos que saibam como escrever sua representação elimina, por completo, erros cometidos pelo programador.

Em função do escasso espaço disponível, o diagrama da figura 4.20 não fornece todos os atributos e operações das classes constituintes. Além disso, o fato de não estar diretamente relacionado com a solução apresentada não nos motivou a explorá-lo com mais detalhes.

4.5 Conclusões

Neste capítulo, apresentamos nossa proposta de arquitetura de interface como solução para o problema de acesso a dados em SSDs baseados na *Web* para usuários não-especialistas e casuais. A fim de validar a arquitetura abstrata definida e criar uma infra-estrutura para futuras implementações, especificamos um projeto e implementação de uma interface que contempla os principais módulos da arquitetura.

Capítulo 5

Extensão da Interface e Resultados Práticos

O capítulo 4 foi destinado à infra-estrutura de projeto e implementação para nossa solução. Entretanto, não mencionamos as principais características que ditam o comportamento da interface: os padrões sintáticos. Neste capítulo, apresentamos o projeto e implementação de dois padrões sintáticos como forma de exemplificar o mecanismo de extensão do projeto do sistema. Posteriormente, comparamos nossa solução como outras interfaces que apresentam arquiteturas semelhantes. Concluímos o capítulo tecendo comentários a respeito do projeto como um todo.

5.1 Criação de padrões sintáticos

O primeiro passo na criação de padrões sintáticos é a verificação da existência de uma classe que se comporte como um *template SQL* e possa ser utilizada como tradutor, auxiliando o processamento final da pergunta em linguagem natural. Embora essa abordagem não seja um requisito para o processamento específico, possibilitará a reutilização do *template* em novos padrões que utilizem modelos de consulta semelhantes. Nessa abordagem, um *template SQL* terá a responsabilidade de produzir uma consulta *SQL* a partir de um conjunto de objetos semânticos a serem determinados pelo padrão.

O segundo passo é a definição das características do padrão. Essas características são determinadas pelo retorno dos métodos que um padrão precisa implementar como resultado da especialização da classe abstrata AbstractQueryPattern.

A última etapa é a implementação do método principal do processamento específico: `execute()`. Esse método será chamado por uma instância da classe NLInterface quando a mesma reconhecer que as características de um determinado padrão são suficientes para interpretar a consulta formulada por um usuário.

5.1.1 Implementação do *template SQL*

O *template* que implementaremos, classe SQuery, servirá para duas classes de perguntas do usuário: navegação simples, sem hierarquias, em uma dimensão; seleção simples de fatos a partir de restrições em atributos de uma dimensão ou nos fatos propriamente ditos. O diagrama a seguir mostra a nossa classe que servirá como *template*, e suas dependências.

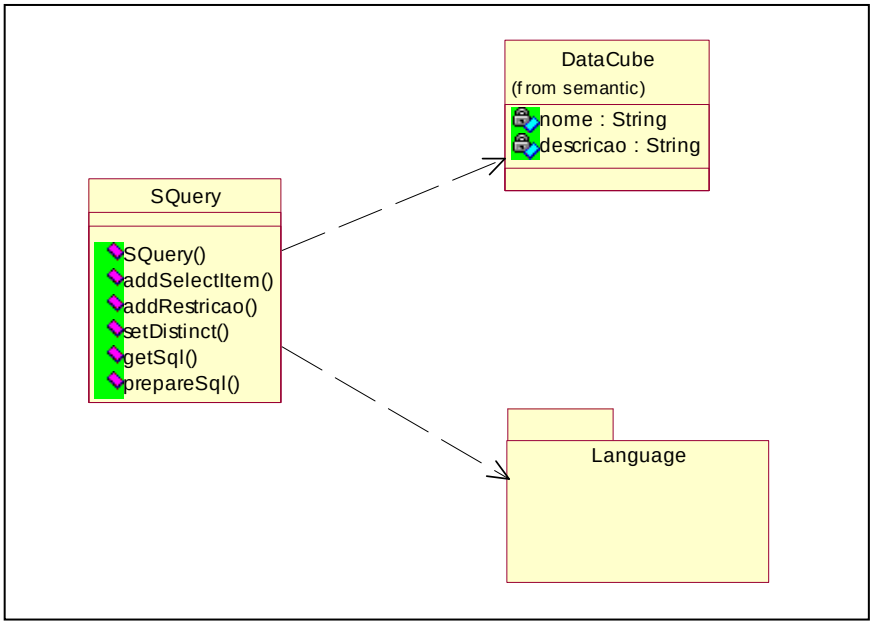


Figura 5.1: Diagrama de classe para o *template SQL* a ser utilizado no processamento específico

No diagrama da figura 5.1 podemos observar as dependências do *template* com o modelo de informação semântico e com o pacote de objetos semânticos. A primeira dependência reafirma a principal característica da arquitetura abstrata dessa interface: apoiada por metadados. Contudo, a segunda dependência não possui relação com a arquitetura, pois representa nossa solução particular para o processamento de linguagem natural. A seguir, detalhamos a interface da classe SQuery, que será utilizada quando implementarmos um padrão concreto.

1. `addSelectItem(SSemanticAtribute atributo)` - adiciona um objeto semântico do tipo SSemanticAtribute como elemento da cláusula *select*;

2. `addRestricao(SRestricao restricao)`- adiciona um objeto semântico do tipo SRestricao como condição da cláusula *where*;

3. `setDistinct(boolean distinct)`- indica a presença da restrição *distinct* na consulta SQL;

4. `prepareSql()` - monta a consulta SQL a partir de seus componentes. Deve ser chamado sempre antes de getSql();

5. `getSql()` - retorna a string SQL resultante do processamento realizado.

5.1.2 Definição das características de um padrão sintático

As características de um padrão sintático indicam os tipos de objetos semânticos que devem existir ou não em uma consulta para que seu processamento específico possa ser utilizado. As operações que seguem são utilizadas por instâncias da classe NLInterface a fim de comparar um padrão sintático com a estrutura resultante da primeira etapa do processamento de linguagem natural e, com isso, resolver possíveis ambigüidades entre padrões sintáticos. Caso as operação não sejam suficientes, NLInterface pode ser auxiliada pela operação `matchPattern()`definida em *AbstractPattern*, que deve oferecer um mecanismo mais específico para comparação do padrão sintático com a pergunta original. Uma implementação do `matchPattern()` poderia, por exemplo, procurar por determinados padrões de caracteres na pergunta original.

1. `hasRestrictionsOfDimensionObjects()` - indica a presença de restrições para atributos de dimensão;

2. `hasRestrictionsOfFact()` - indica a presença de restrições para fatos;

3. `hasSelectionOfFact()` - indica a existência de fatos;

4. `hasSelectionOfDimensionObjects()` - indica a existência de atributos de dimensão;

5. `hasTimeConstraints()` - indica a presença de restrições de tempo.

As operações descritas anteriormente devem retornar valores que se encontram no seguinte conjunto:

6. `AbstractQueryPattern.YES` - Existe a presença de pelo menos um objeto;

7. `AbstractQueryPattern.No` - Não existe a presença de objetos;

8. `AbstractQueryPattern.MAYBE` - A existência de objetos é opcional.

5.1.3 Implementação do processamento específico

A última etapa necessária à criação de padrões sintáticos é a implementação do método `execute()` definido em *AbstractPattern*. Esse método é chamado por instâncias da classe NLInterface após o reconhecimento do melhor padrão a ser utilizado. As principais responsabilidade do método `execute()` são: a) resolver as ambigüidades restantes, validando os objetos semânticos que deverão ser utilizados na consulta; b) produzir uma consulta na linguagem SQL a partir de objetos semânticos válidos. No nosso caso particular, delegamos a última responsabilidade para a classe SQuery. Com o objetivo de prover resultados tangíveis para o projeto e implementação propostos, especificamos dois padrões sintáticos: DimensionPattern e StarPattern. O primeiro é destinado a consultas simples em dimensões. O segundo pode ser utilizado para a seleção simples de fatos. Esses padrões não utilizam nenhum conhecimento específico de uma aplicação particular e apoiam-se, simplesmente, nos modelos de informação implementados. Os exemplos que seguem são resultados do processamento da interface através do uso desses dois padrões.

Qual a descricao, a categoria e subcategoria de cada produto que tenha a marca igual a 'Nestle' ?

```
SELECT
     PRODUTO.DESCRICAO, PRODUTO.CATEGORIA,
     PRODUTO.SUBCATEGORIA, PRODUTO.MARCA
FROM
     OLAP.PRODUTO
WHERE
     PRODUTO.MARCA='Nestle'
```

Qual o total de vendas por estado, nome da loja, categoria e subcategoria para o ano igual a '1994' onde estado seja diferente de 'Sergipe' ?

```
SELECT
     TEMPO.ANO, LOJA.ESTADO, LOJA.NOME,
     PRODUTO.CATEGORIA, PRODUTO.SUBCATEGORIA,
     SUM(VENDAS.QUANTIDADE)
FROM
     OLAP.TEMPO, OLAP.LOJA, OLAP.PRODUTO, OLAP.VENDAS
WHERE
     TEMPO.ANO='1994' AND
     LOJA.ESTADO <> 'Sergipe' AND
     TEMPO.TEMPO_KEY=VENDAS.TEMPO_KEY AND
     LOJA.LOJA_KEY=VENDAS.LOJA_KEY AND
     PRODUTO.PROD_KEY=VENDAS.PROD_KEY
GROUP BY
     TEMPO.ANO, LOJA.ESTADO, LOJA.NOME,
     PRODUTO.CATEGORIA, PRODUTO.SUBCATEGORIA
```

Qual o total de vendas para produtos de marca igual a 'Nestle' no ano igual a '1994' quando o total de vendas é superior a '30000' ?

```
SELECT
     PRODUTO.MARCA, TEMPO.ANO, SUM(VENDAS.QUANTIDADE)
FROM
     OLAP.PRODUTO, OLAP.TEMPO, OLAP.VENDAS
WHERE
     PRODUTO.MARCA='Nestle' AND
     TEMPO.ANO='1994' AND
     PRODUTO.PROD_KEY=VENDAS.PROD_KEY AND
     TEMPO.TEMPO_KEY=VENDAS.TEMPO_KEY
GROUP BY
     PRODUTO.MARCA, TEMPO.ANO
HAVING
     SUM(VENDAS.QUANTIDADE)>'30000'
```

Os resultados obtidos acima dão uma idéia geral do que pode ser alcançado com essa arquitetura. Pode-se observar que existe grande flexibilidade sintática para a seleção e restrição de atributos. É fácil verificar que para outros domínios de aplicações, desde que apresentem um esquema dimensional, os padrões demostrados acima poderiam ser totalmente reutilizados. Se, por exemplo, estivéssemos trabalhando com uma aplicação da área de saúde que informasse o número de casos de doenças em relação a variáveis como localidade, tempo, sexo, cor, faixa etária e muitas outras, poderíamos formular perguntas do tipo:

Qual o número de casos para a doença igual a 'AIDS' , em pessoas do sexo igual a 'masculino', na região igual a 'Nordeste' para uma data superior a '12/08/1994' abrangendo indivíduos com idade inferior a '24' anos ?

A partir desses resultados, podemos afirmar que se os padrões sintáticos implementados forem baseados, exclusivamente, nos conceitos do modelo multidimensional: fatos e objetos de dimensão, a Interface resultante mostrar-se-á transportável para diferentes aplicações. A sintaxe utilizada para as condições envolvendo atributos e fatos fazem parte de um conjunto de restrições da implementação presente e não deve ser vista como uma limitação da arquitetura.

5.2 Comparação com Interfaces semelhantes

Nesta seção faremos comparações entre nossa abordagem e outras interfaces que apresentam arquiteturas semelhantes. Os sistemas PRE [Epstein, 1985] e gNarLI [Shankar & Yung, 2000] utilizam variações da arquitetura de comparação de padrões (ver seção 2.3.1) para abordar o problema de processamento de linguagem natural. No capítulo 2, foram apresentadas as principais características dessas interfaces.

A grande vantagem da nossa abordagem em relação ao sistema PRE é a camada conceitual apresentada ao usuário. A utilização do modelo multidimensional permite que usuários casuais formulem consultas sem a necessidade de conhecer a estrutura do esquema lógico de dados da aplicação. No PRE, o usuário é obrigado a tomar conhecimento das diversas associações existentes no esquema do banco de dados a fim de poder especificar sua consulta com exatidão. Embora ambas as Interface exijam que os dados de uma aplicação estejam em um esquema lógico particular, o esquema relacional adotado por nossa arquitetura, por ser um padrão de projeto bastante conhecido e por possuir um mapeamento direto para o modelo multidimensional, supera em muito o utilizado na interface PRE.

A interface gNarLI apresenta-se como uma interface de rápida configuração. O emprego da abordagem de comparação de padrões baseada em expressões regulares e regras provê uma grande flexibilidade sintática para perguntas em linguagem natural. Contudo, sua superficialidade no tratamento de metadados impede que a mesma possa trabalhar com esquemas que possuam um número relativamente grande de junções ou que apresente junções

formadas a partir de múltiplos atributos, característica suportada e demonstrada por nosso projeto e implementação.

A nossa abordagem é a única que pode comprovar uma transportabilidade entre aplicações. Os padrões de consulta utilizados pelo PRE não deixam claro a sua utilização para diferentes domínios. A interface gNarLI precisa especificar novos padrões de consulta para cada nova aplicação a ser considerada.

5.3 Conclusões

No capítulo 4 e 5 especificamos uma proposta de projeto e implementação para a arquitetura de interface apresentada nesta dissertação. O sistema desenvolvido, ainda que imaturo para ser utilizado ou testado por usuários casuais, serviu para validar a arquitetura e prover uma infra-estrutura para futuras implementações. Os resultados obtidos através da implementação de dois padrões sintáticos foram suficientes para demonstrar dois propósitos iniciais dessa solução: ser transportável entre aplicações e poder ser estendida facilmente por programadores que não estejam familiarizados com a área de processamento de linguagem natural. A propriedade *habitability* não pôde ser demonstrada. Para sua verificação, seria necessário um conjunto suficientemente completo de padrões para testes com usuários casuais e não-especialistas.

Capítulo 6

Conclusões e Trabalhos Futuros

Os atuais Sistemas de Suporte à Decisão, em especial *Data Warehouses*, surgiram da necessidade de se obter informações integradas e relevantes em tempo hábil para auxiliar analistas e executivos na atividade de tomada de decisões. Até pouco tempo, a alta gerência representava o único usuário do ambiente de apoio à decisão corporativo, pois a tarefa de difundir a informação por toda uma empresa encontrava obstáculos tecnológicos e financeiros que não poderiam ou não possuíam justificativa para serem ultrapassados. A conseqüência direta dessa prática é a retenção de informação nas mãos de uma minoria privilegiada, diminuindo, dessa forma, as chances de um maior sucesso corporativo e individual.

A grande popularidade e a disseminação do uso da Internet veio acompanhada pelo desenvolvimento de tecnologias que tornaram possível o surgimento de uma nova categoria de SSD: os Sistemas de Suporte à Decisão baseados na *Web*. Os novos SSDs são definidos como sistemas que fornecem informações e ferramentas de apoio à decisão através do uso de um *browser* para *Web* [Power, 1998]. Esse novo paradigma, que se aproveita da infra-estrutura da *Web*, traz inúmeras vantagens em relação à tradicional arquitetura clientes/servidor. Dentre as mais importantes, podemos citar: a possibilidade de atender a um maior número de usuários, redução de custos na aquisição de software e instalação. Em resumo, SSDs baseados na *Web* reduzem barreiras tecnológicas e tornam mais fácil e menos onerosa a disseminação de informações por toda uma empresa.

Além de ser de grande valia para as empresas, essa nova categoria de SSD permite a exploração global de informações em outras áreas que não estão diretamente ligadas ao ramo empresarial. Como exemplo, podemos destacar sistemas de informações geográficas, hospitalares, turismo e saúde em geral. Logo, surgem novas classes de usuários não relacionados com o ambiente corporativo.

Todavia, a continuidade do desenvolvimento de ferramentas de consulta que se baseiam no mesmo paradigma de utilização das ferramentas para a arquitetura cliente/servidor, voltadas principalmente para analistas e executivos, impede que os novos usuários dessa categoria de sistema possam acessar informações relevantes a suas atividades. Esses novos usuários, muitas vezes ocasionais, apresentam uma grande diversidade de perfil técnico e, em sua maioria, nunca tiveram contato direto com quaisquer interfaces de acesso a SSDs. Logo, apesar da alegação que as ferramentas de acesso a SSD são fáceis, intuitivas e apresentam os dados de forma natural para seus usuários, é notório a necessidade de um mínimo perfil técnico e treinamento para sua utilização.

Porém, o ambiente da Internet, onde esses novos usuários estão inseridos, não é benevolente com tempo despendido em treinamento e, dificilmente, um usuário casual gastará seu tempo explorando as funcionalidades de uma ferramenta a fim de conseguir alguma interação. Por esse motivo, faz-se necessário a resolução de problemas dessa natureza antes de avaliar os verdadeiros benefícios trazidos pela *Web* para o desenvolvimento de SSDs.

Nesta dissertação, apresentamos uma proposta de arquitetura de interface como solução para o problema de acesso a dados em SSDs baseados na *Web* por usuários não-especialistas e casuais. A arquitetura proposta é baseada em conceitos como: linguagem natural restrita, comparação de padrões sintáticos e metadados. A adoção dos conceitos mencionados anteriormente faz parte da solução para problemas como *habitability* e transportabilidade entre domínios, que apresentam-se como principais obstáculos à aceitação e disseminação das atuais interfaces em linguagem natural para banco de dados de propósito geral.

Definimos uma arquitetura abstrata para a solução e especificamos seus principais módulos. A solução apresentada pressupõe a existência de esquemas relacionais que estejam em conformidade com a disciplina Modelagem Dimensional. Mais precisamente, é necessário que o esquema relacional de uma aplicação seja o esquema Estrela ou *Snowflake* a fim de que a arquitetura atenda seus propósitos.

Com o objetivo de validar a arquitetura proposta e prover uma infra-estrutura para futuras implementações, apresentamos a especificação de um projeto e implementação para a arquitetura abstrata. Como resultado do processo de desenvolvimento, especificamos os

requisitos funcionais e não-funcionais do sistema a ser implementado, sua visão arquitetural e os diversos pacotes e diagramas de classes que modelam o sistema. Durante a explanação desses artefatos, detalhamos os modelos de informação técnico e semântico utilizados pelo sistema como mecanismo para auxiliar o processamento de linguagem natural e prover uma visão mais natural dos dados para o usuário final.

Por fim, implementamos dois padrões sintáticos com o objetivo de apresentar o mecanismo de extensão do projeto desenvolvido e, ao mesmo tempo, produzir resultados tangíveis que pudessem exemplificar os propósitos gerais da arquitetura. O mecanismo de extensão utilizado pelo sistema não pressupõe conhecimentos sobre o processamento de linguagem natural. Logo, poderá ser efetuado por qualquer programador. Ao analisarmos os resultados produzidos pelo uso dos padrões sintáticos, salientamos que os mesmos, por terem sido definidos a partir de objetos do modelo conceitual, contribuem para a transportabilidade de domínio e, portanto, poderiam ser utilizados por outras aplicações. Embora a arquitetura se apresente como solução para o problema da *habitability*, o mesmo não foi demonstrado. Para isso, seria necessário a implementação de um conjunto maior de padrões sintáticos e testes reais com usuários casuais.

6.1 Trabalhos Futuros

Esta dissertação deixa algumas questões em aberto que poderiam servir como ponto de partida para extensões do nosso trabalho. A primeira delas diz respeito aos verdadeiros benefícios trazidos pela arquitetura e implementação apresentadas. Logo, seria necessário uma implementação adicional de um conjunto suficientemente completo de padrões sintáticos que pudessem ser utilizados em testes por usuários finais a fim de comprovar questões como *habitability* e transportabilidade entre domínios. Essa investigação deverá ser precedida do desenvolvimento de uma interface para o usuário final.

A arquitetura apresentada possui uma restrição muito forte, que é a exigência do esquema Estrela ou esquema *Snowflake* como esquema lógico de dados para uma aplicação. O principal problema em relação a essa restrição é que ela limita as aplicações contempladas por uma interface que segue a arquitetura proposta. Um outro aspecto é que o desenvolvimento de um esquema dimensional a partir de um esquema ER não representa uma tarefa trivial. Uma solução para esse problema seria a construção automática ou, pelo menos, semi-automática de um esquema dimensional a partir de um esquema ER. Embora a

modelagem dimensional seja considerada por muitos como uma arte e não uma ciência, alguns trabalhos, a exemplo de [Golfarelli, *et al*., 1998], procuram provar o contrário.

[Golfarelli *et al*., 1998] apresenta um novo modelo conceitual para Data Warehouses, chamado *Dimension Fact model*, e propõe uma metodologia semi-automática para a construção de um esquema conceitual a partir de um esquema ER. O modelo conceitual proposto é independente do modelo lógico utilizado. Portanto, se o trabalho de [Golfarelli *et al*., 1998] for completado no sentido de prover uma metodologia automática para o mapeamento entre um esquema conceitual e um esquema dimensional, eliminaremos a principal restrição de nossa arquitetura.

O esquema de dados de um *Data Warehouse* pode ser analisado através de diferentes visões (figura 6.1): Técnica, Semântica e Negócio. Os modelos de metados apresentados nessa dissertação englobam, principalmente, as camadas Técnica e Semântica. Embora nosso modelo de metados semânticos apresente algumas caraterísticas sobre a camada de negócios, elas não são suficientes para garantir o sucesso na extração de informações em um ambiente corporativo [Lehmann & Jaszewski, 1999]. Logo, uma outra extensão possível ao nosso trabalho seria a definição de um modelo de metados de negócios como extensão do modelo de metados semânticos. Tal definição, que seria seguida por um mapeamento para o modelo de metados semânticos, permitiria uma interação mais natural e consistente para o usuário final.

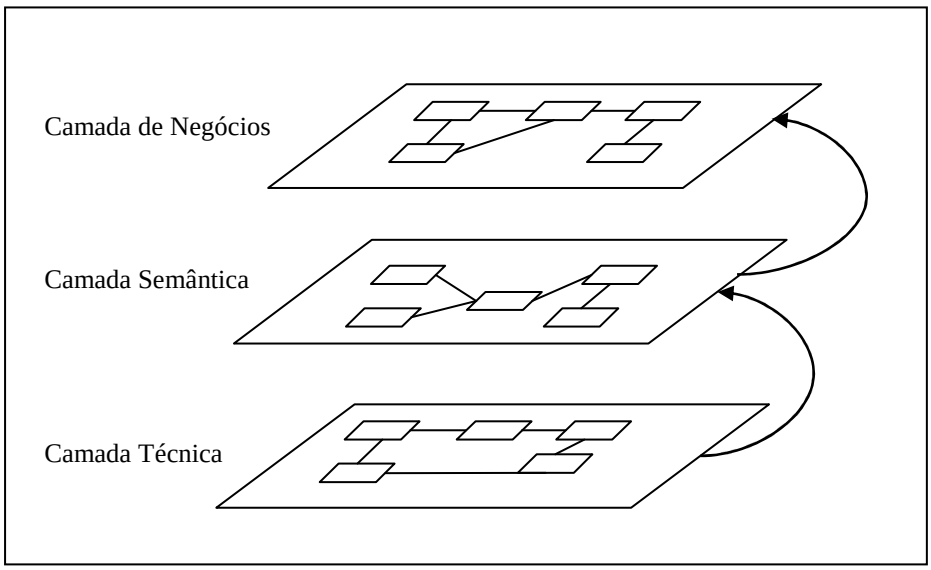


Figura 6.1: Diferentes visões de um esquema de dados

Bibliografia

[About.com, 1999] About.com, Inc. Article. *Talking to Your Database*. About.com, Inc., World Wide Web, *http://home.about.com/index.htm*, December, 1999.

[Adam & Gangopadhyay, 1997] ADAM, N. R., GANGOPADHYAY, A. *A Form-Based Natural Language Front-End to a CIM Database*. IEEE Transactions on Knowledge and Data Engineering, vol. 9, no. 2, pp. 238-250, April 1997.

[Allen, 1995] ALLEN, J. *Natural Language Understanding*. 2nd edition. CA: Benjamim/Cummings, 1995.

[Androutsopoulos *et al*., 1995] ANDROUTSOPOULOS, I., RITCHIE, G. D., THANISCH, P. *Natural Language Interfaces to Databases: An Introduction*. DAÍ Research Paper Nº. 709, Dep. of Artificial Intelligence, Edinburgh University, Scotland, UK.

[Barquin *et al*., 1997] BARQUIN, R. C. *et al*. *Planning and Designing The Data Warehouse*. New Jersey: Prentice Hall, 1997.

[Bhargava & Power, 2001] BHARGAVA, H., POWER, D. J. Power. *Decision Support Systems and Web Technologies: A Status Report*. Prepared for AMCIS 2001, Americas Conference on Information Systems, Boston, Massachusetts, August 3th - 5th, 2001, "Decision Support Systems" Mini Track. (URL *http://dssresources.com/papers/dsstrackoverview.pdf*).

[Booch *et al*., 1999] BOOCH, G. *et al*. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[Braker & Moore, 1996] STONE BRAKER, M., MOORE, D. *Object-Relational DBMSs: The Next Great Wave*. San Francisco: Morgan Kaufmann, 1996.

[Burg *et al.*, 1993] BURG, J. F. M., VAN DE REIT, R. P., CHANG, S. C. *A Data Dictionary as a Lexicon: An Application of Linguistics in Information Systems*. Proceedings of the 2nd International Conference on Information and Knowledge Management, pp. 114-123, 1993.

[Cantor, 1998] CANTOR, M. R. *Object Oriented Project Management with UML*. John Wiley & Sons, 1998.

[Chaudhuri & Dayal, 1997] CHAUDHURI, S., DAYAL, U. *An Overview of Data Warehousing and OLAP Technology*. SIGMOD Record, vol. 26, no. 1, pp. 65-74, March 1997.

[Church *et al.*, 1995] CHURCH, K. W., RAU, L. F. *Commercial Applications of Natural Language Processing*. Communications of the ACM, vol. 38, no. 11, pp. 71-79, November 1995.

[Coalition, 1998] Metadata Coalition. *Metadata Interchange Specification version 1.1*. World Wide Web, *http://www.mdcinfo.com*, August 1998.

[Coalition, 1999] Metadata Coalition. *Open Information Model version 1.0*. World Wide Web, *http://www.mdcinfo.com*, August 1999.

[Copestake & Jones, 1989] COPESTAKE, A., JONES, K. S. *Natural Language Interfaces to Databases*. Computer Laboratory, University of Cambridge, 1989.

[Dinter *et al*., 1998] DINTER, B. *et al*. *The OLAP Market: State of the Art and Research Issues*. Proceedings of the ACM 1st International Workshop on Data Warehousing and OLAP, pp. 22-27, Washington, United States, 1998.

[Elmasri & Navathe, 1994] ELMASRI, R., NAVATHE, S. B. *Fundamentals of Database Systems*. 2nd edition. CA: Benjamim/Cummings, 1994.

[Embley & Kribell, 1985] EMBLEY, D. W., KRIMBELL, R. E. A *Schema-Driven Natural Language Query Translator*. Proceedings of the 13th Annual Computer Science Conference, pp. 292-297, 1985.

[Epstein, 1985] EPSTEIN, S. S. *Transportable Natural Language Processing Through Simplicity*: The PRE System. ACM Transactions on Office Information Systems, vol. 3, no. 2, pp. 107-120, April 1985.

[Ferreira, 1993] FERREIRA, A. B. H. *Dicionário da Língua Portuguesa*. Rio de Janeiro: Nova Fronteira, 1993.

[Flanagan, 1997] FLANAGAN, D. *Java in a Nutshell*. 2nd edition. CA: O'Reilly, 1997.

[Fowler & Scott, 1999] FOWLER, M., SCOTT, K. *UML Distilled*: A Brief Guide to the Standard Object Modeling Language. 2nd edition. Addison Wesley Longman, 1999.

[Gamma, et al., 1994] GAMMA, E., HELM, R., JOHNSON, R., VLISSIDES, J., *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[Golfarelli, 1998] GOLFARELLI, M., MAIO, D., RIZZI, S. *Conceptual Design of Data Warehouses from E/R Schemes*. IEEE Proceedings of the Hawaii International Conference On Systems Sciences, January 1998.

[Gupta, 1997] GUPTA, V. R. *An Introduction to Data Warehousing*. System Services Corporation, August 1997. *http://www.system-services.com/dwintro.htm*.

[Hammer, *et al*., 1995] HAMMER J. *et al*. *The Stanford Data Warehousing Project*. Computer Science Departament, Stanford University, 1995.

[Inmon, 1997] INMON, W. H. *Como Construir o Data Warehouse*. Tradução da segunda edição. Rio de Janeiro: Campus, 1997.

[Jarke, *et al*., 1999] JARKE, M. *et al*. *Fundamentals of Data Warehouses*. New York: Springer-Verlag, 1999.

[Kimball, 1996] KIMBALL, R. *The Data Warehouse Toolkit*. New York: John Wiley & Sons, 1996.

[Kimball, 1996b] KIMBALL, R. *Aggregate Navigation With (Almost) No Metadata*. DBMS Data Warehouse Supplement, August 1996.

[Kimball, 1997] KIMBALL, R. *A Dimensional Modeling Manifesto*. DBMS and Internet Systems, July 1997.

[Kimball *et al*., 1998] KIMBALL, R. *et al*. *The Data Warehouse Lifecycle Toolkit*. New York: John Wiley & Sons, 1998.

[Lehmann & Jaszewski, 1999] LEHMANN, P., JASZEWSKI, J. *Business Terms as a Critical Success Factor for Data Warehousing*. Proceedings of the International Workshop on Design and Management of Data Warehouses, Heidelberg, Germany, 1999.

[Longman, 1990] Longman Dictionary of Contemporary English. England:Longman, 1990.

[Meng & Chu, 1999] MENG, F., CHU, W. W. *Database Query Formation from Natural Language using Semantic Modeling and Statistical Keyword Meaning Disambiguation*. Technical Report CSD-TR 990003, University of California, Los Angeles, 1999.

[Nicola & Infante, 1991] NICOLA, J., INFANTE, U. *Gramática Contemporânea da Língua Portuguesa*. São Paulo:Scipione, 1991.

[Ogden, 1985] OGDEN, W. C. *The Human Factors of Natural Language Query Systems*. Proceedings of the 13th Annual Computer Science Conference, pp. 174-175, 1985.

[Pendse, 2000] PENDSE, N. *The OLAP Report:What is OLAP ?. , http://www.olapreport.com/fasmi.htm.*, February 2000.

[Peterson, 1994] PETERSON, S. *Stars: A Pattern Language for Query Optimized Schema.* White Paper, Sequent Computer Systems Inc., *http://c2.com/ppr/stars.html*, 1994.

[Pilot Software] Pilot Software, Inc. White Paper. *Web-Based Decision Support: Scaling to Support Very Large User Communities*. Pilot Software, Inc., World Wide Web, *http://www.pilotsw.com*.

[Poe et al., 1988] POE, V. *et al*. *Building a Data Warehouse for Decision Support*. Prentice Hall PTR, 1998.

[Power, 1997] POWER, D. J. *Justifying a Data Warehouse Project*. The On-Line Executive Journal for Data-Intensive Decision Support, vol. 2, no. 5, February 1998.

[Power, 1998] POWER, D. J. *Web Based Desicion Support Systems*. DSstar, The On-Line Executive Journal for Data-Intensive Decision Support, vol. 2, nos. 33-34, August 1998.

[Power, 1999] POWER, D. J. *A Brief History of Decision Support Systems*. DSSResources.COM, World Wide Web, *http://DSSResources.com/history/dsshistory.html*, 1999.

[Power, 2000a] POWER, D. J. *Web-Based and Model-Driven Decision Support Systems: Concepts and Issues*. Prepared for AMCIS 2000, Americas Conference on Information Systems, Long Beach, California, August 10th - 13th, 2000, "Model-Driven and Web-Based Decision Support Systems" Mini Track. (URL http://dssresources.com/papers/amcis/TT08overview.pdf).

[Power, 2000b] POWER, D. J. *Supporting Decision-Makers: An Expanded Framework*. DSSREsources.COM, World Wide Web, *http://dssresources.com/papers/supportingdm/sld001.htm*, 2000.

[Raden, 1996] RADEN, N. *Modeling the Data Warehouse*. World Wide Web, *http://user.aol.com/nraden/iw0196_1.htm*, 1996.

[Ramakrishnan, 1998] RAMAKRISHNAN, R. *Database Management Systems*. USA: WBC/McGraw-Hill, 1998.

[Reis *et al*., 1997] REIS, P., MATIAS, J., MAMEDE, N. *Edite - A Natural Language Interface to Databases: a New Dimension for an Old Approach*. Proceeding of the Fourth International Conference on Information and Communication Technology in Tourism, ENTER'97, Edinburgh, Escócia. Springer-Verlag, 1997.

[Rudloff, 1996] RUDLOFF, D. *Terminological Reasoning and Conceptual Modeling for Datawarehouse*. KRDB-96, Budapest.

[Rumbaugh *et al*., 1991] RUMBAUGH, J. *et al*. *Object Oriented Modeling and Design.* Prentice Hall, 1991.

[Russel & Norvig, 1998] RUSSEL, S. J., NORVIG, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, December 1998.

[Sachdeva, 1999] SACHDEVA, S. *Metadata for Data Warehouse*. Sybase, Inc. World Wide Web, *http://www.sybase.com/services/dwpractice/meta.html*, 1999.

[Santos, 2000] SANTOS, G. F. *Realização de Consultas Visuais a Banco de Dados Temporais*. Dissertação de Mestrado, Departamento de Sistemas e Computação, UFPB - Campus II, 2000.

[Savadovsky, 1988] SAVADOVSKY, P. *Introdução ao Projeto de Interfaces em Linguagem Natural*. São Paulo: SID Informática, 1988.

[Sethi, 1986] SETHI, V. *Natural Language Inferfaces to Databases: MIS Impact, and a Survey of Their Use and Importance*. Proceedings of the 22nd Annual Computer Personnel Research Conference, pp. 12-26, 1986.

[Shankar & Yung, 2000] SHANKAR, A., YUNG, W. *gNarLI: A Practical Approach to Natural Language Interfaces to Databases.* World Wide Web, *http://www.people.fas.harvard.edu/~shankar2/stuff/gnarli.html*, 2000.

[Sherman, 1997] SHERMAN, R. P. *Metadata: The Missing Link - Business Information Directories Catalog Decision-Support Information Throughout the Enterprise*. DBMS and Internet Systems, August 1997.

[Shin, 1990] SHIN, D. G. *Semantics Modeling Issues for Processing Natural Language Database Queries*. Proceedings of the 1990 ACM Conference on Cooperation, pp. 8-14, 1990.

[Silva *et al*., 1997] SILVA, S. L. F., SCHIEL, U., CATARCI, T. *Visua Query Operators for Temporal Databases.* Proceedings of the 4th International Workshop on Temporal Representation and Reasoning, Florida, USA, pp. 46-53, May, 1997.

[Stöhr, *et al*., 1999] STÖHR, T., MÜLLER, R., RAHM, E. *An Integrative and Uniform Model for Metadata Management in Data Warehousing Environments*. Proceedings of the International Workshop on Design and Management of Data Warehouses, Heidelberg, Germany, 1999.

[Srivastava & Chen, 1999] SRIVASTAVA, J., CHEN P. *Warehouse Creation: A Potential Roadblock to Data Warehousing*. IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 1, pp. 118-126, February 1999.

[Tanenbaum, 1997] TANENBAUM, A. S., *Computer Networks. Prentice-Hall*. 1997

[Tuck, 1997] TUCK, T. C. *Natural Language Processing: Understanding what you say*. Surveys and Presentations in Information Systems Engineering. London: Imperial College of Science Technology and Medicine, *http://www.dse.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/ctt/*, 1997.

[Ullman & Widom, 1997] ULLMAN, J. D., WIDOM, J. *A First Course in Database Systems*. New Jersey: Prentice Hall, 1997.

[van der Lans, 1993] VAN DER LANS, R. F. *Introduction to SQL*. Addison-Wesley, 1993.

[Watson, 1999] WATSON, M. NLBean version 4: *A Natural Language Interface for Databases*. World Wide Web, *http://www.markwatson.com/opensource/nlbeandoc.htm*, 1999.

[Wiederhold, 1991] WIEDERHOLD, G. *Mediators in the Architecture of Future Information Systems*. Stantford University, September 1991.

[Wu & Buchmann, 1997] WU, M-C., BUCHMANN, A. P. *Research Issues in Data Warehousing*. BTW'97 (German Database Conference), Uml, pp. 61-82, 1997.

Apêndice A

Metalinguagem de descrição dos padrões sintáticos

As interfaces baseadas em padrões sintáticos não são regidas por gramáticas a nível sintático ou semàntico. Logo, o processo de *parser*, ou identificação de estruturas válidas, é baseado na observação de palavras chaves que aglutinam-se para formar um todo semântico. Desse modo, faz-se necessário a apresentação de uma metalinguagem para descrever a estrutura de sentenças válidas e guiar o usuário na formulação de sentenças passíveis de processamento.

Nesse apêndice, serão apresentados elementos que compõem a metalinguagem utilizada para descrever os padrões sintáticos, como também exemplos de sentenças válidas para composições desses elementos. A metalinguagem apresentada não tem por objetivo ser suficientemente completa para descrever quaisquer padrões que venham a ser definidos. Seu papel nessa dissertação é elucidar, formalmente, o poder de expressão dos padrões implementados. A descrição desses padrão através da metalinguagem, assim como exemplos de perguntas em linguagem natural correspondentes são encontrados no capítulo 4. A metalinguagem utilizada é composta por três grupos: Elementos Básicos, Elementos Derivados e Construções Adicionais. O primeiro engloba os objetos semânticos primitivos reconhecidos pela interface (ver pacote Language no capítulo 4); o segundo representa estruturas sintáticas formadas pela composição de elementos primitivos, objetos semânticos derivados; as Construções Adicionais auxiliam a formulação de sentenças através de elementos que indicam repetições e construções opcionais.

Elementos Básicos

<Nome de atributo>	Representa agrupamentos de palavras que identificam um atributo de uma dimensão no esquema conceitual multidimensional da aplicação. Mais precisamente,

	correspondem aos nomes e sinônimos atribuídos a um atributo de dimensão no modelo de metados semântico.
<Nome de medida>	Representa agrupamentos de palavras que identificam as medidas ou fatos no esquema conceitual multidimesional da aplicação.
<Operador>	Representa agrupamentos de palavras que identificam operadores binários utilizados em restrições de atributos de dimensão e fatos.
<Constante>	Representa constantes utilizadas em restrições de atributos de dimensão e fatos. Elementos dessa categoria devem ser delimitados por apóstrofos. Exemplos: 'Janeiro', '1999', 'Região Norte'.
<;>	Representa quaiquer intercalação de símbolos que não são identificados como os elementos básicos descritos anteriormente.

Elementos Compostos

<Restrição de atributo>	Representa a combinação dos seguintes elementos básicos: *<Nome de atributo>*, *<Operador>* e *<Constante>*. Descreve uma restrição em um atributo de dimensão. Estrutura da composição: *<Nome de atributo> <;> <Operador> <;> <Constante>*
<Restrição de medida>	Representa a combinação dos seguintes elementos básicos: *<Nome de medida>*, *<Operador>* e *<Constante>*. Descreve uma restrição em uma medida ou fato. Estrutra da composição: *<Nome de medida> <;> <Operador> <;> <Constante>*

Contruções Adicionais

||	Utilizado para indicar *ou* lógico.
( )	Utilizado para agregar elementos da metalinguagem
[<X>]	Utilizado para indicar items opcionais. <X> representa algum elemento básico ou composto.
$<Y>	Utilizado para indicar repetições de um item. <Y> representa qualquer elemento da metalinguagem.

Exemplos de construções

C.1) *<;>* <*Nome de atributo*> <;> [$ (<*Nome de atributo*><*;*>)]

A construção C.1 mostra que sentenças válidas iniciam por quaisquer expressões e são seguidas por um ou mais nomes de atributos de dimensão, intercalados por quaisquer expressões.

Considere que Nome da Loja , Nomes de Loja, Nome, Endereço (s) e Telefone (s) são expressões que identificam atributos de uma dimensão em um esquema conceitual multidimensional. Dessa forma, as seguintes sentenças são válidas para a construção C.1:

a) Mostre-me o Nome da Loja, o Endereço e o Telefone

b) Quais os Telefones que encontram-se no cadastro de Lojas

C.2) *<;>*<*Nome de atributo*> <,> [$(<*Restrição de atributo*>)]

A construção C.2 mostra que sentenças válidas iniciam por quaisquer expressões e são seguidas por um nome de atributo ou opeador que podem vir seguidos por uma ou mais constantes. As seguinte sentenças são válidas para a construção C.2:

a) Qual o Nome da Loja cujo Telefone é igual a '223-2345'

b) Qual o Endereço da Loja de Nome igual a 'SuperMarket' e Telefone igual a '298-3498'

Apêndice B

Processamento de Consulta em Linguagem Natural

Nesse apêndice serão apresentado os estágios de processamento pelos quais passa uma sentença em linguagem natural para que a interface descrita na dissertação produza uma expressão correspondente na linguagem *SQL*. Partimos do pressuposto que existem esquemas lógico e conceitual definidos nos modelos de metados suportados pela interface. A explanação a seguir representa uma descrição textual das interações entre classes e módulos do sistema.

L.1) Qual o total de vendas para produtos de marca igual a 'Nestlê no ano igual a '1994' , quando o total de vendas é superior a '30000'?

A classe reponsável por obter a pergunta em linguagem natural é NLInterface, que representa o coordenador para o processamento de linguagem. NLInterface instancia um objeto do tipo QueryExplorer, responsável por armazenar a pergunta original e as estruturas derivadas do seu processamento, e delega a primeira etapa o processamento para NLEngine. Nesse momento o objeto do tipo QueryExplorer passa a ser controlado por uma instância da classe NLEngine.

NLEngine executa o pré-processamento de pergunta original de modo que os módulos subsequentes recebem uma estrutura conhecida. Os principais processos dessa fase são: remoção de símbolos irrelevantes e separação de símbolos. O primeiro é responsável por desconsiderar caracteres como sinais de pontuação: ponto, vírgula, ponto e vírgula. O segundo tem a responsabilidade de separar símbolos que podem representar objetos semânticos. Em C.1 temos a pergunta original depois da fase de pré-processamento. Pode-se observar a remoção de caracteres irrelevantes: vírgulas e espaços em branco; e a separação de símbolos: '30000'? em '30000' ?.

C.1) *Qual o total de vendas para produtos de marca igual a 'Nestlê' no ano igual a '1994' quando o total de vendas é superior a '30000' ?*

Após o pré-processamento, a pergunta é passada para uma instância da classe NLParser. Essa classe é responsável por gerar uma estrutura que armazena os objetos semânticos primitivos contidos na pergunta do usuário. Esse processamento é auxiliado por classes do tipo IFToken e pelo léxico da interface. A estrutura gerada pelo processamento da classe NLParser identifica as seguintes relações:

Qual	:objeto do tipo string
o	:objeto do tipo string
total de vendas	:objeto semântico do tipo SFactAtribute
para	:objeto do tipo string
produtos	:objeto do tipo string
de	:objeto do tipo string
marca	:objeto semântico do tipo SDObject
igual a	:objeto semântico do tipo SOperador
'Nestlê'	:objeto semântico do tipo SConstant
no	:objeto do tipo string
ano	:objeto semântico do tipo SDObject
igual a	:objeto semântico do tipo SOperador
'1994'	:objeto semântico do tipo SConstant
quando	:objeto do tipo string
o	:objeto do tipo string
total de vendas	:objeto semântico do tipo SFactAtribute

é superior a :objeto semântico do tipo SOperador

'30000' :objeto semântico do tipo SConstant

A estrutura contendo as relações citadas anteriormente é passada para uma instância da classe NLCondition, responsável por gerar uma nova estrutura contendo relações que identificam objetos semânticos derivados:

Qual :objeto do tipo string

o :objeto do tipo string

total de vendas :objeto semântico do tipo SFactAtribute

para :objeto do tipo string

produtos :objeto do tipo string

de :objeto do tipo string

marca igual a 'Nestlê' :objeto semântico do tipo SRestricao

no :objeto do tipo string

ano igual a '1994' :objeto semântico do tipo SRestricao

quando :objeto do tipo string

o :objeto do tipo string

total de vendas é superior a '30000' :objeto semântico do tipo SRestricao

As estrutura geradas por instâncias das classes NLParser e NLCondition são armazenadas na instância da classe QueryExplorer que é repassada para a classe NLInterface. De posse da instância do tipo QueryExplorer, NLInterface é capaz de identificar as características da pergunta original e compará-las com as características dos padrões sintáticos implementados. Desse modo, NLInterface pode eleger o melhor padrão sintático para traduzir a pergunta em linguagem natural. Após a escolha do padrão, que no nosso exemplo é o StarPattern, NLInterface invoca o método execute do padrão escolhido passando

como parâmetro a instância da classe QueryExplorer. Com isso, o processamento específico do padrão sintático poderá identificar os relacionamentos existentes entre objetos semânticos e elementos da pergunta original.

O processamento específico dos padrões sintáticos são implementações particulares e não possuem dependências com outros padrões. O StarPattern possui um template que representa um *star join*. Logo, através da estrutura de objetos semânticos e do conhecimento do modelos de metadados técnico e semântico, o processamento específico do padrão StarPattern identifica atributos de dimensão, fatos, agregações, junções diretas e indiretas (*Snowflake*), restrições em atributos de dimensão e restrições em fatos ou em agregações. Como resultado final, temos uma expressão na linguagem SQL que representa a pergunta em linguagem natural em função do esquema conceitual multidimensional da aplicação:

```
SELECT
     PRODUTO.MARCA, TEMPO.ANO, SUM(VENDAS.QUANTIDADE)
FROM
     OLAP.PRODUTO, OLAP.TEMPO, OLAP.VENDAS
WHERE
     PRODUTO.MARCA='Nestle' AND
     TEMPO.ANO='1994' AND
     PRODUTO.PROD_KEY=VENDAS.PROD_KEY AND
     TEMPO.TEMPO_KEY=VENDAS.TEMPO_KEY
GROUP BY
     PRODUTO.MARCA, TEMPO.ANO
HAVING
     SUM(VENDAS.QUANTIDADE)>'30000'
```

Glossário

SSD	Sistema de Suporte à Decisão
CASE	Computer-Aided Software Engineering
DW	Data Warehouse
ER	Entidade-Relacionamento
ILNBD	Interface em Linguagem Natural para Banco de Dados
MDIS	Metadata Interchange Specification
MDX	Multidimensional Expression
MOLAP	Multidimensional OLAP
OIM	Open Information Model
OLAP	On-Line Analytical Processing
OLTP	On-Line Transaction Processing
QBE	Query-By-Example
ROLAP	Relational OLAP
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
SVC	Sistema Visual de Consulta
UML	Unified Modeling Language